



Attractor Basins in Concurrent Systems

Giann Karlo Aguirre Samboni, Stefan Haar, Loic Paulevé, Stefan Schwoon,
Nick Würdemann

► To cite this version:

Giann Karlo Aguirre Samboni, Stefan Haar, Loic Paulevé, Stefan Schwoon, Nick Würdemann. Attractor Basins in Concurrent Systems. 2024. hal-04687172

HAL Id: hal-04687172

<https://inria.hal.science/hal-04687172v1>

Preprint submitted on 4 Sep 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License

ATTRACTOR BASINS IN CONCURRENT SYSTEMS

GIANN KARLO AGUIRRE-SAMBONÍ ^{a,e}, STEFAN HAAR ^d, LOÏC PAULEVÉ ^b,
STEFAN SCHWOON ^a, AND NICK WÜRDEMANN ^c

^a Université Paris-Saclay, INRIA and LMF, CNRS and ENS Paris-Saclay, Gif-sur-Yvette, France
e-mail address: (giann-karlo.aguirre-samboni,stefan.schwoon)@inria.fr

^b Univ. Bordeaux, Bordeaux INP, CNRS, LaBRI, UMR5800, Talence, France
e-mail address: loic.pauleve@labri.fr

^c Department of Computing Science, University of Oldenburg, Oldenburg, Germany
e-mail address: wuerdemann@informatik.uni-oldenburg.de

^d INRIA Saclay center, MUSCA team
e-mail address: stefan.haar@inria.fr

^e Mines Paris, PSL Research University, Centre for Computational Biology, 75006 Paris, France
Institut Curie, PSL Research University, 75005 Paris, France
INSERM, U900, 75005 Paris, France
e-mail address: giann-karlo.aguirre_samboni@minesparis.psl.eu

ABSTRACT. A crucial question in analyzing a concurrent system is to determine its long-run behaviour, and in particular, whether there are irreversible choices in its evolution, leading into parts of the reachability space from which there is no return to other parts. Casting this problem in the unifying framework of safe Petri nets, our previous work [CHJ⁺14] has provided techniques for identifying *attractors*, i.e. terminal strongly connected components of the reachability space. What we aim at is to determine the *attraction basins* associated to those attractors; that is, those states from where all infinite runs are doomed to end in the given attractor, as opposed to those that are *free* to evolve differently. Here, we provide a solution for the case of safe Petri nets. Our algorithm uses net unfoldings and provides a map of all of those configurations (concurrent executions of the system) that lead onto *cliff-edges*, i.e. any maximal extension for those configurations lies in some basin that is considered fatal.

1. INTRODUCTION

With the growing interest in formal methods for biology, the key feature of *multistability* of systems [Td90, PMO95, OTL⁺04, PF14] comes into focus. It has been studied in other qualitative models such as Boolean and multivalued networks [Tho80, TT95, Ric19]. Multistability characterizes many fundamental biological processes, such as cellular differentiation, cellular reprogramming, and cell-fate decision; in fact, stabilization of a cell regulatory network corresponds reaching one of the - possibly many - phenotypes of the

cell, thus explaining the important role of multistability in cell biology. However, multistability emerges also in many other branches of the life sciences; our own motivation is the qualitative analysis of the fate of *ecosystems*, see [PTG22].

Multistability can be succinctly described as the presence of several *attractors* in the system under study. Attractors characterize the stable behaviours, given as the smallest subsets of states from which the system cannot escape; in other words, they are terminal strongly connected components of the associated transition system. In the long run, the system will enter one of its attractors and remain inside; multi-stability arises when there is more than one such attractor.

Remark: Regardless of the application domain, there are typically some attractors that play a more ‘negative’ role than others; in cell regulation, e.g., some attractors simply represent different healthy phenotypes into which a cell may differentiate, while others can be cancerous. In ecology, there may even be not a single attractor that can be considered ‘healthy’ in the sense that every such stable region may be characterized by the collapse of some species or sub-ecosystems. The survival, or the *avoidance of doom* as we will call it, consists for such systems in staying forever in some transient but doom-free loop. The purpose of this article is to provide formal tools for addressing these forms of doom avoidance, in the context of concurrent systems modeled by safe Petri nets.

Returning to the discussion of basic notions, the *basin* $\mathcal{B}(\mathbf{A})$ of a given attractor \mathbf{A} consists of the states that are *doomed* in the sense that any infinite run from them inevitably leads the system into \mathbf{A} .

The basin includes the attractor itself, and possibly one or several transient states [KHNS20].

We aim at finding the basin boundaries at which the system switches from an undetermined or *free* state into some basin. While interesting beyond that domain, this is a recurrent question in the analysis of signalling and gene regulatory networks [CMR⁺15, MHR⁺18]. In [FRGP17], the authors provide a method for identifying, in a boolean network model, the states in which one transition leads to losing the reachability of a given attractor (called *bifurcation transitions* there; we prefer to speak of *tipping points* instead). However, enumerating the states in which the identified transitions make the system branch away from the attractor can be highly combinatorial and hinders a fine understanding of the branching. Thus, the challenge resides in identifying the specific contexts and sequences of transitions leading to a strong basin.

Unfoldings of Petri nets [EH08], which are essentially event structures in the sense of Winskel et al. [NPW79] with additional information about *states*, are an acyclic representation of the possible sequences of transitions, akin to Mazurkiewicz traces but enriched with branching information.

Many reachability-related verification problems for concurrent systems have been successfully addressed by Petri-net unfolding methods over the past decades, see [McM92, ERV02, EH08]. However, questions of long-term behaviour and stabilization have received relatively little attention.

We have shown in previous work [CHJ⁺14] how all reachable attractors can be extracted using bounded unfolding prefixes. Also, we have exhibited ([HPS20]) the particular shape of basins that can arise in a concurrent model.

In the present paper, we build on these previous results; the point of view taken here is that all attractors correspond to the *end* of the system’s free behaviour, in other words to its *doom*. We will give characterizations of basin boundaries (called *cliff-edges* below), and of those behaviours that remain *free*, in terms of properties of the unfolding, reporting

also on practical experiments with an implementation of the algorithms derived. We finally introduce a novel type of quantitative measure, called *protectedness*, to indicate how far away (or close) a system is from doom, in a state that is still free per se. General discussions and outlook will conclude this paper.

2. PETRI NETS AND UNFOLDINGS

We begin now by recalling the basic definitions needed below. A **Petri net** is a bipartite directed graph whose nodes are either *places* or *transitions*, and places may carry *tokens*. In this paper, we consider only *safe* Petri nets where a place carries either one or no token in any reachable marking. The set of currently active places form the state, or *marking*, of the net.

Note. Some remarks are in order concerning our use of Petri nets versus that of *boolean networks*, which are more widely used in systems biology. Safe (or 1-bounded) Petri nets [Mur89] are close to Boolean and multivalued networks [CHK⁺20], yet enable a more fine-grained specification of the conditions for triggering value changes. Focussing on safe PNs entails no limitation of generality of the model, as two-way behaviour-preserving translations between Boolean and multivalued models exist (see [CHK⁺20] and the appendix of [CHJ⁺14] for discussion). We are thus entitled to move between these models without loss of expressiveness; however, Petri nets provide more convenient ways to develop and present the theory and the algorithms here.

Formally, a *net* is a tuple $N = \langle P, T, F \rangle$, where T is a finite set of *transitions*, P a finite set of *places*, and $F \subseteq (P \times T) \cup (T \times P)$ is a *flow relation* whose elements are called *arcs*. In figures, places are represented by circles and the transitions by boxes (each one with a label identifying it).

For any node $x \in P \cup T$, we call *pre-set* of x the set $\bullet x = \{y \in P \cup T \mid \langle y, x \rangle \in F\}$ and *post-set* of x the set $x^\bullet = \{y \in P \cup T \mid \langle x, y \rangle \in F\}$. A *marking* for N is a mapping $M : P \rightarrow \mathbb{N}_0$. A *Petri net* is a tuple $\mathcal{N} = \langle P, T, F, M_0 \rangle$, with $M_0 \subseteq P$ (i.e., $M_0 : P \rightarrow \{0, 1\}$) an *initial marking*. Markings are represented by dots (or tokens) in the marked places. A transition $t \in T$ is *enabled* at a marking M , denoted $M \xrightarrow{t}$, if and only if $\forall p \in \bullet t : M(p) \geq 1$. An enabled transition t can *fire*, leading to the new marking M' given by $M'(p) = (M(p) - \mathbf{1}_{\bullet t}(p)) + \mathbf{1}_{t^\bullet}(p)$ in that case we write $M \xrightarrow{t} M'$. A *firing sequence* from a marking M'_0 is a (finite or infinite) sequence $w = t_1 t_2 t_3 \dots$ over T such that there exist markings M'_1, M'_2, \dots with $M'_0 \xrightarrow{t_1} M'_1 \xrightarrow{t_2} M'_2 \xrightarrow{t_3} \dots$. If w is finite and of length n , we write $M'_0 \xrightarrow{w} M'_n$, and we say that M'_n is *reachable* from M'_0 , also simply written $M'_0 \rightarrow M'_n$. We denote the set of markings reachable from some marking M in a net N by $\mathbf{R}_N(M)$.

A *cycle* is a tuple $\langle M_1, \dots, M_n \rangle$ of reachable markings such that there exists a finite firing sequence $w = t_1 t_2 \dots t_n$ for which $M_1 \xrightarrow{t_1} M_2 \xrightarrow{t_2} M_3 \xrightarrow{t_3} \dots \xrightarrow{t_n} M_1$. Clearly, $\langle M_1, \dots, M_n \rangle$ is a cycle iff $\langle M_2, \dots, M_n, M_1 \rangle$ is. A Petri net $\langle N, M_0 \rangle$ is *n-bounded* if $M(p) \leq n$ for every reachable marking $M \in \mathbf{R}_N(M_0)$ and every place $p \in P$. A 1-bounded Petri net is called *safe*. In this paper, we assume that all Petri nets considered are safe.

From an initial marking of the net, one can recursively derive all possible transitions and reachable markings, resulting in a *marking graph* (Def. 1).

Definition 1. Let $N = \langle P, T, F \rangle$ be a net and \mathcal{M} a set of markings. The marking graph induced by \mathcal{M} is a directed graph $\langle \mathcal{M}, \mathcal{E} \rangle$ such that $\mathcal{E} \subseteq \mathcal{M} \times \mathcal{M}$ contains $\langle M, M' \rangle$ iff

$M \xrightarrow{t} M'$ for some $t \in T$; the arc $\langle M, M' \rangle$ is then labeled by t . The reachability graph of a Petri net $\langle N, M_0 \rangle$ is the graph induced by $\mathbf{R}_N(M_0)$.

Figure 1b shows the reachability graph for our running example 1a. Note that the reachability graph is always finite for safe Petri nets.

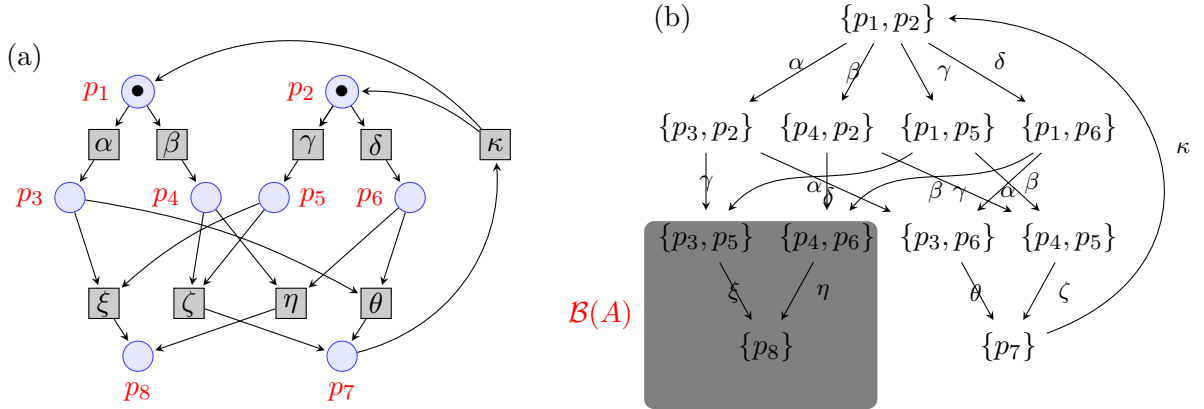


FIGURE 1. Petri net example from [HPS20] in (a), and its reachability graph in (b). The only attractor A is highlighted in dark gray, and its (strong) basin $\mathcal{B}(A)$ in light gray.

