Avoid One's Doom: Finding Cliff-Edge Configurations in Petri Nets

Giann Karlo Aguirre-Samboní*[1] Stefan Haar*[1] Loïc Paulevé*[2]
Stefan Schwoon*[1] Nick Würdemann*[3]

[1]INRIA and LMF, CNRS and ENS Paris-Saclay, Université Paris-Saclay, Gif-sur-Yvette, France
{giann-karlo.aguirre-samboni,stefan.haar,stefan.schwoon}@inria.fr

[2]Univ. Bordeaux, Bordeaux INP, CNRS, LaBRI, UMR5800, Talence, France
loic.pauleve@labri.fr

[3]Department of Computing Science, University of Oldenburg, Oldenburg, Germany
wuerdemann@informatik.uni-oldenburg.de

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 [3] has provided techniques for identifying *attractors*, i.e. terminal strongly connected components of the reachability space, whose attraction basins we wish to determine. Here, we provide a solution for the case of safe Petri nets. Our algorithm uses net unfoldings and provides a map of all of the system's configurations (concurrent executions) that act as *cliff-edges*, i.e. any maximal extension for those configurations lies in some basin that is considered fatal. The computation turns out to require only a relatively small prefix of the unfolding, just twice the depth of Esparza's complete prefix.

1 Introduction

Unfoldings of Petri nets [6], which are essentially event structures in the sense of Winskel et al. [18] 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 [15, 7, 6]. However, questions of long-term behaviour and stabilization have received relatively little attention. With the growing interest in formal methods for biology, the key feature of *multistability* of systems [30, 24, 20, 23] comes into focus. It has been studied in other qualitative models such as Boolean and multivalued networks [29, 28, 26]. 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 to 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 [25].

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

*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).

D. Della Monica and P. Ganty (Eds.): 13th International Symposium on Games, Automata, Logics and Formal Verification (GandALF 22)
EPTCS 370, 2022, pp. 178–193, doi:10.4204/EPTCS.370.12

transition system. In the long run, the system will enter one of its attractors and remain inside; multistability arises when there is more than one such attractor. The *basin* of attractor **A** consists of the states from which the system inevitably reaches **A**.

The basin includes the attractor itself, and possibly one or several transient states [14].

We aim at finding the tipping points in 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 [5, 16]. In [8], 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.

Using a bounded unfolding prefix, all reachable attractors [3] can be extracted. Also, we have exhibited ([11]) the particular shape of basins that are visible in a concurrent model.

In the present paper, we build on this previous analyses; 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 [17] are close to Boolean and multivalued networks [4], 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 [4] and the appendix of [3] 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 subset $M \subseteq P$ of the places is called a *marking*. A *Petri net* is a tuple $\mathcal{N} = \langle P, T, F, M_0 \rangle$, with $M_0 \subseteq P$ 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 ${}^\bullet t \subseteq M$. An enabled transition t can *fire*, leading to the new marking $M' = (M \setminus {}^\bullet t) \cup t^\bullet$;[1] in that case we write

[1] This definition does not correspond to the standard semantics of Petri nets, but is equivalent for safe Petri nets, and we

$M \xrightarrow{t} M'$. A *firing sequence* from a marking M'_0 is a (finite or infinite) sequence $w = t_1 t_2 t_3 \ldots$ over T such that there exist markings $M'_1, M'_2, \ldots$ with $M'_0 \xrightarrow{t_1} M'_1 \xrightarrow{t_2} M'_2 \xrightarrow{t_3} \ldots$. 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 $M'_0 \to M'_n$. We denote the set of markings reachable from some marking M in a net N by $\mathbf{R}_N(M)$. A Petri net $\langle N, M_0 \rangle$ is considered *safe* if every marking in $M \in \mathbf{R}_N(M_0)$ and every transition t enabled in M satisfy $(M \cap t^\bullet) \subseteq {}^\bullet t$. In this paper, we assume that all our Petri nets 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 $\mathscr{M}$ a set of markings. The* marking graph *induced by $\mathscr{M}$ is a directed graph $\langle \mathscr{M}, \mathscr{E} \rangle$ such that $\mathscr{E} \subseteq \mathscr{M} \times \mathscr{M}$ 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 $\mathbf{R}_N(M_0)$.*

The reachability graph is always finite for safe Petri nets.

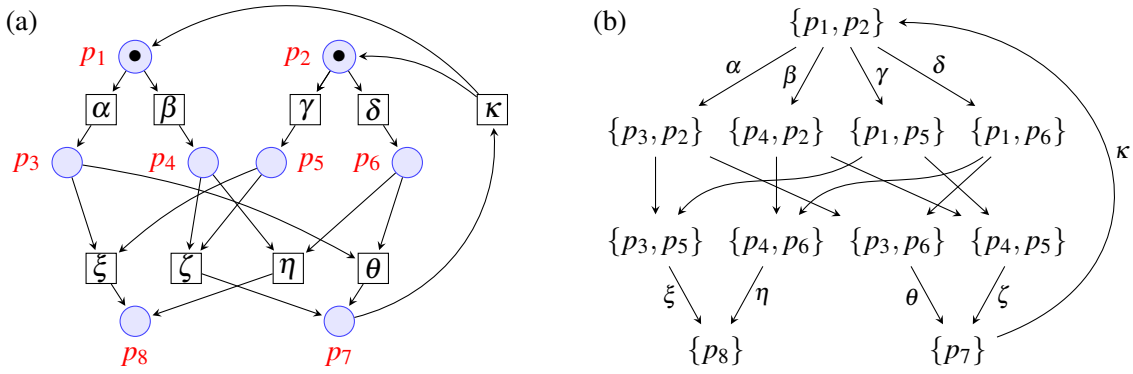

Figure 1: Petri net example from [11] in (a), and its reachability graph in (b).

