Resilience in Discrete Event Systems
Eric Fabre

To cite this version:
Eric Fabre. Resilience in Discrete Event Systems. WODES 2022 - 16th IFAC Workshop on Discrete Event Systems, Sep 2022, Prague, Czech Republic. pp.1-6. hal-03777858

HAL Id: hal-03777858
https://hal.science/hal-03777858v1
Submitted on 15 Sep 2022

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.
Resilience in Discrete Event Systems

E. Fabre *

* Univ Rennes, INRIA Rennes Bretagne Atlantique, France

Abstract. This paper explores the notion of resilience of a fault in a DES, as the ability to spontaneously return to a normal behavior, without leaking information about this fault occurrence to an external observer. Resilience expresses some form of insensitivity or robustness, related but different from classical notions like diagnosability or opacity. Different definitions of resilience are examined, expressed in terms of recovery points or of language relations. It is shown that verifying resilience is PSPACE-complete. However, the complexity reduces to PTIME for the recently introduced history deterministic automata, a sub-class of NFA. We examine in detail the instructive pathway to this complexity reduction, which could be instrumental for the complexity reduction of other DES problems.

1. MOTIVATION

Modern software production proceeds by adapting and assembling ready-made components, inevitably leading to unforeseen side effects. But not all deviations from an expected behavior are detectable to the global functioning. The chaos engineering approach specifically aims at designing components that would recover from failures of critical resources. Consider a software component with a hidden internal failsafe: in case of some distant server address holding a database is unspecified or the server is not responding (the “fault”), an address book can be queried (recovery actions). Clearly, the fault will change temporarily the behavior of the system, and after some transient it will be back to normal. This may even go unnoticed by a user.

This suggests that a long standing perspective which focused on the detectability of faults could be complemented by a resilience analysis, that would separate harmful faults from more benign ones. We are interested in characterizing this resilience property, which could be defined as follows: after a bounded number of steps following the fault, an external observer can no longer distinguish the consequences of the faulty run from those of a non-faulty run that would have produced the same observations. So one could consider that the system has returned to a normal behavior, or recovered from the fault. Which does not mean that all memory of the fault is erased, nor that it cannot be detected in the future.

Resilience has connections to but is different from well established notions like diagnosability or opacity (Lafortune 2018). Diagnosability of a fault event examines whether it can be detected in bounded time after its occurrence. At a recovery time following this fault, the faulty run is by definition ambiguous as it cannot be distinguished from at least one safe run, which furthermore has the same visible future. However, according to the definition one adopts for resilience, it is possible that all the equivalent safe runs will either be discarded by future observations, or will themselves suffer a fault event, entailing a fault detection. Resilience of a faulty run is therefore different from its diagnosability, but we will show that if all faulty runs are resilient, that is if the system itself is resilient, then faults cannot be detected. Similarly, the dual notion of opacity checks if a faulty (or “secret”) run of the system will always be confused with a normal (“non-secret”) one. Opacity is more generally defined for non-permanent secrets, so applying it here to the notion of fault assumes a restriction to permanent secrets. With the same argument as above, resilience of a faulty run is different from its opacity in the sense that a diagnosable fault (thus non opaque) can or not be resilient.

The paper is organized as follows. Section 2 proposes different definitions of resilience, through the generic notion of recovery point. It proves that deciding resilience is PSPACE-complete, then examines connections with diagnosability and opacity. Section 3 explores an original path to bring back the notion of resilience to PTIME, by constraining the plant to be weakly non-deterministic. The newly introduced class of history deterministic systems (Henzinger 2006), and its extension to the hierarchy of k-width systems (Kuperberg 2019) covering the class of NFA, offers remarkable tools to do so, that we examine in detail as they could be adapted to other DES problems.

2. RESILIENCE THROUGH RECOVERY POINTS

Consider a non-deterministic finite state automaton (NFA) $A = (\Sigma, S, T, s_0, S_M)$ over alphabet $\Sigma$, with $S$ as state set, $s_0 \in S$ as initial state, $S_M \subseteq S$ as marked states, and $T \subseteq S \times \Sigma \times S$ as transition set. For $t = (s, \alpha, s') \in T$ we denote $t^- = t^+ = s'$ and $\sigma(t) = \alpha$. A path in $A$ is a sequence of transitions $\pi = t_1 \cdots t_n$ such that $t_i^- = t_{i+1}^-$ for $1 \leq i < n$. By extension, $\pi^- = t_1^- \cdots t_n^-$, $\sigma(\pi) = \sigma(t_1) \cdots \sigma(t_n)$, and $|\pi| = n$. We denote $\pi = \pi_1 \pi_2$ the concatenation of paths, whenever $\pi_1^+ = \pi_2^-$, and denote $\pi_1 \leq \pi$ the prefix relation. A run of $A$ is a path $\pi$ rooted at the initial state, $\pi^- = s_0$, and this run is accepted by $A$ if it terminates in a marked state, $\pi^+ \in S_M$. The set of runs in $A$ (resp. accepted runs) is denoted by $R(A)$ (resp. $R^+(A)$). The language $L(A) \subseteq \Sigma^*$ of $A$ is formed by the signatures of accepted runs $L(A) = \sigma(R^+(A))$. For $s \in S$, we also denote as $L_s(A)$ the language of $A$ rooted at $s$ instead of $s_0$. Two automata are equivalent if they have the same language. For $L \subseteq \Sigma^*$, $T = \{u : \exists \nu, uv \in L\}$ is the prefix closure of $L$, and $u^{-1}L = \{v : \nu v \in L\}$ is the set of suffixes of $u$ in $L$. Without loss of generality, $A$ is assumed live. $A' = (\Sigma', \Sigma, T', s_0, S_M')$ is a sub-automaton of $A$ (or is nested in $A$, denoted $A' \subseteq A$) if $\Sigma' \subseteq \Sigma$, $T' \subseteq T$, and $S_M' \subseteq S_M$.