Unfoldings. Roughly speaking, the unfolding of a Petri net \mathcal{N} is an acyclic Petri net (with particular structural properties), \mathcal{U} exhibiting reproduces exactly the same non-sequential behaviours as \mathcal{N} .

Let us now give the technical definitions to introduce unfoldings formally. A more extensive treatment can be found, e.g., in [ERV02, EH08].

Definition 2 (Causality, conflict, concurrency). *Let $N = \langle P, T, F \rangle$ be a net and $x, y \in P \cup T$ two nodes of N . We say that x is a causal predecessor of y , noted $x < y$, if there exists a non-empty path of arcs from x to y . We note $x \leq y$ if $x < y$ or $x = y$. If $x \leq y$ or $y \leq x$, then x and y are said to be causally related. Transitions u and v are in direct conflict, noted $u \#_\delta v$, iff $\bullet u \cap \bullet v \neq \emptyset$; nodes x and y are in conflict, noted $x \# y$, if there exist $u, v \in T$ such that $u \neq v$, $u \leq x$, $v \leq y$, and $u \#_\delta v$. We call x and y concurrent, noted $x \text{ co } y$, if they are neither causally related nor in conflict. A set of concurrent places is called a co-set.*

In Figure 2, α_1 and β_1 are in conflict, while α_1 and γ_1 are concurrent. Further, α_1 is a causal predecessor of ξ_1 , θ_1 , and κ_2 ; readers will easily identify other relations in this figure.

Definition 3 (Occurrence net). *Let $\mathcal{O} = \langle B, E, G, \mathbf{c}_0 \rangle$ be a Petri net. We say that \mathcal{O} is an occurrence net if it satisfies the following properties:*

- (1) *The causality relation $<$ is acyclic and well-founded;*
- (2) *$|\bullet b| \leq 1$ for all places $b \in B$, and $b \in \mathbf{c}_0$ iff $|\bullet b| = 0$;*
- (3) *For every transition $e \in E$, $e \# e$ does not hold, and $\{x \mid x \leq e\}$ is finite.*

The reader is invited to check that the net in Figure 2 is indeed an occurrence net.

Following the convention in the unfolding literature, we refer to the places of an occurrence net as *conditions* (B) and to its transitions as *events* (E). Due to the structural constraints, the firing sequences of occurrence nets have special properties: if some condition b is marked during a run, then the token on b was either present initially in \mathbf{c}_0 , or produced by one particular event (the single event in $\bullet b$); moreover, once the token on b is consumed, it can never be replaced by another token, due to acyclicity of $<$.

Definition 4 (Configurations, cuts). *Let $\mathcal{O} = \langle B, E, G, \mathbf{c}_0 \rangle$ be an occurrence net. A set $C \subseteq E$ is called a configuration of \mathcal{O} if (i) C is causally closed, i.e. $e' < e$ and $e \in C$ imply $e' \in C$; and (ii) C is conflict-free, i.e. if $e, e' \in C$, then $\neg(e \# e')$. In particular, for any $e \in E$, $[e] \triangleq \{e' \in E : e' \leq e\}$ and $\langle e \rangle \triangleq \{e' \in E : e' < e\}$ are configurations, called the cone and stump of e , respectively; any C such that $\exists e \in E : C = [e]$ is called a prime configuration. Denote the set of all configurations of \mathcal{O} as $\mathcal{C}(\mathcal{O})$, and its subsets containing all **finite** configurations as $\mathcal{C}^f(\mathcal{O})$, where we drop the reference to \mathcal{O} if no confusion can arise. The cut of a finite C , denoted $\mathbf{cut}(C)$, is the set of conditions $(\mathbf{c}_0 \cup C^\bullet) \setminus \bullet C$. A run is a maximal element of $\mathcal{C}(\mathcal{O})$ w.r.t. set inclusion; denote the set of \mathcal{O} 's runs as $\Omega = \Omega(\mathcal{O})$, and its elements generically by ω . Denote by $\hat{\mathcal{C}}^\infty(\mathcal{O})$ the set of all infinite configurations, and let*

$$\mathcal{C}^\infty(\mathcal{O}) \triangleq \hat{\mathcal{C}}^\infty(\mathcal{O}) \cup \Omega(\mathcal{O})$$

If $C \in \mathcal{C}^f$, let the crest of C be the set $\mathbf{crest}(C) \triangleq \max_{<}(C)$ of its maximal events. We say that configuration C enables event e , written $C \xrightarrow{e}$, iff i) $e \notin C$ and ii) $C \cup \{e\}$ is a configuration. Configurations C_1, C_2 are in conflict, written $C_1 \# C_2$, iff $(C_1 \cup C_2) \notin \mathcal{C}$ or, equivalently, iff there exist $e_1 \in C_1$ and $e_2 \in C_2$ such that $e_1 \# e_2$.¹ Let C be a configuration and $E \cap C = \emptyset$ such that $C \cup E$ is a configuration. Write $C \oplus E \triangleq C \cup E$ in that case; we call $C \oplus E \triangleq C \cup E$ an extension of C , and E a suffix of C .

Intuitively, a configuration is the partially ordered set of transition firings occurring during an enabled firing sequence of \mathcal{N} , and its cut (if it exists) is the set of conditions marked after completing that firing sequence. Note that \emptyset is a configuration, that $\mathbf{crest}(\emptyset) = \emptyset$, and that \mathbf{c}_0 is the cut of the configuration \emptyset . Moreover, if $\mathcal{C}^\infty(\mathcal{O}) \neq \emptyset$, then $\mathcal{C}^\infty(\mathcal{O}) \cap \Omega(\mathcal{O}) \neq \emptyset$; however, it is in general not the case that $\mathcal{C}^\infty(\mathcal{O}) \subseteq \Omega(\mathcal{O})$. The crest of a prime configuration $[e]$ is $\{e\}$.

In Figure 2, the initial cut is $\mathbf{c}_0 = \{b_1^1, b_2^1\}$; we have prime configurations, e.g., $\{\alpha_1\}$, $\{\beta_1\}$, $\{\alpha_1, \gamma_1, \xi_1\}$, $\{\beta_1, \gamma_1, \zeta_1\}$ etc, and non-prime configurations $\{\alpha_1, \gamma_1\}$, $\{\alpha_1, \delta_1\}$ etc.

Definition of Unfoldings.

Definition 5 (Net homomorphism). *Let $N_1 = \langle P_1, T_1, F_1 \rangle$ and $N_2 = \langle P_2, T_2, F_2 \rangle$ be two nets. A homomorphism from N_1 to N_2 is a mapping $\phi : P_1 \cup T_1 \rightarrow P_2 \cup T_2$ such that $\phi(P_1) \subseteq P_2$ and $\phi(T_1) \subseteq T_2$, and that satisfies, in addition, for every $t \in T_1$,*

$$\phi(\bullet t) = \bullet \phi(t) \quad \text{and} \quad \phi(t^\bullet) = \phi(t)^\bullet$$

Definition 6 (Branching Process and Unfolding). *Let $\mathcal{N} = \langle P, T, F, M_0 \rangle$ be a safe Petri net. A branching process of \mathcal{N} is a pair $\Pi = (\mathcal{O}, \pi)$ with $\mathcal{O} = \langle B, E, G, \mathbf{c}_0 \rangle$ an occurrence*

¹The use of the same symbol $\#$ is motivated by the fact that $C_1 = [e_1]$ and $C_2 = [e_2]$ implies $C_1 \# C_2 \Leftrightarrow e_1 \# e_2$.

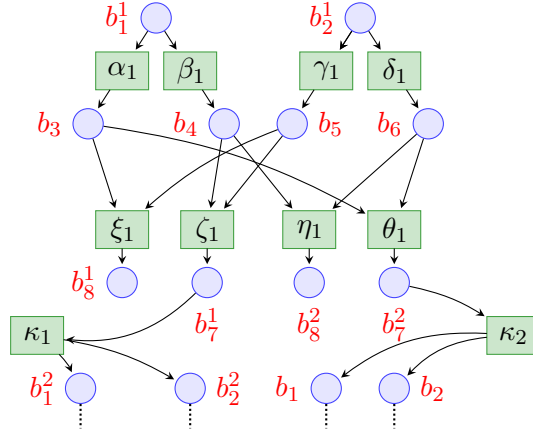


FIGURE 2. A prefix of the unfolding for the Petri net of Figure 1a.

net and $\pi : B \cup E \rightarrow P \cup T$ a homomorphism that satisfies the following parsimony property:

$$\forall e, e' \in E : \left. \begin{array}{l} \bullet e = \bullet e' \\ \pi(e) = \pi(e') \end{array} \right\} \Rightarrow e = e' \quad (2.1)$$

If $\Pi_1 = (\mathcal{O}_1, \pi_1)$ and $\Pi_2 = (\mathcal{O}_2, \pi_2)$ are two branching processes of \mathcal{N} , we say that Π_1 is a prefix of Π_2 iff i) \mathcal{O}_1 is a prefix modulo net isomorphism of \mathcal{O}_2 , and ii) π_1 agrees on its domain, modulo net isomorphism, with π_2 's restriction to π_1 's domain. There exists a unique (up to isomorphism) branching process $\Pi_{\mathcal{U}} = (\mathcal{U}, \pi_{\mathcal{U}})$ that is maximal in the sense that any branching process of which $\Pi_{\mathcal{U}}$ is a prefix, must be an isomorphic copy of $\Pi_{\mathcal{U}}$; we call unfolding of \mathcal{N} the maximal branching process $\Pi_{\mathcal{U}}$ and, by abuse of terminology, also the occurrence net \mathcal{U} if no confusion can arise.

In the unfolding (\mathcal{U}, π) of \mathcal{N} with $\mathcal{U} = \langle B, E, G, \mathbf{c}_0 \rangle$, the firing sequences and reachable cuts of \mathcal{U} correspond exactly the firing sequences and reachable markings of \mathcal{N} under the homomorphism π , see below. Note that the occurrence net \mathcal{U} may be infinite; it can be inductively constructed as follows:

- (1) Every condition in B is characterized by a pair $(e, p) \in (E \cup \{\perp\}) \times P$. For condition $b = \langle e, p \rangle$, we will have $e = \perp$ iff $b \in \mathbf{c}_0$; otherwise e is the singleton event in $\bullet b$. Moreover, $\pi(b) = p$. The initial cut \mathbf{c}_0 contains as many conditions $\langle \perp, p \rangle$ for each token initially on place p under M_0 in \mathcal{N} .
- (2) The events of E are a subset of $2^B \times T$. More precisely, for every co-set $B' \subseteq B$ such that $\pi(B') = \bullet t$, we have an event $e = \langle B', t \rangle$. In this case, we add edges $\langle b, e \rangle$ for each $b \in B'$ (i.e. $\bullet e = B'$), we set $\pi(e) = t$, and for each $p \in t^\bullet$, we add to B a condition $b = \langle e, p \rangle$ connected by an edge $\langle e, b \rangle$.

