Compositional analysis of Boolean networks using local fixed-point iterations

A. Le Coënt[1], L. Fribourg[2], and R. Soulat[3]

[1] CMLA, ENS Cachan, CNRS, Université Paris-Saclay
61 av. du Président Wilson, 94235 Cachan cedex, France
adrien.le-coent@ens-cachan.fr
[2] LSV, ENS Cachan, CNRS, Université Paris-Saclay
61 av. du Président Wilson, 94235 Cachan cedex, France
fribourg@lsv.ens-cachan.fr
[3] Thales Research & Technology
1 av. Augustin Fresnel, 91767 Palaiseau, France
romain.soulat@thalesgroup.com

Abstract. We present a compositional method which allows to over-approximate the set of attractors and under-approximate the set of basins of attraction of a Boolean network (BN). This merely consists in replacing a global fixed-point computation by a composition of local fixed-point computations. Once these approximations have been computed, it becomes much more tractable to generate the exact sets of attractors and basins of attraction. We illustrate the interest of our approach on several examples, among which is a BN modeling a railway interlocking system with 50 nodes and millions of attractors.

1 Introduction

Boolean Networks (BNs) have been widely used to model biological systems [15]. The BN is a discrete model that comprises a number of nodes and corresponding update rules. Classically, each node represents a gene and takes a value of 1 or 0, meaning that the gene is expressed or not. Each update rule represents interactions between genes. BNs have also been used in industrial networks such as railway yards [10]. The states of objects in a railway yard (railway interlocking system) can be captured by means of Boolean variables. The control and management of such systems can then be expressed under the form of rules of BNs.

We consider here *synchronous* BNs, which means that the updates are performed synchronously. Synchronous BNs can be considered as a class of deterministic finite state machines. Any sequence of consecutive states eventually converges to a cycle of states, called an *attractor*. In biological systems, attractors capture long-term behaviors of biological systems (e.g., growth, differentiation, and apoptosis) [15]. In railway interlocking systems, attractors also convey important information.

Practically, all algorithms for computing attractors in Boolean networks face a potential state-space explosion that must be addressed to handle large-scale models. (The problem of finding attractors in BNs is NP-hard [1].) A common approach is to use *symbolic* algorithms (Binary Decision Diagrams (BDDs) [3] or SAT-based methods) which avoid representing explicit states and transitions.

Algorithms based on BDDs are usable to process BNs with up to a hundred of state variables [17]. However for larger networks, BDDs become too memory-consuming. Propositional decision procedures (SAT) do not suffer from the potential space explosion of BDDs and can handle propositional satisfiability with thousands of variables [8]. The approach in [9] relies on SAT-based bounded model checking [2] to compute attractors. Those algorithms enable to scale up hundreds of nodes with K (maximum indegree or maximum node connectivity) ≤ 3 (i.e., low maximum connectivity). However, in case of BNs with higher Ks, a state explosion still occurs. The same phenomenon occurs when the number of attractors N_a increases, which makes the problem of finding attractors impracticable for the case studies that we consider here (for which N_a increases exponentially with the number n of state variables).

To expand the range of feasible BNs, partitioning-based attractor detection algorithms have been published recently [12, 19]. Both works use a partitioning strategy based on strongly connected component (SCC). Attractors are independently detected in each block, and then combined to construct the attractors of the original BN. Therefore better scalability can be achieved, but still for low K (≤ 3). For BNs with large K, the size of the largest SCC is too large to be analyzed within a reasonable time.

In order to overcome this problem, in [14], the authors propose a partitioning not based on SCC. They are thus able to find attractors for networks with a number n of nodes up to 1000 and $K = 5$. Unfortunately the method generates only "steady states", i.e, cyclic attractors of length 1, and ignores cyclic attractors of greater length.

Here, we propose a method which uses the same kind of partitioning as in [14] but we use a different algorithm for detecting the local attractors inside each component: while [14] uses SAT-based bounded model checking methods for finding local attractors, we use an "iterative reduction" method (similar to [6], Sec. 11.2 and [7], Sec. VIII.B). For constructing the global attractors, we first combine the results obtained by the local iterative reductions similarly to [7]. This allows us to compute an *over-approximation* of the union of all the attractors (not only the steady states). In a second step, starting from this over-approximation, we then compute the *exact set* of all the attractors using global fixed-point iterations.

We have implemented the method in Octave. Using this prototype, we are able to find all the attractors of a BN with $n = 50$ and $K = 6$, which models a portion of the New York City subway [11].

We also explain how our compositional method can be adapted to construct (under-approximations of) basins of attraction.

Plan

We explain how to find (a superset of) all the attractors in a compositional way in Section 2 and (subsets of) basins of attraction in Section 3. We present some experiments performed with a prototype implementation in Section 4. We conclude in Section 5.

2 Attractors

2.1 Concrete functions

A *synchronous Boolean Control Network (BCN)* is a discrete-time dynamical system subject to the rules

$$x(t+1) = f(x(t), u(t)) \tag{1}$$

where x is a vector of n Boolean variables (called *state*), u is a vector of m Boolean variables (called *control input*), and f is a vector of n Boolean functions on these variables and inputs. We denote by S the set of all possible instantiations of variables x ($S = \{0,1\}^n$). We denote by U the set of all possible instantiations of inputs u ($U = \{0,1\}^m$).

