Reduction of Qualitative Models of Biological Networks for Transient Dynamics Analysis

Loïc Paulevé

Reduction of Qualitative Models of Biological Networks for Transient Dynamics Analysis

Loïc Paulevé

Abstract—Qualitative models of dynamics of signalling pathways and gene regulatory networks allow to capture temporal properties of biological networks while requiring few parameters. However, these discrete models typically suffer from the so-called state space explosion problem which makes the formal assessment of their potential behaviours very challenging.
In this paper, we describe a method to reduce a qualitative model for enhancing the tractability of analysis of transient reachability properties. The reduction does not change the dimension of the model, but instead limits its degree of freedom, therefore reducing the set of states and transitions to consider. We rely on a transition-centered specification of qualitative models by the mean of automata networks. Our framework encompass usual asynchronous Boolean and multi-valued network, as well as 1-bounded Petri nets.
Applied to different large-scale biological networks from the litterature, we show that the reduction can lead to drastic improvement for the scalability of verification methods.

Index Terms—Model reduction, reachability, automata networks, Petri nets, systems biology

1 Introduction

Automata networks model dynamical systems resulting from simple interactions between entities. Each entity is typically represented by an automaton with few internal states which evolve subject to the state of a narrow range of other entities in the network. Richness of emerging dynamics arises from several factors including the topology of the interactions, the presence of feedback loop, and the concurrency of transitions.

Automata networks, which subsume Boolean and multi-valued networks, are notably used to model dynamics of biological systems, including signalling networks or gene regulatory networks (e.g., [1], [2], [3], [4], [5], [6], [7], [8]). The resulting models can then be confronted with biological knowledge, for instance by checking if some time series data can be reproduced by the computational model. In the case of models of signalling or gene regulatory networks, such data typically refer to the possible activation of a transcription factor, or a gene, from a particular state of the system, which reflects both the environment and potential perturbations. Automata networks have also been used to infer targets to control the behaviour of the system. For instance, in [1], [6], the author use Boolean networks to find combinations of signals or combinations of mutations that should alter the cellular behaviour.

From a formal point of view, numerous biological properties can be expressed in computation models as reachability properties: from an initial state, or set of states, the existence of a sequence of transitions which leads to a desired state, or set of states. For instance, an initial state can represent a combination of signals/perturbations of a signalling network; and the desired states the set of states where the concerned transcription factor is active. One can then verify the (im)possibility of such an activation, possibly by taking into account mutations, which can be modelled, for instance, as the freezing of some automata to some fixed states, or by the removal of some transitions.

Due to the increasing precision of biological knowledge, models of networks become larger and larger and can gather hundreds to thousands of interacting entities making the formal analysis of their dynamics a challenging task: the reachability problem in automata networks/bounded Petri nets is PSPACE-complete [9], which limits its scalability.

Facing a model too large for a raw exhaustive analysis, a natural approach is to reduce its dynamics while preserving important properties. Multiple approaches, often complementary, have been explored since decades to address such a challenge in dynamical and concurrent systems [10], [11], [12]. In the scope of rule-based models of biological networks, efficient static analysis methods have been developed to lump numerous global states of the systems based on the fragmentation of interacting components [13]; and to *a posteriori* compress simulated traces to obtain compact witnesses of dynamical properties [14]. Reductions preserving the attractors of dynamics (long-term/steady-state behaviour) have also been proposed for chemical reaction networks [15] and Boolean networks [16]. The latter approach applies to formalisms close to automata networks but does not preserve reachability properties. On Petri nets, different structural reductions have been proposed to reduce the size of the model specification while preserving bisimulation [17], or liveness and LTL properties [18], [19]. Procedures such as the cone of influence reduction [20] or relevant subnet computation [21] allow identifying variables/transitions which have no influence on a given dynamical property. Our work has a motivation similar to the two latter approaches.

In this paper, we define a reduction of automata networks which identifies transitions that do not contribute to a given reachability property and hence can be ignored. The considered automata networks are finite sets of finite-state machines where transitions between their local states are

CNRS, LRI UMR 8623, Univ. Paris-Sud – CNRS, Université Paris-Saclay, 91405 Orsay, France
E-mail: loic.pauleve@lri.fr – http://loicpauleve.name

conditioned by the state of other automata in the network. We use a general concurrent semantics where any number of transitions can be applied in one step. We call a *trace* a sequential interleaved execution of steps. Our framework encompass usual Boolean and multi-valued networks, as well as standard 1-bounded/safe Petri nets.

Our reduction preserves all the minimal traces satisfying reachability properties of the form "from state s there exist successive steps that lead to a state where a given automaton g is in local state $g_\top$". A trace is *minimal* if no step nor transition can be removed from it and resulting in a sub-trace that satisfies the concerned reachability property. The complexity of the procedure is polynomial in the number of local transitions, and exponential in the maximal size of automata. Therefore, the reduction is scalable for networks of multiple automata, where each have a few local states.

The identification of the transitions that are not part of any minimal trace is performed by a static analysis of the causality of transitions within automata. It extends previous static analysis of reachability properties by abstract interpretation [22], [23]. In [22], necessary or sufficient conditions for reachability are derived, but they do not allow to capture all the (minimal) traces towards a reachability goal. In [23], the static analysis extracts local states, referred to as cut-sets, which are necessarily reached prior to a given reachability goal. The results presented here are orthogonal: we identify transitions that are never part of a minimal trace for the given reachability property. It allows us to output a reduced model where all such transitions are removed while preserving all the minimal traces for reachability. Hence, whereas [23] focuses on identifying necessary conditions for reachability, this article focuses on preserving sufficient conditions for reachability.

The effectiveness of our goal-oriented reduction is experimented on actual models of biological networks and show significant shrinkage of the dynamics of the automata networks, enhancing the tractability of a concrete verification. Compared to other model reductions, our goal is similar to the cone of influence reduction [20] or relevant subnet computation [21] mentioned above, which identify variables/transitions that do not impact a given property. Here, our approach offers a much more fine-grained analysis in order to identify the sufficient transitions and values of variables that contribute to the property, which leads to stronger reductions, although preserving less temporal properties.

This paper is an extension of the conference paper [24]. The automata network framework has been enhanced to account for coupled (or synchronized) transitions. This adds the expressivity to encode 1-bounded Petri nets (appendix A), and lead to generalize the reduction method and its proof (appendix B). This paper also details the algorithmic aspect of our goal-oriented reduction (section 3.2.2), and the application of automata networks to biological networks (section 4), notably by showing and proving the encoding of Boolean and multi-valued networks.

Outline

Section 2 sets up the definition and semantics of the automata networks, together with the local causality analysis for reachability properties, based on prior work. Section 3 first depicts a necessary condition using local causality analysis for satisfying a reachability property and then details the goal-oriented reduction which preserves minimal traces. Section 4 shows how automata networks and their reduction can be applied to biological models. Section 5 benchmark the tractability of transient analysis before and after goal-reduction on a range of biological networks. Finally, section 6 discusses the results and motivates further work.

Notations

Integer ranges are noted $[m;n] \triangleq \{m, m+1, \cdots, n\}$. Given a finite set A, $|A|$ is the cardinality of A; 2^A is the power set of A. Given $n \in \mathbb{N}$, $x = (x^i)_{i\in[1;n]}$ is a sequence of elements indexed by $i \in [1;n]$; $|x| = n$; $x^{m..n}$ is the subsequence $(x^i)_{i\in[m;n]}$; $x :: e$ is the sequence x with an additional element e at the end; ε is the empty sequence.

2 Automata Networks and Local Causality

2.1 Automata Networks

An Automata Network (AN) is composed of a finite set of finite-state machines having transitions between their local states conditioned by the state of other automata in the network. In this paper, we allow the local transitions to be coupled, or synchronized, as it is common for Synchronous Products of Transition Systems [25].

An AN is defined by a triple (Σ, S, T) (definition 1) where Σ is the set of automata identifiers; S associates to each automaton a finite set of local states: if $a \in \Sigma$, $S(a)$ refers to the set of local states of a; T is the set of local transitions. Each local state is written of the form a_i, where $a \in \Sigma$ is the automaton in which the state belongs to, and i is a unique identifier; therefore given $a_i, a_j \in S(a)$, $a_i = a_j$ if and only if a_i and a_j refer to the same local state of the automaton a.

A local transition $t \in T$ is a pair $t = (l, x)$ where l is the set of local state changes for the automata in which the transition takes place, and x is the set of local states of other automata that are necessary to trigger the transition. The local state changes l are specified by pairs of local states of a same automaton, indicating the starting local state and the ending local state.

For instance, a local transition $t = (l, x)$ with $l = \{(a_i, a_j)\}$ and $x = \{b_k\}$ specifies that, when b is in state b_k, automaton a can change from state a_i to state a_j. If $l = \{(a_i, a_j), (c_q, c_r)\}$, the transition specifies that automata a and c change simultaneously from their respective local states a_i and c_q to a_j and c_r. Such a transition is applicable only if a and c are in states a_i and c_q, respectively.

The *pre-condition* of transition $t = (l, x)$, noted ${}^\bullet t$, is the set composed of the starting local states specified in l and of the local states in x; the *post-condition*, noted $t^\bullet$ is the set composed of the ending local states specified in l and of the local states in x.

Definition 1 (Automata Network (Σ, S, T))**.** An *Automata Network* (AN) is defined by a tuple (Σ, S, T) where

- Σ is the finite set of automata identifiers;
- For each $a \in \Sigma$, $S(a) = \{a_i, \ldots, a_j\}$ is the finite set of local states of automaton a; $S \triangleq \prod_{a\in\Sigma} S(a)$ is the

finite set of global states;
$L \triangleq \bigcup_{a\in\Sigma} S(a)$ denotes the set of all the local states.

- $T \subseteq \{2^{\bigcup_{a\in\Sigma} S(a)\times S(a)} \times 2^L\}$ is the set of local transitions $t = (l, x)$ where l is a set of local state changes (pairs of automata local states) and x is a condition (set of local states) such that: $\forall (l, x) \in T$, $\forall a \in \Sigma, |\{(a_i, a_j) \in l\}| \leq 1$ and $\{(a_i, a_j) \in l\} \neq \emptyset \Rightarrow x \cap S(a) = \emptyset$; $\forall b \in \Sigma, |x \cap S(b)| \leq 1$.

Given $t = (l, x) \in T$, $\mathrm{orig}(t) \triangleq \{a_i \mid (a_i, a_j) \in l\}$, $\mathrm{dest}(t) \triangleq \{a_j \mid (a_i, a_j) \in l\}$, $\mathrm{enab}(t) \triangleq x$, ${}^\bullet t \triangleq \mathrm{orig}(t) \cup \mathrm{enab}(t)$, and $t^\bullet \triangleq \mathrm{dest}(t) \cup \mathrm{enab}(t)$. We write $a_i \xrightarrow{x} a_j$ for $(\{(a_i, a_j)\}, x)$ and $a_i, \ldots, b_m \xrightarrow{x} a_j, \ldots, b_n$ for $(\{(a_i, a_j), \ldots, (b_m, b_n)\}, x)$.

At any time, each automaton is in one and only one local state, forming the global state of the network. Assuming an arbitrary ordering between automata identifiers, the set of global states of the network is referred to as S as a shortcut for $\prod_{a\in\Sigma} S(a)$. Given a global state $s \in S$, $s(a)$ is the local state of automaton a in s, i.e., the a-th coordinate of s. Moreover we write $a_i \in s \triangleq s(a) = a_i$; and for any $x \in 2^L$, $x \subseteq s \triangleq \forall a_i \in x, s(a) = a_i$.

In the scope of this paper, we allow, but do not enforce, the parallel application of local transitions. This leads to the definition of a *step* as a set of transitions, with at most one change per automaton (definition 2). For notational convenience, we allow empty steps. The pre-condition (resp. post-condition) of a step τ, noted ${}^\bullet\tau$ (resp. $\tau^\bullet$), extends the similar notions on transitions: the pre-condition (resp. post-condition) is the union of the pre-conditions (resp. post-conditions) of composing transitions. A step τ is *playable* in a state $s \in S$ if and only if ${}^\bullet\tau \subseteq s$, i.e., all the local states in the pre-conditions of transitions are in s. If τ is playable in s, $s \cdot \tau$ denotes the state after the applications of all the transitions in τ, i.e., where for each transition $(l, x) \in \tau$, $\forall (a_i, a_j) \in l$, the local state a_i of automaton a has been replaced with a_j.

Definition 2 (Step)**.** Given an AN (Σ, S, T), a *step* $\tau \subseteq T$ is a subset of local transitions T where, for each automaton $a \in \Sigma$, there is at most one local change of a: $|\{(l, x) \in \tau \mid \exists (a_i, a_j) \in l\}| \leq 1$; and the pre-condition of compositing local transitions are compatible: $|S(a) \cap \bigcup_{t\in\tau} {}^\bullet t| \leq 1$.
We note ${}^\bullet\tau \triangleq \bigcup_{t\in\tau} {}^\bullet t$ and $\tau^\bullet \triangleq \bigcup_{t\in\tau} t^\bullet \setminus \bigcup_{t\in\tau} \mathrm{orig}(t)$.
Given a state $s \in S$ where τ is playable (${}^\bullet\tau \subseteq s$), $s \cdot \tau$ denotes the state where $\forall a \in \Sigma$, $(s \cdot \tau)(a) = a_j$ if $\exists (l, x) \in \tau$ s.t. $(a_i, a_j) \in l$; otherwise, $(s \cdot \tau)(a) = s(a)$.

