Resilience in Discrete Event Systems

Eric Fabre

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 damageable 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 some distant server address holding a database is unspecified or the server is not responding (the "fault"), an address book of backup servers can be used or an old partial copy of the database 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) $A = (S, \Sigma, T, s_0, S_M)$ over alphabet Σ, 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 $t^- = s, t^+ = s'$ and $\sigma(t) = \alpha$. A path in A is a sequence of transitions $\pi = t_1 ... t_n$ such that $t_i^+ = t_{i+1}^-$ for $1 \leq i < n$. By extension, $\pi^- = t_1^-$, $\pi^+ = t_n^+$, $\sigma(\pi) = \sigma(t_1)...\sigma(t_n)$, 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 π 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. $R^a(A)$). The language $L(A) \subseteq \Sigma^*$ of A is formed by the signatures of accepted runs $L(A) = \sigma(R^a(A))$. 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 $L \subseteq \Sigma^*$, $\overline{L} = \{u : \exists v, uv \in L\}$ is the prefix closure of L, and $u^{-1}L = \{v : uv \in L\}$ is the set of suffixes of u in L. Without loss of generality, A is assumed live. $A' = (S', \Sigma, T', s_0, S'_M)$ is a sub-automaton of A (or is nested in A, denoted $A' \subseteq A$) if $S' \subseteq S$, $T' \subseteq T$, 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

label of interest $f \in \Sigma_u$ called a fault.This starting point then requires extra processings, like the removal of silent transitions through ε-reduction. Equivalently, one can directly assume that all transitions are observable (*i.e.* $\Sigma = \Sigma_o$) and that they are partitioned into normal (or safe) ones and faulty (or secret) ones : $T = T_N \uplus T_F$. This partition extends to runs : $R(A) = R_N(A) \uplus R_F(A)$, with $R_N(A) = R(A) \cap T_N^*$. Run $\pi = t_1 ... t_n$ is a minimal faulty (or just faulty) run iff $\pi \in R_F(A)$ and $t_1 ... t_{n-1} \in R_N(A)$. $R_F^{min}(A)$ denotes the set of such minimal faulty runs. The partition of T also induces the normal language $L_N(A) = \sigma(R_N(A))$ and the faulty language $L_F(A) = \sigma(R_F(A))$. As A is non-deterministic, $L(A) = L_N(A) \cup L_F(A)$ is *not* a partition, otherwise 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 π and π' 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 $Diag : L(A) \to \{\top, \bot, \mathrm{A}\}$ is defined as

- $Diag(w) = \top$ if $w \in L_F(A) \setminus L_N(A)$ or equivalently if $\sigma^{-1}(w) \subseteq R_F(A)$: all runs matching w contain at least one faulty transition,
- $Diag(w) = \bot$ if $w \in L_N(A) \setminus L_F(A)$ or equivalently if $\sigma^{-1}(w) \subseteq R_N(A)$: all runs matching w are safe, and
- $Diag(w) = \mathrm{A}$ otherwise (for "ambiguity").

Diagnosibility 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 : $(s,\alpha,s') \in T_N$ induce transitions $((s,N),\alpha,(s',N))$ and $((s,F),\alpha,(s',F))$ in T', and $(s,\alpha,s') \in T_F$ induce transitions $((s,N),\alpha,(s',F))$ and $((s,F),\alpha,(s',F))$ 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 $\pi \in R(A)$ and $\pi^+ = s$, if $\pi \in R_N(A)$ then $\phi(\pi)^+ = (s,N)$, and if $\pi \in R_F(A)$ 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 $S = S_N \uplus S_F$ is partitioned into safe/normal states and faulty ones, instead of partitioning transitions, and then assume that S_F is absorbing (no outgoing transitions to S_N) to account for the permanence of faults. All notions of safe, faulty, just faulty runs/words extend naturally to this setting : faulty transitions being those transiting 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 non-deterministic A with transitions partitioned as $T = T_N \uplus T_F$ and before the state augmentation. Let $w = t_1 ... t_n \in R_F(A)$ be a faulty run of A, and assume there exists an equivalent safe run $u \in R_N(A)$ such that $u^+ = w^+ = s$. The future of run 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 $L_s(A) = L_{s'}(A)$. 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 $S = S_N \uplus S_F$, S_F absorbing, and an induced partition $T = T_N \uplus T_F$.

