Sufficient Conditions for Reachability in Automata Networks with Priorities

Maxime Folschette[a,b], Loïc Paulevé[c], Morgan Magnin[a], Olivier Roux[a]

[a] *LUNAM Université, École Centrale de Nantes, IRCCyN UMR CNRS 6597 (Institut de Recherche en Communications et Cybernétique de Nantes) 1 rue de la Noë - B.P. 92101 - 44321 Nantes Cedex 3, France.*
[b] *School of Electrical Engineering and Computer Science, University of Kassel, Germany*
[c] *CNRS, Laboratoire de Recherche en Informatique (LRI) Université Paris-Sud - CNRS UMR 8623, France*

Abstract

In this paper, we develop a framework for an efficient under-approximation of the dynamics of Asynchronous Automata Networks (AANs). An AAN is an Automata Network with synchronised transitions between automata, where each transition changes the local state of exactly one automaton (but any number of synchronizing local states are allowed). The work we propose here is based on static analysis by abstract interpretation, which allows to prove that reaching a state with a given property is possible, without the same computational cost of usual model checkers: the complexity is polynomial with the total number of local states and exponential with the number of local states within a single automaton. Furthermore, we address AANs with classes of priorities, and give an encoding into AANs without priorities, thus extending the application range of our under-approximation. Finally, we illustrate our method for the model checking of large-scale biological networks.

Keywords: discrete networks, abstract interpretation, reachability, qualitative models, systems biology

1. Introduction

Discrete modelling frameworks for biological networks is an active research field where formal methods have be proven very powerful [1, 2, 3]. Such a work started in the seventies, with the emergence of the notion of Boolean Network [4] and its use to represent biological phenomena [5]. It was later enriched in many directions and widely used to elucidate many biological questions. Among these questions, a major one is to understand precisely how biological systems evolve and behave; why and how they change their usual behaviour, etc. These

Email address: `Maxime.Folschette@irccyn.ec-nantes.fr` (Maxime Folschette)

questions are strongly linked to the (possible or inevitable) reachability of some states. The ultimate goal is to discover how it could be possible to prevent biological systems from reaching some pathological conditions.

Of course, such formal models on which analyses are performed are abstract representations of the actual studied systems. They are associated with parameters that have to be synthesised to give the most faithful representation of the real systems with their observed behaviours. As a matter of fact, the abstractions we get are more or less rough or accurate. Prevalent formal frameworks for such modelling activities are state-transition systems or process algebras. We developed a quite similar framework named the Process Hitting [6], consisting in a restriction of these frameworks where the evolution of a component is determined by the state of at most one other component that does not evolve. This is modelled by actions of the form $X + Y \to X + Y'$, where X behaves like a catalyst molecule that "hits" another molecule Y and changes it into Y', without being changed itself. Assuming catalysts are always available, this can represent any biochemical system made of monomolecular reactions, and can also represent catalytic networks such as metabolic networks. Our motivation behind this framework was to design a model and analysis techniques adapted to biological modelling. These analyses avoid building the whole state space, which allows to tackle very large systems (that would have led to a huge number of states, hopelessly too huge to be analysed). They are based on the fact that most biological models have few levels of expression per component: in Boolean networks [4, 5] there are only two levels per component, and in their multivalued equivalent [2], components rarely have more than four levels.

Besides, one further objective of our work is now to improve the accuracy of the description of the studied systems dynamics. The idea for this is to introduce timing features into models: we are interested in taking into account some knowledge about the relative length of some phenomena as it is a way to refute some models (or parameters) that are inconsistent with the observed dynamic behaviours. In this paper, we are dealing with these timing properties through priorities, that are based on the simple founding idea that actions with higher priority have to be processed before the ones with lower priority. Furthermore, due to the Process Hitting framework restrictions, multimolecular reactions were previously not immediately available, but one could simulate them with an encoding called "cooperative sort". That encoding however introduces extra reactions, that produce a temporal shift between the presence of the reactants and the playability of the reaction. This is where the priorities become useful, if not necessary: the extra reactions can for example be given "infinite speed" (highest priority) so that they do not affect the behaviour of "normal" (lower priority) reactions, including the multimolecular ones.

The approach used in this paper consists in considering a broader class of models, that we call Asynchronous Automata Network, and that allows to naturally model these cooperations by defining several requisites for a transition. Moreover, such automata networks are still compatible with the notion of priority, that can also be used to model different reaction rates in the model. Asynchronous Automata Networks (and, a fortiori, their restriction, the Pro-

cess Hitting framework) can be considered as a subset of Communicating Finite State Machines or safe Petri Nets [7]. Our work is thus related to such semantic ramifications of extending traditional process algebras with the concept of priority that allows for some transitions to be given precedence over others. We focus here on static priorities that allow to model time constraints such as reaction rates or delays between regulations, but can also model preemptions between evolution branches.

Until now, such a priority scheduling of the actions was not studied extensively in the different formal modelling frameworks dedicated to systems biology. Nevertheless, such an attempt has been carried out for Petri nets by F. Bause [8], and the concept of priority relations among the transitions of a network has also more recently been introduced by A. K. Wagler *et al.* [9, 10] in order to allow modelling deterministic systems for biological applications. The concept of priority is rather straightforward in the approach of process algebras as it was shown by R. Cleaveland and M. Hennessy in [11] and their abstractions and equivalences were studied in [12]. It was later extended for applications in the field of systems biology by M. John *et al.* [13].

Contributions

In this paper, we develop a method that allows to efficiently compute under-approximations of the dynamics of Asynchronous Automata Networks (AANs), generalising a prior work carried out on Process Hitting [14]. Rather than using brute force or symbolic model checking techniques, our method focuses on static analysis by abstract interpretation. Our work aims at checking reachability properties that, for a given initial state s and the discrete expression level n of a given component x, have the form: "Starting from state s, is it possible to reach a state in which x is at level n?", thus answering either "True" (in which case the reachability is formally proven) or "Inconclusive" (which however does not stand for "False", that can be proven by other analysis tools, such as the over-approximation proposed in [14]). Moreover, the successive or simultaneous reachability of several local states can also be checked, and in the case of a "True" response, an execution path satisfying the property can be produced. Extensions of AANs where transitions are split in priority classes are addressed with an encoding in AANs without classes of priorities. Our work thus allows to efficiently analyse the dynamics of regulation networks, especially the widespread Logical Networks [15, 2], which encompass variables with a limited number of discrete values alongside with evolution functions or focal parameters. To show the scalability of our method, we apply it to two large-scale biological models with around 100 components.

Sect. 2 presents the Asynchronous Automata Networks (AANs) without any notion of classes of priorities. In Sect. 3, we develop our under-approximation method allowing to efficiently compute reachability analyses; we also show how to extract a valid execution path if the response is positive, and propose two refinements in case one needs to check successive or simultaneous reachability properties. The framework of AANs with classes of priorities is given in Sect. 4,

alongside with an encoding into AANs without classes of priorities (or, equivalently, with one class of priority). Finally, Sect. 5 provides a detailed example and two large-scale examples of the application of our method, and Sect. 6 gives a conclusion and a discussion about our work.

The main additions in this paper compared to [16] are Subsection 3.3 that allows the extraction of a concrete trace of execution when our static analysis method is conclusive, Subsection 3.4 that refines our approach in the case of a successive reachability in order to increase the conclusiveness, and Sect. 4 which states that any ANN with any number of classes of priorities can be represented into a simple ANN (or, equivalently, with only one class of priority), thus extending the scope of our method. We note however that the use of AANs without priorities alone instead of Process Hitting models allows to simplify the notations, but does not increase the range of applicability of the results. Finally, Sect. 5 has been improved with a detailed example and a new large-scale example with a quantitative examination of the results.

Notations

Sets. If A is a finite set, $|A|$ is the cardinality of A and $\wp(A)$ is the power set of A. $\mathbb{N}$ is the set of natural numbers, $\mathbb{N}^* = \mathbb{N} \setminus \{0\}$ is the set of positive natural numbers and $[\![x; y]\!] = \{x, x+1, \ldots, y-1, y\}$ is the set of natural numbers from x to y included. The Cartesian product of sets is denoted by $\times$ and if z is a tuple of n components, $\widetilde{z}$ denotes the corresponding set: $\widetilde{z} = \{z_1, \cdots, z_n\}$.

Sequences. We denote by ε the empty sequence. If $n \in \mathbb{N}$ and $x = (x_i)_{i \in [\![1;n]\!]}$ is a sequence of elements indexed by $i \in [\![1; n]\!]$, then $|x| = n$ is the length of x and $\mathbb{I}^x = [\![1; n]\!]$ is the set of indexes of this sequence. Furthermore, if $a, b \in \mathbb{I}^x$ with $a \leq b$, then $x_{a..b} = (x_i)_{i \in [\![a;b]\!]}$ is the subsequence of x between indexes a and b included. Finally, if x is a sequence, $\widetilde{x}$ also denotes the corresponding set: $\widetilde{x} = \{x_1, \cdots, x_{|x|}\}$.

Digraphs. If (V, E) is a directed graph whose set of nodes is V and whose set of edges is $E \subseteq V \times V$, the children of a node n are given by $\mathsf{children} : V \to \wp(V)$, with $\mathsf{children}(n) = \{m \in V \mid (n, m) \in E\}$; its parents are given by $\mathsf{parents} : V \to \wp(V)$ with $\mathsf{parents}(n) = \{m \in N \mid (m, n) \in E\}$; and its successors are given by $\mathsf{conn}_{(V,E)}(n)$ which is the least fixed point containing n of the function $\mathsf{fconn} : \wp(V) \to \wp(V)$ with $\mathsf{fconn}(W) = \bigcup_{m \in W} \mathsf{children}(m)$.

2. Asynchronous Automata Networks

We give in this section the definition and the semantics of the Asynchronous Automata Networks (AANs). It is a restriction of the classical (synchronous) Automata Networks where each set of transitions sharing the same label can only change one local state at a time. We also discuss how it is related to the Process Hitting framework (with or without classes of priorities). Another definition of AANs introducing classes of priorities is proposed in Sect. 4, where we also show that they have the same expressivity as AANs.

We consider an AAN (Def. 1) which gathers a finite number of *automata*, each one containing a finite number of *local states*. A local state is noted a_i, where a is the name of the automaton it belongs to, and i is the identifier of this local state within automaton a. A *global state* of the system is a Cartesian product with exactly one local state from each automata.

The concurrent interactions between local states are defined by a set of *actions*. An action stands for a set of transitions sharing the same label, so that playing them changes exactly one local state. Therefore, an action is denoted by $A \to b_j \nearrow b_k$, where A is a set of local states and b_j and b_k are local states of a same automaton b; it is moreover required that $b_j \neq b_k$ and that A does not contain a local state of b or two local states of the same automaton. An action $h = A \to b_j \nearrow b_k$ is read as "*A hits* b_j to make it bounce to b_k", and A, b_j, b_k are called respectively the (set of) *hitters*, the *target* and the *bounce* of the action, and can be referred to as $\mathsf{hitters}(h)$, $\mathsf{target}(h)$ and $\mathsf{bounce}(h)$, respectively.

Definition 1 (Asynchronous Automata Networks). An *Asynchronous Automata Network* (AAN) is a triplet $\mathcal{A} = (\Sigma; \mathcal{L}; \mathcal{H})$, where:

- $\Sigma \triangleq \{a, b, \ldots, z\}$ is the finite set of *automata*;
- $\mathcal{L} \triangleq \underset{a \in \Sigma}{\times} \mathcal{L}_a$ is the finite set of (global) *states*, where $\mathcal{L}_a = \{a_0, \ldots, a_{l_a}\}$ is the finite set of *local states* of automaton $a \in \Sigma$, with $l_a \in \mathbb{N}^*$, and so that: $\forall (a_i; b_j) \in \mathcal{L}_a \times \mathcal{L}_b, a \neq b \Rightarrow a_i \neq b_j$;
- $\mathcal{H} \triangleq \{A \to b_j \nearrow b_k \mid b \in \Sigma \wedge (b_j; b_k) \in \mathcal{L}_b \times \mathcal{L}_b \wedge b_j \neq b_k \wedge \forall a \in \Sigma, |A \cap \mathcal{L}_a| \leq 1 \wedge A \cap \mathcal{L}_b = \emptyset\}$ is the finite set of *actions*.

Furthermore, we call $\mathbf{LS} \triangleq \bigcup_{a \in \Sigma} \mathcal{L}_a$ the set of all local states in the model.