In the following, we will consider that a BCN can be decomposed into two systems of the form

$$x_1(t+1) = f_1(x_1(t), x_2(t), u_1(t)) \tag{2}$$
$$x_2(t+1) = f_2(x_1(t), x_2(t), u_2(t)) \tag{3}$$

where x_1 and x_2 are vectors of respectively n_1 and n_2 Boolean variables with $n = n_1 + n_2$, u_1 and u_2 are vectors of respectively m_1 and m_2 Boolean inputs with $m = m_1 + m_2$, and f_1 and f_2 are vectors of respectively n_1 and n_2 Boolean functions on these variables and inputs. We denote by S_1 the set of all possible instantiations of variables x_1 ($S_1 = \{0,1\}^{n_1}$), and by S_2 the set of all possible instantiations of variables x_2 ($S_2 = \{0,1\}^{n_2}$). Likewise, we denote by U_1 the set of all possible instantiations of inputs u_1 ($U_1 = \{0,1\}^{m_1}$), and by U_2 the set of all possible instantiations of inputs u_2 ($U_2 = \{0,1\}^{m_2}$).

Remark: The way of finding interesting splittings of the system into two sub-systems is beyond the scope of this paper. It can be done using the method of [14].

A *BN* is a BCN without control inputs:

$$x(t+1) = f(x(t)). \tag{4}$$

For the sake of simplicity, we will focus in the sequel of this section on BN.

The definitions of f, f_1 and f_2 can naturally be "lifted" to the powerset level. We will use the same notation f for the functions and their lifted versions. For

$X \in 2^S$, we have: $f(X) = \{f(x) \mid x \in X\}$. Likewise, given a set $X_1 \in 2^{S_1}$ and a set $X_2 \in 2^{S_2}$, we have for $i = 1, 2$: $f_i(X_1, X_2) = \{f_i(x_1, x_2) \mid x_1 \in X_1, x_2 \in X_2\}$.

In the rest of the paper, all the fixed-point results will concern functions lifted at the powerset level.

We have:

Proposition 2.1. *Suppose:* $F^\times \subseteq X \subseteq S$. *Then* $F^\times = \bigcap_{k\geq 0} f^k(X)$.

Proof. Suppose $F^\times \subseteq X \subseteq S$. Then $\bigcap_{k\geq 0} f^k(F^\times) \subseteq \bigcap_{k\geq 0} f^k(X) \subseteq \bigcap_{k\geq 0} f^k S$. Hence: $F^\times \subseteq \bigcap_{k\geq 0} f^k(X) \subseteq F^\times$. It follows: $F^\times = \bigcap_{k\geq 0} f^k(X)$. □

As already mentioned, since a BN is subject to deterministic rules (applied here synchronously) and since the number of elements is finite (equal to 2^n), every derivation from an arbitrary element ends to a cycle. The set of elements composing this cycle is called an *attractor*. We have:

Proposition 2.2. *The union of the attractors of the BN is equal to* $F^\times$.

2.2 Abstract functions

We are going to give a method for computing an *over-approximation* (i.e., a superset) of $F^\times$. This will be done by constructing the greatest fixed-point of an "abstraction" $\tilde{f}$ of f. Let $\tilde{S} = (S_1, S_2)$.

Definition 2.3. *The function* $\tilde{f} : 2^{S_1} \times 2^{S_2} \to 2^{S_1} \times 2^{S_2}$ *is defined for all* $X_1 \in 2^{S_1}$ *and* $X_2 \in 2^{S_2}$, *by:*

$$\begin{aligned}\tilde{f}(X_1, X_2) &= (f_1(X_1, X_2), f_2(X_1, X_2)) \\ &= \{(f_1(x_1, x_2), f_2(w_1, w_2)) \mid x_1 \in X_1, x_2 \in X_2, w_1 \in X_1, w_2 \in X_2\}.\end{aligned}$$

At the concrete (resp. abstract) level, we consider the finite lattice of functions from 2^S to 2^S (resp. from $2^{S_1} \times 2^{S_2}$ to $2^{S_1} \times 2^{S_2}$). We say that two abstract functions φ and ψ are *ordered*, and write $\varphi \leq_a \psi$ if for any $(X_1, X_2) \in 2^{S_1} \times 2^{S_2}$: $\varphi(X_1, X_2) = (Y_1, Y_2)$, $\psi(X_1, X_2) = (Z_1, Z_2)$ with $Y_i \subseteq_i Z_i$ for $i = 1, 2$, where '$\subseteq_i$' denotes the inclusion ordering between elements of 2^{S_i}. Likewise, we say that two concrete functions f and g are *ordered*, and write $f \leq_c g$ if $f(X) \subseteq_c g(X)$ where '$\subseteq_c$' denotes the inclusion ordering between elements of 2^S. Without loss of understanding, we will omit the indices of symbols '$\leq$' and '$\subseteq$'. Context will make it clear. The identity function at the abstract (resp. concrete) level will be denoted by Id_a (resp. Id_c). Note that, abstract functions and concrete functions are *monotonic* since they are lifted at the powerset level. We have (see [7]):

Definition 2.4. *The* abstraction *function* $\alpha : 2^S \to 2^{S_1} \times 2^{S_2}$ *and the* concretization *function* $\gamma : 2^{S_1} \times 2^{S_2} \to 2^S$ *are defined as follows:*

- *for all* $X \in 2^S = 2^{S_1 \times S_2}$,

$$\alpha(X) = (\pi_1(X), \pi_2(X)),$$

  where π_i *is (the lift of) the* i*-th projection of* S *to* S_i $(i = 1, 2)$.

– *for all* $X_1 \in 2^{S_1}$ *and* $X_2 \in 2^{S_2}$,

$$\gamma(X_1, X_2) = X_1 \times X_2 = \{(x_1, x_2) \mid x_1 \in X_1, x_2 \in X_2\}.$$

The abstraction function α "separates" an element of 2^S, i.e., a set X of n-vectors of bits into two elements X_1 and X_2 of 2^{S_1} and 2^{S_2} respectively, i.e., into a set of n_1-vectors and a set of n_2-vectors with $n = n_1 + n_2$. Conversely, the concretization function γ "gathers" two elements X_1 and X_2 of 2^{S_1} and 2^{S_2} into an element of $2^S = 2^{S_1 \times S_2}$.