Figure 1b shows the reachability graph for our running example 1a.

Unfoldings. Roughly speaking, the unfolding of a Petri net $\mathscr{N}$ is an acyclic Petri net (with particular structural properties), $\mathscr{U}$, that reproduces exactly the same behaviours as $\mathscr{N}$.

We now give some technical definitions to introduce unfoldings formally. A more extensive treatment can be found, e.g., in [7, 6].

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 **co** y, if they are neither causally related nor in conflict. A set of concurrent places is called a* co-set*.*

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

prefer it for the sake of simplicity.

1. *The causality relation $<$ is acyclic;*
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.*

Following the convention in the unfolding literature, we refer to the places of an occurrence net as *conditions* and to its transitions as *events*. 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 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 $\mathscr{O} = \langle B, E, G, \mathbf{c}_0 \rangle$ be an occurrence net. A set $C \subseteq E$ is called a* configuration *of $\mathscr{O}$ 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 $e \in E$, $[e] \stackrel{\triangle}{=} \{e' \in E : e' \leqslant e\}$ and $\langle e \rangle \stackrel{\triangle}{=} \{e' \in E : e' < e\}$ 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 $\mathscr{O}$ as $\mathscr{C}(\mathscr{O})$, and its subset containing all **finite** configurations as $\mathscr{C}^{\mathbf{f}}(\mathscr{O})$, where we drop the reference to $\mathscr{O}$ 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 $\mathscr{C}(\mathscr{O})$ w.r.t. set inclusion; denote the set of $\mathscr{O}$'s runs as $\Omega = \Omega(\mathscr{O})$, and its elements generically by ω. If $C \in \mathscr{C}^{\mathbf{f}}$, let the* crest *of C be the set $\mathbf{crest}(C) \stackrel{\triangle}{=} \max_<(C)$ of its maximal events. We say that configuration C* enables *event e, written $C \stackrel{e}{\rightsquigarrow}$, 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 $(C_1 \cup C_2) \notin \mathscr{C}$ or, equivalently, iff there exist $e_1 \in C_1$ and $e_2 \in C_2$ such that $e_1 \# e_2$.*[2]

Intuitively, a configuration is a set of events that can fire during a firing sequence of $\mathscr{N}$, and its cut is the set of conditions marked after 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 $\emptyset$. The crest of a prime configuration $[e]$ is $\{e\}$. In Figure 2, the

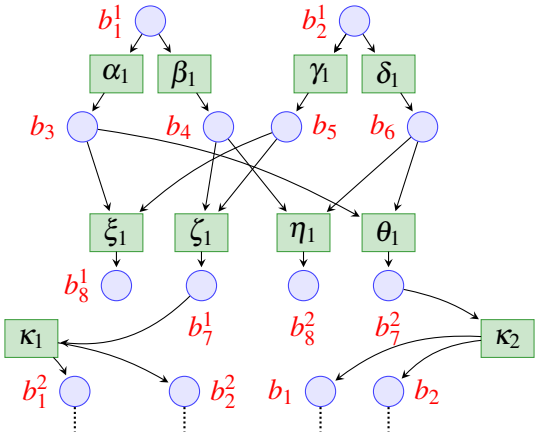

Figure 2: A prefix of the unfolding for the Petri net of Figure 1a.

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

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

Definition of Unfoldings. Let $\mathcal{N} = \langle P, T, F, M_0 \rangle$ be a safe Petri net. The unfolding $\mathcal{U} = \langle B, E, G, \mathbf{c}_0 \rangle$ of $\mathcal{N}$ is an occurrence net (equipped with a mapping π) such that the firing sequences and reachable markings of $\mathcal{U}$ are exactly the firing sequences and reachable markings of $\mathcal{N}$ (modulo π), see below. $\mathcal{U}$ may be infinite; it can be inductively constructed as follows:

1. The condition set B is a subset of $(E \cup \{\bot\}) \times P$. For a condition $b = \langle e, p \rangle$, we will have $e = \bot$ 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 exactly one condition $\langle \bot, p \rangle$ for each initially marked place $p \in M_0$ of $\mathcal{N}$.
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 $\langle B', e \rangle$ represents a possibility of firing transition e in a particular context.

Configurations and Markings. The following fact from the literature will be used below:

Lemma 1 (see e.g. [7]) *Fix a safe Petri net $\mathcal{N} = \langle P, T, F, M_0 \rangle$ 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 **co** b' (e **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 as $Mark(C) \stackrel{\triangle}{=} \{ \pi(b) \mid b \in \mathbf{cut}(C) \}$.*

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. The following definition formalizes this.

Definition 5 *Let $\mathcal{O} = \langle B, E, G, \mathbf{c}_0 \rangle$ be an occurrence net. For any finite configuration $C \in \mathcal{C}^{\mathbf{f}}(\mathcal{O})$, denote by $\mathcal{O}_C \stackrel{\triangle}{=} \mathcal{U}(\langle N, Mark(C) \rangle)$ the* shift *of $\mathcal{O}$ by C. C is the* concatenation *of C_1 and C_2, written $C = C_1 \oplus C_2$, iff one has*

1. *$C_1 \in \mathcal{C}^{\mathbf{f}}(\mathcal{O})$ and $C_1 \subseteq C$,*
2. *$C_2 \in \mathcal{C}^{\mathbf{f}}(\mathcal{O}_{C_1})$ and $C_2 = C \setminus C_1$.*

Clearly, the empty configuration $\emptyset$ satisfies $C \oplus \emptyset = \emptyset \oplus C = C$. If $C = C_1 \oplus C_2$, write $C_1 = C \ominus C_2$ and $C_2 = C \oslash C_1$. Moreover, write

$$C = \bigoplus_{i=1}^{n} C_i \quad \text{iff} \quad C = C_1 \oplus \ldots \oplus C_n.$$

In figure 2, setting $C_1 \stackrel{\triangle}{=} \{\beta_1, \gamma_1\}$, $C_2 \stackrel{\triangle}{=} \{\zeta_1, \kappa_1\}$ and $C_3 \stackrel{\triangle}{=} \{\beta_1, \gamma_1, \zeta_1, \kappa_1\}$, one has $C_3 = C_1 \oplus C_2$ and consequently $C_1 = C_3 \ominus C_2$ and $C_2 = C_3 \oslash C_1$.