Notations. The automaton that a local state a_i belongs to is referred to as $\Sigma(a_i) = a$, and if $A \subseteq \mathbf{LS}$, we denote: $\Sigma(A) = \{\Sigma(a_i) \in \Sigma \mid a_i \in A\}$. Given a state $s \in \mathcal{L}$, the local state of automaton $a \in \Sigma$ present in s is denoted by $s[a]$, that is, the a-coordinate of the state s. If $a_i \in \mathbf{LS}$, we denote $a_i \in s \Leftrightarrow s[a] = a_i$; and if $A \subseteq \mathbf{LS}$, $A \subseteq s \Leftrightarrow \forall a_i \in A, s[a] = a_i$.

Definition 2 (Semantics of an AAN ($\to_{\mathcal{A}}$)). If $\mathcal{A} = (\Sigma; \mathcal{L}; \mathcal{H})$ is an AAN, an action $h = A \to b_j \nearrow b_k \in \mathcal{H}$ is *playable* in $s \in \mathcal{L}$ if and only if $A \subseteq s$ and $s[b] = b_j$. In such a case, $(s \cdot h)$ stands for the state resulting from the play of the action h in s, which is defined by: $(s \cdot h)[b] = b_k$ and $\forall a \in \Sigma, a \neq b, (s \cdot h)[a] = s[a]$. Moreover, we denote: $s \to_{\mathcal{A}} (s \cdot h)$.

If $s \in \mathcal{L}$, a *scenario* δ from s is a (possibly empty) sequence of actions of $\mathcal{H}$ that can be played successively in s. The set of all scenarios from s is noted $\mathbf{Sce}(s)$.

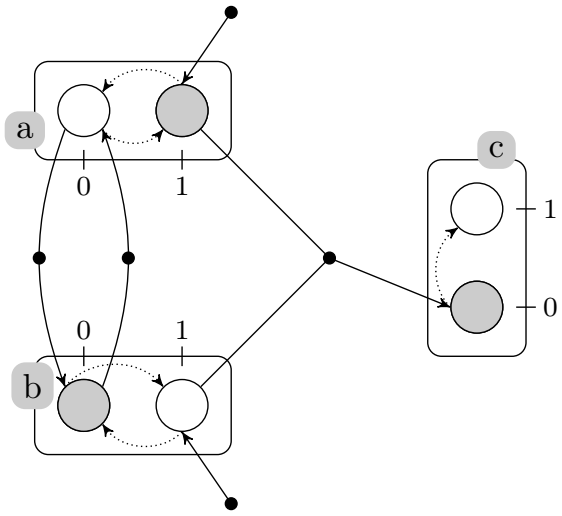

Figure 1: An example of AAN. This model represents the interaction of two components a and b, whose production is mutually exclusive and that degrade over time. Moreover, these two components can cooperate to activate c if their "active" states (a_1 and b_1) are present in the same state. Automata are represented by labelled boxes and local states by circles with their identifier on the side. Actions are represented by a dot connected by an edge to the set of hitters and by an arrow to the target, followed by another dotted arrow towards the bounce. Greyed local states stand for the following possible global state: $\langle a_1, b_0, c_0 \rangle$.

Example. Fig. 1 gives an example of AAN where:

$$\Sigma = \{a, b, c\} \ , \qquad \mathcal{L}_a = \{a_0, a_1\} \ ,$$
$$\mathcal{L}_b = \{b_0, b_1\} \ , \qquad \mathcal{L}_c = \{c_0, c_1\} \ ,$$
$$\mathcal{H} = \{ \quad \emptyset \to a_1 \Rsh a_0 \ , \quad \{b_0\} \to a_0 \Rsh a_1 \ ,$$
$$\emptyset \to b_1 \Rsh b_0 \ , \quad \{a_0\} \to b_0 \Rsh b_1 \ ,$$
$$\{a_1, b_1\} \to c_0 \Rsh c_1 \qquad \}$$

Remark (Comparison with the Process Hitting). The Process Hitting framework previously introduced in [6] is a restriction of the AAN formalism; indeed, a Process Hitting model is an AAN so that $\forall h \in \mathcal{H}, 0 \leq |\mathsf{hitters}(h)| \leq 1$. However, the AANs defined in this paper have the same expressivity than the Process Hitting with classes of priorities, as previously introduced in [16]. Indeed, in the Process Hitting with at least 2 classes of priorities, it is possible to use additional automata called "cooperative sorts" in order to model the actions in an AAN that have more than one hitter.

3. Under-approximation of Reachability

We present a static analysis that takes as input an AAN $\mathcal{A} = (\Sigma; \mathcal{L}; \mathcal{H})$, an initial state in $\mathcal{L}$, and a local state in **LS**. Its objective it to identify sufficient

conditions that ensure the existence of a scenario starting from the initial state, and leading to a state where the given local state is present.

A classical approach for determining if a local state of an automaton can be reached from a given initial state is to build sequences of transitions from the initial state until reaching a state in which the given local state is active. This is essentially what model-checkers do, and the complexity of such analysis (PSPACE-complete [17]) makes it intractable on large systems, even with advanced symbolic approaches [14].

Our approach relies on abstractions of scenarios (Subsection 3.1) that have been introduced in [14] for the static analysis of reachability in the Process Hitting framework, a particular sub-class of AAN (see the Remark at the end of the previous section). In this paper, we generalize the static analysis for the case of the under-approximation of reachability in any AAN (Subsection 3.2).

In Subsection 3.3, we detail a procedure to extract a witness scenario concretizing a given local state reachability property. Finally, we discuss in Subsection 3.4 how the static analysis can be extended to sequential (sub)states reachability properties.

3.1. Abstractions for Scenarios

The approach presented in this section is based on two complementary notions, the *objectives* and their *local causality*, that are intertwined in so-called *Local Causality Graphs*.

3.1.1. Objectives

An *objective* (Def. 3) denotes the reachability of a local state (e.g., a_j) of a given automaton a from the initial local state of that automaton (e.g., a_i). Such an objective is written $a_i \Rsh^* a_j$. Successive objectives are described with objective sequences.

Definition 3 (Objective (Obj) & Objective Sequence (OSeq)). If $a \in \Sigma$, the reachability of a local state a_j from a local state a_i is called an *objective*, noted $a_i \Rsh^* a_j$. The set of all objectives is noted:

$$\mathbf{Obj} \triangleq \{a_i \Rsh^* a_j \mid a \in \Sigma \wedge (a_i, a_j) \in \mathcal{L}_a \times \mathcal{L}_a\} \ .$$

For an objective $P = a_i \Rsh^* a_j \in \mathbf{Obj}$, we define: $\Sigma(P) \triangleq a$, $\mathsf{target}(P) \triangleq a_i$, $\mathsf{bounce}(P) \triangleq a_j$. Finally, P is said *trivial* iff $a_i = a_j$.

We define an *objective sequence* as a sequence of objectives in which each objective target must be equal to the previous objective bounce of the same automaton, if it exists. The set of all objective sequences is denoted by **OSeq**. Given $\omega \in \mathbf{OSeq}$, $\Sigma(\omega) \triangleq \{\Sigma(\omega_i) \mid i \in \mathbb{I}^\omega\}$. For each automaton $a \in \Sigma(\omega)$, the first local state of a referenced in ω is denoted by $\mathsf{first}_a(\omega) \triangleq \mathsf{target}(\omega_m)$, where $m = \min\{n \in \mathbb{I}^\omega \mid \Sigma(\omega) = a\}$. The set of objective sequences starting in a state $s \in \mathcal{L}$ are denoted by $\mathbf{OSeq}(s) \triangleq \{\omega \in \mathbf{OSeq} \mid \forall a \in \Sigma(\omega), \mathsf{first}_a(\omega) \in s\}$.

We define the partial ordering relation $\preccurlyeq_{\mathbf{OSeq}}$ between two objective sequences (Def. 4) as follows: $w \preccurlyeq_{\mathbf{OSeq}} w'$ if and only if there exists a mapping between all the objective bounces of w' in w that preserves the sequentiality, provided that each automaton starts in the same local state.

Definition 4 ($\preccurlyeq_{\mathbf{OSeq}} \subset \mathbf{OSeq} \times \mathbf{OSeq}$). $\omega \preccurlyeq_{\mathbf{OSeq}} \omega'$ if and only if $|\omega| \geq |\omega'|$, $\forall a \in \Sigma(\omega'), a \in \Sigma(\omega) \wedge \mathsf{first}_a(\omega') = \mathsf{first}_a(\omega)$; and there exists a mapping $\phi : \mathbb{I}^{\omega'} \mapsto \mathbb{I}^{\omega}$ such that $\forall n, m \in \mathbb{I}^{\omega'}, n < m \Leftrightarrow \phi(n) < \phi(m)$, and $\forall n \in \mathbb{I}^{\omega'}, \mathsf{bounce}(\omega'_n) = \mathsf{bounce}(\omega_{\phi(n)})$.

Example.

$$b_0 \Rsh^* b_1 :: a_0 \Rsh^* a_1 :: b_1 \Rsh^* b_2 \preccurlyeq_{\mathbf{OSeq}} a_0 \Rsh^* a_1 :: b_0 \Rsh^* b_2 \preccurlyeq_{\mathbf{OSeq}} b_0 \Rsh^* b_2$$

$$b_0 \Rsh^* b_1 \not\preccurlyeq_{\mathbf{OSeq}} b_0 \Rsh^* b_2 \quad \text{and} \quad b_0 \Rsh^* b_2 \not\preccurlyeq_{\mathbf{OSeq}} b_0 \Rsh^* b_1$$

An objective sequence can be seen as an abstract representation of a set of scenarios that describe (part of) the successive state changes of the automata. We denote by $\gamma_s(\omega)$ (Def. 5) the set of scenarios matching with an objective sequence ω in the state $s \in \mathcal{L}$. It is essentially all the scenarios for which there exists a mapping from the bounces of each objective to the bounce of an action in the scenario which preserve the sequential ordering.

Definition 5 ($\gamma_s : \mathbf{OSeq} \to \wp(\mathbf{Sce})$). Given a state $s \in \mathcal{L}$ and an objective sequence $\omega \in \mathbf{OSeq}(s)$, $\gamma_s(\omega)$ is the set of scenarios matching with ω:

$$\begin{aligned}\gamma_s(\omega) \triangleq \{\delta \in \mathbf{Sce}(s) \mid \omega^{\Delta} \neq \varepsilon \Rightarrow \mathsf{bounce}(\delta_{|\delta|}) = \mathsf{bounce}(\omega_{|\omega|}) \\ \wedge \exists \phi : \mathbb{I}^{\omega} \mapsto \mathbb{I}^{\delta}, (\forall n, m \in \mathbb{I}^{\omega}, n < m \Leftrightarrow \phi(n) \leq \phi(m) \\ \wedge \forall n \in \mathbb{I}^{\omega}, \mathsf{bounce}(\omega_n) \in s \cdot \delta_{1..\phi(n)})\} \ ,\end{aligned}$$

in which ω^{Δ} refers to the objective sequence ω where trivial objectives have been removed. The notation $\delta_{j..k}$ denotes the subsequence of δ between indexes j and k, as defined on page 4.

From Def. 4 and Def. 5, we derive that if $\omega \preccurlyeq_{\mathbf{OSeq}} \omega'$, then the scenarios matching ω also match ω' (Lemma 1).

Lemma 1. $\forall \omega, \omega' \in \mathbf{OSeq}, \forall s \in \mathcal{L}, \omega \preccurlyeq_{\mathbf{OSeq}} \omega' \Longrightarrow \gamma_s(\omega) \subseteq \gamma_s(\omega')$.

Given a non-empty set $\Delta \subseteq \mathbf{Sce}(s)$ of non-empty scenarios that have a common last bounce ($\exists a_i \in \mathbf{LS}, \forall \delta \in \Delta, \mathsf{bounce}(\delta_{|\delta|}) = a_i$), one can define an abstraction α_s of such a set as the smallest (according to $\preccurlyeq_{\mathbf{OSeq}}$) objective sequence $\omega \in \mathbf{OSeq}(s)$ such that $\mathsf{bounce}(\omega_{|\omega|}) = \mathsf{bounce}(\delta_{|\delta|}), \delta \in \Delta$ and $\forall \delta \in \Delta, \exists \phi : \mathbb{I}^{\omega} \mapsto \mathbb{I}^{\delta} (\forall n, m \in \mathbb{I}^{\omega}, n < m \Leftrightarrow \phi(n) \leq \phi(m)) \wedge \forall n \in \mathbb{I}^{\omega}, \mathsf{bounce}(\omega_n) \in s \cdot \delta_{1..\phi(n)})$. In such a setting, α_s and γ_s form a Galois connection between sets of scenarios and objective sequences.