Intuitively, a condition $\langle e, p \rangle$ represents the possibility of putting a token onto place p through a particular set of events, while an event $\langle B', t \rangle$ represents a possibility of firing transition t in a particular context.

Configurations and Markings. The following facts from the literature will be useful:

Lemma 1 (see e.g. [ERV02]). *Fix $\mathcal{N} = \langle P, T, F, M_0 \rangle$ and its unfolding $\mathcal{U} = \langle B, E, G, \mathbf{c}_0, \pi \rangle$. Then for any two conditions (events) b, b' (e, e') such that $b \mathbf{co} b'$ ($e \mathbf{co} e'$), one has $\pi(b) \neq \pi(b')$ ($\pi(e) \neq \pi(e')$). Moreover, every finite configuration C of \mathcal{U} represents a*

possible firing sequence whose resulting marking corresponds, due to the construction of \mathcal{U} , to a reachable marking of \mathcal{N} . This marking is defined by $\text{Mark}(C) \triangleq \pi^{-1}(\mathbf{cut}(C))$. Moreover, for any two distinct configurations C_1, C_2 that satisfy $\text{Mark}(C_1) = \text{Mark}(C_2)$, we have an isomorphism of labeled occurrence nets

$$\mathbf{I}_{(C_1, C_2)} : \mathcal{U}_{/C_1} \rightarrow \mathcal{U}_{/C_2}, \quad (2.2)$$

where $\mathcal{U}_{/C_1}$ is the suffix of \mathcal{U} after removing configuration C_1 and all nodes in conflict with C_1 .

In fact, $\mathcal{U}_{/C_1}$ and $\mathcal{U}_{/C_2}$ in (2.2) are isomorphic copies of $\mathcal{U}(\mathcal{N}, \text{Mark}(C_1))$. This means, informally speaking, that any configuration of the system can be split into consecutive parts in such a way that each part is itself a configuration obtained by unfolding the Petri net ‘renewed’ with the marking reached by the previous configuration.

Complete Prefix. In general, \mathcal{U} is an infinite net, but if \mathcal{N} is bounded, then it is possible to compute a finite prefix Π of \mathcal{U} that is “complete” in the sense that every reachable marking of \mathcal{N} has a reachable counterpart in Π , and vice versa. One may require other completeness properties, as we will see below; here, the definition follows the notion dominant in the literature.

Definition 7 (complete prefix, see [McM92, ERV02, EH08]). *Let $\mathcal{N} = \langle N, M_0 \rangle$ be a bounded Petri net and $\mathcal{U} = \langle B, E, G, \mathbf{c}_0, \pi \rangle$ its unfolding. A finite occurrence net $\Pi = \langle B', E', G', \mathbf{c}_0 \rangle$ is said to be a prefix of \mathcal{U} if $E' \subseteq E$ is causally closed, $B' = \mathbf{c}_0 \cup E'^\bullet$, and G' is the restriction of G to B' and E' . A prefix Π is said to be complete if for every reachable marking M of \mathcal{N} there exists a configuration C of Π such that (i) $\text{Mark}(C) = M$, and (ii) for each transition $t \in T$ enabled in M , there is an event $\langle B'', t \rangle \in E'$ enabled in $\mathbf{cut}(C)$.*

We shall write $\Pi_0 = \Pi_0(\mathcal{N})$ to denote an arbitrary complete prefix of the unfolding of \mathcal{N} . The construction of such a complete prefix is indeed possible ([McM92, ERV02]), and efficient tools such as ECOFOLDER ([AS24]) and MOLE ([Sch14]) exist for this purpose. Several ingredients of this construction will play a role below, so we sketch them here.

Cutoff events and the complete prefix scheme. The unfolding is stopped on each branch when some *cutoff event* is added.

The criterion for classifying an event e as cutoff is given by marking equivalence: the marking $\text{Mark}([e])$ that e ‘discovers’ has already been discovered by a *smaller* (wrt some ordering relation \prec) configuration. Now, to ensure completeness, the ordering relation \prec to compare two configurations must be an *adequate* order, in the following sense:

Definition 8 ([ERV02], Def. 4.5). *A partial order \prec on \mathcal{C}^f is called adequate order iff*

- \prec is well-founded,
- $C_1 \subseteq C_2$ implies $C_1 \prec C_2$, and
- \prec preserves extensions, i.e. for any $C_1 \prec C_2$ such that $\text{Mark}(C_1) = \text{Mark}(C_2)$, one has $C_1 \oplus E \prec C_2 \oplus \mathbf{I}_{(C_1, C_2)}(E)$ for the isomorphism $\mathbf{I}_{(C_1, C_2)}$ from (2.2).

As shown in [ERV02], for some choices of \prec , the obtained prefix may be bigger than the reachability graph for some safe nets; however, if \prec is a *total* adequate order, the number of non-cutoff events of the prefix Π_0 thus obtained never exceeds the size of the reachability graph.

We will refer throughout this paper to the complete prefixes Π_0^\prec computed according to some adequate *total* order, as is done in the tools MOLE [Sch14] and ECOFOLDER [AS24],

as *España prefixes*. If instead one chooses $\prec = \sqsubset$, the resulting prefix $\Pi^{McM} \triangleq \Pi_0^{\sqsubset}$, as can be done in ECOFOLDER [AS24], is referred to as the *McMillan prefix*.

3. ATTRACTORS, BASINS, AND FAIRNESS

3.1. Attractors and Basins.

Definition 9. An attractor $\mathbf{A} \subseteq 2^P$ is a terminal SCC of the marking graph; that is, two states in \mathbf{A} are reachable from one another, and no state outside \mathbf{A} is reachable from any state in \mathbf{A} . Denote the set of attractors of \mathcal{N} reachable from a marking M in N by \mathcal{A} . Attractor \mathbf{A} is a fixed point iff there is $M \in \mathcal{M}$ such that $\mathbf{A} = \{M\}$, and for any $t \in T$, $M \xrightarrow{t} M'$ implies $M = M'$.

Unfoldings do not show attractors directly; however, the following observations are useful (fixing $\mathcal{N} = (\mathcal{N}, M_0)$ and $\mathbf{A} \in \mathcal{A}(\mathcal{N})$):

- If some finite configuration C satisfies $\text{Mark}(C) \in \mathbf{A}$, then so does any finite configuration C' such that $C \subseteq C'$. Write $\mathcal{C}_{\mathbf{A}}^f \triangleq \{C \in \mathcal{C}^f : \text{Mark}(C) \in \mathbf{A}\}$.
- In the light of the above, call a maximal run ω an \mathbf{A} -run iff there exists $C \in \mathcal{C}_{\mathbf{A}}^f$ such that $C \subseteq \omega$. Denote the set of \mathbf{A} -runs by $\Omega_{\mathbf{A}}$.

Definition 10. The \mathcal{C} -basin $\mathcal{B}^*(\mathbf{A})$ of \mathbf{A} is the set of finite configurations all of whose maximal extensions land in \mathbf{A} : $\mathcal{B}^*(\mathbf{A}) \triangleq \{C \in \mathcal{C}^f : \forall \omega \in \Omega : C \subseteq \omega \Rightarrow \omega \in \Omega_{\mathbf{A}}\}$. The basin $\mathcal{B}(\mathbf{A})$ of \mathbf{A} is the set of markings from which reaching \mathbf{A} is inevitable: $\mathcal{B}(\mathbf{A}) \triangleq \{M \in \mathcal{R}(M_0) : \forall C \in \mathcal{C}^f(\mathcal{N}) : \text{Mark}(C) = M \Rightarrow C \in \mathcal{B}^*(\mathbf{A})\}$

By definition, any attractor is an *absorbing* set of states (and so is any basin); once a system run enters some attractor (some basin), it will stay there forever. A different question is whether any infinite execution will eventually enter some attractor basin; in general, the answer is negative, since the system may exhibit transient loops in which it can forever remain active without ever entering any basin.

One of our central tasks below is to identify whether or not a system state allows to loop in such away as to avoid a fatal attractor's basin, or *doom* as we will say. To clarify this point, and to close a gap in the literature on attractors, we will next discuss which *fairness* properties prevent transient loops. Thus, the conclusion will be - as our title suggests - that, roughly speaking, fairness in behaviour may lead the system into doom. Put otherwise, avoiding doom requires to impose some sort of control in the system to prevent its free action from fatality.

3.2. Fairness, and how it leads into an attractor basin.

Situation Fairness. Despite their name, attractors do not in any way ‘attract’ the system’s behaviour in their direction, nor is the system necessarily entering any attractor eventually. Standard examples are non-attractor loops in the state graph; restrictions to behaviour are needed to ensure that the system eventually leaves such loops. Such an intuition is often captured by the notion of (*strong*) *fairness*, cf. [KW97, Vog95]: any transition that is *enabled* infinitely often, must also eventually occur. It is often assumed that strong fairness is sufficient to guarantee that all maximal runs eventually enter one or another attractor (and will obviously stay in it). We report here that in concurrent systems, this assumption is false. To see the point, consider Figure 3. The Petri net’s only attractor is formed by the marking $\{A\}$ (which coincides with its basin). However, the Petri net depicted might cycle forever in the set of states in which p_3 and p_4 are never jointly marked, and therefore never enter the attractor basin. This shows that in order to ensure that the system eventually enters some attractor, we need to restrict its behavior to those runs that ‘eventually explore all accessible branches’².

However, strong fairness is not sufficient to ensure that a concurrent system eventually enters a terminal SCC; the example of Figure 3 illustrates this.

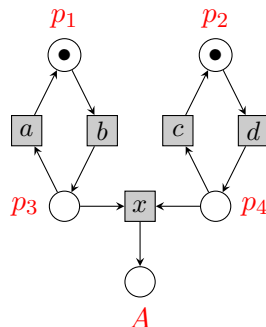


FIGURE 3. Illustration of the fairness condition underlying the attractor notion. Clearly, the system has exactly one attractor, given by fixed point $\mathbf{A} \triangleq \{\{A\}\}$. However, sequence $(badc)^\infty$ is strongly fair and never reaches \mathbf{A} ; on the other hand, every situation fair execution leads, after finitely but unboundedly many steps, into \mathbf{A} .

In this Petri net, a finite firing sequence can be strongly fair only if its final marking is $\{A\}$; At the same time, no infinite strongly fair firing sequence in this net would permit x to be enabled infinitely often. Nevertheless, there are infinite strongly fair executions that avoid enabling x too often, e.g. the sequences $(badc)^\infty$ or $(babadc)^\infty$, etc. Such sequences, however, must necessarily be unfair to some transition *in the context of the same marking*; in the example, the sequence $(badc)^\infty$ yields infinitely often the marking $M \triangleq \{p_2, p_3\}$, and from this marking it ‘chooses’ a constantly, although d is also enabled in M . Yet, strong fairness is fulfilled because d fires infinitely often, but not from instances of M . To eliminate such ‘missed opportunities’, we introduce a finer fairness notion. We will call an execution *situation fair* iff any transition enabled in an infinitely visited marking, also fires infinitely often *from that marking*. More formally:

²Dually, of course, such behaviour is to be *avoided* at any cost if it is undesirable to enter some attractor; we will return to this point in the next sections.