Complete Prefix. In general, $\mathcal{U}$ is an infinite net, but if $\mathcal{N}$ is safe, 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.

Definition 6 (complete prefix, see [15, 7, 6]) *Let $\mathcal{N} = \langle N, M_0 \rangle$ be a safe Petri net and $\mathcal{U} = \langle B, E, G, \mathbf{c}_0 \rangle$ 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) $Mark(C) = M$, 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 $\Pi_0 = \Pi_0(\mathscr{N})$ to denote an arbitrary complete prefix of the unfolding of $\mathscr{N}$. It is known ([15, 7]) that the construction of such a complete prefix is indeed possible, and efficient tools such as MOLE ([27]) exist for this purpose. While the precise details of this construction are out of scope for this paper; some ingredients of it will play a role below, so we sketch them here.

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 $Mark([e])$ that e 'discovers' has already been discovered by a *smaller* configuration. Now, the ordering relation $\prec$ to compare two configurations must be an *adequate* order, i.e. $C_1 \subseteq C_2$ must imply $C_1 \prec C_2$, to ensure the completeness of the prefix obtained. As shown in [7], 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* order, the number of non-cutoff events of the prefix Π_0 thus obtained never exceeds the size of the reachability graph.

We will assume throughout this paper that complete prefixes are computed according to some adequate total order, as is done in particular in the MOLE tool [27]. Below, we will propose a new such order relation that underlies a novel concept of distance between markings.

The nested family $(\Pi_n)_{n\geq 0}$ of finite prefixes. Denote the complete prefix for $\mathscr{N}$ obtained according to definition 6 as Π_0; we extend Π_0 to increasing prefixes $\Pi_1, \Pi_1, \ldots$ as follows. Starting at $n = 0$,

- let $\mathscr{C}^n \stackrel{\triangle}{=} \max(\mathscr{C}(\Pi_n))$,
- set $\mathscr{M}_n \stackrel{\triangle}{=} \{M \in 2^P : \exists\, C \in \mathscr{C}^n :\ M = Mark(C)\}$,
- for all $M \in \mathscr{M}_n$, compute a complete prefix $\preceq^M$ of $\langle N, M\rangle$;
- obtain Π_{n+1} by appending, to every $C \in \mathscr{C}^n$, a copy of $\preceq^{Mark(C)}$ to every $C \in \mathscr{C}^n$.

3 Doomed configurations, and how to avoid them

3.1 Bad, Free and Doomed Configurations and Markings.

In this section, we present an algorithm that identifies precisely those configurations of a Petri net unfolding from which one can no longer avoid reaching a certain long-term behaviour, its theoretical foundations, and some experimental results. The formal setting here contains and extends the one established in [10], specialized to the 1-safe case. We assume that we are given a set of *bad markings* $\mathscr{Z} \subseteq 2^P$. Since we are interested in long-term behaviours, we assume that $\mathscr{Z}$ is reachability-closed, i.e. $M \in \mathscr{Z}$ and $M \to M'$ imply $M' \in \mathscr{Z}$.

Define $\mathscr{B} \stackrel{\triangle}{=} \{C \in \mathscr{C}^{\mathbf{f}} : Mark(C) \in \mathscr{Z}\}$ as the set of *bad configurations*, and let $\mathscr{B}_0$ be the set of configurations in $\mathscr{B}$ that are contained in Π_0. $\mathscr{B} \subseteq \mathscr{C}$ is *absorbing* or *upward closed*, that is, for all $C_1 \in \mathscr{B}$ and $C_2 \in \mathscr{C}^{\mathbf{f}}$ such that $C_1 \subseteq C_2$, one must have $C_2 \in \mathscr{B}$.

For any $C \in \mathscr{C}$, let $\Omega_C \stackrel{\triangle}{=} \{\omega \in \Omega :\ C \subseteq \omega\}$ denote the maximal runs into which C can evolve. We are interested in those finite configurations all of whose maximal extensions are 'bad', where we consider infinite configurations as bad if they contain a bad finite configuration. 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 7 *Configuration $C \in \mathscr{C}^{\mathbf{f}}$ is* doomed *iff*

$$\forall\, \omega \in \Omega_C :\ \exists\, C' \in \mathscr{C}^{\mathbf{f}} : \left\{ \begin{array}{l} C \subseteq C' \subseteq \omega \\ \wedge\ Mark(C') \in \mathscr{Z} \end{array} \right. \tag{1}$$

The set of doomed configurations is denoted $\mathscr{D}$; denote the set of minimal elements in $\mathscr{D}$ by $\check{\mathscr{D}}$. If C is not doomed, it has at least one maximal extension that never reaches bad markings. We call configurations that are not doomed free*, and denote the set of free configurations by $\mathscr{F}$.*

All reachable markings are represented by at least one configuration. Moreover, since the future evolution of $\mathscr{N}$ depends only on the current marking, $Mark(C_1) = Mark(C_2)$ 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 $Mark(C)$ free or doomed whenever C is.

