Methods for control strategy identification in Boolean networks

Dissertation
zur Erlangung des Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)

am Fachbereich Mathematik und Informatik
der Freien Universität Berlin

vorgelegt von

Laura Cifuentes Fontanals

Berlin, 2022

1. Gutachter:
Prof. Dr. Heike Siebert
Freie Universität Berlin

2. Gutachter:
Prof. Dr. Claudine Chaouiya
Aix-Marseille University

Tag der Disputation:
15. Dezember 2022

Contents

Chapter 1

Introduction

Understanding control mechanisms in biological processes plays a crucial role in the development of potential applications in the fields of bioengineering and medicine. The identification of drug targets for disease treatments or the reprogramming of cancer cells to induce apoptosis are some examples of its practical applications [FBR$^+$15]. Exhaustive experimental approaches are too costly and time-consuming to be used for identification of potential control targets. Mathematical modeling helps to address this problem by providing a formal framework to predict potential successful candidate interventions in silico [PAM22].

Boolean networks were introduced by Kauffman [Kau91, GK73] to study the modeling of gene regulatory networks. The modeling of biological systems is often hindered by the lack of information about reaction parameters. Boolean modeling stands out among other modeling frameworks for its ability to capture the qualitative behavior of the system using coarse representations of the interactions between the different components. These components are represented by discrete variables, admitting only two activity levels, 0 and 1, representing for instance the activity of a gene in a gene regulatory network or if the concentration of a protein or compound is above or below a certain threshold. The activation and inhibition relations between the different components are described by logical regulatory functions. Boolean models are able to capture the relevant behaviors of many gene regulatory networks and signaling networks [FNCT06, ZSY$^+$08] and have proven useful to study how different perturbations might affect their dynamics [FBR$^+$15, CTF$^+$10]. They have also been widely used to predict or design therapeutic treatments [MBT$^+$22, PAM22].

Multiple dynamics can be defined from the same Boolean network depending on the type of update chosen for the regulatory functions. The synchronous dynamics updates simultaneously all components, whereas the asynchronous dynamics aims at capturing more realistically the different (possibly unknown) time scales that might coexist in a biological system, by updating only one component at each time step. The long-term dynamics of

the modeled system is captured by the attractors. Attractors are sets of states where the system stabilizes. Attractors that consist of only one state (steady states) can be identified with different cell types or cell fates in biological systems, whereas attractors consisting of more than one state (complex attractors) might be associated with different oscillating processes.

Despite the apparent simplicity of Boolean networks, the brute-force exploration of the state space is usually not feasible since it grows exponentially with the size of the network. Consequently, problems such as attractor identification or controllability are non-trivial [Aku18]. Attempts to relate dynamical properties from patterns in the interaction network, such as Thomas conjectures [Tho81], have produced results that are too general to be useful when applied to specific biological networks. Other approaches have studied the prime-implicant hypergraph [KBS15] or extended network [ZA13], a network with two nodes for each component that encodes the regulatory functions in terms of implications between these nodes, to deduce dynamical properties of the system. In recent years, multiple methods have been developed aiming at a more efficient attractor identification [DT11, BAFRM17, RZG$^+$21].

Control of dynamical systems has been another important research field in systems biology in the last years. Many approaches focus on the structure and topology of the network, for example by considering feedback loops [ZYA17] or stable motifs [ZA15], and several studies discuss the complexity and characteristics of such problems [KPC13, LSB11, Aku18]. Other works analyze the characteristics of the control sets by studying its average size [BD21] or the properties of nodes that are part of them [WIS$^+$22].

Moreover, control of biological systems encompasses a great variety of scenarios and goals. Numerous approaches focus on leading the system to a specific attractor of interest from a particular initial state (source-target control) [MSH$^+$19, BPSS21] or from any possible initial state (full-network control) [ZA15]. This type of control problem is known as attractor control. We can find an example of such control problem in the context of a cell-fate decision network with two attractors representing two possible cell fates: survival or apoptosis. Identifying control strategies to induce the apoptotic attractor in cancer cells might help the development of potential treatments that would lead the system towards the elimination of the unhealthy cells. In other scenarios, it is sufficient to focus on a set of relevant components defining the state of the system rather than specific attractors. This might be the case for biological systems with multiple attractors that are grouped in different phenotypes. Biological phenotypes are often defined by the values of a set of observable components or biomarkers, which capture relevant characteristics of biological processes. Many approaches have been developed for this type of control, known as target control, which targets a set of relevant variables instead of a specific attractor [SKK10, YGTZA18, BD19].

Another relevant aspect when defining a control problem is the type of perturbations considered. A usual approach is the use of control interventions that fix the state of a component to a certain value (0 or 1). This type of intervention, called node intervention or node control, can represent for instance the knockout or sustained activation of a gene in a gene-regulatory network. A great variety of approaches have been developed to deal with node control [SPP19, ZA15, YGTZA18]. One of the potential limitations of node control is that fixing the value of a specific node directly affects all the components regulated by this node. In the context of biological systems, the regulated node might play a potentially crucial role in other processes that should not be disrupted. In such cases, it is useful to consider a more focused perturbation, for instance an edge intervention. An edge intervention targets a specific interaction between two components, leaving the rest of the nodes and interactions unaltered. Edge perturbations have been shown to play a relevant role in the understanding of control mechanisms in biological processes [ZSL$^+$09, CZD$^+$11]. Consequently, many approaches for control strategy identification using edge control have been developed [MVCAL16, BD19, CA19]. Control interventions can also be classified depending on their duration. We talk about permanent perturbations when they are maintained indefinitely and temporary perturbations if they are released after a certain amount of time [SPP19]. Moreover, interventions can be applied all at once (one-step control) [SKK10] or at different time steps (sequential control) [MSH$^+$19].

The main goal of this thesis is the development of efficient and complete approaches for control strategy identification. To do so, we first focus on value percolation. At the core of many control approaches, it uses the propagation of fixed values through the network to determine whether a set of initial control interventions is sufficient to induce the target state [SKK10]. Approaches based on value percolation can be implemented efficiently [KSSV13] but they usually miss many control strategies. The first part of this work aims at increasing the number of identified control strategies while keeping the efficiency of value percolation. With this goal, we propose the use of trap spaces, subspaces of the state space that are closed with respect to the dynamics. By definition, trap spaces contain at least one attractor. In fact, minimal trap spaces are often good approximations of attractors in biological systems [KS15] and can be efficiently computed for relatively large networks [KBS15]. The main idea behind the method is to broaden the final target region in order to detect sets of interventions that, despite not percolating directly to the target, percolate to the so-called selected trap spaces. Selected trap spaces are trap spaces containing only attractors included in the target. This condition allows us to guarantee that, if the system percolates to a selected trap space, its long-term dynamics will be the desired one. This idea, control via trap spaces, lead us to an approach for control strategy identification, published in [CFTS20, CFTS22c], using both node and edge interventions. The implementation of the method can be found in [CFa]. In Chapter 3, we present this work and extend the idea of the selected trap spaces to extend the method to deal with transient control.

Despite providing an efficient and more complete approach than just direct percolation,

the approach via-trap-spaces is still not able to identify all possible control strategies for a given target. With the aim of a more exhaustive identification, recent works have dealt with attractor control using basins of attraction [SPP19, MSH$^+$19]. However, these methods are still limited to attractor control and lack the flexibility to adapt to more complex targets such as groups of attractors or attractor avoidance. The second part of this work deals with control focusing on exhaustivity and flexibility, aiming to develop a control strategy identification method that is able to obtain all the minimal control strategies for a given arbitrary target. Providing a complete solution set of minimal controls is a complex problem [Aku18] that might require the exploration of large regions of the state space. We deal with this type of exploration problems by using model checking techniques. Model checking is a formal verification method that allows us to determine if a given transition system satisfies a specific property. Model checking techniques have been successfully applied to analyze biological networks [CGR12] and also widely used in the context of Boolean networks, for instance in the verification of reachability properties [MAJTC14], basin computation [KHNS18] or attractor approximation [KS15]. One of the main advantages of model checking is the use of symbolic representation, which allows us to work with transition systems with a large amount of states, like the state transition graph. However, dealing with a wider and more complex control problem naturally implies higher computational costs since most of the usual reduction methods might not apply. Consequently, we also develop alternative and more tailored preprocessing methods to reduce the computational costs of the exhaustive exploration. These include, for instance, the reduction of the region of the state space that needs to be searched or the amount of control candidates to verify. This exhaustive approach based on model checking is published in [CFTS22b] and is presented Chapter 4, explaining in more detail some of the reduction methods developed for it. Its implementation can be found in [CFb].

We show the applicability of the developed approaches by analyzing a network modeling the epithelial-to-mesenchymal transition [SCP$^+$20]. We consider different control targets (attractor, target and phenotype avoidance) and use both node and edge interventions. We analyze the different control strategies obtained in each case and explore their relevance in the biological context. Additionally, we compare our approaches to current control methods for attractor and target control considering different dynamics in multiple Boolean networks. We show that the percolation and trap-space approaches are significantly faster than the rest of the methods, while the exhaustive approach can identify many control strategies missed by the other control methods.

The thesis is organized as follows. We start by giving a general introduction to Boolean networks and their dynamics in Chapter 2, recalling the concept of value percolation and deducing relevant dynamical properties that are used in the following chapters. We also formalize the effect that the node and edge interventions have in the controlled dynamics and define the concept of control strategy that is used throughout this work. In Chapter 3, we introduce our first method for control strategy identification, which is based on value

percolation and uses trap spaces to uncover new control strategies. First, we provide a theoretical explanation of our approach (Section 3.1) and its extension to also deal with transient control (Section 3.2). We then describe its implementation using Answer Set Programming, discussing the factors affecting the efficiency of the approach (Section 3.3). We finish by showing its applicability to a biological case study (Section 3.4). In Chapter 4 we introduce our second method for control strategy identification, which consists of an exhaustive approach based on model checking that is able to identify all the minimal control strategies for a given arbitrary target. We start this chapter by introducing an approach based on the completeness of the network (Section 4.1) followed by a brief introduction to model checking (Section 4.2). We then describe our exhaustive approach in Section 4.3 and present the details of its implementation and the reduction methods that can be used to diminish the computational time in Section 4.4.1. We finish with the continuation of the biological case study from the previous chapter (Section 4.5). Lastly, Chapter 5 explores more exhaustively the applicability of our approaches in a biological case study modeling the epithelial-to-mesenchymal transition (Section 5.1). We also perform a comparison of our methods to other existing approaches (Section 5.2).

Chapter 2

Boolean networks and control

In this chapter, we introduce the main concepts regarding Boolean networks and control that are used through this work. We start by defining a Boolean network and its dynamics, including attractors and trap spaces. We continue by recalling the concept of value percolation and deducing useful dynamical properties related to percolated subspaces. In the second part, we define the control interventions that are considered in this work and their effect on the dynamics of a Boolean network. The notation and definitions related to control interventions, controlled function and control strategy correspond to the ones introduced in [CFTS22c]. We finish the chapter by giving a rough overview of the different types of control and defining the control problem tackled in this work.

2.1 Boolean networks and dynamics

We define a *Boolean network* on n variables as a function $f = (f_1, \dots, f_n) \colon \mathbb{B}^n \to \mathbb{B}^n$, with $\mathbb{B} = \{0, 1\}$. The set of variables or components $\{0, \dots n\}$ is denoted by V. $\mathbb{B}^n$ is the *state space* of a Boolean function and every $x = (x_1, \dots, x_n) \in \mathbb{B}^n$ is a *state* of the state space. Given $x \in \mathbb{B}^n$, $c \in \mathbb{B}$ and $I \subseteq V$, we define $\bar{x}_i = 1 - x_i$, $\bar{c} = 1 - c$ and $\bar{x}^I$ as $\bar{x}^I_i = x_i$ for $i \in V \backslash I$ and $\bar{x}^I_i = 1 - x_i$ for $i \in I$. If $I = \{i\}$, $\bar{x}^I$ is written as $\bar{x}^i$.

The *interaction graph* of a Boolean network f is defined as the labeled multi-digraph (V, E) with $E \subseteq V \times V \times \{+, -\}$, admitting an edge from i to j if there exists $x \in \mathbb{B}^n$, such that $s = (f_j(\bar{x}^i) - f_j(x))(\bar{x}^i_i - x_i) \neq 0$. The label of the edge is given by the sign of s. The interaction graph captures the activation (positive) and inhibition (negative) relations between the components of a Boolean network. Figure 2.1 shows the interaction graph of a Boolean network in three variables.

(a)

$f_1 = x_2 \vee x_3$
$f_2 = x_1 x_2$
$f_3 = x_1 x_2 \bar{x}_3$

x	f(x)
000	000
001	100
010	100
011	100
100	000
101	100
110	111
111	110

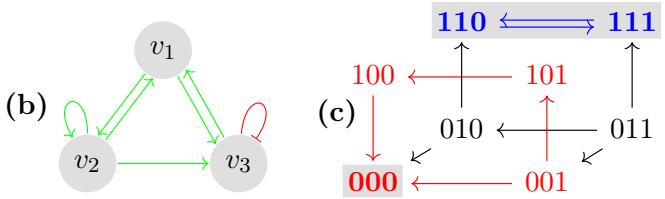


Figure 2.1: (a) Boolean network in three variables. (b) Interaction graph of the Boolean network in (a). Green edges denote activation interactions and red edges denote inhibitions. (c) Asynchronous dynamics of the Boolean network in (a). The two attractors $\mathcal{A}_1 = \{000\}$ and $\mathcal{A}_2 = \{110, 111\}$ are marked in gray. The states belonging to the $SB_{\mathcal{A}_1}$ and $SB_{\mathcal{A}_2}$ are marked in red and blue respectively. The states 010 and 011 belong to $WB_{\mathcal{A}_1}$ and $WB_{\mathcal{A}_2}$, since both attractors are reachable from them.

The dynamics of a Boolean network is defined by the *state transition graph (STG)*, a directed graph with node set $\mathbb{B}^n$. Given a Boolean function f, we can define different dynamics depending on the way the components are updated, giving rise to different state transition graphs.

- The *synchronous dynamics* $SD(f)$ defines transitions that update at the same time all the components that can be updated. The synchronous STG has an edge from $x \in \mathbb{B}^n$ to $y \in \mathbb{B}^n$ if $x \neq y$ and $y = f(x)$.
- The *asynchronous dynamics* $AD(f)$ defines transitions updating only one component at a time. The asynchronous STG has an edge from $x \in \mathbb{B}^n$ to $y \in \mathbb{B}^n$ if there exists $i \in V$ such that $y = \bar{x}^i$ and $y_i = f_i(x)$.
- The *generalized asynchronous dynamics* $GD(f)$ defines transitions that update a non-empty subset of components. Given $x, y \in \mathbb{B}^n$ there is a transition from x to y in the generalized asynchronous STG if there exists a subset $\emptyset \neq I \subseteq V$ such that $y = \bar{x}^I$ and $y_i = f_i(x)$ for all $i \in I$.

In order to better capture the different (and sometimes unknown) time scales that might coexist in a biological system, the asynchronous or generalized asynchronous dynamics are often used. In the cases when information about the relation between the update rates of the different components is available, other types of updates for instance sequential or block sequential update can be defined [DS20]. In this work we focus on the main three dynamics defined above, although most of the results are also applicable to other types of updates.

To ease the notation, we use $D(f)$ to refer to any of the three dynamics. Figure 2.2 shows a representation of the three different dynamics (synchronous, asynchronous and generalized asynchronous) for a Boolean network in four variables.

A *path or trajectory* in an STG is defined as a sequence of nodes $\pi = x^0, x^1, \ldots$ such that there exists an edge from x^{i-1} to x^i for all $i \geq 1$. We might represent a path using arrows to denote transitions: $\pi = x^0 \to x^1 \to \ldots \to x^k$. The set of all paths starting at a state x is denoted by $\text{Paths}(x)$. Given a state $x \in \mathbb{B}^n$, we define $\text{Reach}(x) = \{y \in \mathbb{B}^n \mid \exists \pi \in \text{Paths}(x)$ s.t. $y \in \pi\}$ and given a subset $S \subseteq \mathbb{B}^n$, $\text{Reach}(S)$ is the set $\{y \in \mathbb{B}^n \mid y \in \text{Reach}(x)$ for some $x \in S\}$. Note that $x \in \text{Reach}(x)$, since x is the 1-element path to x. A set $T \subseteq \mathbb{B}^n$ such that $T = \text{Reach}(T)$ is called a *trap set*. Trap sets are therefore subsets closed with respect to the dynamics and they correspond to unions of strongly connected components of the STG that do not admit any outgoing edge.

An *attractor* is a minimal trap set under inclusion. Attractors correspond to the terminal strongly connected components of the STG. Given an attractor $\mathcal{A} \subseteq \mathbb{B}^n$, we call it *steady state* (or *fixed point*) if $|\mathcal{A}| = 1$ and *cyclic attractor* (or *complex attractor*) if $|\mathcal{A}| > 1$. In biological systems, steady states might correspond to different cell fates or cell types, while cyclic attractors might be associated with cell cycles or cell processes with sustained oscillation. The Boolean network in Figure 2.1 has two attractors in the asynchronous dynamics, one steady state and one cyclic attractor. Since the STG might vary in different updates, cyclic attractors and trap sets might also vary in different dynamics. Steady states, however, are the same in any update. Figure 2.2 shows the synchronous, asynchronous and general asynchronous dynamics of a Boolean function. All three dynamics have two attractors: a steady state and a cyclic attractor. The steady state is the same in all the dynamics whereas the cyclic attractor is different in each of them. Note that, in non-deterministic dynamics such as asynchronous or generalized asynchronous, there can be non-attractive cycles in the dynamics, that is, cycles that trajectories can leave after entering. In application, trajectories that stay indefinitely in a non-attractive cycle are usually taken as modeling artifacts and are not further considered. In this work, we accept the standard view assuming that the trajectories of interest will eventually leave the non-attractive strongly connected components in the state transition graph.

Given an attractor $\mathcal{A}$, we define the *weak basin of attraction of* $\mathcal{A}$ as the set $\text{WB}_\mathcal{A} \subseteq \mathbb{B}^n$ such that $\mathcal{A} \subseteq \text{Reach}(x)$ for every $x \in \text{WB}_\mathcal{A}$ and the *strong basin of attraction of* $\mathcal{A}$ as the set $\text{SB}_\mathcal{A} \subseteq \mathbb{B}^n$ such that for every $x \in \text{SB}_\mathcal{A}$, $x \in \text{WB}_\mathcal{A}$ and $x \notin \text{WB}_{\mathcal{A}'}$ for every attractor $\mathcal{A}' \neq \mathcal{A}$. Note that $\text{SB}_\mathcal{A}$ is a trap set. Figure 2.1 (c) shows the basins of attraction of two attractors in the asynchronous dynamics.

Given a set of variables $I \subseteq V$ and a state $c \in \mathbb{B}^n$ we define the *subspace induced by* I *and* c as the set of states $\Sigma(I, c) = \{x \in \mathbb{B}^n \mid x_i = c_i \ \forall i \in I\}$. We denote subspaces by writing a 0 or a 1 for the fixed variables and a $*$ for the free variables. Thus, $10{*}{*}$ denotes the subspace fixing the first component to 1 and the second component to 0, that is, $\{x \in \mathbb{B}^n \mid x_1 = 1$ and $x_2 = 0\}$. A subspace that is also a trap set is called *trap space*.

Trap spaces are subspaces closed with respect to the dynamics. They can also be thought of as generalized or partial steady states. In particular, each steady state is a trap space. Note that there is always a trivial trap space that is the complete state space $\mathbb{B}^n$. While attractors and trap sets might vary in different updates, trap spaces (as steady states) are always the same in all the dynamics. The Boolean network in Figure 2.2 has five trap spaces: 0∗∗0, 10∗∗, 10∗1, 1001, ∗∗∗∗.

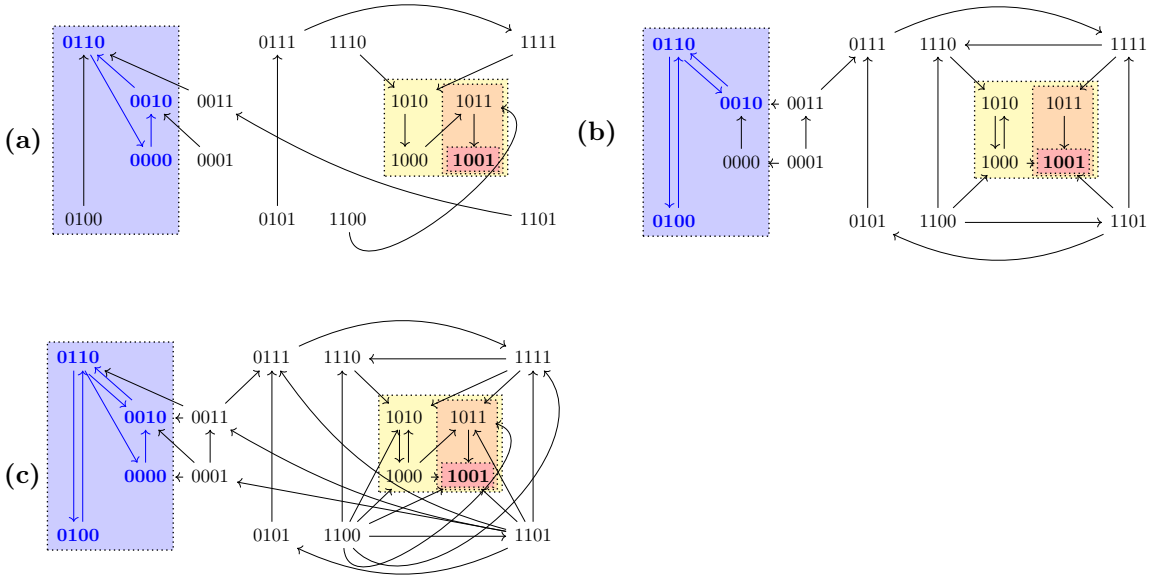


Figure 2.2: State transition graphs of the Boolean function $f(x) = (x_1\bar{x}_2 \vee x_1x_3 \vee x_1\bar{x}_4 \vee x_2x_3x_4,\ \bar{x}_1\bar{x}_2x_3 \vee \bar{x}_1x_2\bar{x}_3 \vee \bar{x}_1x_3x_4,\ x_1x_2 \vee \bar{x}_1\bar{x}_2 \vee \bar{x}_1x_2\bar{x}_3 \vee \bar{x}_1x_2x_4 \vee \bar{x}_2\bar{x}_3\bar{x}_4,\ \bar{x}_1x_2x_4 \vee x_1\bar{x}_2\bar{x}_3 \vee x_1x_2\bar{x}_3 \vee x_1\bar{x}_2x_4)$ in (a) synchronous, (b) asynchronous and (c) generalized asynchronous dynamics. The trap spaces, denoted by colored boxes, (0∗∗0, 10∗∗, 10∗1, 1001, ∗∗∗∗) and the steady state (1001) are the same in the three dynamics, while the cyclic attractors (marked in blue) vary. All three dynamics admit one cyclic attractor, the set $\{0110, 0010, 0000\}$ in the synchronous case, $\{0100, 0110, 0010\}$ in the asynchronous and $\{0100, 0110, 0010, 0000\}$ in the general asynchronous.

By definition, every trap space contains at least one attractor. In some cases, minimal trap spaces can be good approximations of attractors, that is, they are in one-to-one correspondence with the attractors. We can determine when minimal trap spaces are good approximation of attractors by checking three properties: univocality, faithfulness and completeness [KS15]. Given a Boolean function f, a dynamics D and a trap space

T, we say that T is *univocal* in $D(f)$ if it only contains one attractor of $D(f)$ and that T is *faithful* in $D(f)$ if for every attractor $\mathcal{A}$ of $D(f)$ such that $\mathcal{A} \subseteq T$, T is the smallest subspace containing $\mathcal{A}$. A set of trap spaces $\mathcal{T}$ is *complete* in $D(f)$ if for any attractor $\mathcal{A}$ of $D(f)$, there exists a trap space $T \in \mathcal{T}$ such that $\mathcal{A} \subseteq T$. Given a Boolean function f and a dynamics D, if all the minimal trap spaces of f are univocal and faithful in $D(f)$ and the set of minimal trap spaces is complete in $D(f)$, then the minimal trap spaces of f are in one-to-one correspondence with the attractors of $D(f)$ [KS15]. In such cases, we say that the minimal trap spaces of f are a *good approximation* of the attractors of $D(f)$. In the examples of Figure 2.2, the two minimal trap spaces 1001 and $0{*}{*}0$ are a good approximation of the attractors in the three dynamics. The properties of univocality, faithfulness and completeness are often satisfied by the minimal trap spaces of networks modeling biological systems [KS15].

2.1.1 Value percolation

In the following, we introduce the concept of value percolation in Boolean functions and derive some properties that set the basis of many results in this work. We start by defining the percolation function with respect to a Boolean function f, which associates to every subspace S the subspace determined by the components fixed by f within S.

Definition 2.1.1. Let $\mathcal{S}$ be the set of all subspaces in $\mathbb{B}^n$ and f a Boolean function. We define the *percolation function* with respect to f as the function $F(f)\colon \mathcal{S} \to \mathcal{S}$ that maps a subspace $S \in \mathcal{S}$ to the smallest subspace that contains the image of S under f, $f(S)$, with respect to inclusion.

Explicitly, given a subspace $S \in \mathcal{S}$, $F(f)(S) = \Sigma(I, c)$ with $I = \{i \in V \mid |f_i(S)| = 1\}$ and c any state in $\mathbb{B}^n$ such that $c_i = f_i(x)$ for all $x \in S$ for $i \in I$. Given a Boolean function f and two subspaces $S, S' \subseteq \mathbb{B}^n$, we say that the subspace S *percolates to* S' *under* f if and only if there exists $k \geq 0$ such that $F(f)^k(S) = S'$.

Example 2.1.2. Consider the Boolean network in Figure 2.1, $f(x) = (x_2 \vee x_3,\ x_1x_2,\ x_1x_2\bar{x}_3)$, and the subspaces $S_1 = {*}1{*}$ and $S_2 = {*}0{*}$. Since $f(x) = (1, x_1,\ x_1\bar{x}_3)$ for $x \in S_1$ and $f(S_2) = (x_3,\ 0, 0)$ for $x \in S_2$, $F(f)(S_1) = 1{**}$ and $F(f)(S_2) = {*}00$. Taking $S_1' = 1{**}$ and $S_2' = {*}00$ and applying the percolation function again, we see that $F(f)^2(S_1) = F(f)(S_1') = {***}$ and $F(f)^2(S_2) = F(f)(S_2') = 000$. Figure 2.3 shows the image of $F(f)$ for every subspace.

For any trap space T and its image $T' = F(f)(T)$, we have $T' \subseteq T$, since by definition F preserves the fixed components of T. The free components in T might get fixed or remain free depending on the Boolean function f. In fact, T' is a trap space of f, since for any fixed variable $i \in I' \subset V$ with $T' = \Sigma(I', c')$, $f_i(x) = c_i'$ by definition of $F(f)$. Moreover,

S	F(f)(S)
***	***
**0	***
**1	1*0
0	*00
*00	000
*01	100
1	1**
*10	1**
*11	1*0
0**	*00
0*0	*00
0*1	100
00*	*00
000	000
001	100
01*	100
010	100
011	100
1**	***
1*0	***
1*1	1*0
10*	*00
100	000
101	100
11*	11*
110	111
111	110

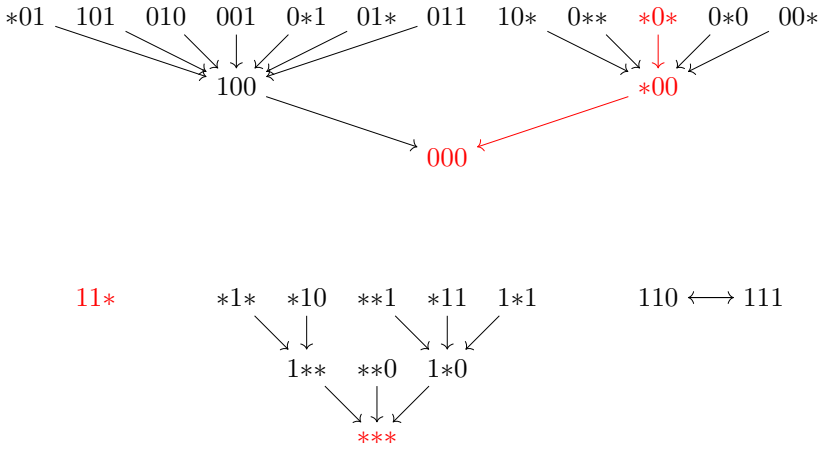


Figure 2.3: Table showing all the images under $F(f)$, with $f(x) = (x_2 \vee x_3,\, x_1x_2,\, x_1x_2\bar{x}_3)$, for each subspace in $\mathbb{B}^3$ (left). Synchronous dynamics of $F(f)$ (right). Trap spaces of f and edges between them are marked in red. Nodes without outgoing edges represent fixed points.

$F(f)^k(T)$ is a trap space for any $k \geq 0$. In Example 2.1.2, both S_2 and $F(f)(S_2)$ are trap spaces of f.

A dynamics of $F(f)$ can be defined by assigning a transition from every subspace S to its image under the percolation function, $F(f)(S)$, when $S \neq F(f)(S)$. This dynamics can be represented by a directed graph with node set $\mathcal{S}$, set of all subspaces in $\mathbb{B}^n$. Figure 2.3 shows the synchronous dynamics of $F(f)$ for the Boolean function discussed in Example 2.1.2.

Note that paths starting at a trap space T cannot have cycles of length greater than one, since the number of fixed variables is non-decreasing. Consequently, all the reachable attractors from T in this dynamics are fixed points. When considering the synchronous dynamics of $F(f)$, each initial trap space T leads to a unique fixed point. Note that all the trap spaces in Figure 2.3 (in red) lead to a unique fixed point. Given a Boolean function f and a trap space T, we call the unique fixed point $F^*(f)(T)$ of the synchronous dynamics of $F(f)$ reachable from T the *percolated subspace from T with respect to f*, that is, $F^*(f)(T) = F(f)^k(T)$ with k such that $F(f)^k(T) = F(f)^r(T)$ for all $r \geq k$.

The following lemma introduces a dynamical property about percolated subspaces.

Lemma 2.1.3. *Let f be a Boolean function, $T \subseteq \mathbb{B}^n$ a trap space. Let $k \geq 0$ and $T^k = F(f)^k(T)$. Then for every $x \in T$ there exists a path in $D(f)$ from x to some $y \in T^k$.*

Proof. It is enough to show that if T is a trap space, then for every $x \in T$ there exists a path in $D(f)$ from x to some $y \in F(f)(T)$. Set $T' = F(f)(T)$, with $T' = \Sigma(I', c')$. Since $T' \subseteq T$, for all $i \in I'$, $f_i(x) = c'_i$ by definition of F. If $f(x) = x$, then $x \in T'$ and so there exists a 1-element path to $x = y \in T'$. If $f(x) \neq x$, let us look at every update separately. In the asynchronous dynamics, for every $i \in I'$, x admits a successor y in $AD(f)$ with $y_i = c'_i$ and $y_j = x_j$ for $j \neq i$; therefore there exists a path from any state in T to T'. In the synchronous dynamics, $f_i(x) = c'_i$ for all $i \in I'$ and so x admits a successor $y \in T'$. The case of the generalized asynchronous dynamics follows from the other cases, since all the paths in $SD(f)$ or $AD(f)$ are also paths in $GD(f)$. □

Example 2.1.4. Let us consider the Boolean function from Figure 2.1, $f(x) = (x_2 \vee x_3,\ x_1x_2,\ x_1x_2\bar{x}_3)$, and the subspace $S = *0*$. $F(f)(S) = *00$ and $F(f)^2(S) = 000$. Thus, according to Lemma 2.1.3 for every $x \in S$, there exists a path to 000 in the asynchronous dynamics: $\pi_1 = 000 \to 000$, $\pi_2 = 001 \to 000$, $\pi_3 = 100 \to 000$, $\pi_4 = 101 \to 100 \to 000$.

A consequence of Lemma 2.1.3 is that, given a trap space T that percolates to a trap space T', there cannot be an attractor $\mathcal{A} \subseteq T$ that is not contained in T', since for every state $x \in T$ there exists a path to some $y \in T'$.

Corollary 2.1.4.1. *Let f be a Boolean function, $\mathcal{A}$ an attractor of $D(f)$ and $T, T' \subseteq \mathbb{B}^n$ two trap spaces such that $T' = F(f)^k(T)$ for some $k \geq 0$. If $\mathcal{A} \subseteq T$ then $\mathcal{A} \subseteq T'$.*

Note that the properties deduced in Lemma 2.1.3 and Corollary 2.1.4.1 are valid for any of the three updates considered in this work.