It is easy to show that the function $\alpha f \gamma : 2^{S_1} \times 2^{S_2} \to 2^{S_1} \times 2^{S_2}$ coincides with the definition of $\tilde{f}$ given in Definition 2.3, i.e: $\tilde{f} = \alpha f \gamma$. We have:

Proposition 2.5. $\gamma\alpha \geq Id_c$, *i.e., for all* $X \in 2^S$: $\gamma(\alpha(X)) \supseteq X$.

Proof. Let $X \in 2^S$. Write $\alpha(X) = (X_1, X_2)$. If $x = x_1 x_2 \in X$ then $x_1 \in X_1$ and $x_2 \in X_2$, thus $x \in \gamma(X_1, X_2) = \gamma\alpha(X)$.

Intuitively, this inclusion expresses the fact that by separating the arguments at the abstract level, we lose the information of interdependence between these arguments. The functions f, $\tilde{f}$, α and γ satisfy basic properties of Abstract Interpretation (see [7]):

Lemma 2.6. *We have:*

1. $\alpha\gamma \leq Id_a$, *i.e., for all* $(X_1, X_2) \in 2^{S_1} \times 2^{S_2}$: $\alpha(\gamma(X_1, X_2)) \subseteq (X_1, X_2)$.
2. $f\gamma \leq \gamma\tilde{f}$, *i.e., for all* $(X_1, X_2) \in 2^{S_1} \times 2^{S_2}$: $f\gamma(X_1, X_2) \subseteq \gamma\tilde{f}(X_1, X_2)$.

Proof. 1. Write $(Y_1, Y_2) = \alpha\gamma(X_1, X_2)$. Assume $y_i \in Y_i$ for $i = 1, 2$. Then $\exists y \in \gamma(X_1, X_2) : y_i = \pi_i(y)$ for $i = 1, 2$. Since $y \in X_1 \times X_2$, $y_i \in X_i$ for $i = 1, 2$. We have shown $Y_i \subseteq X_i$ for $i = 1, 2$, i.e.: $(Y_1, Y_2) \subseteq (X_1, X_2)$.
2. Since $Id_c \leq \gamma\alpha$ by Proposition 2.5 and $\tilde{f} = \alpha f \gamma$, we have: $\gamma\tilde{f} = \gamma\alpha f\gamma \geq f\gamma$. □

Remark. In general we do not have $\alpha\gamma(X_1, X_2) = (X_1, X_2)$ because, in the case $X_1 = \emptyset$, we have $\alpha\gamma(\emptyset, X_2) = \alpha(\emptyset \times X_2) = (\emptyset, \emptyset)$ which is distinct from (X_1, X_2) when $X_2 \neq \emptyset$.

Let us write the greatest fixed-point $\mathrm{gfp}(\tilde{f})$ of $\tilde{f}$ as $(F_1^\times, F_2^\times)$.

Proposition 2.7. *We have:*

1. $gfp(f) \leq \gamma\ gfp(\tilde{f})$, *i.e.:* $F^\times \subseteq F_1^\times \times F_2^\times$.
2. $F^\times = \bigcap_{k \geq 0} f^k(F_1^\times \times F_2^\times)$.

This proposition is a consequence of Theorem 2.8 that is given below. The pair $(F_1^\times, F_2^\times)$ can be thus used as a 'seed' for the computation of the greatest fixed-point $F^\times$ of f: one starts the iteration of f from $F_1^\times \times F_2^\times$ instead of starting from S. It may be easier to compute $F^\times$ starting from $F_1^\times \times F_2^\times$ rather than S because $|F_1^\times \times F_2^\times|$ is sometimes much smaller than $|S| = 2^n$.

For a given integer $\ell \geq 1$, let us define g and $\tilde{g}$ by: $g = f^\ell$ and $\tilde{g} = \alpha g \gamma$.

Let us write the greatest fixed-point of $\tilde{g}$ as $(G_1^\times, G_2^\times)$. We have:

Theorem 2.8. *For all integer $\ell \geq 1$:*

1. $F^\times = G^\times$.
2. $G^\times \subseteq G_1^\times \times G_2^\times$.
3. $F^\times = \bigcap_{k\geq 0} f^k(G_1^\times \times G_2^\times) = \bigcap_{k\geq 0} g^k(G_1^\times \times G_2^\times)$.
4. $G_1^\times \times G_2^\times \subseteq F_1^\times \times F_2^\times$.

Proof. 1. We have: $F^\times = \bigcap_{k\geq 0} f^k(S) = \bigcap_{k\geq 0} f^{k\ell}(S) = G^\times$.

2. By Lemma 2.6.3 with g in place of f, we have: $g\gamma \subseteq \gamma\tilde{g}$. One can then prove by induction on k: for all $k \geq 0$, $g^k\gamma(\tilde{S}) \subseteq \gamma\tilde{g}^k(\tilde{S})$. Passing to the limit and using $\gamma\tilde{S} = S_1 \times S_2 = S$, it follows: $G^\times \subseteq G_1^\times \times G_2^\times$.
3. By items 1 and 2, we have: $F^\times \subseteq G_1^\times \times G_2^\times \subseteq S$. It follows by by Proposition 2.1: $F^\times = \bigcap_{k\geq 0} f^k(G_1^\times \times G_2^\times)$.
4. Using $\tilde{f} = \alpha f \gamma$ and $\gamma\alpha \supseteq Id_c$ (Proposition 2.5), we can prove by induction on ℓ: $\tilde{f}^\ell \supseteq \alpha f^\ell \gamma$, for $\ell \geq 0$. Passing to the limit, we have: $(F_1^\times, F_2^\times) \supseteq (G_1^\times, G_2^\times)$. Hence: $G_1^\times \times G_2^\times \subseteq F_1^\times \times F_2^\times$.

