frontiers in Bioengineering and Biotechnology

ORIGINAL RESEARCH
published: 08 September 2015
doi: 10.3389/fbioe.2015.00130

Approximating attractors of Boolean networks by iterative CTL model checking

Hannes Klarner and Heike Siebert*

Fachbereich Mathematik und Informatik, Freie Universität Berlin, Berlin, Germany

This paper introduces the notion of approximating asynchronous attractors of Boolean networks by minimal trap spaces. We define three criteria for determining the quality of an approximation: "faithfulness" which requires that the oscillating variables of all attractors in a trap space correspond to their dimensions, "univocality" which requires that there is a unique attractor in each trap space, and "completeness" which requires that there are no attractors outside of a given set of trap spaces. Each is a reachability property for which we give equivalent model checking queries. Whereas faithfulness and univocality can be decided by model checking the corresponding subnetworks, the naive query for completeness must be evaluated on the full state space. Our main result is an alternative approach which is based on the iterative refinement of an initially poor approximation. The algorithm detects so-called autonomous sets in the interaction graph, variables that contain all their regulators, and considers their intersection and extension in order to perform model checking on the smallest possible state spaces. A benchmark, in which we apply the algorithm to 18 published Boolean networks, is given. In each case, the minimal trap spaces are faithful, univocal, and complete, which suggests that they are in general good approximations for the asymptotics of Boolean networks.

Keywords: Boolean networks, asynchronous dynamics, attractors, CTL model checking, ASP, signaling, gene regulation

Edited by:
David A. Rosenblueth, Universidad Nacional Autónoma de México, Mexico

Reviewed by:
Stalin Muñoz, Universidad Nacional Autónoma de México, Mexico
Sepinoud Azimi, Åbo Akademi University, Finland

****Correspondence:***
Hannes Klarner, Fachbereich Mathematik und Informatik, Freie Universität Berlin, Arnimallee 6, Berlin 14195, Germany
hannes.klarner@fu-berlin.de

Specialty section:
This article was submitted to Bioinformatics and Computational Biology, a section of the journal Frontiers in Bioengineering and Biotechnology

Received: *30 June 2015*
Accepted: *14 August 2015*
Published: *08 September 2015*

Citation:
Klarner H and Siebert H (2015) Approximating attractors of Boolean networks by iterative CTL model checking. Front. Bioeng. Biotechnol. 3:130. doi: 10.3389/fbioe.2015.00130

1. Introduction

Boolean and multi-valued networks are frequently used to model the dynamics of biological processes that involve gene regulation and signal transduction. The dynamics of such models is captured by the state transition graph, a directed graph that relates states to potential successor states. Different transition relations have been suggested, among them the synchronous update of Kauffman (1993) and the asynchronous update of Thomas (1991). An important type of prediction that can be obtained from such models concerns the long-term behavior of the represented processes. Formally, the long-term behaviors correspond to the minimal trap sets of the state transition graph which are also called its attractors.

Recently, we have suggested to compute the minimal trap spaces of a network to obtain an approximation for its cyclic attractors (Klarner et al., 2014) and proposed an efficient, *Answer Set Programing* (ASP)-based method for their computation. This paper presents an iterative algorithm that combines *Computation Tree Logic* (CTL) model checking with the computation of minimal trap spaces to determine the quality of the approximation.

The paper is organized as follows. Section 2 recapitulates the background including directed graphs, the dynamics of Boolean networks, trap spaces, and model checking. It is only meant to introduce the notation required for the subsequent sections. Section 3 briefly discusses the attractor detection problem. In Section 4, we describe three conditions under which there is a one-to-one correspondence between the minimal trap spaces and the attractors of a network, and how CTL queries may be used to decide whether they hold. The computationally most challenging one is treated in Section 5. In Section 6, we present a full analysis of a MAPK signaling network as well as the results for 18 Boolean models that are currently in the GINSIM repository. Section 7 is an outlook and conclusion. There is a Supplementary Material that contains proofs for the formal statements in the main text.

2. Background

2.1. Directed Graphs

Since several aspects of Boolean networks involve directed graphs (digraphs) we introduce the general terminology. Let (V, A) be a digraph with vertices V and arcs $A \subseteq V \times V$.

An **infinite path** in (V, A) is an infinite sequence of vertices $\pi = (v_0, v_1, \ldots)$ such that $(v_i, v_{i+1}) \in A$ for all $i \in \mathbb{N}_0$. **Finite paths** are defined analogously for finite sequences. In particular, $\pi = (v_0)$ is an admissible finite path. We denote the set of all infinite paths that start in $v \in V$ by $InfPaths(v)$ and finite paths by $FinPaths(v)$. The ith vertex of π is denoted by $\pi[i] := v_i$. For finite paths we denote by $FinPaths(u, v)$ all finite paths that start with u and end with v. The number of vertices in a finite path $\pi = (v_0, v_1, \ldots, v_k)$ is denoted by $len(\pi) := k + 1$.

A vertex $v \in V$ is **reachable** from $u \in V$ iff $FinPaths(u, v) \neq \varnothing$. We denote by $Above(v)$ the vertices that can reach v. A subset $U \subseteq V$ is **strongly connected** if every $u \in U$ is reachable from every other $v \in U$. A **strongly connected component** (SCC) is an inclusion-wise maximal subset $U \subseteq V$ that is strongly connected. We denote the set of SCCs of a digraph by $SCCs(V, A)$. Note that since $\pi = (v_0)$ is an admissible finite path, every vertex is trivially reachable from itself. Hence, each node belongs to some SCC and $SCCs(V, A)$ is a partition of V.

2.2. Boolean Networks

We consider variables from the Boolean domain $\mathbb{B} = \{0,1\}$, where 1 and 0 represent the truth values *true* and *false*. A Boolean **expression** f over the variables $V = \{v_1, \ldots, v_n\}$ is defined by a formula over the grammar

$$f ::= 0 \mid 1 \mid v \mid \bar{f} \mid f_1 \cdot f_2 \mid f_1 + f_2$$

where $v \in V$ signifies a variable, $\bar{f}$ the negation, $f_1 \cdot f_2$ the conjunction and $f_1 + f_2$ the (inclusive) disjunction of the expressions f, f_1, and f_2. Given an **assignment** $x : V \to \mathbb{B}$, an expression f can be evaluated to a value $f(x) \in \mathbb{B}$ by substituting the values $x(v)$ for the variables $v \in V$. If $f(x) = f(y)$ for all assignments $x, y : V \to \mathbb{B}$, we say f is **constant** and write $f = c$, with $c \in \mathbb{B}$ being the constant value. A **Boolean network** (V, F) consists of n variables $V = \{v_1, \ldots, v_n\}$ and n corresponding Boolean expressions $F = \{f_1, \ldots, f_n\}$ over V. In this context, an assignment $x : V \to \mathbb{B}$ is also called a **state** of the network and the **state space** $S = S_V$ consists of all possible 2^n states. We specify states by a sequence of n values that correspond to the variables in the order given in V, i.e., $x = 110$ should be read as $x(v_1) = 1$, $x(v_2) = 1$, and $x(v_3) = 0$. The expressions F can be thought of as a function $F : S \to S$ governing the network behavior. The **image** $F(x)$ of a state x under F is defined to be the state y that satisfies $y(v_i) = f_i(x)$.

The **interaction graph** of a network (V, F) captures the dependencies between the variables and their expressions. It is a digraph $(V, \to)$ where $\to \subseteq V \times V$ and $(u, v) \in \to$ iff there are $x, y \in S$ such that $x(w) = y(w)$ for all $w \in V \setminus \{u\}$ and $f_v(x) \neq f_v(y)$. As for state transitions we write $u \to v$ iff $(u,v) \in \to$.