Running Example. In the context of Figures 1a and 2, we consider $\mathscr{L}$ the singleton set containing the marking $M_8 = \{P_8\}$. Clearly, $C_1 \stackrel{\triangle}{=} \{\alpha_1, \gamma_1, \xi_1\}$ and $C_2 = \{\beta_1, \delta_1, \eta_1\}$ satisfy $Mark(C_1) = Mark(C_2) = M_8$ and therefore $C_1, C_2 \in \mathscr{B}$. But note that $C_1' \stackrel{\triangle}{=} \{\alpha_1, \gamma_1\}$ and $C_2' = \{\beta_1, \delta_1\}$ produce markings outside $\mathscr{L}$, but they are doomed since any extension of these configurations leads into $\mathscr{L}$. Therefore, $C_1', C_2' \in \mathscr{B}$. 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. In a first step towards that, Theorem 1 below uses a similar proof idea as Lemma 8 in [12] in the context of fault diagnosis. Let us first recall the notion of *spoilers*, introduced in [12]:

Definition 8 *A* spoiler *of transition t (or event e) is any $t' \in T$ ($e' \in E$) such that ${}^\bullet t' \cap {}^\bullet t \neq \emptyset$ (${}^\bullet e \cap {}^\bullet e' \neq \emptyset$). We write* $\mathbf{spoil}(t)$ *(*$\mathbf{spoil}(e)$*) for the set of t's (e's) spoilers.*

Note that $t \in \mathbf{spoil}(t)$ for all $t \in T$. The spoilers of t are characterized by the fact that their firing cancels any enabling of t; that is, by being either in conflict with t, or identical with t.

Theorem 1 *A configuration $C \in \mathscr{C}^{\mathbf{f}}$ is* ***free*** *iff either a) there exists a finite maximal configuration C' such that $C \subseteq C' \notin \mathscr{B}$, or b) there exist configurations $C_1, C_2 \in \mathscr{C}^{\mathbf{f}}$ such that*

1. *$C \subseteq C_1 \subseteq C_2 \notin \mathscr{B}$;*
2. *$Mark(C_1) = Mark(C_2)$;*
3. *for all events $e \in E$ such that $C_1 \stackrel{e}{\rightsquigarrow}$, one has $\mathbf{spoil}(e) \cap C_2 \neq \emptyset$.*

Some comments are in order before giving the proof of Theorem 1. First of all, the requirement to check whether $C_2 \notin \mathscr{B}$ can be met by checking whether $Mark(C_2) \in \overline{\mathscr{L}}$. Second, the spoiling condition (3) ensures that the process that takes C_1 to C_2 forms a loop whose iteration yields a run.

Proof: In the following, let $M := Mark(C_1)$. We first prove the right-to-left implication. Case a) is obvious, so assume that b) holds. Let $C_2 = C_1 \oplus \hat{C}$; then by 2., we can append $\hat{C}$ to C_2, yielding a strictly increasing sequence of configurations $(C_n)_{n\in\mathbb{N}}$ such that $C_{n+1} = C_n \oplus \hat{C}$, and $Mark(C_n) = M$ for all n. By property 3, we know that $\hat{C}$ contains spoilers for all its initially enabled events , hence no transition remains enabled forever, and $\omega \stackrel{\triangle}{=} \bigcup_{n\in\mathbb{N}} C_n$ is a maximal configuration. It remains to show that ω contains no bad configuration: Suppose that there is $C' \subseteq \omega$ with $C' \in \mathscr{B}$. Since C' is finite, we have $C' \subseteq C_n$ for some n. But then, $M = Mark(C_n)$ is reachable from $Mark(C') \in \mathscr{L}$, contradicting our assumptions. Thus ω never enters a bad state, and C is free.

For the forward implication, assume that C is free. Then there exists $\omega \in \Omega_C$ such that $C' \notin \mathscr{B}$ for all finite $C' \subseteq \omega$. If this ω can be chosen finite, then a) holds and we are done; so assume henceforth that ω must be chosen infinite. Clearly, there must exist a reachable marking M that is visited an infinite number of times by a family of nested finite configurations $(C^n)_{n\in\mathbb{N}}$ such that $\bigcup_{n\in\mathbb{N}} C^n = \omega$. Let $C_1 \stackrel{\triangle}{=} C^1$

and $E' := \{e : C_1 \stackrel{e}{\rightsquigarrow}\}$. Let K be the smallest index such that for all $e \in E'$, one has $\mathbf{spoil}(e) \cap C^K \neq \emptyset$; such a K must exist since ω is maximal. Then C_1 and $C_2 \stackrel{\triangle}{=} C^K$ have the required properties. □

Notice that the proof could be restructured by observing that case a) of Theorem 1 is indeed a special instance of case b). In fact, taking $C_1 \stackrel{\triangle}{=} C_2 \stackrel{\triangle}{=} C'$ with C' according to case a), conditions 1 and 2 of part b) are obviously satisfied, and condition 3 holds vacuously since no event is enabled in $C_1 = C'$. We note in passeing that this observation is helpful in simplifying the implementation used for the experiments below.

The interest of Theorem 1 lies in the following fact:

Lemma 2 *For $C \in \mathscr{C}^{\mathbf{f}}$, checking whether C is free can be done using finite prefix Π_1 of $\mathscr{U}(N, Mark(C))$.*

Proof: If C is free, let C_1 and C_2 be the configurations witnessing this fact from Theorem 1, and let $M := Mark(C_1) = Mark(C_2)$. If such configurations exist, then C_1 can be chosen from the complete prefix Π_0, and C_2 can be chosen from Π_1, notably in the copy of $\Pi_{Mark(C_1)}$ appended after C_1.

Checking whether the configuration C_2 thus found is in $\mathscr{B}$ is immediate, since it suffices to check whether its marking is in $\mathscr{Z}$, using the fact that $\mathscr{Z}$ is reachability-closed. To check the spoiler condition (3) of Theorem 1, it suffices to check whether the conditions of the cut of C_1 that are not consumed by C_2 enable some event. □

3.2 Finding Minimally Doomed Configurations: Algorithm MINDOO

Shaving and Rubbing. Let us start by observing that $\mathscr{B}$, an upward closed set by construction, also has some downward closure properties, meaning one can restrict control to act on 'small' configurations.