□

It can be interesting to perform the computation of $F^\times$, starting from $G_1^\times \times G_2^\times$ rather than $F_1^\times \times F_2^\times$, because $|G_1^\times \times G_2^\times|$ may be much smaller than $|F_1^\times \times F_2^\times|$. Note that, for $\ell = 1$, g coincides with f, $G^\times$ with $F^\times$, and $G_i^\times$ with $F_i^\times$ $(i = 1, 2)$. Hence Proposition 2.7 is an immediate consequence of Theorem 2.8.

2.3 Example

We will illustrate our approach with the example 6.2 of [4], which is a BN given by the rules:

$A(t+1) = 1 \wedge H(t)$,
$B(t+1) = A(t) \wedge (A(t) \vee C(t))$,
$C(t+1) = I(t)$,
$E(t+1) = 1 \wedge C(t) \wedge (C(t) \vee F(t))$,
$F(t+1) = E(t) \wedge (E(t) \vee G(t))$,
$G(t+1) = 1 \wedge (B(t) \vee E(t))$,
$H(t+1) = F(t) \wedge (F(t) \vee G(t))$,
$I(t+1) = H(t) \wedge (H(t) \vee I(t))$.

This corresponds to state variable $x = (A, B, C, E, F, G, H, I)$ and $S = \{0, 1\}^8$. The system is split in two as follows: $x_1 = (A, F, G, H, I)$, $x_2 = (B, C, E)$, $S_1 = \{0, 1\}^5$, $S_2 = \{0, 1\}^3$.

Computation of $F_1^\times \times F_2^\times$

In order to compute $\text{gfp}(\tilde{f}) = (F_1^\times, F_2^\times)$, we use a strategy related to the application of Bekić-Leszczyłowski theorem (see [16]). Roughly speaking, we compute at step i an intermediate fixed-point $F_{1,i+1}$ starting from the current value $F_{1,i}$ of the 1st component using the current value $F_{2,i}$ of the 2nd component as a parameter. Then, we compute a new intermediate fixed-point $F_{2,i+1}$ starting

from $F_{2,i}$ using $F_{1,i+1}$ as a parameter, and so on alternatively until stabilization. We have:

$F_{1,0} = S_1$,
$F_{2,0} = S_2$.

$F_{1,1} = \{00000, 00001, 00010, 00011, 00100, 00101, 00110, 00111, 01000, 01001, 01010, 01011, 01100, 01101, 01110, 01111, 10000, 10001, 10010, 10011, 10100, 10101, 10110, 10111, 11000, 11001, 11010, 11011, 11100, 11101, 11110, 11111\}$,
$F_{2,1} = \{000, 001, 010, 011, 101, 111\}$.

$F_{1,2} = \{00000, 00010, 00100, 00110, 01000, 01010, 01011, 01100, 01101, 01110, 01111, 10000, 10010, 10100, 10110, 11000, 11010, 11011, 11100, 11101, 11110, 11111\}$,
$F_{2,2} = F_{2,1}$.

$F_{1,3} = F_{1,2}$.

Hence $F_1^\times \times F_2^\times = F_{1,2} \times F_{2,1}$ has $22 \times 6 = 132$ elements. The computation of $(F_1^\times, F_2^\times)$ is obtained in 2 iterations and 0.87 seconds of CPU time using our prototype implementation (see Section 4). The computation of $F^\times$ by iteration of f starting from $F_1^\times \times F_2^\times$ is then obtained in 11 iterations and takes 0.11 seconds of CPU time.

Computation of $G_1^\times \times G_2^\times$, with $g = f^2$

We have:

$G_{1,0} = S_1$,
$G_{2,0} = S_2$.

$G_{1,1} = \{00000, 00001, 00100, 00101, 00110, 00111, 01010, 01011, 01110, 01111, 10000, 10100, 10110, 11000, 11010, 11100, 11110\}$,
$G_{2,1} = \{000, 001, 011, 101\}$.

$G_{1,2} = \{10000, 11000, 11010, 01011, 01010, 01111, 00100, 10100, 10100\}$,
$G_{2,2} = G_{2,1}$.

$G_{1,3} = G_{1,2}$.

Hence $G_1^\times \times G_2^\times = G_{1,2} \times G_{2,1}$ has only $9 \times 4 = 36$ elements. This shows that the over-approximation $G_1^\times \times G_2^\times$ of $F^\times$ is much finer than $F_1^\times \times F_2^\times$. The computation of $(G_1^\times, G_2^\times)$ is obtained in 2 iterations and 0.84 seconds of CPU time with our prototype implementation. The fixed point $F^\times$ of f is then obtained from $G_1^\times \times G_2^\times$ via 11 applications of f, which takes 0.10 seconds of CPU time. This yields:

$F^\times = \{01111 \cdot 001, 00100 \cdot 011, 01010 \cdot 011, 01011 \cdot 101, 10100 \cdot 000, 10000 \cdot 101, 10100 \cdot 101, 11000 \cdot 101, 11010 \cdot 101\}$
where each element of $F^\times$ is an instance of $x = x_1 \cdot x_2 = AFGHI \cdot BCE$. It

is easy to see that all the elements of $F^\times$ here belong to a *unique* attractor of length 9 (i.e., $F^\times$ is of the form $\{\sigma_1, \ldots, \sigma_9\}$ with $f(\sigma_i) = \sigma_{i+1}$ for $1 \leq i \leq 8$, and $f(\sigma_9) = \sigma_1$).

By comparison, the global computation of $F^\times$ by iterated application of f to S takes 12 iterations and 0.92 seconds of CPU time. On such a small example, the compositional approach does not bring a significant difference with the global approach, neither in terms of number of iterations nor in terms of computation time.