Given a Boolean function f and using Corollary 2.1.4.1 with $T = \mathbb{B}^n$, we can deduce that the percolated trap space $T' = F^*(f)(\mathbb{B}^n)$ contains all the attractors of f. Since T' is a trap space, it is enough to explore the dynamics of f in T' to identify the attractors. Moreover, since the long-term dynamics inside a trap space only depends on the free variables, the attractors of f in T' would correspond to the attractors of the restriction of f to these free

variables within T'. To formalize this idea, we define the restriction of a Boolean function to a trap space.

Definition 2.1.5. Let f be a Boolean function, $S = \Sigma(I, c)$ a subspace and $U = V \backslash I$ the set of free variables of S. Without loss of generality, we assume that $U = \{1, \dots, m\}$ with $m = |U|$. We define the *restriction of f to S* as $f_S \colon \mathbb{B}^m \to \mathbb{B}^m$, with $f_S = \pi_S \circ f \circ \iota_S$, where $\pi_S \colon \mathbb{B}^n \to \mathbb{B}^m$ with $(\pi_S(x))_i = x_i$ for all $i \in U$, $x \in \mathbb{B}^n$ and ι_S is defined as

$$\iota_S \colon \mathbb{B}^m \to \mathbb{B}^n, \text{ with } (\iota_S(\tilde{x}))_i = \begin{cases} \tilde{x}_i, & i \in U, \\ c_i, & i \notin U, \end{cases} \quad \text{for all } \tilde{x} \in \mathbb{B}^m.$$

ι_S represents the inclusion of a state from the smaller state space, $\tilde{x} \in \mathbb{B}^m$, in the subspace $S = \Sigma(I, c) \subseteq \mathbb{B}^n$, since it assigns the value c_i to each component i fixed in S, $i \in I$, and keeps the original value for the free variables. Thus, $\iota_S(\tilde{x}) \in S$ for all $\tilde{x} \in \mathbb{B}^m$.

Remark 2.1.6. Note that for any trap space T and $x, y \in \mathbb{B}^n$, if there is a path from $\iota_T(x)$ to $\iota_T(y)$ in $D(f_T)$ then there is a path from x to y in $D(f)$ and vice versa.

The previous remark can be extended to attractors.

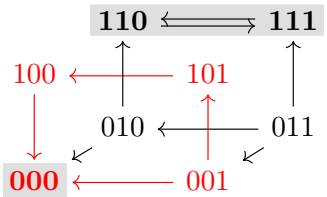
Proposition 2.1.7. *Let f be a Boolean function, $T = \Sigma(I, c)$ a trap space of f and $U = V \backslash I$ the set of free variables of T. Then,*

(i) if $\mathcal{A} \subseteq \mathbb{B}^n$ is an attractor of f such that $\mathcal{A} \subseteq T$, then there exists an attractor $\mathcal{A}' \subseteq \mathbb{B}^m$ such that $\pi_S(x) \in \mathcal{A}'$ for all $x \in \mathcal{A}$.

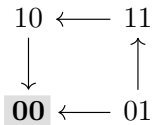
(ii) if $\mathcal{A}' \subseteq \mathbb{B}^m$ is an attractor of f, then there exists an attractor $\mathcal{A} \subseteq T \subseteq \mathbb{B}^n$ such that $\iota_S(\tilde{x}) \in \mathcal{A}$ for all $\tilde{x} \in \mathcal{A}'$.

Example 2.1.8. Let us consider the Boolean function from Figure 2.1, $f(x) = (x_2 \vee x_3, x_1 x_2, x_1 x_2 \bar{x}_3)$ for $x \in \mathbb{B}^3$, and the trap space $T = *0*$, with $I = \{2\}$ and $U = \{1, 3\}$. The restriction of f to T is given by $f_T(\tilde{x}) = (\pi_T \circ f \circ \iota_T)(\tilde{x}) = (f_1(\tilde{x}_1, 0, \tilde{x}_2), f_3(\tilde{x}_1, 0, \tilde{x}_2)) = (\tilde{x}_2, 0)$, for $\tilde{x} \in \mathbb{B}^2$. The asynchronous dynamics of f and f_T are shown in (a) and (b) respectively. Attractors are marked in bold and gray background. The trap space T is marked in red. We observe that all the paths in $AD(f)$ that are contained in T are preserved in $AD(f_T)$ and vice versa. Note that Proposition 2.1.7 is satisfied by the only attractor in $AD(f)$ contained in T, $\mathcal{A} = 000$, and the only attractor in $AD(f_T)$, $\mathcal{A}' = 00$.

(a)



(b)



2.2 Controlled function and control strategies

In this work, we consider two type of interventions: node interventions and edge interventions. Node interventions target a component (node) of the network, fixing it to a certain value. More formally, a node intervention (i, c), with $i \in V$ and $c \in \mathbb{B}$, sets the component i and its regulatory function f_i to the value c. Edge interventions, on the other hand, act on the interaction between two components, leaving the rest of the regulatory functions unaltered. Formally, an edge intervention (i, j, c), with $i, j \in V$ and $c \in \mathbb{B}$, fixes the value of the component i in the regulatory function f_j to the value c.

Node interventions can be seen for instance in a gene regulatory network, as the knock-out or permanent activation of a gene, whereas edge interventions can represent the alteration of a binding site of a protein in a way that it cannot interact with a chosen component but still perform other actions in the biological system. In the context of practical application, edge interventions are particularly interesting in components that are vital for other processes in the cell. In the cases where knocking them out or permanently activating them would endanger the survival of cell or cause other undesirable effects, edge interventions could help achieving the desired control without altering the rest of the system.

Given a Boolean network f and a set of interventions $\mathcal{C}$, which might be node or edge interventions, we consider the simultaneous application of all interventions in $\mathcal{C}$ on f, unless otherwise specified. We write $f^{\mathcal{C}}$ for the function resulting from the application of the interventions in the set $\mathcal{C}$. In this section we give the formal definition for the function $f^{\mathcal{C}}$ and a control strategy.

We start by establishing the basic conditions that a set of interventions needs to satisfy in order to be consistent. These conditions aim at preventing, for instance, that a node intervention fixes a component to 1 while another is fixing the same component to 0.

Definition 2.2.1. Let $\mathcal{N} \subseteq V \times \mathbb{B}$ and $\mathcal{E} \subseteq V \times V \times \mathbb{B}$. We say that the set of interventions $\mathcal{C} = \mathcal{N} \cup \mathcal{E}$ is *consistent* if and only if the following conditions are satisfied:

(i) for all $i, j \in V$ and $c, c' \in \mathbb{B}$, if $(j, c) \in \mathcal{C}$, then $(i, j, c') \notin \mathcal{C}$;

(ii) for all $i, j \in V$ and $c, c' \in \mathbb{B}$, if $(i, c) \in \mathcal{C}$, then $(i, \bar{c}) \notin \mathcal{C}$ and $(i, j, c') \notin \mathcal{C}$;

(iii) for all $i, j \in V$ and $c \in \mathbb{B}$, if $(i, j, c) \in \mathcal{C}$, then $(i, j, \bar{c}) \notin \mathcal{C}$.

The first condition ensures that node and edge interventions do not act on the same target. The second guarantees that when a node intervention fixes the value of a component, no other intervention fixes that component. The last one prevents edge interventions from fixing the same component to different values in the same regulatory function.

Example 2.2.2. Let $V = \{1, 2, 3\}$ be the set of variables of a Boolean function and $\mathcal{C}_1 = \{(1, 0), (2, 1, 1)\}$, $\mathcal{C}_2 = \{(1, 1), (2, 3, 0)\}$, $\mathcal{C}_3 = \{(3, 0), (2, 0), (2, 1, 0)\}$, $\mathcal{C}_4 = \{(1, 2, 0), (1, 2, 1)\}$ and $\mathcal{C}_5 = \{(1, 3, 1), (2, 3, 0)\}$ five sets of interventions. According to Definition 2.2.1, only $\mathcal{C}_2$ and $\mathcal{C}_5$ are consistent. $\mathcal{C}_1$, $\mathcal{C}_3$ and $\mathcal{C}_4$ are not consistent, since they do not fulfill (i), (ii) and (iii) respectively.

In order to describe the effect of node and edge interventions in a Boolean network, we first define a function $h^{j,\mathcal{C}} \colon \mathbb{B}^n \to \mathbb{B}^n$ that, given a set of consistent interventions $\mathcal{C}$ and a component $j \in V$, captures the effect of fixing the components involved in the interventions acting on the regulatory function of j. For every $x \in \mathbb{B}^n$, $i, j \in V$, we set

$$h^{j,\mathcal{C}}(x)_i = \begin{cases} c & \text{if } (i, c) \in \mathcal{C} \text{ or } (i, j, c) \in \mathcal{C} \text{ for some } c \in \mathbb{B}, \\ x_i & \text{otherwise.} \end{cases} \tag{2.1}$$

Note that $h^{j,\mathcal{C}}$ is well-defined when $\mathcal{C}$ is consistent. From now on, any set of interventions is assumed to be consistent unless otherwise specified. Given a Boolean network f and a consistent set of interventions $\mathcal{C}$, we can now define the controlled network $f^{\mathcal{C}}$. For every $k \in V$,

$$f_k^{\mathcal{C}} = \begin{cases} c & \text{if } (k, c) \in \mathcal{C} \text{ for some } c \in \mathbb{B}, \\ f_k \circ h^{k,\mathcal{C}} & \text{otherwise.} \end{cases} \tag{2.2}$$

Example 2.2.3. Let us consider the Boolean function from Figure 2.1, $f(x) = (x_2 \vee x_3,\ x_1x_2,\ x_1x_2\bar{x}_3)$, and the consistent intervention set $\mathcal{C} = \{(1, 1), (2, 3, 0)\}$. The corresponding controlled function is $f^{\mathcal{C}}(x) = (1,\ x_2,\ 0)$.

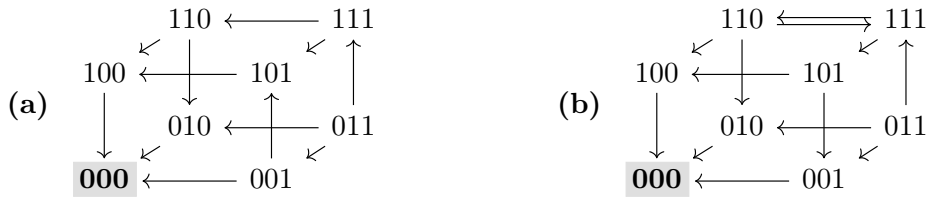
The interventions considered in node control fix certain components (nodes) to certain values. Thus, if a set of interventions consists exclusively of node interventions ($\mathcal{C} = \mathcal{N}$), then it can be associated with a subspace $\Sigma(I, d)$, with $I \subseteq V$ and $d \in \mathbb{B}^n$ such that $(i, d_i) \in \mathcal{N}$ if and only if $i \in I$. When all interventions are edge interventions ($\mathcal{C} = \mathcal{E}$) the controlled regulatory function for any component k is given by $f_k^{\mathcal{E}} = f_k \circ h^{k,\mathcal{E}}$. We define the effect of a node intervention (i, c) by setting the regulatory function f_i and every instance of the variable i to c. Other methods might consider only the fixing of the regulatory function. Both definitions lead to the same long-term behavior of the controlled system.

Given a node intervention (i, c), one could consider a set of edge interventions that fix the regulatory function f_i to c. For instance, if a node i has only one incoming edge from j in the interaction graph of f, that is, the regulatory function satisfies $f_i(x) = x_j$ or $f_i(x) = \bar{x}_j$, the node intervention (i, c) is equivalent in the long-term dynamics to the edge intervention (j, i, c) or $(j, i, \bar{c})$ respectively. Note that a node intervention might have multiple equivalent sets of edge interventions. For example, if $f_i(x) = x_j \vee x_k$ for some $i, j, k \in V$, using either the edge intervention $(j, i, 1)$ or the edge intervention $(k, i, 1)$ would have the same long-term effect in the dynamics as the node intervention $(i, 1)$. Since

the control problem considered within this work focuses on the long-term behavior of the system, the transient behavior of the controlled system is not relevant for our purposes. In the following, we define the concept of equivalent interventions based on their effect in the long-term dynamics. To do so, we consider the percolated subspace from $\mathbb{B}^n$ with respect to the controlled function $f^{\mathcal{C}}$, $F^*(f^{\mathcal{C}})(\mathbb{B}^n)$.

Definition 2.2.4. Let $\mathcal{C}_1, \mathcal{C}_2$ be two consistent sets of interventions and let f be a Boolean function. We say that $\mathcal{C}_1$ and $\mathcal{C}_2$ are *equivalent with respect to* f if and only if $F^*(f^{\mathcal{C}_1})(\mathbb{B}^n) = F^*(f^{\mathcal{C}_2})(\mathbb{B}^n)$ and $f^{\mathcal{C}_1}(x) = f^{\mathcal{C}_2}(x)$ for all $x \in F^*(f^{\mathcal{C}_1})(\mathbb{B}^n)$.

Example 2.2.5. Let $\mathcal{C}_1 = \{(2,0)\}$ and $\mathcal{C}_2 = \{(1,2,0)\}$ be two consistent intervention sets and $f(x)$ the Boolean function from Figure 2.1, $f(x) = (x_2 \vee x_3,\ x_1x_2,\ x_1x_2\bar{x}_3)$. The controlled functions for each set of interventions are $f^{\mathcal{C}_1}(x) = (x_3,0,0)$ and $f^{\mathcal{C}_2}(x) = (x_2 \vee x_3,\ 0,\ x_1x_2\bar{x}_3)$ respectively. Note that, although $f^{\mathcal{C}_1}$ and $f^{\mathcal{C}_2}$ are not the same function and their STGs have different transitions (see below), both of them have the same long-term dynamics: the steady state 000. In fact, $\mathcal{C}_1$ and $\mathcal{C}_2$ are equivalent with respect to f since $F^*(f^{\mathcal{C}_1})(\mathbb{B}^n) = 000 = F^*(f^{\mathcal{C}_2})(\mathbb{B}^n)$ and $f^{\mathcal{C}_1}(000) = 000 = f^{\mathcal{C}_2}(000)$. The asynchronous STG of $f^{\mathcal{C}_1}$ (a) and $f^{\mathcal{C}_2}$ (b) are represented below. The steady state is marked in gray.



In the following, we give a brief overview of the different types of control and the main factors defining a control problem. We also present the control problem considered in this work and give the formal definition of control strategy.

Control goal. A control problem is primarily determined by its goal, which is defined by the desired state of the system that needs to be achieved. In some cases, this state might correspond to a specific attractor. For instance, in a network modeling a cell-fate decision system for cancer cells, it might be desirable to induce the apoptotic attractor in order to lead the system to the elimination of the unhealthy cells. This type of control, which takes an attractor (steady state or cyclic) as goal, is known as *attractor control*. In other cases, it might not be necessary to reach a specific attractor and it might be enough to ensure that a subset of relevant components are in the desired state. For instance, in the previous example of a cell-fate decision network, if there were multiple attractors grouped in two phenotypes (apoptosis or proliferation) it might be enough to ensure that the system reaches any

attractor belonging to the apoptotic phenotype. In many cases, these phenotypes are defined by the value of a subset of observable components or biomarkers. Thus, the control problem reduces to ensuring that these components are in the desired state in all the attractors of the controlled system. This type of control, which focuses on the state of a set of relevant components, is known as *target control*. Note that the set of relevant components and their desired values define a subspace. The control approach presented in Chapter 3 deals with target control, having a subspace as the control target. Formally, the goal of a control problem can be described by a subset $P \subseteq \mathbb{B}^n$. Attractor control ($P = \mathcal{A}$, attractor) and target control ($P = S$, subspace) are the most common control goals. However, in some cases it might be interesting to be able to deal with more generic control goals such as groups of attractors ($P = \bigcup_i \mathcal{A}_i$, union of attractors) and attractor avoidance ($P = \mathbb{B}^n \backslash (\bigcup_i \mathcal{A}_i)$). The control approach presented in Chapter 4 deals with control targeting a generic subset.

Control strategies. A control strategy can intuitively be defined as a set of control interventions that make the controlled system evolve to a desired state or set of states P. Following this idea, we define the control problem in terms of the long-term behavior of the controlled system. In particular, given a target P, we say that a set of interventions defines a control strategy for P when all the attractors of the controlled network are contained in P. We formalize this idea in the following definition.

Definition 2.2.6. Given a Boolean network f and a subset $P \subseteq \mathbb{B}^n$, a set of interventions $\mathcal{C}$ is a *control strategy for the target* P in $D(f)$ if $\mathcal{A} \subseteq P$ for any attractor $\mathcal{A}$ of $D(f^{\mathcal{C}})$.

Note that when considering only node control, since a set of node interventions $\mathcal{N}$ defines a subspace, a control strategy can also be identified with the subspace associated with $\mathcal{N}$ (see [CFTS20, CFTS22b]).

We define the size of a control strategy $\mathcal{C}$ as the number of interventions $|\mathcal{C}|$. In the case of $\mathcal{E} = \emptyset$, the number of interventions corresponds to the number of fixed variables. In practical applications, we are interested in intervention sets that are minimal with respect to inclusion. This is a natural approach when considering intervention sets that contain only node interventions or only edge interventions. For simplicity, in this work we use the same definition of minimality for intervention sets that mix edge and node interventions. Depending on the context, the resources required to implement different interventions can vary, and more sophisticated objective functions might take these differences into account.

There might be node or edge interventions whose practical application is not feasible. For this reason, the methods developed in this work allow for the exclusion of certain nodes or edges whose interventions are not considered in the control problem.

An example of a control strategy using node interventions is shown in Figure 2.4, where the set $\mathcal{N} = \{(1, 1)\}$, associated with the subspace $1{*}{*}$, is a control strategy for the one-element target subspace $P = 110$, since $f^{\mathcal{N}}$ only has one attractor that is the steady state

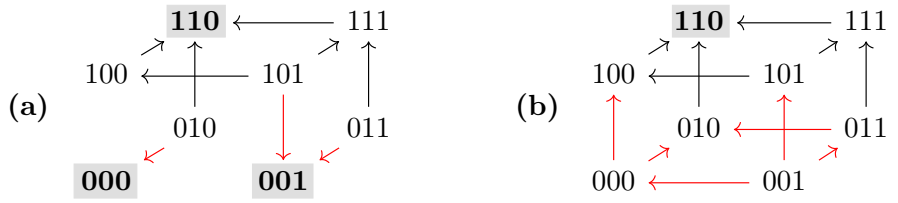


Figure 2.4: Asynchronous dynamics of the Boolean function $f(x) = (x_2 \vee x_1\bar{x}_3,\ x_1,\ \bar{x}_1x_3)$ (left) and $f^{\mathcal{N}}(x) = (1,\ 1,\ 0)$ with $\mathcal{N} = \{(1,1)\}$ (right). Transitions that vary between $AD(f)$ and $AD(f^{\mathcal{N}})$ are marked in red. Attractors are marked in bold. $\mathcal{N}$ is a control strategy for $P = 110$ in $AD(f)$.

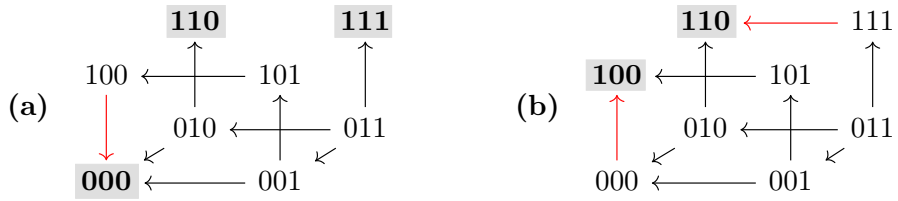


Figure 2.5: (a) Asynchronous dynamics of the Boolean function $f(x) = (x_2 \vee x_3,\ x_1x_2,\ x_1x_2x_3)$. (b) Asynchronous dynamics of the Boolean function $f^{\mathcal{E}}(x) = (1, x_1x_2, 0)$, with $\mathcal{E} = \{(2,1,1),(2,3,0)\}$. Transitions that vary between $AD(f)$ and $AD(f^{\mathcal{E}})$ are marked in red. Attractors are marked in bold. $\mathcal{E}$ is a control strategy for $P = 1{*}0$. There is no control strategy for P consisting only of a node intervention on the second component.

110. Figure 2.5 shows an example where the set of edge interventions $\mathcal{E} = \{(2,1,1),(2,3,0)\}$ is a control strategy for the target $P = 1{*}0$. Apart from the trivial node interventions targeting the fixed variables of P, there are no node control strategies for P, since $000 \notin P$ is a steady state of $f^{\mathcal{N}_0}$, with $\mathcal{N}_0 = \{(2,0)\}$, and $111 \notin P$ is a steady state of $f^{\mathcal{N}_1}$, with $\mathcal{N}_1 = \{(2,1)\}$. Thus, in this scenario, which excludes node interventions in the fixed variables of P, control can only be achieved by using edge interventions. This example illustrates how edge interventions can broaden the possibilities for control.

Initial states. Another aspect to consider when defining a control problem is the initial state of the system. The subset of states $\mathcal{I} \subseteq \mathbb{B}^n$ defining the initial state of the system can be an attractor or set of attractors ($\mathcal{I} = \bigcup_i \mathcal{A}_i$), for instance in the context of cell reprogramming, when we aim to drive the dynamics from an initial cell type to the desired one. It can also be determined by the value of some observable components or biomarkers. In this case, $\mathcal{I}$ is the subspace defined by the value of the biomarkers. We set $\mathcal{I} = \mathbb{B}^n$ in the cases in which the initial state of the system is not known or the control interventions must be valid for any possible initial state. The control problem with $\mathcal{I} = \mathbb{B}^n$ is known

as *full-network control*. The control approaches developed in this work aim at full-network control and identify strategies for any possible initial state.

Strategy type and transient control. The control strategies described by Definition 2.2.6 consist of permanent interventions (*permanent control*) applied all at once (*one-step control*). In some cases, the control interventions can be released once the dynamics has entered a particular region of the state space (see Section 3.1), which we call *flexible control*. Definition 2.2.6 can be extended to consider sets of interventions applied sequentially $C_1, \mathcal{C}_2, \ldots, \mathcal{C}_k$ (*sequential control*). Sequential control methods can provide alternative control strategies, involving potentially fewer perturbations [MSH+19]. In this work, we focus on permanent, one-step control (Chapters 3 and 4) and a simple type of sequential control that takes only two steps, applying the control and releasing it (Section 3.2), that represents the transient behavior of the control interventions (*transient control*). Formally, the sequential set of interventions consists of $\mathcal{C}_1 = \mathcal{C}$ and $C_2 = \emptyset$. The concept of transient control strategy is given in the following definition.

Definition 2.2.7. Given a Boolean function f and a subset $P \subseteq \mathbb{B}^n$, a set of interventions $\mathcal{C}$ is a *transient control strategy for the target* P in $D(f)$ if there exists a subset $P' \subseteq \mathbb{B}^n$, such that if $\mathcal{A}'$ is an attractor of $f^{\mathcal{C}}$ then $\mathcal{A}' \subseteq P'$ and for any attractor $\mathcal{A}$ of $D(f)$, if $\mathcal{A} \cap \mathrm{Reach}_{D(f)}(P') \neq \emptyset$ then $\mathcal{A} \subseteq P$.

In other words, given a transient control strategy $\mathcal{C}$, first the set of interventions $\mathcal{C}$ induces the system to reach a subset P' and then the release of these interventions guarantees that the long-term dynamics evolves to the desired target. In particular, the subset P' acts as an intermediate region that the system needs to reach while the control interventions are active. For this reason, we require that the subset P' contains the long-term dynamics of $f^{\mathcal{C}}$. Once this intermediate region is reached, the control interventions can be released, since the only attractors of f reachable in $D(f)$ from P' are contained in the target P.

Figure 2.6 shows an example of a transient control strategy. The original asynchronous dynamics has two steady states: 100 and 011, and the desired target is $P = *0*$. $\mathcal{C} = \{(1,1)\}$ is a transient control strategy for P because there exists a subset $P' = 1*0$ such that the two attractors of $AD(f^{\mathcal{C}})$, 100 and 110, are contained in P' and the only attractor of $D(f)$ that has non-empty intersection with $\mathrm{Reach}_{D(f)}(P') = **0$ is 100, which is contained in P. Note that $\mathcal{C}$ is not a permanent control strategy for P since $AD(f^{\mathcal{C}})$ has a steady state (110) not contained in P.

In the example of Figure 2.6, three update steps are enough to ensure that the system has reached the intermediate subset P'. In practical applications, where different reactions might have different time scales, it might be necessary to verify that the intermediate subset has been reached before releasing the control interventions. This could be done, for instance, by measuring some components or by waiting enough time for the system to stabilize.

Figure 2.6: Asynchronous dynamics of the Boolean function $f(x) = (\bar{x}_2\bar{x}_3,\ \bar{x}_1x_3 \vee x_1x_2\bar{x}_3,\ \bar{x}_1x_2x_3)$ (left) and $f^{\mathcal{C}}(x) = (1,\ x_2\bar{x}_3,\ 0)$ with $\mathcal{C} = \{(1,1)\}$ (right). Transitions that vary between $AD(f)$ and $AD(f^{\mathcal{C}})$ are marked in red. Attractors are marked in bold and gray background. $\mathcal{C}$ is a transient control strategy for $P = *0*$ since there exists a subset $P' = 1*0$ (green) that contains the two attractors of $AD(f^{\mathcal{C}})$, 100 and 110, and the only attractor of $AD(f)$ that is reachable from P' is $100 \in P$. The set of states reachable from P' is marked in blue.

Transient control strategies can help broadening the possibilities for potential applications by considering interventions whose effect only lasts temporarily. Some interventions, for instance the intake of drugs, might require a periodical application when their effects decay over time. The use of transient controls could help address this challenge by identifying strategies whose interventions would not need to be maintained permanently over time.

Many different control problems can be defined by considering different target goals, initial states, duration of the interventions, types of strategies or intervention targets (see Figure 2.7).

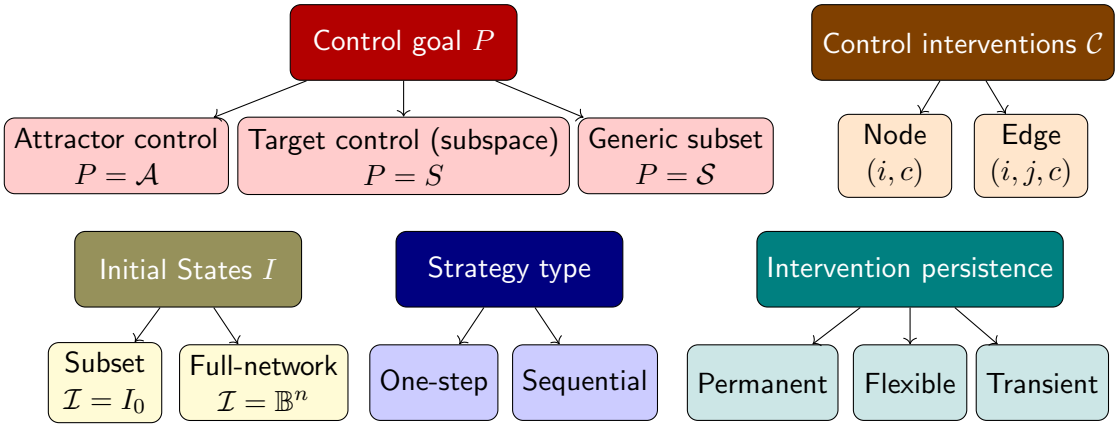


Figure 2.7: Control problems in terms of goal, initial states, strategy type, intervention persistence and intervention target.

The methods developed in this work deal with the problem of full-network control, providing control strategies that are valid for any possible initial state, and can take as goal a subspace (Chapter 3) or any type of subset (Chapter 4). They consider node and edge interventions applied permanently in one-step in Section 3.1 and Chapter 4, although the method in Section 3.1 allows the interventions to be eventually released. In addition, a method for transient control, or two-step sequential, is presented in Section 3.2. All these methods are valid for the three updates introduced in this chapter: asynchronous, synchronous and generalized asynchronous. However, since the asynchronous dynamics is often used for modeling biological systems, we focus our examples and analyses on this update. Comments on the results for the other updates are shown in the application sections.

Chapter 3

Control via trap spaces

A common approach to control in the Boolean framework is the use of value percolation, which consists in checking how the fixed values propagate through the network, to establish whether the effect of percolating the interventions is sufficient to induce the target state [SKK10]. Although methods based uniquely on value percolation allow for efficient computation [KSSV13], they can miss many control strategies. In order to increase the number of control strategies identified while still benefiting from an efficient implementation, we introduce the use of trap spaces [CFTS20, CFTS22c].

In this chapter we describe two approaches to control strategy identification, which are based on value percolation and use trap spaces to identify new control strategies. Both approaches take a subspace as control goal and allow for node and edge interventions. The first one considers permanent perturbations while the second one uses temporary interventions.

The implementation of these approaches is based on Answer Set Programming (ASP). ASP is a form of declarative programming that allows us to deal with the combinatorial explosion associated with the exponential number of candidate sets in the computation of control strategies. The use of ASP was proposed by Kamisnki et al. [KSSV13] in their implementation for node control strategy identification using value percolation. In [CFTS22a] we extended the work done in [KSSV13] to use ASP to identify control strategies with the trap-space approach initially presented in [CFTS20]. Later, we extended this implementation to also deal with edge control [CFTS22c].

The main results related to the permanent control method presented in this chapter are published in Cifuentes-Fontanals et al. [CFTS22c], from which I am first author and main developer of the method. In particular, Sections 3.1, 3.3 and 3.4 are extracted from [CFTS22c], with the permission of the co-authors. The chapter is organized as follows. In the first section, we recall the usual approach to control by direct percolation and present our approach using trap spaces. In Section 3.2, we extend the approach via trap spaces

to transient control. The implementation of both approaches is detailed in Section 3.3, including the ASP encoding. We finish by applying our approaches to a biological case study (Section 3.4).

3.1 Permanent control strategies

We start by recalling the property described in Corollary 2.1.4.1, which states that, given a trap space T that percolates to a subspace S, there cannot be an attractor $\mathcal{A} \subseteq T$ that is not contained in S. Taking $T = \mathbb{B}^n$, the following result can be derived.

Proposition 3.1.1. *Let $P \subseteq \mathbb{B}^n$ be a subspace and f a Boolean function. Let $\mathcal{C}$ be a set of interventions such that $\mathbb{B}^n$ percolates to a subspace $S \subseteq P$ under $f^{\mathcal{C}}$. Then $\mathcal{C}$ defines a control strategy in $D(f)$ for P.*

The idea of Proposition 3.1.1 is that, since all the attractors of $D(f^{\mathcal{C}})$ in $\mathbb{B}^n$ are also contained in $S = F^k(f^{\mathcal{C}})(\mathbb{B}^n)$ for any $k > 0$, if $S \subseteq P$ then all the attractors of $f^{\mathcal{C}}$ are contained in P and $\mathcal{C}$ is a control strategy for P in $D(f)$.

We refer to the control strategies satisfying the conditions of Proposition 3.1.1 as *control strategies by direct percolation*. This type of control strategy requires, in general, that the interventions are applied permanently. An example of such a control strategy is shown in Figure 3.1. There, $\mathcal{N} = \{(1, 1)\}$ is a control strategy by direct percolation in $D(f)$ for the target $P = **1$. In particular, $\mathbb{B}^n$ percolates to 111 under $f^{\mathcal{N}}$. The only attractor of $D(f^{\mathcal{N}})$ is 111 which is not an attractor of the original system $D(f)$. If the control interventions were released, the system would evolve to the steady state 010, which is not in P, and the desired target would be lost. Note that permanent interventions may induce the creation of new attractors, as seen in Figure 3.1. Thus, control strategies by direct percolation are useful when the definition of the control problem does not require the dynamics to evolve to an attractor of the original network but rather to any attractor that is contained in the target subspace. This is the case for the control problem defined in this work. Several approaches to the identification of control strategies by direct percolation using node control have been developed [SKK10, YGTZA18] and there exist implementations that identify all control strategies by direct percolation efficiently [KSSV13]. However, there are still many control strategies that do not fulfill the conditions of Proposition 3.1.1. Figure 3.2 shows an example of a control strategy that does not percolate to the target subspace.