Definition 9 *An event e is* unchallenged *iff there is no e' such that $e \#_\delta e'$, i.e. $({}^\bullet e)^\bullet = \{e\}$.*

Lemma 3 *Let $C \in \mathscr{C}^{\mathbf{f}}$ and $e \in \mathbf{crest}(C)$ unchallenged; set $C' \stackrel{\triangle}{=} C \backslash \{e\}$. Then $C' \in \mathscr{C}^{\mathbf{f}}$, and $\Omega_C = \Omega_{C'}$.*

Proof: $C' \in \mathscr{C}^{\mathbf{f}}$ holds by construction. Also, $\Omega_C \subseteq \Omega_{C'}$ follows from $C' \subseteq C$; it remains to show the reverse inclusion. Assume there exists $\omega \in \Omega_{C'} \backslash \Omega_C$; then $C \backslash \omega = \{e\}$, 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 10 *A configuration $C \in \mathscr{C}^{\mathbf{f}}$ such that $\mathbf{crest}(C)$ contains no unchallenged event is called* shaved.

Clearly, every $C \in \mathscr{C}^{\mathbf{f}}$ 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 3, 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 3, any $C \in \mathscr{C}^{\mathbf{f}}$ is in $\mathscr{B}$ 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 $C \backslash \{e\}$ is free (e.g. because some event in conflict with e may allow to move away from doom) or still doomed.

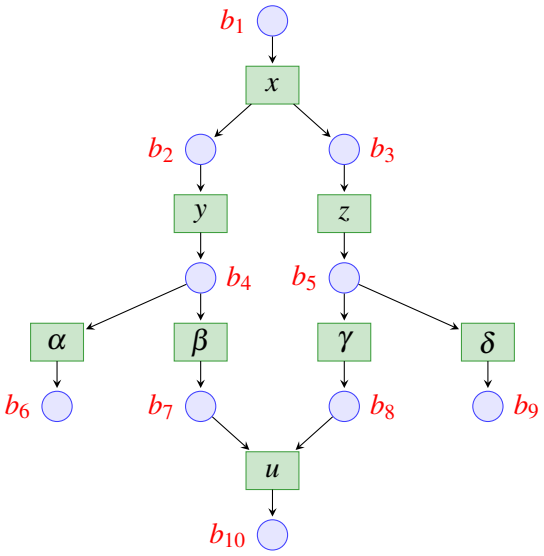

Figure 3: An occurrence net. With $C \stackrel{\triangle}{=} \{x,y,z,\beta,\gamma\}$ and $C' \stackrel{\triangle}{=} C \cup \{u\}$, suppose $\mathscr{Z} = \{Mark(C')\} = \pi(\{b_{10}\})$. Then $\mathbf{shave}(C') = C$, and C is doomed. Moreover, $C \in \check{\mathscr{D}}$ since both $C_4 \stackrel{\triangle}{=} C\backslash\{\beta\}$ and $C_5 \stackrel{\triangle}{=} C\backslash\{\gamma\}$ are free.

If the latter is the case, then C was not minimally doomed, and analysis continues with $C\backslash\{e\}$ (we say that we 'rub away' e). If $C\backslash\{e\}$ 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.

Algorithm 1 uses a 'worklist' set wl of doomed, shaved configurations to be explored; 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 wl and added to $\mathfrak{D}$.

Every branch 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. $C\backslash\{e\}$ 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 $\emptyset \in$ 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 $\mathscr{B}$, itself strictly contained in the complete finite prefix used to find all bad markings. According to [7], this prefix can be chosen of size equal or smaller (typically: considerably smaller) than the reachability graph of $\mathscr{N}$.

Theorem 2 *For any safe Petri net $\mathscr{N} = \langle N, M_0\rangle$ and bad states set $\mathscr{Z} \subseteq \mathbf{R}_N(M_0)$, Algorithm* MINDOO *terminates, with output set $\mathfrak{D}$ containing exactly all minimal doomed configurations, i.e. $\mathfrak{D} = \check{\mathscr{D}}$.*

Proof: Termination follows from the finiteness of $\min_{\subseteq}(\mathscr{B}_0)$, since in each round of MINDOO there is one configuration C that is either replaced by a set of strict prefixes or removed from wl. Therefore, after a finite number of steps wl is empty. According to Lemma 2, the status (doomed or free) of a given finite configuration can effectively be checked on a fixed finite prefix of $\mathscr{U}$. Assume that after termination of MINDOO, one has $C \in \mathfrak{D}$; we need to show $C \in \check{\mathscr{D}}$. Clearly, when C was added to $\mathfrak{D}$, it had been

Algorithm 1: Algorithm MINDOO

```
Data: Safe Petri Net 𝒩 = ⟨P,T,F,M0⟩ and 𝒵 ⊆ 2^P
Result: The set 𝔇 of 𝒩's ⊆-minimal doomed configurations
𝔇 ← ∅; wl ← ∅;
foreach C ∈ min⊆(ℬ0) do
    C' ← shave(C);
    wl ← wl ∪ {C'};
end
while wl ≠ ∅ do
    Pick C ∈ wl; add ← true;
    if (C\crest(C)) is doomed then
        add ← false;
        C' ← shave(C\crest(C));
        wl ← wl ∪ {C'};
    else
        foreach e ∈ crest(C) do
            if (C\{e}) is doomed then
                add ← false;
                C' ← shave(C\{e});
                wl ← wl ∪ {C'};
            end
        end
    end
    wl ← wl\{C};
    if add then
        𝔇 ← 𝔇 ∪ {C};
    end
end
return 𝔇
```

detected as doomed; it remains to show that C is also minimal with this property. Assume that there is $C' \subsetneq C$ that as doomed as well. But in that case there exists $e \in \mathbf{crest}(C)$ such that $C' \subseteq (C\backslash\{e\}) \subsetneq C$, which implies that this $(C\backslash\{e\})$ 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 $C \in \check{\mathscr{D}}$. Then $(C\backslash\{e\})$ is free for all $e \in \mathbf{crest}(C)$; 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. □