The **state transition graph** (STG) of a Boolean network (V, F) is the digraph $(S, \to)$ where the transitions $\to \subseteq S \times S$ are obtained from F via a given update rule. We usually write $x \to y$ iff $(x, y) \in \to$. We mention two update rules here, the **synchronous rule** and its transition relation $\twoheadrightarrow \subseteq S \times S$, and the **asynchronous rule** and its transition relation $\hookrightarrow \subseteq S \times S$. The former is defined by $x \twoheadrightarrow y$ iff $F(x) = y$. To define $\hookrightarrow$ we need the Hamming distance $\Delta : S \times S \to \{1, \ldots, n\}$ between states which is given by $\Delta(x, y) := |\{v \in V \mid x(v) \neq y(v)\}|$. We define $x \hookrightarrow y$ iff either $x = y$ and $F(x) = x$ or $\Delta(x,y) = 1$ and $\Delta(y, F(x)) < \Delta(x, F(x))$. In the context of the STG, the expressions $f \in F$ are also called **update functions**.

A non-empty set $T \subseteq S$ is a **trap set** of $(S, \to)$ iff for every $x \in T$ and $y \in S$ with $x \to y$ it holds that $y \in T$. An inclusion-wise minimal trap set is also called an **attractor** of $(S, \to)$. Every trap set contains at least one minimal trap set and therefore at least one attractor. A variable $v \in V$ is **steady** in an attractor $A \subseteq S$ iff $x(v) = y(v)$ for all $x, y \in A$ and **oscillating** otherwise. We distinguish two types of attractors depending on their size. If $A \subseteq S$ is an attractor and $|A| = 1$, then A is called a **steady state** and if $|A| > 1$, we call it a **cyclic attractor**. The cyclic attractors of $(S, \twoheadrightarrow)$ are, in general, different from the cyclic attractors of $(S, \hookrightarrow)$. The steady states, however, are identical in both transition graphs because $x \in S$ is steady iff $x \to x$ which is characterized, for both update rules, by the equation $F(x) = x$. Hence, we may omit the update rule and denote the set of steady states by S_F.

A **subspace** of S is characterized by its fixed and free variables. It may be specified by an assignment $p : D \to \mathbb{B}$ where $D \subseteq V$ is the subset of **fixed** variables, $p(u)$ the value of $u \in D$ and the remaining variables, $V \setminus D$, are said to be **free**. Subspaces are sometimes referred to as "symbolic states" (Siebert, 2011) or "partial states" (Irons, 2006). We specify subspaces like states but allow in addition the symbol $\star$ to indicate that a variable is free, i.e., $p = {\star\star}10$ means $D = \{v_3, v_4\}$ and $p(v_3) = 1$, $p(v_4) = 0$. The set $S^\star = S_V^\star$ denotes all possible 3^n subspaces. States are therefore a special kind of subspace and $S \subset S^\star$ holds. We denote the fixed variables D of a specific $p \in S^\star$ by D_p. A subspace p references the states $S[p] := \{x \in S \mid \forall v \in D_p : x(v) = p(v)\}$. We denote the unique subspace that does not fix any variables by $\epsilon \in S^\star$, i.e., $D_\epsilon = \varnothing$. Two subspaces $p, q \in S^\star$ are said to be **consistent** iff $p(v) = q(v)$ for all $v \in D_p \cap D_q$. We define the **intersection** $z := q \sqcap p$ of two consistent $p, q \in S^\star$ to be the unique $z \in S^\star$ that satisfies $S[z] = S[p] \cap S[q]$.

A **trap space** is a subspace that is also a trap set. Trap spaces are therefore trap sets with a particularly simple geometry. They

generalize the notion of steadiness from states to subspaces. In Klarner et al. (2014), we proved that trap spaces are independent of the update strategy. It is therefore meaningful to denote the trap spaces of $(S, \hookrightarrow)$ by $S_F^\star$ independent of $\to$. If a network (V, F) satisfies $S_F^\star = \{\epsilon\}$, then we say it is **trap-space-free**. We also showed that the dynamics inside a trap space p is fully specified by the **reduced network** (V_p, F_p) with

$$V_p := \{v \in V \,|\, v \notin D_p\}, \quad F_p := \{f_i[p] \,|\, f_i \in F : v_i \notin D_p\}$$

where $f[p]$ denotes the Boolean expression that is obtained by substituting the values $p(v)$ for $v \in D_p$ into $f \in F$, as introduced in Section 2.1 of Klarner et al. (2014).

Since every trap set contains at least one attractor, inclusion-wise minimal trap spaces can be used to predict the location of a particular attractor. Hence, we define a partial order on $S^\star$ based on whether the referenced subspaces are nested: $p \leq q$ iff $S[p] \subseteq S[q]$. The **minimal** trap spaces are defined by $\min(S_F^\star) := \{p \in S_F^\star \,|\, \nexists\, q \in S_F^\star : \; q < p\}$.

2.3. CTL Model Checking

Model checking is a formal method from computer science to determine whether a transition system satisfies a temporal specification. See Carrillo et al. (2012) for a review of its application to computational biology.

A **transition system** is a 5-tuple $TS = (S, \to, AP, L, I)$ where $(S, \to)$ is a state transition graph, AP a set of atomic propositions, $L : S \to 2^{AP}$ a labeling function and $I \subseteq S$ a set of initial states. We use the atomic propositions $AP := \{v = c, \delta_v = d \mid v \in V, c \in \mathbb{B}, d \in \{-1, 0, 1\}\}$ and define the labeling function L by

$$v = c \in L(x) :\Leftrightarrow x(v) = c$$
$$\delta_v = d \in L(x) :\Leftrightarrow f_v(x) - x(v) = d$$

for all $c \in \mathbb{B}$, $d \in \{-1, 0, 1\}$ and $x \in S$. The label $\delta_v = d$ therefore indicates whether a variable v is decreasing, steady or increasing in a state. In addition to "=" we need the inequality operator "$\neq$", e.g., $v \neq c \in L(x)$ iff $x(v) \neq c$, and the special atom *true* which satisfies $true \in L(x)$ for all $x \in$ S.

Next, we define a fragment of the temporal specification language CTL that is sufficient for the subsequent sections. A formula φ of this fragment is defined by

$$\varphi ::= a \;\Big|\; \varphi_1 \wedge \varphi_2 \;\Big|\; \varphi_1 \vee \varphi_2 \;\Big|\; \mathbf{EF}(\varphi) \;\Big|\; \mathbf{AG}(\varphi)$$

where $\alpha \in AP$, **EF** is the "exists finally" operator and **AG** the "always globally" operator. The semantics of the operators and the satisfaction relation $\models$ for transitions systems and CTL formulas is defined in **Table 1**. Since the atomic propositions and labeling function are fixed for the remainder of this article, we will specify transition systems by 3-tuples $TS = (S, \to, I)$. In practice, we use the model checking tool NUSMV (Cimatti et al., 2000) to decide whether a given transition system satisfies a CTL query.

TABLE 1 | The satisfaction relation $\models$ for CTL formulas φ, states $x \in S$, and transition systems $TS = (S, \to, AP, L, I)$.