Definition 1. The pair $(s,s') \in S_F \times S_N$ forms a recovery point (RP) of

- type E (Equality) iff $L_s(A) = L_{s'}(A)$
- type I (Inclusion) iff $L_s(A) \subseteq L_{s'}(A)$
- type EN (Equality Normal) iff $L_s(A) = L_{N,s'}(A)$
- type IN (Inclusion Normal) iff $L_s(A) \subseteq L_{N,s'}(A)$

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 $\mathscr{R} \subseteq S_F \times S_N$ 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 $\mathscr{R}$ 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 $w \in R_F^{min}(A)$ is n-resilient w.r.t. recovery points $\mathscr{R}$ iff

$$\forall v' : wv' \in R_F(A) \wedge |v'| \geq n,$$
$$\exists v \leq v',\ \exists u \in R_N(A) :\ wv \sim u \wedge (wv^+, u^+) \in \mathscr{R} \qquad (1)$$

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 $\mathscr{R}$ of recovery points iff all miminally faulty runs $w \in R_F^{min}(A)$ are resilient w.r.t. $\mathscr{R}$.

This definition may attach a different resilience bound n to each w, possibly resulting in an unbounded resilience horizon as $R_F^{min}(A)$ can be infinite. A finite uniform bound actually holds : see the decision procedure below, that reduces $R_F^{min}(A)$ to a finite set of interest.

Theorem 4. Given a set $\mathscr{R}$ 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 $Det(A) = (Q, \Sigma, T', q_0, Q_M)$ be the determinized version of A (classical powerset construction), with $Q = 2^S$ and $q_0 = \{s_0\}$. Consider the product automaton $A \times Det(A)$, which simply performs a state augmentation on A to keep track of states that are jointly accessible through the same observation. Specifically, there is a natural one-to-one morphism $\phi : R(A) \to R(A \times Det(A))$ between runs of A and runs of $A \times Det(A)$. For $\pi \in R(A)$, $\pi^+ = s$, one has $\phi(\pi)^+ = (s,q)$ where $q = \{u^+, u \in R(A), u \sim \pi\}$ and thus $s \in q$. By extension, let us say that state $(s,q) \in S \times Q$ of $A \times Det(A)$ is a recovery point iff $s \in S_F$ and $\exists s' \in q \cap S_N$, $(s,s') \in \mathscr{R}$. Consider a minimal faulty run $w \in R^{min}(A)$ and its extension wv with $wv^+ = s$. By definition, wv 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 $A \times Det(A)$. Verifying (1) on $A \times Det(A)$ is now straightforward. Minimal faulty runs $w \in R^{min}(A)$ are characterized by a transition from $S_N \times Q$ to $S_F \times Q$ in $A \times Det(A)$. So one only needs to consider such "just faulty" states in $A \times Det(A)$ to capture all just faulty runs. If along all their continuations of length n they cross a recovery point, then (1) holds for range n. Conversely, (1) holds for no n iff after 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 $A \times Det(A)$, which suggests an exponential time in the size of A. To get the tighter PSPACE bound, consider checking non-resilience instead. This amounts to proving the existence of a non-resilient "lasso" in $A \times Det(A)$, that is a just faulty run followed by a path and then a (single run of a) loop, both free of recovery points. A lasso contains no state repetition excepted the last one which closes the loop, so one only needs to examine paths of length at most $|S| \times 2^{|S|} + 1$. Given such a path, one needs $O(\log(|S| \times 2^{|S|})) = O(|S|)$ bits to store a position along this path. Two positions are enough to detect occurrence of the first loop and thus check the lasso property, and one position is enough to detect fault occurrence followed by non-resilience along the lasso. Therefore a non-deterministic machine can check non-resilience of A with polynomial space, which proves that verifying Def.(3) lies in co-NPSPACE. We conclude by NPSPACE=PSPACE (Savitch's theorem) and by the closure of PSPACE by complementation.