3.3 Implementation and Experiments.

A prototype implementation of MINDOO is available at [21]. It takes as input a safe Petri net in the PEP format and relies on MOLE [27] for computing the initial finite prefix Π_0 and its extensions. Algorithm 1 is implemented in Python, where the identification of maximal configurations, bad configurations, as well as the verification of doomed status of a configuration is performed in Answer-Set Programming

Table 1: Statistics of Algorithm 1 on Petri net models of biological systems. The size of Π_0 and Π_1 is the number of their events; "# min doomed cfg" is the number of minimally doomed configurations; "# doom checks" is the number of SAT checks for doom status of a configuration. "time" is the total computation time on a 1.8Ghz CPU

Model	size Π_0	size Π_1	# min doomed cfg	# doom checks	time
Lambda switch	126	1,060	10	29	1s
Cell death receptor	791	19,262	57	407	37s
Budding yeast cell cycle	1,413	184,363	114	837	8m3s

(ASP) employing the CLINGO solver [9], a logic programming technology close to SAT solving.

We illustrate in Table 1 the behavior of the implementation on different instances of Petri nets modeling biological processes. In each case, we report the size (number of events) of prefixes Π_0 and Π_1 (including cut-off events), the number of minimally doomed configurations, and the number of configurations which have been tested for being doomed. The purpose of the conducted experiments was to study the tractability of our approach on literature models of biological systems for which the study of doomed configuration was relevant. As exhibited in [3], one of the first potential bottleneck is the tractability of the computation of the finite complete prefix Π_0 and the enumeration of maximal configurations, which is required for computing Π_1. Then, our experiments have focused on assessing how evolved the number of minimally doomed configurations, the number of candidate configurations screened by Algorithm 1, and the overall computation time, with different sizes of prefixes Π_1.

We selected 3 models published as Boolean networks, which can be translated as safe Petri nets using the encoding described in [3] implemented in the tool PINT [22]. The "Lambda switch" model [28] comprises 11 places and 41 transitions, and possesses two limit behaviors, one being a deadlock, marked as a bad marking. The "Cell death receptor" model [2] comprises 22 places and 33 transitions, and reproduces a bifurcation process into different cell fates, one of which has been declared as bad (apoptosis). In these two cases, the minimally doomed configurations identify configurations in which a decisive event has just taken place, committing the system to the attractor marked as bad. The "Budding yeast cell cycle" model [19] comprises 18 places and 32 transitions, and represents the oscillation of gene activity during the cell cycle. In this model, the cycle can exit and eventually reach a marking corresponding to all genes being inactive, which is our bad marking. In this later case, the minimally doomed configurations identify precisely when the system exits its oscillatory behavior.

It appears that the computation time for identifying minimally doomed configurations seems mostly affected by the size of Π_1 for the verification of the doom property of a configuration by ASP solving, implementing the conditions of Theorem 1. In each case, the number of minimally doomed configurations is a fraction of the size of the finite complete prefix Π_0. Future work may explore compact representations of the set of minimally doomed configurations, as they typically share a large amount of events, and may ease biological interpretations.

4 Protectedness

4.1 Cliff-Edges and Ridges.

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

Definition 11 *An event set $\gamma \subseteq E$ is called a* cliff-edge *iff there exists a minimally doomed configuration*

$C \in \breve{\mathscr{D}}$ *such that* $\gamma = \mathbf{crest}(C)$. *The set of cliff-edges is denoted* Γ. *The folding* $\chi \stackrel{\triangle}{=} \pi(\gamma) \subseteq T$ *of a cliff-edge* γ *is called a* ridge.

To complete the map of the evolutional landscape for $\mathscr{N}$, 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 $\mathscr{N}$ 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 4 *For every ridge* χ *of* $\mathscr{N}$ *there is a witness in* Π_0*, i.e. there exists a minimally doomed configuration* C *in* Π_1 *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 $M^C \stackrel{\triangle}{=} Mark(C_\chi)$, and let M^C_χ the unique reachable marking such that $M^C_\chi \stackrel{\chi}{\to} M^C$. Then any such M^C_χ is represented by some C^χ in Π_0. By construction, there exists a cliff-edge γ such that $C^\chi \stackrel{\gamma}{\rightsquigarrow}$ and $\pi(\gamma) = \chi$. Then $C \stackrel{\triangle}{=} C^\chi \cup \gamma$ is a minimally doomed configuration that lies within Π_1. □

4.2 Measuring the Distance from Doom

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. We chose to measure this distance not in terms of the *length* of paths, or 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 nonsequential 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 12 *Let* $C \in \mathscr{C}^{\mathbf{f}}$*, and define*

$$\mathbf{dech}(C) \stackrel{\triangle}{=} \left|\left\{e \in C : \exists\, e' \in E : e \#^C_\sigma e'\right\}\right|,$$

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

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

$\mathbf{dech}(C)$ *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 $\mathbf{dech}(C_0) = 1$.

Note that $\#^C_\sigma$ is more restrictive than direct conflict $\#_\delta$; it is also more restrictive than the *immediate conflict* in the literature (e.g. [1]). 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.