Diagnosis studies, just like opacity studies, traditionally partition transition signatures into observable and unobservable ones, $\Sigma = \Sigma_o \uplus \Sigma_u$, and consider a distinguished unobservable

---

*LAFAUTUNE, 2018*
Diagnosis issues are more easily discussed on an extension different from the future of the normal run allows:

- ambiguity will last forever, and thus will never betray the firing of a faulty transition, while conversely opacity examines whether diagnosis and opacity problems lose any interest. Specifically, a word (or observation) \( w \in L_N(A) \cap L_F(A) \) is called ambiguous. Diagnosis examines whether this ambiguity will eventually vanish in continuations of \( w \), revealing the firing of a faulty transition, while conversely opacity examines whether ambiguity will last forever, and thus will never betray the firing of a secret transition. Runs \( \pi \) and \( \pi' \) are (observationally) equivalent iff \( \sigma(\pi) = \sigma(\pi') \); this equivalence relation is denoted \( \pi \sim \pi' \). By extension, a faulty (resp. normal) run is said to be ambiguous if it is equivalent to a normal (resp. faulty) one.

The fault detection function \( \text{Diag} : L(A) \to \{T, U, L, R\} \) is defined as

- \( \text{Diag}(w) = T \) if \( w \in L_W(A) \setminus L_N(A) \) or equivalently if \( \sigma^{-1}(w) \subseteq R_P(A) \) : all runs matching \( w \) contain at least one faulty transition,
- \( \text{Diag}(w) = U \) if \( w \in L_N(A) \setminus L_F(A) \) or equivalently if \( \sigma^{-1}(w) \subseteq R_P(A) \) : all runs matching \( w \) are safe, and
- \( \text{Diag}(w) = A \) otherwise (for “ambiguity”).

Diagnosability issues are more easily discussed on an extension of \( A \) that keeps track of the occurrence of a fault. This augmented version of \( A \) writes \( A' = (S \times \{N, F\}, \Sigma, T', (s_0, N), S_M \times \{N, F\}) \). Transitions \( T' \) replicate all transitions of \( A \) as follows:

\( (s, \alpha, s') \in T \in \text{induce transitions } ((s, N), \alpha, (s', F)) \) and \((s, F), (s', F)) \) in \( T' \), and \( (s, \alpha, s') \in T \in \text{induce transitions } ((s, N), \alpha, (s', F)) \) and \((s, F), (s', F)) \) in \( T' \). Observe that runs of \( A \) and \( A' \) are in one to one correspondence through a natural morphism \( \phi : R(A) \to R(A') \). Let \( s \in R(A) \) and \( \pi = s \) if \( \pi \in R_P(A) \) then \( \phi(\pi) = (s, N) \), and if \( \pi \in R_E(A) \) then \( \phi(\pi) = (s, F) \).

To avoid the burden of this heavier notation, one can directly take as starting point a non-deterministic automaton \( A \) where the state space \( S = S_N \sqcup S_F \) is partitioned into safe/normal states and faulty ones, instead of partitioning transitions, and then assume that \( S_P \) is absorbing (no outgoing transitions to \( S_N \)) to account for the permanence of faults. All notions of safe, faulty, just faulty runs/maps extend naturally to this setting: faulty transitions being those transitioning from \( S_N \) to \( S_F \). We use this simplified setting in this paper, but will at places refer to the “traditional” setting.

### 2.1 Recovery point

Let us consider for a while the traditional setting: a nondeterministic \( A \) with transitions partitioned as \( T = T_N \sqcup T_F \) and before the state augmentation. Let \( w = t_1 \ldots t_n \in R_P(A) \) be a faulty run of \( A \), and assume there exists an equivalent safe run \( u \in R_N(A) \) such that \( u^w = w^+ = s \). The future of \( w \) will be no different from the future of the normal run \( u \), so one can assume the system has “recovered” from the fault in \( w \), which also went unnoticed since \( w \sim u \). This of course does not preserve \( A \) from firing another faulty transition after state \( s \). State \( s \) is thus a strong recovery point after \( w \). Assume now that there exists an equivalent safe run \( u \sim w \) with \( u^+ = s' \) and such that \( L_N(A) = L_N(A') \). Then an external observer can not distinguish the future behaviour of \( w \) from the future behavior of the safe run \( u \). The pair of states \((s, s')\) is thus a (weak) recovery point. We shall elaborate on these ideas to define various notions of recovery points. Notice right away that crossing a recovery point does not mean that the fault in \( w \) can never be detected: it is possible that all equivalent safe runs \( u \) fire a faulty transition in their future, leading to detection.