For the hardness part, we consider the reduction of the universality problem for a non-deterministic automaton, known to be PSPACE-complete. Let $C = (S^C, \Sigma, T^C, c_0, S^C_M)$ be an NFA over alphabet Σ, and let $B = (S^B, \Sigma, T^B, b_0, S^B_M)$ be a deterministic automaton accepting Σ^*. Let us build $A = (S, \Sigma \uplus \{\$, \#\}, s_0, T, S_M)$ by assembling B and C as follows ($\$$ and $\#$ are supposed to be fresh symbols, not present in Σ). $S = \{s_0, s_1, s_2\} \uplus S^B \uplus S^C$, $S_M = \{s_1, s_2\}$. For transitions, let $T^B \uplus T^C \subseteq T$, then add $(s_0, \$, b_0)$ and $(s_0, \$, c_0)$ to T, so A chooses non-deterministically between B or C on its first move. Then add exiting transitions $(b_m, \#, s_1)$ to all marked states $b_m \in S^B_M$ in 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, $L(A) = \$\Sigma^*\#\$^*$. 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 $\$L(C)\#\$^*$ 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 $\$L(B)\# = \$\Sigma^*\#$ 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 $\$L(B)\# \subseteq \$L(C)\#$, that is iff B accepts Σ^*. □

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

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

2.3 Relations to diagnosability and opacity

Regarding connections to diagnosability, several immediate observations can be made. First, if some minimal faulty run $w \in R^{min}_F(A)$ 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 $wv \in R_F(A)$ is equivalent to a safe one $u \in R_N(A)$, so wv is ambiguous. Since further $L_{wv^+}(A) \subseteq L_{u^+,N}(A)$, this ambiguity never vanishes as any extension wvx of wv is equivalent to a non-faulty extension uy of u.

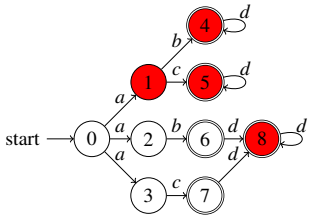

Figure 1. Faulty states are in red. The fault appearing at state 1 is 1-resilient. All faults are diagnosable : detection when d is observed. The fault appearing at state 8 is not resilient.

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 $\mathscr{R} = \{(4,6),(5,7)\}$. 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 $w \in R^{min}_F(A)$, an extension v with $|v| = n$ of w, and a safe run $u \in R_N(A)$ 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 mininal faulty run $w \in R^{min}_F(A)$, and an extension v' with $|v'| \geq m$. Assume w is m-diagnosable, so $Diag(\sigma(wv')) = \top$ (fault is detected). As w is n-resilient and $n < m$ there exists $v < v'$ and $u_1 \in R_N(A)$ with $u_1 \sim wv$, so wv is ambiguous. Furthermore, as $Lwv^+(A) \subseteq L_{u_1^+}(A)$, there exists an extension $u_1 x_1$ such that $u_1 x_1 \sim wv'$, and

as $Diag(\sigma(wv')) = \top$ one has $u_1x_1 \in R_F(A)$. The faulty run u_1x_1 itself decomposes as $u_1x_1 = w_1y_1$ where w_1 is a minimal faulty prefix. One has $wv' \sim w_1y_1$ and $|w_1| > |w|$.

One can repeat this construction. Considering an extension $w_1v'_1$ of w_1y_1, with $|v'_1| \geq m$, one can derive $u_2x_2 \sim w_1v'_1$ with $u_2 \in R_N(A)$ and $u_2x_2 \in R_F(A)$, which decomposes as $u_2x_2 = w_2y_2$, $w_2 \in R_F^{min}(A)$, and $|u_2| > |u_1|$. Although $w_1v'_1 \sim w_2y_2$ is longer than wv', a (strict) prefix of $w_1v'_1$ (and thus of w_2y_2) remains equivalent to wv. In the series of (u_i, w_i) obtained by repeating this procedure, $|u_i|$ and $|w_i|$ are strictly increasing, until one gets $|u_i| - |w| > m$, which contradicts the diagnosability of w: we have evidenced 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 non-deterministic plants (Cassez 2012). The only way to reduce this complexity is to consider smaller classes of non-deterministic systems. We concentrate here on history determinisitc (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 $\rho : \Sigma^* \to S$ such that

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

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 α, $\rho(u\alpha)$ chooses the resulting state in A when firing α 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 ρ is admissible for A iff $\forall u \in \overline{L(A)}$, $L_{\rho(u)}(A) = u^{-1}L(A)$.