In order to exploit the efficiency of value percolation to identify more control strategies, we developed a method based on percolation that uses trap spaces [CFTS20]. As mentioned before, trap spaces are subspaces closed for the dynamics. Thus, each trap space contains at least one attractor. From all trap spaces of a Boolean function, we select the ones that

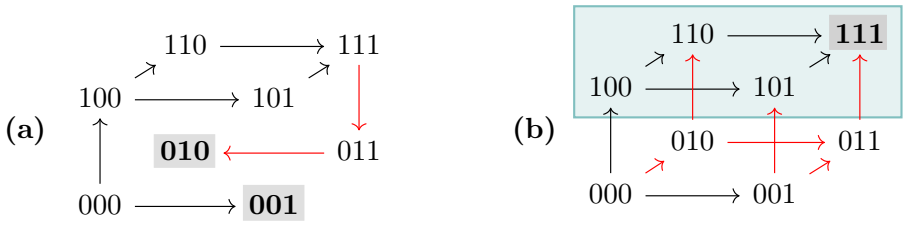


Figure 3.1: (a) Asynchronous dynamics of the Boolean function $f(x) = (x_1\bar{x}_2 \vee x_1\bar{x}_3 \vee \bar{x}_2\bar{x}_3, x_1 \vee x_2, x_1 \vee \bar{x}_2)$. (b) Asynchronous dynamics of the Boolean function $f^{\mathcal{N}}(x) = (1, 1, 1)$ with $\mathcal{N} = \{(1, 1)\}$. Transitions that vary between $AD(f)$ and $AD(f^{\mathcal{N}})$ are marked in red. Attractors are marked in bold and gray background. $\mathcal{N}$ is a control strategy for $P = **1$ in $AD(f)$. $\mathbb{B}^n$ percolates to $111 \in P$ under $f^{\mathcal{N}}$. The only attractor of $f^{\mathcal{N}}$ is 111 which is not an attractor of the original system. If the control interventions were released, the system would evolve to the steady state 010, which is not in P, and the desired target would be lost.

contain only attractors belonging to the target subspace. We call such trap spaces *selected trap spaces*. Proposition 3.1.2 introduces sufficient conditions for a set of interventions to be a control strategy for a target via a selected trap space.

Proposition 3.1.2. *Let $P \subseteq \mathbb{B}^n$ be a subspace and f a Boolean function. Let $T = \Sigma(I, d)$ be a trap space such that if $\mathcal{A} \subseteq T$ is an attractor of $D(f)$, then $\mathcal{A} \subseteq P$. Let $\mathcal{C}$ be a set of interventions such that $\mathbb{B}^n$ percolates to T under $f^{\mathcal{C}}$ and for all $(i, c) \in \mathcal{C}$ holds $i \in I$ and for all $(i, j, c) \in \mathcal{C}$ holds $j \in I$. Then $\mathcal{C}$ defines a control strategy in $D(f)$ for P.*

Proof. Let $\mathcal{A} \subseteq \mathbb{B}^n$ be an attractor for $D(f^{\mathcal{C}})$. Since $\mathbb{B}^n$ percolates to $T = \Sigma(I, d)$ under $f^{\mathcal{C}}$, for every $x \in \mathbb{B}^n$, in particular for every $x \in \mathcal{A}$, there exists a path in $D(f^{\mathcal{C}})$ from x to some $y \in T$. Therefore, $\mathcal{A} \subseteq T$. Since $\mathbb{B}^n$ percolates to T under $f^{\mathcal{C}}$, T is also a trap space in $f^{\mathcal{C}}$, so $f^{\mathcal{C}}_k(x) = d_k = f_k(x)$ for all $k \in I$ and $x \in T$. Since for all $(i, c) \in \mathcal{C}$, $i \in I$ and for all $(i, j, c) \in \mathcal{C}$, $j \in I$, we have that $f^{\mathcal{C}}_k(x) = f_k \circ h^{k,\mathcal{C}}(x) = f_k(x)$ for all $k \notin I$ and $x \in T$. Consequently, $f^{\mathcal{C}}(x) = f(x)$ for all $x \in T$. Since $\mathcal{A} \subseteq T$ and for all $x \in T, f(x) = f^{\mathcal{C}}(x)$, $\mathcal{A}$ is also an attractor of $D(f)$ and, therefore, $\mathcal{A} \subseteq P$. □

In the case $\mathcal{C} = \mathcal{N}$, the condition of $i \in I$ for all $(i, c) \in \mathcal{C}$ corresponds to $T \subseteq \Omega$, where Ω is the subspace associated to $\mathcal{N}$ [CFTS20].

We call the control strategies satisfying the conditions of Proposition 3.1.2 *control strategies via trap spaces*. Note that control strategies via trap spaces, contrary to control strategies by direct percolation, cannot introduce new attractors. All the attractors of the controlled network are attractors of the original system, since the dynamics within the trap space is preserved. Figure 3.2 shows an example of this type of control strategy. $T = **0$ is a selected trap space for the target $P = 110$, since it contains only the attractor $\mathcal{A} = 110$.

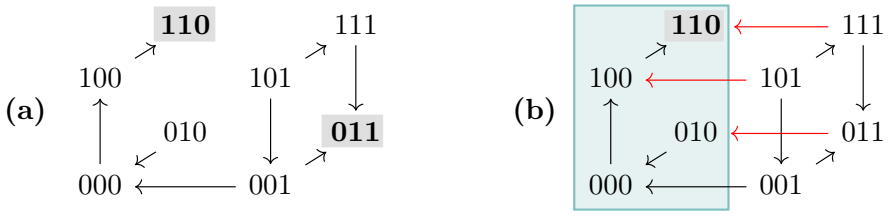


Figure 3.2: (a) Asynchronous dynamics of the Boolean function $f(x) = (x_1\bar{x}_3 \vee \bar{x}_2\bar{x}_3,\ x_1 \vee x_3,\ x_1x_3 \vee x_2x_3)$. (b) Asynchronous dynamics of the Boolean function $f^{\mathcal{N}}(x) = (x_1 \vee \bar{x}_2,\ x_1,\ 0)$ with $\mathcal{N} = \{(3, 0)\}$. Transitions that vary between $AD(f)$ and $AD(f^{\mathcal{N}})$ are marked in red. Attractors are marked in bold and gray background. $\mathcal{N}$ is a control strategy for $P = 110$ in $AD(f)$. $\mathbb{B}^n$ does not percolate to P under $f^{\mathcal{N}}$ but percolates to the selected trap space $T = **0$ (green). T is a selected trap space for the target P since the only attractor of $D(f)$ contained in T is also contained in P.

$\mathbb{B}^n$ percolates to T under $f^{\mathcal{N}}$, with $\mathcal{N} = \{(3, 0)\}$, since $F(f^{\mathcal{N}})(\mathbb{B}^n) = **0$. Consequently, $\mathcal{N}$ is a control strategy via trap spaces for P. Since $\mathbb{B}^n$ does not percolate to P under $f^{\mathcal{N}}$, $\mathcal{N}$ is not a control strategy by direct percolation. Figure 3.3 illustrates the main idea of the methods for control strategy identification by direct percolation and via trap spaces.

We can easily identify all the selected trap spaces if the attractors of the Boolean network are known or, alternatively, if they can be approximated by minimal trap spaces [KS15], that is, if each minimal trap space contains only one attractor and every attractor is included in a minimal trap space (see Section 2.1). Although attractor identification can be hard to achieve depending on the particular problem, the second property is easier to verify and is relatively common in Boolean networks modeling biological systems [KS15].

Control strategies by direct percolation do not depend on the update, since the properties deduced from the percolation function are valid for every update (see Chapter 2). The selected trap spaces, on the other hand, are defined in terms of the attractors, which might vary in different updates. As a consequence, control strategies via trap spaces are in general update-dependent. This provides the method with enough flexibility to identify control strategies that are valid in one update but not in another.

A further advantage of the control strategies identified by Proposition 3.1.2 is that they allow for the control interventions to be eventually released. Once a selected trap space is reached, the system will remain in the trap space, regardless of whether the control interventions are active or not (see Figure 3.3(b)). Moreover, if the dynamics has already stabilized, since all the attractors of the controlled network are also attractors in the original network, a release of the control would not alter the state of the system. Thus, control strategies via trap spaces can be applied using permanent interventions as well as transient ones. This additional property widens the range of possible choices for control since interventions relying on agents that decay over time could also be considered.

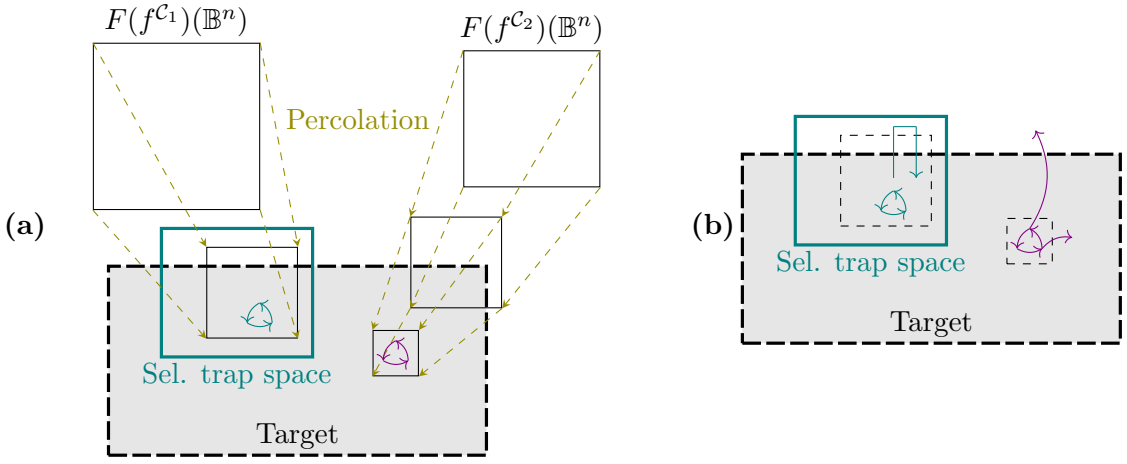


Figure 3.3: (a) Illustration of the ideas behind the methods for control strategy identification by direct percolation and via trap spaces. Rectangles represent subspaces, using solid lines in case of trap spaces and dashed lines otherwise. The yellow dashed lines represent the percolation process, with the state space percolating towards the target under two different control strategies ($\mathcal{C}_1$ and $\mathcal{C}_2$). The three-arrow cycles represent attractors. The selected trap space used for control via trap spaces is marked in green. (b) Possible trajectories of the system once the control is released. Note that none of the trajectories starting in a state belonging to the selected trap space can reach any attractor outside the target subspace (green arrows) whereas that might be possible in trajectories starting at any other point within the target subspace (purple arrows).

Although the methods using direct percolation and percolation via trap spaces are both based on value propagation, the control strategies that they identify can be very different. As mentioned before, control strategies by direct percolation might introduce new attractors on the controlled system, while control strategies via trap spaces preserve the original attractors within the selected trap space. Moreover, the interventions used in control via trap spaces must target the components fixed in the selected trap space, while control by direct percolation has no restriction on the interventions. Even when the candidate strategy consists of interventions targeting components fixed in a selected trap space, the controlled system could still percolate directly to the target subspace but not to the selected trap space. As a consequence, in many cases, there might be control strategies that are obtained by direct percolation that are not identified via trap spaces and vice versa. For this reason, it is useful to use the two methods in combination, that is, identifying sets of interventions that lead the state space to percolate directly to the target subspace or to any of the selected trap spaces (see Section 3.4.1 for an example of this scenario).

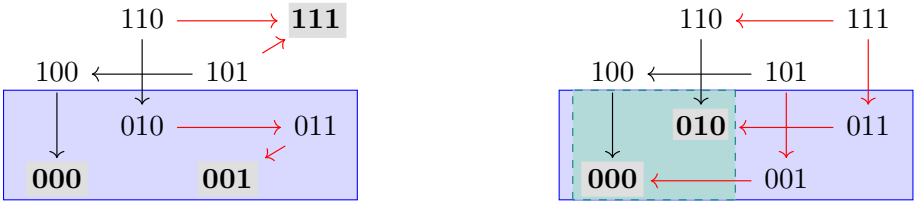


Figure 3.4: Asynchronous dynamics of the Boolean function $f(x) = (x_1x_3,\ x_1x_3 \vee x_2\bar{x}_3,\ \bar{x}_1x_3 \vee x_2)$ (left) and $f^{\mathcal{C}}(x) = (0,\ x_2,\ 0)$ with $\mathcal{C} = \{(3, 0)\}$ (right). Transitions that vary between $AD(f)$ and $AD(f^{\mathcal{C}})$ are marked in red. Attractors are marked in bold and gray background. $T = 0 * *$ (blue) is a trap space only containing attractors belonging to $P = 00*$. $\mathcal{C}$ is a transient control strategy for P since $\mathbb{B}^n$ percolates to $T' = 0 * 0$ (green) under $f^{\mathcal{C}}$, with $T' \subseteq T$. $\mathcal{C}$ is not a permanent control strategy for P since $AD(f^{\mathcal{C}})$ has a steady state (010) not contained in P.

3.2 Transient control strategies

In the previous section, we consider control strategies using permanent interventions applied all at once. In this section, we focus on transient control, or two-step sequential, as explained in Section 2.2. To avoid possible confusions, we call the usual control strategies *permanent control strategies*. We start by recalling the definition of transient control strategy from Definition 2.2.7, which states that, given a Boolean function f, a set of interventions $\mathcal{C}$ is a transient control strategy for a target P in $D(f)$ if there exists a subset $P' \subseteq \mathbb{B}^n$ that contains all the attractors of the controlled system $D(f^{\mathcal{C}})$ and that for any attractor $\mathcal{A}$ of the original network $D(f)$ that is reachable from P', $\mathcal{A}$ is contained in P.

A control strategy via trap spaces is always a transient control strategy, since the selected trap space T satisfies the conditions of P'. Once the system has reached the selected trap space the control interventions can be released since all the attractors reachable from T belong to the target subspace. This is not the case in control strategies by direct percolation, since they do not necessarily satisfy the requirements of Definition 2.2.7. An example of this is shown in Figure 3.1. The only attractor of the controlled function $f^{\mathcal{C}}$ is the steady state 111, so any possible intermediate subset P' must include 111. However, the steady state of the original network 010 is reachable from 111 in $D(f)$ and $010 \notin P$. Thus, $\mathcal{C}$ is a permanent control strategy by direct percolation but not a transient control strategy. On the other hand, not all the transient control strategies are permanent control strategies. Figure 3.4 shows an example of a transient control strategy $\mathcal{C} = \{(3, 0)\}$ for the target $P = 00*$ that is not a permanent control strategy, since $AD(f^{\mathcal{C}})$ has a steady state $010 \notin P$.

In the following, we focus on transient control strategies that use a selected trap space as intermediate subset (P'). The following proposition characterizes them.

Proposition 3.2.1. *Let P be a subspace, T a trap space that only contains attractors in P and $\mathcal{C}$ a set of interventions. If $\mathbb{B}^n$ percolates to a subspace $S \subseteq T$ under $f^{\mathcal{C}}$, then $\mathcal{C}$ is a transient control strategy for P.*

We call these control strategies *transient control strategies via trap spaces*. Note that given a transient control strategy $\mathcal{C}$ for the target P, there might be attractors in $D(f^{\mathcal{C}})$ that are not contained in P since $\mathcal{C}$ is not necessarily a permanent control strategy for P. Once the dynamics enters a selected trap space T, the release of the control can take place. Since T is a trap space in $D(f)$, the dynamics cannot leave T and since all the attractors contained in T are also contained in P, all the reachable attractors belong to P.

3.3 Implementation

The methods for control strategy identification presented in this chapter are based on the identification of sets of interventions that cause the state space to percolate either to the target subspace or to one of the selected trap spaces under the controlled function. Identifying all the minimal control strategies of this type might entail the exploration of all possible sets of interventions, whose number grows exponentially with the size of the network (for node control) or with the number of edges (for edge control).

The use of Answer Set Programming (ASP) was proposed by Kaminski et al. [KSSV13] to deal with the combinatorial explosion associated with node control in the computation of control strategies by direct percolation. Answer Set Programming is a form of declarative programming that works well with hard combinatorial, search and optimization problems. This type of problems often entail a decision-making process over a set of candidates to decide whether they satisfy a specified constraint and possibly identify an optimized output.

In this section, we recall the implementation presented in [CFTS22c], which identifies control strategies for node and edge control using the methods of direct percolation and percolation via trap spaces, and add its extension to transient control. We start by giving an introduction to Answer Set Programming, describing the basic semantics required to understand our implementation. Then, we provide a detailed description of the full ASP encoding for all the control methods. We end by giving an overview of the main algorithm for control strategy identification.

3.3.1 Basic introduction to Answer Set Programming

As mentioned above, Answer Set Programming is a form of declarative programming. This type of programming is based on knowing *what* the problem is and how a possible solution looks like rather than *how* to solve it. This differs from traditional programming where an algorithm to solve the problem needs to be provided. In declarative programming it is enough to describe the problem and the program will find a solution by the process of automated reasoning [GKKS12].

In order to solve a problem with ASP, it is needed to provide a description of the problem using logical rules. Solving the original problem is then reduced to identifying the solutions of its corresponding logic program. This process takes place in two steps: grounding and solving. In the first step, a grounder generates the propositional representation of the problem from the logical rules. In the second step, the solver computes the *stable models* or *solutions* of the propositional program. The interpretation of the stable models gives the solutions to the original problem. Figure 3.5 illustrates the main steps of the full process. To implement our problem we use *clingo*, which consists of the grounder *gringo* and the solver *clasp*, included in Potassco, the Potsdam Answer Set Solving Collection [GKK+11]. Therefore, our explanations about ASP syntax and semantics are based on the language used by *clingo*. In the following, we give a basic introduction to the ASP encoding that we use in the rest of the section. More details about ASP and its syntax and semantics can be found in [GKKS12, Lif19].

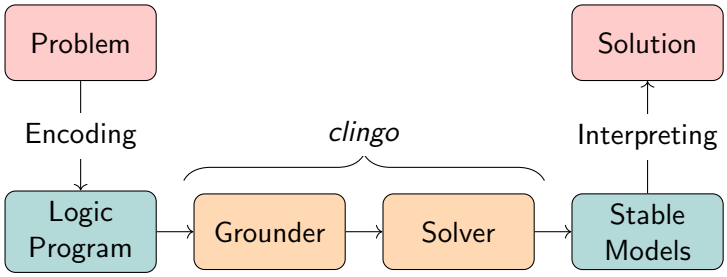


Figure 3.5: Main steps of the ASP solving process [GKKS12].

A *logic program* P over a set of atoms or propositional variables is defined as a finite set of logic rules, which are usually of the form

$$\texttt{b :- a}_1\texttt{, ..., a}_m\texttt{, not a}_{m+1}\texttt{, ..., not a}_n \tag{3.1}$$

The left part of a rule r is called *head* ($h(r) = \texttt{b}$) and the right part *body* ($B(r) = \texttt{a}_1\texttt{, ..., a}_m\texttt{, not a}_{m+1}\texttt{, ..., not a}_n$). The symbol `:-` that separates the head and the body of a rule denotes implication. Thus, a rule can be understood as the statement: the *head*

is true if the *body* is true. A rule that has no body is called a *fact*, since it is always true. A rule with an empty head represents an *integrity constraint*, since the body must always be false. Facts and integrity constraints are very useful when describing the control problem. For instance, we use facts to represent the Boolean function and the control goals and integrity constraints to guarantee that the interventions that are considered are consistent (see Section 3.3.2). To ease the encoding, it is common to use variables. Variables can be used to describe generic rules that can be applied to many different literals. For example, `a(T) :- b(T)` denotes that for any value of `T`, `a(T)` is true if `b(T)` is true. This allows us to simplify the encoding, since it is enough to write the conditions that must be satisfied by every component only once in terms of `V`.

Simple rules like (3.1) are built by the conjunction or disjunction of several *literals*, which are (possibly negated) propositional variables or *atoms* ($\mathtt{a}_i$). More complex rules can also include further language constructs such as *conditional literals* or *cardinality constraints*. A conditional literal is of the form

$$\mathtt{a} : \ \mathtt{b}_1, \ldots, \mathtt{b}_n$$

for $n \geq 0$, where $\mathtt{a}, \mathtt{b}_i$ are literals and the symbol `:` denotes implication. A cardinality constraint is expressed as

$$\mathtt{l} \ \{ \ \mathtt{c}_1; \ \ldots; \ \mathtt{c}_n \} \ \mathtt{u}$$

where $\mathtt{l}, \mathtt{u} \in \mathbb{N}$ and each c_i is a literal. The numbers `l`,`u` are the lower and upper bounds respectively of the number of literals that are satisfied in the cardinality constraint. We use cardinality constraints in our control problem to limit the size of the control strategies. Furthermore, rules can also include aggregates and optimization options, such as minimization or maximization on the number of satisfied literals, which could be used to give different weights to different potential interventions. For more details about ASP syntax and semantics we refer the reader to [GKKS12, Lif19].

A model of a logic program P is an assignment mapping variables to truth values that satisfies the set of rules of P. A *stable model* is a model in which for every atom assigned to true there exists a rule r in P such that $h(r)$ is true and $B(r)$ is satisfied by the assignment. Thus, every true literal of a stable model is supported by at least one rule. Every set of interventions whose literals are assigned to true in a solution (stable model) of the control problem is a control strategy. Furthermore, we run the control problem using a minimization option that allows us to obtain minimal solutions with respect to inclusion.

3.3.2 Problem encoding

In this section we explain in more detail the ASP encoding used for control strategy identification. We recall the ASP implementation introduced in [KSSV13] and extended in

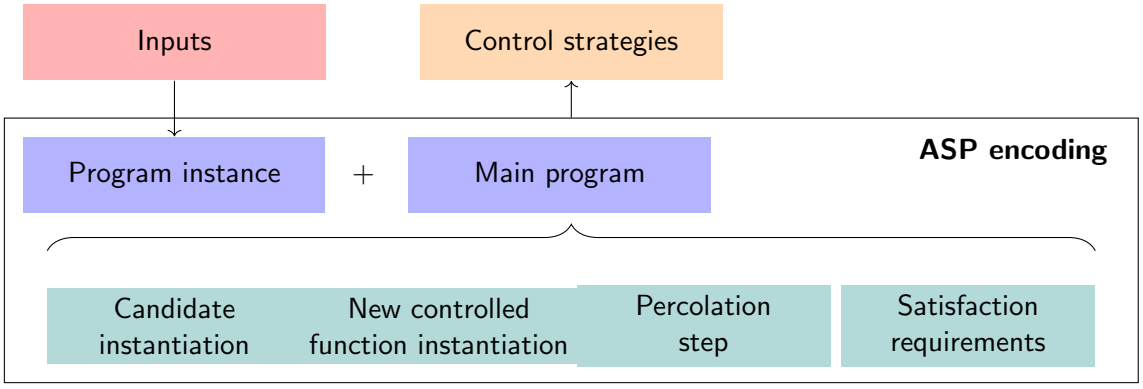


Figure 3.6: Diagram showing the main blocks of the ASP encoding, consisting of two main blocks: program instance and main program. The inputs (Boolean function in DNF form, target subspaces, selected trap spaces, list of forbidden interventions and limit size of the control strategies) are used to create the program instance. The main program is divided in four code blocks, marked in green.

[CFTS22a] and [CFTS22c] to identify control strategies by direct percolation and via trap spaces using node and edge control [CFTS20, CFTS22c] and describe how it can also identify transient control strategies.

The ASP encoding consists of two different parts: the encoding of the control problem (program instance) and the encoding of the computation process (main program). The program instance is created using the inputs of the control problem: the Boolean function, assumed to be given in disjunctive normal form, the target subspaces and selected trap spaces, the limit size of the control strategies and the restrictions on the nodes and edges that can be used for control. The main program is divided in four blocks: candidate instantiation, new controlled function instantiation (only necessary for edge interventions), percolation step and satisfaction requirements. A diagram representing the main blocks of the ASP encoding is shown in Figure 3.6.

The Boolean network from the example in Figure 3.7 is encoded as follows. The literal `formula` (line 1) links every variable with its DNF, described by the literals `dnf` and `clause` (lines 2-9). The regulatory function of the first component $f_1(x) = x_2\bar{x}_3 \vee x_1x_2 \vee x_1x_3$ is declared in the literal `formula(x1,0)` (line 1) and linked to its three clauses `dnf(0,0)`, `dnf(0,1)` and `dnf(0,2)` (line 2). The first clause $x_2\bar{x}_3$ is encoded in the literals `clause(0,x2,1)` and `clause(0,x3,-1)` (line 4). Note that we use `-1` and `1` in the third variable of the literal `clause` to denote whether a variable is negated or not, respectively. To ease the encoding, we also use the value `-1` to represent the Boolean value 0 in the rest of the program. Note that all the rules of the program instance are *facts* that describe the control problem.

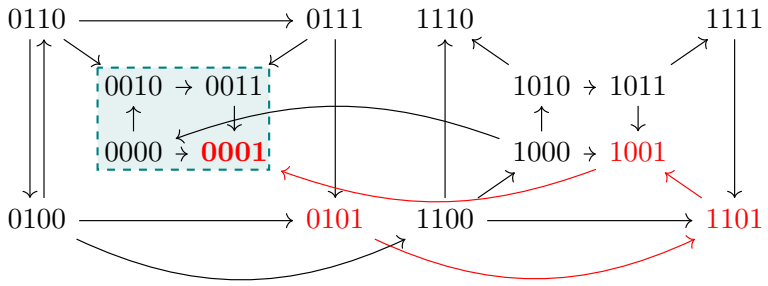


Figure 3.7: The asynchronous dynamics of the Boolean function $f(x) = (x_2\bar{x}_3 \vee x_1x_2 \vee x_1x_3,\ x_1x_3 \vee \bar{x}_1x_2\bar{x}_3,\ x_1\bar{x}_4 \vee \bar{x}_2\bar{x}_4 \vee \bar{x}_3\bar{x}_4,\ \bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3 \vee x_4)$ has two attractors $\mathcal{A}_1 = \{0001\}$ and $\mathcal{A}_2 = \{1110\}$. The selected trap spaces of f containing only attractors in the target subspace $P = 00{*}{*}$ (teal) are $T_1 = {*}{*}01$ (red) and $T_2 = 0001$ (the first steady state, bold red). $\mathbb{B}^n$ percolates to the selected trap space T_1 under the controlled functions $f^{\mathcal{N}_1}(x) = (x_1x_2 \vee x_2,\ \bar{x}_1x_2,\ 0,\ 1)$, with $N_1 = \{(3, 0)\}$, and $f^{\mathcal{N}_2}(x) = (x_2\bar{x}_3 \vee x_1x_2 \vee x_1x_3,\ x_1x_3 \vee \bar{x}_1x_2\bar{x}_3,\ 0,\ 1)$, with $N_2 = \{(4, 1)\}$.

```
1 formula(x1,0). formula(x2,1). formula(x3,2). formula(x4,3).
2 dnf(0,0). dnf(0,1). dnf(0,2). dnf(1,3). dnf(1,4). dnf(2,5).
3 dnf(2,6). dnf(2,7). dnf(3,8). dnf(3,9). dnf(3,10). dnf(3,11).
4 clause(0,x2,1). clause(0,x3,-1). clause(1,x1,1). clause(1,x2,1).
5 clause(2,x1,1). clause(2,x3,1). clause(3,x1,1). clause(3,x3,1).
6 clause(4,x1,-1). clause(4,x2,1). clause(4,x3,-1). clause(5,x1,1).
7 clause(5,x4,-1). clause(6,x2,-1). clause(6,x4,-1). clause(7,x3,-1).
8 clause(7,x4,-1). clause(8,x1,-1). clause(9,x2,-1). clause(10,x3,-1).
9 clause(11,x4,1).
```

Listing 3.1: Program instance: Boolean function.

We allow the possibility of excluding certain interventions from the control candidates, for instance in the case that an intervention is not feasible for application. The node and edge interventions that we want to exclude from the control are declared in the literals `avoid_node` or `avoid_edge` respectively (line 10). The target subspace and the target trap spaces are encoded in the literal `subspace` (line 12). We use two types of identifier: positive and negative. The sign of the subspace identifier marks whether we need to impose restrictions on the control interventions when the system percolates to that subspace. For example, when using a method for permanent control (direct percolation, via trap spaces or combined), the positive identifier is used if the subspace is a selected trap space since, by Proposition 3.1.2, the control interventions need to target a component fixed in the selected trap space, whereas the negative identifier is used if the subspace is the direct target since Proposition 3.1.2 does not impose any condition on the control interventions. When using the method for transient control, the direct target is not declared and the

selected trap spaces are marked with the negative identifier, since by Proposition 3.2.1 there are no restrictions on the control interventions. The fixed variables of each subspace are encoded in the variable `goal` (lines 13-14). The selected trap space $**01$ from Figure 3.7 is encoded by the variables `subspace(1)`, `goal(1,x3,-1)` and `goal(1,x4,1)` (lines 12-13). Note that for transient control, lines 12-14 would be substituted by the literals `subspace(-1)`, `subspace(-2)`, `goal(-1,x3,-1)`, `goal(-1,x4,1)`, `goal(-2,x1,-1)`, `goal(-2,x2,-1)`, `goal(-2,x3,-1)`, `goal(-2,x4,1)`. In fact, the program encoding for transient control is the same as the program encoding for direct percolation but marking with a negative identifier all the selected trap spaces and removing the target subspace. Finally, a limit size on the number of interventions is set in line 16.

```
10 avoid_node(x1). avoid_node(x2). avoid_edge(x1, x1). avoid_edge(x2, x2).
11
12 subspace(-1). subspace(1). subspace(2).
13 goal(-1,x1,-1). goal(-1,x2,-1). goal(1,x3,-1). goal(1,x4,1).
14 goal(2,x1,-1). goal(2,x2,-1). goal(2,x3,-1). goal(2,x4,1).
15
16 #const maxsize=2.
```

Listing 3.2: Program instance: forbidden interventions, target subspace, selected trap spaces and limit size.

As mentioned above, the main ASP program for control strategy identification is divided in four parts. The first two (candidate instantiation and new controlled function instantiation) differ in node and edge control whereas the last two (percolation step and satisfaction requirements) are the same. In the following, we describe each step in detail. To simplify the explanation, we describe the method for permanent control that combines direct percolation and percolation via trap spaces. The ASP program for transient control is the same as for direct percolation, differing only on the program instance as explained in the previous paragraph.

The candidate instantiation for node control is encoded as follows. The literal `node(V,S)` is used to denote the node intervention that fixes the node `V` to the value 0 (`S` = -1) or 1 (`S` = 1). Lines 1-5 in Listing 3.3 instantiate the literals `node(V,S)` for direct percolation. Note that, in order to reduce the number of candidates, a candidate node intervention `node(V,S)` for direct percolation is instantiated only when there is a positive (respectively negative) path in the IG from the node `V` to one of the nodes fixed in the target subspace to the value `S` (respectively `-S`). The variables that are part of such paths in the IG are tracked by the literal `closure(V,T)` that becomes true when the variable `V` is in the path with the correct sign `T`. This reduction method is explained in more detail in the second part of Section 4.4.2 (see [KSSV13] and [SKK10] for a full explanation). The conditions `satisfied(Z), Z < 0` are added to the rule in line 5 to guarantee that these node interventions are only instantiated when the state space percolates to the target, marked with a negative identifier. The variable `Z` is used to identify which subspace is satisfied by the set

of interventions and allows us to encode different conditions depending on the sign of `Z`: the requirements for control strategies by direct percolation when `Z < 0` and the requirements for control strategies via trap spaces when `Z > 0`.

Line 6 instantiates the candidate interventions for control via trap spaces. When the state space percolates to a selected trap space, it is necessary that the candidate interventions are chosen among the variables fixed in the trap space, as required in Proposition 3.1.2 and ensured in line 6. In this case, the conditions `satisfied(Z), Z > 0` are added in line 6 to guarantee that these node interventions are only instantiated when the state space percolates to the selected trap space identified by `Z`. The instantiations of the fixed variables of the selected trap space $T = **01$ from the example in Figure 3.7 are `goal(1,x3,-1)` and `goal(1,x4,1)`. Consequently, the node interventions considered for control via this trap space are `node(x3, -1)` and `node(x4,1)`. Line 7 excludes contradictory node interventions, that is, two node interventions fixing the same node to opposite values. In line 8, the variable `node(V)` is declared to keep track of the controlled nodes.

```
1 goal(T,S) :- goal(Z,T,S), Z < 0.
2 satisfy(V,W,S) :- f(W,D); dnf(D,C); cl(C,V,S).
3 closure(V,T)   :- goal(V,T).
4 closure(V,S*T) :- closure(W,T); satisfy(V,W,S); not goal(V,-S*T).
5 { node(V,S) : closure(V,S), not avoid_node(V), satisfied(Z), Z < 0 }.
6 { node(V,S) : goal(Z,V,S), not avoid_node(V), satisfied(Z), Z >=0 }.
7 :- node(V,1); node(V,-1).
8 node(V) :- node(V,S).
```

Listing 3.3: Candidate instantiation: node interventions.