Let us now come back to the simplified setting: a non-deterministic \( A \) with states partitioned as \( S = S_N \sqcup S_F \), \( S_F \) absorbing, and an induced partition \( T = T_N \sqcup T_F \). **Definition 1.** The pair \((s, s') \in S_F \times S_N \) forms a recovery point (RP) of

- type E (Equality) iff \( L_N(A) = L_N(A') \)
- type I (Inclusion) iff \( L_N(A) \subseteq L_N(A') \)
- type EN (Equality Normal) iff \( L_N(A) = L_N(A') \)
- type IN (Inclusion Normal) iff \( L_N(A) \subseteq L_N(A') \)

with the obvious inclusions of types \( E \Rightarrow I \) and \( EN \Rightarrow IN \Rightarrow I \). One can imagine numerous other variants, and for example require that after \( s \) conditionally to the absence of a new fault event, the language is equal to (included in) the normal language after \( s' \), etc. We denote by \( R \subseteq S_F \times S_N \) a given set of recovery points. Notice that checking whether \((s, s')\) forms a recovery point is PSPACE-complete (language inclusion problem for NFA). Here, we assume \( R \) is given beforehand together with \( A \) and needs not be computed. For example, referring to the traditional setting, one may naturally consider pairs \(((s, F), (s, N))\) as recovery points of type E.

#### 2.2 Resilience

**Definition 2.** Given \( n \in \mathbb{N} \), the minimal faulty run \( w \in R^m(A) \) is \( n \)-resilient w.r.t. recovery points \( R \) iff

\[
\forall v' : wv' \in R_F(A) \land |v'| \geq n, \exists v \leq v', \forall u \in R_N(A) : wv \sim u \land (wv^+, w') \in R
\]

(1) \( w \) is resilient when it is \( n \)-resilient for some horizon \( n \in \mathbb{N} \).

In words, along all extensions \( v' \) of length \( n \) following the minimal faulty run \( w \), one will cross a recovery point \( (wv^+, u^+) \). One shall say that resilience “takes place” at \( wv \) on path \( wv' \). See Fig. 1 for examples of resilient and non-resilient faults. Notice that (1) concerns both the past and the future. The past because the recovery point must be jointly reached by the extension \( wv \) and by a safe run \( u \) through the same observed sequence, so that an external observer may not distinguish them. This entails the ambiguity of the faulty run \( w \) at least up to each recovery point. And the future, as by definition the language after the “recovery time” \( wv \) can not be distinguished from the one produced by a continuation of the normal run \( u \).

**Definition 3.** \( A \) is resilient w.r.t. set \( R \) of recovery points iff all minimally faulty runs \( w \in R^m(A) \) are resilient w.r.t. \( R \). This definition may attach a different resilience bound \( n \) to each \( w \), possibly resulting in an unbounded resilience horizon as \( R^m(A) \) can be infinite. A finite uniform bound actually holds: see the decision procedure below, that reduces \( R^m(A) \) to a finite set of interest.
Theorem 4. Given a set \( \mathcal{R} \) of recovery points in \( A \), verifying the resilience of \( A \) is PSPACE-complete.

Proof. We first build a decision procedure, showing the inclusion in PSPACE. Let \( \text{Det}(A) = (Q, \Sigma, T', q_0, Q_m) \) be the deterministic version of \( A \) (classical powerset construction), with \( Q = 2^Q \) and \( q_0 = \{s_0\} \). Consider the product automaton \( A \times \text{Det}(A) \), which simply performs a state augmentation on \( A \) to keep track of states that are jointly accessible through runs of \( A \) and runs of \( A \times \text{Det}(A) \). For \( \pi \in R[A] \), \( \pi^{-1} = s \), one has \( \phi(\pi)^- = (s, q) \) where \( q = \{u, u \in R[A], u \sim \pi\} \) and thus \( s \in q \).

By definition, let us say that state \( (s, q) \in S \times Q \) of \( A \times \text{Det}(A) \) is a recovery point iff \( s \in S_p \) and \( \exists \pi^+ \in q \) s.t. \( (s, \pi') \in \mathcal{R} \). Consider a minimal faulty run \( w \in \mathcal{R}^* \) and its extension \( wv \) with \( wv^+ = s \). By definition, \( w \) reaches a recovery point that is co-accessible by a safe run with the same observation \( \sigma(wv) \) iff \( \phi(wv)^- = (s, q) \) and \( (s, q) \) is a recovery point in \( A \times \text{Det}(A) \).