Remark that $\tau^\bullet \subseteq s \cdot \tau$; and that ${}^\bullet\tau \subseteq s$ implies that any sub-step $\tau' \subseteq \tau$ is also playable in s: ${}^\bullet\tau' \subseteq s$.

A *trace* (definition 3) is a sequence of successively playable steps from a state $s \in S$. The pre-condition ${}^\bullet\pi$ of a trace π is the set of local states that are required to be in s for applying π (${}^\bullet\pi \subseteq s$); and the post-condition $\pi^\bullet$ is the set of local states that are present in the state after the full application of π ($\pi^\bullet \subseteq s \cdot \pi$).

Definition 3 (Trace)**.** Given an AN (Σ, S, T) and a state $s \in S$, a *trace* π is a sequence of steps such that $\forall i \in [1; |\pi|]$, ${}^\bullet\pi^i \subseteq (s \cdot \pi^1 \cdot \cdots \cdot \pi^{i-1})$.
The pre-condition ${}^\bullet\pi$ and the post-condition $\pi^\bullet$ are defined as follows: for all $n \in [1; |\pi|]$, for all $a_i \in {}^\bullet\pi^n$, $a_i \in {}^\bullet\pi \triangleq \forall m \in [1; n-1], S(a) \cap {}^\bullet\pi^m = \emptyset$; similarly, for all $n \in [1; |\pi|]$, for all $a_j \in \pi^{n\bullet}$, $a_j \in \pi^\bullet \triangleq \forall m \in [n+1; m], S(a) \cap \pi^{m\bullet} = \emptyset$.
If π is empty, ${}^\bullet\pi = \pi^\bullet = \emptyset$.
The set of transitions composing a trace π is noted $\mathrm{tr}(\pi) \triangleq \bigcup_{n=1}^{|\pi|} \pi^n$.

Given an automata network (Σ, S, T) and a state $s \in S$, the local state $g_\top \in L$ is *reachable* from s if and only if either $g_\top \in s$ or there exists a trace π with ${}^\bullet\pi \subseteq s$ and $g_\top \in \pi^\bullet$.

We consider a trace π for $g_\top$ reachability from s is *minimal* if and only if there exists no different trace reaching $g_\top$ having each successive step being a subset of a step in π with the same ordering (definition 4). Say differently, a trace is minimal for $g_\top$ reachability if no step or transition can be removed from it without breaking the trace validity or $g_\top$ reachability.

Definition 4 (Minimal trace for local state reachability)**.** A trace π is *minimal* w.r.t. $g_\top$ reachability from s if and only if there is no trace ϖ from s, $\varpi \neq \pi$, $|\varpi| \leq |\pi|$, $g_\top \in \varpi^\bullet$, such that there exists an injection $\phi : [1; |\varpi|] \to [1; |\pi|]$ with $\forall i, j \in [1; |\varpi|]$, $i < j \Leftrightarrow \phi(i) < \phi(j)$ and $\varpi^i \subseteq \pi^{\phi(i)}$.

Automata networks as defined here are very similar to 1-bounded (also called safe) Petri nets [26]. Actually, any 1-bounded Petri net can be encoded as AN and conversely, as detailed in appendix A. The methods presented in this paper, and in particular the local causality analysis, rely of the component (automata) decomposition of the system, therefore automata networks have the advantage of making this partition explicit.

The semantics considered in this paper where transitions can be applied in parallel echoes with Petri net step-semantics and concurrent/maximally concurrent semantics [27], [28], [29]. In the Boolean network community, such a semantics is referred to as the asynchronous generalized update schedule [30].

Example 1. Let us consider the automata network (Σ, S, T), graphically represented in figure 1, where:

$$\Sigma = \{a, b, c, d\}$$
$$S(a) = \{a_0, a_1\} \qquad T = \{a_0 \xrightarrow{\{b_0\}} a_1; b_0 \xrightarrow{\{a_1\}} b_1;$$
$$S(b) = \{b_0, b_1\} \qquad a_1, b_1 \xrightarrow{\emptyset} a_0, b_0;$$
$$S(c) = \{c_0, c_1, c_2\} \qquad c_0 \xrightarrow{\{a_1\}} c_1; c_1 \xrightarrow{\{b_1\}} c_0;$$
$$S(d) = \{d_0, d_1\} \qquad c_1 \xrightarrow{\{b_0\}} c_2; c_0 \xrightarrow{\{d_1\}} c_2\}$$

Starting from the state $s = \langle a_0, b_0, c_0, d_0\rangle$, only one transition is playable: $t = a_0 \xrightarrow{b_0} a_1$. Playing t updates the state of automaton a: $s \cdot t = \langle a_1, b_0, c_0, d_0\rangle$. From this new state, two transitions are playable: $b_0 \xrightarrow{a_1} b_1$ and $c_0 \xrightarrow{a_1} c_1$. This lead to several possible playable steps: $\tau^1 = \{b_0 \xrightarrow{a_1} b_1\}$, $\tau^2 = \{c_0 \xrightarrow{a_1} c_1\}$, $\tau^3 = \{b_0 \xrightarrow{a_1} b_1; c_0 \xrightarrow{a_1} c_1\}$. Playing τ^3 results in state $\langle a_1, b_1, c_1, d_0\rangle$. In this state, the synchronized transition $a_1, b_1 \xrightarrow{\emptyset} a_0, b_0$ is playable, which results in state $\langle a_0, b_0, c_1, d_0\rangle$.

From the state $\langle a_0, b_0, c_0, d_0\rangle$, instances of traces are
$\pi^1 = \{a_0 \xrightarrow{\{b_0\}} a_1\} :: \{b_0 \xrightarrow{\{a_1\}} b_1; c_0 \xrightarrow{\{a_1\}} c_1\}$
$:: \{a_1, b_1 \xrightarrow{\emptyset} a_0, b_0\} :: \{c_1 \xrightarrow{\{b_0\}} c_2\}$;
$\pi^2 = \{a_0 \xrightarrow{\{b_0\}} a_1\} :: \{c_0 \xrightarrow{\{a_1\}} c_1\} :: \{c_1 \xrightarrow{\{b_0\}} c_2\}$

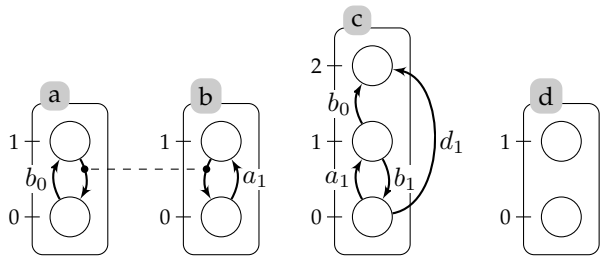

Fig. 1. An example of automata network. Automata are represented by labelled boxes, and local states by circles where ticks are their identifier within the automaton – for instance, the local state a_0 is the circle ticked 0 in the box a. A transition is a directed edge between two local states within the same automaton. It can be labelled with a set of local states of other automata. In this example, all the transitions are conditioned by at most one other local state. Synchronized transitions are linked with a dashed edge.

the latter only being a minimal trace for c_2 reachability.

2.2 Local Causality

Locally reasoning within one automaton a, the reachability of one of its local state a_j from some global state s with $s(a) = a_i$ can be described by a (local) *objective*, that we note $a_i \rightsquigarrow a_j$ (definition 5).

Definition 5 (Objective)**.** Given an automata network (Σ, S, T), an *objective* is a pair of local states $a_i, a_j \in S(a)$ of a same automaton $a \in \Sigma$ and is denoted $a_i \rightsquigarrow a_j$. The set of all objectives is referred to as $\mathbf{Obj} \triangleq \{a_i \rightsquigarrow a_j \mid (a_i, a_j) \in S(a) \times S(a), a \in \Sigma\}$.

Given an objective $a_i \rightsquigarrow a_j \in \mathbf{Obj}$, $\mathrm{lpaths}(a_i \rightsquigarrow a_j)$ is the set of path of transitions of T that are acyclic within automaton a from a_i to a_j (definition 6).

Definition 6 (lpaths)**.** Given $a_i \rightsquigarrow a_j \in \mathbf{Obj}$, if $i = j$, $\mathrm{lpaths}(a_i \rightsquigarrow a_i) \triangleq \{\varepsilon\}$; if $i \neq j$, a sequence η of transitions in T is in $\mathrm{lpaths}(a_i \rightsquigarrow a_j)$ if and only if $|\eta| \geq 1$, $a_i \in \mathrm{orig}(\eta^1)$, $a_j \in \mathrm{dest}(\eta^{|\eta|})$, $\forall n \in [1; |\eta| - 1]$, $\exists a_k \in S(a)$ s.t. $a_k \in \mathrm{dest}(\eta^n)$, $a_k \in \mathrm{orig}(\eta^{n+1})$, and $\forall n, m \in [1; |\eta|], n > m \Rightarrow S(a) \cap \mathrm{dest}(\eta^n) \cap \mathrm{orig}(\eta^m) = \emptyset$.

As stated by property 1, any trace reaching a_j from a state containing a_i uses all the transitions of at least one local acyclic path in $\mathrm{lpaths}(a_i \rightsquigarrow a_j)$.

Property 1. For any trace π, for any $a \in \Sigma$, $a_i, a_j \in S(a)$, $1 \leq n \leq m \leq |\pi|$ where $a_i \in {}^\bullet\pi^n$ and $a_j \in \pi^{m\bullet}$, there exists a local acyclic path $\eta \in \mathrm{lpaths}(a_i \rightsquigarrow a_j)$ that is a subsequence of $\pi^{n..m}$, i.e., there is an injection $\phi : [1; |\eta|] \to [n; m]$ with $\forall u, v \in [1; |\eta|], u < v \Leftrightarrow \phi(u) < \phi(v)$ and $\eta^u \in \pi^{\phi(u)}$.

A local acyclic path being of length at most $|S(a)|$ with unique transitions, the number of local acyclic paths is polynomial in the number of transitions $T(a)$ and exponential in the number of local states in a minus 1, $|S(a)| - 1$.

Finally, let us remark that a local path is not necessarily a trace, as transitions may be conditioned by the state of other automata that may need to be reached beforehand.

Example 2. Considering the AN of figure 1,

$$\mathrm{lpaths}(a_0 \rightsquigarrow a_1) = \{a_0 \xrightarrow{b_0} a_1\}$$
$$\mathrm{lpaths}(a_1 \rightsquigarrow a_0) = \mathrm{lpaths}(b_1 \rightsquigarrow b_0) = \{a_1, b_1 \xrightarrow{\emptyset} a_0, b_0\}$$
$$\mathrm{lpaths}(c_0 \rightsquigarrow c_2) = \{c_0 \xrightarrow{\{a_1\}} c_1 :: c_1 \xrightarrow{\{b_0\}} c_2; c_0 \xrightarrow{\{d_1\}} c_2\}$$
$$\mathrm{lpaths}(c_1 \rightsquigarrow c_2) = \{c_1 \xrightarrow{\{b_0\}} c_2; c_1 \xrightarrow{\{b_1\}} c_0 :: c_0 \xrightarrow{\{d_1\}} c_2\}$$
$$\mathrm{lpaths}(d_0 \rightsquigarrow d_1) = \emptyset \qquad \mathrm{lpaths}(a_0 \rightsquigarrow a_0) = \{\varepsilon\}$$

3 Goal-Oriented Reduction

Assuming a global AN (Σ, S, T), an initial state $s \in S$ and a reachability goal $g_\top$ where $g \in \Sigma$ and $g_\top \in S(g)$, the goal-oriented reduction identifies a subset of local transitions T that are sufficient for producing all the minimal traces leading to $g_\top$ from s. The reduction procedure takes advantage of the local causality analysis both to fetch the transitions that matter for the reachability goal and to filter out objectives that can be statically proven impossible.

3.1 Necessary condition for local reachability

Given an objective $a_i \rightsquigarrow a_j$ and a global state $s \in S$ where $s(a) = a_i$, prior work has demonstrated necessary conditions for the existence of a trace leading to a_j from s [22], [23]. Those necessary conditions rely on the local causality analysis defined in previous section for extracting necessary steps that have to be performed in order to reach the concerned local state.

Several necessary conditions have been established in [22], taking into account several features captured by the local paths (dependencies, sequentiality, partial order constraints, ...). The complexity of deciding most of these necessary conditions is polynomial in the total number of local transitions $|T|$ and exponential in $k - 1$, where k the maximum number of local states within an automaton, i.e., $k = \max_{a \in \Sigma} |S(a)|$.

In this section, we consider a generic reachability over-approximation predicate $\mathbf{valid}_s$ which is false only when applied to an objective that has no concrete trace from s: a_j is reachable from s with $s(a) = a_i$ only if $\mathbf{valid}_s(a_i \rightsquigarrow a_j)$.

Definition 7 ($\mathbf{valid}_s$). For all $a \in S$, for all $a_i, a_j \in S(a)$, $\mathbf{valid}_s(a_i \rightsquigarrow a_j)$ if there exists a trace π from s such that $\exists m, n \in [1; |\pi|]$ with $m \leq n$, $a_i \in {}^\bullet\pi^m$, and $a_j \in \pi^{n\bullet}$.