$x \models a$	$:\Leftrightarrow$	$a \in L(x)$
$x \models \varphi_1 \wedge \varphi_2$	$:\Leftrightarrow$	$x \models \varphi_1$ and $x \models \varphi_2$
$x \models \varphi_1 \vee \varphi_2$	$:\Leftrightarrow$	$x \models \varphi_1$ or $x \models \varphi_2$
$x \models \mathbf{EF}(\varphi)$	$:\Leftrightarrow$	$\exists \pi \in InfPaths(x) : \exists i \in \mathbb{N}_0 : \pi[i] \models \varphi$
$x \models \mathbf{AG}(\varphi)$	$:\Leftrightarrow$	$\forall \pi \in InfPaths(x) : \forall i \in \mathbb{N}_0 : \pi[i] \models \varphi$
$TS \models \varphi$	$:\Leftrightarrow$	$\forall x \in I: x \models \varphi$

3. The Attractor Detection Problem

The naive approach to find all attractors of a given network, i.e., a full exploration of its STG, is limited by the state explosion problem. Several groups have developed tools and algorithms that address this problem. They may be grouped into those for deterministic updates (Irons, 2006; Dubrova and Teslenko, 2011; Akutsu et al., 2012; Veliz-Cuba et al., 2014) and non-deterministic updates (Garg et al., 2008; Skodawessely and Klemm, 2011; Berntenis and Ebeling, 2013). The average running times are usually given in terms of randomly generated networks and a connectivity parameter k that describes the distribution of in-degrees in the interaction graph. It seems that finding deterministic attractors is easier than non-deterministic attractors. Intuitively, computing the terminal SCCs of digraphs with all out-degrees equal to one is easier than for digraphs with higher out-degrees. The average running times for synchronous STGs with hundreds of variables is, for example, on the order of seconds with the tool BNS (Dubrova and Teslenko, 2011), which is based on a variant of *bounded linear time logic* (LTL) model checking and uses a *satisfiability* (SAT) solver to detect attractors.

Algorithms for non-deterministic STGs, on the other hand, are likely to run for hours or days for networks with less than even 100 variables (see Section 2). Garg et al. (2008) and the tool GENYSIS is based on the symbolic manipulation of reachable states using *binary decision diagrams* (BDDs), while Skodawessely and Klemm (2011) and Berntenis and Ebeling (2013) rely on a guided exploration and enumeration of the state space.

3.1. Attractor Detection Pre-Process

If $v \in V$ is a constant with $f_v = c$ and A an attractor, then $x(v) = c$ for every $x \in A$. Hence, before we start an attractor detection algorithm, we may safely remove all constants. The result is a reduced network whose attractors are in a one-to-one relationship with the attractors of the original network. During the removal of constants, update functions that depend on them may in turn become constant. The pre-process is therefore improved by an iterative substitution until there are no more constants.

The **percolation** operator $\vec{\bullet} : S_F^\star \to S_F^\star$ is defined on the set of trap spaces by the following recursion. Let p be the initial trap space, for example, defined by the constants $C \subseteq V$ of a network ($D_p := C$ and $p(v) := f_v$). The initial percolation is $\vec{p}_0 := p$ and for each $k \in \mathbb{N}_0$ we define $\vec{p}_{k+1}$ by

$$D_{\vec{p}_{k+1}} := \{v \in V \,|\, f_v[\vec{p}_k] \text{ is constant}\}$$
$$\vec{p}_{k+1}(v) := f_v[\vec{p}_k], \quad \text{for all } v \in D_{\vec{p}_{k+1}}.$$

Note that $f[p]$ denotes the Boolean expression obtained by substituting the values $p(v)$ into f, as introduced in Section 2.1 of Klarner et al. (2014). Because $\vec{p}_0 = p$ it follows that $\vec{p}_{k+1} \leq \vec{p}_k$ and $\vec{p}_k \in S_F^\star$, for all $k \in \mathbb{N}_0$. Since V is finite, there is some $K \in \mathbb{N}_0$ such that $\vec{p}_K = \vec{p}_{K+1}$ and $\vec{p} := \vec{p}_K$ is well-defined. Percolations

are cheap to compute and have the following implication for the location of attractors (see Siebert (2011)):

Proposition 1. If p is a trap space and $A \subseteq S[p]$ an attractor of $(S, \hookrightarrow)$, then $A \subseteq S[\vec{p}\,]$.

In the following sections, we will assume that the initial network is constant-free.

3.2. Attractor Detection by Random Walks

Given a trap space p, for example, the whole space $p = \epsilon$, we can find an attractor $A \subseteq S[p]$ by a sufficiently long random walk $(x_0, x_1, \ldots, x_k)$ where $x_0 \in S[p]$. In practice, we use $k = 10|V|$ and found that so far, without exception, random paths of this length have reached an attractor. To decide whether x_k does really belong to an attractor we use the CTL query of 2. It uses the CTL formula φ_p defined by $\varphi_p := \bigwedge_{v \in D_p} (v = p(v))$ if $p \neq \epsilon$, and $\varphi_p = ture$ otherwise.

Proposition 2 (Attractor State). Let p be a trap space and $x \in S[p]$. The state x belongs to an attractor $A \subseteq S[p]$ of $(S, \hookrightarrow)$ iff

$$TS = (S_{V_p}, \hookrightarrow, \{y\}) \models \mathbf{AG}(\mathbf{EF}(\varphi_y))$$

where $y \in S_{V_p}$ is the projection of $x \in S_V$ onto V_p, i.e., $y(v) := x(v)$ for all $v \in V_p$.

Starting from $x \in A$, we can then enumerate A by listing all states reachable from x. Note that model checking is performed on the reduced system $(S_{V_p}, \hookrightarrow)$ rather than the full system $(S, \hookrightarrow)$ and that there is no equivalent LTL query to decide whether x belongs to an attractor ($\mathbf{G}(\mathbf{F}(\varphi_y))$ does not work). Also, the observation that finding a single attractor is easy using a random walk does not contradict the fact that finding all attractors is hard.

4. Approximating Attractors by Subspaces

The result of attractor detection algorithms are usually sets of states that make up each attractor. The notion of an approximation of an attractor is instead based on information regarding steady and oscillating variables. An **approximation** of the attractors of a STG is a set $P \subseteq S^*$ such that each $S[p]$ contains an attractor. The trivial approximation for any network is $P := \{\epsilon\}$. Approximations differ in what can be learned from them about the number of attractors and their locations. The best approximation for a single attractor is the smallest subspace it is contained in. The **smallest subspace** that contains $A \subseteq S$ is $p \in S^*$ defined by $D_p := \{v \in V \mid \forall x, y \in A : x(v) = y(v)\}$ and $p(v) := x(v)$ for $x \in A$ arbitrary. We denote it by $Sub(A)$. Note that in general, $A \neq Sub(A)$ and that there may be two attractors $A, B \in S$ with $A \neq B$ such that $Sub(A) = Sub(B)$. The quality of an approximation is defined in terms of the following criteria.

Definition 1. A subspace p is **faithful** in $(S, \hookrightarrow)$ iff $Sub(A) = p$ for every attractor $A \subseteq S[p]$ of $(S, \hookrightarrow)$. An approximation P is faithful iff each $p \in P$ is faithful.

Definition 2. A subspace p is **univocal** in $(S, \hookrightarrow)$ iff there is a unique attractor A of $(S, \hookrightarrow)$ such that $A \subseteq S[p]$. An approximation P is univocal iff each $p \in P$ is univocal.

Definition 3. An approximation P is **complete** in $(S, \hookrightarrow)$ iff for every attractor $A \subseteq S$ of $(S, \hookrightarrow)$ there is $p \in P$ such that $A \subseteq S[p]$.

Note that the three properties are independent of each other. If P is faithful, univocal, and complete, then we call it a **perfect approximation**. If P is perfect, then all attractors can be found by the random walk method above.