Remark: We have described here a method which computes attractors for a system split into two sub-systems. The extension to more than two sub-systems is straightforward.

3 Basins of attraction

We now reintroduce the set U of control variables.

3.1 Concrete functions

Given a *stationary point* σ of f (i.e., an element σ of $S = S_1 \times S_2$ such that $f(\sigma, u) = \sigma$ for some $u \in U$), it is interesting to compute the *basin of attraction* of σ, i.e.: the set of elements x such that $f(\cdots f(f(x, u_1), u_2), \ldots, u_k) \ldots) = \sigma$ for some $k > 0$ and $(u_1, u_2, \ldots, u_k) \in U^k$.

Classically, *backward reachability* procedures are used (see, e.g., [18]) to compute basins of attraction. Let us define the *predecessor* operator. For all $X \subseteq S$:

$$p(X) = \{y \in S \mid \exists u \in U : \ f(y, u) \in X\}.$$

As usual, we define $p^k(X)$ by: $p^0(X) = X$ and $p^{k+1}(X) = p(p^k(X))$ for $k \geq 0$. Since σ is a stationary point, the sequence $\{p^i(\{\sigma\})\}_{k \geq 0}$ is increasing. When the fixed-point $p^*(\{\sigma\})$ is reached, at step j (i.e. $p^{j+1}(\{\sigma\}) = p^j(\{\sigma\})$), we have: $p^*(\{\sigma\}) = \bigcup_{k \geq 0} p^k(\{\sigma\}) = p^j(\{\sigma\})$. Since $\{\sigma\} \subseteq p(\{\sigma\})$, the set $p^*(\{\sigma\})$ is, by Kleene fixed-point theorem, the *least fixed-point* of p containing σ; it coincides with the smallest *prefixed-point* of p containing σ, i.e., the smallest set $X \in 2^S$ containing σ such that $\forall x \in X \exists u \in U \ f(x, u) \in X$. We have:

Proposition 3.1. *The basin of attraction of σ is equal to*

$$p^*(\{\sigma\}) = \bigcup_{k \geq 0} p^k(\{\sigma\}).$$

The counterpart of Proposition 2.1 is:

Proposition 3.2. *Suppose: $\{\sigma\} \subseteq X \subseteq p^*(\{\sigma\})$. Then: $p^*(\{\sigma\}) = \bigcup_{k \geq 0} p^k(X)$.*

The proof is analogous to that of Proposition 2.1.

3.2 Abstract functions

We now focus on systems of the form (2)-(3), i.e. that can be decomposed in two parts, this along S and along U. A stationary point σ is of the form (σ_1, σ_2) with $\sigma_1 \in S_1 = \{0,1\}^{n_1}$, $\sigma_2 \in S_2 = \{0,1\}^{n_2}$. Let $\tilde{\sigma} = (\{\sigma_1\}, \{\sigma_2\})$.

We are going to introduce two abstractions $\tilde{p}_{X_2}(X_1)$ and $\tilde{p}_{X_1}(X_2)$ of p. We will then compute least fixed-points, denoted by $\tilde{p}^*_{S_2}(\{\sigma_1\})$ and $\tilde{p}^*_{S_1}(\{\sigma_2\})$, of $\tilde{p}_{S_2}(\cdot)$ and $\tilde{p}_{S_1}(\cdot)$ containing $\{\sigma_1\}$ and $\{\sigma_2\}$ respectively. We will show that $\tilde{p}^*(\tilde{\sigma}) = (\tilde{p}^*_{S_2}(\{\sigma_1\}), \tilde{p}^*_{S_1}(\{\sigma_2\}))$ satisfies $\{\sigma\} \subseteq \gamma\tilde{p}^*(\tilde{\sigma}) \subseteq p^*(\{\sigma\})$. Hence, by Proposition 3.2, the basin of σ can be obtained by iteratively applying p to $\gamma\tilde{p}^*(\tilde{\sigma})$ instead of $\{\sigma\}$. This may reduce the computation time of the basin of σ.

We introduce the following (controlled) abstract predecessor operators:

$$\tilde{p}_{X_2}(X_1) = \{y_1 \in S_1 |\ \exists u_1 \in U_1,\ \forall x_2 \in X_2, f_1(y_1, x_2, u_1) \in X_1\}, \tag{5}$$

$$\tilde{p}_{X_1}(X_2) = \{y_2 \in S_2 |\ \exists u_2 \in U_2,\ \forall x_1 \in X_1, f_2(x_1, y_2, u_2) \in X_2\}. \tag{6}$$

We denote by $\tilde{p}^*_{S_2}(\{\sigma_1\})$ and $\tilde{p}^*_{S_1}(\{\sigma_2\})$ the least fixed-points obtained by iterative application of $\tilde{p}_{S_2}(\cdot)$ and $\tilde{p}_{S_1}(\cdot)$ starting from $\{\sigma_1\}$ and $\{\sigma_2\}$ respectively. Finally, we write $\tilde{p}^*(\tilde{\sigma}) = (\tilde{p}^*_{S_2}(\{\sigma_1\}), \tilde{p}^*_{S_1}(\{\sigma_2\}))$.