Now, for any free marking M (or, equivalently, any free configuration C such that $Mark(C) = M$), 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:

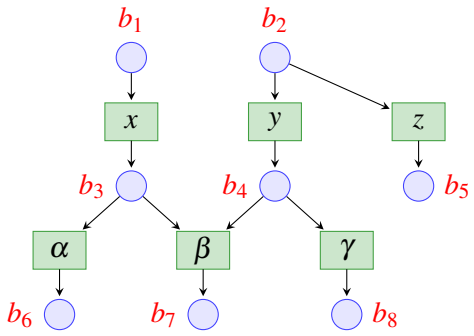

Figure 4: Illustration of direct conflict.

Definition 13 *For $C \in \mathscr{C}^{\mathbf{f}}$, let*

$$\check{\mathscr{D}}_C \triangleq \begin{cases} \{C' \in \check{\mathscr{D}} : C \subseteq C'\} & : \quad C \in \mathscr{F} \\ \{C\} & : \quad C \in \mathscr{B} \end{cases} \tag{2}$$

The protectedness *of C is then*

$$\mathbf{prot}(C) \triangleq \min_{C' \in \check{\mathscr{D}}_C} \{\mathbf{dech}(C' \oslash C)\} \tag{3}$$

In Figure 3, with the definitions introduced there, $\mathbf{prot}(C) = \mathbf{prot}(C') = 0$. Setting $C_1 \triangleq \{x\}$, $C_2 \triangleq \{x,y\}$, $C_3 \triangleq \{x,z\}$, $C_4 \triangleq C_2 \cup C_3$, $C_5 \triangleq C_4 \cup \{\beta\}$, and $C_6 \triangleq C_4 \cup \{\gamma\}$, one further has

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

Returning to Figure 4, 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 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 4. 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 $C \in \check{\mathscr{D}}$, 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.

4.3 Computing Protectedness is Feasible

Computation of $\mathbf{prot}(\bullet)$ does not require any larger data structure than those already required for computing $\check{\mathscr{D}}$ according to Lemma 2:

Lemma 5 *There is a complete prefix scheme producing a complete prefix* Π_0 *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* $\Pi_0(Mark(C))$.

Proof: If $\check{\mathscr{D}} \cap \mathscr{C}(\Pi_0) = \emptyset$, then all extensions of C are free, and we are done. Otherwise, the crucial step is to find an adequate *total* order $\prec$ on finite configurations, that ensures that Π_0 contains at least one minimally doomed configuration that minimizes $\mathbf{dech}(\bullet)$ over all minimally doomed configurations in $\mathscr{U}(Mark(C))$. The following order $\prec$ is obtained by modifying the total order $\prec_F$ introduced in [7], Def. 6.2.: For $C_1, C_2 \in \mathscr{C}^{\mathbf{f}}$, 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 [7] extends immediately, proving that $\prec$ is an adequate total order; therefore, Lemma 5.3. of [7] applies, hence any complete prefix $\Pi_0^{\prec}$ obtained via the scheme using $\prec$ is bounded in size by the reachability graph. Now, let $\mathscr{C}^*$ be the set of configurations from $\check{\mathscr{D}}(Mark(C))$ that minimize $\mathbf{dech}(\bullet)$; by construction of $\prec$, one has $\mathscr{C}^* \cap \mathscr{C}(\Pi_0^{\prec}) \neq \emptyset$. □

5 Discussion

The results presented here give a 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 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 [13], and also the unfolding construction of Giua and Xie [10]. A crucial question is the knowledge that any control player can be assumed to have, as a basis for chosing 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 decisns that can be taken to avoid it. Evaluating this option, along with other approaches, must, however, be left to future work.

References

[1] Samy Abbes & Albert Benveniste (2006): *Probabilistic models for true-concurrency: branching cells and distributed probabilities for event structures*. Information and Computation 204(2), pp. 231–274, doi:10.1016/j.ic.2005.10.001.

[2] Laurence Calzone, Laurent Tournier, Simon Fourquet, Denis Thieffry, Boris Zhivotovsky, Emmanuel Barillot & Andrei Zinovyev (2010): *Mathematical Modelling of Cell-Fate Decision in Response to Death Receptor Engagement*. *PLOS Computational Biology* 6(3), p. e1000702, doi:10.1371/journal.pcbi.1000702.

[3] Thomas Chatain, Stefan Haar, Loïg Jezequel, Loïc Paulevé & Stefan Schwoon (2014): *Characterization of Reachable Attractors Using Petri Net Unfoldings*. In Pedro Mendes, editor: *Proceedings of the 12th Conference on Computational Methods in System Biology (CMSB'14)*, *Lecture Notes in Bioinformatics* 8859, Springer-Verlag, Manchester, UK, pp. 129–142, doi:10.1007/978-3-319-12982-2_10.

[4] Thomas Chatain, Stefan Haar, Juraj Kolcák, Loïc Paulevé & Aalok Thakkar (2020): *Concurrency in Boolean networks*. *Nat. Comput.* 19(1), pp. 91–109, doi:10.1007/s11047-019-09748-4.

[5] David P. A. Cohen, Loredana Martignetti, Sylvie Robine, Emmanuel Barillot, Andrei Zinovyev & Laurence Calzone (2015): *Mathematical Modelling of Molecular Pathways Enabling Tumour Cell Invasion and Migration*. *PLoS Comput Biol* 11(11), p. e1004571, doi:10.1371/journal.pcbi.1004571.

[6] J. Esparza & K. Heljanko (2008): *Unfoldings – A Partial-Order Approach to Model Checking*. Springer. ISBN: 978-3-540-77426-6.

[7] J. Esparza, S. Römer & W. Vogler (2002): *An Improvement of McMillan's Unfolding Algorithm*. *FMSD* 20, pp. 285–310, doi:10.1023/A:1014746130920.

[8] Louis Fippo Fitime, Olivier Roux, Carito Guziolowski & Loïc Paulevé (2017): *Identification of bifurcation transitions in biological regulatory networks using Answer-Set Programming*. *Algorithms for Molecular Biology* 12(1), p. 19, doi:10.1186/s13015-017-0110-3.