For the sake of self-consistency, we give in proposition 1 an instance implementation of such a predicate. It is a simplified version of a necessary condition for reachability demonstrated in [22]. Essentially, the set of valid objectives Ω is built as follows: initially, it contains all the objectives of the form $a_i \rightsquigarrow a_i$ (that are always valid), and in particular the objectives $s(a) \rightsquigarrow s(a)$ for all $a \in \Sigma$. Then an objective $a_i \rightsquigarrow a_j$ is added to Ω only if there exists a local acyclic path $\eta \in \mathrm{lpaths}(a_i \rightsquigarrow a_j)$ where all the objectives from the initial state s to the pre-conditions of the transitions are already in Ω: if $b_k \in {}^\bullet\eta^n, b \neq a$ for some $n \in [1; |\eta|]$, then the objective $b_0 \rightsquigarrow b_k$ is already in the set, assuming $s(b) = b_0$.

Proposition 1. For all objective $P \in \mathbf{Obj}$, $\mathbf{valid}_s(P) \triangleq P \in \Omega$ where Ω is the least fixed point of the monotonic function $\mathrm{F} : 2^{\mathbf{Obj}} \to 2^{\mathbf{Obj}}$ with

$$\mathrm{F}(\Omega) \triangleq \{a_i \rightsquigarrow a_j \in \mathbf{Obj} \mid \exists \eta \in \mathrm{lpaths}(a_i \rightsquigarrow a_j) : \forall n \in [1; |\eta|], \forall b_k \in {}^\bullet\eta^n \setminus S(a), s(b) \rightsquigarrow b_k \in \Omega\} \ .$$

Applied to the AN of figure 1 with $s = \langle a_0, b_0, c_0, d_0\rangle$, because $\mathbf{valid}_s(b_0 \rightsquigarrow b_0)$ and $a_0 \xrightarrow{b_0} a_1 \in \mathrm{lpaths}(a_0 \rightsquigarrow a_1)$, $\mathbf{valid}_s(a_0 \rightsquigarrow a_1)$ is true. Moreover, because $c_0 \xrightarrow{a_1} c_1 \xrightarrow{b_0} c_2 \in \mathrm{lpaths}(c_0 \rightsquigarrow c_2)$, $\mathbf{valid}_s(c_0 \rightsquigarrow c_2)$ is true. On the other hand, $\mathbf{valid}_s(d_0 \rightsquigarrow d_1)$ is false.

Note that Proposition 1 is an instance of $\mathbf{valid}_s$ implementation; any other implementation satisfying definition 7 can be used to apply the reduction proposed in this article. In [22], more restrictive over-approximations are proposed.

3.2 Reduction procedure

This section depicts the goal-oriented reduction procedure which aims at identifying transitions that do not take part in any minimal trace from the given initial state to the goal local state $g_\top$. The reduction relies on the local causality analysis to delimit local paths that may be involved in the goal reachability: any local transitions that is not captured by this analysis can be removed from the model without affecting the minimal traces for its occurrence.

3.2.1 Formal characterisation

The reduction procedure (definition 8) consists of collecting the set $\mathcal{B}$ of objectives for which a local acyclic path can be part of a minimal trace for the goal reachability. The enabling condition of these local paths necessarily lead to valid objectives from s. The set of such local paths associated to an objective P is noted $\mathrm{lpaths}_s(P)$. The local transitions corresponding to the objectives in $\mathcal{B}$ are noted $\mathrm{tr}(\mathcal{B})$. To ease notations, and without loss of generality, we assume that any automaton a is in state a_0 in s.

Initially starting with the main objective $g_0 \rightsquigarrow g_\top$ (definition 8(1)), for each objective $a_i \rightsquigarrow a_j \in \mathcal{B}$, the procedure collects objectives for other automata that may be involved in the pre-condition of transitions in local paths for $a_i \rightsquigarrow a_j$ (definition 8(2)). Then, for each transition (l, x) of these local paths, for each individual automaton transition $(b_j, b_k) \in l$, if there exists another objective in $\mathcal{B}$ of the form $b_\star \rightsquigarrow b_i$, the objective $b_k \rightsquigarrow b_i$ is added in $\mathcal{B}$ (definition 8(3)). This later criteria accounts for the possible interleaving and successions of local paths within a same automaton: e.g., $g_\top$ reachability may require reaching b_k and b_i in some (undefined) order: 4 objectives have to be considered: $b_0 \rightsquigarrow b_k$, $b_k \rightsquigarrow b_i$, $b_0 \rightsquigarrow b_i$, and $b_i \rightsquigarrow b_k$.

Definition 8 ($\mathcal{B}$)**.** Given an AN (Σ, S, T), an initial state s where, without loss of generality, $\forall a \in \Sigma$, $s(a) = a_0$, and a local state $g_\top$ with $g \in \Sigma$ and $g_\top \in S(g)$, $\mathcal{B} \subseteq \mathbf{Obj}$ is the smallest set which satisfies the following properties:

1) $g_0 \rightsquigarrow g_\top \in \mathcal{B}$
2) $\forall a_i \rightsquigarrow a_j \in \mathrm{tr}(\mathcal{B}), \forall t \in \mathrm{tr}(\mathrm{lpaths}_s(a_i \rightsquigarrow a_j)),$
   $\forall b_k \in {}^\bullet t \setminus S(a), b_0 \rightsquigarrow b_k \in \mathcal{B}$
3) $\forall P \in \mathcal{B}, \forall (l, x) \in \mathrm{tr}(\mathrm{lpaths}_s(P)),$
   $\forall (b_j, b_k) \in l, \forall b_\star \rightsquigarrow b_i \neq P \in \mathcal{B}, b_k \rightsquigarrow b_i \in \mathcal{B}$

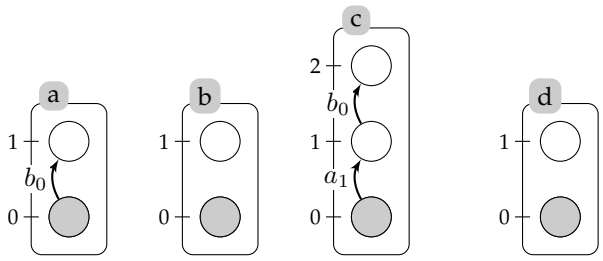

Fig. 2. Reduced automata network from figure 1 for the reachability of c_2 from initial state indicated in grey.

$$\text{with} \quad \mathrm{tr}(\mathcal{B}) \triangleq \bigcup_{P \in \mathcal{B}} \mathrm{tr}(\mathrm{lpaths}_s(P)) \ , \text{ where,}$$

$$\forall a_i \rightsquigarrow a_j \in \mathbf{Obj},$$

$$\mathrm{lpaths}_s(a_i \rightsquigarrow a_j) \triangleq \{\eta \in \mathrm{lpaths}(a_i \rightsquigarrow a_j) \mid \forall n \in [1; |\eta|], \forall b_k \in {}^\bullet\eta^n \setminus S(a), \mathbf{valid}_s(b_0 \rightsquigarrow b_k)\}$$

Theorem 1 states that any trace which is minimal for the reachability of $g_\top$ from initial state s is composed only of transitions in $\mathrm{tr}(\mathcal{B})$. The proof is detailed in appendix B. It results that the AN $(\Sigma, S, \mathrm{tr}(\mathcal{B}))$ contains fewer transitions but preserves all the minimal traces for the reachability of the goal.

Theorem 1. *For each* minimal *trace* π *reaching* $g_\top$ *from* s, $\mathrm{tr}(\pi) \subseteq \mathrm{tr}(\mathcal{B})$.

Example 3. By applying proposition 1 on the AN of figure 1, we obtain the following set of valid objectives:

$$\begin{aligned}\Omega = \{&a_0 \rightsquigarrow a_0, a_1 \rightsquigarrow a_1, b_0 \rightsquigarrow b_0, b_1 \rightsquigarrow b_1,\\ &c_0 \rightsquigarrow c_0, c_1 \rightsquigarrow c_1, c_2 \rightsquigarrow c_2, d_0 \rightsquigarrow d_0, d_1 \rightsquigarrow d_1,\\ &a_0 \rightsquigarrow a_1, b_0 \rightsquigarrow b_1, c_0 \rightsquigarrow c_1, c_0 \rightsquigarrow c_2,\ c_1 \rightsquigarrow c_0\ \}\\ = \mathbf{Obj} \setminus \{&c_2 \rightsquigarrow c_0, c_2 \rightsquigarrow c_1, d_0 \rightsquigarrow d_1, d_1 \rightsquigarrow d_0\}\end{aligned}$$

Let us consider the initial state $\langle a_0, b_0, c_0, d_0\rangle$ and the goal c_2. Following definition 8(1), we initially set $\mathcal{B} = \{c_0 \rightsquigarrow c_2\}$. Note that, because $\mathbf{valid}_s(d_0 \rightsquigarrow d_1)$ is false, we have $\mathrm{lpaths}_s(c_0 \rightsquigarrow c_2) = \{c_0 \xrightarrow{\{a_1\}} c_1 :: c_1 \xrightarrow{\{b_0\}} c_2\}$. Following definition 8(2), the objectives $a_0 \rightsquigarrow a_1$ and $b_0 \rightsquigarrow b_0$ are added to $\mathcal{B}$, and no more objectives are considered. Note that the rule definition 8(3) never occurs in this simple example.

The resulting reduced AN for c_2 reachability is given in figure 2.

3.2.2 Algorithmic aspects

Although the number of objectives is quadratic, the mathematical definition of the reduction procedure in proposition 1 and definition 8 does not address its efficient implementation, notably to avoid useless or redundant computations over objectives and their local paths.

Listing 1 details such a possible algorithm to compute the goal-oriented reduction by implementing jointly the over-approximation of valid objectives (proposition 1) and of the transitions of minimal traces (definition 8).

To avoid the un-necessary analysis of objectives which are not involved for the goal reachability, instead of computing the full set Ω of proposition 1, the validity of

Listing 1. Algorithm for goal-oriented reduction

```
1  G = digraph()
2
3  def compute_valid_nodes(P : objective):
4      new_nodes = set()
5      valid_queue = fifo()
6
7      def set_node_validity(n, valid):
8          G[n].valid = valid
9          valid_queue.push(n)
10
11     def build_obj_deps(a_i ⇝ a_j):
12         G[a_i ⇝ a_j] = node()
13         new_nodes.add(a_i ⇝ a_j)
14         for η in lpaths(a_i ⇝ a_j):
15             G[η] = node()
16             new_nodes.add(η)
17             G[a_i ⇝ a_j].children.add(η)
18             for t in η:
19                 for b_k in •t \ S(a):
20                     if b_0 ⇝ b_k not in G:
21                         build_obj_deps(b_0 ⇝ b_k)
22                     elif G[b_0 ⇝ b_k].valid is defined:
23                         valid_queue.push(b_0 ⇝ b_k)
24                     G[η].children.add(b_0 ⇝ b_k)
25             if G[η].children is empty:
26                 set_node_validity(η, True)
27         if G[a_i ⇝ a_j].children is empty:
28             set_node_validity(a_i ⇝ a_j, False)
29
30     def update_valid_parent(n, child):
31         G[n].defined.add(child)
32         if n is an objective:
33             if G[child].valid:
34                 set_node_valid(n, True)
35             elif G[n].children == G[n].defined:
36                 set_node_valid(n, False)
37         else: # n is a local path
38             if not G[child].valid:
39                 set_node_valid(n, False)
40             elif G[n].children == G[n].defined:
41                 set_node_valid(n, True)
42
43     build_obj_deps(P)
44     while valid_queue is not empty:
45         m = valid_queue.pop()
46         for n in G[m].parents:
47             if G[n].valid is not defined:
48                 update_valid_parent(n, m)
49         new_nodes.remove(m)
50     for n in new_nodes:
51         G[n].valid = False
52
53 def reduce():
54     queue = fifo()
55     targets = {a: set() for a ∈ Σ}
56
57     def register_obj(a_i ⇝ a_j):
58         if a_i ⇝ a_j not in G:
59             compute_valid_nodes(a_i ⇝ a_j)
60         my_targets = {a: set() for a ∈ Σ}
61         for η in lpaths(a_i ⇝ a_j):
62             if G[η].valid:
63                 for (l, x) in η:
64                     for b_k in •(l, x) \ S(Σ(P)):
65                         queue.push(b_0 ⇝ b_k)
66                     for (b_j, b_k) in l:
67                         if k not in targets[b]:
68                             for i in targets[b]:
69                                 queue.push(b_i ⇝ b_k)
70                                 queue.push(b_k ⇝ b_i)
71                         my_targets[b].add(k)
72         targets.merge(my_targets)
73
74     done = set()
75     queue.push(g_0 ⇝ g_⊤)
76     while queue is not empty:
77         P = queue.pop()
78         if P not in done:
79             register_obj(P)
80             done.add(P)
81     keep_trs = set()
82     for local path η in G.nodes:
83         if G[η].valid:
84             keep_trs.update(set(η))
85     return keep_trs
```

objectives can be assessed only when they are considered for $\mathcal{B}$ (compute_valid_nodes). We use a digraph where nodes are objectives and local paths, related by edges: each $\eta \in \text{lpaths}(a_i \rightsquigarrow a_j)$ is a child of objective $a_i \rightsquigarrow a_j$ (lines 14-17), and $b_0 \rightsquigarrow b_k$ is a child of η if $b \neq a$ and if there exists a transition $(l, x) \in \eta$ where $b_k \in {}^\bullet(l, x)$ (lines 18-24).