3.1.2. Local Causality for Objectives

The existence of a scenario from a state s leading to a state where a given local state a_j is present can be reformulated as the checking for the non-emptiness of $\gamma_s(a_i \Rsh^* a_j)$, where $a_i = s[a]$. A first hint for checking this emptiness is to look for actions that have to be played in order to reach the state j from i within the automaton a.

Given an objective $P = a_i \Rsh^* a_j \in \mathbf{Obj}$, we define $\mathbf{BSeq}(P)$ the *bounce sequences* of P as the set of minimal sequences of actions hitting a in which the bounce of each action is the target of the following action (Def. 6).

Definition 6 (Bounce Sequence (BSeq)). A *bounce sequence* ζ is a sequence of actions such that $\forall n \in \mathbb{I}^\zeta, n < |\zeta|, \mathsf{bounce}(\zeta_n) = \mathsf{target}(\zeta_{n+1})$. **BSeq** denotes the set of minimal bounce sequences. We refer to the set of bounce sequences *resolving* the objective P as $\mathbf{BSeq}(P)$:

$$\mathbf{BSeq}(a_i \Rsh^* a_j) \triangleq \{\zeta \in \mathbf{BSeq} \mid \mathsf{target}(\zeta_1) = a_i \wedge \mathsf{bounce}(\zeta_{|\zeta|}) = a_j \\ \wedge \forall m, n \in \mathbb{I}^\zeta, n > m, \mathsf{bounce}(\zeta_n) \neq \mathsf{target}(\zeta_m)\} \ .$$

Therefore, $\mathbf{BSeq}(a_i \Rsh^* a_i) = \{\varepsilon\}$; and $\mathbf{BSeq}(a_i \Rsh^* a_j) = \emptyset$ if there is no possibility to reach a_j from a_i.

EXAMPLE. Given the AAN of Fig. 1, $\mathbf{BSeq}(c_0 \Rsh^* c_1) = \{\{a_1, b_1\} \to c_0 \Rsh c_1\}$; and $\mathbf{BSeq}(a_0 \Rsh^* a_1) = \{\{b_0\} \to a_0 \Rsh a_1\}$.

From Def. 6, we can derive that any scenario matching with an objective includes all the actions of one of the bounce sequences of the objective (Lemma 2).

Lemma 2. *Given a state $s \in \mathcal{L}$ and an objective $a_i \Rsh^* a_j$ with $s[a] = a_i$, $\forall \delta \in \gamma_s(a_i \Rsh^* a_j)$, $\exists \zeta \in \mathbf{BSeq}(a_i \Rsh^* a_j)$ such that $\exists \phi : \mathbb{I}^\zeta \to \mathbb{I}^\delta$ with $\forall n, m \in \mathbb{I}^\zeta, n < m \Leftrightarrow \phi(n) < \phi(m)$ and $\forall n \in \mathbb{I}^\zeta$, $\zeta_n = \delta_{\phi(n)}$.*

3.1.3. Local Causality Graph

The previous subsections introduced the necessary definitions to build the so-called Local Causality Graph (LCG). This graph is denoted $\mathcal{B}_s^u$ where u is the local state to be reached from the initial state s. It will be the basis of our static analysis, as it represents the causality links between the different objectives involved in the solving of the reachability of u. This LCG is built recursively by considering some required local states (e.g., u), linking them to objectives (e.g., $t \Rsh^* u$, if $t \in s$), and locally refining these objectives in order to include new required local states from other automata. An example of LCG is depicted in Fig. 2.

Technically, one can iteratively refine a given objective into (possibly several) objective sequences extracted from its bounce sequences. Indeed, the refinement of an objective P with $\zeta \in \mathbf{BSeq}(P)$ consists in prepending to P the objectives leading to the activation of all hitters of the actions in ζ, in the same sequential order. The generalization of this refinement to an objective sequence would naturally require to consider all possible interleaving between the refinements [14].

For example, given the AAN of Fig. 1, in the state $\langle a_0, b_0, c_0\rangle$, the objective $a_0 \Rsh^* a_1$ can be refined in the objective sequence $b_0 \Rsh^* b_0 :: a_0 \Rsh^* a_1$ and the objective $c_0 \Rsh^* c_1$ can be refined in the objective sequence $a_0 \Rsh^* a_1 :: b_0 \Rsh^* b_1 :: c_0 \Rsh^* c_1$ or $b_0 \Rsh^* b_1 :: a_0 \Rsh^* a_1 :: c_0 \Rsh^* c_1$ (which can then be further refined).

Formally, a Local Causality Graph (LCG) is a digraph where $V_s^u \subseteq \mathbf{LS} \cup \mathbf{Obj} \cup \mathbf{Sol} \cup \mathbf{Sync}$ is the set of vertices, with $\mathbf{Sync} = \wp(\mathbf{LS})$ and $\mathbf{Sol} = \mathbf{Obj} \times \wp(\mathbf{Sync})$, and $E_s^u \subseteq V_s^u \times V_s^u$ is the set of oriented edges (Def. 7). A node in $V_s^u \cap \mathbf{Obj}$ represents an objective to refine. Such a node is linked to nodes in $\mathbf{Sol}$ which represent sets of hitter sets extracted from a bounce sequence of the objective (that is, one node for each bounce sequence in $\mathbf{BSeq}(P)$); thus, given an objective P, if $\zeta \in \mathbf{BSeq}(P)$ and $\zeta^\wedge$ is the set of all hitters of ζ, then P is linked to a node $\langle P, \zeta^\wedge\rangle \in \mathbf{Sol}$ (Def. 7-3). For each hitter set $ps \in \zeta^\wedge$, the node $\langle P, \zeta^\wedge\rangle \in V_s^u \cap \mathbf{Sol}$ is linked to the node $ps \in \mathbf{Sync}$ (Def. 7-4). Then, a node $ps \in V_s^u \cap \mathbf{Sync}$ is linked to each local state $a_i \in ps$ is contains (Def. 7-5). A local state node $a_i \in V_s^u \cap \mathbf{LS}$ is linked to each objective $a_j \Rsh^* a_i \in \mathbf{Obj}$, where a_j is either in the initial state s or is a local state node in the LCG (Def. 7-2). Finally, given an objective node $a_j \Rsh^* a_i \in V_s^u \cap \mathbf{Obj}$, if a local state $a_k \neq a_i$ is in its successors (which is given by $\mathsf{conn}_{(V_s^u, E_s^u)}(a_j \Rsh^* a_i)$), then $a_j \Rsh^* a_i$ is linked to $a_k \Rsh^* a_i$ in the LCG (Def. 7-6).

Definition 7. Given a state $s \in \mathcal{L}$ and a local state $u \in \mathbf{LS}$, the Local Causality Graph (LCG) for reachability under-approximation $\mathcal{B}_s^u \triangleq (V_s^u, E_s^u)$, with $V_s^u \subseteq \mathbf{LS} \cup \mathbf{Obj} \cup \mathbf{Sol} \cup \mathbf{Sync}$ and $E_s^u \subseteq V_s^u \times V_s^u$, is the smallest graph such that:

1. $u \in V_s^u$
2. $a_i \in V_s^u \cap \mathbf{LS} \Leftrightarrow \{(a_i, a_j \Rsh^* a_i) \mid a_j \neq u \wedge (a_j \in s \vee a_j \in V_s^u \cap \mathbf{LS})\} \subseteq E_s^u$
3. $P \in V_s^u \cap \mathbf{Obj} \Leftrightarrow \{(P, \langle P, \zeta^\wedge\rangle) \mid \zeta \in \mathbf{BSeq}(P)\} \subseteq E_s^u$
4. $\langle P, pps\rangle \in V_s^u \cap \mathbf{Sol} \Leftrightarrow \{(\langle P, pps\rangle, ps) \mid ps \in pps\} \subseteq E_s^u$
5. $ps \in V_s^u \cap \mathbf{Sync} \Leftrightarrow \{(ps, a_i) \mid a_i \in ps\} \subseteq E_s^u$
6. $a_i \Rsh^* a_j \in V_s^u \cap \mathbf{Obj} \Rightarrow \{(a_i \Rsh^* a_j, a_k \Rsh^* a_j) \mid a_k \neq a_j,$
   $a_k \in \mathsf{conn}_{(V_s^u, E_s^u)}(\langle a_i \Rsh^* a_j, \zeta^\wedge\rangle),$
   $\zeta \in \mathbf{BSeq}(a_i \Rsh^* a_j)\} \subseteq E_s^u$

with $\zeta^\wedge \triangleq \{\mathsf{hitters}(\zeta_n) \mid n \in \mathbb{I}^\zeta\}$.

The definition of an LCG tackles two cases where the target of an objective may be changed. First, the addition of objectives based on the local states already mentioned anywhere in the LCG, as performed in Def. 7-2, ensures to take into account the possible changes in the active local states made by other objectives. We thus try to ensure that a required local state is reachable even when starting from a local state of the same automaton that is not in the initial state, but that may be active at some point of the solving. Second, Def. 7-6, allows to "re-target" objectives whose own solving changes their target. This happens when playing the actions that activate the local states required to solve an objective P also changes the active local state of automaton $\Sigma(P)$. In this case, the initial objective P is re-targeted to another objective $p \Rsh^* \mathsf{bounce}(P)$,

where p is the new active local state in $\Sigma(P)$. Linking to all objectives of this kind ensures that all possible disturbances are taken into account.

Finally, we note that the LCG of Def. 7 contains synchronizing nodes (**Sync**) that allow to express the need for several local states simultaneously in order to play a given action. This is the main difference regarding [14, 16] which tackled with Process Hitting, whose actions are limited to at most one hitter and thus did not require this kind of synchronization.

3.2. Sufficient condition for reachability of a local state

Given a state $s \in \mathcal{L}$ and a local state $u \in \mathbf{LS}$, the LCG $\mathcal{B}_s^u$ contains a set of objectives that can be used to build a concrete scenario reaching u. Because we are focused on sufficient conditions for reachability, we do not require that all possible scenarios can be derived from the LCG.

In this section, we prove that if the LCG has no cycle, all its nodes in **Obj** have at least one child, and all its nodes in **Sync** satisfy a particular criteria, so-called independence, then there exists a scenario that reaches u from state s (Theorem 1).

A node $ps \in \mathbf{Sync}$ of the LCG is *independent* (Def. 8) if for each local state $a_i \in ps$, none of the other local states $b_j \in ps$ have in their successors a local state of automaton a but that is different than a_i. This criteria ensures that once a local state in ps has been reached, reaching another local state in ps should not impact the first.

Definition 8 (Independent synchronizations). In a LCG $\mathcal{B}_s^u = (V_s^u, E_s^u)$, a node $ps \in V_s^u \cap \mathbf{Sync}$ is *independent* if and only if for each $a_i \in ps$, for each $b_j \in ps, b_j \neq a_i$, $a_k \in \mathsf{conn}_{\mathcal{B}_s^u}(b_j) \cap \mathbf{LS} \Rightarrow a_k = a_i$.

The intuition of the under-approximation of Theorem 1 is the following. Given the initial state s, we recursively refine the initial objective (which is: reaching u from the initial state) according to its children. As the LCG is acyclic, by hypothesis, such a recursion always terminates, and as all the objective nodes have at least one child, it never gets stuck. The refinement of an objective node $a_i \Rsh^* a_j$ acts as follows: if the node has another objective node $a_k \Rsh^* a_j$ as child, we first refine the objective $a_i \Rsh^* a_k$ (which, by construction, is necessarily in the LCG) and then we refine the objective $a_k \Rsh^* a_j$. If the objective node has only successors in **Sol** (bounce sequences), one is picked arbitrarily. If $\langle P, pps \rangle$ is the chosen node, by construction there exists $\zeta \in \mathbf{BSeq}(P)$ such that $\zeta^\wedge = pps \in \wp(\mathbf{Sync})$. If $\zeta = \varepsilon$ (for instance in the case where $a_i = a_j$), the recursion stops and we continue to the next stage. Otherwise, for each $n \in \mathbb{I}^\zeta$, for each $b_i \in \mathsf{hitters}(\zeta_n)$, we refine the objective $b_j \Rsh^* b_i$, where b_j is the state of b in the current state. By induction, $b_j \Rsh^* b_i$ is a child of b_i in the LCG of Def. 7. Thus, we know that the current state of b is b_i. After having repeated this procedure for each $b_i \in \mathsf{hitters}(\zeta_n)$, because all the nodes in **Sync**, and in particular $\mathsf{hitters}(\zeta_n)$, are independent, we know that all the local states in $\mathsf{hitters}(\zeta_n)$ are in the current state. In addition, we know that the state of automaton a has remained unchanged, otherwise $a_i \Rsh^* a_j$ would have an objective

child. Hence, the action ζ_n is playable in the current state. We can thus apply this action, modifying the state of a to a_j and continue to the next stage. In the end, this recursive procedures builds a scenario from s to a state containing u.