The candidate instantiation for edge control is shown below. The candidate edge interventions are generated in lines 9-13. The literal `edge(Vi,Vj,S)` is used to denote the edge intervention that fixes the node `Vi` in the regulatory function of the node `Vj` to the value 0 (`S`=-1) or 1 (`S`=1). Note that the existence of a clause involving i in the DNF of f_j is required in order to instantiate the candidate edge intervention (i, j, c) and that forbidden edges are excluded. It is also ensured that, when the state space percolates to a selected trap space, the edge interventions target the variables fixed in the trap space, as stated in Proposition 3.1.2 (lines 11-13). Contradictory edge interventions are also excluded (line 15). We keep track of the controlled edges with the variable `edge(Vi,Vj)` (line 16).

```
9  { edge(Vi,Vj,1); edge(Vi,Vj,-1) } :- formula(Vj,D), dnf(D,C), clause(C, Vi, S),
10                                      not avoid_edge(Vi,Vj), satisfied(Z), Z < 0.
11 { edge(Vi,Vj,1); edge(Vi,Vj,-1) } :- formula(Vj,D), dnf(D,C), clause(C, Vi, S),
12                                      not avoid_edge(Vi,Vj), goal(Z,Vj,T),
13                                      satisfied(Z), subspace(Z), Z >= 0.
14
15 :- edge(Vi,Vj,1), edge(Vi,Vj,-1).
16 edge(Vi,Vj) :- edge(Vi,Vj,S).
```

Listing 3.4: Candidate instantiation: edge interventions.

When considering node and edge interventions together, the following restrictions are also added, to ensure that the set of interventions is consistent. They prevent that a node and an edge intervention fix or target the same variable (lines 17 and 18 respectively).

```
17 :- node(V), edge(V,Vj).
18 :- node(V), edge(Vi,V).
```

Listing 3.5: Candidate instantiation: consistent requirements for node and edge interventions.

In order to capture the effect of the edge interventions in the Boolean function, a new set of literals `new_clause`, `new_dnf`, `new_formula` is instantiated to represent the DNF of the resulting regulatory functions (lines 19-23). These new literals are directly instantiated for every term, clause and DNF respectively that are not affected by edge interventions. When a clause is affected by an edge intervention, there are two possible situations. The edge intervention fixes the term to 1, in which case the term just disappears from the clause, or it fixes the term to 0, in which case the whole clause is evaluated to 0. In the first situation, the term is just not included in the new clause, that is, the corresponding `new_clause` literal is not instantiated. In the second situation, the clause needs to be removed from the DNF so the literal `remove_dnf` is instantiated to indicate that this clause should be removed (line 20). Line 21 links all the clauses that do not need to be removed to the new DNF. Note that if all the terms of a clause get fixed to 1, the whole DNF is set to 1. In this case, the literal `remove_formula` is instantiated (line 22) so that the `new_formula` literal is not generated, since the regulatory function is constant. To indicate that the component `V` is set to 1, the literal `fixed_node(V,1)` is also instantiated (line 25). If all the clauses of a DNF are evaluated to 0 and consequently removed from the disjunction, the regulatory function becomes the constant 0. Then, the literal `new_formula` is not instantiated and the literal `fixed_node(V,-1)` is generated to indicate that the component `V` is set to 0 (line 26). When none of the non-empty DNF clauses are evaluated to one, the literal `new_formula` is instantiated to link them to the corresponding regulatory function (line 22).

```
19 new_clause(C,V,S) :- clause(C,V,S); dnf(D,C); formula(Vj,D); not edge(V,Vj).
20 remove_dnf(D,C):- clause(C,Vi,-S); edge(Vi,Vj,S); dnf(D,C); formula(Vj,D).
21 new_dnf(D,C) :- new_clause(C,Vi,S); dnf(D,C); formula(Vj,D); not remove_dnf(D,C).
22 remove_formula(Vj,D) :- dnf(D,C); formula(Vj,D); edge(Vi,Vj,S) : clause(C,Vi,S).
23 new_formula(V,D) :- new_dnf(D,C); formula(V,D); not remove_formula(V,D).
24
25 fixed_node(V,1) :- remove_formula(V,D).
26 fixed_node(V,-1) :- not remove_formula(V,D); not new_formula(V,D); formula(V,D).
```

Listing 3.6: New controlled function instantiation, only required for edge control.

Let us consider the case of the edge intervention `edge(x3,x4,1)`, that fixes $x_3 = 1$ in f_4. Since $f_4(x) = \bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3 \vee x_4$, fixing $x_3 = 1$ leads to the clause $\bar{x}_3$ being evaluated to 0 and the regulatory function becomes $f_4^{(3,4,1)}(x) = \bar{x}_1 \vee \bar{x}_2 \vee x_4$ (the same without the clause $\bar{x}_3$). The ASP program will instantiate the literals `new_clause(8, x1, -1), new_clause(9, x2, -1), new_clause(11, x4, 1)` since they are not affected by the edge intervention. It will also instantiate the literals `new_dnf(3,8), new_dnf(3,9), new_dnf(3,11)` that connect the clauses to the new DNF of f_4 and `remove_dnf(3,10)`, since the edge intervention `edge(x3,x4,1)` and the term in the clause `clause(10,x3,-1)` have opposite signs. Although `new_dnf(3,10)` will not be instantiated, `new_formula(x4,3)` will, since its DNF will still have the non-trivial clauses `new_dnf(3,8), new_dnf(3,9), new_dnf(3,11)`.

If instead the edge intervention was `edge(x3,x4,-1)`, fixing $x_3 = 0$ in f_4 would set the regulatory function to 1 ($f_4^{(3,4,0)}(x) = 1$). In this case, `remove_formula(x4,3)` would also be instantiated, since there would be a clause in the DNF that gets evaluated to 1, and `new_formula(x4,3)` will not, since the DNF would become 1.

The regulatory functions that become constants either through node or edge control are captured in the literals `intervention(V,S)` (lines 27 and 28 respectively) and tracked by the literal `intervention(V)` (line 29).

```
27 intervention(V,S) :- node(V,S).
28 intervention(V,S) :- not node(V,S), not node(V,-S), fixed_node(V,S).
29 intervention(V) :- intervention(V,S).
```

Listing 3.7: Setting intervention literals for the percolation step.

The percolation effect is encoded in the same way as described in [KSSV13] (lines 30-35). First, the literal `eval_formula(Z,V,S)` is instantiated for every component that gets fixed to a constant value by an intervention (line 30). The DNFs of the remaining regulatory functions are marked as free using the literal `free(Z,V,D)` (line 31). Then for each clause, the literal `eval_clause(Z,V,-1)` is instantiated if the clause gets fixed to 0 (line 32). The literal `eval_formula(Z,V,1)` is declared for all the components whose DNF is fixed to 1, which happens if all the terms of a DNF clause get fixed to their corresponding values (lines 33-34). When all the DNF clauses are set to 0, the literal `eval_formula(Z,V,-1)` is declared to indicate that the component is fixed to 0 (line 35). See [KSSV13] for more details.

```
30 eval_formula(Z,V,S) :- subspace(Z); intervention(V,S).
31 free(Z,V,D) :- new_formula(V,D); subspace(Z); not intervention(V).
32 eval_clause(Z,C,-1) :- new_clause(C,V,S); eval_formula(Z,V,-S).
33 eval_formula(Z,V,1) :- free(Z,V,D); eval_formula(Z,W,T) : new_clause(C,W,T);
34                       new_dnf(D,C).
35 eval_formula(Z,V,-1) :- free(Z,V,D); eval_clause(Z,C,-1) : new_dnf(D,C).
```

Listing 3.8: Percolation step.

Finally, it is ensured that a candidate subspace is a control strategy by requiring that at least one subspace constraint is satisfied (lines 36-38). A subspace constraint is satisfied when all the variables that are fixed in the subspace have their regulatory functions fixed after the percolation step. This is checked in line 37 by stating that a subspace `Z` is satisfied if for all `goal(Z,T,S)` the regulatory corresponding to component `T` has been fixed to `S`, that is, if the literal `eval_formula(Z,T,S)` is true. A limitation on the total number of interventions is also added (line 39).

```
36 not satisfied(Z) :- goal(Z,T,S), not eval_formula(Z,T,S), subspace(Z).
37 satisfied(Z) :- eval_formula(Z,T,S) : goal(Z,T,S); subspace(Z).
38 0 < { satisfied(Z) : subspace(Z) }.
39 :- maxsize > 0; maxsize + 1 { node(V,R); edge(Vi,Vj,S) }.
```

Listing 3.9: Satisfaction requirements.

3.3.3 Main algorithm

The complete algorithm for control strategy identification is shown in Algorithm 1. It takes as inputs the Boolean function f, the target subspace P, the type of control method *method*, the update D, the limit size for the control strategies k, the (possibly empty) list of forbidden interventions *avoid_intv* and, optionally, the list of attractors *attr* (line 1). The Boolean function, target subspace, selected trap spaces, limit size and list of forbidden interventions are used as input for the ASP program, which is called by *createCandidatesAndPercolate* (lines 3, 10, 12, 14) and returns the corresponding control strategies. Our ASP encoding takes as input a constant-free Boolean function. Therefore, if the network has constant coordinate functions, a preprocessing step takes place so that constant values are percolated and removed from the network. Note that this does not affect the attractors of the network, as explained in Section 2.1.

Algorithm 1 allows for the computation of control strategies by direct percolation (lines 2-3), via the trap spaces method (lines 9-10), using the two previous methods combined, meaning that both percolation to the target subspace and selected trap spaces is considered (lines 11-12) and transient (lines 13-14). When searching for control strategies via the trap spaces (permanent or transient), we distinguish two types of selected trap spaces: trap spaces contained in P (Type 1) (line 6) and trap spaces not contained in P but containing only attractors in P (Type 2) (line 8). Note that selected trap spaces of Type 2 are only identified when all the attractors are known or can be approximated by minimal trap spaces (line 7). Moreover, in order to avoid unnecessary calculations, we only consider non-percolating trap spaces, that is, trap spaces that do not percolate to smaller ones, as selected trap spaces since all the subspaces percolating to a trap space T also percolate to $F(f)(T)$. This step is included in the process of computing the selected trap spaces.

We implemented Algorithm 1 using PyBoolNet [KSS16], a Python package for the generation, modification and analysis of Boolean networks. PyBoolNet also provides an efficient computation of trap spaces for relatively large networks, which we use for the computation of the selected trap spaces, and a method to check whether the attractors of a Boolean network can be approximated by minimal trap spaces [KS15]. To solve the ASP problem, we use *clingo*, included in Potassco, the Potsdam Answer Set Solving Collection [GKK+11]. The source code of the implementation of Algorithm 1 is available at [CFa].

Algorithm 1 Control strategies for a target subspace

```
1:  function ControlStrategies(f, P, method, D, k, avoid_intv, attr)
2:      if method = "direct percolation" then:
3:          CS ← createCandidatesAndPercolate(f, P, −, k, avoid_intv)
4:      else:
5:          T ← trapSpaces(f)
6:          selTS ← selectedTrapSpacesType1(T, P)
7:          if attr ≠ ∅ then:
8:              selTS ← selTS + selectedTrapSpacesType2(T, P, D, attr)
9:      if method = "trap spaces" then:
10:         CS ← createCandidatesAndPercolate(f, −, selTS, k, avoid_intv)
11:     if method = "combined" then:
12:         CS ← createCandidatesAndPercolate(f, P, selTS, k, avoid_intv)
13:     if method = "transient" then:
14:         CS ← createCandidatesAndPercolate(f, selTS, −, k, avoid_intv)
15:     return CS
```

3.3.4 Further considerations: minimality and running times

Since we are interested in minimal intervention sets, the ASP program is run to return all the minimal sets of interventions with respect to inclusion that are control strategies by direct percolation and/or via trap spaces up to the chosen limit size. The output set of control strategies might vary depending on the method that is chosen and consequently there might be control strategies that are minimal by direct percolation and not minimal via trap spaces and vice versa. For instance, for the Boolean network of Figure 3.2 and the target $P = 110$, there is only one minimal control strategy for node control via trap spaces, the set $\mathcal{N}_1 = \{(3, 0)\}$. Direct percolation instead identifies the unique minimal control strategy $\mathcal{N}_2 = \{(1, 1), (3, 0)\}$. When computing the control strategies combining both methods, only $\mathcal{N}_1$ will be identified, since $\mathcal{N}_2$ is a superset of $\mathcal{N}_1$. For this reason, the

set of control strategies identified by the combined method is not necessarily equivalent to the union of the control strategies obtained by each method, even though each control strategy is identified either by direct percolation or via trap spaces. This is the case in the biological network analyzed in Section 3.4, where some of the control strategies identified by direct percolation are non-minimal and therefore not present as control strategies of the combined method since they are supersets of a smaller one.

The main factors influencing the running times of the approach are the size and complexity of the network (number of nodes, edges, prime implicants, etc.) and the number and size of the target subspaces and selected trap spaces. Assuming that there are no nodes nor edges to avoid, the amount of candidate interventions to choose from for direct percolation is twice the number of nodes $2n$ for node control, twice the number of edges $2m$ for edge control or twice the sum of both $2(n+m)$ for the mixed control. Thus, the total number of possible combinations grows exponentially with $2n$, $2m$ or $2(n+m)$ respectively. As mentioned during the encoding, the number of candidate interventions can be reduced by considering only a candidate node intervention (i, c) when there is a positive (respectively negative) path in the IG from i to one of the nodes fixed in the target subspace to the value c (respectively $\bar{c}$) [SKK10]. Similar reduction methods could also be studied for edge control. The efficiency of the ASP approach for node control was analyzed in [KSSV13], where the running times of different Boolean networks and different ASP solvers were studied, showing that the approach was able to deal with networks of 100-200 nodes within milliseconds. The number of edges could, in theory, be as large as n^2, however this is not usually the case in biological systems, which are often rather sparse. Although extending the candidate interventions to edges has an impact on the running times (see Section 3.4.3), the problem is still treatable for relatively large biological networks.

In the case of percolation via trap spaces, the number of candidate interventions might be reduced since, according to Proposition 3.1.2, the intervention candidates are required to target only variables fixed in the corresponding selected trap space. Consequently, the running times are also dependent on the number and size of the selected trap spaces. Note that when the selected trap space is a steady state, the number of candidate interventions is the same as in the method of direct percolation.

In the example of Figure 3.2, taking $P = 110$ as the target subspace, there are 8, 577 and 656 consistent candidate combinations for node, edge and mixed control respectively for the direct percolation method. When considering the method via trap spaces, since one of the selected trap spaces is a steady state ($T_1 = 110$), the amount of candidate interventions is the same. If only $T_2 = **0$ was considered as a selected trap space, the number of combinations of candidate interventions would be 2, 27 and 28 for node, edge and mixed control respectively, since they would only include node and edge interventions targeting the third component.

The last factor to take into account when analyzing the running times is the number of candidate subspaces, both in case of direct percolation (if the target is extended to multiple subspaces) and in the case of percolation via trap spaces when different selected trap spaces act as targets. The more target subspaces, the more likely that a candidate intervention satisfies the condition. However, increasing the number of target subspaces might also increase the number of candidate interventions. Moreover, in some cases, the combined method might require higher running times than the sum of the times required for direct percolation and via trap spaces individually (see results in Section 3.4).

3.4 Application

In this section, we study the applicability of our method to a cell fate decision model (MAPK network). We consider different targets and compare the control strategies obtained by direct percolation and via trap spaces, as well as transient control, for the asynchronous update. We analyze the results first for node and then for edge control. In all the cases, we obtain new control strategies via trap spaces missed by direct percolation.

The network analyzed in this case study was introduced by Grieco et al. (2013) [GCBP$^+$13] to model the effect of the Mitogen-Activated Protein Kinase (MAPK) pathway on cell fate decisions in bladder cancer cells (see Figure 3.8). The network consists of 53 Boolean variables, including the four inputs DNA-damage, EGFR-stimulus, FGFR3-stimulus and TGFBR-stimulus. The states of the three outputs of the network (Apoptosis, Growth-Arrest and Proliferation) indicate the enablement or disablement of the corresponding processes that represent the different cell fates or phenotypes considered in [GCBP$^+$13].

There are 18 attractors in the asynchronous dynamics, of which 12 are steady states and 6 are complex. The attractors are in one-to-one correspondence with the minimal trap spaces, that is, each attractor is contained in a minimal trap space and each minimal trap space only contains one attractor [KS15]. Therefore, we can use the selected trap spaces of Type 1 and Type 2 (see Section 3.3.3) to search for control strategies via trap spaces.

We start by targeting the subspace defined by the apoptotic phenotype and compare the control strategies identified via trap spaces to the ones by direct percolation and to the transient control strategies, first for node control and then for edge control. In the second part, we consider the attractors of the asynchronous dynamics by targeting the minimal trap spaces. We compare the control strategies identified by direct percolation, via trap spaces, the combination of the two methods and by transient control (see Algorithm 1) for four steady states for node and edge control. We end by discussing the running times of our approach. All the results presented in this section were obtained with a regular desktop 8-processor computer, Intel®Core™ i7-2600 CPU at 3.40GHz, 16GB memory.

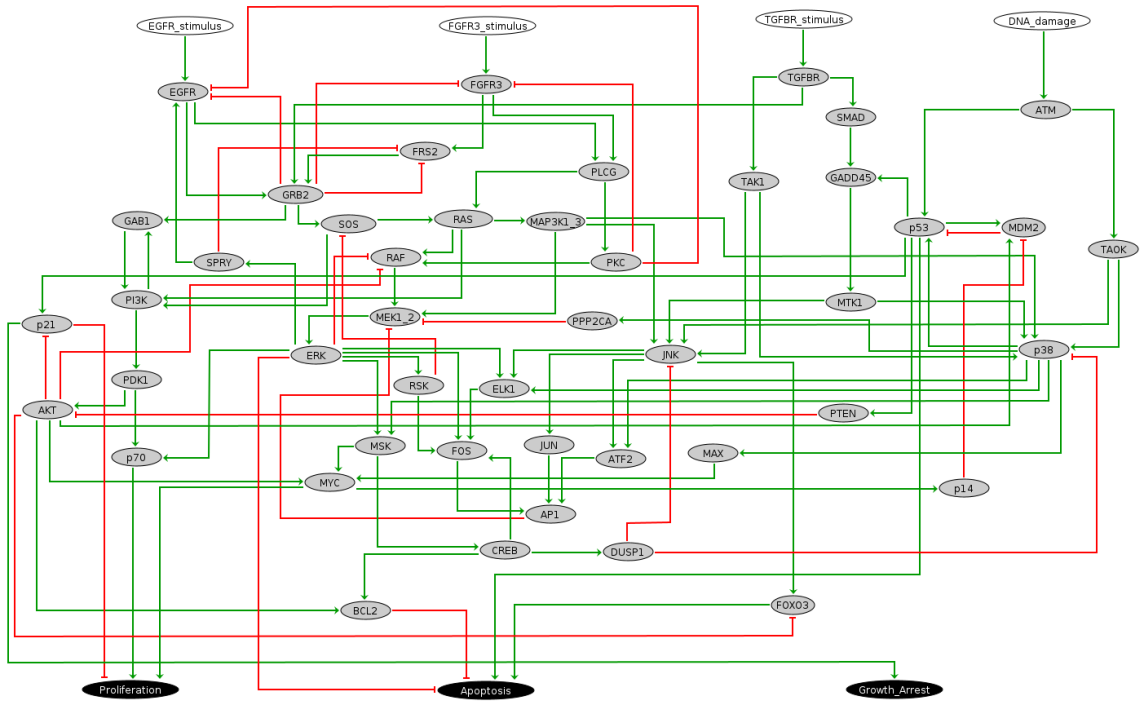


Figure 3.8: MAPK network presented in [GCBP+13]. Figure obtained using GINsim [CNT12]. Input and output nodes are colored in white and black respectively. Green edges and red edges denote activation and inhibition respectively.

3.4.1 Target: apoptotic phenotype

We start by considering as target the apoptotic phenotype that is defined by the subspace obtained by fixing the output nodes (Apoptosis to 1, Growth-Arrest to 1 and Proliferation to 0) as in [GCBP+13]. We refer to this subspace as the apoptotic target. We identify 103 non-percolating selected trap spaces. We set a limit size of three interventions, since this size is large enough to obtain relevant control strategies with all the methods.

In this setting, the combined method identifies 271 control strategies, minimal with respect to inclusion, for node control up to size 3: three of size 1, 106 of size 2, 162 of size 3. The three control strategies of size 1 are {(TGFBR-stimulus, 1)}, {(TGFBR, 1)} and {(DNA-damage, 1)}, the last one being obtained only via trap spaces. Under the control strategy {(DNA-damage, 1)}, the state space percolates to the trap space {ATM = 1, DNA-damage = 1, TAOK = 1}, which contains only attractors in the apoptotic target. This minimal control strategy is not identified by direct percolation. The number of control strategies identified by each method is shown in Table 3.1. All the transient control

strategies identified for the apoptotic target are also obtained via trap spaces. As explained in Section 3.3.4, the list of control strategies obtained by the combination of the two methods might not be equal to the union of the control strategies obtained by each of the methods individually, since minimality is applied to each type of control strategy separately, and a control strategy that is minimal for one method might not be minimal under another method. In this case, there are eighteen control strategies of size 2 and thirteen of size 3 obtained by direct percolation that are supersets of the control strategy {(DNA-damage, 1)}. For example, {(DNA-damage, 1), (SMAD, 1)} is identified as a control strategy by the direct percolation method since neither {(DNA-damage, 1)} nor {(SMAD, 1)} are control strategies by direct percolation. When considering the combination of both methods, {(DNA-damage, 1)} is identified as a control strategy and consequently neither {(DNA-damage, 1), (SMAD, 1)} nor any of the supersets of {(DNA-damage, 1)} are considered. For this reason, there are fewer control strategies of size 2 and 3 for the combined method than by direct percolation (see Table 3.1). The method of direct percolation identifies many control strategies of size 2 and 3 that are missed by the method via trap spaces. These missed strategies fall into two categories. The first category includes the strategies that are not considered by the trap spaces method, since they include interventions targeting nodes not fixed in any selected trap space or fix them to the opposite value. For instance, the intervention set {(DUSP1, 0), (PLCG,1)} is not taken into consideration since the node DUSP1 is not fixed to 0 in any of the selected trap spaces. The second category includes the sets of interventions that percolate directly to the target subspace but do not percolate to any of the selected trap spaces. For these reasons, it is useful to combine the method of direct percolation with the method via trap spaces to increase the total number of control strategies identified (see Section 3.1 for more details).

Using the combined method for edge control we obtain 950 control strategies up to size 3: three of size 1, 117 of size 2 and 830 of size 3 (see Table 3.1). As for node control, all the transient control strategies are also identified via trap spaces. The three edge control strategies of size 1 are equivalent to the node interventions identified as control strategies of size 1. This results from the three variables involved in the size 1 node control strategies having a unique incoming edge. For example (TGFBR-stimulus, TGFBR, 1) has exactly the same effect as (TGFBR, 1), since TGFBR is uniquely regulated by TGFBR-stimulus.

In other cases, edge control allows intervention strategies that would be too restrictive in node control. For example, the two edge interventions (MAP3K1-3, p38, 1) and (MSK, CREB, 0), which fix the activation of MAP3K1-3 on p38 and the inhibition of MSK on CREB, lead the controlled system to percolate to the apoptotic target. However, fixing MAP3K1-3 to 1 and MSK to 0 does not, since the controlled system displays non-apoptotic steady states, which are not present in the original dynamics.

Table 3.1: Number and size of the control strategies identified by the different methods up to size 3 for the apoptotic target. Note that some minimal control strategies of size 2 and size 3 for direct percolation are not minimal for the combined method.

Node control	$\lvert\mathcal{N}\rvert = 1$	$\lvert\mathcal{N}\rvert = 2$	$\lvert\mathcal{N}\rvert = 3$
By direct percolation	2	124	175
Via trap spaces	2	0	0
Combined	3	106	162
Transient	2	0	0

Edge control	$\lvert\mathcal{E}\rvert = 1$	$\lvert\mathcal{E}\rvert = 2$	$\lvert\mathcal{E}\rvert = 3$
By direct percolation	2	137	893
Via trap spaces	2	0	0
Combined	3	117	830
Transient	2	0	0

Node and edge control	$\lvert\mathcal{C}\rvert = 1$	$\lvert\mathcal{C}\rvert = 2$	$\lvert\mathcal{C}\rvert = 3$
By direct percolation	4	530	3569
Via trap spaces	4	0	0
Combined	6	454	3299
Transient	4	0	0

By allowing the combination of node and edge interventions, the combined method identifies over three thousand control strategies up to size 3, as shown in Table 3.1. Note that these include all the control strategies obtained for node and edge control. In particular, the six control strategies of size 1 correspond to the three control strategies of node control and the three of edge control.

We observe that there are many control strategies that mix node and edge interventions. Most of them include interventions already appearing in control strategies consisting exclusively of node interventions or of edge interventions. In some cases, we find mixed control strategies that are equivalent to a node control strategy or an edge control strategy where a node intervention is substituted by an equivalent edge intervention or vice versa. For example, the control strategy {(CREB, DUSP1, 0), (TAOK, 1)} is equivalent to the control strategy {(CREB, DUSP1, 0), (ATM, TAOK, 1)}, since the node intervention (TAOK, 1) is equivalent to the edge intervention (ATM, TAOK, 1). There are also control strategies involving interventions that are not part of any node or edge control strategy. This is the case for {(FGFR3, FRS2, 1), (GRB2, FRS2, 0), (p38, 1)}, where neither (FGFR3, FRS2, 1) nor (GRB2, FRS2, 0) appear in any edge control strategy.

3.4.2 Target: minimal trap spaces

When computing control strategies for the minimal trap spaces, the input components need to be fixed in order to ensure that their value matches the one fixed in the target. Since each input combination identifies a separate trap space, there is at least one attractor per input combination. There are sixteen possible input combinations, fourteen of which identify subspaces that contain a unique attractor. These input combinations therefore give minimal node control strategies for the corresponding attractors. The subspaces induced by the two remaining input combinations ($\mathcal{S}_1$ = {EGFR-stimulus = 0, FGFR3-stimulus = 0, TGFBR-stimulus = 0 and DNA-damage = 0}, $\mathcal{S}_2$ = {EGFR-stimulus = 0, FGFR3-stimulus = 0, TGFBR-stimulus = 0 and DNA-damage = 1}) contain two steady states each and, therefore, further control interventions are needed. Table 3.2 shows the number and size of the control strategies up to size 7 (the number of inputs plus three) of these four steady states for node, edge and mixed control. Note that in all the cases there are control strategies identified via trap spaces not captured by direct percolation and there is no minimal control strategy identified by direct percolation missed via trap spaces. In fact, since by definition every steady state is a selected trap space, when the target is a steady state, all the control strategies identified by direct percolation are also identified via trap spaces. Moreover, no control strategy of size 5 is found for direct percolation for any of the steady states.

As in the case of the apoptotic target, there are edge control strategies allowing interventions that would not be possible using only node control. For example fixing the component GRB2 either to 0 or to 1, together with the corresponding input interventions, does not lead to a system with s_1 as the unique attractor. However, fixing GRB2 in the edge intervention (GRB2, GAB1, 1) in addition to the input interventions leads to a controlled dynamics that has s_1 as a unique attractor.

When considering mixed interventions, in contrast to the apoptotic target case, all the interventions appearing in minimal control strategies also occur in some strategy composed exclusively of node or exclusively of edge interventions. As can be seen from the numbers in Table 3.2, we still gain many mixed control strategies and thus more flexibility for choosing interventions that are both realizable in the lab and as non-invasive as possible for the system.

In this scenario, the transient control approach is able to identify new control strategies for all four steady states. In particular, since all the control strategies via trap spaces are transient control strategies, when the target is a steady state, transient control identifies all the control strategies found by the combined method of percolation and trap spaces. The three transient edge control strategies for the steady state s_1 that are not identified by any other method consist of four edge interventions fixing the inputs, as expected, plus an edge intervention targeting the node GRB2. This node is not fixed in the selected trap space which the controlled system percolates to. For this reason, the method via

Table 3.2: Number and size of the control strategies identified by the different methods up to size 7 for the different steady states. Note that there is no control strategy up to size 4. s_1 and s_1' denote the two steady states in $\mathcal{S}_1$ and s_2 and s_2' the two steady states in $\mathcal{S}_2$.

Node control	s_1			s_1'			s_2			s_2'		
$\lvert\mathcal{N}\rvert$	5	6	7	5	6	7	5	6	7	5	6	7
By direct percolation	0	0	60	0	0	32	0	14	2	0	14	2
Via trap spaces	2	0	0	0	8	12	0	22	14	2	0	0
Combined	2	0	0	0	8	12	0	22	14	2	0	0
Transient	2	0	0	0	8	12	0	28	16	2	0	0

Edge control	s_1			s_1'			s_2			s_2'		
$\lvert\mathcal{E}\rvert$	5	6	7	5	6	7	5	6	7	5	6	7
By direct percolation	0	0	150	0	0	84	0	22	58	0	33	50
Via trap spaces	3	1	0	0	14	20	0	36	64	3	1	0
Combined	3	1	0	0	14	20	0	36	64	3	1	0
Transient	6	11	157	0	16	168	0	50	72	6	33	40

Node and edge control	s_1			s_1'			s_2			s_2'		
$\lvert\mathcal{C}\rvert$	5	6	7	5	6	7	5	6	7	5	6	7
By direct percolation	0	0	12720	0	0	7040	0	1168	2608	0	1440	2048
Via trap spaces	80	16	0	0	704	2112	0	1872	4192	80	16	0
Combined	80	16	0	0	704	2112	0	1872	4192	880	16	0
Transient	128	224	3512	0	768	8960	0	2512	4832	128	736	1080

trap spaces cannot identify any of these strategies. This example shows how considering transient interventions can widen the possibilities for control.

3.4.3 Running times

The running times of the control strategy computation for each method, target and type of control are shown in Table 3.3. These refer to the total times needed for Algorithm 1 to terminate for each method, including the computation of the selected trap spaces when needed. We can observe how the number of candidate interventions affects the time required for each method. Node control is the fastest, with running times in the order of milliseconds, whereas edge control requires a few seconds. In most of the cases, the running times of the different methods vary from a few seconds to a few minutes when combining the two types of interventions. We observe that steady states with three selected trap spaces have slightly higher running times than steady states with two selected trap spaces. The apoptotic phenotype is the target with the highest number of selected trap spaces (over one hundred). However, we do not observe a significant increase of the running time with respect to the steady states. This could result from the additional constraint on

candidate interventions that is imposed when working with selected trap spaces, requiring the interventions to be selected among the variables fixed in the trap space.

Table 3.3: Running times (in seconds) for the control strategy computation targeting the apoptotic phenotype and the four steady states in the MAPK network. The size of the target is defined as the number of fixed components in the target subspace.

Target	**Size of the target**	**Number of selected trap spaces**	**Method**	**Time (s)**		
				Node	**Edge**	**Both**
Apoptotic phenotype	3	103	By direct percolation	0.18	7.05	280.18
			Via trap spaces	0.47	0.82	0.74
			Combined	7.15	117.36	461.37
			Transient	1.13	1.42	1.13
Steady state s_1	53	3	By direct percolation	0.04	8.41	125.65
			Via trap spaces	0.07	7.39	289.18
			Combined	0.08	17.07	149.47
			Transient	0.08	35.00	138.62
Steady state s'_1	53	3	By direct percolation	0.04	1.61	154.79
			Via trap spaces	0.09	1.40	104.80
			Combined	0.15	4.69	211.49
			Transient	0.09	5.48	303.55
Steady state s_2	53	2	By direct percolation	0.03	0.42	10.71
			Via trap spaces	0.10	1.03	33.33
			Combined	0.10	3.26	155.31
			Transient	0.10	3.72	198.61
Steady state s'_2	53	2	Direct percolation	0.03	1.25	12.29
			Via trap spaces	0.10	1.87	6.56
			Combined	0.11	3.60	12.31
			Transient	0.10	8.44	55.22

3.4.4 Other updates

This case study focuses on the asynchronous update, since it is the one used for modeling the MAPK network in [GCBP$^+$13]. Here we add some remarks about the results for this network considering the other two updates: synchronous and generalized asynchronous. As mentioned in Section 3.1, the control strategies obtained by direct percolation are independent of the update and consequently they are the same in any dynamics, whereas the control strategies via trap spaces might vary when the selected trap spaces differ from one update to the another.