Once all the dependencies of the objectives are referenced in the graph (build_obj_deps), the validity of nodes are computed as follows: an objective is *valid* if and only if at least one of its local path children is valid; a local path is *valid* if and only if all its objective children are valid. Therefore, the validity of a node can be assessed only when either the validity of all the children have been assessed; or as soon as one child of an objective is declared valid, or one child of a local path is declared as not valid (update_valid_parent). The computation starts from the leafs of the graph (lines 25-28) and continues to their parents (lines 44-49) creating a topological ordered traversal of the digraph. Remark that the digraph can contain cycles. Two cases arise: either the validity of a node of a connected component can be computed (when one of its children is outside the connected component has been computed), in which case its children in the connected component have no impact on its value; or none of the nodes of the connected component can be computed: this case reflects un-resolvable circular dependencies between objectives and local paths, hence these nodes are invalid (lines 50-51).

At the same time, the graph accounts for the objectives and local paths of $\mathcal{B}$: starting from the goal objective (line 75, definition 8(1)), objectives to be added to $\mathcal{B}$ are iteratively registered in the graph (lines 76-80) with register_obj. Once the validity of the candidate objective is assessed (line 59), rules of definition 8(2) and definition 8(3) are applied, respectively in line 65 and lines 68-70.

Finally, the transitions of the reduced model correspond to the transitions of the valid local paths referenced in the digraph (lines 81-85).

3.2.3 Complexity

Recall that there is quadratic number of objectives ($|\mathbf{Obj}| = \sum_{a \in \Sigma} |S(a)|^2$); for any objective $a_i \rightsquigarrow a_j$, the number of local path in $\text{lpaths}(a_i \rightsquigarrow a_j)$ is polynomial with the number of local transitions in a and exponential in the number of local states in a minus 1, i.e., $|S(a)| - 1$; moreover, a local path is of length at most $|S(a)|$.

Therefore, the size of graph G computed in Listing 1 is polynomial with the number of local transitions $|T|$ and exponential with the maximum number $k - 1$ of local states within an automaton, where $k = \max_{a \in \Sigma} |S(a)|$.

The function compute_valid_node visit each node and edge twice: once for the construction (build_obj_deps) and once for

the computation of nodes validity. The function register_obj is again linear with the size of G.

Overall, the goal-oriented reduction has a time complexity polynomial with the total number of automata and local transitions, and exponential with the maximum number $k-1$ of local states within an automaton. Therefore, assuming $k \ll |\Sigma|$, the goal-oriented reduction offers a very low complexity, especially with regard to a full exploration of the $k^{|\Sigma|}$ states. Moreover, we remark that in the case of binary automata ($k = 2$), as when encoding Boolean networks, the complexity is polynomial.

3.3 Comparison with other reductions

The *relevant subnet computation* [21] identifies places and transitions of Petri nets which can be removed while preserving a subset of LTL properties. As our automata network framework is close to safe Petri nets (appendix A), the subnet computation can be directly applied to AN models. Nevertheless, the subnet computation aims at preserving LTL properties which involve non-minimal traces; and hence preserves more properties than our goal-oriented reduction. In particular, some traces with cycles are preserved by the subnet computation, as they are relevant to some LTL properties. Applied to figure 1 for c_2 reachability, the relevant subnet would remove only the transition $c_0 \xrightarrow{d_1} c_2$.

Another closely related reduction is the so-called *cone of influence* reduction [20] which identifies variables (automata) which are not involved in given LTL/CTL property. The principle is very similar to the relevant subset network computation, and is implemented in the model-checker NuSMV. Next section gives benchmarks on the performance of NuSMV before and after the goal-oriented reduction, and using in both case the cone of influence reduction. It shows that the goal-oriented reduction achieve a much more drastic reduction.

4 Application to Signalling and Gene Regulatory Networks

In this section, we detail the application of automata networks for the analysis of transient reachability in biological regulatory networks, typically signalling and gene networks. First, as qualitative models of these networks are usually defined with Boolean and multi-valued networks, we show how these latter can be encoded exactly in automata networks. Then, we address the biological interpretation of goal reachability and cut sets properties, and mention available implementation of the goal-oriented reduction.

4.1 Encoding of Boolean and multi-valued networks

Most of qualitative models of biological regulatory networks available in literature or databases (such as biomodels, cell-collection, GINsim) are specified as Boolean networks and multi-valued networks. One of their fundamental difference with Automata Networks (ANs) is that Boolean/multi-valued networks define (deterministic) functions for each node of the network instead of transitions between node values. Nevertheless, we detail in this section that both asynchronous Boolean and multi-valued networks dynamics can be encoded *equivalently* as ANs (we demonstrate a bisimulation relation). As we will discuss at the end of this sub-subsection, the converse is not true.

4.1.1 Definitions

There exists several definitions of mutli-valued networks in the literature (e.g., [31], [32], [33]) which are often parametrised by a so-called influence graph with regulation thresholds. In this section, we use a general definition of such networks, using simple discrete maps. It encompasses above mentioned frameworks, and straightforwardly boils down to classical Boolean networks [34], [35] when the discrete domain of every node is binary.

Definition 9 (Multi-valued Network)**.** A multi-valued network of dimension n is defined by a couple $(\mathbb{D}, F)$ where $\mathbb{D} = \mathbb{D}^1 \times \cdots \times \mathbb{D}^n$, with, $\forall i \in [0;n]$, $D^i = [0; m_i]$ is the domain of each node; and where $F = \langle f_1, \ldots, f_n \rangle$, with $\forall i \in [0;n]$, $f_i : \mathbb{D} \to \mathbb{D}^i$.

Definition 10 (Semantics of Multi-valued Network)**.** Given a multi-valued network $(\mathbb{D}, F)$ of dimension n, and given a state $v \in \mathbb{D}$, the node i can change of value only if $v_i \neq f_i(v)$, in which case its new value is given by

$$\mathrm{next}^i_F(v) \triangleq \begin{cases} v_i + 1 & \text{if } f_i(v) > v_i \\ v_i - 1 & \text{if } f_i(v) < v_i \\ v_i & \text{if } f_i(v) = v_i \end{cases}$$

The *asynchronous transition relation* $\to^{\mathrm{async}}_F \subseteq \mathbb{D} \times \mathbb{D}$ is such that, $\forall v, v' \in \mathbb{D}$,

$$v \to^{\mathrm{async}}_F v' \triangleq v \neq v' \wedge \exists i \in [0;n] : v'_i = \mathrm{next}^i_F(v) \\ \wedge \forall j \in [0;n], j \neq i, v'_j = v_j$$

The *asynchronous generalized update transition relation* $\to^{\mathrm{gen}}_F \subseteq \mathbb{D} \times \mathbb{D}$ is such that, $\forall v, v' \in \mathbb{D}$,

$$v \to^{\mathrm{gen}}_F v' \triangleq v \neq v \wedge \forall i \in [0;n], \\ v'_i = \mathrm{next}^i_F(v) \vee v'_i = v_i$$

4.1.2 Encoding in automata networks

Given a multi-valued network $(\mathbb{D}, F)$ of dimension n, we associate to each node $i \in [0;n]$ an automaton a^i with local states corresponding to $\mathbb{D}^i$. The transitions are defined such that they encode $\mathrm{next}^i_F(v)$ when $\mathrm{next}^i_F(v) \neq v_i$, i.e., there is a transition from a^i_j to a^i_{j+1} (resp. a^i_{j-1}) when $f_i(v) > v_i$ (resp. $f_i(v) < v_i$).

The conditions of transitions correspond to the solutions of the Boolean satisfiability formula $[v_i = j \wedge f_i(v) > j]$ (resp. $[v_i = j \wedge f_i(v) < j]$), with variables v_k, $k \in [0;n]$, and where $\square_i < j \equiv \square_i = 0 \vee \cdots \vee \square_i = j-1$, and $\square_i > j \equiv \square_i = j+1 \vee \cdots \vee \square_i = m_i$, assuming $\mathbb{D}^i = [0; m_i]$.

Transitions being of the form $a^i_j \xrightarrow{x} a^i_{j\pm1}$, where the condition x is a conjunction of local states, each transition corresponds to a clause of the Disjunctive Normal Form (DNF, disjunction of conjunctions) of $[v_i = j \wedge f_i(v) \gtrless j]$, we refer to as $\mathrm{DNF}[v_i = j \wedge f_i(v) \gtrless j] \subseteq 2^{\{[v_k = q] | k \in [0;n], q \in \mathbb{D}^k\}}$.

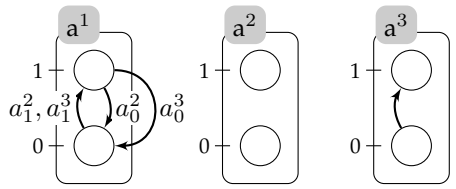

Fig. 3. Automata network encoding the Boolean network of Example 4

We define the automata network (Σ, S, T) encoding the multi-valued network $(\mathbb{D}, F)$ of dimension n, with:

$$\Sigma = \{a^i \mid i \in [0;n]\} \qquad \forall i \in [0;n], S(a^i) = \{a^i_j \mid j \in \mathbb{D}^i\}$$

$$\begin{aligned} T = \{a^i_j \xrightarrow{x^i(C)} a^i_{j+1} \mid i \in [0;n] \wedge j \in \mathbb{D}^i \\ \wedge C \in \mathrm{DNF}[v_i = j \wedge f_i(v) > j]\} \\ \cup \{a^i_j \xrightarrow{x^i(C)} a^i_{j-1} \mid i \in [0;n] \wedge j \in \mathbb{D}^i \\ \wedge C \in \mathrm{DNF}[v_i = j \wedge f_i(v) < j]\} \end{aligned}$$

where $x^i(C) \triangleq \{a^k_q \mid k \neq i \wedge [v_k = q] \in C\}$.

The computation of DNF can rely on the method of prime implicants (Quine-McCluskey algorithm [36]) to obtain minimal sets of transitions.

Example 4. Let us consider the Boolean network of dimension 3 with $\mathbb{D}^1 = \mathbb{D}^2 = \mathbb{D}^3 = \{0,1\}$, and

$$f_1(v) = [v_2 = 1 \wedge v_3 = 1] \quad f_2(v) = [v_2] \quad f_3(v) = [1]$$

From the DNF computation (detailed in figure 4), we obtain the following automata network, represented in figure 3:

$$\Sigma = \{a^1; a^2; a^3\}$$
$$S(a^1) = \{a^1_0; a^1_1\} \quad S(a^2) = \{a^2_0; a^2_1\} \quad S(a^3) = \{a^3_0; a^3_1\}$$
$$T = \{a^1_0 \xrightarrow{\{a^2_1, a^3_1\}} a^1_1; a^1_1 \xrightarrow{\{a^2_0\}} a^1_0; a^1_1 \xrightarrow{\{a^3_0\}} a^1_0; a^3_0 \xrightarrow{\emptyset} a^3_1\}$$

4.1.3 Bisimulation relation

Given an AN (Σ, S, T), we define the asynchronous ($\to^{\mathrm{async}}_T \subseteq S \times S$) and generalised asynchronous ($\to^{\mathrm{gen}}_T \subseteq S \times S$) transition relations following step definition (definition 2): the asynchronous corresponds to steps composed of only one transition, whereas the generalised asynchronous corresponds to any non-empty steps.

$$s \to^{\mathrm{async}}_T s' \triangleq \exists t \in T, s \cdot \{t\} = s'$$
$$s \to^{\mathrm{gen}}_T s' \triangleq \exists \tau \subseteq T, \tau \neq \emptyset, s \cdot \tau = s'$$

Let us consider the bijection relation $\sim \subseteq \times S$ defined as $v \sim \langle a^1_{v_1}, \ldots, a^n_{v_n} \rangle$. It derives that $\sim$ is a bisimulation relation for both asynchronous and generalised asynchronous semantics: if $v \sim s$, then

$$\begin{aligned} &(v \to^{\mathrm{async}}_F v' \Rightarrow \exists s' \in S : s \to^{\mathrm{async}}_T s' \wedge v' \sim s') \\ \wedge &(s \to^{\mathrm{async}}_T s' \Rightarrow \exists v' \in \mathbb{D} : v \to^{\mathrm{async}}_F v' \wedge v' \sim s') \\ \wedge &(v \to^{\mathrm{gen}}_F v' \Rightarrow \exists s' \in S : s \to^{\mathrm{gen}}_T s' \wedge v' \sim s') \\ \wedge &(s \to^{\mathrm{gen}}_T s' \Rightarrow \exists v' \in \mathbb{D} : v \to^{\mathrm{gen}}_F v' \wedge v' \sim s') \end{aligned}$$

Indeed, for any $v \in \mathbb{D}$ and the $\sim$-corresponding s, $\forall i \in [0;n]$, $(\mathrm{next}^i_F(v) = j' \wedge v_i \neq j')$ if and only if $\exists a^i_{v_i} \xrightarrow{x} a^i_{j'} \in T$ where $x \subseteq s$.

4.1.4 Discussion

It results that any asynchronous/generally asynchronous Boolean and multi-valued networks can be encoded equivalently as an automata network.

Remark that the converse is not true: let us consider the automata network (Σ, S, T) with $\Sigma = \{a\}$, $S(a) = \{a_0, a_1, a_2\}$, and $T = \{a_1 \xrightarrow{\emptyset} a_0; a_1 \xrightarrow{\emptyset} a_2\}$. This actually encodes a non-deterministic discrete function which is equal both to 0 and 2 when $a = 1$. In general, automata networks can encode non-deterministic functions, which is not directly possible with usual Boolean and multi-valued networks. See [37] for a thorough comparison of function-centered and transition-centered systems.