[9] Martin Gebser, Roland Kaminski, Benjamin Kaufmann & Torsten Schaub (2014): *Clingo = ASP + Control: Preliminary Report*. *CoRR* abs/1405.3694, doi:10.48550/arXiv.1405.3694.

[10] A. Giua & X. Xie (2005): *Control of safe ordinary Petri nets using unfolding*. *Discrete Event Dynamic Systems* 15(4), pp. 349–373, doi:10.1007/s10626-005-4057-z.

[11] Stefan Haar, Loïc Paulevé & Stefan Schwoon (2020): *Drawing the Line: Basin Boundaries in Safe Petri Nets*. In Alessandro Abate, Tatjana Petrov & Verena Wolf, editors: *Proc.18th Conf. on Computational Methods in System Biology (CMSB'20)*, *Lecture Notes in Bioinformatics* 12314, Springer, pp. 321–336, doi:10.1007/978-3-030-60327-4_17.

[12] Stefan Haar, César Rodríguez & Stefan Schwoon (2013): *Reveal Your Faults: It's Only Fair!* In Marta Pietkiewicz-Koutny & Mihai Teodor Lazarescu, editors: *Proc. 13th Int. Conf. on Application of Concurrency to System Design (ACSD'13)*, IEEE Computer Society Press, Barcelona, Spain, pp. 120–129, doi:10.1109/ACSD.2013.15.

[13] Marian V. Iordache & Panos J. Antsaklis (2006): *Supervisory Control of Concurrent Systems: A Petri Net Structural Approach*. Birkhäuser, Boston, Basel, Berlin.

[14] H. Klarner, H. Siebert, S. Nee & F. Heinitz (2018): *Basins of Attraction, Commitment Sets and Phenotypes of Boolean Networks*. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, doi:10.1109/TCBB.2018.2879097.

[15] K. L. McMillan (1992): *Using Unfoldings to Avoid the State Explosion Problem in the Verification of Asynchronous Circuits*. In: *CAV*, pp. 164–177, doi:10.1007/3-540-56496-9_14.

[16] Nuno D. Mendes, Rui Henriques, Elisabeth Remy, Jorge Carneiro, Pedro T. Monteiro & Claudine Chaouiya (2018): *Estimating Attractor Reachability in Asynchronous Logical Models*. *Frontiers in Physiology* 9, doi:10.3389/fphys.2018.01161.

[17] T. Murata (1989): *Petri nets: Properties, analysis and applications*. *Proc. of the IEEE* 77(4), pp. 541–580, doi:10.1109/5.24143.

[18] M. Nielsen, G. D. Plotkin & G. Winskel (1979): *Petri Nets, Event Structures and Domains*. In: *SCC*, pp. 266–284, doi:10.1016/0304-3975(81)90112-2.

[19] David A. Orlando, Charles Y. Lin, Allister Bernard, Jean Y. Wang, Joshua E. S. Socolar, Edwin S. Iversen, Alexander J. Hartemink & Steven B. Haase (2008): *Global control of cell-cycle transcription by coupled CDK and network oscillators*. Nature 453(7197), pp. 944–947, doi:10.1038/nature06955.

[20] Ertugrul M. Ozbudak, Mukund Thattai, Han N. Lim, Boris I. Shraiman & Alexander van Oudenaarden (2004): *Multistability in the lactose utilization network of Escherichia coli*. Nature 427(6976), pp. 737–740, doi:10.1038/nature02298.

[21] Loïc Paulevé: *Implementation of the search for minimal doomed configurations.* Available at `https://gitub.u-bordeaux.fr/lpauleve/doomed-configurations`.

[22] Loïc Paulevé (2017): *Pint: a static analyzer for transient dynamics of qualitative networks with IPython interface*. In: *CMSB 2017 - 15th conference on Computational Methods for Systems Biology*, *Lecture Notes in Computer Science* 10545, Springer International Publishing, pp. 309–316, doi:10.1007/978-3-319-67471-1_20.

[23] Alexander N. Pisarchik & Ulrike Feudel (2014): *Control of multistability*. *Physics Reports* 540(4), pp. 167–218, doi:10.1016/j.physrep.2014.02.007.

[24] Erik Plahte, Thomas Mestl & Stig W. Omholt (1995): *Feedback Loops, Stability and Multistationarity in Dynamical Systems*. *J. Biol. Syst.* 03(02), pp. 409–413, doi:10.1142/s0218339095000381.

[25] Franck Pommereau, Colin Thomas & Cédric Gaucherel (2022): *Petri Nets Semantics of Reaction Rules (RR), a Language for Ecosystems Modelling*. In: *Proc. 43rd Int. Conf. on Application and Theory of Petri Nets and Concurrency*, Bergen, Norway, pp. 1–20, doi:10.1007/978-3-031-06653-5_10.

[26] Adrien Richard (2019): *Positive and negative cycles in Boolean networks*. *Journal of Theoretical Biology* 463, pp. 67–76, doi:10.1016/j.jtbi.2018.11.028.

[27] S. Schwoon (2014): *The* MOLE *Tool*. URL: `http://www.lsv.ens-cachan.fr/~schwoon/tools/mole/`.

[28] Denis Thieffry & René Thomas (1995): *Dynamical behaviour of biological regulatory networks—II. Immunity control in bacteriophage lambda*. *Bulletin of Mathematical Biology* 57, pp. 277–297, doi:10.1007/BF02460619.

[29] R. Thomas (1980): *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*, pp. 180–193, doi:10.1007/978-3-642-81703-8_24.

[30] René Thomas & Richard d'Ari (1990): *Biological Feedback*. CRC Press, Boca Raton, Florida, USA.