In the synchronous dynamics, the MAPK network is not complete, that is, minimal trap spaces are not good approximations of attractors. In such cases, our approach does not select trap spaces of type 2, unless the full list of complex attractors is provided. Therefore, all the control strategies identified via trap spaces are also identified by direct percolation

and no new control strategies are uncovered.

In the generalized asynchronous dynamics, the MAPK network is complete, as in the asynchronous case. Since the trap spaces of a Boolean function are the same in all the updates, the selected trap spaces are also the same as in the asynchronous dynamics. Consequently, the control strategies identified via trap spaces for the generalized asynchronous update correspond to the ones identified for the asynchronous update.

3.5 Discussion

In this chapter, we presented a method for control strategy identification based on value percolation that uses trap spaces in order to uncover potentially smaller control strategies. This method allows for node interventions, acting on specific components, as well as edge interventions, acting on the interactions between them. The use of edge interventions can widen the possibilities for potential applications in the context of biological systems, as shown in Section 3.4.

The implementation of the method is based on Answer Set Programming (ASP), extending the works from [KSSV13], which helps us to deal with the problem of the combinatorial explosion linked to the control strategy computation. Although our approach is able to handle state-of-the-art biological models (Section 3.4.3), an exhaustive analysis of the factors affecting the running times such as the size of the network, number of edges or complexity of the Boolean functions would be needed.

Although in general control strategies by direct percolation require permanent interventions, control strategies via trap spaces allow the possibility of releasing the control once the selected trap space has been reached. This approach is adapted to identify transient control strategies, by using selected trap spaces as intermediate subsets. In this work, we study transient control as a specific case of sequential control of two steps: applying control and releasing it, using trap spaces as intermediate steps. This use of trap spaces could be extended to identify potential longer control sequences.

Chapter 4

Exhaustive approach

As explained in the previous chapter, approaches based on value percolation can be implemented efficiently but are not able to identify all possible control strategies for a given target. Although percolation-based control strategies might be enough for simple analysis of the network, an exhaustive identification of all possible controls would increase the diversity of strategies and thus could uncover potentially more relevant strategies for application.

Moreover, approaches developed for control strategy identification in Boolean networks often focus on either attractor control, aiming at a specific state of the system, or target control, aming at a certain subspace or phenotype. However, in some cases it might be interesting to be able to deal with more generic control goals such as groups of attractors or attractor avoidance. Control problems with more complex targets, that is, arbitrary subsets which are not necessarily attractors or subspaces, are, to our best knowledge, not being exhaustively tackled. For this reason, different from the method in Chapter 3, the method presented in this chapter considers a subset as target.

Identifying all the minimal control strategies for a generic subset is a complex problem. In some cases, it might require the exploration of the state space, which grows exponentially with the size of the network. To deal with this computational explosion, we explore model checking techniques. Model checking is a form of program verification that determines if a given transition system satisfies a specific property. Model checking has been widely used and many tools have been developed in the context of Boolean networks [CGR12].

In this chapter we describe an approach to control strategy identification, based on model checking, that provides a complete solution set of minimal controls allowing full flexibility on the control target. The main results presented in this chapter are published in Cifuentes-Fontanals et al. [CFTS22b], from which I am first author and main developer of the method. In particular, Sections 4.1 to 4.3 are extracted from [CFTS22b], with the permission of the co-authors.

The chapter is organized as follows. We start by picking up the idea of using trap spaces in control from the previous chapter and go a step forward by presenting an intuitive method based on the completeness of the minimal trap spaces to identify control strategies (Section 4.1). Motivated by the idea of using model checking as a basis for an exhaustive approach, we give an introduction to model checking, focusing on the main concepts needed for this work (Section 4.2). In Section 4.3, we establish the basis for the control strategy computation with model checking. The implementation of the approach is detailed in Section 4.4, together with techniques to reduce the search space size and improve the performance of the method. Finally, in Section 4.5 we show the applicability of our method to a biological case study.

4.1 Control by completeness

As shown in Chapter 3, approaches to control strategy identification based on value percolation, either percolating directly to the target or to a trap space, can be implemented efficiently. However, these methods still might miss many possible control strategies. An example of such a control strategy is shown in Figure 4.1.

In this section we explore sufficient conditions to impose on the minimal trap spaces of the controlled system to identify a set of interventions as a control strategy. As mentioned in Section 2.1, minimal trap spaces are often good approximations of attractors in biological systems, that is, they are in one-to-one correspondence with attractors. To determine when this is the case in a Boolean network, it is enough to check whether its minimal trap spaces are univocal, faithful and complete in the corresponding dynamics [KS15] (see Section 2.1). We recall the definition of completeness for trap spaces and extend it to a Boolean function.

Definition 4.1.1. A set of trap spaces $\mathcal{T}$ is *complete* in $D(f)$ if and only if for every attractor $\mathcal{A}$ of $D(f)$ there exists $T \in \mathcal{T}$ such that $\mathcal{A} \subseteq T$. A Boolean function f is *complete* in the dynamics $D(f)$ if its minimal trap spaces are complete in $D(f)$.

The following proposition presents sufficient conditions for a set of interventions to be a control strategy. Given a candidate intervention set $\mathcal{C}$, if the set of minimal trap spaces of the controlled function is complete and contained in the target subset, then $\mathcal{C}$ is a control strategy for that subset.

Proposition 4.1.2. *Let f be a Boolean function, $P \subseteq \mathbb{B}^n$ a subspace, $\mathcal{C}$ a set of interventions and $\mathcal{T}$ the set of minimal trap spaces of $f^{\mathcal{C}}$. If all the trap spaces of $\mathcal{T}$ are contained in P and $\mathcal{T}$ is complete in $D(f^{\mathcal{C}})$, then $\mathcal{C}$ is a control strategy of P.*

Proof. Let $\mathcal{A}$ be an attractor of $D(f^{\mathcal{C}})$. Since $\mathcal{T}$ is complete in $D(f^{\mathcal{C}})$, there exists a minimal trap space $T \in \mathcal{T}$ such that $\mathcal{A} \subseteq T$. Therefore, $\mathcal{A} \subseteq T \subseteq P$. □

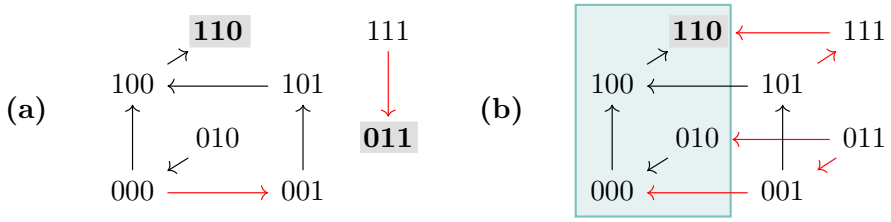


Figure 4.1: (a) Asynchronous dynamics of the Boolean function $f(x) = (x_1\bar{x}_3 \vee \bar{x}_2, x_1\bar{x}_3 \vee x_2x_3, \bar{x}_1\bar{x}_2 \vee x_2x_3)$ with two steady states 110 and 011 and (b) asynchronous dynamics of the controlled function $f^{\mathcal{C}} = (\bar{x}_2 \vee x_1, x_1, 0)$, with $\mathcal{C} = \{(3, 0)\}$. Transitions that vary between $AD(f)$ and $AD(f^{\mathcal{C}})$ are marked in red. Attractors are marked in bold with gray background. A green rectangle marks the subspace associated to the control intervention $**0$. $\mathcal{C}$ is a control strategy for $P = 110$ in $AD(f)$. $\mathbb{B}^n$ does not percolate to P nor to any non-trivial trap space under $f^{\mathcal{C}}$. $T = 110 \subseteq P$ (in gray) is the only minimal trap space of $f^{\mathcal{C}}$ and is complete in $D(f^{\mathcal{C}})$.

Proposition 4.1.2 provides sufficient conditions that allow us to identify new control strategies missed by percolation-based approaches. An example of such a control strategy is shown in Figure 4.1. $\mathbb{B}^n$ percolates to the subspace S under $f^{\mathcal{C}}$ and S is not a selected trap space nor included in the target P. Since $f^{\mathcal{C}}$ is complete in $AD(f^{\mathcal{C}})$ and its only minimal trap space is contained in P, $\mathcal{C}$ is identified as a control strategy. Completeness of the minimal trap spaces can be detected using model checking as described in [KS15]. We refer to this approach for control strategy identification as the *completeness* approach. Although the completeness approach might increase the amount of control strategies identified, it still does not characterize all the possible control strategies satisfying Definition 2.2.6. Figure 4.2 shows an example of a set of interventions $\mathcal{C}$ that is a control strategy for the target P. Although $\mathbb{B}^n$ percolates to S under $f^{\mathcal{C}}$ and S is the unique trap space in $f^{\mathcal{C}}$, $S \nsubseteq P$ and Proposition 4.1.2 cannot be applied. Thus, $\mathcal{C}$ is not detected as a control strategy by the completeness approach.

Motivated by the potential of model checking to verify the completeness of a network, in the following sections we formulate a model-checking approach to obtain the full solution set of control strategies.

4.2 Introduction to model checking

This section provides a practical introduction to the model checking concepts required for the description of our approach. For a more extensive and detailed explanation of model checking we refer the reader to [BK08]. Model checking is a formal method used

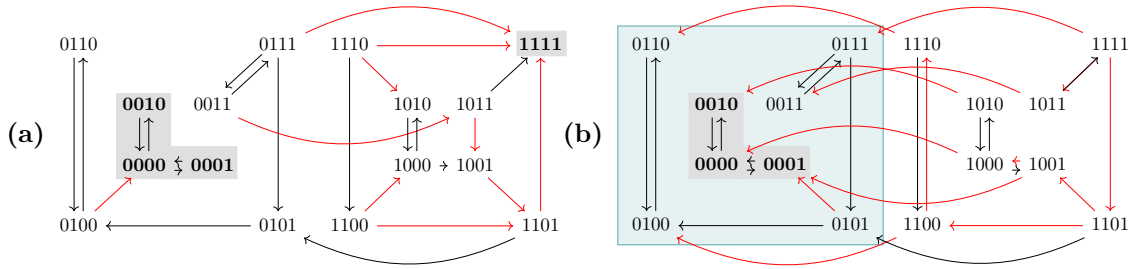


Figure 4.2: The asynchronous dynamics of a Boolean function $f(x) = (x_1\bar{x}_2 \vee \bar{x}_1x_3x_4 \vee x_1x_2\bar{x}_4 \vee x_1x_2x_3,\ x_1x_4 \vee \bar{x}_1x_2x_3\bar{x}_4 \vee \bar{x}_2x_3x_4 \vee x_2\bar{x}_3x_4,\ \bar{x}_2\bar{x}_3\bar{x}_4 \vee x_1x_2x_4 \vee \bar{x}_1\bar{x}_2x_3x_4 \vee \bar{x}_1x_2\bar{x}_3\bar{x}_4,\ x_1x_2 \vee x_1x_4 \vee \bar{x}_2\bar{x}_3\bar{x}_4 \vee x_3x_4)$ and the corresponding controlled function $f^{\mathcal{C}}(x) = (0,\ x_2x_3\bar{x}_4 \vee \bar{x}_2x_3x_4 \vee x_2\bar{x}_3x_4,\ \bar{x}_2\bar{x}_3\bar{x}_4 \vee \bar{x}_2x_3x_4 \vee x_2\bar{x}_3\bar{x}_4,\ \bar{x}_2\bar{x}_3\bar{x}_4 \vee x_3x_4)$, with $\mathcal{C} = \{(1,0)\}$, are represented in (a) and (b) respectively. Transitions that vary between $AD(f)$ and $AD(f^{\mathcal{C}})$ are marked in red. Attractors are marked in bold with gray background. A green rectangle marks the subspace associated to the control intervention $0***$. $\mathcal{C}$ is a control strategy for $P = 00**$ in $AD(f)$. $\mathbb{B}^n$ does not percolate to the target P nor to any other trap space under $f^{\mathcal{C}}$. Since $T = 0***$ is the unique trap space of f^C and $T \not\subseteq P$, $\mathcal{C}$ would not be identified as control strategy by the completeness approach.

in computer science to solve verification problems. Its application to the control strategy problem presents many advantages, for instance the use of symbolic representation, which allows one to deal with systems with a large number of states, like STGs of Boolean networks. Moreover, many efficient algorithms have been developed and are available for running model checking queries. An overview of existing model checking tools in the context of biochemical networks analysis can be found in [CGR12].

Model checking allows one to verify whether a given transition system satisfies a specific property. A transition system is defined as a set of states and a set of transitions, which represent changes from one state to another. Here we define a labeled transition system (LTS) as a tuple $(\mathsf{S}, \mathsf{T}, \mathsf{L})$ where S is a finite set of states, $\mathsf{T} \subseteq \mathsf{S} \times \mathsf{S}$ is a transition relation such that $(x^1, x^2) \in \mathsf{T}$ if there exists a possible transition from state x^1 to state x^2 and $\mathsf{L}\colon \mathsf{S} \to 2^{AP}$ is a labeling function with AP a finite set of atomic propositions. In the following, a transition (x^1, x^2) will also be denoted by $x^1 \to x^2$. The labeling function L gives a set $\mathsf{L}(x) \in 2^{AP}$ of atomic propositions for each state x which includes exactly the atomic propositions satisfied by x. In the Boolean context, an STG defines an LTS, where the set of states is $\mathbb{B}^n$ and the transitions are defined by the Boolean function and the type of update that is chosen. For our purposes, we need a deadlock-free transition system, so we add extra transitions $(x, x) \in \mathsf{T}$ for every steady state $x \in \mathbb{B}^n$. We use the atomic propositions $AP = \{(v = c) \mid v \in V, c \in \mathbb{B}\}$ and define the labeling function by $(v = c) \in \mathsf{L}(x)$ if and only if $x_v = c$.

There are different ways to express properties of a transition system. In our case, we use Computation Tree Logic (CTL). CTL is based on a branching notion of time, where the behavior of the system is represented by a tree of states. In the case of Boolean networks, one can imagine that every path starting in a state $x \in \mathbb{B}^n$ is represented as a branch in a tree rooted in x. In the following we introduce the main concepts of CTL.

We distinguish between state properties and path properties. In this context, a *path* is an infinite sequence $x^0, x^1, \ldots \in \mathsf{S}$ such that $(x^{i-1}, x^i) \in \mathsf{T}$ for all $i \geq 1$. A statement about a state or a path can be made using a CTL formula. A CTL *state formula* ϕ over the set of atomic propositions AP is of the form

$$\phi := a \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \mathbf{E}\varphi \mid \mathbf{A}\varphi$$

where $a \in AP$ is an atomic proposition, $\mathbf{E}$ is the *exists* operator, $\mathbf{A}$ is the *for all* operator, ϕ, ϕ_1 and ϕ_2 are CTL state formulas and φ is a CTL *path formula*, which in our work will be of the form:

$$\varphi := \mathbf{F}\psi \mid \mathbf{G}\psi$$

where $\mathbf{F}$ is the *future* operator, $\mathbf{G}$ the *global* operator and ψ a CTL state formula. Note that we do not use the full expressiveness of CTL but only a subset of operators necessary to formulate our control queries. In accordance to the definition of semantics of CTL formulas, a state formula is evaluated for a state whereas a path formula is evaluated for an infinite path. When $\phi = a$, where a is an atomic proposition, if $a \in \mathsf{L}(x)$ we say that the CTL state formula ϕ is satisfied at the state x and we write $x \models \phi$. For example, if $\phi = (i = 1)$ for some $i \in V$, then $x \models \phi$ if $(i = 1) \in \mathsf{L}(x)$, that is $x_i = 1$. Analogously, we write $\pi \models \varphi$ when the CTL path formula φ is satisfied by a path π. See Table 4.1 for the recursive definition of the satisfaction relation $\models$ for transition systems and CTL formulas used in this work. Since we only use atomic propositions of the form $(v = c)$ where $v \in V$ and $c \in \mathbb{B}$ and $(v = c) \in \mathsf{L}(x)$ if and only if $x_v = c$, atomic propositions can be interpreted in each state x according to the values of each variable in x. Thus, to simplify the notation, given an atomic proposition $\phi_{v,c} = (v = c)$, we define $\phi_{v,c}(x) = (x_v = c)$ so that $\phi_{v,c}$ is satisfied by x if and only if $\phi_{v,c}(x) = \mathit{true}$. Figure 4.3 shows some examples of state and path formulas which are satisfied in an STG.

4.3 Control with model checking

In this section, we present the basis of our new approach for the identification of all the minimal control strategies, based on model checking. To do so, we express the definition

$x \models true$		
$x \models a$	iff	$a \in \mathsf{L}(x)$
$x \models \phi_1 \vee \phi_2$	iff	$x \models \phi_1$ or $x \models \phi_2$
$x \models \phi_1 \wedge \phi_2$	iff	$x \models \phi_1$ and $x \models \phi_2$
$x \models \neg\phi$	iff	$x \not\models \phi$
$x \models \mathbf{E}\varphi$	iff	$\exists \pi \in \text{Paths}(x)$ s.t. $\pi \models \varphi$
$x \models \mathbf{A}\varphi$	iff	$\forall \pi \in \text{Paths}(x), \pi \models \varphi$
$\pi \models \mathbf{F}\phi$	iff	$\exists y \in \pi$ s.t. $y \models \phi$
$\pi \models \mathbf{G}\phi$	iff	$\forall y \in \pi, y \models \phi$
$x \models \mathbf{EF}\phi$	iff	$\exists \pi \in \text{Paths}(x), \exists y \in \pi$ s.t. $y \models \phi$
$x \models \mathbf{AG}\phi$	iff	$\forall \pi \in \text{Paths}(x), \forall y \in \pi, y \models \phi$

Table 4.1: Satisfaction relation and semantics for the CTL formulas used in this work, with $a \in AP$ an atomic proposition, $x \in S$ a state, π a path in the transition system, φ a path formula and ϕ, ϕ_1 and ϕ_2 state formulas.

of control strategy for a target subset in terms of CTL formulas. We start by rewriting it in terms of paths.

Lemma 4.3.1. *Let f be a Boolean function, $P \subseteq \mathbb{B}^n$ a subset, $\mathcal{C}$ a set of interventions and S a subspace such that $\mathbb{B}^n$ percolates to S under $f^{\mathcal{C}}$. The following are equivalent:*

(i) $\mathcal{C}$ is a control strategy for P in $D(f)$.

(ii) For every $x \in S$ there exists $y \in P$ such that there exists a path in $D(f^{\mathcal{C}})$ from x to y and there does not exist any path in $D(f^{\mathcal{C}})$ from y to any state outside P (that is, all paths starting at y are contained in P).

Proof. ($\Rightarrow$) Let $x \in S$ and let $\mathcal{A}$ be an attractor of $D(f^{\mathcal{C}})$ that can be reached from x. Since $\mathcal{C}$ is a control strategy, $\mathcal{A} \subseteq P$. Take $y \in \mathcal{A}$. Since $\mathcal{A}$ is reached from x, there exists a path from x to $y \in \mathcal{A} \subseteq P$ and there are no paths from y leaving P.

($\Leftarrow$) Let $\mathcal{A}$ be an attractor of $f^{\mathcal{C}}$. Let $x \in \mathcal{A}$. Since $x \in \mathcal{A} \subseteq S$, there exists $y \in P$ such that there exists a path in $D(f^{\mathcal{C}})$ from x to y and there does not exist any path in $D(f^{\mathcal{C}})$ from y to any state outside P. Since $\mathcal{A}$ is an attractor, $y \in \mathcal{A}$ and $\text{Reach}_{D(f^{\mathcal{C}})}(y) = \mathcal{A}$. Then, $\mathcal{A} \cap \mathbb{B}^n \backslash P = \emptyset$, that is, $\mathcal{A} \subseteq P$, and $\mathcal{C}$ is a control strategy for P. □

Before expressing Lemma 4.3.1 in terms of CTL formulas, we introduce a state formula ψ_S that is satisfied at a state x if and only if the state x belongs to the subspace $S = \Sigma(I, c)$:

$$\psi_S(x) = \bigwedge_{i \in I} (x_i = c_i).$$

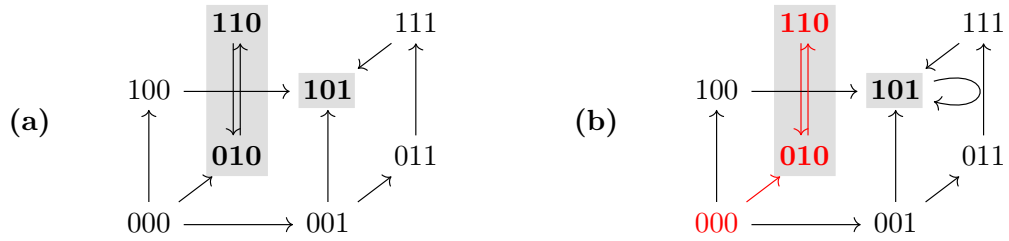


Figure 4.3: (a) Asynchronous state transition graph of a Boolean function with two attractors, $\{101\}$ and $\{010, 110\}$ (marked in gray). (b) TS for the STG shown in (a). Note that a self-loop has been added in the steady state 101 to have a deadlock-free TS. The states 001, 011, 101 and 111 satisfy the state formula $\mathbf{AG}\phi_3$, where $\phi_i = (i = 1)$, while $\mathbf{EF}\phi_3$ is satisfied in all the states except 010 and 110. The path that starts at 000 and then oscillates between 010 and 110 (in red) satisfies for instance $\mathbf{F}\phi_2$ and $\mathbf{G}\neg\phi_3$ but not $\mathbf{G}\phi_1$.

This formulation can be extended to subsets as well. Clearly, every subset can be written as a union of subspaces, since a singleton set constitutes a subspace. Let $P \subseteq \mathbb{B}^n$ be a subset. We define ϕ_P to be satisfied at a state x if and only if the state x belongs to P:

$$\phi_P(x) = \bigvee_{S \in \mathcal{S}} \psi_S(x)$$

where $\mathcal{S}$ is a subspace cover of P.

Now we can express Lemma 4.3.1 in terms of CTL formulas, using $\phi_P(x)$ as defined above.

Proposition 4.3.2. *Let f be a Boolean function, $P \subseteq \mathbb{B}^n$ a subset, $\mathcal{C}$ a set of interventions and S a subspace such that $\mathbb{B}^n$ percolates to S under $f^{\mathcal{C}}$. The following are equivalent:*

(i) $\mathcal{C}$ is a control strategy for P in $D(f)$.

(ii) $\Phi_P(x)$, defined as $\Phi_P(x) = \boldsymbol{EF}(\boldsymbol{AG}(\phi_P))(x)$, is satisfied in $D(f^{\mathcal{C}})$ for every $x \in S$.

Proof. Φ_P is satisfied at a state x if and only if there exists a path $x = x^0, x^1, ...$ such that $\mathbf{AG}(\phi_P)$ is satisfied at x^i for some $i \geq 1$. Let $y = x^i$. $\mathbf{AG}(\phi_P)$ is satisfied at y if and only if for all paths $y = y^0, y^1, ...$, for all $i \geq 0$, ϕ_P is satisfied at y^i, that is, $y^i \in P$. Thus, by Lemma 4.3.1, Φ_P is satisfied for all $x \in S$ if and only if $\mathcal{C}$ is a control strategy for P in $D(f)$. □

Note that Φ_P is not affected by the presence of non-attractive cycles since, by definition, they contain at least one state with an outgoing trajectory leading to an attractor.

The CTL formula Φ_P defined in Proposition 4.3.2 provides a way to determine whether a set of interventions is a control strategy for P. The next section presents the implementation of this idea for control strategy identification.

4.4 Implementation

Given a set of interventions $\mathcal{C}$, we can determine whether $\mathcal{C}$ is a control strategy for a target P by checking whether the CTL formula $\Phi_P(x)$ is satisfied for the conditions in Proposition 4.3.2. However, running a CTL query in a large state space might entail a long computational time. To overcome this problem, we develop some preprocessing methods for the computation of node control strategies to reduce two factors affecting significantly the computational time: the dimension of the state space in which the query is run and the number of interventions to check. We start by giving an overview of the main algorithm for control strategy identification, clarifying the differences for node and edge control. Then, we go into details on some of the reduction methods developed.

4.4.1 Main algorithm

Building on the model checking formulas derived in the previous section, we develop a method for control strategy identification. The formula derived in Proposition 4.3.2 can be used to define a CTL query that can determine whether a candidate set of interventions is a control strategy for a target subset.

The main algorithm takes as inputs a constant-free Boolean function f (i.e. all coordinate functions of f are non-constant), a target subset P, the type of update D (asynchronous, synchronous or generalized), the limit size of the control strategies k and the (possibly empty) list of forbidden interventions *avoid_int* and returns the minimal control strategies for P in the respective dynamics. If the original Boolean function is not constant-free, a preprocessing step is applied in which the constant function values are percolated.

In order to improve the performance of the method, instead of exploring the whole state space for every candidate intervention, it is enough to explore the subspace obtained when percolating the state space under $f^{\mathcal{C}}$, that is, $S^* = F^*(f^{\mathcal{C}})(\mathbb{B}^n)$. Note that S^* is a trap space of $f^{\mathcal{C}}$, since it is the percolated subspace of the trap space $\mathbb{B}^n$. As discussed in Section 2.1.1, this reduction to the percolated subspace does not miss any of the attractors

(see Proposition 2.1.7). Thus, the result of the exploration using the model checking query applied to the function restricted to S^* in the reduced space is equivalent to the result of the exploration in the full state space with the original controlled function. The complete implementation of the control strategy identification for node control is detailed in Algorithm 2 and explained in the following.

- For each candidate intervention set $\mathcal{C}$, the percolated subspace of $\mathbb{B}^n$ under $f^{\mathcal{C}}$, $S^* = \Sigma(I^*, c^*)$, is computed (line 9), as defined in Section 2.1.1. By Proposition 4.3.2, the control query can be run on S^*.
- If S^* is contained in the target subset P, then $\mathcal{C}$ is a control strategy by Proposition 3.1.1 (lines 12-13). If not, the algorithm continues to compute the restriction to S^* (see Definition 2.1.5), f^*, and its minimal trap spaces (lines 16-17).
- If there exists a minimal trap space disjoint from the target, the candidate intervention set is discarded, since each minimal trap space contains at least one attractor (line 18).
- Trap spaces that are partially contained in the target subset are analyzed first (line 19). For each trap space T', the function f^* is restricted to T', and the control query is run for the new restricted function f^{**} (lines 22-24). By Proposition 2.1.7, the control formula needs to be satisfied in each of the trap spaces in order to be satisfied in S^*. Thus, if there is a trap space where the formula is not satisfied, the candidate intervention set is discarded (lines 24-27).
- Otherwise, the algorithm concludes by checking the CTL formula Φ_P for f^* and deciding whether $\mathcal{C}$ is a control strategy (lines 29-30).

Since the aim is to identify minimal control strategies, the candidate intervention sets $\mathcal{C}$ are taken randomly fixing an increasing number of variables, so that supersets of sets already defining a successful intervention strategy are not considered (lines 5-8). Note that the forbidden interventions *avoid_intv*, which can be determined by the biological control problem or by a preprocessing step that discards a priori some candidate interventions, are excluded (line 6). Furthermore, an upper bound k for the size of the control strategies can be set. Moreover, the decisions made for each percolated subspace are stored in two variables ST (for positively checked subspaces) and SF (for negatively checked subspaces) to avoid repeating the same verification query.

The algorithm presented above is implemented using PyBoolNet [KSS16], a Python package that allows generation and analysis of Boolean networks. PyBoolNet uses NuSMV to decide model checking queries for Boolean networks. It also provides an efficient computation of trap spaces for relatively large networks. The source code of the implementation of Algorithm 2 is available at [CFb].

Algorithm 2 Node control strategies for a target subset P

Input: f Boolean function, P target, D update, k limit size, *avoid_intv* forbidden interv.

Output: control strategies for P

```
1: function ControlStrategies(f, P, D, k)
2:     CS ← ∅
3:     ST ← ∅                                  ▷ ST stores positively checked subspaces
4:     SF ← ∅                                  ▷ SF stores negatively checked subspaces
5:     for i in {1, ..., min(k, n)} do:        ▷ n: number of variables of f
6:         C ← {C intervention sets: |C| = i and C ∩ avoid_intv = ∅}
7:         for C in C do:
8:             if (C ⊈ C' for all C' in CS) then:
9:                 S* ← F*(f^C)(B^n)
10:                if S* ∈ ST then: add C to CS
11:                if S* ∈ ST ∪ SF then: break
12:                if S* ⊆ P then:
13:                    add C to CS
14:                    add S* to ST
15:                else:
16:                    f* ← reduce(f, S*)
17:                    minTS ← minimalTrapSpaces(f*)
18:                    if (T ∩ P ≠ ∅ for all T in minTS) then:
19:                        halfTS ← {T in minTS if T ⊈ P}
20:                        valid ← true
21:                        for T in halfTS do:
22:                            f** ← reduce(f*, T)
23:                            Φ_P ← CTLFormula(f**, P)
24:                            if not CTLModelChecking(f**, D, Φ_P) then:
25:                                valid ← false
26:                                add S* to SF
27:                                break
28:                        if valid then:
29:                            Φ_P ← CTLFormula(f*, P)
30:                            if CTLModelChecking(f*, D, Φ_P) then:
31:                                add C to CS
32:                                add S* to ST
33:                            else
34:                                add S* to SF
35:                    else
36:                        add S* to SF
37:    return CS
```

Edge control

Some of the reductions used in the previous algorithm to identify node control strategies are less effective or directly not applicable when considering edge interventions. One of the reasons is that the application of a generic edge intervention does not necessarily cause the fixing of a regulatory function to a constant value. In fact, often no regulatory function gets fixed. Consequently there is none or very little percolation and the dimensionality reduction associated to it has no impact. Moreover, the storing of the decisions made for each percolated subspace is not useful anymore since it would also be necessary to store the corresponding controlled function for which the subspace is a control strategy, which would take too much memory. Consequently, the algorithm for edge control presented here, Algorithm 3, is a basic approach and would require the implementation of performance improvements to be able to deal with relatively large biological networks. For this reason, the examples shown in the application section (Section 4.5) are limited to node control.

4.4.2 Reduction methods

In order to reduce the computational time of the control query, we develop different reduction methods to simplify the network or the candidate intervention space. The following simple reduction techniques are applicable to node and edge control and any type of target.

The first step in Algorithms 2 and 3 is to reduce the search to the percolated subspace S of $\mathbb{B}^n$ under the controlled function $f^{\mathcal{C}}$. Since S is a trap space for $f^{\mathcal{C}}$, by Proposition 2.1.7, we can restrict the controlled function to S and the result of the control query for $f^{\mathcal{C}}_S$ in S is equivalent to the result for $f^{\mathcal{C}}$ in $\mathbb{B}^n$. Thus, we can reduce the dimensions of the search space and the controlled function. Since Boolean networks modeling biological systems often induce a large amount of value propagation, in node control this usually results in a considerable reduction in the dimension of the state space and controlled function that leads to a significant reduction of the computational time.

Another simple check that can be done is to detect whether the target requires the fixing of input components. Since input components are not regulated by any other variable, the only way of getting them fixed to a desired value is by controlling them directly. Thus, these components are always part of all the control strategies. For this reason, before the algorithm starts, a preprocessing step is performed to detect whether there are input components among the fixed components in the target. If they are either free in the network, that is, not fixed to any value, or fixed to the opposite value than the one in the target, they are automatically added to the resulting control strategy and removed from the intervention candidate pool. Although this reduction is valid for any type of target that requires the fixing of input components, it is particularly useful when the target is a steady state, since all the inputs of the network need to be fixed.

Algorithm 3 Edge control strategies for a target subset P

Input: f Boolean function, P target, D update, k limit size, *avoid_intv* forbidden interv.

Output: control strategies for P

```
1: function ControlStrategies(f, P, D, k, avoid_intv)
2:     CS ← ∅
3:     for i in {1, ..., min(k, n)} do:                ▷ n: number of variables of f
4:         C ← {C intervention sets : |C| = i and C ∩ avoid_intv = ∅}
5:         for C in C do:
6:             if (C ⋢ C' for all C' in CS) then:
7:                 S* ← F(f^C)(B^n)
8:                 if S* ⊆ P then:
9:                     add C to CS
10:                else:
11:                    f* ← reduce(f, S*)
12:                    minTS ← minimalTrapSpaces(f*)
13:                    if (T ∩ P ≠ ∅ for all T in minTS) then:
14:                        halfTS ← {T in minTS if T ⋢ P}
15:                        valid ← true
16:                        for T in halfTS do:
17:                            f** ← reduce(f*, T)
18:                            Φ_P ← CTLFormula(f**, P)
19:                            if not CTLModelChecking(f**, D, Φ_P) then:
20:                                valid ← false
21:                                break
22:                        if valid then:
23:                            Φ_P ← CTLFormula(f*, P)
24:                            if CTLModelChecking(f*, D, Φ_P) then:
25:                                add C to CS
26:    return CS
```

More complex reduction steps can also be designed to reduce the computational time. In the following, we present some of them. They are applicable to node control taking a subspace as target, unless specified otherwise. We divided them into two types depending whether they focus on reducing the state space where the control query is checked or on reducing the candidate intervention space in order to diminish the number of calls to the control query.