Lemma 3.3. *For all $(X_1, X_2) \in 2^{S_1} \times 2^{S_2}$, we have $\gamma(\tilde{p}_{S_2}(X_1), \tilde{p}_{S_1}(X_2)) \subseteq p\gamma(X_1, X_2)$.*

Proof. Let $w = (w_1, w_2) \in \gamma(\tilde{p}_{S_2}(X_1), \tilde{p}_{S_1}(X_2))$. We know that:

$$\exists u_1 \in U_1,\ \forall x_2 \in S_2, f_1(w_1, x_2, u_1) \in X_1$$

and

$$\exists u_2 \in U_2,\ \forall x_1 \in S_1, f_2(x_1, w_2, u_2) \in X_2.$$

In particular:

$$\exists u_1 \in U_1,\ f_1(w_1, w_2, u_1) \in X_1$$

and

$$\exists u_2 \in U_2,\ f_2(w_1, w_2, u_2) \in X_2.$$

Hence:

$$\exists u = (u_1, u_2) \in U,\ f((w_1, w_2), u) = (f_1(w_1, w_2, u_1), f_2(w_1, w_2, u_2)) \in X_1 \times X_2,$$

i.e. $w \in p\gamma(X_1, X_2)$. □

Theorem 3.4. *We have:*

1. $\{\sigma\} \subseteq \gamma\tilde{p}^*(\tilde{\sigma}) \subseteq p^*(\{\sigma\})$.
2. $p^*(\{\sigma\}) = \bigcup_{k\geq 0} p^k(\gamma\tilde{p}^*(\tilde{\sigma}))$.

Proof. 1. Ones proves that, for all $k \geq 0$, $\{\sigma\} \subseteq \bigcup_{j\leq k} \gamma(\tilde{p}^j_{S_2}(X_1), \tilde{p}^j_{S_1}(X_2)) \subseteq \bigcup_{j\leq k} p^j\gamma(\tilde{\sigma})$ by induction on k, using Lemma 3.3. Passing to the limit, it follows: $\{\sigma\} \subseteq \gamma\tilde{p}^*(\tilde{\sigma}) \subseteq p^*(\{\sigma\})$, using $\gamma(\tilde{\sigma}) = \{\sigma\}$.

2. Since $\{\sigma\} \subseteq \gamma\tilde{p}^*(\tilde{\sigma}) \subseteq p^*(\{\sigma\})$ by item 1, it follows by Proposition 3.2: $p^*(\{\sigma\}) = \bigcup_{k\geq 0} p^k(\gamma\tilde{p}^*(\tilde{\sigma}))$.

□

Remark:

Note that it is possible to extend the definitions (5) and (6) to the use of sequences of control inputs using the following definitions:

$$\tilde{p}^{\ell_1}_{X_2}(X_1) = \{y_1 \in S_1 \mid \exists u_1^1 \dots u_1^{\ell_1} \in U_1^{\ell_1}\ , \forall u_2^1 \dots u_2^{\ell_1} \in U_2^{\ell_1} \\ \pi_1 f(\dots f(f((y_1, S_2), (u_1^1, u_2^1)), (u_1^2, u_2^2)) \dots, (u_1^{\ell_1}, u_2^{\ell_1})) \in X_1\},$$

$$\tilde{p}^{\ell_2}_{X_1}(X_2) = \{y_2 \in S_2 \mid \exists u_2^1 \dots u_2^{\ell_2} \in U_2^{\ell_2}\ , \forall u_1^1 \dots u_1^{\ell_2} \in U_1^{\ell_2} \\ \pi_2 f(\dots f(f((S_1, y_2), (u_1^1, u_2^1)), (u_1^2, u_2^2)) \dots, (u_1^{\ell_2}, u_2^{\ell_2})) \in X_2\}.$$

4 Experiments

The experiments presented here have been performed with our prototype written in Octave. The computation times given below have been performed on an Intel Core i7-4810MQ CPU running at 2.80GHz with 8GB of RAM memory.

4.1 Attractors

In industrial case studies, such as railway interlocking, it is important to show that all the attractors are cycles of length 1 (*stationary states*). We have tested our method on an example of a railway interlocking system taken from "NXSYS, Signalling and Interlocking Simulator" [11]. The dynamics of the BN model of the system is given in Appendix. This example has 28 variables and 22 parameters. The objective of the analysis is to show that for any valuation ν of the parameters in $\{0,1\}^{22}$, all the attractors are stationary states. We divide the system into 4 sub-systems as explained in Appendix. The computation of the over-approximated set of attractors $\{F^\times_{1,\nu} \times F^\times_{2,\nu} \times F^\times_{3,\nu} \times F^\times_{4,\nu}\}_{\nu\in\{0,1\}^{22}}$ took 2 hours. The computation of the exact set of attractors $\mathcal{A} = \{F^\times_\nu\}_{\nu\in\{0,1\}^{22}}$ then took 12 more hours. The total number of attractors is $|\mathcal{A}| \simeq 24.10^6$, and we check that all the elements of $\mathcal{A}$ are stationary states. By comparison, for a single instantiation ν, the state-of-art program BNS (available at https://people.kth.se/∼dubrova/bns.html) takes 0.02s, which seems to indicate that BNS would take at least twice more time for computing the whole set of attractors of the 2^{22} instantiations of the problem.

4.2 Basins of attraction

We have experimented the compositional computation of basins of attraction on small examples of the literature, e.g.:

– regulation of the mammalian cell cycle [13] (9 variables, 1 input),
– Example 29 of [5] (5 variables and 2 control inputs).