Definition 11 (Situation fairness). In \mathcal{N} as above, a firing sequence $M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} \dots$ is situation-fair iff either (i) it is finite, and its last marking enables no transition, or (ii) for all $t \in T$ and all $M \subseteq P$ such that $M \xrightarrow{t}$:

$$|\{i \in \mathbb{N} : M_i = M\}| = \infty \implies |\{j \in \mathbb{N} : M_j = M \wedge t_{j+1} = t\}| = \infty. \quad (3.1)$$

Note that such executions always exist; they may be obtained e.g. by applying a *round robin* firing policy, in which, for $\{t_0, \dots, t_{n-1}\}$ the transitions enabled at marking M , the transition selected at the k -th visit to marking M is $t_{(k \bmod n)}$. Before establishing the link between fairness and attractors, let us introduce one more auxiliary notion:

Definition 12. For any reachable marking M , let K_M be the smallest integer k such that there exist an attractor \mathbf{A} and $t_1, \dots, t_k \in T$ such that $M \xrightarrow{t_1, \dots, t_k} M_{\mathbf{A}}$ with $M_{\mathbf{A}} \in \mathbf{A}$. Moreover, let $K_{\mathcal{N}} \triangleq \max_{M \in \mathbf{R}(M)} (K_M)$.

Note that in safe Petri nets, both K_M and $K_{\mathcal{N}}$ are well-defined and finite. Obviously K_M must be finite for all reachable markings M of \mathcal{N} ; since \mathcal{N} is finite and safe, $K_{\mathcal{N}}$ is finite as well.

The following Theorem 1 states that *any* situation-fair execution of a safe net, i.e. round robin or other, will eventually leave any transient SCC and sooner or later enter a terminal SCC forever:

Theorem 1. Let $\sigma = M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} \dots$ be a situation-fair execution of a safe Petri net \mathcal{N} . Then either σ is finite and its final marking is a fixed point, or σ is infinite and there exists an attractor \mathbf{A} and $k \in \mathbb{N}$ such that $\forall i \in \mathbb{N} : M_{k+i} \in \mathbf{A}$.

Proof: By the definition of situation fairness, if σ is finite, its final marking must be a fixed point. Thus assume that σ is infinite; it then suffices to show that σ eventually reaches an attractor, since by definition no markings outside the attractor are reachable from there. Since σ is infinite and \mathcal{N} is safe, there must be at least one marking M that is visited infinitely often by σ . We shall prove that M is part of an attractor. Indeed, suppose this is *not* the case. Then M is part of a transient SCC, and by definition, some attractor \mathbf{A} is reachable from M . Thus, we have an executable path $M \xrightarrow{t_1} M_1 \xrightarrow{t_2} \dots \xrightarrow{t_n} M_n \in \mathbf{A}$, for $n = K_M$ and some attractor \mathbf{A} . Since σ is situation fair vis-à-vis t_1 in M_σ , it must visit M_1 infinitely often as well. Repeating this argument, we obtain that all M_k , for $k = 1, \dots, n$, must be visited infinitely often, too. But M is not reachable from M_n , which is a contradiction. Thus M must be part of an attractor. \square

We will now turn to the classification of states according to the long run behaviours available after them. That is, attractors may in general be desirable or undesirable; as long as the system still has some maximal behaviour available in which no *bad* state is reached, we will call it *free*, otherwise it is *doomed*. The next section will make these notions precise.

4. THE BAD, THE GOOD, THE DOOMED, AND THE FREE

Bad states. Our formal setting contains and extends the one presented in [GX05], specialized here to the 1-safe case. We assume that we are given a set of *bad markings* $\mathcal{M}_{\mathcal{B}} \subseteq \mathbf{R}(M_0)$, and write $\mathcal{M}_{\mathcal{G}} \triangleq \mathbf{R}(M_0) \setminus \mathcal{M}_{\mathcal{B}}$. Since we are interested in long-term behaviours, we happily adopt the assumption from [GX05] that $\mathcal{M}_{\mathcal{B}}$ is reachability-closed, i.e.

$M \in \mathcal{M}_{\mathcal{B}}$ and $M \rightarrow M'$ imply $M' \in \mathcal{M}_{\mathcal{B}}$.

Bad configurations. Define $\mathcal{C}_{\mathcal{B}} \triangleq \{C \in \mathcal{C}^{\mathbf{f}} : \text{Mark}(C) \in \mathcal{M}_{\mathcal{B}}\}$ as the set of *bad finite configurations*, and let $\mathcal{C}_{\mathcal{B}}^0$ be the set of configurations in $\mathcal{C}_{\mathcal{B}}$ that are contained in Π_0 . $\mathcal{C}_{\mathcal{B}} \subseteq \mathcal{C}$ is *absorbing* or *upward closed*, that is, for all $C_1 \in \mathcal{C}_{\mathcal{B}}$ and $C_2 \in \mathcal{C}^{\mathbf{f}}$ such that $C_1 \subseteq C_2$, one must have $C_2 \in \mathcal{C}_{\mathcal{B}}$.

This upward closure justifies the following extension of our terminology: let $C \in \mathcal{C}^{\infty}$; then C is bad iff there exists $C' \in \mathcal{C}_{\mathcal{B}}$ such that $C' \subseteq C$.

For any $C \in \mathcal{C}$, let $\Omega_C \triangleq \{\omega \in \Omega : C \subseteq \omega\}$ denote the maximal runs into which C can evolve. We are interested in those finite configurations all of whose extensions are ‘eventually bad’.

We will call such configurations *doomed*, since from them, the system cannot avoid entering a bad marking sooner or later (and from then on, all reachable markings are bad).

Definition 13. Configuration $C \in \mathcal{C}^{\mathbf{f}}$ is

(1) *strongly doomed* iff

$$\forall C^* \in \mathcal{C}^{\infty} : C \subseteq C^* \Rightarrow \exists C' \in \mathcal{C}_{\mathcal{B}} : C' \subseteq C^* \quad (4.1)$$

(2) *doomed* iff

$$\forall \omega^* \in \Omega : C \subseteq \omega \Rightarrow \exists C' \in \mathcal{C}_{\mathcal{B}} : C' \subseteq \omega^* \quad (4.2)$$

Denote the set of strongly doomed configurations by \mathcal{D}^* , that of doomed configurations by \mathcal{D} , and the set of minimal elements in \mathcal{D} by $\check{\mathcal{D}}$. We call finite configurations that are not doomed free. The set of free configurations is denoted by $\mathcal{F} \triangleq \mathcal{C}^{\mathbf{f}} \setminus \mathcal{D}$.

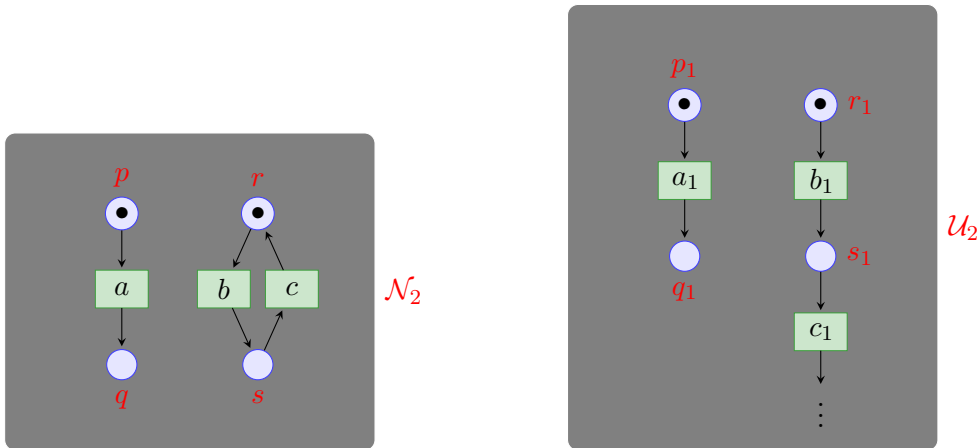


FIGURE 4. Left: Two safe Petri nets, \mathcal{N}_2 properly contains \mathcal{N}_1 ; right: their respective unfoldings \mathcal{U}_1 and \mathcal{U}_2

Remarks. Some comments are in order here, since i) Definition 13 introduces two different doomedness notions, and ii) clearly prefers the second over the first, in that the dual notion of freeness is defined without regard to *strong doom*. To understand the motivation behind this choice, one must first appreciate the difference between doom and strong doom. Consider the toy example shown in Figure 4, and suppose in both nets, a marking is bad iff it contains place q . Then the initial marking $\{p\}$ of \mathcal{N}_1 is clearly both doomed and strongly doomed. However, marking $M_0 = \{p, r\}$ and the configuration \emptyset ‘leading to’ M_0 in \mathcal{N}_2 is doomed, since the only maximal configuration of \mathcal{U}_2 contains a_1 . Nevertheless, \emptyset is not strongly doomed, since with $C \triangleq \{b_i, c_i : i \in \mathbb{N}\}$ we have $\emptyset \subsetneq C \in \mathcal{C}^\infty$, yet no finite configuration contained in C is bad.

The difference in the two notions of doom thus lies in a semantic assumption of progress, or *weak fairness*. While it is interesting in its own right, and potentially a subject for future work, to study the ramifications of the theory that build on the absence of progress, and hence on the notion of ‘*strong doom*’, we choose here to focus on the notion of doom given in (4.2), and to consider those configurations that are *doomed* to be configurations to avoid. Once one assumes that the enabled transition a , that cannot be enabled, will fatally eventually fire, one is indeed led to seeing \emptyset in Figure 4 as fatal, and as a state to avoid in any application.

Returning to the running example of Figures 1a and 2, if we consider $\mathcal{M}_B \triangleq \{\{p_8\}\}$, then configuration $C_1 \triangleq \{\alpha_1\}$ is free, and $C_2 \triangleq C_1 \cup \{\gamma_1\}$ is doomed because $C_3 \triangleq C_2 \cup \{\xi_1\}$ is bad.

By abuse of terminology, we will call any marking M *free* if there is a free $C \in \mathcal{C}^f$ such that $\text{Mark}(C) = M$. In the following, we assume that the *badness problem* ‘ $M \in \mathcal{M}_B$?’ has been decided for every reachable marking M . In fact, a typical badness criteria can be easily decided, e.g. by presence or absence of a fixed submarking. We will not dwell on the complexity of this decision problem as it is outside the scope of this article.

Since all bad markings are automatically doomed, we need to classify good markings into the doomed ones on the one hand, and the free ones on the other; the next two chapters will address this problem.

5. VERIFICATION OF FREENESS

In order to check whether a given marking M is partially free, the key question is whether it is possible to reach, from M , some $M' \in \mathcal{M}_G$ such that M' admits a non-empty firing sequence σ with $M' \xrightarrow{\sigma} M'$. Indeed, under the assumption that \mathcal{N} is *deadlock-free* in the sense that no maximal configuration contains any maximal event, it is easy to see that the existence of such an M' is equivalent to partial freedom of M . Of course, deadlock-freeness can always be obtained by adding dummy loop transitions to any deadlocked partial marking; the assumption therefore means no loss of generality. Note that the classical algorithms for finding loops in a *transition system* will not help us here, since the size of the state graph of most Petri nets we endeavour is prohibitive; we will adapt unfolding-based methods following [EH08]. A procedure that, for any reachable marking M , generalizes that for McMillan’s complete prefix from M , will produce a sufficient data structure for checking freedom of M .

5.1. Search for loops. The key for obtaining this is obviously the power to identify loops in the reachability relation. We need to catch such loops at the earliest possible opportunity, in a small prefix.

Definition 14. $C_1, C_2 \in \mathcal{C}^f$ are marking equivalent, written $C_1 \sim_M C_2$, iff $\text{Mark}(C_1) = \text{Mark}(C_2)$. Say that C_1 and C_2 form a loop, written $\text{loop}(C_1, C_2)$, iff

- (1) they exhibit a cycle in the state graph, i.e. $C_1 \sim_M C_2$ and $C_1 \subsetneq C_2$, and
- (2) there are no configurations between C_1 and C_2 that exhibit such a cycle, i.e. there do not exist $C_3, C_4 \in \mathcal{C}^f$ such that
 - (a) $C_1 \subseteq C_3 \subsetneq C_4 \subsetneq C_2$
 - (b) $C_3 \sim_M C_4$
 - (c) $\{C_1, C_2\} \neq \{C_3, C_4\}$.