Theorem 1 (Under-approximation). *Given an AAN $(\Sigma; \mathcal{L}; \mathcal{H})$, a state s and local state u, if the LCG $\mathcal{B}_s^u$ contains no cycle, all nodes in* **Obj** *have at least one child, and all nodes in* **Sync** *are independent, then there exists a scenario $\delta \in$ **Sce** such as $u \in s \cdot \delta$.*

Regarding the complexity of the method, computing the LCG is polynomial in the number of automata in $\mathcal{A}$ and exponential in the number of local states in one automaton. Checking the properties allowing to apply Theorem 1 is polynomial in the size of the graph. Therefore, the building and checking processes can be considered as polynomial in the size of the AAN, provided that each automaton only contains a few local states. We note that this is particularly true for biological models, where each component usually contains a limited number of expression levels.

We note furthermore that the method does not require any completeness of the bounce sequences **BSeq**. Therefore, in order to reduce the number of bounce sequences to consider, and potentially remove cycles and non-satisfying nodes, one can consider only a sub-set of bounce sequences (and in particular: a unique bounce sequence) for each objective. However, such an approach requires to enumerate all possible combinations of bounce sequences subsets, hence being exponential in the number of considered bounce sequences.

Example. We consider in this example the AAN of Fig. 1, and the initial state $\langle a_1, b_0, c_0 \rangle$ that is also represented. The under-approximation given in Theorem 1 concludes that a_1 is reachable from this initial state, as well as b_1. Nevertheless, it does not conclude regarding the reachability of c_1. This is due to the fact that the node $\{a_1, b_1\} \in V_s^u \cap$ **Sync** is not independent because of its successor a_0 (and b_0) as we can see in the LCG of Fig. 2. (However, from the inconclusiveness of Theorem 1, one cannot conclude about the unreachability of c_1. Such analysis should be driven for instance with over-approximation methods developed in [14].)

This result is new compared to the method proposed in [14]. Indeed, the representation based on the Process Hitting that was proposed in this paper only allowed to represent "over-approximated" Boolean gates with the use of so-called cooperative sorts. This especially did not allow to model the fact that a_1 and b_1 could not be activated in the same state, but only in successive states. Thus, when using Process Hitting, c_1 was indeed reachable, contrary to the behaviour expected from an accurate Boolean gate.

Finally, we note however that if actions $\{a_0\} \to b_0 \nearrow b_1$ and $\{b_0\} \to a_0 \nearrow a_1$ are replaced respectively by $\emptyset \to a_0 \nearrow a_1$ and $\emptyset \to b_0 \nearrow b_1$, then the resulting saturated graph of local causality changes, and Theorem 1 concludes that c_1 is reachable from $\langle a_1, b_0, c_0 \rangle$. The reader can also refer to Subsection 5.1 for a detailed conclusive example.

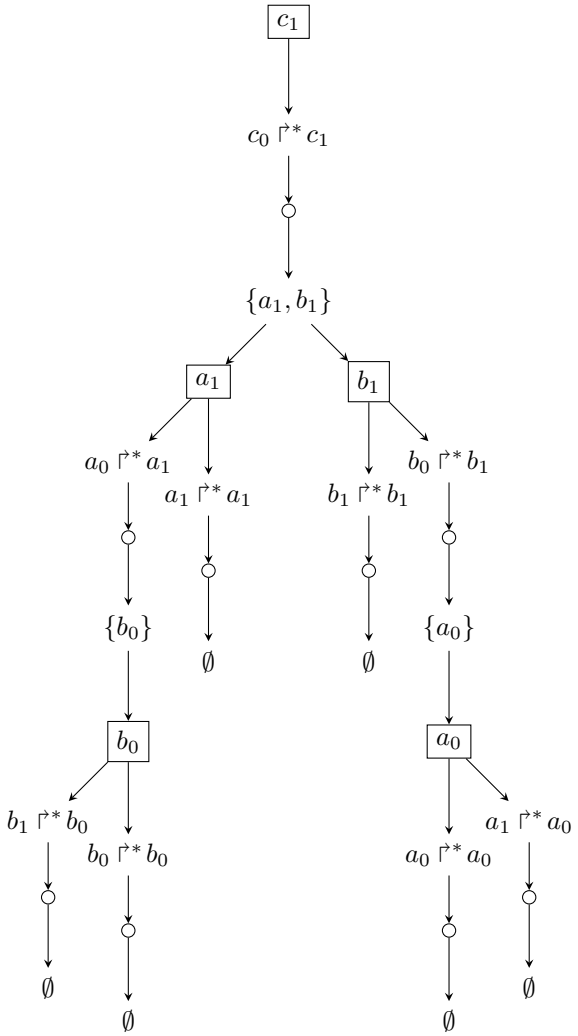

Figure 2: The local causality graph $\mathcal{B}_s^u$ on the AAN in Fig. 1 for the reachability of $u = c_1$ from the initial state $s = \langle a_1, b_0, c_0 \rangle$. Elements in **LS** are represented by rectangular nodes, elements in **Sol** are represented by small circles, and elements in **Sync** and **Obj** are the remaining borderless nodes. Theorem 1 is inconclusive on this example as node $\{a_1, b_1\} \in$ **Sync** is not independent (see Def. 8). Indeed, a_0 is a successor of b_1, but $a_0 \neq a_1$ (and the same also stands for b_0, which is a successor of a_1).

3.3. Extraction of a Scenario

This section gives a recursive method to find a scenario that concretizes the reachability of a given local state $u \in \mathbf{LS}$, from a given initial state $s_0 \in \mathcal{L}$, provided that Theorem 1 answered positively on this couple of inputs. The algorithm proposed in what follows relies on a traversal of the Local Causality Graph $\mathcal{B}^u_{s_0}$ that has been used for this conclusion. The correctness of this extraction can be demonstrated in the same fashion as Theorem 1.

This algorithm consists in visiting all required nodes in a given order, to build a sequence of actions that concretizes the objective. The actions to perform on a given node, listed below, depend on the current state, the type of node, the markers on this node and the state of the traversal which can be either "descending" (D) or "ascending" (A). Nodes in **Sol** can be marked with a sequence of actions and nodes in **Sync** can be marked with a set of local states. When ascending, the traversal always ascend into the node it previously descended from (in order to cover the same path backwards). The traversal starts in node u in descending mode, the starting state is $s = s_0$ and the initial output is the empty sequence ε.

- In a node $a_k \in \mathbf{LS}$:
  - D) Descend in the node $s[a] \Rsh^* a_k \in \mathbf{Obj}$.
  - A) Ascend in the parent **Sync** node, if any, and remove the element a_k from its marking set. If the node has no **Sync** parent, the traversal is complete and the output of the algorithm is the current output.
- In a node $a_j \Rsh^* a_k \in \mathbf{Obj}$:
  - D) Descend in an arbitrarily chosen node $\langle P, pps\rangle \in \mathbf{Sol}$, and mark it with an arbitrarily chosen sequence $\zeta \in \mathbf{BSeq}(P)$ so that $\zeta^\wedge = pps$.
  - A) Ascend in the parent **LS** or **Obj** node.
- In a node $\langle P, pps\rangle \in \mathbf{Sol}$ with the current marking ζ:
  - If $\zeta \neq \varepsilon$, descend in the node $ps \in \mathbf{Sync}$ so that $ps = \mathsf{hitters}(\zeta_1)$ and mark it with the set ps.
  - If $\zeta = \varepsilon$, ascend in the parent node $P \in \mathbf{Obj}$ and carry on ascending.
- In a **Sync** node with the current marking m:
  - If $m \neq \emptyset$, descend in a child node $a_k \in \mathbf{LS}$ arbitrarily chosen, so that $a_k \in m$ and carry on descending.
  - If $m = \emptyset$, ascend in the parent **Sol** node. Let ζ be the current marking of this node. If ζ_1 is playable in s: append ζ_1 to the output, change the current state to $s \cdot \zeta_1$ and mark this node with $\zeta_{2..|\zeta|}$. If ζ_1 is not playable (which means that $\mathsf{target}(\zeta_1)$ has changed), then go to the node $s[\Sigma(\mathsf{bounce}(\zeta_1))] \Rsh^* \mathsf{bounce}(\zeta_1) \in \mathbf{Obj}$ and start descending.

The execution of such a traversal outputs a scenario from s_0. In the following, we denote by $\Delta(\mathcal{B}^u_{s_0})$ the set of all scenarios extracted from the local causality graph $\mathcal{B}^u_{s_0}$. Indeed, several scenarios may exist for a same graph, generated by the arbitrary choices that exist in this algorithm. An example of such a traversal is given on a small example in Subsection 5.1.

3.4. Addressing Sequential and Sub-state Reachability

In this section, we briefly discuss how the presented static analysis for local state reachability properties can be extended to sequential (sub)states reachability properties.

Sub-state reachability

Given an AAN $\mathcal{A} = (\Sigma, \mathcal{L}, \mathcal{H})$, an initial state $s \in \mathcal{L}$ and a sub-set of local states $\tau \subseteq \mathbf{LS}$ so that $\forall a \in \Sigma, |\tau \cap \mathcal{L}_a| \leq 1$, the reachability of a state containing all local states of τ from the initial state $s \in \mathcal{L}$ can be tackled by our method. Indeed, consider the AAN $\mathcal{A}' = (\Sigma', \mathcal{L}', \mathcal{H}')$ with: $\Sigma' = \Sigma \cup \{\sigma\}$, $\mathcal{L}' = \mathcal{L} \times \mathcal{L}_\sigma$, where $\mathcal{L}_\sigma = \{\sigma_0, \sigma_1\}$, and $\mathcal{H}' = \mathcal{H} \cup \{\tau \to \sigma_0 \Rsh \sigma_1\}$. Obviously, the reachability in $\mathcal{A}$ of any state $s' \in \mathcal{L}$ such that $\tau \subset s'$ from the initial state s is equivalent to the reachability in $\mathcal{A}'$ of the local state σ_1 from state $s \times \{\sigma_0\}$.

Such analysis was not possible with the Process Hitting framework, because of the lack of the notion of simultaneity for more than two components.

Sequential reachability

Given a sequence of local states to reach (e.g., reach a_i, then b_j, etc.), one can use the same approach as for sub-state reachability by introducing a new automaton σ having $n + 1$ local states, where n is the size of the sequence of reachability, and n actions making it bounce gradually from σ_0 to σ_n with hitters corresponding the successive local states (e.g., $\{a_i\} \to \sigma_0 \Rsh \sigma_1$, $\{b_j\} \to \sigma_1 \Rsh \sigma_2$, etc.).

An alternative approach is to use the extraction of a scenario of Subsection 3.3: given an initial state s_0, and a first local state reachability property u_1, one can compute a possible scenario $\delta \in \Delta(\mathcal{B}^u_{s_0})$ witnessing this reachability; then the next local state reachability properties (u_2, u_3, etc.) are computed from the state $s_0 \cdot \delta$ that outcomes from the latter scenario.

4. Asynchronous Automata Networks with classes of priorities

In this section, we define the notion of AANs with classes of priorities, and give a transformation from these into AANs without priorities, as defined in Sect. 2.

The idea behind AANs with classes of priorities (Def. 9) is to split the set of actions into several subsets assigned to priorities, and to constrain the behaviour of the model to make any action unplayable until no other action of higher priority is playable (Def. 10). Such a framework allows to model preemptions between sets of actions, which can be helpful to abstract time or duration properties under certain conditions.