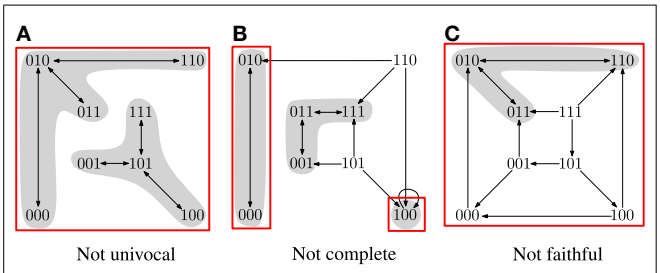

FIGURE 1 | The asynchronous STGs of three different Boolean networks. The minimal trap spaces are indicated by boxes. **(A)** Two attractors in the same box. **(B)** An attractor outside of the boxes. **(C)** An attractor that does not oscillate in all dimensions of the box. Equations for the networks are given in the Supplementary Material.

In Klarner et al. (2014), we observed that $\min(S_F^\star)$ is a good candidate for a perfect approximation. We showed that steady states are minimal trap spaces ($S_F \subseteq \min(S_F^\star)$) and that every $p \in \min(S_F^\star) \setminus S_F$ contains only cyclic attractors. Given that $\min(S_F^\star)$ can be computed efficiently using ASP, we would like to have an efficient method for determining its quality as an approximation. **Figure 1** demonstrates that $\min(S_F^\star)$ is, in general, neither univocal, complete nor faithful.

4.1. Univocality

Proposition 3 (Univocality). Let p be a trap space and $x \in A$ such that $A \subseteq S[p]$ is an attractor of $(S, \hookrightarrow)$. p is univocal in $(S, \hookrightarrow)$ iff

$$TS = (S_{V_p}, \hookrightarrow, S_{V_p}) \models \mathbf{EF}(\varphi_y)$$

where $y \in S_{V_p}$ is the projection of $x \in S_v$ onto V_p.

The intuition behind this proposition is that if A is the only attractor inside the trap space p then x must be reachable from all states S_{V_p}.

4.2. Faithfulness

Proposition 4 (Faithfulness). A trap space p is faithful in $(S, \hookrightarrow)$ iff

$$TS = (S_{V_p}, \hookrightarrow, S_{V_p}) \models \bigwedge_{v \in V_p} \mathbf{EF}(\delta_v \neq 0).$$

This proposition is true because a variable v oscillates in an attractor A iff there is a state $x \in A$ such that $x \models (\delta_v \neq 0)$.

4.3. Completeness

Proposition 5 (Completeness). A set of trap spaces p is complete in $(S, \hookrightarrow)$ iff

$$TS = (S, \hookrightarrow, S) \models \bigvee_{p \in P} \mathbf{EF}(\varphi_p).$$

Although we may restrict the initial states to $S \setminus \cup_{p \in P} S[p]$, the completeness query is still essentially dealing with the whole transition system and is therefore much less efficient than the queries of Proposition 2–4 (which are decided on reduced systems). In Klarner (2015b), we benchmarked NUSMV and found that Boolean networks with $n \approx 39$–55 variables may be considered infeasible for queries of this type. The next section develops a refinement-based approach to decide completeness that can deal with much larger networks.

5. Deciding Completeness by Iterative Refinement

The central idea for the refinement-based approach is to exploit hierarchies in the interaction graph and to use model checking on subnetworks that are in the upper layers of the hierarchy rather than the whole network. Given a complete set of trap spaces p, we keep replacing each $p \in P$ by smaller trap spaces until either $P = \min(S_F^\star)$ and we declare victory, or we find some $p \in P$ that satisfies the failure criterion below which implies that $\min(S_F^\star)$ can not be complete.

Proposition 6 (Refinement). Let $P \subseteq S_F^\star$ be complete in $(S, \hookrightarrow)$ and $p \in P$ some trap space. If $Q \subseteq S_{F_p}^\star$ is complete in $(S_{V_p}, \hookrightarrow)$ then $P' := (P \setminus \{p\}) \cup \{p \sqcap q \mid q \in Q\}$ is complete in $(S, \hookrightarrow)$.

Note that the intersection $p \sqcap q$ is necessary to position the trap space q of $(S_{V_p}, \hookrightarrow)$ correctly in the full transition system $(S_V, \hookrightarrow)$ and that $(p \sqcap q) \leq p$. An example of a refinement is the percolation operator. By Proposition 1, if P is complete, then $\vec{P} := \{\vec{p} \mid p \in P\}$ is also complete. The **failure criterion** is based on the observation that if $\min(S_F^\star)$ is complete in $(S_{V_p}, \hookrightarrow)$, then $\min(S_{F_p}^\star)$ must be complete in $(S_{V_p}, \hookrightarrow)$ for every $p \in S_F^\star$.

Proposition 7 (Failure Criterion). If there is a trap space p such that $\min(S_{F_p}^\star)$ is not complete in $(S_{V_p}, \hookrightarrow)$, then $\min(S_F^\star)$ is not complete in $(S, \hookrightarrow)$.

Example 1. Consider the network defined by $V = \{v_1, v_2, v_3\}$ and F with $f_1 = \overline{v_1}\,\overline{v_2} + v_1 v_2 + v_2 \overline{v_3}$, $f_2 = \overline{v_1}\,\overline{v_2} v_3 + v_1 v_2 v_3$ and $f_3 = v_2 + v_3$. The minimal trap spaces are {111,*00}. The trap space $p := {}^{**}1$ satisfies the failure criterion because $\min(S_{F_p}^\star) = \{11\}$ is not complete in $(S_{V_p}, \hookrightarrow)$ as there is, for example, no path from 01 to 11 in $(S_{V_p}, \hookrightarrow)$. It follows that $\min(S_F^\star)$ is not complete.

5.1. Autonomous Sets

To find the initial $P \subseteq S_F^\star$ and then $Q \subseteq \min(S_{F_p}^\star)$ for a given $p \in P$ we use Proposition 8 below. It is based on so-called autonomous sets, a generalization of inputs. The variables $U \subseteq V$ are **autonomous** iff $Above(U) = U$ in the interaction graph. An autonomous U induces a **restricted network** $(U, F_{|U})$ where $F_{|U} := \{f_u \in F \mid u \in U\}$. Note that if U is autonomous, then $(U, F_{|U})$ is a well-defined network.

Proposition 8. Let U be autonomous and $Q := \min(S_{F_{|U}}^\star)$ the minimal trap spaces of the restriction $(U, F_{|U})$.

(a) If Q is complete in $(S_U, \hookrightarrow)$, then Q is also complete in $(S, \hookrightarrow)$.
(b) If Q is not complete in $(S_U, \hookrightarrow)$, then $\min(S_F^\star)$ is not complete in $(S, \hookrightarrow)$.

Note that the inputs $I \subseteq V$ of a network are autonomous and that P defined by $P := \{p \in S_F^\star \mid D_p = I\}$ (the $|P| = 2^{|I|}$ input combinations) is complete in $(I, F_{|I})$. Proposition 8(a) implies that P is also complete in (V, F). $\vec{P}$ is a refinement of P and if any $\vec{p} \in \vec{P}$ satisfies the failure criterion then $\min(S_F^\star)$ is not complete.

Example 2. Consider the network with $V = \{v_1, \ldots, v_4\}$ and F with $f_1 = v_1$, $f_2 = v_2$, $f_3 = v_1 \overline{v_4}$, $f_4 = v_2 v_3$. The minimal trap spaces are {0000, 0100, 1000, 11**}. To decide whether they are complete we observe that the network has two inputs $\{v_1, v_2\}$ and four input combinations whose minimal trap spaces are $P = \{00^{**}, 01^{**}, 10^{**}; 11^{**}\}$. Since $\vec{P} = \min(S_F^\star) = \{0000, 0100, 1000, 11^{**}\}$, we deduce that $\min(S_F^\star)$ is complete.