We present here the example of the regulation of the mammalian cell cycle [13], which dynamics is the following:

$$
\begin{aligned}
Y_1(t+1) &= (\bar{U}(t) \wedge \bar{Y}_3(t) \wedge \bar{Y}_4(t) \wedge \bar{Y}_9(t)) \vee (Y_5(t) \wedge \bar{U}(t) \wedge \bar{Y}_9(t)),\\
Y_2(t+1) &= (\bar{Y}_1(t) \wedge \bar{Y}_4(t) \wedge \bar{Y}_9(t)) \vee (Y_5(t) \wedge \bar{Y}_1(t) \wedge \bar{Y}_9(t)),\\
Y_3(t+1) &= Y_2(t) \wedge \bar{Y}_1(t),\\
Y_4(t+1) &= (Y_2(t) \wedge \bar{Y}_1(t) \wedge \bar{Y}_6(t) \wedge (\overline{Y_7(t) \wedge Y_8(t)}))\\
&\quad \vee (Y_4(t) \wedge \bar{Y}_1(t) \wedge \bar{Y}_6(t) \wedge (\overline{Y_7(t) \wedge Y_8(t)})),\\
Y_5(t+1) &= (\bar{U}(t) \wedge \bar{Y}_3(t) \wedge \bar{Y}_4(t) \wedge (\bar{Y}_9(t))\\
&\quad \vee (Y_5(t) \wedge (\overline{Y_3(t) \wedge Y_4(t)}) \wedge \bar{U}(t) \wedge \bar{Y}_9(t)),\\
Y_6(t+1) &= Y_9(t+1),\\
Y_7(t+1) &= (\bar{Y}_4(t) \wedge \bar{Y}_9(t)) \vee Y_6(t) \vee (Y_5(t) \wedge \bar{Y}_9(t)),\\
Y_8(t+1) &= \bar{Y}_7(t) \vee (Y_7(t) \wedge Y_8(t) \wedge (Y_6(t) \vee Y_4(t) \vee Y_9(t))),\\
Y_9(t+1) &= \bar{Y}_6(t) \wedge \bar{Y}_7(t).
\end{aligned}
$$

For this example, it is known that there exists a stationary point σ (associated to a sequence of inputs equal to 1), with $\sigma = \{100010100\}$.

Let us explain how we compute the basin of attraction of σ with the compositional method.We split the system in two: $x_1 = (Y_1, Y_2, Y_3, Y_5)$ and $x_2 = (Y_4, Y_6, Y_7, Y_8, Y_9)$, $S_1 = \{0,1\}^4$ and $S_2 = \{0,1\}^5$. The fixed-points $\tilde{p}^*_{S_2}(\{\sigma_1\})$ and $\tilde{p}^*_{S_1}(\{\sigma_2\})$ are obtained in 2 iterations and 1.71 seconds of CPU time, with sequences of control inputs of length lower than 5. The size of $\tilde{p}^*(\sigma_1, \sigma_2)$ is $16 \times 21 = 336$. By computing the iterated predecessors of $\gamma\tilde{p}^*(\sigma_1, \sigma_2)$ via p, we find in 3.03 seconds and 3 iterations, that the basin of attraction is equal to S. By comparison, the global computation of $p^*(\sigma)$ takes 4.15 seconds and 4 iterations. On such a small example, the results of the two approaches in terms of computation times and number of iterations are similar.

5 Final remarks

We proposed a compositional method based on local fixed-point iterations. The method has been successfully applied to an example with 50 variables modeling a part of New York City subway, which allows us to identify more than 24 millions of attractors. We believe that such a finding of all the attractors would be difficult using a global approach with state-of-the-art tools. Our current implementation of the method is a simple prototype with explicit representation of Boolean states. We are currently integrating symbolic structures (BDDs) to the code in order to treat larger examples. One of our objectives is to find the attractors of a real railway interlocking system provided by Thales.

Acknowledgement. We are most grateful to Philippe Schnoebelen for insightful explanations on Abstract Interpretation and numerous comments on an earlier draft of this paper.

This work is supported by Institut Farman (ENS Cachan) and by the French National Research Agency through the "iCODE Institute project" funded by the IDEX Paris-Saclay, ANR-11-IDEX-0003-02.

References

1. Tatsuya Akutsu, Sven Kosub, Avraham A Melkman & Takeyuki Tamura (2012): *Finding a periodic attractor of a Boolean network. Computational Biology and Bioinformatics, IEEE/ACM Transactions on* 9(5), pp. 1410–1421.
2. Armin Biere, Alessandro Cimatti, Edmund M. Clarke, Masahiro Fujita & Yunshan Zhu (1999): *Symbolic Model Checking Using SAT Procedures instead of BDDs.* In: *DAC*, pp. 317–320, doi:10.1145/309847.309942. Available at http://doi.acm.org/10.1145/309847.309942.
3. Randal E Bryant (1986): *Graph-based algorithms for boolean function manipulation. Computers, IEEE Transactions on* 100(8), pp. 677–691.
4. Daizhan Cheng & Hongsheng Qi (2010): *A linear representation of dynamics of Boolean networks. Automatic Control, IEEE Transactions on* 55(10), pp. 2251–2258.
5. Daizhan Cheng, Hongsheng Qi & Yin Zhao (2011): *On Boolean control networks – an algebraic approach.* In: *Proceedings of the 18th IFAC World Congress, Milano*, pp. 8366–8377.
6. P. Cousot (1999): *The Calculational Design of a Generic Abstract Interpreter.* In M. Broy & R. Steinbrüggen, editors: *Calculational System Design*, NATO ASI Series F. IOS Press, Amsterdam.
7. Patrick Cousot (2001): *Compositional separate modular static analysis of programs by abstract interpretation.* In: *Proc. SSGRR 2001 – Advances in Infrastructure for Electronic Business, Science, and Education on the Internet, 6 – 10.*
8. Martin Davis & Hilary Putnam (1960): *A Computing Procedure for Quantification Theory. J. ACM* 7(3), pp. 201–215, doi:10.1145/321033.321034. Available at http://doi.acm.org/10.1145/321033.321034.
9. Elena Dubrova & Maxim Teslenko (2011): *A SAT-based algorithm for finding attractors in synchronous boolean networks. IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)* 8(5), pp. 1393–1399.
10. Wan Fokkink, Paul Hollingshead, J Groote, S Luttik & J van Wamel (1998): *Verification of interlockings: from control tables to ladder logic diagrams.* In: *Proceedings of FMICS*, 98, pp. 171–185.
11. Bernard S. Greenberg: *NXSYS, Signaling and Interlocking Simulator.* Available at http://www.nycsubway.org/wiki/NXSYS,_Signalling_and_Interlocking_Simulator.
12. W. Guo, G. Yang, W. Wu, L. He & M. Sun (2014): *A Parallel Attractor Finding Algorithm Based on Boolean Satisfiability for Genetic Regulatory Networks. PLoS ONE* 9(4).
13. Gal Hochma, Michael Margaliot, Ettore Fornasini & Maria Elena Valcher (2013): *Symbolic dynamics of Boolean control networks. Automatica* 49(8), pp. 2525–2530.
14. C. Hong, J. Hwang, K.-H. Cho & I. Shin (2015): *A Parallel Attractor Finding Algorithm Based on Boolean Satisfiability for Genetic Regulatory Networks. PLoS ONE* 10(12).