Verifying \( (\mathcal{R}) \) on \( A \times \text{Det}(A) \) is now straightforward. Minimal faulty runs \( w \in \mathcal{R}^* \) are characterized by a transition from \( S_p \times Q \) to \( S_p \times Q \) in \( A \times \text{Det}(A) \). So one only needs to consider such “just faulty” states in \( A \times \text{Det}(A) \) to capture all just faulty runs. If along all their continuations of length \( n \) they cross a recovery point, then \( (\mathcal{R}) \) holds for range \( n \). Conversely, \( (\mathcal{R}) \) holds for no \( \pi \) iff no some just faulty state \( (s, q) \) there exists a finite path to a circuit (a “lasso”) both avoiding recovery states.

Resilience can thus be checked in polynomial time on \( A \times \text{Det}(A) \), which suggests an exponential time in the size of \( A \).

For the hardness part, we consider the reduction of the universality problem for a non-deterministic automaton, known to be PSPACE-complete. Let \( C = (S^2, \Sigma, T^2, c_0, S_M) \) be an NFA over alphabet \( \Sigma \) and let \( B = (S^3, \Sigma, T^3, b_0, S_M) \) be a deterministic automaton accepting \( \Sigma^* \). As follows (and \( \# \) is supposed to be fresh symbols, not present in \( \Sigma \)). \( S = \{s_0, s_1, s_2\} \) and \( S^2 \approx S^3 \) and \( S_M = \{s_1, s_2\} \).

For transitions, let \( T^\# \cup T^C \subseteq T \), then add \( (s_0, s, b_0) \) and \( (s_0, c_0, 0) \) to \( T \), so \( A \) chooses non-deterministically between \( B \) or \( C \) on its first move. Then add exiting transitions \( (b_0, \#, s_1) \) to all marked states \( b_m \in S_M \) of \( B \), and similarly transitions \( (c_m, \#, s_2) \) to marked states \( c_m \in S^C_M \) in \( C \). Finally, add self-loops \( (s_1, s_1) \) and \( (s_2, s_2) \) to \( T \) to make \( A \) live. Clearly, \( L(A) = \Sigma^* \# \Sigma^* \). Assume \( S_F = \{s_1\} \) so that all words in \( L(A) \) are faulty, and faulty runs all terminate at the absorbing state \( s_1 \). Similarly, all words in \( \Sigma^C \# \Sigma^* \) are safe, and safe runs all terminate at the absorbing state \( s_2 \). Let us now consider as single recovery point in \( A \) the pair \( (s_1, s_2) \). As \( B \) is deterministic, words in \( SL(B) \# = \Sigma^* \# \) are in one-to-one correspondence with minimal faulty runs of \( A \), and such a minimal faulty run \( w \) can be resilient iff for some extension \( wv \) there exists an equivalent safe run \( u \), which necessarily goes through \( C \). Therefore it is not hard to see that \( A \) is resilient iff \( SL(B) \subset SL(C) \), that is if \( B \) accepts \( \Sigma^* \). \( \square \)

Notice that the decision procedure isolates in \( A \) the non-resilient (minimal) faulty runs \( w \) from the resilient ones, and it also provides a unique recovery horizon \( n \) for the latter.

One could have suspected the PSPACE-hardness to derive from the construction of \( \mathcal{R} \), as deciding whether \( (s, s') \) forms a recovery point is already PSPACE-hard. Interestingly, checking resilience remains PSPACE-hard even for a given set \( \mathcal{R} \).

2.3 Relations to diagnosability and opacity

Regarding connections to diagnosability, several immediate observations can be made. First, if some minimal faulty run \( w \in R^m(A) \) is resilient for type EN or type IN, this fault cannot be detected. The reason is that at the moment where resilience takes place, the faulty run \( wv \in R^m(A) \) is equivalent to a safe one \( u \in R^m_N(A) \), so \( wv \) is ambiguous. Since further \( L^m_B(A) \subseteq L^m_N(A) \), this ambiguity never vanishes as any extension \( wv \) of \( wv \) is equivalent to a non-faulty extension \( uv \) of \( u \).

![Figure 1](image_url)

Figure 1. Faulty states are in red. The fault appearing at state 1 is 1-resilient. All faults are diagnosable : detection when \( d \) is observed. The fault appearing at state 8 is not resilient.

Secondly, for resilience of type E or type I, a fault can be both resilient and diagnosable. Consider the example in Fig. 1 where the minimal faulty run on top is 1-resilient but not 0-resilient: as soon as \( b \) or \( c \) is observed after \( a \), a resilience point is reached for \( \mathcal{R} = \{(4, 6), (5, 7)\} \). But as soon as an extra \( d \) is observed, the fault is detected. However, this behavior vanishes if the whole system is E or I resilient.

Proposition 5. If a system is resilient, it cannot be diagnosable.