5.2. Minimal Autonomous Sets

A refinement-based algorithm requires choosing an autonomous set U and deciding whether Q is complete in $(S_U, \hookrightarrow)$ using the query of Proposition 5. The best performance in terms of model checking is expected if the minimal sets are as small as possible. **Minimal autonomous sets** (set-inclusion-wise) are located in the top layer of the interaction graph $(V, \rightarrow)$ and can be found using any SCC algorithm.

Proposition 9. Let $U \subseteq V$. The following statements are equivalent:

(a) U is a minimal autonomous set of $(V, \rightarrow)$.
(b) U is autonomous and $U \in SCCs(V, \rightarrow)$.

Once it is confirmed that the minimal trap spaces of each restriction are complete, we may consider their **intersection**.

Proposition 10. If $P, Q \subseteq S_F^\star$ are complete in $(S, \hookrightarrow)$ then $P \sqcap Q := \{p \sqcap q \mid p \in P, q \in Q : p \text{ and } q \text{ are consistent}\}$ is also complete in $(S, \hookrightarrow)$.

Note that if P and Q are complete, then for each $p \in P$, there is necessarily a $q \in Q$ such that p and q are consistent. Similarly, for each attractor $A \subseteq S[p]$, there is some consistent $q \in Q$ such that $A \subseteq p \sqcap q$. Hence $P \sqcap Q$ is non-empty and complete. Also, unless there is $p \in P$ with $p \in Q$ we get $|P \sqcap Q| = |P| \cdot |Q|$. Finally, inputs are minimal autonomous sets and if a network has no other minimal autonomous sets, then the intersection is equal to the input combinations. Taking the intersection therefore generalizes the approach of inputs and input combinations.

Example 3. Consider the network with $V = \{v_1, \ldots, v_6\}$ and F with $f_1 = v_2$, $f_2 = v_1$, $f_3 = v_4$, $f_4 = v_3$, $f_5 = v_2 \overline{v_6}$ and $f_6 = v_3 v_5$. The minimal trap spaces are $\min(S_F^\star) = \{000000, 001100, 110010, 1111^{**}\}$. The network has two minimal autonomous sets $U_1 = \{v_1, v_2\}$ and $U_2 = \{v_3, v_4\}$. The corresponding restrictions are $(U_1, F_{|U_1})$ and $(U_2, F_{|U_2})$ with the minimal trap spaces $Q_1 := \min(S_{F_{|U_1}}^\star) = \{11, 00\}$ and $Q_2 := \min(S_{F_{|U_2}}^\star) = \{11, 00\}$. Model checking (or inspection of the STGs) confirms that they are complete in their respective restricted systems. The intersection $P := Q_1 \sqcap Q_2$ and the percolation $\vec{P}$ are $P = \{0000^{**}, 0011^{**}, 1100^{**}, 1111^{**}\}$ and $\vec{P} = \{000000, 001100, 110000, 1111^{**}\}$. As before in Example 2, $\vec{P} = \min(S_F^\star)$ and we deduce that $\min(S_F^\star)$ must be complete in $(S, \hookrightarrow)$.

5.3. Extending Minimal Autonomous Sets

Although minimal autonomous sets are favorable for efficient model checking, there is no guarantee that the respective restricted systems do actually contain non-trivial trap spaces. A refinement based on the trivial trap space ϵ, i.e., $Q = \{\epsilon\}$, is useless because it means replacing p with $p \sqcap \epsilon = p$, that is, with itself. A possible solution is to increase the size of checked autonomous sets until we find non-trivial trap spaces. The question is: by how many variables should we extend an autonomous set U? On the one hand, we want to be generous because new variables increase the chances for finding new trap spaces. On the other hand, we want to add as few variables as possible because the failure criterion requires CTL model checking.

What is the best extension for a given U whose restricted system is trap-space-free? Adding only outputs or cascades to U is not

enough as the emergence of trap spaces requires "self-freezing", positive feedback circuits, see Section 4.7 in Klarner (2015b). Intuitively, we want to extend down to the next SCC.

For a clean definition, we introduce the following notions. The set of **cascade components** consists of all single element SCCs in the interaction graph, whose nodes do not have self-loops. The remaining components are the **non-cascade components**.

$$Casc(V, \to) := \{U \in SCCs(V, \to) \mid \exists v \in V : U = \{v\}, v \not\to v\}$$
$$NonCasc(V, \to) := SCCs(V, \to) \setminus Casc(V, \to)$$

The **condensation graph** $(Z, \rhd)$ of the interaction graph is then the digraph with vertices $Z := NonCasc(V, \to)$ such that an arc $U \rhd W$ indicates whether there is a cascade from U to W. More precisely, $U \rhd W$ iff $U \neq W$ and there is $u \in U$, $w \in W$ such that

$$\exists \pi \in FinPaths(u, w) : \forall 1 \leq i \leq len(\pi) - 2 : \{\pi[i]\} \in Casc(V, \to).$$

Note that $(Z, \rhd)$ is acyclic and so we can partition its vertex set into classes, which we call **layers**, depending on the longest path that reaches them.

$$Lay(W) := \max\{len(\pi) \mid \pi \in FinPaths(U, W), U \in Z\}$$

Note that $Lay(W) \geq 1$ because $\pi = (W)$ is an admissible path from W to W and $len(W) = 1$ and that all minimal autonomous sets can then be found in the first layer of the condensation graph, i.e., $U \subseteq V$ is minimal and autonomous iff $U \in Z$ and $Lay(U) = 1$.

To illustrate how the condensation graph is used for extending autonomous sets, consider the network given in **Figure 2**. First, we compute its minimal autonomous sets, i.e., the top layer of $(Z, \rhd)$. In this example, there is a unique $W \in Z$ with $Lay(W) = 1$. The restriction $(W, F_{|W})$ consists of an isolated negative feedback circuit and is trap-space-free. To determine the smallest extension that contains new feedback circuits, we first compute the graph $(Z', \rhd)$, which is obtained from the condensation graph $(Z, \rhd)$ by removing all $U \in Z$ that satisfy $U \cap W \neq \varnothing$. For each $Y \in Z'$ that satisfies $Lay(Y) = 1$, we get an extended autonomous set W' by considering the variables above Y in the interaction graph $(V, \to)$. In the example, there is again a unique Y and the restriction to $W' := Above(Y)$ contains a non-trivial trap space p. The failure criterion is not satisfied by p and so we have found an initial complete set, namely $P := \{p\}$. Note that in general, there will be several minimal autonomous sets and several possible extensions. We are now ready to design an efficient algorithm for deciding completeness.

5.4. The Algorithm

The first step of the algorithm in **Figure 3** is to compute the minimal trap spaces of a given network using the ASP-based method proposed in Klarner et al. (2014). If the network is trap-space-free, then $\min(S_F^\star) = \{\epsilon\}$ is, by definition, complete and we stop and return *true*. Otherwise the variable *CurrentSet* is initialized. It consists of tuples (p, W), where p is a trap space and $W \subseteq V$ are the variables of the network (V_p, F_p) that have previously been subjected to model checking. The tuples correspond to those trap spaces of a complete set that need further refinement (i.e., are not minimal). Initially $CurrentSet := \{(\epsilon, \varnothing)\}$ because $\{\epsilon\}$ is trivially complete and we have not started model checking $W = \varnothing$. The lines 5–24 execute the iterative refinement of *CurrentSet* until we either find a p that satisfies the failure criterion in line 17 or $CurrentSet = \varnothing$ in which case every p is equal to some minimal trap space (only non-minimal trap spaces are put back onto *CurrentSet*, see lines 23, 24).