Write $\text{mloop}(C_1, C_2)$ iff C_1 and C_2 form a minimal loop wrt inclusion, i.e. iff

- (a) $\text{loop}(C_1, C_2)$
- (b) for $C'_1, C'_2 \in \mathcal{C}^f$, $C'_1 \subsetneq C_1$ and $C'_2 \subseteq C_2$ together imply that $\neg \text{loop}(C'_1, C'_2)$.

We have:

Lemma 2. Every loop-free configuration $C_M \in \mathcal{C}^f$ such that $\text{Mark}(C_M) = M$ is in McMillan's ([McM92]) prefix $\Pi_0^{McM}(M) \triangleq \Pi_0^{\subseteq}(M)$.

Proof: Follows from the construction of $\Pi_0^{McM}(M)$ since configurations are only truncated by the cutoff criterion when they produce a loop. \square

Lemma 3. $\forall C_1, C_2 \in \mathcal{C}^f(M_0)$: if $\text{mloop}(C_1, C_2)$, then C_2 is a configuration of $\Pi_0^{McM}(M)$ (and a fortiori, so is C_1).

Proof: Suppose that under the assumptions of the lemma, C_2 is not a configuration of $\Pi_0^{McM}(M)$. In that case, it must contain $C_2^a \subseteq C_2^b \subsetneq C_2$ such that $\text{loop}(C_2^a, C_2^b)$. But $\text{mloop}(C_1, C_2)$ implies this is impossible unless $C_1 = C_2^a$ and $C_2 = C_2^b$. But if this is the smallest loop in C_2 , then C_2 must be in $\Pi_0^{McM}(M)$. \square

Lemma 4. For every marking sequence $M \xrightarrow{t_1} \dots M$ that contains no properly smaller loop, there is a configuration C_M in $\Pi_0^{McM}(M)$ such that $\text{mloop}(\emptyset, C_M)$.

Proof: A consequence of the definition of $\Pi_0^{McM}(M)$ and of Lemma 3. \square

5.2. Verification of Freeness. By the above discussion, inspection of $\Pi_0^{\subseteq}(M)$ yields all loops reachable from M ; M is free if any of them is good, i.e. such that the loop contains no bad marking. Equivalently, a loop (C_1, C_2) is good iff $\text{Mark}(C_1) = \text{Mark}(C_2) \notin \mathcal{M}_{\mathcal{B}}$. Concretely, it suffices to consider every maximal event e of $\Pi_0^{\subseteq}(M)$, since all these e must have a mirror event $e' < e$ such that $\text{mloop}([e'], [e])$. If $\text{Mark}(e)$ is good, so is the entire loop.

This informal algorithm can be sped up by declaring, in addition to loop-cutoffs, any event e such that $\text{Mark}([e]) \in \mathcal{M}_{\mathcal{B}}$ (or equivalently, $[e] \in \mathcal{C}_{\mathcal{B}}$) as a cutoff event in the unfolding procedure. To decide if $\text{Mark}([e]) \in \mathcal{M}_{\mathcal{B}}$, we take advantage of $\mathcal{M}_{\mathcal{B}}$'s reachability closure; all the initially known bad markings are used in \mathcal{N} (referred as the bad net, $\mathcal{N}_{\mathcal{B}}$) to unfold it so that we can check reachability and test whether a marking is bad or not, i.e., if $\text{Mark}([e]) \in \mathcal{U}_{\mathcal{N}_{\mathcal{B}}}$ then $\text{Mark}([e])$ is bad, and hence e is a cutoff. The resulting prefix is slightly, sometimes considerably smaller than the full McMillan prefix; the worst case size

of the latter remains, however, to be taken into account. Below, this algorithm is assumed invoked by the boolean function $\text{FREECHECK}(M)$ that outputs TRUE iff marking M is free.

6. CLIFF-EDGES AND RIDGES

Recall that every reachable marking is represented by at least one configuration of the unfolding. Moreover, since the future evolution of \mathcal{N} depends only on the current marking, $\text{Mark}(C_1) = \text{Mark}(C_2)$ for two configurations C_1 and C_2 implies that either both C_1 and C_2 are free, or both are doomed. Therefore, by extension, we call $\text{Mark}(C)$ free or doomed whenever C is.

Running Example. In the context of Figures 1a and 2, we consider $\mathcal{M}_{\mathcal{B}}$ the singleton set containing the marking $M_8 = \{P_8\}$. Clearly, $C_1 = \{\alpha_1, \gamma_1, \xi_1\}$ and $C_2 = \{\beta_1, \delta_1, \eta_1\}$ satisfy $\text{Mark}(C_1) = \text{Mark}(C_2) = M_8$ and therefore $C_1, C_2 \in \mathcal{C}_{\mathcal{B}}$. But note that $C'_1 = \{\alpha_1, \gamma_1\}$ and $C'_2 = \{\beta_1, \delta_1\}$ produce markings outside $\mathcal{M}_{\mathcal{B}}$, but they are doomed since any extension of these configurations leads into $\mathcal{M}_{\mathcal{B}}$. Therefore, $C'_1, C'_2 \in \mathcal{C}_{\mathcal{B}}$. On the other hand, \emptyset is free, as well as $\{\beta_1, \gamma_1\}$, $\{\alpha_1, \delta_1\}$, etc. We note in passing that the Petri net in Fig 1a allows to refine the understanding of the ‘tipping point’ by showing that doom is not brought about by a single transition but rather the combined effect of two independent choices; this fact is obscured, or at least far from obvious, in the state graph shown in Figure 1b.

Identifying free and doomed configurations belongs to the core objectives of this paper.

From the minimal doomed configurations, we derive the critical ‘points’ at which a run becomes doomed:

Definition 15. *An event set $\gamma \subseteq E$ is called a cliff-edge iff there exists a minimally doomed configuration $C \in \tilde{\mathcal{D}}$ such that $\gamma = \mathbf{crest}(C)$. The set of cliff-edges is denoted Γ . The folding $\chi \triangleq \pi(\gamma) \subseteq T$ of a cliff-edge γ is called a ridge.*

To complete the map of the evolutionary landscape for \mathcal{N} , it is important to find, in a bounded prefix of the unfolding, all ridges that determine the viability of a trajectory. Notice that the completeness of prefix Π_0 only guarantees that all reachable *markings* of \mathcal{N} are represented by at least one configuration of Π_0 ; this does not extend to a guarantee that all concurrent steps that lead into a doomed marking can be found in Π_0 as well. Fortunately, one has:

Lemma 6.1. *For every ridge χ of \mathcal{N} there is a witness in Π_0^{\subseteq} , i.e. there exists a minimally doomed configuration C in Π_0^{\subseteq} such that $\pi(\mathbf{crest}(C)) = \chi$.*

Proof: Fix χ , and let C_{χ} be any configuration such that $\pi(\mathbf{crest}(C_{\chi})) = \chi$; set $M^C \triangleq \text{Mark}(C_{\chi})$, and let M_{χ}^C the unique reachable marking such that $M_{\chi}^C \xrightarrow{\chi} M^C$. Then any such M_{χ}^C is represented by some C^{χ} in Π_0^{\subseteq} by completeness. \square

7. FINDING MINIMALLY DOOMED CONFIGURATIONS

In the light of the above, we need to proceed in two steps. First, a configuration whose end events are without any immediate conflict - we will call such events *unchallenged* - cannot be minimal; we will thus first describe how to *shave* given configurations in order to approximate minimally doomed ones contained in them. Then we need to check whether

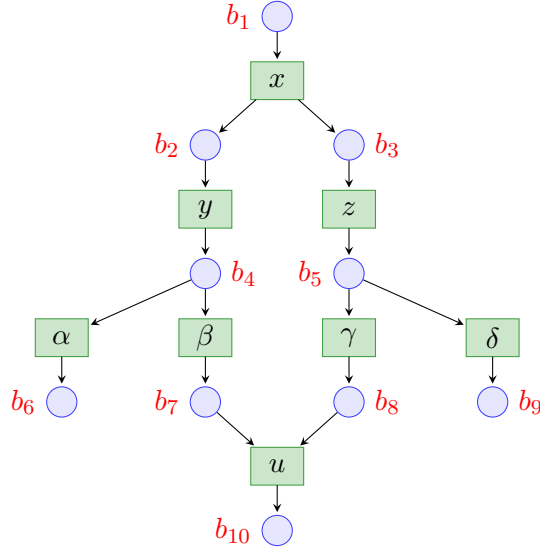


FIGURE 5. An occurrence net. With $C \triangleq \{x, y, z, \beta, \gamma\}$ and $C' \triangleq C \cup \{u\}$, suppose $\mathcal{M}_B = \{\text{Mark}(C')\} = \pi(\{b_{10}\})$. Then $\text{shave}(C') = C$, and C is doomed. Moreover, $C \in \check{\mathcal{D}}$ since both $C_3 \triangleq C \setminus \{\beta\}$ and $C_4 \triangleq C \setminus \{\gamma\}$ are free.

a given configuration is doomed or free. The algorithm MINDOO will then combine both functions into a search for minimally doomed configurations.

7.1. Preparations: Shaving. Let us start by observing that \mathcal{C}_B , an upward closed set by construction, also has some downward closure properties, meaning one can restrict control to act on ‘small’ configurations. The first idea is to remove maximal events e from a configuration C if they are not involved in any direct conflict; the idea is that in such a case, the reduced configuration $C \setminus \{e\}$ has exactly the same maximal extensions as C . In fact, $C \setminus \{e\}$ *reveals* e in a sense made precise in [BCH13, Haa10, HKS13, HRS13]; however, we will not be using exactly the relations introduced there.

Definition 16. An event e is *unchallenged* iff there is no e' such that $e \#_\delta e'$, i.e. $(\bullet e)^\bullet = \{e\}$.

Lemma 7.1. Let $C \in \mathcal{C}^f$ and $e \in \text{crest}(C)$ unchallenged; set $C' \triangleq C \setminus \{e\}$. Then $C' \in \mathcal{C}^f$, and $\Omega_C = \Omega_{C'}$.

Proof: $C' \in \mathcal{C}^f$ holds by construction. Also, $\Omega_C \subseteq \Omega_{C'}$ follows from $C' \subseteq C$; it remains to show the reverse inclusion. Assume there exists $\omega \in \Omega_{C'} \setminus \Omega_C$; then $C \setminus \omega = \{e\}$, and $\langle e \rangle \subseteq \omega$. By maximality, ω must contain some e' such that $e \# e'$. Then by definition, there are events $u \neq v$, $u \leq e$, $v \leq e'$, and $u \#_\delta v$. In particular, $u \# e'$, and since $\{e'\} \cup \langle e \rangle \subseteq \omega$, this implies $u = e$. But e is unchallenged, so v cannot exist, and neither can ω .

□

Definition 17. A configuration $C \in \mathcal{C}^f$ such that $\mathbf{crest}(C)$ contains no unchallenged event is called *shaved*.

Clearly, every $C \in \mathcal{C}^f$ contains a unique maximal shaved configuration, which we call $\mathbf{shave}(C)$; it can be obtained from C by recursively ‘shaving away’ any unchallenged $e \in \mathbf{crest}(C)$, and then continuing with the new crest, until no unchallenged events remain.

Example. In the context of Figure 5, for $C_1 = \{x, y, z\}$ and $C_2 = C_1 \cup \{\beta, \gamma, u\}$, one has $\mathbf{shave}(C_1) = \emptyset$ since x, y , and z are unchallenged, and $\mathbf{shave}(C_2) = C_1 \cup \{\beta, \gamma\}$ since u is unchallenged but neither β nor γ are. Note that in the unfolding of the running example shown in Figure 2, the κ -labeled events are the only unchallenged ones.