Furthermore, remark that coupled transitions such as $a_0, b_0 \xrightarrow{x} a_1, b_1$ would require adhoc semantics specifications for Boolean and multi-valued networks.

4.2 Goal reachability and cut sets

Applied directly to automata networks modelling a biological network, a goal is typically the activation/inactivation of a particular gene/transcription factor/kinase/etc. Given an initial state for each node of the network, the goal is reachable if there exists a sequence of steps leading to a state where the goal is present. For instance, in a signalling network, the initial state usually corresponds to a setting of receptor states with the internal nodes being initially inactive; and the goal corresponds to the activation of downstream transcription factor: the goal reachability means that, in the specified settings, it is possible to see an activation of the given transcription factor.

A minimal trace corresponds to a sequence of steps that present no cycle (the same global state is never visited twice), and in which each node activation/inhibition is causally related to the goal reachability.

Note that the reachability properties we consider are *transient*: there is no guarantee that the goal is still present in the long run.

Beyond the local state goal

Although the formal specification of the goal is always the local state of a single automaton, it is worth noticing that our framework allow to consider much more complex properties, such as (sub-)state reachability, sequence of (sub)-states, and disjunction of these properties.

Indeed, such properties can be encoded by on extra automaton g and whose transitions and their condition express the desired behaviours; the final local state for each of them being the goal local state $g_\top$.

For instance, the property "reach a state where both a_1 and b_1 are present, then reach c_1 or d_1" can be encoded with the automaton g having 3 local states $g_0, g_1, g_\top$, and the transitions $g_0 \xrightarrow{\{a_1,b_1\}} g_1; g_1 \xrightarrow{\{c_1\}} g_\top; g_1 \xrightarrow{\{d_1\}} g_\top$. Similarly, the reachability of global state s can be expressed with a single extra transition $g_0 \xrightarrow{\{s(a) \mid a \in \Sigma, a \neq g\}} g_\top$.

Any trace reaching the goal necessarily verifies the desired dynamical property; and any trace verifying the property can always trigger the goal reachability. Consequently, all minimal traces verifying the property are preserved by the goal-oriented reduction.

$a_0^1 \to a_1^1 : \mathrm{DNF}[v_1 = 0 \wedge f_1(v) = 1] \equiv \mathrm{DNF}[v_1 = 0 \wedge v_2 = 1 \wedge v_3 = 1] = \{\{[v_1 = 0], [v_2 = 1], [v_3 = 1]\}\}$
$a_1^1 \to a_0^1 : \mathrm{DNF}[v_1 = 1 \wedge f_1(v) = 0] \equiv \mathrm{DNF}[v_1 = 1 \wedge (v_2 = 0 \vee v_3 = 0)] = \{\{[v_1 = 1], [v_2 = 0]\}, \{[v_1 = 1, v_3 = 0]\}\}$
$a_0^2 \to a_1^2 : \mathrm{DNF}[v_2 = 0 \wedge f_2(v) = 1] \equiv \mathrm{DNF}[v_2 = 0 \wedge v_2 = 1] = \emptyset$
$a_1^2 \to a_0^2 : \mathrm{DNF}[v_2 = 1 \wedge f_2(v) = 0] \equiv \mathrm{DNF}[v_2 = 1 \wedge v_2 = 0] = \emptyset$
$a_0^3 \to a_1^3 : \mathrm{DNF}[v_3 = 0 \wedge f_3(v) = 1] \equiv \mathrm{DNF}[v_3 = 0 \wedge 1 = 1] = \{\{[v_3 = 0]\}\}$
$a_1^3 \to a_0^3 : \mathrm{DNF}[v_3 = 1 \wedge f_3(v) = 0] \equiv \mathrm{DNF}[v_3 = 1 \wedge 1 = 0] = \emptyset$

Fig. 4. Detail of transition conditions computation for Example 4.

Cut sets for goal reachability

Cut sets are sets of local states such that each trace reaching the goal includes a transition involving one of these local states. For instance, we say that $\{a_1, b_1\}$ is a cut set for $g_\top$ reachability if any trace reaching $g_\top$ includes a transitions having a_1 *or* b_1 in its pre-condition. Hence, the disabling of all the transitions having pre-condition intersecting with the cut set will remove all the traces leading to the goal. Therefore, cut sets predicts mutation to control the biological network, which could be implemented by the knock-out/in of the corresponding species. Cut sets have been studied in the scope of biological networks in [6], [23], and are close to intervention sets [4]. Whereas intervention sets determine mutations which ensure the inevitability of a specified steady state, cut sets determine mutations which disable any trace leading to a goal state, without any steady state assumption.

Remark that it is sufficient to break only the minimal the traces to goal in order to break all the traces to it. On the other hand, verifying if a set of local state is a valid cut set requires to reason on *all* the minimal traces. Therefore, the preservation of all the minimal traces by the goal-oriented reduction (theorem 1) is crucial to ensure the equivalence of the verification on the reduced model.

4.3 Implementation

The software PINT, available at http://loicpauleve.name/pint, implements the static analysis of automata networks for transient reachability properties.

Besides command line utilities, it comes with a PYTHON interface which allows a seamless manipulation of automata network, from the automatic import of Boolean/multi-valued models (notably in GINsim, SBML-qual, or any format supported by BioLQM [38]), to the model reduction and analysis.

A typical usage for model reduction for further analysis with NuSMV model-checker is the following:

```
>>> import pypint
>>> an = pypint.load("http://ginsim.org..model.zginml")
>>> an.initial_state.update(EGFr=1,TGFr=1)
>>> red = an.reduce_for_goal(pRB=1)
>>> red.save_as("model.smv")
# or direct invocation of model-checker
# (reduction is done automatically beforehand)
>>> an.reachability(pRB=1, tool="nusmv")
True
```

5 EXPERIMENTS

We experimented the goal-oriented reduction on several biological networks and quantify the shrinkage of the reachable state space. Then, we illustrate potential applications with the verification of simple reachability, and of cut sets. In both cases, the reduction drastically increases the tractability of those applications.

5.1 Results on model reduction

We conducted experiments on Automata Networks (ANs) that model dynamics of biological networks. For different initial states, and for different reachability goals, we compared the number of local transitions in the AN specifications ($|T|$), the number of reachable states, and the size of the so-called complete finite prefix of the unfolding of the net [39]. This latter structure is a finite partial order representation of all the possible traces, which is well studied in concurrency theory. It aims at offering a compact representation of the reachable state spaces by exploiting the concurrency between transitions: if t_1 and t_2 are playable in a given state and are not in conflict (notably when ${}^\bullet t_1 \cap {}^\bullet t_2 = \emptyset$), a standard approach would consider 4 global transitions (t_1 then t_2, and t_2 then t_1), whereas a partial order structure would simply declare t_1 and t_2 as concurrent, imposing no ordering between them. Hence, unfoldings drop part of the combinatorial explosion of the state space due to the interleaving of concurrent transitions.

The selected networks are models of signalling pathways and gene regulatory networks: two Boolean models of Epidermal Growth Factor receptors (EGF-r) [6], [7], one Boolean model of tumor cell invasion (Wnt) [2], two Boolean models of T-Cell receptor (TCell-r) [4], [5], one Boolean model of Mitogen-Activated Protein Kinase network (MAPK) [3], one multi-valued model of fate determination in the Vulval Precursor Cells (VPC) in C. elegans [8], one Boolean model of T-Cell differentiation (TCell-d) [1], and one Boolean models of cell cycle regulation (RBE2F) [40]. The ANs result from the encoding described in section 4.1. For each of these models, we selected initial states and nodes for which the activation will be the reachability goal (see supplementary material for scripts and data). Typically, the initial states correspond to various input signal combinations in the case of signalling cascades, or to pluripotent states for gene networks; and goals correspond to transcription factors or genes of importance for the model (output nodes for signalling cascades, key regulators for gene networks).

Table 1 sums up the results before and after the goal-oriented reduction. The number of reachable states is computed with ITS-REACH [41] using a symbolic representation, and the size of the complete finite prefix (number of instances of transitions) is computed with MOLE [42]. In each case, the reduction step took less than 0.1s, thanks to its very low complexity when applied to logical networks.

					Verification of goal reachability				Verification of goal cut sets			
Model	Goal	$\|T\|$	# states	$\|$unf$\|$	NuSMV		its-reach		NuSMV		its-ctl	
EGF-r (20)	pRB_1	68	4,200	1,749	0.2s	10Mb	0.2s	7Mb	0.1s	9Mb	0.7s	51Mb
		43	**722**	**336**	**0.1s**	**8Mb**	**0.1s**	**5Mb**	**0.1s**	**8Mb**	**0.2s**	**24Mb**
Wnt (32)	$Migration_1$	197	7,260,160	KO	30s	48Mb	0.3s	18Mb	44s	55Mb	105s	2.1Gb
		117	**241,060**	**217,850**	**0.9s**	**32Mb**	**0.5s**	**17Mb**	**9.1s**	**27Mb**	**16s**	**720Mb**
TCell-r (40)	$AP1_1$	90	$\approx 1.2 \cdot 10^{11}$	KO	KO		1.1s	52Mb	KO		492s	10Gb
		46	**158,400**	**14,071**	**3.8s**	**36Mb**	**0.6s**	**15Mb**	**2.4s**	**34Mb**	**11s**	**319Mb**
MAPK (53) initial state 1	$Proliferation_1$	173	$\approx 3.8 \cdot 10^{12}$	KO	KO		0.9s	60Mb	KO		KO	
		113	$\approx \mathbf{4.5 \cdot 10^{10}}$	**KO**	**KO**		**2s**	**48Mb**	**KO**		**KO**	
MAPK (53) initial state 2	$Apoptosis_1$	173	8,126,465	KO	63s	83Mb	0.2s	15Mb	34s	66Mb	48s	1.9Gb
		69	**269,825**	**155,327**	**1.5s**	**36Mb**	**0.4s**	**18Mb**	**0.3s**	**23Mb**	**6.7s**	**500Mb**
VPC (88)	$LIN39_1$	332	KO	KO	KO		1s	50Mb	KO		KO	
		219	$\mathbf{1.8 \cdot 10^9}$	**43,302**	**236s**	**156Mb**	**0.8s**	**21Mb**	**163s**	**155Mb**	**KO**	
TCell-r (94)	$ap1_1$	217	KO	KO	KO		KO		KO		KO	
		42	**54.921**	**1,017**	**0.4**	**23Mb**	**0.26s**	**14Mb**	**0.2s**	**22Mb**	**3s**	**181Mb**
TCell-d (101) initial state 1	$Th2_1$	384	$\approx 2.7 \cdot 10^8$	257	3s	40Mb	0.5s	24Mb				
		0	**1**	**1**								
TCell-d (101) initial state 2	$BCL6_1$	384	KO	KO	KO		0.5s	23Mb	KO		KO	
		161	**75,947,684**	**KO**	**474s**	**260Mb**	**0.3s**	**19Mb**	**600s**	**360Mb**	**KO**	
EGF-r (104) initial state 1	$ap1_1$	378	9,437,184	47,425	7s	35Mb	0.6s	23Mb				
		0	**1**	**1**								
EGF-r (104) initial state 2	$ap1_1$	378	$\approx 2.7 \cdot 10^{16}$	KO	KO		1.36s	60Mb	KO		KO	
		69	**62,914,560**	**KO**	**11s**	**33Mb**	**0.3s**	**17Mb**	**13s**	**33Mb**	**21s**	**875Mb**
RBE2F (370)	$a858_1$	742	KO	KO	KO		KO		KO		KO	
		56	**2,350,494**	**28,856**	**5s**	**377Mb**	**5s**	**170Mb**	**6s**	**29Mb**	**179s**	**1.8Gb**

TABLE 1
Comparisons before (normal font) and **after** (bold font) the goal-oriented AN reduction and cut sets verification. Each model is identified by the system, the number of automata (within parentheses), an initial state, and the reachability goal. $|T|$ is the number of local transitions in the AN specification; "#states" is the number of reachable global states from the initial state; "|unf|" is the size of the complete finite prefix of the unfolding. "KO" indicates an execution running out of time (30 minutes) or memory. When applied to goal reachability, we show the total execution time and memory used by the tools `NuSMV`, `its-reach`, and `its-ctl`. Computation times where obtained on an Intel® Core™ i7 3.4GHz CPU with 16GB RAM. *For each case, the reduction procedure took less than 0.1s.*

There is a substantial shrinkage of the dynamics for the reduced models, which can turn out to be drastic for large models. In some cases, the model is too large to compute the state space without reduction. For some large models, the unfolding is too large to be computed, whereas it can provide a very compact representation compared to the state space for large networks exhibiting a high degree of concurrency (e.g., TCell-d, RBE2F). In the case of first profile of TCell-d and EGF-r (104) the reduction removed all the transitions, resulting in an empty model. Such a behaviour can occur when the local causality analysis statically detect that the reachability goal is impossible, i.e., the necessary condition of section 3.1 is not satisfied. On the other hand, a non-empty reduced model does not guarantee the goal reachability.