Definition 9 (AAN with k classes of priorities). If $k \in \mathbb{N}^*$, an *Asynchronous Automata Network with k classes of priorities* (AANk) is a triplet $\mathcal{A} = (\Sigma; \mathcal{L}; \mathcal{H}^{\langle k \rangle})$, where $\mathcal{A}^{\langle k \rangle} = (\mathcal{H}^{(1)} \ldots; \mathcal{H}^{(k)})$, and:

- $\Sigma \triangleq \{a, b, \ldots, z\}$ is the finite set of *automata*;
- $\mathcal{L} \triangleq \underset{a \in \Sigma}{\times} \mathcal{L}_a$ is the finite set of (global) *states*, where $\mathcal{L}_a = \{a_0, \ldots, a_{l_a}\}$ is the finite set of *local states* of automaton $a \in \Sigma$, with $l_a \in \mathbb{N}^*$, and so that: $\forall (a_i; b_j) \in \mathcal{L}_a \times \mathcal{L}_b, a \neq b \Rightarrow a_i \neq b_j$;
- $\forall n \in [\![1; k]\!], \mathcal{H}^{(n)} \triangleq \{A \to b_j \Rsh b_k \mid b \in \Sigma \wedge (b_j; b_k) \in \mathcal{L}_b \times \mathcal{L}_b \wedge b_j \neq b_k \wedge \forall a \in \Sigma, |A \cap \mathcal{L}_a| \leq 1 \wedge A \cap \mathcal{L}_b = \emptyset\}$ is the finite set of *actions of priority n*.

Notations. We use the same notations as those defined in Sect. 2, when applicable. Furthermore, we denote by $\mathcal{H} = \bigcup_{n \in [\![1;k]\!]} \mathcal{H}^{(n)}$ the set of all actions and, for all $h \in \mathcal{H}$, by $\mathsf{prio}(h) = \min\{n \in [\![1; k]\!] \mid h \in \mathcal{H}^{(n)}\}$ the priority of action h.

Definition 10 (Semantics of an AANk ($\to_{\mathcal{A}}$)). An action $h = A \to b_j \Rsh b_k \in \mathcal{H}^{(n)}$ of priority n is *playable* in $s \in \mathcal{L}$ if and only if $A \subseteq s$, $s[b] = b_j$ and $\forall m < n, \forall g \in \mathcal{H}^{(m)}, \neg(\mathsf{hitters}(g) \subseteq s \wedge \mathsf{target}(g) \in s)$. In such a case, $(s \cdot h)$ stands for the state resulting from the play of the action h in s, which is defined by: $(s \cdot h)[b] = b_k$ and $\forall a \in \Sigma, a \neq b, (s \cdot h)[a] = s[a]$. Moreover, we denote: $s \to_{\mathcal{A}} (s \cdot h)$.

The translation given in the following relies on the notion of *sub-state* (Def. 11), which is a set of local states containing at most one local state of each automata. Thus, a sub-state can be considered as a partial state.

Definition 11 (Sub-state ($\mathcal{L}^\diamond$)). If $S \subseteq \Sigma$ is a set of automata, a sub-state on S is an element of: $\mathcal{L}_S^\diamond \triangleq \{\widetilde{\rho} \subseteq \mathbf{LS} \mid \rho \in \underset{a \in S}{\times} \mathcal{L}_a\}$ (where the notation $\widetilde{\rho}$ represents the set of components of the Cartesian product ρ, as defined on page 4). The set of all sub-states is denoted by: $\mathcal{L}^\diamond \triangleq \bigcup_{S \in \wp(\Sigma)} \mathcal{L}_S^\diamond$. Furthermore, we recall the notation from Sect. 2, if $\sigma \in \mathcal{L}^\diamond$ and $s \in \mathcal{L}$:

$$\sigma \subseteq s \Leftrightarrow \forall a_i \in \sigma, s[a] = a_i$$

We consider in the following an AANk: $\overline{\mathcal{A}} = (\overline{\Sigma}; \overline{\mathcal{L}}; \overline{\mathcal{H}}^{\langle k \rangle})$ with $k \in \mathbb{N}$, $k > 1$. The aim of the rest this section is to propose a translation of $\overline{\mathcal{A}}$ into an AAN with 1 class of priority $\mathcal{A} = (\Sigma; \mathcal{L}; \mathcal{H}^{\langle 1 \rangle})$ called *flattening*, which is bisimilar. As an AAN with 1 class of priority is equivalent to a regular AAN without priorities, such a translation is particularly useful to be able to study the dynamics of any kind of AAN with priorities by using the static analysis developed in Sect. 3.

The translation of an AANk into an AAN is based on the notion of *playability property* (Def. 12) which is a Boolean formula where the atoms are local states of $\overline{\mathcal{A}}$.

Definition 12 (Playability property language (F)). A *playability property* is an element of the language F inductively defined by:

- $\top$ and $\bot$ belong to F;
- if $a \in \overline{\Sigma}$ and $a_i \in \overline{\mathcal{L}}_a$, then $a_i \in F$ and we call it an *atom*;
- if $P \in F$ and $Q \in F$, then $\neg P \in F$, $P \wedge Q \in F$ and $P \vee Q \in F$.

If $P \in F$ is a playability property and $\sigma \in \overline{\mathcal{L}}^{\diamond}$ is a sub-state of $\overline{\mathcal{A}}$, we note $[P](\sigma)$ the *evaluation* of P in σ in a three-valued Kleene logic (*true*, *undecided* or *false*) which is given by:

- if $P = a_i \in \overline{\mathcal{L}}_a$ is an atom, then $[a_i](\sigma)$ is $\begin{cases} \text{true} & \text{if } a_i \in \sigma \\ \text{undecided} & \text{if } \sigma \cap \overline{\mathcal{L}}_a = \emptyset \\ \text{false} & \text{otherwise} \end{cases}$;
- if P is not an atom, then $[P](\sigma)$ is evaluated in σ with the classical semantics for the logic operators $\top$, $\bot$, $\neg$, $\wedge$ and $\vee$, which is recalled in Fig. 3.

Furthermore, in what follows, we will use $[F(h)](s)$ as a shorthand for $[F(h)](\widetilde{s})$.

For all playability properties $P, Q \in F$ that are not atoms, and for all sub-state $\sigma \in \overline{\mathcal{L}}^{\diamond}$:

- $[\top](\sigma)$ is always true;
- $[\bot](\sigma)$ is always false;
- $[\neg P](\sigma)$ is $\begin{cases} \text{true} & \text{if } [P](\sigma) \text{ is false} \\ \text{false} & \text{if } [P](\sigma) \text{ is true} \\ \text{undecided} & \text{if } [P](\sigma) \text{ is undecided} \end{cases}$;
- $[P \wedge Q](\sigma)$ is $\begin{cases} \text{true} & \text{if both } [P](\sigma) \text{ and } [Q](\sigma) \text{ are true} \\ \text{false} & \text{if } [P](\sigma) \text{ is false or } [Q](\sigma) \text{ is false} \\ \text{undecided} & \text{otherwise} \end{cases}$;
- $[P \vee Q](\sigma)$ is $\begin{cases} \text{true} & \text{if } [P](\sigma) \text{ is true or } [Q](\sigma) \text{ is true} \\ \text{false} & \text{if both } [P](\sigma) \text{ and } [Q](\sigma) \text{ are false} \\ \text{undecided} & \text{otherwise} \end{cases}$.

Figure 3: Explicit semantics of the evaluation of playability properties in the three-valued Kleene logic.

Because we only use classical logic operators, the formulas of Boolean logic on distributivity, associativity and commutativity can be used, together with

De Morgan's laws on negation. We also have the following property for the negation of an atom:

$$\forall a \in \overline{\Sigma}, \forall a_i \in \overline{\mathcal{L}}_a, \neg a_i \iff \bigvee_{\substack{a_j \in \overline{\mathcal{L}}_a \\ a_j \neq a_i}} a_j$$

Indeed, if a local state is not active in a state, this means that another local state of the same automaton is active. Moreover, provided that in what follows we are only interested in properties that are true and thus we make no distinction between false and undecided results, playability properties can be simplified with the following result:

$$\forall a \in \overline{\Sigma}, \forall a_i, a_j \in \overline{\mathcal{L}}_a, a_i \neq a_j \implies a_i \wedge a_j \equiv \bot$$

because two different local states can never be active simultaneously.

In Def. 13, we define the operator F which characterises the playability of an action given the semantics of AANks (see Def. 10). This operator simply states that the hitters of an action have to be active, and every other action of higher priority must not be playable.

Definition 13 (Playability property operator ($F : \mathcal{H} \to F$)). For all $h = A \to b_j \uparrow b_m \in \overline{\mathcal{H}}$, we define:

$$F(h) \equiv b_j \wedge \left(\bigwedge_{a_i \in A} a_i \right) \wedge \left(\bigwedge_{\substack{g \in \overline{\mathcal{H}}^{(n)} \\ 1 \leq n < \mathsf{prio}(h)}} \neg \left(\mathsf{target}(g) \wedge \left(\bigwedge_{c_l \in \mathsf{hitters}(g)} c_l \right) \right) \right)$$

By construction of this operator and given the dynamics of a AANk, an action h is playable in a state $s \in \overline{\mathcal{L}}$ if and only if $[F(h)](s)$ is true.

Because we only use classical logic operators, we can compute the Disjunctive Normal Form (DNF) of any playability property. For any action $h \in \overline{\mathcal{H}}$, this DNF takes the form:

$$F(h) \equiv \bigvee_{i \in [\![1; n^h]\!]} \left(\bigwedge_{j \in [\![1; m^{h,i}]\!]} p_{i,j} \right)$$

where $n^h \in \mathbb{N}$ and $\forall i \in [\![1; n^h]\!], m^{h,i} \in \mathbb{N}^*$. If $n^h = 0$, then $F(h) \equiv \bot$; this means that h can never be played due to preemptions by other actions with higher priorities. If $F(h) \not\equiv \bot$, on the other hand, then in this case $F(h)$ can be seen as a disjunction of n^h smaller playability properties consisting only of conjunctions of atoms. These n^h conjunctions can be translated to as many actions, thus creating a new AAN1. In this case, we denote, for any $i \in [\![1; n^h]\!]$: $\mathsf{dep}^i(h) = \{\Sigma(p_{i,j}) \mid j \in [\![1; m^{h,i}]\!]\}$.

With Lemma 3, we can then characterise the playability of an action in a state only with a sub-state. This sub-state corresponds to one of the conjunctions of its playability property's DNF. Finally, Def. 14 gives the construction

of the flattening of $\overline{\mathcal{A}}$: for each action $h \in \overline{\mathcal{H}}$, several actions $f^{h,i}$ are built to reflect each of the conjunctions in $F(h)$, i.e., for $i \in [\![1; n^h]\!]$. This construction allows to obtain the same dynamics as $\overline{\mathcal{A}}$, as stated by Theorem 2.

Lemma 3. *Let $h \in \overline{\mathcal{H}}$ and $s \in \overline{\mathcal{L}}$; h is playable in s if and only if:*

$$\mathsf{target}(h) \in s \wedge \exists n^h \in \mathbb{N}, \exists \sigma \in \overline{\mathcal{L}}^{\diamond}_{\mathsf{dep}^i(h)}, \sigma \subseteq s \wedge [F(h)](\sigma) \text{ is true } .$$

Proof. ($\Rightarrow$) If h is playable in s, then $\mathsf{target}(h) \in s$ and $[F(h)](s)$ is true. Thus, $F(h) \not\equiv \bot$ and, by property of a DNF, at least one of the n^h conjunctions of $F(h)$ is true in s. Suppose the i^{th} conjunction is true in s, with $i \in [\![1; n^h]\!]$; then we have: $\forall j \in [\![1; m^{h,i}]\!], p_{i,j} \in s$. Let $\sigma \in \overline{\mathcal{L}}^{\diamond}_{\mathsf{dep}^i(h)}$ with $\forall b \in \mathsf{dep}^i(h), \sigma[b] = s[b]$. We immediately have: $\sigma \subseteq s$, and, by construction of $\mathsf{dep}^i(h)$, $[F(h)](\sigma)$ is true.

($\Leftarrow$) $[F(h)](\sigma)$ is true, and therefore $[F(h)](s)$ is true; as $\mathsf{target}(h) \in s$, h is playable in s. □

Definition 14 (Flattening (flat)). If $k \in \mathbb{N}$, $k > 1$ and $\overline{\mathcal{A}} = (\overline{\Sigma}; \overline{\mathcal{L}}; \overline{\mathcal{H}}^{\langle k \rangle})$ is an AANk, we denote by $\mathsf{flat}(\overline{\Sigma}; \overline{\mathcal{L}}; \overline{\mathcal{H}}^{\langle k \rangle}) = (\Sigma; \mathcal{L}; \mathcal{H})$ the *flattening* of $\overline{\mathcal{A}}$, where:

- $\Sigma = \overline{\Sigma}$;
- $\mathcal{L} = \overline{\mathcal{L}}$;
- $\mathcal{H} = \{(\sigma \setminus \{\mathsf{target}(h)\}) \to \mathsf{target}(h) \Rsh \mathsf{bounce}(h) \mid h \in \overline{\mathcal{H}} \wedge n^h \geq 1 \wedge i \in [\![1; n^h]\!] \wedge \sigma \in \overline{\mathcal{L}}^{\diamond}_{\mathsf{dep}^i(h)} \wedge [F(h)](\sigma) \text{ is true}\}$.