The next steps are to select an arbitrary (p, W) for refinement (line 6), compute the reduced network (V_p, F_p), its condensation graph $(Z, \rhd)$ and the graph $(Z', \rhd)$ described in the previous section. The top layer elements U of $(Z', \rhd)$ are minimal autonomous sets if $Z = Z'$ or extended autonomous sets if $Z \neq Z'$. In the latter case, the restricted networks that correspond to minimal autonomous sets of (V_p, F_p) must have previously been found to be trap-space-free. For each U, the variables above U are autonomous (in (V_p, F_p)). If the minimal trap spaces of the restricted networks are complete in $(S_{U'}, \hookrightarrow)$ then, by Proposition 8(a), they are also complete in $(S_{V_p}, \hookrightarrow)$. Otherwise it follows, by Proposition 8(b), that $\min(S_{F_p}^\star)$ is not complete in $(S_{V_p}, \hookrightarrow)$ and hence that p satisfies the failure criterion and we stop and return *false* in line 18.

The variable *Refinement* stores all complete sets that were found in the upper layers of (V_p, F_p), while W' keeps track of the variables

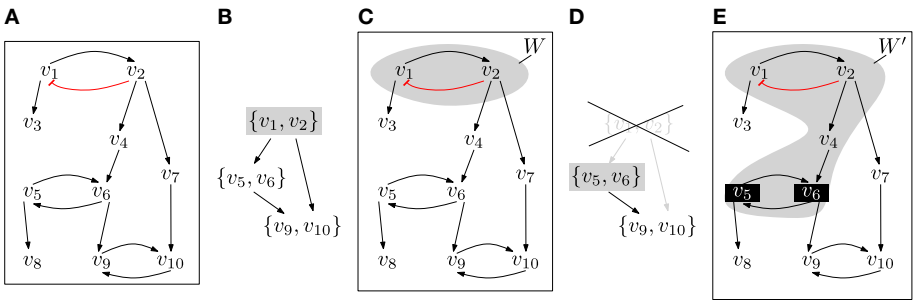

FIGURE 2 | (A) The interaction graph of an example network where each f_i is the disjunction of its inputs (e.g., $f_6 = v_5 + v_4$) except for v_1 which is inhibited by v_2 ($f_1 := \overline{v_2}$). **(B)** The condensation graph $(Z, \rhd)$ with the unique top layer node $\{v_1\ v_2\}$. **(C)** The corresponding minimal autonomous set W. The restriction $(W, F_{|W})$ is trap-space-free. **(D)** To extend W, we compute the graph $(Z', \rhd)$, which is obtained from $(Z, \rhd)$ by removing the node that intersects W. The new top layer node is $Y := \{v_5, v_6\}$. **(E)** The extended autonomous set W' is obtained by considering $Above(Y)$ in the interaction graph. It has a minimal trap space p that is defined by fixing v_5 and v_6 to 1.

Algorithm 1 (V, F)

Input: (V, F) is a constant-free Boolean network
Output: **true** if $\min(S^\star_F)$ is complete, **false** otherwise

1: $MinTrapSpaces \leftarrow$ MINTRAPSPACES(V, F)
2: **if** $MinTrapSpaces = \{\epsilon\}$ **then**
3: **return true**
4: $CurrentSet \leftarrow \{(\epsilon, \varnothing)\}$
5: **while not** $CurrentSet = \varnothing$ **do**
6: $(p, W) \leftarrow CurrentSet$.POP()
7: $(V_p, F_p) \leftarrow$ REDUCEDNETWORK(V, F, p)
8: $(Z, \rhd) \leftarrow$ CONDENSATIONGRAPH(V_p, F_p)
9: $(Z', \rhd) \leftarrow$ REMOVECOMPONENTS$(Z, \rhd, W)$
10: $W' \leftarrow$ COPY(W)
11: $Refinement \leftarrow \varnothing$
12: **for** $U \in$ TOPLAYER$(Z', \rhd)$ **do**
13: $U' \leftarrow$ ABOVE(V_p, F_p, U)
14: $(U', F_{|U'}) \leftarrow$ RESTRICTION(V_p, F_p, U')
15: $Q \leftarrow$ MINTRAPSPACES$(U', F_{|U'})$
16: $\varphi \leftarrow$ COMPLETENESSQUERY(Q)
17: **if not** CTLMODELCHECKING$(S'_U, \hookrightarrow, \varphi)$ **then**
18: **return false**
19: $Refinement$.APPEND(INTERSECTION(p, Q))
20: $W' \leftarrow$ SETUNION(W', U')
21: **for** $q \in$ INTERSECTION$(Refinement)$ **do**
22: $\vec{q} \leftarrow$ PERCOLATION(V, F, q)
23: **if not** $\vec{q} \in MinTrapSpaces$ **then**
24: $CurrentSet$.APPEND$(\vec{q}, W')$
25: **return true**

FIGURE 3 | The iterative, refinement-based algorithm for deciding the question of completeness. See main text for a detailed description.

that were subjected to model checking. Line 21 is an application of Proposition 10, i.e., the intersection of all complete sets is taken (generalization of input combinations). For each trap space q in the intersection, we check whether the percolation $\vec{q}$ needs further refinement (not a minimal trap space of (V, F)) and if so add it back onto *CurrentSet*.

Note that (V_p, F_p) must have non-trivial trap spaces for each $(p, W) \in$ *CompleteSets* (see lines 23, 24). Hence, although it may happen that (p, W) is replaced by (p, W') (if $Q = \{\epsilon\}$ in line 15) eventually it will be replaced by smaller trap spaces. The algorithm is implemented and available as part of our PYTHON toolbox for Boolean networks (Klarner, 2015a).

5.5. Counterexamples for Attractor Detection

If $\min(S^\star_F)$ is not a perfect approximation, we would like to know why. Model checking tools like NUSMV are capable of producing a counterexample in case a formula does not hold. Intuitively, a counterexample is a finite path from an initial state that proves that the query is false. If $\min(S^\star_F)$ is not complete, then the algorithm of the previous section can be used to return some $p \in S^\star_F$ that satisfies the failure criterion together with a counterexample to the respective completeness query for (V_p, F_p) and $\min(S^*_{F_p})$. Every attractor that is reachable from its last state, say x, must then be outside of $\min(S^*_{F_p})$. We then use the random walk approach to find state a z that belongs to an attractor $A \subseteq S_{V_p}$ outside of $\min(S^*_{F_p})$. If the modified completeness query

$$TS = (S_{V_p}, \hookrightarrow, S_{V_p}) \models \varphi_z \vee \bigvee_{q \in \min(S^\star_{F_p})} \mathbf{EF}(\varphi_q)$$

holds then A is the only outside attractor, otherwise we use the next counterexample to find the next outside attractor until they are all found. Note that p is an extension of a minimal autonomous set. A similar approach is possible for trap spaces that are not faithful or not univocal. We end up with a set of states that captures the attractors outside of P, the number of attractors inside $S[p]$ for each $p \in P$ and whether they are faithful or not.

6. Results

All computations in this section were done on a 32-bit Linux laptop with 4×2.60 GHz and 8 GB memory.

6.1. MAPK Case Study

In this case study, we consider the network published in Grieco et al. (2013), which models the influence of the MAPK pathway on cancer cell fate decisions and consists of 53 variables. Using Klarner (2015a), we compute $\min(S^\star_F)$ in under one second. It consists of 12 steady states and six trap spaces that contain only cyclic attractors. The single query approach to deciding completeness runs 35 min, while the refinement-based algorithm confirms completeness in only 28 s. For the six trap spaces in $\min(S^\star_F) \setminus S_F$ we confirmed univocality in 261 s (44 s on average per trap space) and faithfulness in 74 s (12 s on average per trap space) using the CTL queries of Section 4. Hence, $\min(S^\star_F)$ is a perfect approximation of the attractors of $(S, \hookrightarrow)$ and for each attractor we can find an internal state by the random walk approach of Section 4. We stopped GENYSIS after seven hours without a result.