Reduction by restricting to smaller regions of the state space

We recall the concept of autonomous set and the methodology developed in [KS15] to reduce the computational time for determining whether a Boolean network is complete, to adapt it to the method of control strategy identification. The idea is to define necessary conditions for a set of interventions to be a control strategy in terms of the autonomous sets of the interaction graph.

Given a Boolean function f and a set of components $U \subseteq V$, we define Above(U) as the set $\{i \in V \mid \exists$ path in IG(f) from i to some $u \in U\}$. We say that U is *autonomous* if U = Above(U). Note that when U is an autonomous set, for any $u \in U$, f_u does not depend on the value of any component $i \notin U$. Now we recall the restriction of a Boolean function to a subspace S, f_S, as defined in Definition 2.1.5. In particular, given an autonomous set U and $I = V \backslash U$, we observe that $f_S = f_{S'}$ for any two subspaces $S = \Sigma(I, c)$ and $S' = \Sigma(I, c')$, with $c, c' \in \mathbb{B}^n$. Thus, the choice of c is not relevant. From now on we write f_U to represent any of these restrictions. Analogously, we define π_U and ι_U. The following remark extends Remark 2.1.6 to autonomous sets.

Remark 4.4.1. For any autonomous set U and any two states $x, y \in \mathbb{B}^n$, if there is a path from $\iota_U(x)$ to $\iota_U(y)$ in $D(f_U)$ then there is a path from x to y in $D(f)$ and vice versa.

The following proposition and corollary establish a necessary condition for a set of interventions to be a control strategy based on autonomous sets. They state that, given a set of interventions $\mathcal{C}$, it is necessary that all the autonomous sets of the controlled system ($f^{\mathcal{C}}$) satisfy the control strategy query in order for a set of interventions $\mathcal{C}$ to be a control strategy. To ease the notation in the proposition, we consider a Boolean function f and an empty set of interventions $\mathcal{C} = \emptyset$. Note that, given a Boolean function f and a non-empty set of interventions $\mathcal{C} \neq \emptyset$, $\mathcal{C}$ is a control strategy for a given target P in $D(f)$ if and only if $\mathcal{C}' = \emptyset$ is a control strategy for P in $D(f^{\mathcal{C}})$.

Proposition 4.4.2. *Let f be a Boolean function, $P = \Sigma(I, c)$ a subspace and $U \subseteq V$ an autonomous subset in IG(f) such that $I \cap U \neq \emptyset$. If $\emptyset$ is not a control strategy for $P' = \Sigma(I', c)$ with $I' = I \cap U$ in $D(f_U)$, then $\emptyset$ is not a control strategy for P in $D(f)$.*

Proof. $\emptyset$ is not a control strategy in $D(f)$ iff there exists $x \in \mathbb{B}^n$ such that for all $\pi \in \text{Paths}_{D(f)}(x)$, for all $y \in \pi$, there exists a path in $D(f)$ from y to some $z \in \mathbb{B}^n \backslash P$. Assume $\emptyset$ is not a control strategy for P' in $D(f_U)$. Then there exists an $\tilde{x} \in \mathbb{B}^m$, with $m = |U|$, such that for all $\tilde{\pi} \in \text{Paths}_{D(f_U)}(\tilde{x})$, for all $\tilde{y} \in \tilde{\pi}$, there exists a path in $D(f_U)$ from $\tilde{y}$ to some $\tilde{z} \in \mathbb{B}^m \backslash P'$. Take $x \in \mathbb{B}^n$ such that $x = \iota_U(\tilde{x})$. Let $\pi \in \text{Paths}_{D(f)}(x)$ and $y \in \pi$. By Remark 4.4.1, there exists a path in $D(f_U)$ from $\tilde{x}$ to $\hat{y}$ with $y = \iota_U(\hat{y})$. Then, there exists a path in $D(f_U)$ from $\hat{y}$ to some $\hat{z} \in \mathbb{B}^m \backslash P'$. By Remark 4.4.1, there exists a path in $D(f)$ from y to $z \in \mathbb{B}^n$ with $z = \iota_U(\hat{z})$. Since $\hat{z} \in \mathbb{B}^m \backslash P'$, there exists $i \in I' \subseteq I$ such that $z_i = \hat{z}_i \neq c_i$ and, consequently, $z \in \mathbb{B}^n \backslash P$. Therefore $\emptyset$ is not a control strategy for P in $D(f)$. □

Corollary 4.4.2.1. *Let f be a Boolean function, $\mathcal{C}$ a set of interventions, $P = \Sigma(I, c)$ a subspace. $\mathcal{C}$ is not a control strategy for P if there exists an autonomous subset $U \subseteq V$ in $IG(f^{\mathcal{C}})$ such that $I \cap U \neq \emptyset$ and $\emptyset$ is not a control strategy for $P' = \Sigma(I', c)$ with $I' = I \cap U$ in $D(f_U^{\mathcal{C}})$.*

Corollary 4.4.2.1 allows us to define an algorithm to check if there exists an autonomous set not satisfying this requirement and therefore discard the candidate intervention set without having to check the control query in the whole state space.

The use of Corollary 4.4.2.1 as a performance improvement depends on the target subspace and the type of the network. If the Above set of the components defining the target subspace is large, the reduction applied might not be significant. This can occur when the components describing the target are outputs of the system, since they are likely to be influenced directly or indirectly by most of the components of the network. In such cases, the checking of autonomous sets should be avoided since they could lead to an increase of the total computational time. In cases where the Above set of the target components is small, checking autonomous sets first could significantly reduce the computational time.

Reduction of the candidate space

In this section, we present two methods for reducing the candidate intervention space. The first one is adapted to control problems taking a subspace as target, while the second is only discussed for control problems targeting a minimal trap space. Both of them are only developed for node control, although the first one could potentially be extended to edge control.

The first reduction method is introduced by Samaga et al. for the control identification method by direct percolation [SKK10]. The key idea is to use the dependency between variables, derived from the interaction graph (IG), to discard a priori unsuccessful candidate interventions. A node i has a positive (resp. negative) dependency on a node j if there

exists a directed positive (resp. negative) path from i to j in the IG. These dependencies can be used to avoid considering interventions that do not influence correctly at least one target node [SKK10]. Given a target subspace $P = \Sigma(J, d)$, a candidate node intervention (i, c) is considered if and only if there exists a path from i to some $j \in J$ with positive sign for $c = d_j$ and negative sign for $c \neq d_j$. This property, encoded in the ASP implementation of the node control for direct percolation presented by Kaminski et al. [KSSV13] and recalled in this work (Section 3.3.2, Listing 3.3), allows us to discard a priori candidate interventions that will not have the desired effect on the control target.

The second reduction method is based on marker sets and it is discussed here for control problems targeting a minimal trap space. Borriello and Daniels [BD21] explored the idea of using distinguishing components to identify candidate node interventions for attractor control. Here we formalize this idea for minimal trap spaces using marker sets. Marker sets are usually defined as sets of components that separate attractors or phenotypes [KTF+21]. Here we extend the concept of marker set to distinguish a minimal trap space from the others.

Definition 4.4.3. Let f be a Boolean function and $M \subseteq V$. M is a *distinguishing marker set for a minimal trap space* $T = \Sigma(I, c)$ *in* f if, given any other minimal trap space $T' = \Sigma(J, d)$, $T \neq T'$, there exists $i \in I \cap J \cap M$ such that $c_i \neq d_i$.

We can use distinguishing marker sets to identify candidate nodes for control intervention.

Proposition 4.4.4. *Let f be a Boolean function, P a minimal trap space of f, $\mathcal{N}$ a node control strategy for P in $D(f)$ and $\mathcal{M}$ the set of distinguishing marker sets separating P from the rest of the minimal trap spaces of f. Then there exists an $M \in \mathcal{M}$ such that for all $i \in M$, $(i, c) \in \mathcal{N}$ for some $c \in \mathbb{B}$.*

Proposition 4.4.4 provides us with necessary conditions that a node control strategy must satisfy in terms of the marker sets. Since computing the marker sets is much faster than exhaustively identifying control strategies, the marker information can be used as a preprocessing step, so that the number of candidate subspaces to check is reduced.

4.5 Application

In this case study, we look at node control strategies for the MAPK network analyzed in the previous chapter (see Section 3.4 for a full description of the network). We start by targeting the subspace defined by the apoptotic phenotype in asynchronous, synchronous and general asynchronous updates. We compare the node control strategies identified by

Table 4.2: Number and size of the control strategies identified by each method (DP, TS, CO, CN, and MC) for the corresponding target subspace in the asynchronous, generalized asynchronous and synchronous dynamics. The numbers in parenthesis denote the amount of truly minimal control strategies identified by each method.

	Asynchronous and Generalized Asynchronous			**Synchronous**		
Method	$\|\mathcal{N}\| = 1$	$\|\mathcal{N}\| = 2$	$\|\mathcal{N}\| = 3$	$\|\mathcal{N}\| = 1$	$\|\mathcal{N}\| = 2$	$\|\mathcal{N}\| = 3$
DP	2	124 (59)	175 (45)	2	124 (88)	175 (49)
TS	2	0	0	1	0	0
CO	3	106 (59)	162 (45)	2	124 (88)	175 (49)
CN	8	105	66	2	164 (112)	195 (155)
MC	8	105	66	4	118	216

the methods for permanent control presented in Chapter 3 (direct percolation (DP), via trap spaces (TS) and combined (CO)) to the ones identified by completeness (CN) and model checking (MC). We also deal with attractor control in the asynchronous update and compare the control strategies obtained targeting the four steady states selected in Section 3.4 (s_1, s_1', s_2, s_2'). As expected, percolation-based approaches (DP, TS and CO) are significantly faster than the completeness and exhaustive approaches. The running times are in the order of milliseconds for the ASP-based approaches (DP, TS and CO) while both the completeness and model checking approaches require several hours. In all the cases, the exhaustive approach is able to identify new control strategies missed by direct percolation and via trap spaces.

4.5.1 Target: apoptotic phenotype

Table 4.2 shows the number of control strategies up to size 3 obtained by each method for the different dynamics. The control strategies for asynchronous and generalized asynchronous dynamics are represented in the same column since they are the same in all the approaches. Note that although all the methods look for minimal control strategies under inclusion, only the exhaustive approach is able to guarantee that no smaller control strategy exists. Thus, the control strategies identified by non-exhaustive methods might not be minimal with respect to all the strategies. The numbers in parenthesis in Table 4.2 denote the amount of truly minimal control strategies identified by each method. Note that the exhaustive approach is able to uncover new minimal control strategies with respect to percolation-based methods in all the dynamics.

As mentioned in the previous chapter, the control strategies identified by direct percolation are the same in all the updates. The minimal trap spaces of the network are

a good approximation of the attractors (see Chapter 2) for the MAPK network in the asynchronous and generalized dynamics but not in the synchronous one. Consequently, no additional control strategy is identified by TS compared to DP in the synchronous dynamics. CN and MC obtain the same number of control strategies in the asynchronous and generalized dynamics. This is not the case in the synchronous dynamics, where additional control strategies are obtained by MC. This might be caused by the fact that the CN method cannot decide on a set of interventions being a control strategy if the controlled network is not complete, which is more likely to happen when the original network is not complete.

The eight minimal control strategies of size 1 identified for the apoptotic phenotype in the asynchronous and generalized asynchronous dynamics are

$$\{(\text{TGFBR-stimulus, 1})\},\ \{(\text{TGFBR, 1})\},\ \{(\text{DNA-damage, 1})\},\ \{(\text{ATM, 1})\},\\ \{(\text{FRS2, 1})\},\ \{(\text{GRB2, 1})\},\ \{(\text{TAK1, 1})\},\ \{(\text{TAOK, 1})\}$$

and the four in the synchronous dynamics,

$$\{(\text{TGFBR-stimulus, 1})\},\ \{(\text{TGFBR, 1})\},\ \{(\text{DNA-damage, 1})\},\ \{(\text{ATM, 1})\}$$

Note that the four control strategies identified for the synchronous dynamics are also control strategies in the other updates. Moreover, the first three are identified by direct percolation or via trap spaces while the last five are only identified by the completeness method and the exhaustive approach.

4.5.2 Target: minimal trap spaces

We now consider attractor control in the asynchronous update. As in Section 3.4.2, we choose as targets the steady states s_1, s_1', s_2, s_2', with $s_1, s_1' \in \mathcal{S}_1 = \{$EGFR-stimulus = 0, FGFR3-stimulus = 0, TGFBR-stimulus = 0 and DNA-damage = 0$\}$ and $s_2, s_2' \in \mathcal{S}_2 = \{$EGFR-stimulus = 0, FGFR3-stimulus = 0, TGFBR-stimulus = 0 and DNA-damage = 1$\}$. Table 4.3 shows the number and size of the node control strategies up to size 7 (the number of inputs plus three) of these four steady states identified by each method.

We observe that in all the cases CN and MC are able to identify minimal control strategies of size 5, while DP does not find any and TS can only find them for two of the four steady states. Interestingly, all the control strategies of size 5 have the same structure: four of the five node interventions fix the inputs to the right value and the fifth variable is either GAB1 or PI3K. In the interaction graph of the MAPK network (Figure 3.8), we can observe that these two variables form a positive cycle of length 2 which might be responsible for the multi-stability within S_1 and S_2 after the four input components have

Table 4.3: Number and size of the control strategies identified by the different methods up to size 7 for the different steady states in asynchronous dynamics. Note that there is no control strategy up to size 4. s_1 and s_1' denote the two steady states in $\mathcal{S}_1$ and s_2 and s_2' the two steady states in $\mathcal{S}_2$.

Node control	s_1			s_1'			s_2			s_2'		
$\lvert\mathcal{N}\rvert$	5	6	7	5	6	7	5	6	7	5	6	7
DP	0	0	60	0	0	32	0	14	2	0	14	2
TS	2	0	0	0	8	12	0	22	14	2	0	0
CO	2	0	0	0	8	12	0	22	14	2	0	0
CN	2	0	0	2	0	0	2	0	0	2	0	0
MC	2	0	0	2	0	0	2	0	0	2	0	0

been fixed. In particular, either GAB1 or PI3K needs to be set to 1 to guarantee that system will evolve to s_1 or s_2' and to 0 for s_1' or s_2.

4.6 Discussion

In this chapter we presented an approach based on model checking to identify all the minimal control strategies for an arbitrary target subset. As shown in Section 4.5, the model checking approach is able to uncover many relevant control strategies that would be missed by other non-exhaustive methods. Moreover, it provides full flexibility on the definition of the control target, allowing to study different control problems such as attractor avoidance, which can lead to gain additional insights into the network. In particular, this approach is specially interesting when a deep analysis of the control strategies of the model is required, as shown in the following chapter.

Due to its exhaustiveness, this approach entails significantly more computational time than other approaches, such as direct percolation or trap spaces (Chapter 3), since in some cases the full exploration of the state space might be required. For this reason, we developed several reduction methods for our approach to diminish the computational time and make it feasible for application. As seen in Section 4.5, our approach is able to deal with relatively large biological networks for node control. Further reduction methods should be developed for edge control to make it accessible for state-of-the-art biological networks, such as the extension of the reduction methods for node control to edge control or the identification of equivalent intervention sets.

Chapter 5

Application

In this chapter, we study more extensively the applicability of the methods for control strategy identification presented in Chapters 3 and 4. We start by applying our methods to a network modeling the epithelial-to-mesenchymal transition, considering different control targets: attractor, subspace and subset avoidance, and studying both node and edge interventions. We analyze the different control strategies obtained for the asynchronous dynamics in each case and explore their relevance in the biological context. In the second section, we compare our approaches to current control methods for attractor and target control considering different dynamics in multiple Boolean networks. All the results presented here were obtained with a regular desktop 8-processor computer, Intel®Core™ i7-2600 CPU at 3.40GHz, 16GB memory.

The results for permanent node control presented in this chapter are published in Cifuentes-Fontanals et al. [CFTS22b, CFTS22c] and included here with the permission of the co-authors.

5.1 Case study: EMT network

The network considered in this case study was recently introduced by Selvaggio et al. [SCP+20] to model how micro-environmental signals influence cancer-related phenotypes along the epithelial-to-mesenchymal transition (EMT). During the EMT, epithelial cells progressively lose connections with their neighboring cells, which are held together laterally by tight junctions and adherens junctions (AJ), to adopt a migratory capacity, mediated by the remodeling of focal adhesions (FA) [DW19, SCP+20]. Moreover, this transition is reversible, since mesenchymal cells can lose their migratory capacities and return to the epithelial state (mesenchymal-to-epithelial transition, MET). Hybrid phenotypes can be

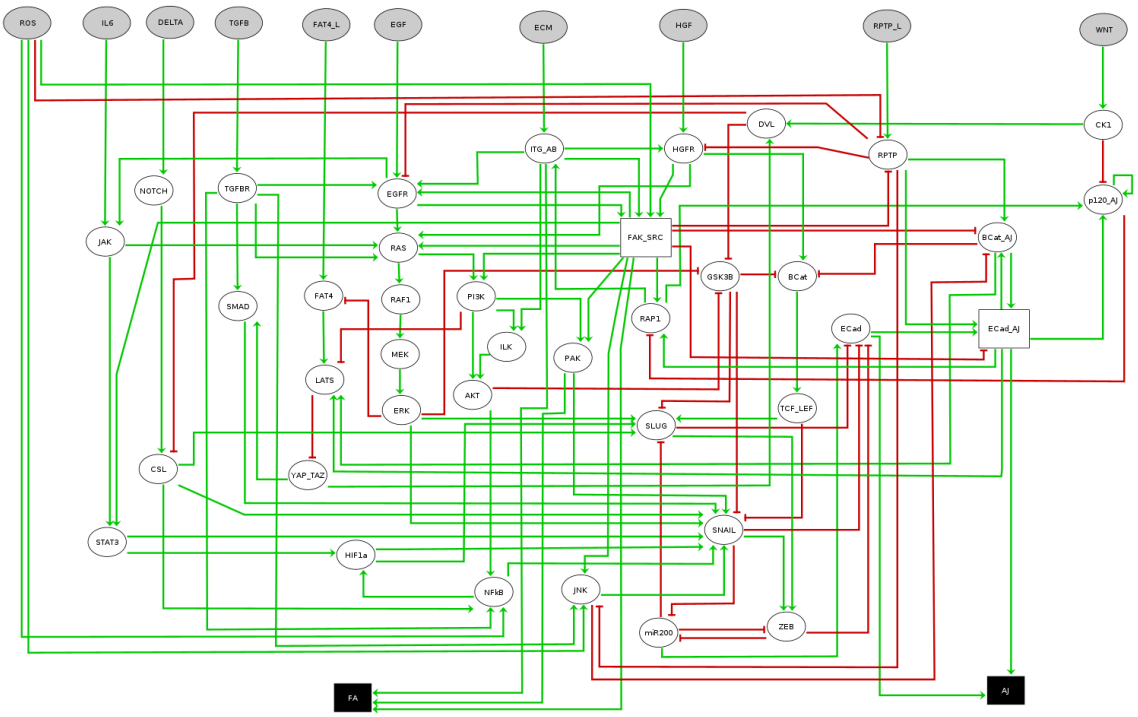


Figure 5.1: EMT multi-valued network. Boolean nodes are represented by ellipses and multi-valued nodes by rectangles. Input and output nodes are colored in gray and black respectively. Image obtained using GINsim [CNT12], model from [SCP+20]. Further information about the model can be found in [SCP+20].

found at intermediate states between these transitions and often show traits from epithelial and mesenchymal cells, which may provide advantageous abilities to cancer cells such as drug resistance or tumor-initiating potential [SCP+20, DW19].

To model this process, Selvaggio et al. (2020) created a network consisting of 51 components (see Figure 5.1). The ten inputs of the model aim to capture the micro-environmental signals and factors that can induce EMT. They represent cell-cell signaling molecules (WNT, DELTA), growth factors (EGF, HGF), inflammatory signals (IL6, ROS, TGFb), cell adhesion molecules (FAT4, RPTP) and the stiffness of the extracellular matrix (ECM) [SCP+20]. The two multi-valued readouts of the model represent the qualitative degrees of cell adhesions by adherens junctions (AJ) and focal adhesions (FA). The internal components of the network are modeled by Boolean variables except for two, FAK-SRC and ECad-AJ, both of them assigned to a three-level variable. The levels of FAK-SRC capture different interactions happening at different activity levels, whereas the three levels of ECad-AJ (0,1,2) represent whether the protein ECad is not located at the membrane, it

Table 5.1: Relation of the phenotypes of the EMT network, the values of the multivalued readouts (AJ, FA) in bold, and the values of the equivalent Boolean components (AJ1, AJ2, FA1, FA2, FA3). The number of steady states belonging to each phenotype is also shown.

		AJ	AJ1	AJ2	**FA**	FA1	FA2	FA3	Number of steady states
Epithelial phenotype	E1	**2**	1	1	**0**	0	0	0	60
Hybrid phenotypes	H1	**2**	1	1	**1**	1	0	0	40
	H2	**1**	1	0	**2**	1	1	0	36
	H3	**2**	1	1	**3**	1	1	1	48
Mesenchymal phenotypes	M1	**0**	0	0	**1**	1	0	0	208
	M2	**0**	0	0	**2**	1	1	0	368
	M3	**0**	0	0	**3**	1	1	1	672
Undefined phenotype	UN	**0**	0	0	**0**	0	0	0	20

is correctly located and bound to BCat-AJ and whether the stable complex (BCat-ECad-p120) has been formed, respectively [SCP$^+$20].

Since the original model is multi-valued, we work with its booleanised version obtained with GINsim [CNT12]. This booleanisation maps a multi-valued component of maximum value m to m Boolean components. For instance, a component taking four values FA = 0, 1, 2, 3 is encoded using 3 Boolean variables (FA1, FA2, FA3) that take values 000, 100, 110, 111 respectively to represent the four possible values of FA (see Table 5.1). Although this method introduces states that do not correspond to any value of the multi-valued variable (non-admissible states), these cannot be part of any attractor since they always have at least one path to an admissible state and do not have incoming transitions from admissible states. Therefore, the asymptotic behavior generated strictly replicates the original model. The booleanised network of this case study consists of 56 Boolean variables, whose regulatory functions can be found in the PyBoolNet repository [KSS16].

The asynchronous dynamics has 1452 attractors, all of them steady states. They are classified into different phenotypes, which are defined in terms of the values of the readout components **AJ** and **FA**. The presence of adherens junction (AJ $>$ 0, FA $=$ 0) is a feature of epithelial states, whereas focal adhesions (FA $>$ 0, AJ $=$ 0) are displayed by mesenchymal cells. Hybrid phenotypes are characterized by the presence of both traits (AJ $>$ 0 and FA $>$ 0) [SCP$^+$20]. This results in eight biological phenotypes, divided in four groups, epithelial (E1), mesenchymal (M1, M2, M3), hybrid (H1, H2, H3) and unknown (UN), according to the values of AJ and FA (see Table 5.1). Note that phenotypes are defined according to the values of AJ and FA of the steady states. Thus, the value combinations that are not present in any steady state do not appear as phenotypes.

We analyze different control targets for node and edge control. We start by targeting single steady states (attractor control). Then, we target the subspaces corresponding to each phenotype (target control). Finally, we study the avoidance of the hybrid phenotype, setting as target the complement of the general hybrid phenotype. Since the exhaustive approach considering edge interventions takes too long to be run for all steady states and phenotypes, the results for edge control shown in Sections 5.1.1 and 5.1.2 are obtained using the method combining direct percolation and trap spaces. The results for the hybrid avoidance using node and edge interventions are obtained using the exhaustive approach.

5.1.1 Attractor control: steady states

Here we consider the problem of attractor control, starting with node control. Since the control targets are steady states, the minimum number of interventions in each control strategy is at least the number of inputs (each input component needs to be fixed to the corresponding value in the attractor).

For each of the 1452 steady states, the minimal node control strategies up to size 13 are identified. 788 steady states have minimal control strategies of size 10, meaning that the dynamics can be controlled to the steady state only by fixing the values of the input components to their values in the attractor. Of the remaining ones, 396 need an extra component to be fixed, 212 require fixing at least two more components and the last 56 steady states require fixing three extra components. Figure 5.2 shows the number of minimal control strategies identified for each size (10-13) with steady states grouped by phenotype, distinguishing between control strategies identified by the method combining direct percolation and trap spaces or only by the model checking approach. Note that in most of the cases the exhaustive approach is able to identify many control strategies that are missed by the direct percolation and trap spaces combined approach.

By definition, every steady state is a selected trap space, thus all the control strategies identified by direct percolation are also identified via trap spaces. Although most of the steady states only have one selected trap space (the steady state itself), there are 32 steady states that have an additional selected trap space. Control via trap spaces is able to identify new control strategies for all these 32 steady states that are missed by direct percolation. However, these new control strategies are supersets of the ones identified by the exhaustive approach. Consequently, they are not minimal with respect to all the strategies and do not appear in Figure 5.2.

In addition, we observe that the mesenchymal phenotypes are the ones with the highest amount of control strategies, which is to be expected since they are also the ones containing more attractors. The number of steady states per phenotype can be found in Table 5.1. Interestingly, no control strategies consisting of only input variables lead to hybrid steady states.

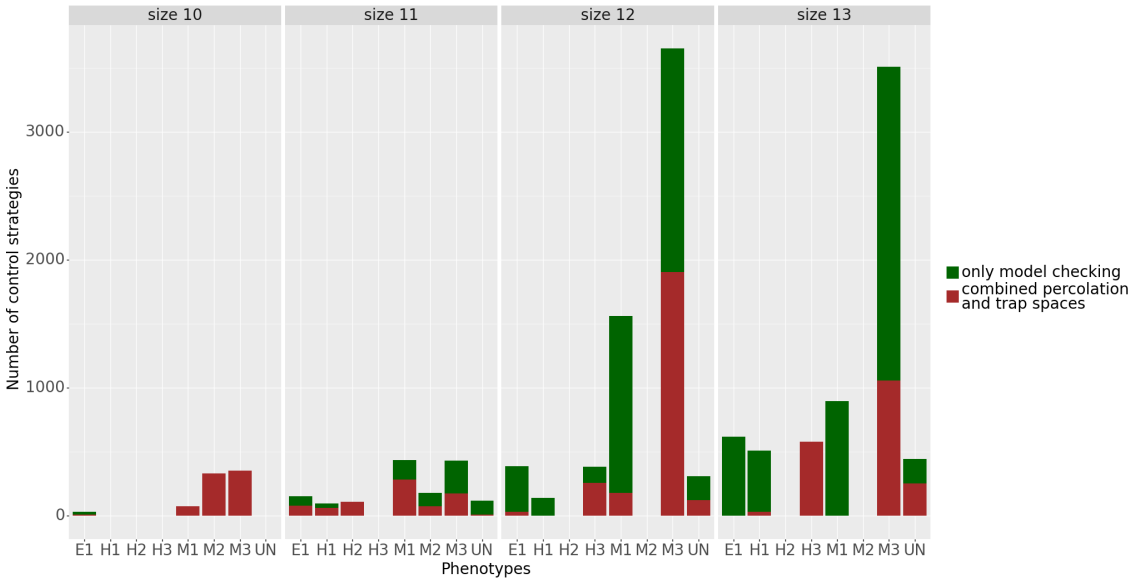


Figure 5.2: Number of node control strategies identified for the steady states grouped by phenotype and size. The minimal control strategies obtained by the combined method of direct percolation and trap spaces are represented in red and the additional control strategies identified by model checking in green. Non-minimal control strategies are excluded. The number of steady states per phenotype can be found in Table 5.1.

Next, we consider both node and edge control. Figure 5.3 shows the amount of control strategies obtained by the method combining direct percolation and trap spaces for each size (10-13) with steady states grouped by phenotype, distinguished by the type of interventions used: node, edge or both types. The amount of node-only and edge-only control strategies of size 10 is the same, since these strategies consist of interventions that fix the inputs to the values in the steady state. Since from the point of view of application edge interventions targeting inputs are identical to the corresponding node interventions, we exclude these edge interventions when computing mixed control strategies. For this reason, there are no combined interventions of size 10. Steady states in the hybrid phenotype H3 have significantly more control strategies using edge interventions than the rest of phenotypes. These edge interventions could indicate crucial interactions in the decision making process of the system leading to the steady states in H3.

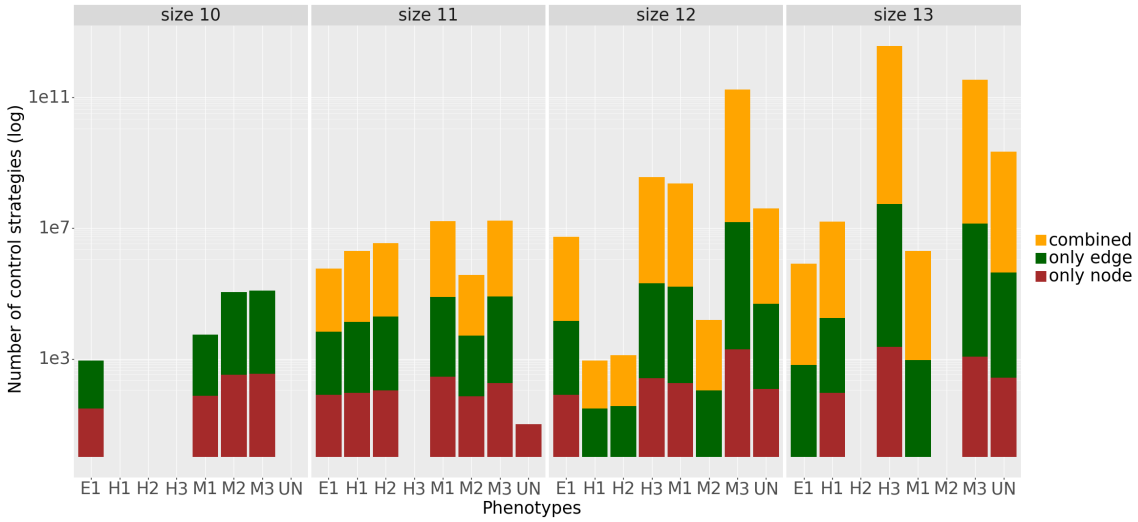


Figure 5.3: Number of control strategies identified by the combined method of direct percolation and trap spaces for the steady states grouped by phenotype and size. Note that the number of control strategies is given in logarithmic scale. The control strategies obtained by using only node or only edge interventions are represented in red and green respectively, while the control strategies including both node and edge interventions are represented in yellow. The number of steady states per phenotype can be found in Table 5.1.

5.1.2 Target control: phenotypes

The minimal node control strategies up to size 3 are identified for each of the phenotypes, taking as target the subspace defined by the corresponding values of the phenotypic components in each case (see Table 5.1). The five Boolean phenotypic components, representing the two adhesion properties, are excluded from the candidate interventions, since they represent the readouts of the model that we want to control. Table 5.2 shows the number of control strategies identified per phenotype and size. Note that no control strategies up to size 3 are identified for E1 and H1. However, when the method is run with a larger limit size, several control strategies of size 4 appear for both phenotypes. Similarly to the case of attractor control, we observe that the phenotypes with higher number of control strategies are the mesenchymal phenotypes (over a hundred), while the epithelial and the hybrid phenotypes have fewer or no control strategies up to size 3. This is consistent with the bias of the model towards the mesenchymal phenotypes in terms of attractors.

Table 5.2: Number of minimal control strategies identified per size for each phenotype using node control.

Phenotype	**E1**	**H1**	**H2**	**H3**	**M1**	**M2**	**M3**	**UN**
Size 1	0	0	0	0	0	0	0	0
Size 2	0	0	0	0	3	3	17	0
Size 3	0	0	6	2	113	111	83	14

All of the minimal control strategies obtained are also identified by direct percolation, except for three minimal control strategies for the phenotype M3 that are only identified by model checking. These are: {BCat-AJ = 1, GSK3B = 1, ITG-AB = 1}, {ECad-AJ1= 1, GSK3B = 1, ITG-AB = 1} and {ECad-AJ2 = 1, GSK3B = 1, ITG-AB = 1}. The method via trap spaces is only able to identify control strategies for the phenotypes M2 and M3 that are also identified by direct percolation.