We note that the set of global states of an AANk and the set of global states of its flattening are the same.

Theorem 2 ($(\overline{\Sigma}; \overline{\mathcal{L}}; \overline{\mathcal{H}}^{\langle k \rangle}) \approx \mathsf{flat}(\overline{\Sigma}; \overline{\mathcal{L}}; \overline{\mathcal{H}}^{\langle k \rangle})$). *If $\overline{\mathcal{A}} = (\overline{\Sigma}; \overline{\mathcal{L}}; \overline{\mathcal{H}}^{\langle k \rangle})$ is an AANk and $\mathcal{A} = \mathsf{flat}(\overline{\Sigma}; \overline{\mathcal{L}}; \overline{\mathcal{H}}^{\langle k \rangle}) = (\Sigma; \mathcal{L}; \mathcal{H})$ is its flattening, then:*

$$\forall s, s' \in \mathcal{L}, s \to_{\overline{\mathcal{A}}} s' \iff s \to_{\mathcal{A}} s'$$

Proof. By definition of flat. □

We showed in this subsection that it is possible to model any AANk as an AAN (or, equivalently, as an AAN1). This translation thus extends the applicability of the static analysis developed in Sect. 3 to any AANk, with $k \in \mathbb{N}^*$. Moreover, it allows to represent any Process Hitting model with classes of priorities [16] under the form of an AAN (or, equivalently, of a Process Hitting model with 2 classes of priorities). The translation given in this section is exponential in the number of actions of higher priority for each action.

5. Biological Examples

This section aims at giving application examples of the static analysis method that we developed in Sect. 3. In Subsection 5.1, we apply our method to a small model of the metazoan segmentation process, and demonstrate how classes of priorities help in the modelling process, and how the flattening of Sect. 4 can be used in such a case. In Subsection 5.2, we apply our method to two large-scale models in order to show the scalability of our method.

5.1. Under-approximation of a Model with Priorities: Metazoan Segmentation

We give here a detailed example of the use of classes of priorities in order to model a system with temporal constraints. This model also allows us to give a detailed example of the application of the sequential under-approximation proposed in Subsection 3.4, which consists of several applications of the method developed in Subsection 3.2. For this, we first have to use the flattening method presented in Sect. 4 because the considered model contains 2 classes of priorities.

Let us consider a model of metazoan segmentation inspired from a first translation to Process Hitting model given in [6]. This model was originally established in silico in [18] in a differential equations framework. It is composed of a wavefront gene f that activates the gap-gene a whose products are responsible for stripes formation. Gene f also activates a gene c whose products represses the gene a. The auto-inhibition of c generalises a chain of repressors on a. The auto-inhibition of f, which normally terminates the stripes formation in the original model, has been removed in order to focus on the stationary dynamics of the model.

The actions of the original model are split into 2 classes of priorities, as represented in Fig. 4:

$$\overline{\mathcal{H}}^{(1)} = \{ \quad \{c_1\} \to a_1 \restriction\!\!\!\nearrow a_0 \quad , \quad \{f_1, c_0\} \to a_0 \restriction\!\!\!\nearrow a_1 \quad \}$$
$$\overline{\mathcal{H}}^{(2)} = \{ \quad \emptyset \to c_1 \restriction\!\!\!\nearrow c_0 \quad , \quad \{f_1\} \to c_0 \restriction\!\!\!\nearrow c_1 \quad \}$$

Indeed, without this use of priorities, some unwanted behaviours emerge, allowing the formation of irregular stripes. In order to fix this, a high priority is affected to the actions hitting a and a low priority to the actions hitting c, in order to model the fact that the switch of c has to be immediately followed by a switch of a. This forces the evolution of genes a and c to alternate in order not to miss a stripe; a and c thus have intertwined oscillations. We can thus consider that these two classes of priorities are derived from known relative reaction rates, the evolution of the clock c being slower and regular, while the evolution of a has to follow the changes of c.

Fig. 5 gives the flattening of this model, that is, an AAN with the equivalent dynamics (but only one class of priority). Its actions are:

$$\begin{aligned} \mathcal{H} = \{ \quad & \{c_1\} \to a_1 \restriction\!\!\!\nearrow a_0 \quad , \quad \{f_1, c_0\} \to a_0 \restriction\!\!\!\nearrow a_1 \quad , \\ & \{a_0\} \to c_1 \restriction\!\!\!\nearrow c_0 \quad , \quad \{f_1, a_1\} \to c_0 \restriction\!\!\!\nearrow c_1 \quad \} \end{aligned}$$

We note that the two actions in $\overline{\mathcal{H}}^{(2)}$ have been replaced by the equivalent actions $\{a_0\} \to c_1 \not\Cap c_0$ and $\{f_1, a_1\} \to c_0 \not\Cap c_1$, in order to model the preemption of the actions in $\overline{\mathcal{H}}^{(1)}$.

At this point, the static analysis results presented in Subsection 3.2 can be used to check if the model is functional, i.e., if gene a can oscillate, thus leading to the formation of stripes. Starting from state $s^1 = \langle f_1, a_0, c_0 \rangle$, we thus want to check the reachability of $u^1 = a_1$; then, starting from any new state obtained, we want to check the reachability of $u^2 = a_0$, and finally $u^3 = a_1$ once again to ensure that we entered a cycle. For this, we apply the method proposed in Subsection 3.4: we consider each reachability step independently and we use Theorem 1 three times; the initial states of steps 2 and 3 are computed with the extraction method of Subsection 3.3.

We first build the local causality graph $\mathcal{B}^{u^1}_{s^1}$ related to the reachability of $u^1 = a_1$ from the initial state s^1. This graph is depicted in Fig. 6(left). Theorem 1 allows to conclude that this reachability is true. One can thus extract a concretizing scenario from this local causality graph with a traversal of the graph, as explained in Subsection 3.3. Such a traversal is detailed in Table 1; the scenario extracted from this example consists of only one action: $\{c_0, f_1\} \to a_0 \not\Cap a_1$. Another traversal of the same graph would consist in visiting node f_1 before node c_0 when first descending from node $\{c_0, f_1\}$; however, the same scenario would be extracted. As there exists no other traversal, we thus have in this case: $\Delta(\mathcal{B}^{u^1}_{s^1}) = \{\{c_0, f_1\} \to a_0 \not\Cap a_1\}$.

We denote in the following: $s^2 = \langle f_1, a_0, c_0 \rangle \cdot \{c_0, f_1\} \to a_0 \not\Cap a_1 = \langle f_1, a_1, c_0 \rangle$ the state resulting from the play of the scenario concretizing u^1 in s^1. As explained in Subsection 3.4, one can use this resulting state in order to carry on with another successive reachability, such as the reachability of $u^2 = a_0$ in our case. The local causality graph used to check the reachability of u^2 from s^2 is given in Fig. 6(middle). The same reasoning allows to conclude that this reachability is true, and to extract the following set containing only one concretizing scenario: $\Delta(\mathcal{B}^{u^2}_{s^2}) = \{\{a_1, f_1\} \to c_0 \not\Cap c_1 :: \{c_1\} \to a_1 \not\Cap a_0\}$. We note that once again, two traversals are possible (by visiting node a_1 before or after note f_1) but both output the same scenario, and therefore end up in the same sate $s^3 = \langle f_1, a_0, c_1 \rangle$. Finally, the last local causality graph $\mathcal{B}^{u^3}_{s^3}$, depicted in Fig. 6(right), allows to conclude that this final reachability is true.

In conclusion, by following Subsection 3.4, we showed that it is possible to reach successively a_1, a_0 and a_1 and thus that the AAN of Fig. 5 is functional. This result can be extended to the model of Fig. 4 because they have the same dynamics (given Theorem 2).

5.2. Large-scale Applications

In order to support the scalability and applicability of our under-approximation of reachability, we apply our new approach to the analysis of two large-scale models: a T-cell receptor (TCR) signalling pathway [19] and an epidermal growth factor receptor (EGFR) signalling pathway [20]. These models both

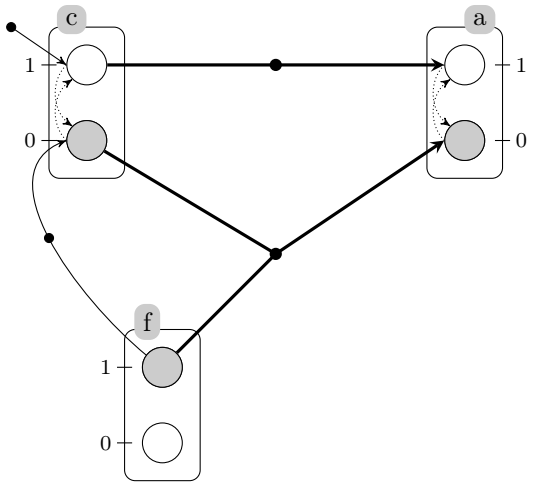

Figure 4: An example of AAN2 modelling the process of metazoan segmentation. Component a models the pigment production, and is influenced by component c that has the role of a clock, while f represents the wavefront propagation. If component a oscillates (that is, its active local state changes regularly) then regular stripes are created on the metazoan. Actions of $\overline{\mathcal{H}}^{(2)}$ (low priority) are represented in thin lines and actions of $\overline{\mathcal{H}}^{(1)}$ (high priority) are in thick lines. The greyed local states represent a possible initial state: $\langle f_1, a_0, c_0 \rangle$.

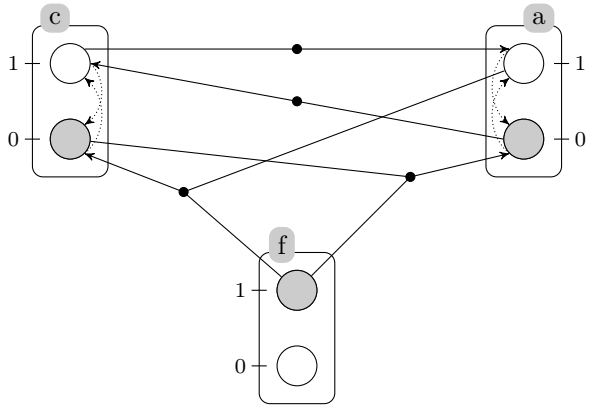

Figure 5: An example of AAN, which is the flattening of the AAN2 in Fig. 4; in other words, this model has exactly the same dynamics as Fig. 4, but its actions make only one class of priority. The greyed local states represent the same initial state: $\langle f_1, a_0, c_0 \rangle$.