Figure 4 is an illustration of the steps performed during the iterative refinement for the MAPK network. The information is represented as a **decision tree**. The root represents the initial and trivially complete set $P := \{\epsilon\}$. Boxes are split into a left side, representing the size $|U|$ of a minimal autonomous set (or an extension), and a right hand side that is split vertically into cells that contain the numbers $|D_q|$ of fixed variables for each minimal trap space q of $(U, F_{|U})$. Boxes are colored according to whether $(U, F_{|U})$ is trap-space-free (white) or not in which case model checking is required to find out whether the minimal trap spaces of $(U, F_{|U})$ are complete (failure criterion). Boxes with more than one minimal trap space are outlined in red to emphasize that a decision process between competing trap spaces exists. The intersection of several autonomous sets is indicated by $\otimes$ but occurs for this network only for the inputs. Arcs are labeled by the number of variables that are fixed during percolations, i.e., $\left|D_q \setminus D_{\vec{q}}\right|$ (see line 22 in **Figure 3**). If a restricted network is trap-space-free, the extension is indicated by a dashed arc. Along each branch of the decision tree, the number of fixed and oscillating variables must add up to 53. The bottom branch, for example, starts with four fixed variables, percolates seven more, extends an autonomous set whose restriction consists of four variables and is trap-space-free, finds a single trap space with three fixed variables and finishes as the remaining trap space is minimal (and $4 + 7 + 3 + 0 + 39 = 53$).

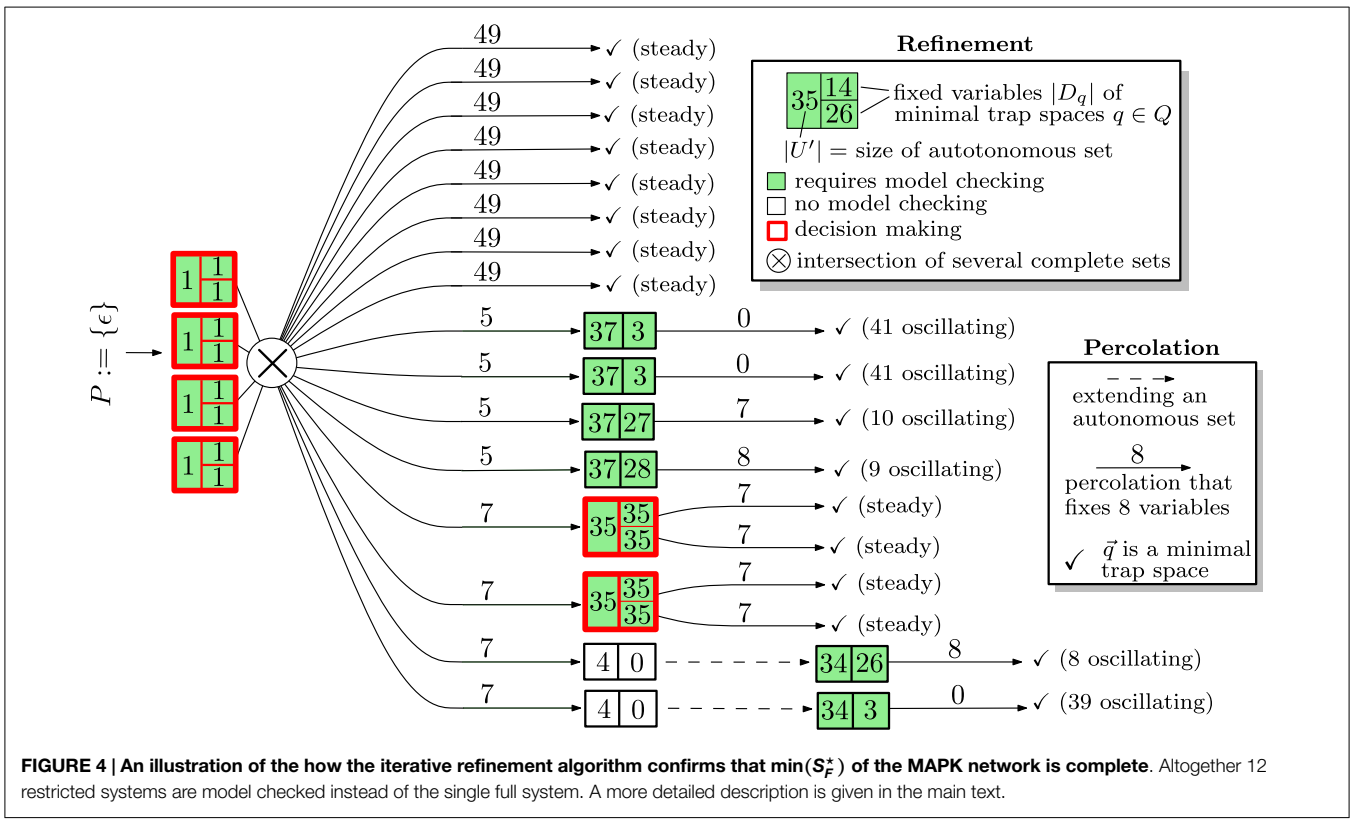

FIGURE 4 | An illustration of the how the iterative refinement algorithm confirms that $\min(S_F^\star)$ of the MAPK network is complete. Altogether 12 restricted systems are model checked instead of the single full system. A more detailed description is given in the main text.

TABLE 2 | The minimal trap spaces of all Boolean models in the GINSIM repository are perfect approximations of the attractors of $(S, \hookrightarrow)$.

Network file (.zginml)	$\|V\|$	Steady	Cyclic	Faithful	Univocal	Complete
buddingYeastOrlando2008	9	1	–	0.08*s*	0.03*s*	0.23*s*
fissionYeastDavidich2008	10	12	–	0.01*s*	0.02*s*	0.08*s*
boolean_cell_cycle	10	1	1	0.03*s*	0.19*s*	0.12*s*
Toll_Pathway_12Jun2013	11	4	–	0.01*s*	0.01*s*	0.09*s*
drosophilaCellCycleVariants	14	1	–	0.01*s*	0.05*s*	0.11*s*
MAPK_red3_19062013	16	12	6	0.15*s*	1.11*s*	0.93*s*
MAPK_red1_19062013	17	12	6	0.18*s*	1.25*s*	0.87*s*
VEGF_Pathway_12Jun2013_0	18	256	–	0.04*s*	0.05*s*	0.28*s*
MAPK_red2_19062013	18	12	6	0.14*s*	1.26*s*	0.67*s*
buddingYeastIrons2009	18	–	1	0.16*s*	0.48*s*	0.02*s*
ErbB2_model	20	1	–	0.08*s*	0.00*s*	0.02*s*
FGF_Pathway_12Jun2013	23	512	–	0.09*s*	0.09*s*	0.51*s*
Hh—Pathway_11Jun2013_0	24	8192	–	1.29*s*	1.43*s*	6.34*s*
Spz—Processing_12Jun2013	24	64	–	0.04*s*	0.03*s*	0.22*s*
Wg_Pathway_11Jun2013	26	16384	–	2.38*s*	2.38*s*	17.16*s*
TCRsig40	40	7	1	1.07*s*	3.34*s*	0.12*s*
MAPK_large_19june2013	53	12	6	40.15*s*	565.84*s*	20.72*s*
T_LGL	60	86	70	1.07*s*	6.57*s*	5669.57*s*

The number of variables, steady states, and cyclic attractors are recorded in the first three columns. The remaining three columns record the time needed to confirm faithfulness, univocality, and completeness.