Impact of statically-proven impossible objectives

The goal-oriented reduction relies on two intertwined analyses of the local causality in ANs: (1) the computation of potentially involved objectives (section 3.2) and (2) the filtering of objectives that can be proven impossible (section 3.1). The second part can be considered optional: one could simply define the predicate $\mathbf{valid}_s$ to be always true. In order to appreciate the impact of this second part, we show in table 2 the intermediary results of model reduction without the filtering of impossible objectives. As we can see, for some models it has no effect on the reduction, for some others the filtering part is necessary to obtain substantial reduction of the state space (e.g., MAPK, TCell-r (94), TCell-d).

5.2 Example of application: goal reachability

In order to illustrate practical applications of the goal-oriented model reduction, we first systematically applied model-checking for the goal reachability on the initial and reduced model (table 1).

We compared two different softwares: NuSMV [43] which combines Binary Decision Diagrams and SAT approaches for synchronous systems, and ITS-REACH which implements efficient decision diagram data structures [44]. In both cases, the transition systems specified as input of these tools is an exact encoding of the asynchronous semantics of the automata networks, where steps (definition 2) are always composed of only one transition. For NuSMV, the reachability property is specified with CTL [45] ("EF $g_\top$", $g_\top$ being the goal local state, and EF the *exists eventually* CTL operator). It is worth noting that NuSMV implements the *cone of influence* reduction [20], although our experiments show it does not bring a noticeable performance improvement for the selected models. ITS-REACH is optimized for checking if a state belongs to the reachable state space, and cannot perform CTL checking.

Experiments show a remarkable gain in tractability for the model-checking of reduced networks. For large cases, we observe that the dynamics can be tractable only after model reduction (e.g., TCell-r (94), RBE2F (370)). ITS-REACH is significantly more efficient than NuSMV because it is tailored for simple reachability checking, whereas NuSMV handles much more general properties.

Because the goal-reduction preserves all the minimal

Model	# tr	# states	|unf|
EGF-r (20)	68	4,200	1,749
	43	*722*	*336*
	43	**722**	**336**
Wnt (32)	197	7,260,160	KO
	134	*241,060*	*217,850*
	117	**241,060**	**217,850**
TCell-r (40)	90	$\approx 1.2 \cdot 10^{11}$	KO
	46	*158,400*	*14,071*
	46	**158,400**	**14,071**
MAPK (53) initial state 1	173	$\approx 3.8 \cdot 10^{12}$	KO
	147	$\approx 9 \cdot 10^{10}$	*KO*
	113	$\approx \mathbf{4.5 \cdot 10^{10}}$	**KO**
MAPK (53) initial state 2	173	8,126,465	KO
	148	*1,523,713*	*KO*
	69	**269,825**	**155,327**
VPC (88)	332	KO	KO
	278	$\approx 2.9 \cdot 10^{12}$	*185,006*
	219	$\mathbf{1.8 \cdot 10^{9}}$	**43,302**
TCell-r (94)	217	KO	KO
	112	*KO*	*KO*
	42	**54.921**	**1,017**
TCell-d (101) initial state 1	384	$\approx 2.7 \cdot 10^{8}$	257
	275	$\approx 1.1 \cdot 10^{8}$	*159*
	0	**1**	**1**
TCell-d (101) initial state 2	384	KO	KO
	253	$\approx 2.4 \cdot 10^{12}$	*KO*
	161	**75,947,684**	**KO**
EGF-r (104) initial state 1	378	9,437,184	47,425
	120	*12,288*	*1,711*
	0	**1**	**1**
EGF-r (104) initial state 2	378	$\approx 2.7 \cdot 10^{16}$	KO
	124	$\approx 2 \cdot 10^{9}$	*KO*
	69	**62,914,560**	**KO**
RBE2F (370)	742	KO	KO
	56	*2,350,494*	*28,856*
	56	**2,350,494**	**28,856**

TABLE 2
Comparison of AN size (# tr), number of reachable states (# states), and complete prefix size (|unf|) before (normal font) the goal-oriented reduction with the goal-oriented reduction *without the filtering of impossible objective* (italic) and with the **full** reduction (bold font)

traces for the goal reachability, it preserves the goal reachability: the results of the model-checking is equivalent in the initial and reduced model.

5.3 Example of application: cut set verification

The above application to simple reachability does not requires the preservation of *all* the minimal traces. Here, we apply the goal-oriented reduction to the cut sets for reachability, where the *completeness of minimal traces is crucial* (section 4.2)

We focus on verifying if a given set of local states is a cut set for the goal reachability. In the scope of this experiment, we consider cut sets that are disjoint with the initial state. The cut set property can be expressed with CTL as follows: $\{a_1, b_1\}$ is a cut set for $g_\top$ reachability if the model satisfies the CTL property `not E [ (not` a_1 `and not` b_1`) U` $g_\top$ `]` (`U` being the *until* operator). The property states that there exists no trace where none of the local state of the cut set is reached prior to the goal. It is therefore required that *all* the minimal traces to the goal reachability are present in the model: if one is missing, a set of local states could be validated as cut set whereas it may not be involved in the missed trace.

Table 1 compares the model-checking of cut sets properties using NuSMV and ITS-CTL [41] on a range of the biological networks used in the previous sections and on cut sets computed beforehand with PINT or arbitrarily designed. Because the goal-oriented reduction preserves all the minimal traces to the goal, the results are equivalent in the reduced models. As for the simple reachability, the goal-oriented reduction drastically improves the tractability of large models.

6 Discussion

This paper establishes a reduction of automata networks parametrized by a goal, a reachability property of the form: from a state s there exists a trace which leads to a state where a given automaton g is in state $g_\top$. As discussed in section 4, such kind of properties allows the expression of sequential (sub-)state reachability properties.

The goal-oriented reduction preserves *all* the minimal traces satisfying the reachability property under a general concurrent semantics which allows at each step simultaneous transitions of an arbitrary number of automata. These results directly apply to the asynchronous semantics where only one transition occurs at a time: any minimal trace of the asynchronous semantics is a minimal trace in the general concurrent semantics.

Its time complexity is polynomial in the total number of transitions and exponential with the maximal number of local states within an automaton minus 1 (and hence polynomial on binary automata network). Therefore, the procedure is extremely scalable when applied on networks between numerous automata, but where each automaton has a few local states.

The main benefit of this reduction is the enhancement of tractability for formal verification of the transient dynamics of the model, notably for reachability properties, as well as for cut sets properties, which require the completeness of minimal traces in the reduced model. These properties are preserved by the reduction. The experiments on several models from the literature confirm this practical impact.

Usually, input models are specified as Boolean or multi-valued networks. We showed that our framework can encode them equivalently. In some cases, it is also possible to convert the obtained reduced automata network back to functional specification, for instance using [46].

The reduction method can be applied to any automata networks, and therefore may have applications beyond Boolean and multi-valued networks for systems biology. Thus, it would be of interest to pursue a more general evaluation with different kind of models, and for these different models compare with other reduction methods, such as relevant subnet computation.

Further work consider the embedding of this reduction in different methods relying on transient dynamics analysis. For instance, one could consider applying the reduction iteratively during the state graph computation. Indeed, in general, the closer we get to the goal, the more transitions can be ignored: in the case of RB2EF models, this lead to a further shrinkage from 2 millions states to 28,000. Another

question is the assessment of the optimality of the reduction: if one can compute the concrete minimal traces, one can evaluate how close the static analysis can get to the optimal reduced network. Unfortunately, the computation of the exact and complete set of minimal traces is hardly tractable.

Acknowledgments

This work has been partially funded by ANR-FNR project "AlgoReCell" (ANR-16-CE12-0034).

Appendix A
Safe Petri nets vs Automata Networks

In this section, we detail how safe Petri nets can be encoded as automata networks, and vice versa.

A *safe or 1-bounded Petri net* is a tuple $(\mathcal{P}, \mathcal{T}, F, M_0)$ where $\mathcal{P}$ and $\mathcal{T}$ are sets of nodes, called *places* and *transitions* respectively, and $F \subseteq (\mathcal{P} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P})$ is a *flow relation* whose elements are called *arcs*. A subset $M \subseteq \mathcal{P}$ of the places is called a marking, and M_0 is a distinguished *initial marking*. For any node $x \in \mathcal{P} \cup \mathcal{T}$, we call *pre-set* of x the set ${}^\bullet x = \{y \in \mathcal{P} \cup \mathcal{T} \mid (y, x) \in F\}$ and *post-set* of x the set $x^\bullet = \{y \in \mathcal{P} \cup \mathcal{T} \mid (x, y) \in F\}$.

A transition $t \in \mathcal{T}$ is *enabled* at a marking M if and only if ${}^\bullet t \subseteq M$. The application of such a transition leads to the new marking $M' = (M \setminus {}^\bullet t) \cup t^\bullet$.

A.1 Safe Petri nets to automata networks

In general, each place of the Petri net can be modeled as a single binary automaton, where local states indicates if the place is marked or not. Then, transitions update the related automata accordingly. However, in many cases, safe Petri nets actually model the synchronized product of transitions systems: in such cases, the places model the (local) states of the compositing systems. It is therefore more natural to encode such nets with automata which makes explicit this partionning of places.

A set of places is said mutually exclusive if at most one place of this set can be present in any reachable marking. In the case when this set of places always contains one place in the marking, it can then be modeled as a single automaton where places are its local states. The identification of these set of places can be done automatically using so-called P-invariants (they are particular cases of them) [47].

In the following, we assume a set of k subsets of places $\mathcal{P}$, $\forall i \in [1;k], A^i \subseteq \mathcal{P}$, which are mutually exclusive and have one marked place in each reachable marking from M_0. Note that the sets A^i are not necessarily disjoint (see example below). We note $B \triangleq \mathcal{P} \setminus \bigcup_{i\in[1;k]} A^i$ the set of places not belonging to these subsets.

The encoding of the Petri net in an automata network (Σ, S, T) is the following. We instantiate one automaton per set A^i and per place in B: $\Sigma = \{a^i \mid i \in [1;k]\} \cup \{p \in B\}$. The local states are defined as follows: $\forall i[1;k], S(a^i) = \{a^i_p \mid p \in A^i\}$; and $\forall p \in B, S(p) = \{p_0, p_1\}$. To each Petri net transition $t \in \mathcal{T}$, we define a transition $(l, x) \in T$ where the automata changes l are $\forall p \in {}^\bullet t \setminus t^\bullet$ (consumed places), if $p \in B$, $(p_1, p_0) \in l$, otherwise, $\forall i \in [1;k]$ such that $p \in A^i$, with $q \in A^i \cap t^\bullet \setminus {}^\bullet t$, $(a^i_p, a^i_q) \in l$; and $\forall p \in B \cap t^\bullet \setminus {}^\bullet t$ (produced places), $(p_0, p_1) \in l$; finally the conditions x correspond to the places present both in pre- and post-set of t: $x = \bigcup_{p\in {}^\bullet t \cap t^\bullet} L(p)$ with, if $p \in B$, $L(p) = \{p_1\}$, otherwise $L(p) = \{a^i_p \mid i \in [1;k], p \in A^i\}$.

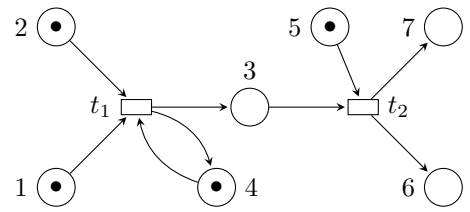

Fig. 5. Petri net of Example 5. Places are circle nodes and transitions rectangle nodes.

Example 5. Let us consider the Petri net in figure 5 with $\mathcal{P} = \{1, 2, 3, 4, 5, 6, 7\}$ and $\mathcal{T} = \{t_1, t_2\}$. Two P-invariants of mutually exclusive places can be identified: $A^1 = \{1, 3, 6\}$ and $A^2 = \{2, 3, 7\}$; hence $B = \{4, 5\}$. Remark that the place 3 belongs to two subsets of places: it will be duplicated in the automata network encoding. The resulting automata network gathers 4 automata $\Sigma = \{a^1, a^2, 4, 5\}$ with local states $S(a^1) = \{a^1_1, a^1_3, a^1_6\}$, $S(a^2) = \{a^2_2, a^2_3, a^2_7\}$, $S(4) = \{4_0, 4_1\}$, $S(5) = \{5_0, 5_1\}$. The Petri net transition t_1 is encoded as the AN transition $a^1_1, a^2_2 \xrightarrow{4_1} a^1_3, a^2_3$; and Petri net transition t_2 is encoded as the AN transition $a^1_3, a^2_3, 5_1 \xrightarrow{\emptyset} a^1_6, a^2_7, 5_0$.

A.2 Automata networks to safe Petri nets

An automata network (Σ, S, T) can be straightforwardly encoded as a safe Petri net having groups of mutually exclusive places acting as the automata, and where each transition $t \in T$ of the AN is encoded as a Petri net transition with incoming arcs from $\mathrm{orig}(t)$ and $\mathrm{enab}(t)$, out-going arcs to $\mathrm{dest}(t)$ and $\mathrm{enab}(t)$. Formally, the Petri net is defined with $\mathcal{P} = \bigcup_{a\in\Sigma} S(a)$, $\mathcal{T} = T$, and $F = \{(a_i, t) \mid a_i \in {}^\bullet t, t \in T\} \cup \{(t, a_j) \mid a_j \in t^\bullet, t \in T\}$.

Appendix B
Proof of minimal traces preservation

We assume a global AN (Σ, S, T) where $g \in \Sigma$, $g_\top \in S(g)$, and $s \in S$ with $s(g) \neq g_\top$.