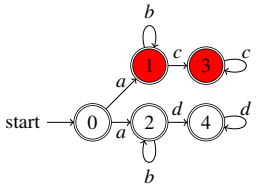

Figure 2. A non-diagnosable system which is also non-resilient. Replacing d by c would grant resilience.

There exist related notions of weak non-determinism, see (Kupferman 2021). For example, A is said to be *determinisable 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 ρ from the remaining transitions in A'. Being state-based, this resolver has a finite memory. A is said to be *semantically deterministic* (SD) iff $\forall (s, \alpha) \in S \times \Sigma,\ (s, \alpha, s') \in T \wedge (s, \alpha, s'') \in T \ \Rightarrow\ L_{s'}(A) = L_{s''}(A)$. In other words, whatever the way non-determinism is solved at s for action α, 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 14). 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 ρ, *i.e.* by keeping only transitions $(\rho(u), \alpha, \rho(u\alpha))$ for $u\alpha \in \overline{L(A)}$. 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, α, s') and (s, α, s'') remain in A'. Nevertheless, one has $L_{s'}(A') = L_{s''}(A')$, which makes A' SD. This derives from Lemma 8: Let $w \in L_{s'}(A') = (u\alpha)^{-1}L(A')$, so $u\alpha w \in L(A')$. But as $\rho(u) = s = \rho(v)$, one has that $v\alpha w \in L(A')$, and so $w \in (v\alpha)^{-1}L(A') = L_{s''}(A')$ (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 indistinctly HD and DBP.

3.2 Determinization procedure

Theorem 9. Deciding whether an NFA is determinisable 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 (Kupferman 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 $L_{s''}(A) \subseteq L_{s'}(A)$. Branch (s, α, s') is dominant iff $\forall s''$ such that $(s, \alpha, s'') \in T,\ L_{s''}(A) \subseteq L_{s'}(A)$.

For $(s, \alpha) \in S \times \Sigma$, the dominance relation defines a partial order on branches $(s, \alpha, .)$, so there can be several dominant branches or none at s for label α.

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 α, we show below that they are equivalent to build A' (Lemma 14). If there are none while α 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 $u\alpha v \in L(A)$ 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 $s \in S$ 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-domininant 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 14 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 $L_s(A) \subseteq L_{s'}(A'') = L_{s'}(A)$.

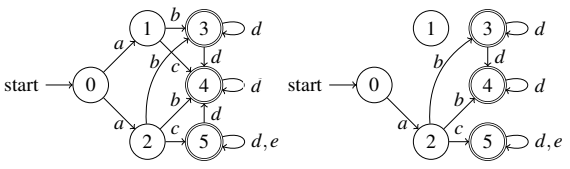

Figure 3. A DBP automaton A (left) and its SD segment A'' (right) obtained by selecting dominant branches and trimming.

HD automata are thus convenient as they can easily be determinised (by pruning), which brings back problems like language inclusion from PSPACE to PTIME and suggests that resilience could also be brought back to PTIME. Notice that determinizing 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 I 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 $w \in R_F^{min}(A)$ be a minimal faulty run of A, w is (I-)resilient iff

$$\exists n \in \mathbb{N},\ \forall v' : wv' \in R_F(A) \wedge |v'| \geq n,\ \exists v \leq v',$$
$$\exists u \in R_N(A) :\ wv \sim u \ \wedge\ L_{wv^+}(A) \subseteq L_{u^+}(A) \qquad (2)$$

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 determinized 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).

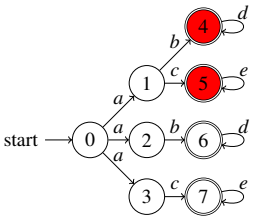

Figure 4. A resilient DBP automaton. Determinization by pruning kills resilience, as only transition $(0, a, 1)$ is preserved for label a : the two safe branches to 2 and 3 are dominated.

To go around this difficulty, one needs to express that safe paths are preferable to faulty ones in the determinisation by pruning. In this way, the pruning both deals with language inclusion and removes faulty segments when safe ones cover them. This motivates the restriction to HD automata *with fault memory*.

Definition 18. NFA A is history deterministic with fault memory (HD-FM, or DBP-FM) iff it admits a resolver of its nondeterminism that also preserves the normal language, *i.e.*

- $\forall u \in \overline{L_N(A)},\ \rho(u) \in S_M \cap S_N \ \Leftrightarrow\ u \in L_N(A)$

ρ is then called an admissible FM-resolver (a resolver with fault memory).