Note that the algorithm encounters roughly four types of refinements. The first type (branches 1–8) leads directly to a steady state. The second type (branches 9–12) discovers a single minimal autonomous set consisting of 37 variables, whose restriction has a single minimal trap spaces in which between nine and 41 variables oscillate. The third type (branches 13–14) discovers a single minimal autonomous set that has two minimal trap spaces that commit the network to different steady states. The fourth type (branches 15–16) discovers a single minimal autonomous set consisting of four variables that is trap-space-free. An extension leads an autonomous set of 34 variables with a single minimal trap space.

6.2. GINsim Repository Benchmark

To test whether the MAPK network is unusual in that its minimal trap spaces are perfect approximations, we ran the same analysis for every Boolean model currently in the GINSIM model repository (see Naldi et al. (2009)). In every case, the minimal trap spaces are perfect approximations of the attractors of $(S, \hookrightarrow)$. The time

needed to confirm faithfulness, univocality and completeness is given in **Table 2**. We confirmed the number of steady states and cyclic attractors with GINSIM and GENYSIS. The execution of GINSIM is, like the computation of minimal trap spaces, instantaneous. The running times of GENYSIS are comparable to that of our algorithm for networks with $|\mathrm{V}| < 40$ and on the order of 24–72 h for the three networks with $|V| \geq 40$. The networks and attractors are available for benchmarking at Klarner (2015a).

7. Conclusion and Outlook

In this paper, we developed the notion of an approximation of attractors of a Boolean network. Minimal trap spaces are approximations that can be computed for networks with hundreds of variables using ASP solvers. Since available attractor detection tools for asynchronous systems are only feasible for about 50 variables, approximations via minimal trap spaces might yield attractor information otherwise inaccessible. We defined three criteria to assess the quality of an approximation and showed that they can be decided using model checking. The main contribution in this paper is an algorithm that improves the efficiency of deciding completeness by dividing the problem into smaller subproblems according to autonomous sets in the interaction graph.

We ran the algorithm on the 18 Boolean networks that are currently in the GINSIM repository and found that each time, the minimal trap spaces are a perfect approximation of the asynchronous attractors, i.e., that we can find all asynchronous attractors using random walks and $\min(S_F^\star)$.

Section 5.3 explains that autonomous sets must be extended if the corresponding restricted systems are trap-space-free. Strategies by which extensions are constructed must compromise between adding variables to increase the likelihood of discovering non-trivial trap spaces and the efficiency of model checking the respective transition systems. The strategy in Section 5.3 can be considered optimal in the sense that it adds as few variables at a time as necessary for the emergence of new trap spaces.

There are several directions in which the algorithm may be improved further, for example, by removing so-called "mediator variables" (see, e.g., Saadatpour et al. (2013)) from the interaction graph of the subnetworks. The relationship to other reduction methods, e.g., Naldi et al. (2011) or Veliz-Cuba (2011), may also yield improvements by reducing the size of the transition systems passed to the model checking software further.

The decision tree in **Figure 4** might be an interesting tool for questions regarding network control, an idea that was recently developed in Zañudo and Albert (2015). It also suggests that the dynamics of Boolean networks is governed by two very different regimes: the *percolation regime* in which the long-term activities are pre-determined, and the *decision-making regime* in which the long-term activities are determined by which of the competing trap spaces is reached first.

Supplementary Material

The Supplementary Material for this article can be found online at http://journal.frontiersin.org/article/10.3389/fbioe.2015.00130

References

Akutsu, T., Yang, Z., Hayashida, M., and Tamura, T. (2012). Integer programming-based approach to attractor detection and control of boolean networks. *IEICE Trans. Inf. Syst.* 95, 2960–2970. doi:10.1587/transinf.E95.D.2960

Berntenis, N., and Ebeling, M. (2013). Detection of attractors of large boolean networks via exhaustive enumeration of appropriate subspaces of the state space. *BMC Bioinformatics* 14:361. doi:10.1186/1471-2105-14-361

Carrillo, M., Góngora, P. A., and Rosenblueth, D. A. (2012). An overview of existing modeling tools making use of model checking in the analysis of biochemical networks. *Front. Plant Sci.* 3:155. doi:10.3389/fpls.2012.00155

Cimatti, A., Clarke, E., Giunchiglia, F., and Roveri, M. (2000). NuSMV: a new symbolic model checker. *Int. J. Software Tool. Technol. Tran.* 2, 410–425. doi:10.1007/s100090050046

Dubrova, E., and Teslenko, M. (2011). A SAT-based algorithm for finding attractors in synchronous boolean networks. *IEEE/ACM Trans. Comput. Biol. Bioinform.* 8, 1393–1399. doi:10.1109/TCBB.2010.20

Garg, A., Di Cara, A., Xenarios, I., Mendoza, L., and De Micheli, G. (2008). Synchronous versus asynchronous modeling of gene regulatory networks. *Bioinformatics* 24, 1917–1925. doi:10.1093/bioinformatics/btn336

Grieco, L., Calzone, L., Bernard-Pierrot, I., Radvanyi, F., Kahn-Perlès, B., and Thieffry, D. (2013). Integrative modelling of the influence of mapk network on cancer cell fate decision. *PLoS Comput. Biol.* 9:e1003286. doi:10.1371/journal.pcbi.1003286

Irons, D. J. (2006). Improving the efficiency of attractor cycle identification in boolean networks. *Physica D* 217, 7–21. doi:10.1016/j.physd.2006.03.006

Kauffman, S. A. (1993). *The Origins of Order: Self Organization and Selection in Evolution*. Oxford University Press.

Klarner, H. (2015a). Available at: http://sourceforge.net/Projects/BoolNetFixpoints

Klarner, H. (2015b). *Contributions to the Analysis of Qualitative Models of Regulatory Networks*. PhD thesis, Freie Universität Berlin, Germany.

Klarner, H., Bockmayr, A., and Siebert, H. (2014). "Computing symbolic steady states of boolean networks," in *Cellular Automata*, Vol. 8751. eds J. Was, G. C. Sirakoulis, and S., Bandini (Switzerland: Springer International Publishing), 561–570.

Naldi, A., Berenguier, D., Fauré, A., Lopez, F., Thieffry, D., and Chaouiya, C. (2009). Logical modelling of regulatory networks with ginsim 2.3. *BioSystems* 97, 134–139. doi:10.1016/j.biosystems.2009.04.008

Naldi, A., Remy, E., Thieffry, D., and Chaouiya, C. (2011). Dynamically consistent reduction of logical regulatory graphs. *Theor. Comp. Sci.* 412, 2207–2218. doi:10.1016/j.tcs.2010.10.021

Saadatpour, A., Albert, R., and Reluga, T. C. (2013). A reduction method for boolean network models proven to conserve attractors. *SIAM J. Appl. Dyn. Syst.* 12, 1997–2011. doi:10.1137/13090537X

Siebert, H. (2011). Analysis of discrete bioregulatory networks using symbolic steady states. *Bull. Math. Biol.* 73, 873–898. doi:10.1007/s11538-010-9609-1

Skodawessely, T., and Klemm, K. (2011). Finding attractors in asynchronous boolean dynamics. *Adv. Complex Syst.* 14, 439–449. doi:10.1142/S0219525911003098

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

Veliz-Cuba, A. (2011). Reduction of boolean network models. *J. Theor. Biol.* 289, 167–172. doi:10.1016/j.jtbi.2011.08.042

Veliz-Cuba, A., Aguilar, B., Hinkelmann, F., and Laubenbacher, R. (2014). Steady state analysis of boolean molecular network models via model reduction and computational algebra. *BMC Bioinformatics* 15:221. doi:10.1186/1471-2105-15-221

Zañudo, J. G. T., and Albert, R. (2015). Cell fate reprogramming by control of intracellular network dynamics. *PLoS Comput. Biol.* 11:e1004193. doi:10.1371/journal.pcbi.1004193

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.