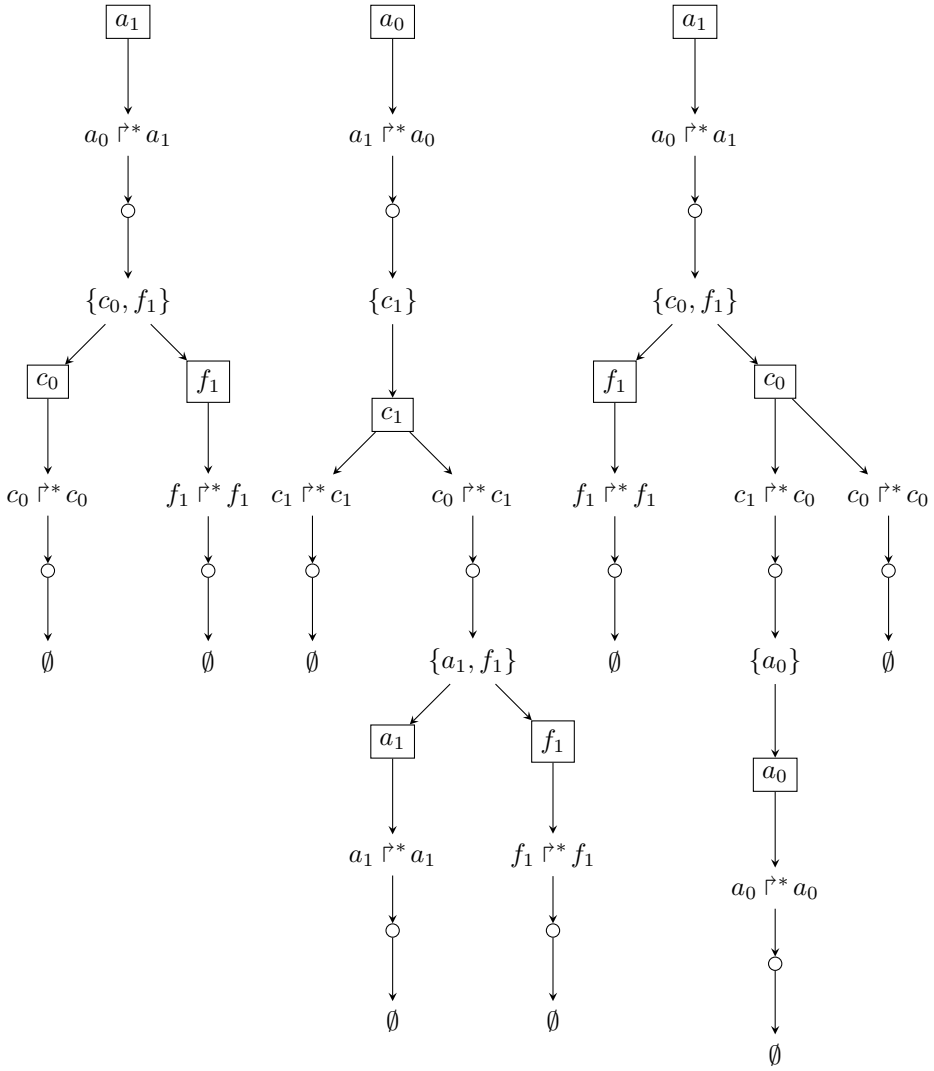

Figure 6: The three successive saturated graphs of local causality of the AAN in Fig. 5 for the successive reachability of a_0, a_1 and a_0 from the initial state $s = \langle f_1, a_0, c_0 \rangle$. The (left) graph allows to check the reachability of a_1 from the initial state s. The (middle) graph is for the reachability of a_0 and the state $\langle f_1, a_1, c_0 \rangle$. The (right) graph is for the last reachability, a_1, and the state $\langle f_1, a_0, c_1 \rangle$.

#	Dir.	Node	Marking	Output	State
1	$\searrow$	a_1			$\langle f_1, a_0, c_0 \rangle$
2	$\searrow$	$a_0 \Rsh^* a_1$			$\langle f_1, a_0, c_0 \rangle$
3	$\searrow$	$\circ$	$\{c_0, f_1\} \to a_0 \Rsh a_1$		$\langle f_1, a_0, c_0 \rangle$
4	$\searrow$	$\{c_0, f_1\}$	$\{c_0, f_1\}$		$\langle f_1, a_0, c_0 \rangle$
5*	$\searrow$	c_0			$\langle f_1, a_0, c_0 \rangle$
6	$\searrow$	$c_0 \Rsh^* c_0$			$\langle f_1, a_0, c_0 \rangle$
7	$\searrow$	$\circ$	ε		$\langle f_1, a_0, c_0 \rangle$
8	$\nearrow$	$c_0 \Rsh^* c_0$			$\langle f_1, a_0, c_0 \rangle$
9	$\nearrow$	c_0			$\langle f_1, a_0, c_0 \rangle$
10	$\nearrow$	$\{c_0, f_1\}$	$\{f_1\}$		$\langle f_1, a_0, c_0 \rangle$
11	$\searrow$	f_1			$\langle f_1, a_0, c_0 \rangle$
12	$\searrow$	$f_1 \Rsh^* f_1$			$\langle f_1, a_0, c_0 \rangle$
13	$\searrow$	$\circ$	ε		$\langle f_1, a_0, c_0 \rangle$
14	$\nearrow$	$f_1 \Rsh^* f_1$			$\langle f_1, a_0, c_0 \rangle$
15	$\nearrow$	f_1			$\langle f_1, a_0, c_0 \rangle$
16	$\nearrow$	$\{c_0, f_1\}$	$\emptyset$		$\langle f_1, a_0, c_0 \rangle$
17	$\nearrow$	$\circ$	ε	$\{c_0, f_1\} \to a_0 \Rsh a_1$	$\langle f_1, a_1, c_0 \rangle$
18	$\nearrow$	$a_0 \Rsh^* a_1$			$\langle f_1, a_1, c_0 \rangle$
19	$\nearrow$	a_1			$\langle f_1, a_1, c_0 \rangle$

Table 1: Example of the extraction of a concretizing scenario from the local causality graph of Fig. 6(left), by using the algorithm of Subsection 3.3. The first column denotes the step number in the traversal, the second depicts the direction of traversal, either "$\searrow$" for descending or "$\nearrow$" for ascending, the third one is the name of the current node, the fourth one gives the marking of this node when it is left, the fifth one gives the actions output by the algorithm and the last column gives the current state of the model when leaving each node. In step #5, marked with an asterisk, the traversal visits node c_0, but this was arbitrarily chosen amongst the set $\{c_0, f_1\}$. Another traversal thus consists in visiting node f_1 first, although in this example it does not change the result.

gather about a hundred components and are detailed below. They are originally specified as Boolean networks, and have been automatically encoded into a Process Hitting model with two classes of priorities, whose dynamics is equivalent to an AAN[1]. Boolean networks being a subclass of AANs, this translation is straightforward. In the rest of this section, we first present the Pint implementation, then the two models that were used for our tests, and the finally summarise the quantitative results of these tests.

The Pint Implementation

The under-approximation presented in Sect. 3 has been implemented in the existing Pint software[2]. It comes in two versions:

- One version consists in building the local causality graph once and checking Theorem 1 on this graph only. This version is polynomial in the size of the model, and therefore scalable by nature by its low complexity. However, it may be non-conclusive due to the presence of cycles or non-independent synchronizations that could be avoided.

- The other version consists in checking Theorem 1 on every sub-graph obtained by considering sub-sets of the nodes in **Sol** of the local causality graph. This enumeration of "sub-solutions" may allow to find conclusive responses by removing cycles or unnecessary local states in the graph. However, due to its exhaustive nature, it is exponential in the number of solutions of each objective, and thus not as scalable.

In other words, when this implementation is unable to conclude with the local causality graph alone, it tries to enumerate sub-solutions in order to reach a conclusion. Although this search is guided in order to remove in priority the solutions that could create cycles or depended synchronizations, in the worst case all combinations have to be checked. Therefore, for the following tests, we capped the execution time of each call to Pint to 3 seconds, considering that the implementation is inconclusive above this threshold.

The under-approximation presented in this paper can therefore answer either *True* or *Inconclusive* regarding the reachability of a given local state. We used it together with a previously defined reachability over-approximation [14] that allows to answer either *False* or *Inconclusive*, but which was not tailored specifically, but still valid, for AANs.

The T-cell Receptor Model

The TCR signalling pathway consists of 94 components. We checked the reachability for the independent activation of the 4 outputs of the signalling

[1]Files are available at `http://maxime.folschette.name/underapprox-aan.zip`.

[2]Pint gathers tools related to the Process Hitting and is freely available at `http://loicpauleve.name/pint`. The release "2015-02-11" was used for these experiments.

cascade (SRE, AP1, NFkB, NFAT) for all possible combinations of the 3 inputs (CD4, CD28, TCRlig). All result in conclusive decisions, and our under-approximation has been satisfied in 12 cases (over 32) proving the satisfiability of the concerned reachability properties in the encoded Boolean network (and non-satisfiability in the other cases, using the previously existing over-approximation).

By analysing the detail of the results, one can find out that one of the outputs, NFkB, can never be activated. In other words, FkB_1 is never reachable, whatever the initial state of the inputs, while the other outputs (SRE, AP1, NFAT) can be independently activated for some configurations of inputs. We thus wanted to check if these three outputs could be activated together, that is, if the activation of one of the outputs did not prevent the activation another one. For this, as proposed in Subsection 3.4, we have added a new automaton σ into the model with two local states ($\mathcal{L}_\sigma = \{\sigma_0, \sigma_1\}$) and the action:

$$\{\mathrm{SRE}_1, \mathrm{AP1}_1, \mathrm{NFAT}_1\} \to \sigma_0 \Rsh \sigma_1 \ .$$

Finally, checking the reachability of σ_1 in all configurations of the inputs has always been conclusive, and the existence of 4 positive answers (amongst all 8 possible initial configurations of the inputs) allows to conclude that it is possible to reach a state where SRE, AP1 and NFAT are simultaneously active.

The Epidermal Growth Factor Receptor Model

The EGFR signalling pathway gathers 104 components, amongst which one can distinguish 21 inputs and 12 outputs. We selected 13 of all inputs (erbb1, erbb2, erbb3, erbb4, bir, btc, egf, epr, nrg1a, nrg1b, nrg2b, nrg4, tgfa) and observed the impact of their variations on all outputs (elk1, creb, ap1, hsp27, actin_reorg, cmyc, pro_apoptotic, p70s6_2, pkc, stat1, stat3, stat5). We note that some of the experiments trigger the exponential search of sub-solutions described above, and are cut after 3 seconds of computation (thus leading to *Inconclusive* cases). These "interrupted" experiments represent about 10% of all tests. We note that amongst all tests that terminated "normally" (without being cut after 3 seconds) none of them responded with an inconclusive answer, which is however theoretically possible.

Synthesis of the Results

The results of all tests performed with the implementation of our under-approximation are summarised in Table 2 (columns "AAN"). We held all these experiments on a personal computer, and regarding the experiments that were not cut at the 3 seconds limit, computations times were in the order of a few tenths of a second to about one second. To give a comparison, we did the same experiments with a standard symbolic model-checker, LibDDD [21], known for its good performances, the input model being the Boolean network expressed as a Petri net. However, due to the large scale of the model, this program takes at least several minutes to terminate, and runs out of memory for the majority of all experiments. On the other hand, our method is able to conclude with

limited memory and computation time usage in the majority of the cases, and is expected to be scalable to models that are even larger, even by orders of magnitude.

Finally, we note that similar experiments were conducted in [14]. However, if these experiments allowed to obtain some information on the TCR and EGFR models to some extent, they did not provide a formal "True" response regarding the reachabilities that have been checked in the examples above. Indeed, the semantics of Process Hitting (without classes of priorities) does not allow to model accurate Boolean gates, thus leading to some unwanted spurious behaviours that especially take the form of temporal shifts. The adding of classes of priorities allows to remove these temporal shifts, as explained in [16], with a construction that is equivalent to the AANs presented in Sect. 2. Therefore, only the method presented in this paper provides a formal proof that the observed behaviours are the result of the true dynamics of the systems. Thus, additionally to the experiments previously mentioned, we conducted similar experiments with the previous version of the under-approximation, that are reported alongside in Table 2 (in columns "PH"). We note that indeed, about 15% of the positive results regarding the EGFR model could not be proven with our new under-approximation, and are thus still formally uncertain.

	TCR		EGFR	
Inputs	3		13	
Outputs	$4 + \sigma$		12	
Total tests	40		98'304	
	PH	AAN	PH	AAN
True	16 (40%)	16 (40%)	74'268 (75,55%)	64'282 (65,39%)
Inconclusive	0 (0%)	0 (0%)	0 (0%)	9'986 (10,16%)
False	24 (60%)		24'036 (24,45%)	
Max time	0.043s	0.20s	0.37s	0.87s
Total time	<1s	<1s	45min	9h50min

Table 2: Results of the tests on large-scale examples. The "AAN" column gives the related results on AAN models, using the under-approximation presented in this paper, while the "PH" column gives the results for PH models using cooperative sorts to model actions with multiple hitters, and using the under-approximation of [14]. The lines labelled "True", "Inconclusive" and "False" give respectively the number of positive answers, experiments cut after 3 seconds and negative answers; while "Max time" and "Total time" depict respectively the maximum time of the individual computations (except those cut at 3 seconds) and the overall execution time of all tests (including those cut at 3 seconds). The greyed cells highlight the results that are proper to the under-approximation presented in this paper.

In conclusion, while ensuring a low complexity for the analysis of reachability in Boolean and discrete networks, our under-approximation method turns out to be conclusive in numerous cases when applied to real large-scale biological models, which were not tractable in most cases with exact model-checking.

6. Discussion & Conclusion

In this paper, we focused on Asynchronous Automata Networks (AANs), which are a restriction of classical Automata Networks and are equivalent to Process Hitting models with classes of priorities [16]. This formalism proves useful to model accurate Boolean gates, which was not possible with the standard form of the Process Hitting, and avoid an unwanted over-approximation of the dynamics. Furthermore, we proposed an extension of this formalism with classes of priorities, which prove convenient to abstract time parameters into these kinds of models or more simply to add preemption relations between actions, and showed that any AAN with priorities can be translated into an equivalent AAN without priorities.