As a consequence of Lemma 7.1, any $C \in \mathcal{C}^f$ is in \mathcal{C}_B iff $\mathbf{shave}(C)$ is. Still, it may be possible that such a $\mathbf{shave}(C)$ can still be reduced further by removing some of its crest events. This would be the case, e.g., if two conflicting events both lead to a bad state. Thus, given a crest event e , we test whether $C \setminus \{e\}$ is free (e.g. because some event in conflict with e may allow to move away from doom) or still doomed. If the latter is the case, then C was not minimally doomed, and analysis continues with $C \setminus \{e\}$ (we say that we ‘rub away’ e). If $C \setminus \{e\}$ is free, we leave e in place and test the remaining events from $\mathbf{crest}(C)$. A configuration that is shaved and from which no event can be rubbed away is minimally doomed.

7.2. Algorithm MinDoo. Algorithm 1 uses a ‘worklist’ set \mathbf{wl} of doomed, shaved configurations to be explored; \mathbf{wl} is modified when a configuration is replaced by a set of rubbed (and again, shaved) versions of itself, or when a configuration C is identified as minimally doomed, in which case it is removed from \mathbf{wl} and added to \mathfrak{D} .

Every branch of MINDOO stops when a minimally doomed configuration is reached, i.e., a doomed configuration C such by rubbing off any crest event e from C makes it free, i.e. $C \setminus \{e\}$ is free for all $e \in \mathbf{crest}(C)$. When the worklist is empty, all minimally doomed configurations have been collected in \mathfrak{D} . Note that if $\emptyset \in \mathbf{wl}$ at any stage during the execution of Algorithm MINDOO, then \emptyset will be added to \mathfrak{D} , since MINDOO will not enter the second **foreach**-loop in that case. In fact, if this situation arises, *every* configuration is doomed, and thus \emptyset is the unique minimally doomed configuration.

The configurations produced in the course of the search strictly decrease w.r.t both size and inclusion. Moreover, an upper bound on the prefixes explored at each step is given by \mathcal{C}_B , itself strictly contained in the complete finite prefix used to find all bad markings. According to [ERV02], this prefix can be chosen of size equal or smaller (typically: considerably smaller) than the reachability graph of \mathcal{N} (number of non-cut-off events in the prefix are less or equal than the number of reachable markings in \mathcal{N}).

Theorem 7.2. For any safe Petri net $\mathcal{N} = \langle N, M_0 \rangle$ and bad states set $\mathcal{M}_B \subseteq \mathbf{R}_N(M_0)$, Algorithm MINDOO terminates, with output set \mathfrak{D} containing exactly all minimal doomed configurations, i.e. $\mathfrak{D} = \tilde{\mathfrak{D}}$.

Proof: Termination follows from the finiteness of $\min_{\subseteq}(\mathcal{C}_{B_0})$, since in each round of MINDOO there is one configuration C that is either replaced by a set of strict prefixes or removed from \mathbf{wl} . Therefore, after a finite number of steps, \mathbf{wl} is empty.

As shown in Section 5.1, the status (doomed or free) of a given finite configuration can effectively be checked on a fixed finite prefix of \mathcal{U} . Assume that after termination of MINDOO, one has $C \in \mathfrak{D}$; we need to show $C \in \tilde{\mathfrak{D}}$. Clearly, when C was added to

Algorithm 1: Algorithm MINDOO

Data: Complete prefix Π_0 of safe Petri Net $\mathcal{N} = \langle P, T, F, M_0 \rangle$ and the set \mathcal{C}_B^0 of bad configurations of Π_0
Result: The set \mathfrak{D} of Π_0 's \subseteq -minimal doomed configurations
 $\mathfrak{D} \leftarrow \emptyset$; $\text{wl} \leftarrow \emptyset$;
foreach $C \in \min_{\subseteq}(\mathcal{C}_B^0)$ **do**
 $C' \leftarrow \text{shave}(C)$;
 $\text{wl} \leftarrow \text{wl} \cup \{C'\}$;
end
while $\text{wl} \neq \emptyset$ **do**
 Pick $C \in \text{wl}$; $\text{add} \leftarrow \text{true}$;
 if *NOT* $\text{FREECHECK}(\text{Mark}(C \setminus \text{crest}(C)))$ **then**
 $\text{add} \leftarrow \text{false}$;
 $C' \leftarrow \text{shave}(C \setminus \text{crest}(C))$;
 $\text{wl} \leftarrow (\text{wl} \cup \{C'\})$;
 else
 foreach $e \in \text{crest}(C)$ **do**
 if *NOT* $\text{FREECHECK}(\text{Mark}((C \setminus \{e\})))$ **then**
 $\text{add} \leftarrow \text{false}$;
 $C' \leftarrow \text{shave}(C \setminus \{e\})$;
 $\text{wl} \leftarrow \text{wl} \cup \{C'\}$;
 end
 end
 end
 $\text{wl} \leftarrow \text{wl} \setminus \{C\}$;
 if add **then**
 $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{C\}$;
 end
end
return \mathfrak{D}

\mathfrak{D} , it had been detected as doomed; it remains to show that C is also minimal with this property. Assume that there is $C' \subsetneq C$ that is doomed as well. But in that case there exists $e \in \text{crest}(C)$ such that $C' \subseteq (C \setminus \{e\}) \subsetneq C$, which implies that this $(C \setminus \{e\})$ is doomed as well. But then add has been set to **false** in the second **foreach**-loop, before C could have been added to \mathfrak{D} .

Conversely, let $C \in \check{\mathfrak{D}}$. Then $(C \setminus \{e\})$ is free for all $e \in \text{crest}(C)$; the variable add remains thus at the value **true** because no round of the second **foreach**-loop can flip it. Thus C is added to \mathfrak{D} , from which MINDOO never removes any configuration. \square

7.3. Implementation and Experiments. An implementation of MINDOO is available at [AS24] using the module DOOMED. It takes as input a safe Petri net in the PEP [Ste04] format, the list of undesired (or bad) markings separated by newlines and the unfolding of the bad net, it relies on ECOFOLDER [AS24] for computing the complete finite prefix

TABLE 1. Statistics of Algorithm 1 on Petri net models of biological systems. $|P|$ is the number of places in the system; $|T|$ is the number of transitions; the size of Π_0 is the number of their events; $|\mathcal{M}_B|$ is the number of bad markings; $|\mathbf{wl}|$ is the number of bad configurations initially identified; “# free checks” is the number of freeness checks for *free* status of a configuration. “time” is the total computation time on a 1.8Ghz CPU.

Model	$ P $	$ T $	size Π_0	$ \mathcal{M}_B $	$ \mathbf{wl} $	$ \check{\mathcal{D}} $	# free checks	time
Lambda switch	11	41	126	1	5	5	13	1s
Mammalian cell cycle	20	38	176	1	25	0	78	1s
Cell death receptor	22	33	791	1	16	14	146	9m57s
Budding yeast cell cycle	18	32	1,413	1	30	28	165	2m28s

Π_0 of both the system and the bad net. Also, ECOFOLDER is used to apply McMillan’s criterion [McM92] to unfold every corresponding system’s net initialized with a marking of a configuration’s crest choosen from \mathbf{wl} . In the unfolding process, we perform reachability checks using Answer-Set programming (ASP) to know whether the marking is reachable in the bad net, then decide its freeness status. We have implemented Algorithm 1 in Python, using ASP for the identification of bad configurations, specifically with the help of the CLINGO solver [GKKS14].

Table 1 illustrates the performance of the implementation on different instances of Petri nets modeling biological processes. In each case, we report the number of places and transitions, the size (number of events) of prefix Π_0 (including cut-off events), the number of *bad* markings initially given to the algorithm ($|\mathcal{M}_B|$), the number of bad configurations in \mathbf{wl} leading to the given bad markings, the number of minimally doomed configurations ($|\check{\mathcal{D}}|$), the number of configurations which have been tested for being free, and the total time. The aim of the experiments conducted in this study was to investigate the feasibility and effectiveness of our approach for analyzing standard models of biological systems from the literature. Specifically, we focused on models for which the study of doomed configurations was relevant, as these configurations can provide important insights into the behavior and properties of the system. Potential bottlenecks include the computation of those maximal configurations that lead to a *bad* marking, and most importantly the unfolding process using McMillan’s cutoff criterion [McM93] for the freeness test since, as shown in [ERV02], it can be exponentially larger than the number of reachable markings of the net. Hereafter, our experiments focused on evaluating the impact of different number of places and transitions, and prefix sizes (Π_0) on several metrics, including the number of minimally doomed configurations, the number of candidate configurations screened by Algorithm 1 (# of free checks in Table 1), and the overall computation time as the prefix size and the number of nodes in the net increased. We have selected four models that had been initially published as Boolean networks, which can be translated into equivalent safe Petri nets using the encoding described in [CHJ⁺14] and implemented in the tool PINT [Pau17].

The “Lambda switch” model [TT95] comprises 11 places and 41 transitions, and is a gene regulatory network for a bacterial virus known as the lambda phage. This virus is a type of temperate bacteriophage, which means it can establish a long-term symbiotic relationship with its bacterial host, known as the *lysogenic* response. In this state, the virus is faithfully transmitted to the bacterial progeny. However, in most cases, the virus follows

the *lytic* response, where it replicates itself, destroys the host cell, and ultimately lyses the cell. This dichotomous decision is cell-dependent and is regulated by intertwined feedback mechanisms involving four genes: *cI*, *cro*, *cII*, and *N*. In our experiments, we define the lytic response as “bad” or undesired, and initialize the *cro* gene with a token while leaving all other genes without tokens.

The “Mammalian cell cycle” model [FNCT06] comprises 20 places and 38 transitions, reproducing the main known dynamical features of the wild-type biological system. Mammalian cell division is a highly regulated process that must be coordinated with the overall growth and development of the organism. This coordination is necessary to ensure that cell division occurs only when needed, such as during tissue repair or in response to hormonal signals. The decision of whether a cell will divide or remain in a resting state (known as quiescence or G0 phase) is determined by a complex interplay of extracellular positive and negative signals. These signals can include growth factors, cytokines, and other signaling molecules that bind to specific receptors on the cell surface. The balance of these signals ultimately determines whether the cell will enter the cell cycle or arrest the process; one of the factors causing cell arrest is disruption of Cyclin D (CycD) and its associated CDKs since they are vital for the transition from G1 to S phase. Therefore, we determine CycD disruption as a bad marking resulting in an empty set of minimally doomed configurations. In other words, the only configuration in the prefix that can avoid cell arrest is the empty set, a finding that is suggestive in itself.

The “Cell death receptor” model [CTF⁺10] comprises 22 places and 33 transitions, and reproduces a bifurcation process into different cell fates, one of which has been declared as bad (apoptosis). The model focuses on the activation of death receptors (TNF and FAS) in various cell types and conditions. The cell’s fate can vary significantly as the same signal can trigger survival by activating the NFkB signaling pathway or lead to death by apoptosis or necrosis. The study reveals the complex interplay and mutual inhibition between the NFkB pro-survival, RIP1-dependent necrosis, and apoptosis pathways. Our analysis shows that the minimally doomed configurations identify the configurations in which a decisive event has occurred, committing the system to the undesirable attractor marked as bad. By detecting these configurations, biologists gain insight into the causal steps that lead to cell death via apoptosis, and identify decisive points where alternative pathways are still possible.