Proof. This readily holds for EN- and IN-resilience, so we focus on I-resilience, which covers E-resilience as well. Observe first that if a system is both n-resilient, \( n \) being minimal, and m-diagnosable, then necessarily \( n < m \). If \( n \) is minimal, there exists a minimal faulty run \( w \in R^m_n(A) \), an extension \( v \) with \( \|v\| = n \) of \( w \), and a safe run \( u \in R^m(A) \) such that \( wv \sim u \), so ambiguity is still valid \( n \) steps after \( w \). Diagnosability of this fault could only occur with more observations, so \( n + 1 \leq m \).

Recall \( A \) is assumed live. Consider a minimal faulty run \( w \in R^m_n(A) \), and an extension \( v' \) with \( \|v'\| \geq m \). Assume \( w \) is m-diagnosable, so \( \text{Diag}(\sigma(wv')) \neq 1 \) (fault is detected). As \( w \) is n-resilient and \( n < m \) there exists \( v < v' \) and \( u_1 \in R^m(A) \) with \( u_1 \sim wv \), so \( wv \) is ambiguous. Furthermore, as \( Lw(v') \subseteq L_{u_1}(A) \), there exists an extension \( u_1 \sim wv \), and
as $\text{Diag}(\sigma wv') = T$ one has $u_1 x_1 \in R_F(A)$. The faulty run $u_1 x_1$ itself decomposes as $u_1 x_1 = w_1 y_1$ where $w_1$ is a minimal faulty prefix. One has $wv' \sim w_1 y_1$ and $|w_1| > |w|$. 

One can repeat this construction. Considering an extension $w_1 v'_1$ of $w_1 y_1$, with $|v'_1| \geq m$, one can derive $v_2 x_2 \sim w_1 v'_1$ with $u_2 \in R_F(A)$ and $v_2 x_2 \in R_F(A)$, which decomposes as $u_2 x_2 = w_2 y_2$, $v_2 \in R^\text{min}_F(A)$, and $|v_2| > |u_1|$. Although $w_1 v'_1 \sim w_2 y_2$ is longer than $wv'$, a (strict) prefix of $w_1 v'_1$ (and thus of $w_2 y_2$) remains equivalent to $w v'$. In the series of $(u_i, w_i)$ obtained by repeating this procedure, $|u_i|$ and $|w_i|$ are strictly increasing, until one gets $|u_i| - |w_i| > m$, which contradicts the diagnosability of $w$: we have established a safe run equivalent to $wv'$.

We have actually proved a little more: non-diagnosability of a system means that there exists at least one non-diagnosable fault in that system. Here, resilience of a system entails that no fault is actually diagnosable in that system, so one has:

**Corollary 6.** A resilient system is opaque.

Of course, system resilience is strictly stronger than non-diagnosability (Fig. 2). But it is still unclear whether it is also strictly stronger than opacity for NFA with permanent secrets.

### 3. Resilience in Weakly Non-Deterministic Plants

Resilience can be seen as a reinforcement of opacity, the verification of which is also PSPACE-complete on generic nondeterministic plants (Cassez 2012). The only way to reduce this complexity is to consider smaller classes of non-deterministic systems. We concentrate here on history deterministic (HD) automata (Colcombet 2012, Kupferman 2021), as a first step to $k$-width automata (Kuperberg 2019), for which the language inclusion problem is known to be in PTIME instead of PSPACE-complete. This suggests that verifying resilience could also benefit from this gap. Due to space limitations, several proofs of this section are omitted (they will appear in an online and extended version of this work).

#### 3.1 History deterministic automata

We start by recalling several notions of weak non-determinism.

**Definition 7.** Let $A = (S, \Sigma, T, s_0, S_M)$ be an NFA. An admissible strategy (or a resolver) to resolve the non-determinism in $A$ is a mapping $\rho: \Sigma^* \rightarrow S$ such that

- $\rho(\varepsilon) = s_0$,
- $\forall (u, \alpha) \in \Sigma^* \times S$, $u \alpha \in \overline{L(A)} \Rightarrow (\rho(u, \alpha, \rho(u \alpha)) \in T$, and
- $\forall u \in \overline{L(A)}$, $\rho(u) \in S_M$ if $u \in L(A)$.

Automaton $A$ is history deterministic (HD) iff it admits a resolver of its non-determinism.

For any prefix $u$ of a word in $L(A)$ and any action $\alpha$, $\rho(u \alpha)$ chooses the resulting state in $A$ when firing $\alpha$ from state $\rho(u)$. This choice only depends on the beginning of the word, $u \alpha$, and does not reduce the language of $A$ (last item), so in particular it preserves all suffixes of $u \alpha$ without any look ahead. Whence the name history deterministic. This is better illustrated by the following result (Kuperberg 2019).