Then, we developed a method to perform a reachability analysis of a local state in an AAN, based on an under-approximation of the true reachability solutions. We also extended this analysis to global and partial states, and to the successive reachability of several local states. The method can be considered efficient, because it is polynomial in the size of the model. A more conclusive analysis also exists, but at the price of being exponential in the number of local solutions. Finally, AANs with classes of priorities can also be studied in this way, at the cost of an exponential translation that we gave in this paper. We applied it to a large number of experiments on two large-scale biological models, and obtained in the worst case a ratio of 90% of conclusive cases with the joint use of a previously proposed over-approximation, although limiting the computation time to 3 seconds for each test.

AANs are also equivalent to Logical Networks, that is, either multivalued or Boolean networks with evolution functions or focal parameters, such as Thomas' models with Snoussi parameters. This especially allows to efficiently compute reachability results on large biological models, provided that they are equivalent to Logical Networks, which ensures that a translation to AANs is possible. For example, such a translation for generalized Interaction Graphs, that is, Discrete Networks without evolution functions or parameters, was proposed in [6]. AANs can also be used to represent sub-sets of cellular automata [22] or multiplayer games [23], although more experiments would be required to evaluate the conclusiveness of our static analysis method on models of such particular forms.

Further work can also be directly derived to improve the method presented in this paper. The over-approximation on Process Hitting models without priorities proposed in [14] and that was used in this work is still accurate on AANs (by over-approximating dynamics) but may be refined given the particular for of AANs proposed in this paper. A specific search of key local states or cut sets [24] may especially be derived. Furthermore, we are investigating alternative under-approximations that can be applied directly to the whole class of AANs or Process Hitting models with priorities, and not only to a sub-class with particular restrictions; such improvement may permit to increase the conclusiveness of the static analysis while allowing to analyse any model without the need of a translation. Finally, in order to take into account quantitative

data in transition delays, the overall approximation method could be extended to handle evolutions that are chronometric instead of only chronologic. This may require the addition of information such as time delays in the model, that would be exploited during the solving.

Acknowledgement. The European Research Council has provided financial support under the European Community's Seventh Framework Programme (FP7/2007–2013) / ERC grant agreement no. 259267.

References

[1] R. Thomas, R. d'Ari, Biological feedback, CRC press, 1990.

[2] H. De Jong, Modeling and simulation of genetic regulatory systems: a literature review, Journal of computational biology 9 (1) (2002) 67–103.

[3] E. H. Snoussi, R. Thomas, Logical identification of all steady states: The concept of feedback loop characteristic states, Bulletin of Mathematical Biology 55 (5) (1993) 973–991.

[4] S. A. Kauffman, Metabolic stability and epigenesis in randomly constructed genetic nets, Journal of theoretical biology 22 (3) (1969) 437–467.

[5] R. Thomas, Boolean formalization of genetic control circuits, Journal of Theoretical Biology 42 (3) (1973) 563 – 585.

[6] L. Paulevé, M. Magnin, O. Roux, Refining dynamics of gene regulatory networks in a stochastic π-calculus framework, in: Transactions on Computational Systems Biology XIII, Springer, 2011, pp. 171–191.

[7] L. Paulevé, M. Magnin, O. Roux, From the process hitting to petri nets and back, Tech. rep. (2012).

[8] F. Bause, Analysis of Petri nets with a dynamic priority method, in: P. Azéma, G. Balbo (Eds.), Application and Theory of Petri Nets 1997, Vol. 1248 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1997, pp. 215–234.

[9] A. Wagler, R. Weismantel, The combinatorics of modeling and analyzing biological systems, Natural Computing 10 (2011) 655–681.

[10] A. Wagler, J.-T. Wegener, On minimality and equivalence of Petri nets, in: L. Popova-Zeugmann (Ed.), CS&P, Vol. 928 of CEUR Workshop Proceedings, CEUR-WS.org, 2012, pp. 382–393.

[11] R. Cleaveland, M. Hennessy, Priorities in process algebras, Information and Computation 87 (1) (1990) 58–77, special Issue: Selections from 1988 IEEE Symposium on Logic in Computer Science.

[12] R. Cleaveland, G. Lüttgen, V. Natarajan, Priority and abstraction in process algebra, Information and Computation 205 (9) (2007) 1426–1458.

[13] M. John, C. Lhoussaine, J. Niehren, A. Uhrmacher, The attributed pi-calculus with priorities, in: C. Priami, R. Breitling, D. Gilbert, M. Heiner, A. Uhrmacher (Eds.), Transactions on Computational Systems Biology XII, Vol. 5945 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2010, pp. 13–76.

[14] L. Paulevé, M. Magnin, O. Roux, Static analysis of biological regulatory networks dynamics using abstract interpretation, Mathematical Structures in Computer Science 22 (04) (2012) 651–685.

[15] R. Thomas, D. Thieffry, M. Kaufman, Dynamical behaviour of biological regulatory networks — I. Biological role of feedback loops and practical use of the concept of the loop-characteristic state, Bulletin of Mathematical Biology 57 (2) (1995) 247–276.

[16] M. Folschette, L. Paulevé, M. Magnin, O. Roux, Under-approximation of reachability in multivalued asynchronous networks, Electronic Notes in Theoretical Computer Science 299 (2013) 33 – 51, 4th International Workshop on Interactions between Computer Science and Biology (CS2Bio'13).

[17] D. Harel, O. Kupferman, M. Y. Vardi, On the complexity of verifying concurrent transition systems, Information and Computation 173 (2) (2002) 143–161.

[18] P. François, V. Hakim, E. D. Siggia, Deriving structure from evolution: metazoan segmentation, Molecular Systems Biology 3 (1) (2007) .

[19] J. Saez-Rodriguez, L. Simeoni, J. A. Lindquist, R. Hemenway, U. Bommhardt, B. Arndt, U.-U. Haus, R. Weismantel, E. D. Gilles, S. Klamt, B. Schraven, A logical model provides insights into t cell receptor signaling, PLoS Comput Biol 3 (8) (2007) e163.

[20] R. Samaga, J. Saez-Rodriguez, L. G. Alexopoulos, P. K. Sorger, S. Klamt, The logic of egfr/erbb signaling: Theoretical properties and analysis of high-throughput data, PLoS Computational Biology 5 (8) (2009) e1000438.

[21] LIP6/Move, the libDDD environment, `http://ddd.lip6.fr` (libDDD).

[22] D. Cornforth, D. G. Green, D. Newth, M. Kirley, Do Artificial Ants March in Step? Ordered Asynchronous Processes and Modularity in Biological Systems, in: Proceedings of the eighth international conference on Artificial life, MIT Press, 2003, pp. 28–32.

[23] R. Mazala, Infinite Games, in: E. Grädel, W. Thomas, T. Wilke (Eds.), Automata Logics, and Infinite Games, Vol. 2500 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2002, pp. 23–38.

[24] L. Paulevé, G. Andrieux, H. Koeppl, Under-approximating cut sets for reachability in large scale automata networks, in: N. Sharygina, H. Veith (Eds.), Computer Aided Verification, Vol. 8044 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 69–84.

A. Proof of Under-approximation (Subsection 3.2)

We introduce for this proof the notion of context (Def. 15), which extends the notion of state to a set of possible initial states: to each automaton in the model, a context maps a set of local states in this automaton.

Definition 15 (Context). A *context* ς associates to each automaton in Σ a non-empty subset of its local states: $\forall a \in \Sigma, \varsigma(a) \subseteq \mathcal{L}_a \wedge \varsigma(a) \neq \emptyset$.

For a given context ς, and for all state $s \in \mathcal{L}$, we note $s \subseteq \varsigma$ if and only if $\forall a_i \in s, a_i \in \varsigma(a)$.

In the following, we also denote: $E^{u\,X}_{s\;Y} = E^u_s \cap (X \times Y)$, with X, Y amongst **LS**, **Obj**, **Sync** and **Sol**.

Proof of Theorem 1. Given the LCG $\mathcal{B}^u_s = (V^u_s, E^u_s)$, we note $\varsigma = \{a \mapsto V^u_s \cap \mathcal{L}_a \mid a \in \Sigma\}$ the context supported by $\mathcal{B}^u_s$.

Let $ps \in V^u_s \cap \mathbf{Sync}$ be a set of hitters that is a node in the LCG, and suppose all of its successors are concretizable. We first want to demonstrate that there exists a scenario that activates all the local states it contains. We label all local states of ps by an integer: $ps = \{p_m\}_{m \in \mathbb{I}^{ps}}$. Let us prove by induction that for all $n \in \{0\} \cup \mathbb{I}^{ps}$, there exists a scenario δ_n so that: $\forall i \in [\![1; n]\!], (s \cdot \delta_n)[\Sigma(p_i)] = p_i$.

- It is straightforward that ε is a valid value for δ_0.
- Suppose such δ_n exists and let $q = (s \cdot \delta_n)[\Sigma(p_{n+1})]$. By construction, $p_{n+1} \in V^u_s \cap \mathbf{LS}$ is a child of ps. Furthermore, by hypothesis, ps is independent (see Def. 8). This means that amongst all the successors in **LS** of p_{n+1}, there does not exist a local state b_j to that $\exists b_k \in ps, \Sigma(b_j) = \Sigma(b_k) \wedge b_j \neq b_k$; in other words, the resolution of p_{n+1} does not require a local state that may change the other local states of the set ps. Therefore, there exists $\delta' \in \gamma_{s \cdot \delta_n}(q \restriction^* p_{n+1})$, so that $\forall i \in [\![1; n+1]\!]$, $(s \cdot \delta_n \cdot \delta')[\Sigma(p_i)] = p_i$. We denote then: $\delta_{n+1} = \delta_n \cdot \delta'$.

Therefore, $\delta = \delta_{|ps|}$ exists, and given its properties, we have: $\forall i \in [\![1; |ps|]\!], (s \cdot \delta)[\Sigma(p_i)] = p_i$.

As there is no cycle in $\mathcal{B}^u_s$, we show by induction in the following that $\forall s \in \mathcal{L}$ so that $s \subseteq \varsigma$ and $\forall P \in V^u_s \cap \mathbf{Obj}, \mathsf{target}(P) \in s \Longrightarrow \exists \delta \in \gamma_s(P)$.

- If $(P, \langle P, \{\emptyset\}\rangle) \in E^{u\,\mathbf{Obj}}_{s\;\mathbf{Sol}}$, either $\mathsf{target}(P) = \mathsf{bounce}(P)$ and $\delta = \varepsilon$; or $\exists \zeta \in \mathbf{BSeq}(P), \zeta \in \mathbf{Sce}(s) \wedge \forall i \in \mathbb{I}^\zeta, \mathsf{hitters}(\zeta_i) = \emptyset$ and in this case, $\delta = \zeta$ is a valid scenario in s.

- Suppose all successors objectives of P are concretizable. If $\exists(P, Q) \in E_s^{u\,\mathbf{Obj}}{}_{\mathbf{Obj}}$, then by hypothesis, $\gamma_s(\mathsf{target}(P)\mathrel{\mathsf{r}}^*\mathsf{target}(Q) :: Q) \neq \emptyset$, thus $\gamma_s(P) \neq \emptyset$. Else, by Def. 7-6, the concretizations of the successors of P require no local state of automaton $\Sigma(P)$. Furthermore, there exists $\zeta \in \mathbf{BSeq}(P)$ so that $(P, \zeta^\wedge) \in E_s^{u\,\mathbf{Obj}}{}_{\mathbf{Sol}}$. We show by induction that for all $n \in \mathbb{I}^\zeta$, there is a scenario δ_n so that $(s \cdot \delta_n)[\Sigma(P)] = \mathsf{bounce}(\zeta_n)$.

  - Suppose that δ_n exists and let $\zeta_n = A \to a_j \mathrel{\mathsf{r}}^{\nearrow} a_k$. By construction, there exists $A \in V_s^u \cap \mathbf{Sync}$ amongst the children of $\zeta^\wedge$. By the first result of this demonstration, there exists a scenario δ' in $s \cdot \delta_n$ so that $\forall a_i \in A, (s \cdot \delta_n \cdot \delta')[a] = a_i$. Therefore, ζ_n is playable in $s \cdot \delta_n \cdot \delta'$, and $\delta_{n+1} = \delta_n :: \delta' :: \zeta_n$.

  Thus, $\delta_{|\zeta|} \in \gamma_s(P)$.