15. Stuart Alan Kauffman (1993): *The origins of order : self organization and selection in evolution.* Oxford university press, New York. Available at http://opac.inria.fr/record=b1077782.
16. Viktor Kuncak & K. Rustan M. Leino (2004): *On computing the fixpoint of a set of boolean equations.* cs.PL/0408045. Available at http://arxiv.org/abs/cs.PL/0408045.
17. Aurélien Naldi, Denis Thieffry & Claudine Chaouiya (2007): *Decision Diagrams for the Representation and Analysis of Logical Models of Genetic Networks.* In: *Computational Methods in Systems Biology, International Conference, CMSB 2007, Edinburgh, Scotland, September 20-21, 2007, Proceedings*, pp. 233–247.
18. Andrew Wuensche et al. (1998): *Discrete dynamical networks and their attractor basins. Complexity International* 6, pp. 3–21.
19. Yin Zhao, Jongrae Kim & Maurizio Filippone (2013): *Aggregation algorithm towards large-scale Boolean network analysis. Automatic Control, IEEE Transactions on* 58(8), pp. 1976–1985.

Appendix: Railway Interlocking System

Railway Interlocking is one part of a complete railway system that ensures the safety of all trains and passengers. Given the routes that trains wish to travel along and current train positions (among other information), the Interlocking computes a safe environment for trains (mainly signals indications, and switches positions).

A part of the railway interlocking functioning can be translated to boolean equations. In this example, we have taken the boolean variables and equations associated with protection, movement command and locking of two railway switches while abstracting away all the timed components of those systems. Computing the attractors allows to find what states those switches will reach upon reception of a new command.

The dynamics of the system tested in Section 4.1 is given by the rules:

23NLP = !23RLP (OR (AND 23ANN 23ANS) (AND 23BNN 23BNS) 18PBS 23NL)
23RLP = !23NLP (OR (AND 23RN 23RS) 23RL)
23ANS = (OR 16PBS 16XR) !23RS !23RWK
23BNS = (OR 25ANS 25RS) !23RN !23RWK
23ANN = (OR 25BNN 25RN) !23RS !23RWK
23BNN = (OR 32XS 32PBS) !23RN !23RWK
23RS = 25ANS !23ANS !23BNN !23NWK
23RN = 25BNN !23ANS !23BNN !23NWK
23NWZ = (OR (AND 23NWZ !23RWZ) (AND 23NLP 23LS))
23RWZ = (OR (AND 23RWZ !23NWZ) (AND 23RLP 23LS))
23NWC = (AND 23NWZ 23NWP)
23RWC = (AND 23RWZ 23RWP)
23NWK = 23NWC (OR 23NLP !23LS)
23RWK = 23RWC (OR 23RLP !23LS)
25ANS = (OR 22PBS 28ZS 22XS 22XR) !25RS !25RWK
25BNS = (OR 23ANS 23RS) !25RN !25RWK

25ANN = (OR 23BNN 23RN) !25RS !25RWK
25BNN = (OR 34XS 34PBS) !25RN !25RWK
25RN = 23BNN !25ANS !25BNN !25NWK
25RS = 23ANS !25ANS !25BNN !25NWK
25NLP = !25RLP (OR (AND 25ANN 25ANS) (AND 25BNN 25BNS) 28XS 25NL)
25RLP = !25NLP (OR (AND 25RN 25RS) 25RL)
25NWZ = (OR (AND 25NWZ !25RWZ) (AND 25NLP 25LS))
25RWZ = (OR (AND 25RWZ !25NWZ) (AND 25RLP 25LS))
25NWC = (AND 25NWZ 25NWP)
25RWC = (AND 25RWZ 25RWP)
25NWK = 25NWC (OR 25NLP !25LS)
25RWK = 25RWC (OR 25RLP !25LS)

The left-hand sides of these equations correspond to the 28 state variables. The other expressions appearing in the right-hand sides correspond to the 22 parameters.

The system is split into 4 sub-systems as follows:

- For sub-system 1, the state variables are: 23NLP, 23RLP, 23ANS, 23BNS, 23ANN, 23BNN, 23RS and 23RN.
  The parameters are: 18PBS, 23RL, 16PBS, 16XR, 32XS, 32PBS and 23NL.
- For sub-system 2, the state variables are: 23NWZ, 23RWZ, 23NWC, 23RWC, 23NWK and 23RWK.
  The parameters are: 23LS, 23NWP and 23RWP.
- For sub-system 3, the state variables are: 25ANS, 25BNS, 25ANN, 25BNN, 25RN, 25RS and 25NLP.
  The parameters are: 22PBS, 28ZS, 22XS, 22XR, 34XS, 34PBS, 28XS and 25NL.
- For sub-system 4, the state variables are: 25RLP, 25NWZ, 25RWZ, 25NWC, 25RWC, 25NWK and 25RWK.
  The parameters are: 25RL, 25LS, 25NWP and 25RWP.