The “Budding yeast cell cycle” model [OLB⁺08] comprises 18 places and 32 transitions, capturing the oscillatory behavior of gene activity throughout the cell cycle. The biochemical oscillator controlling periodic events during the cell cycle is centered on the activity of cyclin-dependent kinases (CDKs), which are thought to play a crucial role in controlling the temporally ordered program of transcription in somatic cells and yeast. However, the study [OLB⁺08] had integrated genome-wide transcription data and built models, in which periodic transcription emerges as a property of a transcription factor network. The authors investigated the dynamics of genome-wide transcription in budding yeast cells disrupted for all S-phase and mitotic cyclins to determine the extent to which CDKs and transcription factor networks contribute to global regulation of the cell-cycle transcription program. In our analysis, we use this model to identify the minimally doomed configurations that lead to a bad marking, in which the cycle exits its oscillatory behavior and all genes become inactive. By detecting these configurations, we can precisely identify the underlying factors that lead to the system exiting its oscillatory behavior.

In each case, the number of minimally doomed configurations is a fraction of the size of the finite complete prefix Π_0 . The computation time for identifying minimally doomed configurations is primarily affected, as previously mentioned, by the potential bottleneck of checking for loops using McMillan's cutoff criterion. In general, as the number of places increases, it naturally becomes more difficult to determine whether a marking is reachable from a bad one. However, the particularity of the freeness check lies in the fact that one needs to unfold McMillan's prefix in order to be sure to have detected all cases in which a loop exists. In the worst case, this prefix may be exponentially larger than the reachability graph; and in any case, one has to explore individual branches to a considerable depth, leading in turn to exponential branching in the prefix constructed.

Future work may explore ways to handle this bottleneck more efficiently, by accelerating computation speed for loop checks, particularly in hard and complex examples.

Another relevant challenge is to find compact representations of the set of minimally doomed configurations; indeed, as these configurations often share many events, biological interpretation may be facilitated by regrouping them in a helpful way.

8. PROTECTEDNESS

8.1. Measuring the Distance from Doom.

Decisional Height. With the above, we have the tools to draw a map of the 'landscape' in which the system evolves, with doomed zones and cliff-edges highlighted. What we wish to add now is to assist *navigation* in this landscape: we intend to give a meaningful measure of how well, or badly, a current system state is protected against falling from a cliff-edge, that is, how far the system is from entering a doomed state. We have chosen to measure this distance in terms not of the *length* of paths, or of similar notions, but rather in terms of the *choices* that are made by the system in following a particular path.

Consider a configuration C and the non-sequential process that it represents. Some of the events in C can be seen as representing a *decision*, in the sense that their occurrence took place in conflict with some event that was enabled by some prefix of C . The number of such events gives a measure of the information contained in C , in terms of the decisions necessary to obtain C :

Definition 18. Let $C \in \mathcal{C}^f$, and define

$$\mathbf{dech}(C) \triangleq |\{e \in C : \exists e' \in E : e \#_\sigma^C e'\}|,$$

where $\#_\sigma^C$ is the strict C -conflict relation defined, for all $e \in C$, by

$$e \#_\sigma^C e' \triangleLeftrightarrow e \#_\delta e' \wedge \langle e' \rangle \subseteq C.$$

$\mathbf{dech}(C)$ is called the *decisional height* of C .

In Figure 2, the configuration $C_1 = \{\xi_1, \alpha_1, \gamma_1\}$ satisfies $\mathbf{dech}(C_1) = 2$, whereas for $C_0 = \{\beta_1\}$, one has $\mathbf{dech}(C_0) = 1$.

Figure 6 shows an occurrence net with configuration $C_\beta \triangleq \{x, y, \beta\}$ that has $\mathbf{dech}(C_\beta) = 3$ (because of z, α , and γ) and configuration $C_\alpha \triangleq \{x, z, \alpha\}$ with $\mathbf{dech}(C_\alpha) = 1$ (because of y).

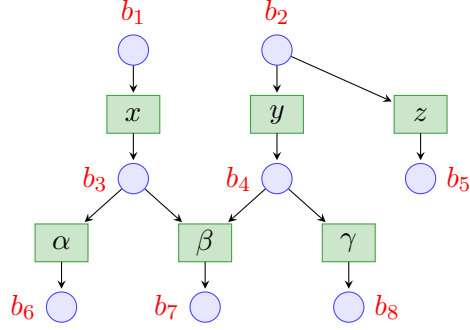


FIGURE 6. Illustration of direct conflict.

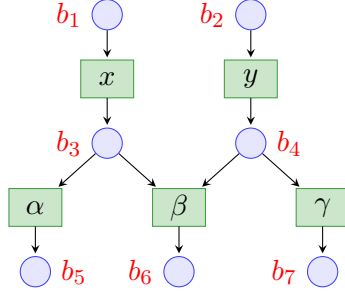


FIGURE 7. An ‘unprotected’ branching process

Note that $\#_\sigma^C$ is more restrictive than direct conflict $\#_\delta$; it is also more restrictive than the *immediate conflict* in the literature (e.g. [AB06]). It is closely dependent on the configuration C under study, and describes precisely those events *against* which the process had to decide in performing C .

One may wonder why we choose this particular definition of *decision*, rather than using the customary direct or immediate conflicts. The reason is that we wish to consider as decisions only deliberate actions against or in favor particular branches in a bifurcation situation, and *not* any resolution of conflicts brought about by the nondeterminism in the ‘race’ between two concurrent processes. Consider Figure 7, assuming that the only bad configuration is $C_{\mathcal{C}_B} = \{\alpha, \gamma, x, y\}$. Then clearly, the configurations $C_\alpha \triangleq \{\alpha, x\}$ and $C_\gamma \triangleq \{\gamma, y\}$ are both doomed. Their height, measured by either direct or immediate conflict, would be 1; in our definition, it is 0, since α (or γ , respectively) was enabled by $C_\alpha \setminus \{\alpha\}$ (or $C_\gamma \setminus \{\gamma\}$) without any conflict, since neither $C_\alpha \setminus \{\alpha\}$ nor $C_\gamma \setminus \{\gamma\}$ enabled β . Underlying this is the fact that the ‘decision’ against β is taken here without any choice, merely by the fact that of the two concurrent events x and y , one may occur much faster than the other, creating a situation in which α (or γ) has no competitor.

Defining Protectedness. Now, for any free marking M (or, equivalently, any free configuration C such that $\text{Mark}(C) = M$), we wish to measure the threat represented by doomed markings reachable from M : how far away from doom is the system when it is in M ? Using the decisional height introduced above, we can define a height difference in terms of the conflicts that lead from one marking to another:

Definition 19. For $C \in \mathcal{C}^f$, let

$$\check{\mathcal{D}}_C \triangleq \begin{cases} \{C' \in \check{\mathcal{D}} : C \subseteq C'\} & : C \in \mathcal{F} \\ \{C\} & : C \in \mathcal{C}_B \end{cases} \quad (8.1)$$

The protectedness of C is then

$$\mathbf{prot}(C) \triangleq \min_{C' \in \check{\mathcal{D}}_C} \{\mathbf{dech}(C' \setminus C)\} \quad (8.2)$$

In Figure 5, with the definitions introduced there, $\mathbf{prot}(C) = \mathbf{prot}(C') = 0$. Setting $C_1 \triangleq \{x\}$, $C_2 \triangleq \{x, y\}$, $C_3 \triangleq \{x, z\}$, $C_4 \triangleq C_2 \cup C_3$, $C_5 \triangleq C_4 \cup \{\beta\}$, $C_6 \triangleq C_4 \cup \{\gamma\}$, one further has

$$\begin{aligned} \mathbf{prot}(C_1) = \mathbf{prot}(C_2) = \mathbf{prot}(C_3) = \mathbf{prot}(C_4) &= 2 \\ \mathbf{prot}(C_5) = \mathbf{prot}(C_6) &= 1. \end{aligned}$$

Returning to Figure 6, suppose that $C' = \{x, y, \beta\}$ is the only minimally doomed configuration. Then for $C = \{x, z, \alpha\}$ as above, we have $\mathbf{prot}(C) = 1$, because the only strict (and direct) conflict here is the one between z and y .

Note that the definition of protectedness is parametrized by the choice of conflict relation in computing $\mathbf{dech}(\bullet)$. Using direct conflict instead of strict conflict would increase $\mathbf{dech}(\bullet)$ and lead to an overevaluation of protectedness.

To see the point, consider the occurrence net in Figure 6. Let $C_\alpha = \{x, z, \alpha\}$, $C_\beta = \{x, y, \beta\}$ and $C_\gamma = \{x, y, \alpha, \gamma\}$. We have $\mathbf{dech}(C_\alpha) = 1$, $\mathbf{dech}(C_\beta) = 3$ and $\mathbf{dech}(C_\gamma) = 2$. Were $\#_\sigma$ replaced by $\#_\delta$ in the computation of $\mathbf{dech}(\bullet)$, these values would not change *except* for C_α where it would change to 2. As a result, if $C \in \check{\mathcal{D}}$, the protectedness of the empty configuration would be evaluated as 2, whereas by our definition $\mathbf{prot}(\emptyset) = 1$. Indeed, \emptyset is *just one wrong decision away from doom*, and this is what protectness is meant to express.

An even starker illustration is once again provided by Figure 7: in fact, we have $\mathbf{prot}(\emptyset) = 0$, as follows from the discussion above.

8.2. Computing Protectedness is Feasible. Computation of $\mathbf{prot}(\bullet)$ does not require any larger data structure than those already required for computing $\check{\mathcal{D}}$. In fact, an alternative - and often much smaller prefix - is also sufficient:

Lemma 8.1. *There exists an adequate total order \prec such that the associate complete prefix scheme producing Π_0^\prec whose size is bounded by the number of reachable markings, and such that for every finite configuration C , $\mathbf{prot}(C)$ can be computed on $\Pi_0^\prec(\text{Mark}(C))$.*

Proof: If $\check{\mathcal{D}} \cap \mathcal{C}(\Pi_0) = \emptyset$, then all extensions of C are free, and we are done. Otherwise, we have to find an adequate *total* order \prec on finite configurations, that ensures that Π_0^\prec contains at least one minimally doomed configuration that minimizes $\mathbf{dech}(\bullet)$ over all minimally doomed configurations in $\mathcal{U}(\text{Mark}(C))$. The following order \prec is obtained by

modifying the total order \prec_F introduced in [ERV02], Def. 6.2.: For $C_1, C_2 \in \mathcal{C}^f$, write $C_1 \prec C_2$ iff either

- $\mathbf{dech}(C_1) < \mathbf{dech}(C_2)$, or
- $\mathbf{dech}(C_1) = \mathbf{dech}(C_2)$ and $C_1 \ll C_2$, or
- $\mathbf{dech}(C_1) = \mathbf{dech}(C_2)$ and $C_1 \equiv C_2$, and $FC(C_1) \ll FC(C_2)$,

where $\ll (\equiv)$ denote lexicographic ordering (lexicographic equivalence) wrt some total ordering of the transition set T , and FC denotes Cartier-Foata normal form. The proof of Theorem 6.4. of [ERV02] extends immediately, proving that \prec is an adequate total order; therefore, Lemma 5.3. of [ERV02] applies, hence any complete prefix Π_0^\prec obtained via the scheme using \prec is bounded in size by the reachability graph. Now, let \mathcal{C}^* be the set of configurations from $\check{D}(\text{Mark}(C))$ that minimize $\mathbf{dech}(\bullet)$; by construction of \prec , one has $\mathcal{C}^* \cap \mathcal{C}(\Pi_0^\prec) \neq \emptyset$. \square

9. DISCUSSION

The results presented here contain, extend and complete those in our conference paper [AHP⁺22]. The toolkit for the analysis of tipping situations in a safe Petri net, i.e. when and how a basin boundary is crossed; an algorithmic method for finding minimally doomed configuration has been developed, implemented and tested.

Moreover, we have introduced a measure of *protectedness* that indicates the number of *decisions* that separate a free state from doom. It uses an intrinsic notion of decisional height that allows to warn about impending dangerous scenarios; at the same time, this height is also ‘natural’ for unfoldings, in the sense that it induces an adequate linear order that allows to compute complete prefixes of bounded size.