**Lemma 8.** Strategy $\rho$ is admissible for $A$ iff $\forall u \in \overline{L(A)}$, $L_{\rho(u)}(A) = u^{-1} L(A)$.

![Figure 2. A non-diagnosable system which is also non-resilient.](image)

There exist related notions of weak non-determinism, see (Kupferman 2021). For example, $A$ is said to be determinizable by pruning (DBP) iff by removing transitions from $A$ one obtains an equivalent deterministic automaton $A'$. Clearly, if $A$ is DBP then it is also HD, as one can define the resolver $\rho$ from the remaining transitions in $A'$. Being state-based, this resolver has a finite memory. $A$ is said to be semantically deterministic (SD) iff $\forall (s, \alpha, s') \in S \times \Sigma, (s, \alpha, s') \in T \Rightarrow L_{\rho}(A) = L_{\rho'}(A)$. In other words, whatever the way non-determinism is solved at $s$ for action $\alpha$, this does not change the future language of $A$. So SD implies DBP as cutting all branches but one at a non-deterministic choice preserves the language (see Lemma 7). Finally, an HD automaton contains an equivalent SD automaton. This last point derives from the following observation. Consider the sub-automaton $A'$ of $A$ obtained by removing transitions which are never chosen by resolver $\rho$, i.e. by keeping only transitions $(\rho(u), \alpha, \rho(u \alpha))$ for $u \alpha \in L(A)$. As $A$ is HD, $L(A') = L(A)$, but $A'$ is not deterministic: one may have $\rho(u) = \rho(v) = s$, but $\rho(u \alpha) = s' \neq \rho(v \alpha) = s''$, so the two transitions $(s, \alpha, s')$ and $(s, \alpha, s'')$ remain in $A'$. Nevertheless, one has $L_{\rho}(A') = L_{\rho'}(A')$, which makes $A'$ SD. This derives from Lemma 8. Let $w \in L_{\rho}(A) = (u \alpha)^{-1} L(A)$, so $\omega w \in L(A)$. But as $\rho(u) = s = \rho(v)$, one has that $\forall w \omega \in L(A)$, and so $w \in (\alpha \omega)^{-1} L(A') = L_{\rho'}(A')$ (see also Prop. 14 in Colcombet 2012).

So the above notions of non-determinism - DBP, HD/GFG, SD - almost coincide for automata on finite words, and in the sequel we use indistinguishably HD and DBP.

#### 3.2 Determinization procedure

**Theorem 9.** Deciding whether an NFA is determinizable by pruning (i.e. DBP, HD, GFG, or SD) is in PTIME.

This result appears as Theorem 15 in (Colcombet 2012), without proof, as Theorem 4.1 in (Kupferberg 2010) which provides a first proof, and as Theorem 2 in (Löding 2020) in the setting of transducers. Proofs of Theorem 9 are constructive: Given some NFA $A$, they try to build a nested deterministic automaton $A'$ equivalent to $A$ by pruning transitions in $A$. The construction fails iff $A$ is not DBP, otherwise it provides one (or all) equivalent nested $A'$. An alternative version of this proof will appear in the extended version of this work. It relies on the following central notions and properties.

**Definition 10.** In NFA $A = (S, \Sigma, T, s_0, S_M)$, transition (branch) $(s, \alpha, s') \in T$ dominates $(s, \alpha, s'') \in T$ iff $L_{\rho'}(A) \subseteq L_{\rho''}(A)$. Branch $(s, \alpha, s')$ is dominant iff $\forall s''$ such that $(s, \alpha, s'') \in T$, $L_{\rho''}(A) \subseteq L_{\rho'}(A)$.

For $(s, \alpha) \in S \times \Sigma$, the dominance relation defines a partial order on branches $(s, \alpha, \cdot)$, so there can be several dominant branches or none at $s$ for label $\alpha$.
Lemma 11. Assume $A$ is DBP, and let $A'$ be an equivalent deterministic automaton nested in $A$. Then $A'$ uses only dominant branches of $A$.

If there are several dominant branches in $A$ at state $s$ for label $\alpha$, we show below that they are equivalent to build $A'$ (Lemma 14). If there are none while $\alpha$ is firable at $s$, then $s$ must not be reachable by any $u$ in any $A'$. This means both that $s$ is a "bad" state of $A$ for a candidate $A'$, and that words $wv \in L(A)$ that can be accepted through $s$ in $A$ must be realized through other states in $A'$.

Definition 12. The sub-automaton of dominant branches of $A$, denoted $A''$, is obtained by removing all non-dominant transitions of $A$ and trimming\(^1\) the result.

This makes $A''$ a semantically deterministic (SD) automaton. So $A''$ could also be called the SD segment of $A$.

Proposition 13. $A$ is DBP iff its sub-automaton of dominant branches $A'' \subseteq A$ satisfies $L(A'') = L(A)$. In that case, for any $s$ in $A$ that remains reachable in $A''$, one has $L_s(A'') = L_s(A)$.

So the effort of the DBP test for $A$ all lies in pruning non-dominant branches to obtain $A''$, and checking that $L(A)$ is preserved. In conjunction with Lemma 11 when $A$ is DBP, Prop. 13 also shows that all deterministic equivalent automata $A'$ nested into $A$ are also nested into $A''$.