Figure 5.4 shows a graphic representation of the components involved in the control strategies of size 2 obtained for the M1, M2 and M3 phenotypes. We observe that the intervention (ROS,1), where ROS is an input component that represents reactive oxygen species present in the extracellular micro-environment, appears in at least one control strategy in all three phenotypes. In fact, the intervention (ROS,1) is able to induce the suppression of all the epithelial markers (ECad, ECad-AJ, BCat-AJ, miR200), AJ1 and AJ2, which get fixed to 0, and the activation of FA1 and FA2, which get fixed to 1. Consequently, (ROS,1) leads the system to the phenotypes M2 and M3. A second intervention is needed to distinguish between these two phenotypes. Note that the intervention (ROS,1) is also present in a control strategy for M1. In this case, the second intervention (PAK,0) is crucial to avoid phenotypes M2 and M3 since it limits the level of FA, preventing FA2 and FA3 to become 1.

Another interesting remark is that all the control strategies for M3 except one, have the intervention (ITG-AB,1) in common. ITG-AB represents the presence of integrins, transmembrane receptors composed of two subunits (alpha and beta) capable of interacting and adhering to the extracellular matrix. They provide a link between the outside environment and cellular responses related to motility, including migration of cancer cells [JPI20]. The only control strategy that does not involve ITG-AB is {(ECM,1), (ROS,1)}. The extracellular matrix (ECM) is a mesh of molecules that support physically and biochemically the surrounding cells. ECM = 1 denotes ECM stiffening, which can induce EMT and promote tumor invasion and metastasis [SCP$^+$20, WFT$^+$15]. However, combined with the intervention (ROS,1), they are enough to induce the mesenchymal phenotype M3.

As in Section 5.1.1, the control strategies for node and edge control are obtained using the method combining direct percolation and trap spaces. The number of control strategies identified for each phenotype is shown in Table 5.3 and its representation in terms of the type of interventions used is shown in Figure 5.5. Note that we exclude edge interventions

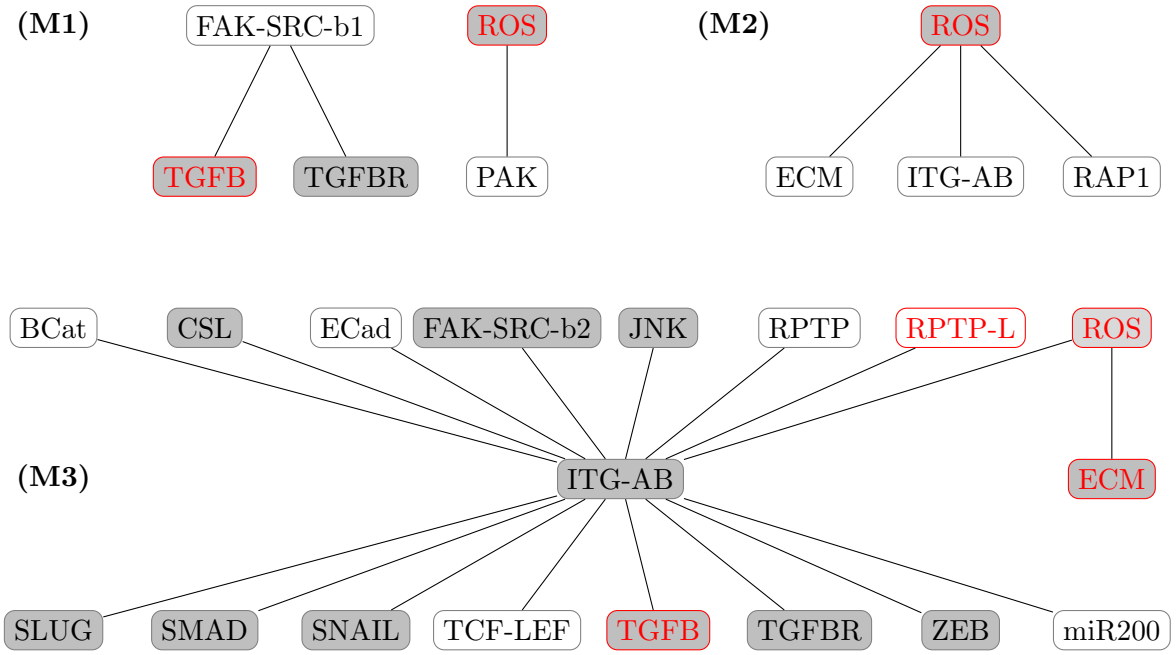


Figure 5.4: Control strategies of size 2 for the M1, M2 and M3 phenotypes using node interventions. Pairs of nodes linked by an edge represent control strategies of size 2. The color of the node (white and gray) represents the value that the component is fixed to (0 and 1, respectively). Input nodes are marked in red.

Table 5.3: Number of control strategies per size for each phenotype identified by the percolation method considering node and edge interventions. Note that mixed control strategies do not include edge interventions targeting inputs, since from the point of view of application they are identical to the corresponding node interventions.

Phenotype	**E1**	**H1**	**H2**	**H3**	**M1**	**M2**	**M3**	**UN**
Size 1	0	0	0	0	0	0	0	0
Size 2	0	0	0	0	4	6	35	0
Size 3	0	0	12	2	436	930	777	36

targeting inputs when computing mixed control strategies, since from the point of view of application they are identical to the corresponding node interventions. We observe that there are significantly more control strategies mixing node and edge control than just using a unique type of interventions. This is not surprising since mixed control strategies consider more combinations of interventions and some edge interventions can be equivalent to a node intervention. Although there are node control strategies for six out of the eight phenotypes, only two of them have edge control strategies up to size 3. This might be

caused by the limitation on the size of the control strategies, since edge control might require the application of more interventions than node control.

The control strategies of size 2 identified for the targets M1, M2 and M3 using node and edge interventions are shown in Figure 5.6. Note that all the components appearing in edge interventions are also part of a node control strategy, except for FAK-SRC-b1 in M2. Many edge interventions are equivalent to the corresponding node ones. Others, despite not fixing any component to a constant, are able to trigger, in combination with ITG-AB, the same phenotype as the corresponding node strategy.

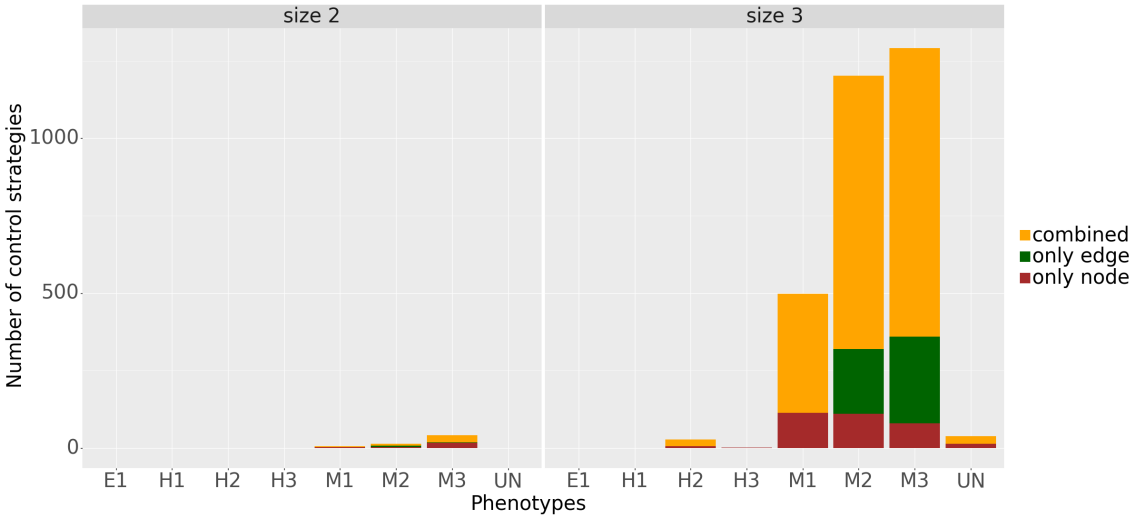


Figure 5.5: Number of control strategies identified for each phenotype up to size 3. The control strategies obtained by using only node or only edge interventions are represented in red and green respectively, while the control strategies including both node and edge interventions are represented in yellow.

5.1.3 Subset control: avoidance of hybrid phenotypes

As mentioned above, hybrid phenotypes may provide cancer cells advantageous features such as drug resistance or tumor-initiating potential [SCP+20, DW19]. Therefore, interventions avoiding these phenotypes might be good candidates for drug targets in therapeutic treatment against cancer cells presenting these traits.

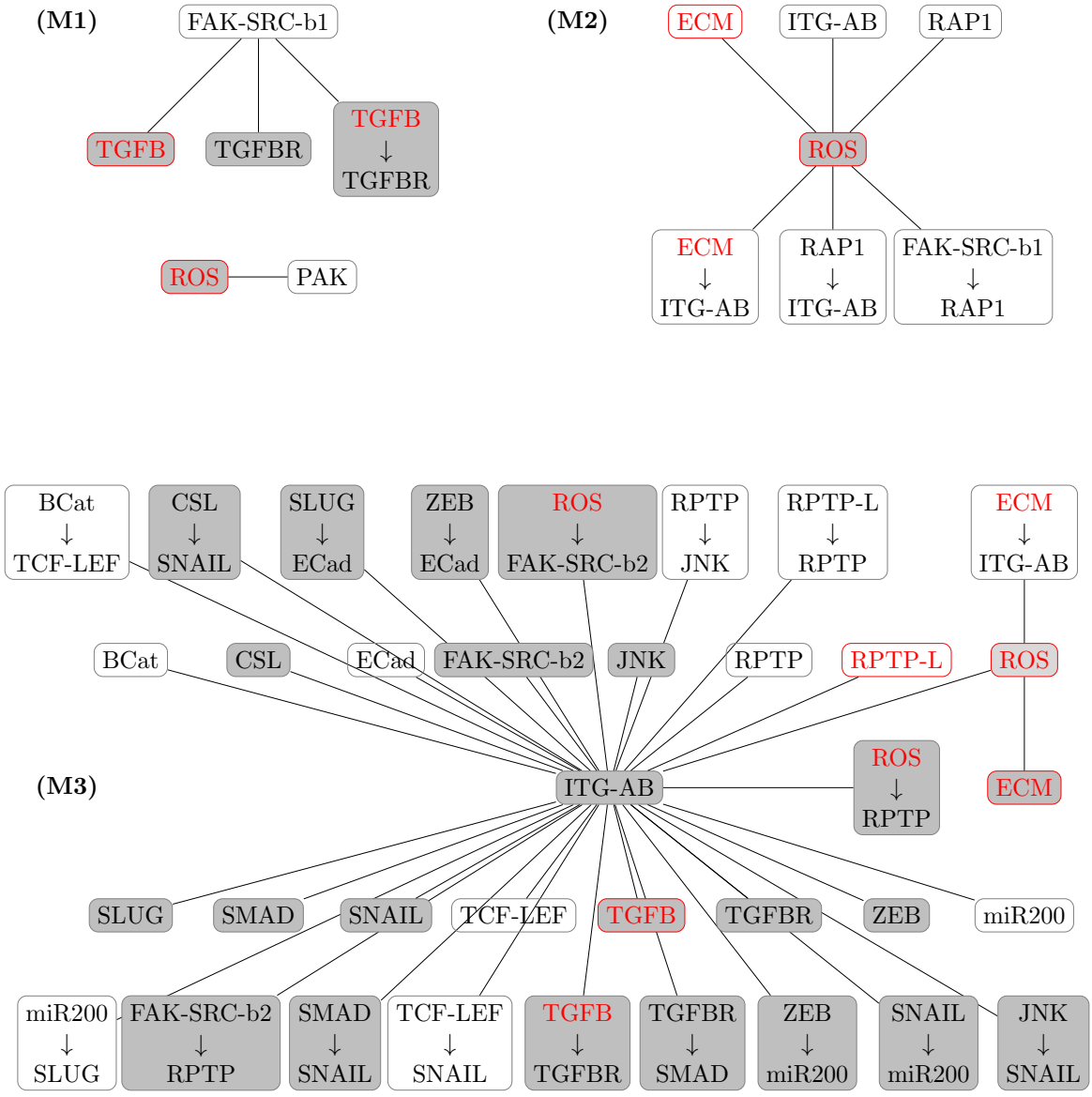


Figure 5.6: Control strategies of size 2 for the M1, M2 and M3 phenotypes using node and edge interventions. Pairs of nodes linked by an edge represent control strategies of size 2. The color of the node (white and gray) represents the value that the component is fixed to (0 and 1, respectively). Input nodes are marked in red.

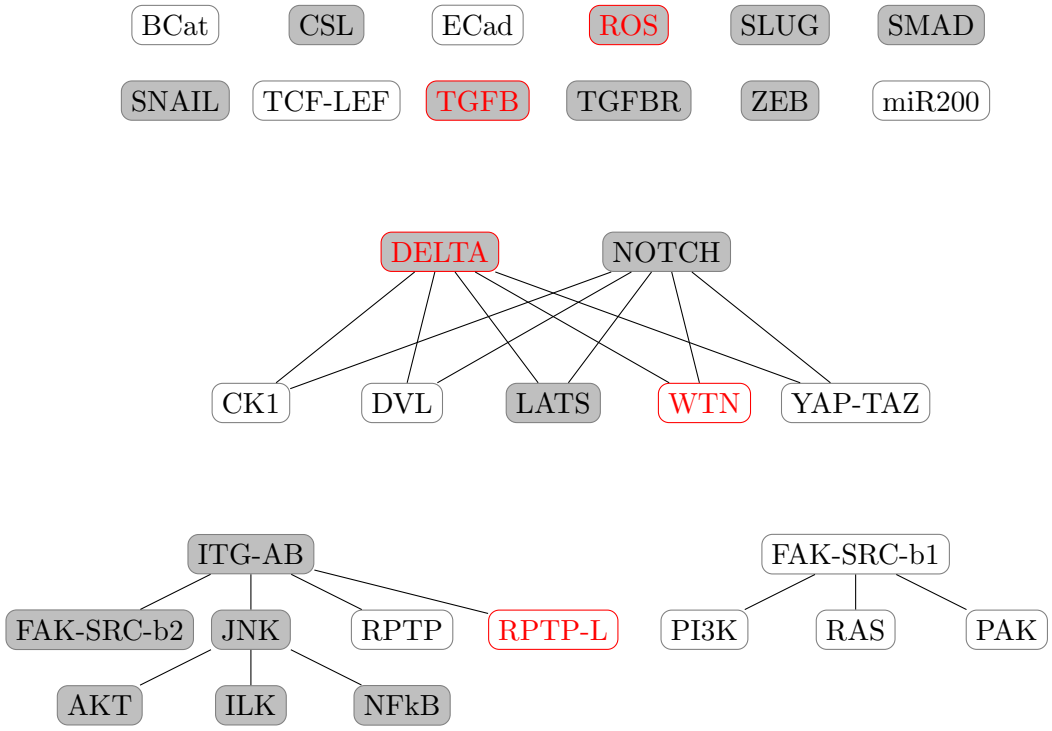


Figure 5.7: Control strategies of size 1 and 2 for the avoidance of the hybrid phenotype using node interventions. Single nodes represent control strategies of size 1 and pairs of nodes linked by an edge represent control strategies of size 2. The color of the node (white and gray) represents the value that the component is fixed to (0 and 1, respectively). Input nodes are marked in red.

Hybrid phenotypes are characterized by having both components AJ and FA activated, that is, $\mathrm{AJ} \geq 1$ and $\mathrm{FA} \geq 1$. Thus, the subset defining the avoidance of the hybrid phenotype is

$$P = \{\mathrm{AJ1} = 0, \mathrm{AJ2} = 0\} \cup \{\mathrm{FA1} = 0, \mathrm{FA2} = 0, \mathrm{FA3} = 0\}.$$

As in the previous case, the five phenotypic components are excluded from the candidate interventions. Setting the upper bound on the size of the control strategies to 2, we obtain 32 control strategies (see Figure 5.7): twelve of size 1 and twenty of size 2.

The twelve control strategies of size 1 obtained are:

$\{\mathrm{ECad} = 0\}$, $\{\mathrm{ROS} = 1\}$, $\{\mathrm{SLUG} = 1\}$, $\{\mathrm{SNAIL} = 1\}$, $\{\mathrm{TGFB} = 1\}$, $\{\mathrm{TGFBR} = 1\}$, $\{\mathrm{ZEB} = 1\}$, $\{\mathrm{BCat} = 0\}$, $\{\mathrm{CSL} = 1\}$, $\{\mathrm{SMAD} = 1\}$, $\{\mathrm{TCF\text{-}LEF} = 0\}$, $\{\mathrm{miR200} = 0\}$.

The first seven control strategies can be identified using direct percolation, targeting the epithelial state $\{\mathrm{AJ1} = 0, \mathrm{AJ2} = 0\}$ or the mesenchymal one $\{\mathrm{FA1} = 0, \mathrm{FA2} = 0, \mathrm{FA3} = 0\}$, which can be represented as subspaces. The last five are not identified by using only percolation but are captured by the model checking approach. Two of the obtained interventions correspond to input variables (ROS and TGFB) while the other ten correspond to internal components. Each of the input interventions (ROS,1) and (TGFB,1) lead the system to mesenchymal phenotypes. TGFB represents the transforming growth factor beta-1 (TGFB), which is a cytokine that regulates growth and differentiation of cells. Looking at the regulatory functions, we observe that TGFBR is uniquely regulated by TGFB, so setting TGFB to 1 implies that TGFBR is also set to 1. In addition, since TGFB does not regulate any other component, we can deduce that these interventions are equivalent in terms of their effect on phenotypic components (see Definition 2.2.4). Moreover, since there are no control strategies of size 1 for individual phenotypes (see Table 5.2), we deduce that these interventions lead to systems where multiple epithelial, mesenchymal or unknown phenotypes coexist, none of them being hybrid.

The internal components involved in the control strategies identified include the epithelial markers (ECad and miR200) and the mesenchymal ones (BCat, SNAIL, SLUG, TCF-LEF and ZEB) as described in [SCP$^+$20], which is consistent with the hybrid avoidance. In addition, the authors of [SCP$^+$20] performed a systematic analysis of the effect of single mutants on the attractor landscape, excluding the input variables. All the single mutants corresponding to the non-input interventions found by our approach were identified as having only attractors in non-hybrid phenotypes. Moreover, there was no other single mutation that produced this result. In other words, the results obtained by our approach are in complete correspondence to the ones presented in [SCP$^+$20].

The control strategies of size 2 for the avoidance of the hybrid phenotype are separated in three different groups (see Figure 5.7). This differs from the structure of the control strategies obtained for the mesenchymal phenotypes, where almost all the control strategies share a common component (see Figure 5.4). Note that all the control strategies for the mesenchymal phenotypes are control strategies for the avoidance of the hybrid ones. However, most of them do not appear in Figure 5.7, since they are supersets of control strategies of size 1. We also observe that DELTA and NOTCH are interchangeable. In fact, looking at Figure 5.1, we observe that NOTCH is uniquely regulated by DELTA and DELTA does not regulate any other component, as in the case of TGFB and TGFBR mentioned above. Consequently, interventions in any of these two components are equivalent (see Definition 2.2.4).

When considering edge interventions, we obtain ten edge control strategies of size 1 for the avoidance of the hybrid phenotype. Seven of them are equivalent to a node control strategy. These equivalences are shown in Table 5.4.

Table 5.4: Edge interventions from the control strategies of size 1 for the avoidance of the hybrid phenotypes and their corresponding equivalent node interventions, also control strategies of size 1.

Edge intervention	**Node intervention**
(BCat, TCF-LEF) = 0	TCF-LEF = 0
(SLUG, ECad) = 1	ECad = 0
(SLUG, ECad) = 1	ECad = 0
(SLUG, ECad) = 1	ECad = 0
(miR200, SLUG) = 0	SLUG = 1
(TGFB, TGFBR) = 1	TGFBR = 1
(ZEB, miR200) = 1	miR200 = 0
(SNAIL, miR200) = 1	miR200 = 0

Interestingly, the three remaining ones,

$$\{(\text{SMAD, SNAIL}) = 1\}, \{(\text{TCF-LEF, SNAIL}) = 0\}, \{(\text{TGFBR, SMAD}) = 1\},$$

do not correspond to any node control strategy nor get any component fixed to a constant value. However, they alter the system enough to destroy all the attractors from the hybrid phenotype. Note that the components targeted by these edge strategies, SNAIL and SMAD, are also node control strategies when fixed to 1. Moreover, each of these three edge interventions in combination with the node intervention (ITG-AB, 1) are control strategies for the phenotype M3 (see Figure 5.6).

In order to understand the effect of these edge interventions, we compute the steady states for each of the perturbed networks. The number of steady states per phenotype is the same as in the original network except for the hybrid ones, which are not steady states any more. Thus, each of these perturbations destroys only hybrid steady states and preserves the remaining steady states with minor alterations.

There are 116 control strategies of size 2 for the avoidance of the hybrid phenotype. Of them, 21 consist only of node interventions, 42 only of edge interventions and 53 include both node and edge interventions. Exploring the biological relevance of the interactions related to the suggested edge interventions could potentially lead to a better understanding of the hybrid phenotypes and the development of possible therapeutic strategies.

5.2 Comparison of methods

In this section, we compare the two approaches presented in Chapter 3 and Chapter 4 to other control methods currently available. In order to be able to compare different approaches, certain common features need to be chosen. Here, we consider control for any possible initial state. We focus on two different scenarios: one for attractor control and one for target control. Although methods for target control are usually aimed at targeting larger subspaces, determined for example by a phenotype, which often include several attractors, they can are also be used for attractor control when the target attractor is a steady state or a minimal trap space. Since this is the case for the target attractors used in this comparison, methods for target control are also included in the first scenario.

The comparison presented here encompasses each of the approaches for control strategy identification discussed in previous sections and two other control methods currently available: the method based on stable-motifs presented by Zañudo and Albert [ZA15] and the method based on basins of attraction introduced by Su and Pang [SP21]. The implementation used for the direct percolation approach is the one presented in [CFTS22c], originally adapted from [KSSV13]. All the methods used for the comparison are listed below. An overview of their main features is shown in Table 5.5.

- *Stable-motifs approach (SM)*, attractor control method based on the identification of the stable motifs of the network as described in [ZA15].
- *Basins approach (BA)*, attractor control method that uses the basin of attraction of the target attractor to identify control strategies as implemented in [SP20].
- *Direct percolation approach (DP)*, target control method based on percolation into the target subspace as implemented in [CFTS22c], originally introduced in [KSSV13].
- *Trap-spaces approach (TS)*, target control method based on percolation into selected trap spaces introduced in [CFTS20] and extended in [CFTS22c], as described in Section 3.1.
- *Combined approach (CO)*, target control method combining direct percolation and percolation via trap spaces [CFTS22c], as explained in Chapter 3.
- *Completeness approach (CN)*, target control method based on the completeness of the minimal trap spaces introduced in [CFTS22b] and described in Section 4.1.
- *Model checking approach (MC)*, target control method based on the query of a CTL formula in model checking introduced in [CFTS22b] and described in Section 4.3.

Table 5.5: Overview of the versatility of the different control methods in terms of the types of targets, interventions and update schemes.

	Control target					Update			Interv.	
Method	steady state	trap space attractor	complex attractor	subspace	arbitrary subset	async.	sync.	gen. async.	node	edge
BA	✓	✓	✓	-	-	✓	-	-	✓	-
SM	✓	✓	-	-	-	✓	-	-	✓	-
DP	✓	✓	-	✓	-	✓	✓	✓	✓	✓
TS	✓	✓	-	✓	-	✓	✓	✓	✓	✓
CO	✓	✓	-	✓	-	✓	✓	✓	✓	✓
CN	✓	✓	-	✓	-	✓	✓	✓	✓	✓
MC	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

The methods for attractor control considered here (BA and SM) only work for asynchronous update, so the comparison is only made for this dynamics. In the case of target control, control strategies are identified for the three dynamics. However, for simplicity only the synchronous and asynchronous are compared, since the control strategies obtained for the generalized asynchronous are the same as the ones obtained for the asynchronous dynamics.

Although some methods for target control can deal with edge interventions in the asynchronous dynamics [BD19] or in the synchronous one [MVCAL16, SVLM20], they only require that the steady states of the controlled system belong to the target subspace, and impose no restrictions on the complex attractors that might also exist. Since our approach identifies control strategies that require that all attractors (steady states or complex) belong to the target, a direct comparison to these methods is not possible.

In order to capture different control scenarios, three biological networks of various sizes with different type and number of attractors are considered. A short description of each network is provided below. See Table 5.6 for an overview of the networks and their features. The Boolean rules for each biological network can be found in the PyBoolNet repository [KSS16].

(a) T-LGL network, originally introduced by Zhang et al. (2008) [ZSY$^+$08] to model the T cell large granular lymphocyte (T-LGL) survival signaling network, which we use in the version presented in [ZA15]. This network consists of 60 Boolean variables and its asynchronous dynamics has 3 cyclic attractors.

(b) MAPK network, analyzed in Sections 3.4 and 4.5. As previously explained, it was introduced by Grieco et al. (2013) [GCBP$^+$13] to model the effect of the Mitogen-Activated Protein Kinase (MAPK) pathway on cell fate decisions taken in pathological cells. The network consists of 53 Boolean variables and it has 18 attractors in the asynchronous dynamics, 12 steady states and 6 cyclic attractors.

Table 5.6: Main features of the biological networks used in the comparison. Input components are fixed in the T-LGL network and free in the Cell-Fate and MAPK networks, unless specified otherwise.

Network		**Size**	**Inputs**	**Outputs**	**Attractors**	
					steady	cyclic
Cell-Fate	[CTF$^+$10]	28	3	3	27	0
MAPK	[GCBP$^+$13]	53	4	3	12	6
T-LGL	[ZSY$^+$08]	60	6	3	0	3

(c) Cell-Fate network, introduced by Calzone et al. (2010) [CTF$^+$10] to model the cell fate decision process. The network uses 28 Boolean variables and its asynchronous dynamics has 27 attractors, all of them steady states. These are classified in four different phenotypes (Apoptosis, Survival, Non-Apoptotic Cell Death and Naive) according to the values of the output components of the network.

As mentioned in Chapters 3 and 4 our methods take as input a constant-free Boolean function. Therefore, since the T-LGL network has constant coordinate functions, it has been pre-processed so that the constant values are percolated and removed from the network. For the sake of the comparison, all the methods have taken as input the reduced network.

5.2.1 Attractor control

As mentioned above, we compare the approaches presented in this work (DP, TS, CO, CN and MC) to two different control strategy identification methods for attractor control: the stable-motifs approach (SM) and the basins approach (BA). The stable-motifs approach is based on the identification of the network stable motifs as described in [ZA15]. It works for steady states and the complex attractors captured by the stable motifs (in some cases, complex attractors are not identified and the method cannot be applied). The basins approach works for any kind of attractor and computes the basins of attraction of the network attractors in order to identify control strategies of minimum size as explained in [SP21]. The stable motifs of the network are related to the trap spaces of the Boolean function, since they capture components locking each other to certain values, so it is reasonable that similar control strategies are obtained by TS and SM in most of the cases. The computation of the basins of attraction, on the other hand, provides full information about the network dynamics, therefore BA could have the potential to identify all the minimal control strategies of minimum size similar to MC. However, BA is not exhaustive and although it is able to identify more control strategies than most of the methods, in some cases it is not

able to find all of them. Both BA and SM only consider node control in the asynchronous update. Thus, the comparison is only made for node control in this dynamics.

For each network, we choose one of its attractors as the control target. To enrich the comparison, we select attractors for which the control strategies obtained by each method differ. The control target for the T-LGL network is the apoptotic attractor, that is, the attractor with Apoptosis = 1 and Proliferation = 0. This is the only attractor where the control strategies obtained for SM, BA and TS differ. The three methods identify the same control strategies for the other two attractors. The target attractor for the MAPK network is s_2 (see Section 3.4.2). Both SM and TS obtain the same control strategies for the remaining attractors of the MAPK network. The results for BA could not be obtained since the software was not able to process this network. The control target for the cell-fate network is the apoptotic steady state (Apoptosis = 1) that has the three inputs (FADD, FASL and TNF) set to 1. It is one of the ten attractors for which TS identifies more control strategies than SM. In the remaining seventeen attractors the two methods obtain the same control strategies. TS identifies more control strategies than BA in eighteen attractors, including the selected steady state.

Table 5.7 shows the number and size of the control strategies identified by each method for the selected attractors. In the T-LGL network, TS and SM are able to identify three of the four minimal control strategies identified by BA, CN and MC, {(SPHK1, 0)}, {(PDGFR, 0)} and {(S1P, 0)}. They both miss the fourth one {(Ceramide, 1)}. However, TS is able to identify five non-minimal control strategies combining (Ceramide, 1) with (PLCG1, 1), (GRB2, 1), (IL2RBT, 1), (IL2RB, 1) and (RAS, 1) respectively. Note that no control strategy for this attractor up to size 2 is identified by direct percolation. In the MAPK and cell-fate networks, TS is able to identify all the control strategies identified by SM and BA and, in some cases, new control strategies missed by the other non-exhaustive methods. Only CN and MC are able to identify the two minimal control strategies of size 5 in the MAPK network. They both fixed the four inputs to the variables fixed in the attractor plus an additional component (see Table 5.8).

The running times for each method are shown in Table 5.9. They include the time for reading the network, computing the attractors or minimal trap spaces and identifying the control strategies. Note that while BA and TS allow the selection of a specific target (attractor in case of BA, subspace in case of TS), SM computes the control strategies for all the attractors in one run. For that reason, we observe that the running times of SM are significantly higher than the ones for BA and TS. Note that CN and MC take longer time than other approaches, except SM for the first network. In all the cases we see that the running times for DP, TS and CO are lower than for the rest of the methods.

Table 5.7: Number and size of the control strategies for the corresponding attractor identified by the different methods up to sizes 6, 5 and 2 (number of free inputs plus two) for the networks MAPK, cell-fate and T-LGL respectively. Note that no smaller control strategies are obtained by any method.

	MAPK network		**Cell-fate network**		**T-LGL network**	
	$\|\mathcal{N}\| \leq 5$	$\|\mathcal{N}\| = 6$	$\|\mathcal{N}\| \leq 4$	$\|\mathcal{N}\| = 5$	$\|\mathcal{N}\| = 1$	$\|\mathcal{N}\| = 2$
SM	0	16	0	16	3	0
BA	-	-	0	15	4	0
DP	0	14	0	22	0	0
TS	0	22	0	22	3	5
CO	0	22	0	22	3	5
CN	2	0	0	22	4	0
MC	2	0	0	22	4	0

Table 5.8: Minimal control strategies up to sizes 5 and 2 (number of free inputs plus two) for the networks MAPK and T-LGL respectively, for the selected attractor of each biological network. ✓ and − denote whether the control strategy is obtained by the method or not, respectively. For simplicity, the interventions that fix the input components to the values of the attractor are omitted.

Network	**Minimal CS**	**SM**	**BA**	**DP**	**TS**	**CO**	**CN**	**MC**
MAPK	{fixed inputs + (GAB1, 0)}	-	-	-	-	-	✓	✓
	{fixed inputs + (PI3K, 0)}	-	-	-	-	-	✓	✓
T-LGL	{(Ceramide, 1)}	-	✓	-	-	-	✓	✓
	{(PDGFR, 0)}	✓	✓	-	✓	✓	✓	✓
	{(S1P, 0)}	✓	✓	-	✓	✓	✓	✓
	{(SPHK1, 0)}	✓	✓	-	✓	✓	✓	✓

5.2.2 Target control

We now compare all the approaches for target control described above (DP, TS, CO, CN and MC). An overview of the features of each control method is shown in Table 5.5. The node control strategies for the different target subspaces are computed for the three dynamics. However, since the control strategies obtained for the generalized asynchronous dynamics are the same as the ones obtained for the asynchronous dynamics, we focus our comparison just in asynchronous and synchronous dynamics.

Table 5.9: Running times for the control strategy computation targeting the chosen attractor. Note that the time for SM includes the computation of the control strategies for all the attractors, since the method does not allow to select a unique target.

	Running times		
Method	Cell-fate	MAPK	T-LGL
SM	4h54'58"	3'14"	5'56"
BA	3h30'52"	-	2'10"
DP	1.4"	1.9"	1.2"
TS	2.4"	2.2"	1.4"
CO	2.4"	2.2"	1.4"
CN	1'35"	13'18"	29'1"
MC	1'21"	12'15"	33'11"

The target subspaces chosen for each network are the ones corresponding to the apoptotic phenotype, a common target in drug identification studies for cancer therapeutic treatments [CED20]. They are defined in terms of the output components of each network:

(a) **T-LGL:** {Apoptosis = 1, Proliferation = 0}.