On a more general level, the results here are part of a broader effort to provide a discrete, Petri-net based framework for dynamical systems analysis in the life sciences. The applications that we target here lie in systems biology and ecology.

Future work will investigate possibilities for *Doom Avoidance Control*, i.e. devising strategies that allow to steer away from doom; we expect to complement the existing approaches via structural methods of e.g. Antsaklis et al [IA03, IA06], and also the unfolding construction of Giua and Xie [GX05]. A crucial question is the knowledge that any control player can be assumed to have, as a basis for choosing control actions. We believe the protectedness measure is a valid candidate for coding this information, so that a controller may take action when the system is too close to doom (wrt some thresholds to be calibrated) but there still remain decisions that can be taken to avoid it. Evaluating this option, along with other approaches, must, however, be left to future work.

Acknowledgments: We gratefully acknowledge the fruitful exchanges with Cédric Gaucherel and Franck Pommereau. This work was supported by the *DIGICOSME* grant ESCAPE, *DIGICOSME RD 242-ESCAPE-15203*, and by the French Agence Nationale pour la Recherche (ANR) in the scope of the project “BNeDiction” (grant number ANR-20-CE45-0001).

REFERENCES

- [AB06] Samy Abbes and Albert Benveniste. True-concurrency probabilistic models: Branching cells and distributed probabilities for event structures. *Inf. Comput.*, 204(2):231–274, 2006. doi:10.1016/j.ic.2005.10.001.

- [AHP⁺22] Giann Karlo Aguirre-Samboní, Stefan Haar, Loïc Paulevé, Stefan Schwoon, and Nick Würdemann. Avoid one's doom: Finding cliff-edge configurations in petri nets. In Pierre Ganty and Dario Della Monica, editors, *Proceedings of the 13th International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2022, Madrid, Spain, September 21-23, 2022*, volume 370 of *EPTCS*, pages 178–193, 2022. doi:10.4204/EPTCS.370.12.
- [AS24] Giann Karlo Aguirre-Samboní. ECOFOLDER, 2024. URL: <https://github.com/giannkas/ecofolder>.
- [BCH13] Sandie Balaguer, Thomas Chatain, and Stefan Haar. Building occurrence nets from reveals relations. *Fundamenta Informaticae*, 123(3):245–272, May 2013. URL: <http://www.lsv.fr/Publis/PAPERS/PDF/BCH-fi12.pdf>, doi:10.3233/FI-2013-809.
- [CHJ⁺14] Thomas Chatain, Stefan Haar, Loïc Jezequel, Loïc Paulevé, and Stefan Schwoon. Characterization of reachable attractors using petri net unfoldings. In *Proc. CMSB 2014*, LNCS 8859, pages 129–142. Springer, 2014. doi:10.1007/978-3-319-12982-2_10.
- [CHK⁺20] Thomas Chatain, Stefan Haar, Juraj Kolčák, Loïc Paulevé, and Aalok Thakkar. Concurrency in boolean networks. *Nat. Comput.*, 19(1):91–109, 2020. doi:10.1007/s11047-019-09748-4.
- [CMR⁺15] David P. A. Cohen, Loredana Martignetti, Sylvie Robine, Emmanuel Barillot, Andrei Yu. Zinovyev, and Laurence Calzone. Mathematical modelling of molecular pathways enabling tumour cell invasion and migration. *PLoS Comput. Biol.*, 11(11), 2015. doi:10.1371/journal.pcbi.1004571.
- [CTF⁺10] Laurence Calzone, Laurent Tournier, Simon Fourquet, Denis Thieffry, Boris Zhivotovsky, Emmanuel Barillot, and Andrei Yu. Zinovyev. Mathematical modelling of cell-fate decision in response to death receptor engagement. *PLoS Comput. Biol.*, 6(3), 2010. doi:10.1371/journal.pcbi.1000702.
- [EH08] Javier Esparza and Keijo Heljanko. *Unfoldings - A Partial-Order Approach to Model Checking*. Monographs in Theoretical Computer Science. An EATCS Series. Springer, 2008. doi:10.1007/978-3-540-77426-6.
- [ERV02] Javier Esparza, Stefan Römer, and Walter Vogler. An improvement of mcmillan's unfolding algorithm. *Formal Methods Syst. Des.*, 20(3):285–310, 2002. doi:10.1023/A:1014746130920.
- [FNCT06] Adrien Fauré, Aurélien Naldi, Claudine Chaouiya, and Denis Thieffry. Dynamical analysis of a generic boolean model for the control of the mammalian cell cycle. *Bioinformatics*, 22(14):e124–31, 2006. doi:10.1093/bioinformatics/btl1210.
- [FRGP17] Louis Fippo Fitime, Olivier F. Roux, Carito Guziolowski, and Loïc Paulevé. Identification of bifurcation transitions in biological regulatory networks using answer-set programming. *Algorithms Mol. Biol.*, 12(1):19:1–19:14, 2017. doi:10.1186/s13015-017-0110-3.
- [GKKS14] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Clingo = ASP + control: Preliminary report. *CoRR*, abs/1405.3694, 2014. doi:10.48550/arXiv.1405.3694.
- [GX05] Alessandro Giua and Xiaolan Xie. Control of safe ordinary petri nets using unfolding. *Discret. Event Dyn. Syst.*, 15(4):349–373, 2005. doi:10.1007/s10626-005-4057-z.
- [Haa10] Stefan Haar. Types of asynchronous diagnosability and the reveals-relation in occurrence nets. *IEEE Transactions on Automatic Control*, 55(10):2310–2320, October 2010. URL: <http://www.lsv.fr/Publis/PAPERS/PDF/haar-tac10.pdf>, doi:10.1109/TAC.2010.2063490.
- [HKS13] Stefan Haar, Christian Kern, and Stefan Schwoon. Computing the reveals relation in occurrence nets. *Theoretical Computer Science*, 493:66–79, July 2013. URL: <http://www.lsv.fr/Publis/PAPERS/PDF/HKS-tcs13.pdf>, doi:10.1016/j.tcs.2013.04.028.
- [HPS20] Stefan Haar, Loïc Paulevé, and Stefan Schwoon. Drawing the line: Basin boundaries in safe petri nets. In *Proc. CMSB 2020*, LNCS 12314, pages 321–336. Springer, 2020. doi:10.1007/978-3-030-60327-4_17.
- [HRS13] Stefan Haar, César Rodríguez, and Stefan Schwoon. Reveal your faults: It's only fair! In *Proc. ACSD 2013*, pages 120–129. IEEE Computer Society, 2013. doi:10.1109/ACSD.2013.15.
- [IA03] Marian V. Iordache and Panos J. Antsaklis. Decentralized control of petri nets. In *Proc. Workshop on Discrete Event Systems Control at the 24th International Conference on Application and Theory of Petri Nets (ATPN 2003)*, pages 143–158, 2003.
- [IA06] Marian V. Iordache and Panos J. Antsaklis. *Supervisory Control of Concurrent Systems: A Petri Net Structural Approach*. Birkhäuser, Boston, Basel, Berlin, 2006.

- [KHNS20] Hannes Klarner, Frederike Heintz, Sarah Nee, and Heike Siebert. Basins of attraction, commitment sets, and phenotypes of boolean networks. *IEEE ACM Trans. Comput. Biol. Bioinform.*, 17(4):1115–1124, 2020. doi:10.1109/TCBB.2018.2879097.
- [KW97] Ekkart Kindler and Rolf Walter. Mutex needs fairness. *Inf. Process. Lett.*, 62:31–39, 04 1997. doi:10.1016/S0020-0190(97)00033-1.
- [McM92] Kenneth L. McMillan. Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In *Proc. CAV '92*, LNCS 663, pages 164–177. Springer, 1992. doi:10.1007/3-540-56496-9_14.
- [McM93] Ken McMillan. Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In Gregor von Bochmann and David Karl Probst, editors, *Computer Aided Verification*, pages 164–177, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg. doi:10.1007/3-540-56496-9_14.
- [MHR⁺18] Nuno D. Mendes, Rui Henriques, Elisabeth Remy, Jorge Carneiro, Pedro T. Monteiro, and Claudine Chaouiya. Estimating attractor reachability in asynchronous logical models. *Frontiers in Physiology*, 9, 2018. doi:10.3389/fphys.2018.01161.
- [Mur89] Tadao Murata. Petri nets: Properties, analysis and applications. *Proc. IEEE*, 77(4):541–580, 1989. doi:10.1109/5.24143.
- [NPW79] Mogens Nielsen, Gordon D. Plotkin, and Glynn Winskel. Petri nets, event structures and domains. In *Proc. SCC 1979*, LNCS 70, pages 266–284. Springer, 1979. doi:10.1007/BFb0022474.
- [OLB⁺08] David A. Orlando, Charles Y. Lin, Allister Bernard, Jean Y. Wang, Joshua E. S. Socolar, Edwin S. Iversen, Alexander J. Hartemink, and Steven B. Haase. Global control of cell-cycle transcription by coupled CDK and network oscillators. *Nature*, 453(7197):944–947, 2008. doi:10.1038/nature06955.
- [OTL⁺04] Ertugrul M. Ozbudak, Mukund Thattai, Han N. Lim, Boris I. Shraiman, and Alexander van Oudenaarden. Multistability in the lactose utilization network of escherichia coli. *Nature*, 427(6976):737–740, 2004. doi:10.1038/nature02298.
- [Pau17] Loïc Paulevé. Pint: A static analyzer for transient dynamics of qualitative networks with ipython interface. In *Proc. CMSB 2017*, LNCS 10545, pages 309–316. Springer, 2017. doi:10.1007/978-3-319-67471-1_20.
- [PF14] Alexander N. Pisarchik and Ulrike Feudel. Control of multistability. *Physics Reports*, 540(4):167–218, 2014. doi:10.1016/j.physrep.2014.02.007.
- [PMO95] Erik Plahte, Thomas Mestl, and Stig W. Omholt. Feedback Loops, Stability and Multistationarity in Dynamical Systems. *J. Biol. Syst.*, 03(02):409–413, 1995. doi:10.1142/s0218339095000381.
- [PTG22] Franck Pommereau, Colin Thomas, and Cédric Gaucherel. Petri nets semantics of reaction rules (RR) - A language for ecosystems modelling. In *Proc. PETRI NETS 2022*, LNCS 13288, pages 175–194. Springer, 2022. doi:10.1007/978-3-031-06653-5_10.
- [Ric19] Adrien Richard. Positive and negative cycles in boolean networks. *Journal of Theoretical Biology*, 463:67–76, 2019. doi:10.1016/j.jtbi.2018.11.028.
- [Sch14] Stefan Schwoon. The MOLE tool, 2014. URL: <http://www.lsv.ens-cachan.fr/~schwoon/tools/mole/>.
- [Ste04] Christian Stehno. The PEP tool, 2004. URL: <https://sourceforge.net/projects/peptool/>.
- [Td90] René Thomas and Richard d’Ari. *Biological Feedback*. CRC Press, Boca Raton, Florida, USA, 1990.
- [Tho80] René Thomas. On the relation between the logical structure of systems and their ability to generate multiple steady states or sustained oscillations. *Springer Series in Synergies 9*, pages 180–193, 1980. doi:10.1007/978-3-642-81703-8_24.
- [TT95] Denis Thieffry and René Thomas. Dynamical behaviour of biological regulatory networks—ii. immunity control in bacteriophage lambda. *Bulletin of Mathematical Biology*, 57:277–297, 1995. doi:10.1007/BF02460619.
- [Vog95] Walter Vogler. Fairness and partial order semantics. *Inf. Process. Lett.*, 55(1):33–39, 1995. doi:10.1016/0020-0190(95)00049-1.