Lemma 14. Let $A_n$ be an automaton made of dominant branches only (i.e. $A_n$ is SD), and let $A_{n+1}$ be obtained by removing one branch at a non-deterministic choice in $A_n$. Then $A_{n+1}$ is also made of dominant branches only, and satisfies $L(A_{n+1}) = L(A_n)$ and further $\forall s \in S, L_s(A_{n+1}) = L_s(A_n)$.

Proposition 15. If $A$ is DBP, given its automaton of dominant branches $A''$, any equivalent deterministic $A''$ nested into $A$ can be obtained by a greedy determinization at all states of $A''$.

This is an immediate consequence of Lemma 13 used in the proof of Prop. 13 as one can form a decreasing sequence $A'' = A_0 \supset A_1 \supset \ldots \supset A_N = A'$ preserving $L(A)$, where two consecutive $A_n$ only differ by one transition (reducing non-determinism for one letter $a \in \Sigma$ at some state $s$). This result allows one to easily enumerate all equivalent $A'$ for $A$. See Fig. 3.

We finish with a useful property for resilience verification.

Proposition 16. Let $A$ be DBP, with $A''$ as its nested SD segment. Let $wv \in R(A)$ be a run of $A$ such that $wv^+ = s$ and $L_s(A) \neq \emptyset$. Consider any run $u \in R(A'')$ such that $u \sim wv$, and let $s' = u^+$ in $A''$. One has $L_s(A) \subseteq L_{s'}(A'' = L_{s'}(A)$.

This resilience could also be brought back to PTIME. Notice that determining an HD automaton should go through the identification of a nested equivalent subautomaton $A'$ rather than the subset construction: the latter can still suffer an exponential blowup (Kuperberg 2019).

3.3 Resilience in HD automata

We consider here the type 1 notion of resilience, expressed directly through language inclusion rather than recovery points. Specifically, at most $n$ steps after the fault occurrence, the future language of a faulty run must be covered by the future language of an equivalent safe run.

Definition 17. Let $w \in R_F^n(A)$ be a minimal faulty run of $A$, $w$ is ($I$-)resilient iff

$$\exists n \in \mathbb{N}, \forall v : wv \in L_F(A) \land |v| \geq n, \exists v' \leq v', \exists u \in R_N(A) : u \sim v \land L_{wv'}(A) \subseteq L_{u'}(A)$$

$A$ is ($I$-)resilient iff all its minimal faulty runs are.

While HD automata form a convenient sub-class of NFA where language inclusion can be easily checked, they are not yet sufficient to easily check resilience. A counter-example appears in Fig. 4. This resilient system $A$, for $n = 0$, is also HD, but its determined version only preserves faulty runs, as they dominate safe ones. Therefore in any $A'$ or even $A''$ resilience vanishes. The phenomenon remains if an extra transition $(2, c, 7)$ is added, as some nested deterministic segment $A' \subseteq A''$ could either preserve the faulty branch (through 1) or the safe one (through 2).

![Figure 4](image-url)
Lemma 19. Strategy $\rho$ is an admissible FM-resolver for $A$ iff $\forall u \in L(A), L_{P(\rho)}(A) = u^{-1}L(A) \land L_{P(\rho),N}(A) = u^{-1}L_N(A)$.

Observe that if $u \in L_N(A)$, then $u^{-1}L_N(A) \neq \emptyset$ so $P(u)$ must belong to $S_N$, while this constraint does not hold otherwise. This translates the fact that safe runs are preferred to faulty ones in the determination of $A$, and leads to a stronger notion of dominance between transitions.

Definition 20. Transition $(s, \alpha, s')$ (FM)-dominates $(s, \alpha, s'')$ in $A$ iff $L_{s'}(A) \subseteq L_s(A)$ and $L_{s''}(A) \subseteq L_{s'}(A)$. Branch $(s, \alpha, s''')$ is (FM)-dominant iff it (FM)-dominates all other transitions $(s, \alpha, s')$ of $A$ rooted at $s$ for label $\alpha$.

With this new notion of dominance, Def. 1 applies and results like Prop. 13 follow, with the preservation of both $L(A)$ and $L_N(A)$ for DBP-FM automata. Similarly, Lem. 14 Prop. 15 Thm. 16 generalize to DBP-FM automata, and Prop. 18 becomes Proposition 21. Let $A$ be DBP-FM, with $A''$ as its nested SD segment. Let $wv \in R(A)$ be a run of such a way that $wv = s$ and $L_s(A) \neq \emptyset$. Consider any run $u \in R(A'')$ such that $u \sim wv$, and let $s' = u^+ + v^+$ in $A'$. One has $L_s(A) \subseteq L_{s'}(A'') \subseteq L_{s''}(A)$ and $L_{N_s}(A) \subseteq L_{N_{s''}}(A'')$. This entails that $u \sim vw$ and $s' \in S_N$, i.e. $u \in R_s(A'''),$ which proves the resilience of $w$ in $A''$.