From property 1 and definition 7, any trace reaching first a_i and then a_j uses all the transitions of at least one local path in $\mathrm{lpaths}_s(a_i \rightsquigarrow a_j)$.

We first prove with lemma 2 that the last transition of a minimal trace π for $g_\top$ reachability, of the form $\pi^{|\pi|} = \{(l, x)\}$ with $(g_i, g_\top) \in l$, is necessarily in $\mathrm{tr}(\mathcal{B})$. Indeed, by definition of $\mathcal{B}$, $g_0 \rightsquigarrow g_\top \in \mathcal{B}$; and by lemma 1, $(l, x) \notin \mathrm{lpaths}_s(g_0 \rightsquigarrow g_\top)$ implies that reaching g_i requires reaching $g_\top$ beforehand.

Lemma 1. *Given $(l, x) \in T$ with $(a_j, a_i) \in l$, if $(l, x) \notin \mathrm{tr}(\mathrm{lpaths}_s(a_0 \rightsquigarrow a_i))$, then for any trace π from s with $a_j \in \pi^{v\bullet}$ and $a_i \in \pi^{w\bullet}$ for some $v, w \in [1; |\pi|]$, there exists $u < v$ with $a_i \in \pi^{u\bullet}$.*

Proof. Let $\eta \in \mathrm{lpaths}_s(a_0 \rightsquigarrow a_j)$ be an a-acyclic local path such that $\forall n \in [1; |\eta|]$, $a_i \notin \mathrm{dest}(\eta^n)$. The sequence $\eta :: (l, x)$

is then acyclic and, by definition, belongs to $\text{lpaths}_s(a_0 \rightsquigarrow a_i)$, which is a contradiction. □

Lemma 2. *If π is a minimal trace for $g_\top$ reachability from state s, then, necessarily, $\pi^{|\pi|} \subseteq \text{tr}(\mathcal{B})$.*

Proof. As π is minimal for $g_\top$ reachability, without loss of generality, we can assume that $\pi^{|\pi|} = \{(l, x)\}$ with $(g_i, g_\top) \in l$. By definition, $\text{tr}(\text{lpaths}_s(g_0 \rightsquigarrow g_\top)) \subseteq \text{tr}(\mathcal{B})$. By lemma 1, if $(l, x) \notin \text{tr}(\text{lpaths}_s(g_0 \rightsquigarrow g_\top))$, then there exists $u < |\pi|$ with $g_\top \in \pi^u{}^\bullet$; hence, π would be non minimal. □

The rest of the proof of theorem 1 is derived by contradiction: if a transition of π is not in $\text{tr}(\mathcal{B})$, we can build a sub-trace of π which preserves $g_\top$ reachability, therefore π is not minimal.

Given a transition (l, x) in the q-th step of π that is not in $\text{tr}(\mathcal{B})$, removing (l, x) from π^q would imply to remove any further transition that depend causally on it. Two cases arise from this fact: either all further transitions that depend on (l, x) must be removed, if any; or (l, x) is part of loop which can be removed from π.

Lemma 3 ensures that if $a_z \rightsquigarrow a_k$ is in $\mathcal{B}$ and if a_z occurs before the q-th step and a_k after the q-th step of π, then $(l, x) \notin \text{tr}(\text{lpaths}_s(a_z \rightsquigarrow a_k))$ with $(a_i, a_j) \in l$ only if (l, x) is part of a loop in automaton a, i.e., there are two steps surrounding q where the automaton a is in the same state before their application. Intuitively, lemma 3 imposes that π has the form illustrated in figure 6.

Lemma 3. *Let $a \in \Sigma$ and $u, q, v \in [1; |\pi|]$, $u \leq q < v$, such that $a_z \in {}^\bullet\pi^u$, $a_k \in {}^\bullet\pi^v \cup \pi^v{}^\bullet$, and $(l, x) \in \pi^q \setminus \text{tr}(\mathcal{B})$ with $(a_i, a_j) \in l$. If $a_z \rightsquigarrow a_k \in \mathcal{B}$ then $\exists m, n \in [u; v]$, $m \leq q \leq n$ such that $(\pi^{1..m-1})^\bullet \cap S(a) = (\pi^{1..n})^\bullet \cap S(a)$; and $a_k \in {}^\bullet\pi^v \Rightarrow n < v$.*

Proof. If $(l, x) \notin \text{tr}(\mathcal{B})$ and $a_z \rightsquigarrow a_k \in \text{tr}(\mathcal{B})$, necessarily $(l, x) \notin \text{tr}(\text{lpaths}_s(a_z \rightsquigarrow a_k))$. Therefore, (l, x) belongs to a loop of a local path in automata a from a_z (at index u in π) to a_k (at index v in π). Hence, $\exists m, n \in [u; v]$ with $m \leq q \leq n$ and $a_h \in S(a)$ such that $a_h \in {}^\bullet\pi^m$ and $a_h \in \pi^n{}^\bullet$ therefore $(\pi^{1..m-1})^\bullet \cap S(a) = (\pi^{1..n})^\bullet \cap S(a) = a_h$. In the case where $a_k \in {}^\bullet\pi^v$, $a_k \neq a_h$, hence $n < v$. □

The idea is then to remove the transitions forming the loop within automaton a. However, transitions in other automata may depend causally on the transitions that compose the local loop in automaton a within steps m and n, following the notations in lemma 3.

Lemma 4 establishes that we can always find m and n such that none of the transitions within these steps with an enabling condition depending on an automaton modified by the transition (l, x) are in $\text{tr}(\mathcal{B})$. Indeed, for any a such that $\exists (a_i, a_j) \in l$, if a transition in $\text{tr}(\mathcal{B})$ depends on a local state of a, let us call it a_p, the objectives $a_0 \rightsquigarrow a_p$ and $a_p \rightsquigarrow a_k$ are in $\mathcal{B}$, due to the second and third condition in definition 8. Lemma 3 can then be applied on the subpart of π that contains the transition (l, x) not in $\text{tr}(\mathcal{B})$ and that concretizes either $a_0 \rightsquigarrow a_p$ or $a_p \rightsquigarrow a_k$ to identify a smaller loop containing (l, x).

In the following, we denote the set of automata changed by a transition (l, x) by $\Sigma(l) \triangleq \{a \in \Sigma \mid \exists (a_i, a_j) \in l\}$.

Lemma 4. *Let $q \in [1; |\pi|]$ where $(l, x) \in \pi^q \setminus \text{tr}(\mathcal{B})$. For each $a \in \Sigma(l)$, there exists $m, n \in [1; |\pi|]$ with $m \leq q \leq n$ such that $\forall t \in \text{tr}(\pi^{m+1..n})$, $\text{enab}(t) \cap S(a) \neq \emptyset \Rightarrow t \notin \text{tr}(\mathcal{B})$, and, if $g = a$ or $\exists t \in \text{tr}(\pi^{n+1..|\pi|}) \cap \text{tr}(\mathcal{B})$ with $\text{enab}(t) \cap S(a) \neq \emptyset$, then $(\pi^{1..m-1})^\bullet \cap S(a) = (\pi^{1..n})^\bullet \cap S(a)$.*

Proof. • Case 1 – $\forall a \in \Sigma(l)$, $a \neq g$ and for any $t \in \pi^{q+1..|\pi|}$, $\text{enab}(t) \cap S(a) \neq \emptyset \Rightarrow t \notin \text{tr}(\mathcal{B})$: the lemma is verified with $m = q$ and $n = |\pi|$.

• Case 2 – $\exists a \in \Sigma(l)$ and $v \in [q + 1; |\pi|]$ such that $\exists t \in \text{tr}(\pi^v) \cap \text{tr}(\mathcal{B})$ with $a_k \in \text{enab}(t)$. By definition 8, this implies $a_0 \rightsquigarrow a_k \in \mathcal{B}$. By lemma 3, there exists $m, n \in [1; v - 1]$ with $m \leq q \leq n$ such that $(\pi^{1..m-1})^\bullet \cap S(a) = (\pi^{1..n})^\bullet \cap S(a)$.

• Case 3 – $\exists a \in \Sigma(l)$ with $a = g$, by lemma 3 applied with $a_k = g_\top$, there exists $m, n \in [1; |\pi|]$ with $m \leq q \leq n$ and $m \neq n$ such that $(\pi^{1..m-1})^\bullet \cap S(a) = (\pi^{1..n})^\bullet \cap S(a)$. Remark that it is necessary that $n < |\pi|$, otherwise, if $n = |\pi|$, $g_\top \in (\pi^{1..m-1})^\bullet$ and π would be not minimal.

In both Case 2 and 3, if there exists $r \in [m + 1; n]$ such that $\exists a_p \in S(a)$ and $\exists t \in \pi^r$ with $a_p \in \text{enab}(t)$, then $t \in \text{tr}(\mathcal{B})$ implies that $a_0 \rightsquigarrow a_p \in \mathcal{B}$ and $a_p \rightsquigarrow a_k \in \mathcal{B}$ (definition 8). If $r > q$, by lemma 3 with $a_k = a_p$ and $v = r$, there exists $m', n' \in [m + 1; n]$ such that $m' \leq q \leq n' < r \leq n$ with $(\pi^{1..m'-1})^\bullet \cap S(a) = (\pi^{1..n'})^\bullet \cap S(a)$. If $r \leq q$, by lemma 3 with $a_0 = a_p$ and $u = r$, there exists $m', n' \in [m + 1; n]$ such that $r \leq m' \leq q \leq n'$ with $(\pi^{1..m'-1})^\bullet \cap S(a) = (\pi^{1..n'})^\bullet \cap S(a)$. Therefore, by induction with lemma 3, there exists $m, n \in [1; |\pi|]$ such that $\forall t \in \text{tr}(\pi^{m+1..n})$, $\text{enab}(t) \cap S(a) \neq \emptyset \Rightarrow t \notin \text{tr}(\mathcal{B})$. □

Using lemma 4, we show how we can identify a subset of transitions in π that can be removed to obtain a sub-trace for $g_\top$ reachability. In the following, we refer to the couple (m, n) for $a \in \Sigma(l)$ of lemma 4 with $\text{cb}(\pi, a, q)$ (definition 11).

Definition 11 ($\text{cb}(\pi, a, q)$)**.** Given $a \in \Sigma$, $q \in [1; |\pi|]$ with $t = (l, x) \in \pi^q \setminus \text{tr}(\mathcal{B})$ and $a \in \Sigma(l)$, $\text{cb}(\pi, a, q) \triangleq (m, n)$ where $m, n \in [1; |\pi|]$ are such that,

- $\forall t \in \text{tr}(\pi^{m+1..n})$, $\text{enab}(t) \cap S(a) \neq \emptyset \Rightarrow t \notin \text{tr}(\mathcal{B})$;
- $a = g \vee \exists t \in \text{tr}(\pi^{n+1..|\pi|}) \cap \text{tr}(\mathcal{B})$ with $\text{enab}(t) \cap S(a) \neq \emptyset \Longrightarrow (\pi^{1..m-1})^\bullet \cap S(a) = (\pi^{1..n})^\bullet \cap S(a)$. Moreover, if $a = g$, then $n < |\pi|$.

We use lemma 4 to collect the portions of π to remove depending on automata. We start from the last transition in π that is not in $\text{tr}(\mathcal{B})$: if $\text{tr}(\pi) \not\subseteq \text{tr}(\mathcal{B})$, there exists $d \in [1; |\pi|]$ such that $\pi^d \not\subseteq \text{tr}(\mathcal{B})$ and $\forall n > d, \pi^n \subseteq \text{tr}(\mathcal{B})$. By lemma 2, we know that $d < |\pi|$.

Let us define $\Psi \subseteq \Sigma \times [1; |\pi|] \times [1; |\pi|]$ the smallest set which satisfies:

- $(a, m, n) \in \Psi$ if $\text{cb}(\pi, a, d) = (m, n)$ where $(l, x) \in \pi^d \setminus \text{tr}(\mathcal{B})$.
- $\forall (a, m, n) \in \Psi$, $\forall q \in [m + 1; n]$, $\forall t = (l, x) \in \pi^q$, ${}^\bullet(t) \cap S(a) \neq \emptyset \Longrightarrow (b, m', n') \in \Psi$ for each $b \in \Sigma(l)$ where $\text{cb}(\pi, b, q) = (m', n')$.

Finally, let us define the sequence of steps ϖ as the sequence of steps π where the transitions delimited by Ψ are removed: for each $(a, m, n) \in \Psi$, all the transitions of automaton a occurring between π^m and π^n are removed.

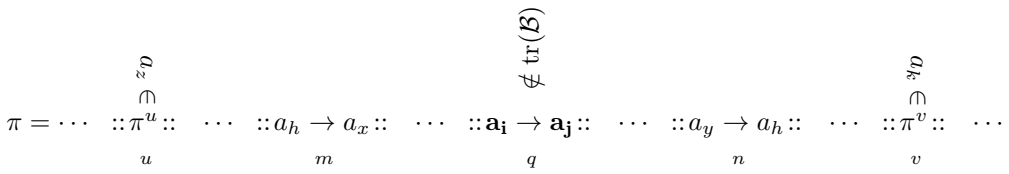

Fig. 6. By lemma 3, given that $a_z \rightsquigarrow a_k \in \mathcal{B}$, $a_i \to a_j \in \pi^q \setminus \mathrm{tr}(\mathcal{B})$ only if it is part of a cycle within automata a, here between steps m and n.