(b) **MAPK:** {Apoptosis = 1, Proliferation = 0, Growth-Arrest = 1}.

(c) **Cell-Fate:** {Apoptosis = 1, Survival = 0, NonACD = 0}.

Table 5.10 contains the size and number of the control strategies computed by each approach for the asynchronous and synchronous dynamics. As mentioned in the previous section, although all the methods look for minimal control strategies under inclusion, only the exhaustive approach is able to guarantee that no smaller control strategy exists. Thus, the control strategies identified by non-exhaustive methods might not be minimal with respect to all the strategies. The numbers in parenthesis in Table 5.10 denote the amount of truly minimal control strategies obtained by each method. In some cases, DP is able to identify a high number of minimal control strategies. This is not surprising, since regulatory networks modeling biological systems often induce a large amount of value propagation. In particular, all the minimal control strategies for the cell-fate network in the asynchronous dynamics are control strategies by direct percolation. Nonetheless, in the other two networks the number of strategies identified by DP is still far from the number of minimal control strategies identified by MC.

Control strategies are update-dependent by definition. However, all the control strategies identified by DP are valid in all the dynamics considered here. The number of these control strategies that are minimal might vary from one update to another (see results for T-LGL and MAPK networks). On the other hand, methods TS, CN and MC are sensitive

Table 5.10: Number and size of the control strategies identified by each method (DP, TS, CO, CN, and MC) for the corresponding target subspace of each biological network in the asynchronous and synchronous dynamics. The numbers in parenthesis denote the amount of truly minimal control strategies obtained by each method.

		Asynchronous			**Synchronous**		
Network	**Method**	$\|\mathcal{N}\| = 1$	$\|\mathcal{N}\| = 2$	$\|\mathcal{N}\| = 3$	$\|\mathcal{N}\| = 1$	$\|\mathcal{N}\| = 2$	$\|\mathcal{N}\| = 3$
Cell-Fate	DP	0	17	173	0	17	173
	TS	0	8	28	0	8	28
	CO	0	17	173	0	17	173
	CN	0	21	191	0	17	189
	MC	0	21	191	0	17	189
T-LGL	DP	0	224 (77)	327 (77)	0	224 (164)	327 (97)
	TS	3	5	7	0	20	28
	CO	3	195 (77)	282 (77)	0	224 (164)	327 (97)
	CN	10	116	204	0	232 (172)	762 (109)
	MC	10	116	204	3	251	261
MAPK	DP	2	124 (59)	175 (45)	2	124 (88)	175 (49)
	TS	2	0	0	1	0	0
	CO	3	106 (59)	162 (45)	2	124 (88)	175 (49)
	CN	8	105	66	2	164 (112)	195 (155)
	MC	8	105	66	4	118	216

to the update. In the case of TS, since none of the networks is complete in the synchronous dynamics, attractors cannot be approximated by minimal trap spaces and, therefore, no additional control strategies compared to DP are obtained. The methods CN and MC obtain the same number of control strategies for the asynchronous dynamics. This is not the case for the synchronous dynamics, where additional control strategies are obtained by MC for the T-LGL and MAPK networks. As mentioned in Section 4.5, this might be caused by the fact that the CN method cannot classify a subspace as a control strategy if the restricted network is not complete. The CN method is likely to obtain better results when the original network is complete, which is usually the case for the asynchronous dynamics of biological networks [KS15]. As to the running times, direct percolation, via trap spaces and the combined method finish in a few seconds, whereas the completeness approach and the model checking take several hours.

The results presented in this section illustrate the main differences between the two approaches presented in this work. The first one provides a limited result in a short amount of time, suitable for a fast analysis of the network, whereas the second one provides an exhaustive enumeration of all the possible controls at the price of a higher computational time, ideal for an in-depth analysis of the network.

Chapter 6

Discussion

The work presented in this thesis deals with the study of the control problems in biological systems, in the context of Boolean networks. In particular, it focuses on the development of approaches to identify sets of minimum controls that are able to induce the desired states in Boolean models of biological systems.

Control of biological systems encompasses a great variety of scenarios and goals. Our first task was to study and understand the multiple factors defining a control problem, from the definition of the control goal to the strategy type, as well as the role of the initial state of the system and the type of perturbations. Once the control problem was defined, our next step was to decide how to approach it. With the goal of making our approaches attractive for application, we established two key factors: efficiency and diversity. We were interested in approaches that are able to deal with state-of-the-art networks in a reasonable amount of time. We also aimed at providing as many different control sets as possible to broaden the possibilities for potential applications. Thus, we developed two different approaches, each of them aiming at one of these factors.

We started by studying methods based on value percolation, which is one of the most efficient approaches to control. However, these methods are usually limited and might miss many control strategies. With the aim of increasing the number of control strategies identified while still benefiting from the efficient implementation of value percolation, we introduced the use of trap spaces (Chapter 3). The main idea of our first approach is to compute the so-called selected trap spaces, trap spaces that contain only attractors included in the desired target, and consider intervention sets that percolate to these trap spaces. Although this approach is not able to identify all the minimal control strategies for a given target, it is able to uncover, in some cases, relevant control strategies missed by direct-percolation techniques, as shown in Section 3.4. Its efficient implementation makes it a competitive and practical tool when looking for a fast and useful analysis of the network.

Our second approach focuses on exhaustivity and flexibility. It is based on model checking, using CTL semantics, and allows us to identify all the minimal control strategies for a given target (Chapter 4). It also provides maximum flexibility in the definition of the control target, accepting any arbitrary subset as target. This allows us to deal with more complex control goals beyond the usual attractor or target control, such as groups of attractors or attractor avoidance (see Section 5.1.3). The exhaustivity and flexibility of this approach make it very appealing when a more in-depth analysis of the model is desired. To deal with the high computational costs associated to the exhaustive exploration, we developed reduction methods to narrow the region of the state space to search and to discard ineffective interventions a priori.

We studied the applicability of our two approaches in different biological networks (Sections 3.4 and 4.5 and Chapter 5), showing that both approaches are able to deal with state-of-the-art biological models. We compared our approaches to other available control methods in terms of control strategies identified and running times. As expected, the approaches by direct percolation and via trap spaces were the most efficient ones while the model-checking approach was the one identifying more minimal control strategies. In some cases, direct percolation was able to identify all minimal control strategies while in others it could not identify any of them. The examples of Chapter 5 illustrate how the first approach is more suitable for a fast and useful analysis of the network, whereas the second one is ideal for an in-depth analysis of the biological model.

The scalability of the ASP implementation proposed by Kaminski et al., which we extended to deal with trap spaces and edge control, was also studied in [KSSV13]. Further experiments would be required to fully evaluate the scalability of our extended implementation. Factors such as the size of the network, the number of edges, prime implicants, trap spaces or other topological properties of the network, would require further analyses to determine the impact that they can potentially have in the computational times of our ASP implementation. In some cases, the control problem could be run in parallel, for instance when the number of trap spaces is too high for the ASP solver to handle. This was the case in one of the mesenchymal phenotypes in Section 5.1.2, whose control problem required the analysis of over 30000 selected trap spaces.

Our second approach would also benefit from a scalability study in order to rigorously determine the main factors affecting its computational time and develop reduction methods aiming at mitigating their impact. Additionally, the model checking implementation could potentially benefit from the use of the so-called Action Restricted CTL (arCTL) [LPR07] that allows for a restriction on the paths with a given action formula. This extension is particularly useful for the verification of properties in models containing many input variables [MAJTC14]. A potential improvement would be adapting the encoding of the control problem to include the input variables as restrictions on the paths, since it would allow us to reduce the dimension of the explored network. In this direction, we could explore the possibility of using arCTL to encode edge interventions directly on the transitions.

One of the main challenges of implementing edge control in an efficient way is the large amount of slightly different Boolean networks that our method needs to verify. Being able to deal with an ensemble of such networks in order to check multiple edge interventions simultaneously could significantly reduce the overall computational cost. Moreover, the study of the interaction graph or the prime implicants of the controlled network could help us to predict the potential effect of edge interventions [MD15, GCWR21], allowing for instance the identification of irrelevant interventions or equivalent intervention sets.

As the EMT network studied in Section 5.1, many biological models often include discrete multi-valued variables on top of the usual Boolean modeling [NCCT10, RRC$^+$15]. Multi-valued variables can be used to represent different interactions happening at distinct activity levels [NCCT10], to capture multiple levels of the model outputs [SCP$^+$20, FBR$^+$15] or to represent concentration gradients [GCT08]. Multi-valued networks can be booleanised preserving their long-term dynamics [DRC11] and consequently can be analyzed by Boolean-aimed control tools (see Section 5.1). However, booleanisation often introduces non-admissible states, causing the state space to become larger than necessary, and could lead to potential misinterpretations of results when translating the interventions identified back to the original network. Consequently, having control methods directly dealing with multi-valued networks would help to improve their efficiency. The methods for trap space computation and value percolation are easily extendable to multi-valued networks, as well as the corresponding ASP encoding. Furthermore, extensions of ASP that can deal with multi-valued networks have been developed [MSH$^+$18] and model checking has already been successfully applied to the multi-valued framework [AJMN$^+$15]. Yet, since more activity levels could significantly increase the computational time, methods for dimensionality reduction and performance improvement tailored to our control problem should also be developed.

Although the control methods presented in this work are primarily aimed at the asynchronous dynamics, they are also valid for synchronous and generalized asynchronous updates. An interesting addition to this work would be to study how our approaches would deal with other updates, for instance block-sequential [DS20] or most permissive dynamics [PKCH20], and how the control strategies would differ from the ones obtained here. Other potential additions to this work would be its extension to consider different initial states in the control problem or sequential control, as mentioned in Section 2.2.

The work presented here studies control in Boolean networks by developing two competitive approaches for control strategy identification. The efficiency of the first one makes it appropriate for a fast and practical analysis, while the exhaustivity and flexibility of the second are suitable for a more in-depth analysis. In addition, both methods have enough versatility to be extended to other control problems and frameworks. We think that our approaches can help providing deeper insights into biological systems and contribute to broaden the potential control applications.

Bibliography

[AJMN+15] Wassim Abou-Jaoudé, Pedro T. Monteiro, Aurélien Naldi, Maximilien Grandclaudon, Vassili Soumelis, Claudine Chaouiya, and Denis Thieffry. Model checking to assess t-helper cell plasticity. *Frontiers in Bioengineering and Biotechnology*, 2, 2015.

[Aku18] Tatsuya Akutsu. *Algorithms for Analysis, Inference, and Control of Boolean Networks*. WORLD SCIENTIFIC, 2018.

[BAFRM17] Emna Ben Abdallah, Maxime Folschette, Olivier Roux, and Morgan Magnin. Asp-based method for the enumeration of attractors in non-deterministic synchronous and asynchronous multi-valued networks. *Algorithms for Molecular Biology*, 12(20), 2017.

[BD19] C. Biane and F. Delaplace. Causal reasoning on Boolean control networks based on abduction: Theory and application to cancer drug discovery. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 16(5):1574–1585, 2019.

[BD21] E. Borriello and B. C. Daniels. The basis of easy controllability in boolean networks. *Nature Communications*, 12(5227), 2021.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.

[BPSS21] Lubos Brim, Samuel Pastva, David Safránek, and Eva Smijáková. Parallel one-step control of parametrised boolean networks. *Mathematics*, 9(5):560, 2021.

[CA19] Colin Campbell and Réka Albert. Edgetic perturbations to eliminate fixed-point attractors in boolean regulatory networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(2):023130, 2019.

[CED20] Benedito A. Carneiro and Wafik S. El-Deiry. Targeting apoptosis in cancer therapy. *Nature Reviews Clinical Oncology*, 17:395–417, 2020.

[CFa] Laura Cifuentes Fontanals. Source code of the implementation for control via trap spaces. *https://github.com/Lauracf/trap-space-control*.

[CFb] Laura Cifuentes Fontanals. Source code of the implementation for control with model checking. *https://github.com/Lauracf/model-checking-control*.

[CFTS20] Laura Cifuentes Fontanals, Elisa Tonello, and Heike Siebert. Control strategy identification via trap spaces in Boolean networks. In Alessandro Abate, Tatjana Petrov, and Verena Wolf, editors, *Computational Methods in Systems Biology*, pages 159–175, Cham, 2020. Springer International Publishing.

[CFTS22a] Laura Cifuentes Fontanals, Elisa Tonello, and Heike Siebert. Computing trap space-based control strategies for Boolean networks using answer set programming. *AIP Conference Proceedings*, 2611(1):110002, 2022.

[CFTS22b] Laura Cifuentes-Fontanals, Elisa Tonello, and Heike Siebert. Control in Boolean Networks With Model Checking. *Frontiers in Applied Mathematics and Statistics*, 8, 2022.

[CFTS22c] Laura Cifuentes-Fontanals, Elisa Tonello, and Heike Siebert. Node and edge control strategy identification via trap spaces in Boolean networks. *Preprint available at https://arxiv.org/abs/2203.13632*, 2022.

[CGR12] Miguel Carrillo, Pedro A. Góngora, and David Rosenblueth. An overview of existing modeling tools making use of model checking in the analysis of biochemical networks. *Frontiers in Plant Science*, 3:155, 2012.

[CNT12] Claudine Chaouiya, Aurélien Naldi, and Denis Thieffry. *Logical Modelling of Gene Regulatory Networks with GINsim.*, volume 804, pages 463–79. 2012.

[CTF+10] Laurence Calzone, Laurent Tournier, Simon Fourquet, Denis Thieffry, Boris Zhivotovsky, Emmanuel Barillot, and Andrei Zinovyev. Mathematical modelling of cell-fate decision in response to death receptor engagement. *PLOS Computational Biology*, 6(3):1–15, 2010.

[CZD+11] Benoit Charloteaux, Quan Zhong, Matija Dreze, Michael E. Cusick, David E. Hill, and Marc Vidal. *Protein–Protein Interactions and Networks: Forward and Reverse Edgetics*, pages 197–213. Humana Press, Totowa, NJ, 2011.

[DRC11] Gilles Didier, Elisabeth Remy, and Claudine Chaouiya. Mapping multivalued onto boolean dynamics. *Journal of Theoretical Biology*, 270(1):177–184, 2011.

[DS20] J. Demongeot and S. Sené. About block-parallel Boolean networks: a position paper. *Natural Computing*, 19:5–13, 2020.

[DT11] Elena Dubrova and Maxim Teslenko. A sat-based algorithm for finding attractors in synchronous boolean networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(5):1393–1399, 2011.

[DW19] Anushka Dongre and Robert A. Weinberg. New insights into the mechanisms of epithelial-mesenchymal transition and implications for cancer. *Nature Reviews Molecular Cell Biology*, (20):69–84, 2019.

[FBR+15] Åsmund Flobak, Anaïs Baudot, Elisabeth Remy, Liv Thommesen, Denis Thieffry, Martin Kuiper, and Astrid Lægreid. Discovery of drug synergies in gastric cancer cells predicted by logical modeling. *PLOS Computational Biology*, 11(8):1–20, 2015.

[FNCT06] Adrien Fauré, Aurélien Naldi, Claudine Chaouiya, and Denis Thieffry. Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle. *Bioinformatics*, 22(14):e124–e131, 2006.

[GCBP+13] Luca Grieco, Laurence Calzone, Isabelle Bernard-Pierrot, François Radvanyi, Brigitte Kahn-Perlès, and Denis Thieffry. Integrative modelling of the influence of MAPK network on cancer cell fate decision. *PLOS Computational Biology*, 9(10):1–15, 10 2013.

[GCT08] Aitor González, Claudine Chaouiya, and Denis Thieffry. Logical modelling of the role of the Hh pathway in the patterning of the Drosophila wing disc. *Bioinformatics*, 24(16):i234–i240, 2008.

[GCWR21] 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. *Proceedings of the National Academy of Sciences*, 118(12):e2022598118, 2021.

[GK73] Leon Glass and Stuart A. Kauffman. The logical analysis of continuous, non-linear biochemical control networks. *Journal of Theoretical Biology*, 39(1):103–129, 1973.

[GKK+11] Martin Gebser, Benjamin Kaufmann, Roland Kaminski, Max Ostrowski, Torsten Schaub, and Marius Schneider. Potassco: The potsdam answer set solving collection. *AI Commun.*, 24(2):107–124, 2011.

[GKKS12] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. *Answer Set Solving in Practice*. Morgan & Claypool Publishers, 2012.

[JPI20] Michalina Janiszewska, Marina Candido Primi, and Tina Izard. Cell adhesion in cancer: Beyond the migration of single cells. *Journal of Biological Chemistry*, 295:2495– 2505, 2020.

[Kau91] Stuart Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution*. 1991.

[KBS15] H. Klarner, A. Bockmayr, and H. Siebert. Computing maximal and minimal trap spaces of Boolean networks. *Natural Computing*, 14:535–544, 2015.

[KHNS18] H. Klarner, F. Heinitz, S. Nee, and H. Siebert. Basins of attraction, commitment sets and phenotypes of Boolean networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, page 1, 2018.

[KPC13] Junil Kim, Sang-Min Park, and Kwang-Hyun Cho. Discovery of a kernel for controlling biomolecular regulatory networks. *Scientific Reports*, 3:2223, 2013.

[KS15] Hannes Klarner and Heike Siebert. Approximating attractors of Boolean networks by iterative CTL model checking. *Frontiers in Bioengineering and Biotechnology*, 3:130, 2015.

[KSS16] Hannes Klarner, Adam Streck, and Heike Siebert. PyBoolNet: a Python package for the generation, analysis and visualization of Boolean networks. *Bioinformatics*, 33(5):770–772, 2016.

[KSSV13] Roland Kaminski, Torsten Schaub, Anne Siegel, and Santiago Videla. Minimal intervention strategies in logical signaling networks with ASP. *Theory and Practice of Logic Programming*, 13(4-5):675–690, 2013.

[KTF+21] Hannes Klarner, Elisa Tonello, Laura Fontanals, Florence Janody, Claudine Chaouiya, and Heike Siebert. Detection of markers for discrete phenotypes. In *The 12th International Conference on Computational Systems-Biology and Bioinformatics*, CSBio2021, page 64–68, New York, NY, USA, 2021. Association for Computing Machinery.

[Lif19] Vladimir Lifschitz. *Answer Set Programming*. Springer Nature Switzerland AG, 2019.

[LPR07] Alessio Lomuscio, Charles Pecheur, and Franco Raimondi. Automatic verification of knowledge and time with nusmv. pages 1384–1389, 01 2007.

[LSB11] Yang-Yu Liu, Jean-Jacques Slotine, and Albert-László Barabási. Controllability of complex networks. *Nature*, 473:167–173, 2011.

[MAJTC14] P.T. Monteiro, W. Abou-Jaoudé, D. Thieffry, and C. Chaouiya. Model checking logical regulatory networks. *IFAC Proceedings Volumes*, 47(2):170–175, 2014. 12th IFAC International Workshop on Discrete Event Systems (2014).

[MBT+22] Arnau Montagud, Jonas Béal, Luis Tobalina, Pauline Traynard, Vigneshwari Subramanian, Bence Szalai, Róbert Alföldi, László Puskás, Alfonso Valencia, Emmanuel Barillot, Julio Saez-Rodriguez, and Laurence Calzone. Patient-specific boolean models of signalling networks guide personalised treatments. *eLife*, 11:e72626, 2022.

[MD15] David Murrugarra and Elena S. Dimitrova. Molecular network control through boolean canalization. *EURASIP Journal on Bioinformatics and Systems Biology*, 2015(1):9, 2015.

[MSH+18] Mushthofa Mushthofa, Steven Schockaert, Ling-Hong Hung, Kathleen Marchal, and Martine De Cock. Modeling multi-valued biological interaction networks using fuzzy answer set programming. *Fuzzy Sets and Systems*, 345:63–82, 2018.

[MSH+19] Hugues Mandon, Cui Su, Stefan Haar, Jun Pang, and Loïc Paulevé. Sequential reprogramming of Boolean networks made practical. In Luca Bortolussi and Guido Sanguinetti, editors, *Computational Methods in Systems Biology*, volume 11773, pages 3–19, Cham, 2019. Springer International Publishing.

[MVCAL16] David Murrugarra, Alan Veliz-Cuba, Boris Aguilar, and Reinhard Laubenbacher. Identification of control targets in Boolean molecular network models via computational algebra. *BMC Systems Biology*, 10(1):94, 2016.

[NCCT10] Aurélien Naldi, Jorge Carneiro, Claudine Chaouiya, and Denis Thieffry. Diversity and plasticity of th cell types predicted from regulatory network modelling. *PLOS Computational Biology*, (9):1–16, 2010.

[PAM22] Daniel Plaugher, Boris Aguilar, and David Murrugarra. Uncovering potential interventions for pancreatic cancer patients via mathematical modeling. *Journal of Theoretical Biology*, 548:111197, 2022.

[PKCH20] Loïc Paulevé, Juraj Kolčák, Thomas Chatain, and Stefan Haar. Reconciling qualitative, abstract, and scalable modeling of biological networks. *Nature Communications*, 11:4256, 2020.

[RRC+15] Elisabeth Remy, Sandra Rebouissou, Claudine Chaouiya, Andrei Zinovyev, François Radvanyi, and Laurence Calzone. A Modeling Approach to Explain Mutually Exclusive and Co-Occurring Genetic Alterations in Bladder Tumorigenesis. *Cancer Research*, 75(19):4042–4052, 09 2015.

[RZG+21] 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. *Science Advances*, 7(29):eabf8124, 2021.

[SCP+20] Gianluca Selvaggio, Sara Canato, Archana Pawar, Pedro T. Monteiro, Patrícia S. Guerreiro, M. Manuela Brás, Florence Janody, and Claudine Chaouiya. Hybrid epithelial–mesenchymal phenotypes are controlled by microenvironmental factors. *Cancer Research*, 80(11):2407–2420, 2020.

[SKK10] Regina Samaga, Axel Von Kamp, and Steffen Klamt. Computing combinatorial intervention strategies and failure modes in signaling networks. *Journal of Computational Biology*, 17(1):39–53, 2010.

[SP20] Cui Su and Jun Pang. CABEAN: a software for the control of asynchronous Boolean networks. *Bioinformatics*, 37(6):879–881, 2020.

[SP21] Cui Su and Jun Pang. Target control of asynchronous boolean networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, pages 1–1, 2021.

[SPP19] Cui Su, Soumya Paul, and Jun Pang. Controlling large boolean networks with temporary and permanent perturbations. In Maurice H. ter Beek, Annabelle McIver, and José N. Oliveira, editors, *Formal Methods – The Next 30 Years*, volume 11800, pages 707–724, Cham, 2019. Springer International Publishing.

[SVLM20] L. Sordo Vieira, R.C. Laubenbacher, and D. Murrugarra. Control of intracellular molecular networks using algebraic methods. *Bulletin of Mathematical Biology*, 82(2), 2020.

[Tho81] R. Thomas. On the relation between the logical structure of systems and their ability to generate multiple steady states or sustained oscillations. In Jean Della Dora, Jacques Demongeot, and Bernard Lacolle, editors, *Numerical Methods in the Study of Critical Phenomena*, pages 180–193, Berlin, Heidelberg, 1981. Springer Berlin Heidelberg.

[WFT+15] Spencer C. Wei, Laurent Fattet, Jeff H. Tsai, , Yurong Guo, Vincent H. Pai, Hannah E. Majeski, Albert C. Chen, Robert L. Sah, Susan S. Taylor, Adam J. Engler, and Jing Yang. Matrix stiffness drives epithelial–mesenchymal transition and tumour metastasis through a twist1–g3bp2 mechanotransduction pathway. *Nature Cell Biology*, 17:678–688, 2015.

[WIS+22] Silke D. Werle, Nensi Ikonomi, Julian D. Schwab, Johann M. Kraus, Felix M. Weidner, K. Lenhard Rudolph, Astrid S. Pfister, Rainer Schuler, Michael Kühl, and Hans A. Kestler. Identification of dynamic driver sets controlling phenotypical landscapes. *Computational and Structural Biotechnology Journal*, 20:1603–1617, 2022.

[YGTZA18] Gang Yang, Jorge Gómez Tejeda Zañudo, and Réka Albert. Target control in logical models using the domain of influence of nodes. *Frontiers in Physiology*, 9:454, 2018.

[ZA13] 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):025111, 2013.

[ZA15] J. G. T. Zañudo and R. Albert. Cell fate reprogramming by control of intracellular network dynamics. *PLOS Computational Biology*, 11(4):1–24, 2015.

[ZSL+09] Quan Zhong, Nicolas Simonis, Qian-Ru Li, Benoit Charloteaux, Fabien Heuze, Niels Klitgord, Stanley Tam, Haiyuan Yu, Kavitha Venkatesan, Danny Mou, Venus Swearingen, Muhammed A Yildirim, Han Yan, Amélie Dricot, David Szeto, Chenwei Lin, Tong Hao, Changyu Fan, Stuart Milstein, Denis Dupuy, Robert Brasseur, David E Hill, Michael E Cusick, and Marc Vidal. Edgetic perturbation models of human inherited disorders. *Molecular Systems Biology*, 5(1):321, 2009.

[ZSY+08] Ranran Zhang, Mithun Vinod Shah, Jun Yang, Susan B. Nyland, Xin Liu, Jong K. Yun, Réka Albert, and Thomas P. Loughran. Network model of survival signaling in large granular lymphocyte leukemia. *Proceedings of the National Academy of Sciences*, 105(42):16308–16313, 2008.

[ZYA17] J.G.T. Zañudo, G. Yang, and R. Albert. Structure-based control of complex networks with nonlinear dynamics. *Proceedings of the National Academy of Sciences*, 114(28):7234–7239, 2017.

Acknowledgments

First of all, I would like to thank my supervisor, Heike Siebert, for her support, encouragement and guidance through all of my doctoral studies.

I would like to express my gratitude to Elisa Tonello for her support, useful discussions and invaluable mathematical insights.

I also thank the other wonderful members of the Discrete Biomathematics group, Melania Nowicka, Hannes Klarner and Robert Schwieger, for all the discussions, shared moments and support during the different phases of my PhD.

I would like to thank Alexander Bockmayr and Martin Vingron for the useful insights, encouragement and support provided during TAC meetings. I would also like to thank Claudine Chaouiya, Florence Janody, Denis Thieffry and Élisabeth Remy for many useful discussions and valuable biological and mathematical insights.

Many thanks to all the colleagues and friends that have accompanied me during this journey. Last but not least, I thank my family and my partner for their continuous support along the way.

Summary

Understanding control mechanisms present in biological processes is crucial for the development of potential therapeutic applications, for instance cell reprogramming or drug target identification. Experimental approaches aimed at identifying possible control targets are usually costly and time-consuming. Mathematical modeling provides a formal framework to study biological systems and to predict potential successful candidate interventions. A common modeling framework is Boolean modeling, which stands out for its ability to capture the qualitative behavior of the system using coarse representations of the interactions between the components, overcoming the usual parametrization problem.

The main goal of this thesis is the study of the control problems present in biological systems and the development of efficient and complete approaches for control strategy identification. In particular, we aim at developing methods to identify sets of minimal controls that are able to induce the desired states in biological systems modeled by Boolean networks. With the goal of making our approaches attractive for application, we establish two key factors: efficiency and diversity. We want our approaches to be able to deal with state-of-the-art networks in a reasonable amount of time while providing as many different optimal control sets as possible. With these factors in mind, we developed two different approaches.

Our first method is based on value percolation, one of the most simple and efficient approaches to control strategy identification in Boolean networks. Percolation-based methods can be implemented efficiently but are limited and might miss many control strategies. Our approach introduces the use of trap spaces, regions of the state space closed under the dynamics. This allows us to increase the number of control strategies identified while still benefiting from an efficient implementation. Our second approach focuses on exhaustivity and flexibility. Based on model checking techniques, it allows us to identify all the minimal control strategies for a given target. This approach is also able to deal with more complex control problems, since it can handle any type of target. To overcome the higher computational costs associated with the comprehensiveness of the method, we also introduce several reduction techniques to improve its performance.

In the last chapter, we show the applicability of our approaches to different biological systems. We study the control strategies obtained for a network modeling the epithelial-to-mesenchymal transition, considering different control targets and types of interventions. We also explore the relevance of the intervention strategies identified in the biological context. Finally, we compare our approaches to other current control methods in different Boolean networks.

Zusammenfassung

Das Verständnis von Kontrollmechanismen in biologischen Prozessen ist von entscheidender Bedeutung für die Entwicklung potenzieller therapeutischer Anwendungen, z. B. die Reprogrammierung von Zellen oder die Identifizierung von Zielstrukturen für Medikamente. Experimentelle Ansätze zur Identifizierung möglicher Kontrollziele sind in der Regel kostspielig und zeitaufwändig. Die mathematische Modellierung bietet einen formalen Rahmen zur Untersuchung biologischer Systeme und zur Vorhersage potenziell erfolgreicher Interventionskandidaten. Ein etablierter Formalismus ist die boolesche Modellierung, die sich durch ihre Fähigkeit auszeichnet, das qualitative Verhalten des Systems mit Hilfe grober Darstellungen der Wechselwirkungen zwischen den Komponenten zu erfassen und so das übliche Parametrisierungsproblem zu überwinden.

Das Hauptziel dieser Arbeit ist die Untersuchung der Kontrollprobleme in biologischen Systemen und die Entwicklung von effizienten und vollständigen Ansätzen zur Identifikation von Kontrollstrategien. Insbesondere geht es um die Entwicklung von Methoden zur Identifizierung von Mengen minimaler Steuerungen, die in der Lage sind, die gewünschten Zustände in biologischen, durch boolesche Netzwerke modellierten Systemen zu induzieren. Um unsere Ansätze für die Anwendung attraktiv zu machen, legen wir zwei Schlüsselfaktoren fest: Effizienz und Vielfalt. Unsere Methoden sollen in der Lage sein, biologische Netzwerke von aktuellem Interesse in angemessener Zeit zu bearbeiten und dabei so viele verschiedene optimale Kontrollsätze wie möglich bereitzustellen. Mit Blick auf diese Faktoren haben wir zwei verschiedene Ansätze entwickelt.

Unsere erste Methode basiert auf der Wertperkolation, einem der einfachsten und effizientesten Ansätze zur Berechnung von Steuerungen boolescher Netze. Auf Perkolation basierende Methoden können zwar effizient implementiert werden, lassen aber möglicherweise viele Kontrollstrategien außer Acht. Unser Ansatz führt die Verwendung von Trap-Spaces ein, d.h. Regionen des Zustandsraums, die unter der Dynamik abgeschlossen sind. Dadurch können wir die Anzahl der identifizierten Kontrollstrategien erhöhen und gleichzeitig von einer effizienten Implementierung profitieren. Unser zweiter Ansatz konzentriert sich auf Vollständigkeit und Flexibilität. Auf der Grundlage von Modellprüfungstechniken können wir alle minimalen Kontrollstrategien für ein bestimmtes Ziel identifizieren. Dieser Ansatz ist auch in der Lage, komplexere Steuerungsprobleme zu behandeln, da er mit jeder Art von Ziel umgehen kann. Um die mit der Vollständigkeit der Methode verbundenen höheren Rechenkosten zu überwinden, führen wir mehrere leistungsverbessernde Reduktionstechniken ein.

Im letzten Kapitel zeigen wir die Anwendbarkeit unserer Ansätze auf verschiedene biologische Systeme. Wir untersuchen die Kontrollstrategien, die wir für ein Netzwerk erhalten, das den Übergang von Epithel- zu Mesenchymzellen modelliert, wobei wir verschiedene Kontrollziele und Arten von Eingriffen berücksichtigen. Wir untersuchen auch die Relevanz der ermittelten Interventionsstrategien im biologischen Kontext. Schließlich vergleichen wir unsere Ansätze mit anderen aktuellen Kontrollmethoden angewandt auf verschiedene boolesche Netzwerke.

Erklärung

Selbstständigkeitserklärung

Name: Cifuentes Fontanals

Vorname: Laura

Ich erkläre gegenüber der Freien Universität Berlin, dass ich die vorliegende Dissertation selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe. Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht. Diese Dissertation wurde in gleicher oder ähnlicher Form noch in keinem früheren Promotionsverfahren eingereicht.

Mit einer Prüfung meiner Arbeit durch ein Plagiatsprüfungsprogramm erkläre ich mich einverstanden.

Datum: 27.10.2022

Declaration of authorship

Name: Cifuentes Fontanals

First name: Laura

I declare to the Freie Universität Berlin that I have completed the submitted dissertation independently and without the use of sources and aids other than those indicated. The present thesis is free of plagiarism. I have marked as such all statements that are taken literally or in content from other writings. This dissertation has not been submitted in the same or similar form in any previous doctoral procedure.

I agree to have my thesis examined by a plagiarism examination software.

Date: 27.10.2022