Proof. From Thm. 22 and the following remarks, $A$ is resilient if it is equivalent to a deterministic $A' \subseteq A'' \subseteq A$ which is also resilient. But resilience in a deterministic $A'$ means that there are no faults, thus no faulty state. As the resilience of $A''$ transfers into resilience of any equivalent nested $A'$, this means that $A''$ also has no faulty state.

So intuitively, when building $A'$, resilient faulty runs are covered by safe paths, and thus vanish from the construction. The burden of verifying resilience of DBP-FM automata then all lies in the derivation of $A''$, which was shown to be in PTIME (generalization of Thm. 9).

4. CONCLUSION

This paper explored the notion of fault resilience for DES, as the property of not damaging too much the expected behavior of the system from the perspective of an external observer. Verifying resilience is PSPACE-complete for general NFA, but in the restricted sub-class of HD systems, it becomes polynomial (even quadratic), making this notion of practical interest. These ideas potentially adapt to other problems, like opacity verification. An extended version of this work will address the case of $k$-width automata, that cover all the NFA class as $k$ varies. Numerous notions of resilience can be envisioned, beyond recovery points or language inclusions, and quantitative versions would be useful, for example to measure the speed at which the “memory” of some fault vanishes.

REFERENCES

[Laforet 2018] Stéphane Laforet, Feng Lin, Christoforos N.Hadjicostis. On the history of diagnosability and opacity in discrete event systems. Annual Reviews in Control 45:257-266, 2018. https://doi.org/10.1016/j.arcontrol.2018.04.002

[Colcombet 2012] Thomas Colcombet. Forms of Determinism for Automata. STACS’12 (29th Symposium on Theoretical Aspects of Computer Science), Feb 2012, Paris, France. pp.21-23. https://hal.archives-ouvertes.fr/hal-00678155/document

[Kuperberg 2015] Denis Kuperberg, Michał Skrzypczak. On Determinisation of Good-for-Games Automata. ICALP (Automata Languages and Programming) 2015. LNCS vol. 9135 pp. 299-310. https://doi.org/10.1007/978-3-662-47666-6_24

[Kuperberg 2019] Denis Kuperberg, Anirban Majumdar. Computing the Width of Non-deterministic Automata. Logical Methods in Computer Science, vol.15, No.4. 2019. https://doi.org/10.23638/LMCS-15(4:10)2019

[Löding 2020] Emmanuel Filiot, Christof Löding, Sarah Winter. Synthesis on Algorithms, Vol. 6(2) pp. 1-36, March 2010. https://doi.org/10.1016/11874683_26

[Laforet 2018] Stéphane Laforet, Feng Lin, Christoforos N.Hadjicostis. On the history of diagnosability and opacity in discrete event systems. Annual Reviews in Control 45:257-266, 2018. https://doi.org/10.1016/j.arcontrol.2018.04.002

[Cassez 2012] Franck Cassez, Jeremy Dubreil, Hervé Marchand. Synthesis of Opaque Systems with Static and Dynamic Masks. Formal Methods in System Design 40(1):88-115, 2012. https://doi.org/10.1007/s11047-013-9147-6

[Henzinger 2006] Thomas A. Henzinger, Nir Piterman. Solving Games Without Determinization. Computer Science Logic 2006. LNCS vol. 4207, pp. 395-410. https://doi.org/10.1007/11874683_26

[Colcombet 2012] Thomas Colcombet. Forms of Determinism for Automata. STACS’12 (29th Symposium on Theoretical Aspects of Computer Science), Feb 2012, Paris, France. pp.21-23. https://hal.archives-ouvertes.fr/hal-00678155/document

[Kuperberg 2015] Denis Kuperberg, Michał Skrzypczak. On Determinisation of Good-for-Games Automata. ICALP (Automata Languages and Programming) 2015. LNCS vol. 9135 pp. 299-310. https://doi.org/10.1007/978-3-662-47666-6_24

[Kuperberg 2019] Denis Kuperberg, Anirban Majumdar. Computing the Width of Non-deterministic Automata. Logical Methods in Computer Science, vol.15, No.4. 2019. https://doi.org/10.23638/LMCS-15(4:10)2019

[Kupferman 2010] Benjamin Aminof, Orna Kupferman, Robby Lampert. Reasoning about online algorithms with weighted automata. ACM Transactions on Algorithms, Vol. 6(2) pp. 1-36, March 2010. https://doi.org/10.1145/1721837.1721844

[Kupferman 2021] Bader Abu Radi, Orna Kupferman, Ofer Leshkowitz. A Hierarchy of Nondeterminism. 46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021), pp. 85-1-21. DOI: 10.4208/LMPTcs.MFCS2021.85. https://drops.dagstuhl.de/opus/volltexte/2021/145257

[Löding 2020] Emmanuel Filiot, Christof Löding, Sarah Winter. Synthesis from weighted specifications with partial domains over finite words. In FSTTCS 2020, vol.182 of LIPIcs, pp. 46:1–46:16. Extended version with proofs at http://arxiv.org/abs/2103.05550v1