Formally, $|\varpi| = |\pi|$ and for all $q \in [1;|\pi|]$, $\varpi^q \triangleq \{(l,x) \in \pi^q \mid \forall a \in \Sigma(l), \nexists (a,m,n) \in \Psi : m \leq q \leq n\}$.

From lemma 4 and Ψ definition, ϖ is a valid trace. Moreover, by lemma 4, there is no $q \in [1;|\pi|]$ such that $(g,q,|\pi|) \in \Psi$, hence $g_\top \in \varpi^\bullet$. Therefore, π is not minimal, which contradicts our hypothesis. □

Example 6. Let us consider the reachability of c_2 in the AN of figure 1 from state $\langle a_0, b_0, c_0, d_0\rangle$. The transitions $\mathrm{tr}(\mathcal{B})$ preserved by the reduction for that goal are listed in figure 2.

Let π be the following trace in the AN of figure 1:

$$\pi = \{a_0 \xrightarrow{\{b_0\}} a_1\} :: \{b_0 \xrightarrow{\{a_1\}} b_1; c_0 \xrightarrow{\{a_1\}} c_1\} :: \\ :: \{a_1, b_1 \xrightarrow{\emptyset} a_0, b_0\} :: \{c_1 \xrightarrow{\{b_0\}} c_2\} \ .$$

The latest transition not in $\mathrm{tr}(\mathcal{B})$ is $a_1, b_1 \xrightarrow{\emptyset} a_0, b_0$ at step 3. One can compute $\mathrm{cb}(\pi, b, 3) = (2, 3)$, and as there is no transition involving b between steps 3 and 4, and $\mathrm{cb}(\pi, a, 3) = (3, |\pi|)$ as a is no longer involve in subsequent transitions. Hence, $\Psi = \{(b, 2, 3), (a, 3, 4)\}$; therefore, the sequence

$$\varpi = \{a_0 \xrightarrow{\{b_0\}} a_1\} :: \{c_0 \xrightarrow{\{a_1\}} c_1\} ::: \{\} :: \{c_1 \xrightarrow{\{b_0\}} c_2\}$$

is a valid sub-trace of π reaching c_2, proving π non-minimality.

In conclusion, if π is a minimal trace for $g_\top$ reachability from state s, then, $\mathrm{tr}(\pi) \subseteq \mathrm{tr}(\mathcal{B})$.

REFERENCES

[1] W. Abou-Jaoudé, P. T. Monteiro, A. Naldi, M. Grandclaudon, V. Soumelis, C. Chaouiya, and D. Thieffry, "Model checking to assess t-helper cell plasticity," *Frontiers in Bioengineering and Biotechnology*, vol. 2, Jan 2015.

[2] D. P. A. Cohen, L. Martignetti, S. Robine, E. Barillot, A. Zinovyev, and L. Calzone, "Mathematical modelling of molecular pathways enabling tumour cell invasion and migration," *PLoS Comput Biol*, vol. 11, no. 11, p. e1004571, Nov 2015.

[3] L. Grieco, L. Calzone, I. Bernard-Pierrot, F. Radvanyi, B. Kahn-Perlès, and D. Thieffry, "Integrative modelling of the influence of MAPK network on cancer cell fate decision," *PLoS Comput Biol*, vol. 9, no. 10, p. e1003286, oct 2013.

[4] S. Klamt, J. Saez-Rodriguez, J. Lindquist, L. Simeoni, and E. Gilles, "A methodology for the structural and functional analysis of signaling and regulatory networks," *BMC Bioinformatics*, vol. 7, no. 1, p. 56, 2006.

[5] J. Saez-Rodriguez, L. Simeoni, J. A. Lindquist, R. Hemenway, U. Bommhardt, B. Arndt, U.-U. Haus, R. Weismantel, E. D. Gilles, S. Klamt, and B. Schraven, "A logical model provides insights into t cell receptor signaling," *PLoS Comput Biol*, vol. 3, no. 8, p. e163, 08 2007.

[6] O. Sahin, H. Frohlich, C. Lobke, U. Korf, S. Burmester, M. Majety, J. Mattern, I. Schupp, C. Chaouiya, D. Thieffry, A. Poustka, S. Wiemann, T. Beissbarth, and D. Arlt, "Modeling ERBB receptor-regulated G1/S transition to find novel targets for de novo trastuzumab resistance," *BMC Systems Biology*, vol. 3, no. 1, pp. 1–20, 2009.

[7] R. Samaga, J. Saez-Rodriguez, L. G. Alexopoulos, P. K. Sorger, and S. Klamt, "The logic of egfr/erbb signaling: Theoretical properties and analysis of high-throughput data," *PLoS Comput Biol*, vol. 5, no. 8, p. e1000438, 08 2009.

[8] N. Weinstein and L. Mendoza, "A network model for the specification of vulval precursor cells and cell fusion control in caenorhabditis elegans," *Frontiers in Genetics*, vol. 4, no. 112, 2013.

[9] A. Cheng, J. Esparza, and J. Palsberg, "Complexity results for 1-safe nets," *Theor. Comput. Sci.*, vol. 147, no. 1&2, pp. 117–136, 1995.

[10] J. Sifakis, "Property preserving homomorphisms of transition systems," in *Logics of Programs*, ser. Lecture Notes in Computer Science, E. Clarke and D. Kozen, Eds. Springer Berlin Heidelberg, 1984, vol. 164, pp. 458–473.

[11] R. P. Kurshan, *Computer-aided verification of coordinating processes: the automata-theoretic approach*. Princeton university press, 1994.

[12] C. Loiseaux, S. Graf, J. Sifakis, A. Bouajjani, S. Bensalem, and D. Probst, "Property preserving abstractions for the verification of concurrent systems," *Formal methods in system design*, vol. 6, no. 1, pp. 11–44, 1995.

[13] J. Feret, H. Koeppl, and T. Petrov, "Stochastic fragments: A framework for the exact reduction of the stochastic semantics of rule-based models," *International Journal of Software and Informatics*, vol. 7, no. 4, pp. 527 – 604, 2013.

[14] V. Danos, J. Feret, W. Fontana, R. Harmer, J. Hayman, J. Krivine, C. D. Thompson-Walsh, and G. Winskel, "Graphs, rewriting and pathway reconstruction for rule-based models," in *FSTTCS 2012*, ser. LIPIcs, vol. 18. Schloss Dagstuhl, 2012, pp. 276–288.

[15] G. Madelaine, C. Lhoussaine, and J. Niehren, "Attractor Equivalence: An Observational Semantics for Reaction Networks," in *First International Conference on Formal Methods in Macro-Biology*, ser. Lecture Notes in Computer Science, vol. 8738. Springer-Verlag, 2014, pp. 82–101.

[16] A. Naldi, E. Remy, D. Thieffry, and C. Chaouiya, "Dynamically consistent reduction of logical regulatory graphs," *Theoretical Computer Science*, vol. 412, no. 21, pp. 2207 – 2218, 2011.

[17] P. Schnoebelen and N. Sidorova, "Bisimulation and the reduction of Petri nets," in *Application and Theory of Petri Nets 2000*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2000, vol. 1825, pp. 409–423.

[18] G. Berthelot, "Checking properties of nets using transformations," in *Advances in Petri Nets 1985*, ser. Lecture Notes in Computer Science, G. Rozenberg, Ed. Springer Berlin Heidelberg, 1986, vol. 222, pp. 19–40.

[19] S. Haddad and J.-F. Pradat-Peyre, "New efficient Petri nets reductions for parallel programs verification," *Parallel Processing Letters*, vol. 16, no. 1, pp. 101–116, Mar. 2006.

[20] A. Biere, E. Clarke, R. Raimi, and Y. Zhu, "Verifying safety properties of a powerpc microprocessor using symbolic model checking without bdds," in *In Proc. 11 th Int. Conf. on Computer Aided Verification*. Springer-Verlag, 1999, pp. 60–71.

[21] C. Talcott and D. L. Dill, "Multiple representations of biological processes," in *Transactions on Computational Systems Biology VI*. Springer Science Business Media, 2006, pp. 221–245.

[22] L. Paulevé, M. Magnin, and O. Roux, "Static analysis of biological regulatory networks dynamics using abstract interpretation," *Mathematical Structures in Computer Science*, vol. 22, no. 04, pp. 651–685, 2012.

[23] L. Paulevé, G. Andrieux, and H. Koeppl, "Under-approximating cut sets for reachability in large scale automata networks," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, N. Sharygina and H. Veith, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, vol. 8044, pp. 69–84.

[24] L. Paulevé, "Goal-Oriented Reduction of Automata Networks," in *CMSB 2016 - 14th conference on Computational Methods for Systems*

Biology, ser. Lecture Notes in Bioinformatics, vol. 9859. Springer, 2016.
[25] A. Arnold, *Finite Transition Systems: Semantics of Communicating Systems*. Hertfordshire, UK, UK: Prentice Hall International (UK) Ltd., 1994.
[26] L. Bernardinello and F. De Cindio, "A survey of basic net models and modular net classes," in *Advances in Petri Nets 1992*, ser. Lecture Notes in Computer Science, G. Rozenberg, Ed. Springer Berlin / Heidelberg, 1992, vol. 609, pp. 304–351.
[27] R. Janicki, P. E. Lauer, M. Koutny, and R. Devillers, "Concurrent and maximally concurrent evolution of nonsequential systems," *Theoretical Computer Science*, vol. 43, no. 0, pp. 213 – 238, 1986.
[28] L. Priese and H. Wimmel, "A uniform approach to true-concurrency and interleaving semantics for petri nets," *Theoretical Computer Science*, vol. 206, no. 1–2, pp. 219 – 256, 1998.
[29] R. Janicki, J. Kleijn, M. Koutny, and Ł. Mikulski, "Step traces," *Acta Informatica*, Jun 2015.
[30] J. Aracena, E. Goles, A. Moreira, and L. Salinas, "On the robustness of update schedules in Boolean networks," *Biosystems*, vol. 97, no. 1, pp. 1 – 8, 2009.
[31] D. Thieffry and R. Thomas, "Dynamical behaviour of biological regulatory networks—ii. immunity control in bacteriophage lambda," *Bulletin of Mathematical Biology*, vol. 57, pp. 277–297, 1995.
[32] G. Bernot, F. Cassez, J.-P. Comet, F. Delaplace, C. Müller, and O. Roux, "Semantics of biological regulatory networks," *Electronic Notes in Theoretical Computer Science*, vol. 180, no. 3, pp. 3 – 14, 2007.
[33] A. Richard, "Negative circuits and sustained oscillations in asynchronous automata networks," *Advances in Applied Mathematics*, vol. 44, no. 4, pp. 378 – 392, 2010.
[34] R. Thomas, "Boolean formalization of genetic control circuits," *Journal of Theoretical Biology*, vol. 42, no. 3, pp. 563 – 585, 1973.
[35] L. Paulevé and A. Richard, "Static analysis of boolean networks based on interaction graphs: a survey," *Electronic Notes in Theoretical Computer Science*, vol. 284, pp. 93 – 104, 2011, proceedings of The Second International Workshop on Static Analysis and Systems Biology (SASB 2011).
[36] E. J. McCluskey, "Minimization of boolean functions," *The Bell System Technical Journal*, vol. 35, no. 6, pp. 1417–1444, Nov 1956.
[37] F. Fages, T. Martinez, D. A. Rosenblueth, and S. Soliman, *Influence Systems vs Reaction Systems*. Cham: Springer International Publishing, 2016, pp. 98–115.
[38] Colomoto, "bioLQM," https://github.com/colomoto/logicalmodel.
[39] J. Esparza and K. Heljanko, *Unfoldings: A Partial-Order Approach to Model Checking (Monographs in Theoretical Computer Science. An EATCS Series)*, 1st ed. Berlin, Heidelberg: Springer Publishing Company, Incorporated, 2008.
[40] I. Curie/Sysbio, "RB/E2F pathway," http://bioinfo-out.curie.fr/projects/rbpathway/.
[41] LIP6/Move, "Its tools," http://ddd.lip6.fr/itstools.php.
[42] S. Schwoon, "Mole," http://www.lsv.ens-cachan.fr/~schwoon/tools/mole/.
[43] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella, "NuSMV 2: An open-source tool for symbolic model checking," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2002, vol. 2404, pp. 241–268.
[44] A. Hamez, Y. Thierry-Mieg, and F. Kordon, "Building efficient model checkers using hierarchical set decision diagrams and automatic saturation," *Fundam. Inf.*, vol. 94, no. 3-4, pp. 413–437, 2009.
[45] E. M. Clarke and E. A. Emerson, "Design and synthesis of synchronization skeletons using branching-time temporal logic," in *Logic of Programs*. London, UK: Springer-Verlag, 1981, pp. 52–71.
[46] M. Folschette, L. Paulevé, K. Inoue, M. Magnin, and O. Roux, "Identification of Biological Regulatory Networks from Process Hitting models," *Theoretical Computer Science*, vol. 568, no. 0, pp. 49 – 71, 2015.
[47] T. Murata, "Petri nets: properties, analysis and applications," *Proc. of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

Loïc Paulevé is a tenured researcher at CNRS in the Bioinfo group of LRI (computer science) laboratory at Université Paris-Sud, Université Paris-Saclay. He got his M.Sc. in computer science from École Normale Supérieure de Cachan, Brittany extension, and his PhD in computational systems biology from École Centrale de Nantes. His research focuses on the development and application of static analysis by abstract interpretation of automata networks for biological models.