The example in Fig. 4 is not DBP-FM, unless the extra transition $(2, c, 7)$ is added. In that case, the FM-resolver choses branch $(0, a, 2)$ instead of branch $(0, a, 1)$, to favor safe runs.

This notion is associated to a stronger version of Lemma 8

[1] *i.e.* removing all states that are non-accessible or non-coaccessible

Lemma 19. Strategy ρ is an admissible FM-resolver for A iff $\forall u \in \overline{L(A)},\ L_{\rho(u)}(A) = u^{-1}L(A) \wedge L_{\rho(u),N}(A) = u^{-1}L_N(A)$.

Observe that if $u \in \overline{L_N(A)}$, then $u^{-1}L_N(A) \neq \emptyset$ so $\rho(u)$ 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 determinization of A, and leads to a stronger notion of dominance between transitions.

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

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. 9 generalize to DBP-FM automata, and Prop. 16 becomes

Proposition 21. Let A be DBP-FM, 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 $L_s(A) \subseteq L_{s'}(A'') = L_{s'}(A)$ and $L_{s,N}(A) \subseteq L_{s',N}(A'') = L_{s',N}(A)$.

With this material, one gets the following properties.

Theorem 22. Let A be a DBP-FM automaton. Then A is resilient iff its nested SD-segment A'' is resilient.

Proof. Without loss of generality, one can assume that each normal state of A leads to at least one normal word: $\forall s \in S_N, L_{s,N}(A) \neq \emptyset$.

($\Rightarrow$) Assume A is resilient. Let w be a mininal faulty run in A'', $w \in R_F^{min}(A'')$, and let v' be a continuation of w in A'' with $|v'| \geq n$. As $A'' \subseteq A$ and by the resilience of A, there exists $v \leq v'$ and an equivalent safe run $u \in R_N(A)$ such that $u \sim wv$ and $L_{wv^+}(A) \subseteq L_{u^+}(A)$. One has $s = u^+ \in S_N$, so $L_{s,N}(A) \neq \emptyset$. Consider any $u' \in R(A'')$ such that $u' \sim u$ (there exists at least one), and let $s' = u'^+$. By Prop. 21, $L_s(A) \subseteq L_{s'}(A'')$ and $\emptyset \neq L_{s,N}(A) \subseteq L_{s',N}(A'')$. This entails that $u' \sim wv$ and $s' \in S_N$, *i.e.* $u' \in R_N(A'')$, which proves the resilience of w in A''.

($\Leftarrow$) Assume A'' is resilient. Let $w \in R_F^{min}(A)$, there exists $\bar{w} \in R(A'')$ such that $w \sim \bar{w}$, and $L_{w^+}(A) \subseteq L_{\bar{w}^+}(A'')$. If $\bar{w}^+ \in S_N$, then w is resilient in A. Otherwise, $\bar{w} \in R_F(A'')$ is a faulty run, not necessarily minimal. Let v' be a continuation of w in A with $|v'| \geq n$, then there exists a continuation $\bar{v}'$ of $\bar{w}$ in A'' with $wv' \sim \bar{w}\bar{v}'$, and $|\bar{v}'| \geq n$. From the resilience of A'', there exists $\bar{v} \leq \bar{v}'$ and $u \in L_N(A'')$ such that $L_{\bar{w}\bar{v}^+}(A'') \subseteq L_{u^+}(A'')$. By Prop. 21, one even has $L_{\bar{w}\bar{v}^+}(A'') = L_{u^+}(A'') = L_{u^+}(A)$. Let $v \leq v'$ be such that $wv \sim \bar{w}\bar{v} \sim u$, then $L_{wv^+}(A) \subseteq L_{\bar{w}\bar{v}^+}(A'') = L_{\bar{u}^+}(A)$, which proves the resilience of w in A. □

With a similar reasoning, one can actually derive finer properties. A minimal faulty run $w \in R_F^{min}(A)$ of A is resilient iff it is equivalent to a resilient faulty run w' in A''. This also entails that all such equivalent faulty runs w' in A'' are either all resilient, or none is. Consequently, the resilience of A is equivalent to the resilience of any deterministic $A' \subseteq A''$ equivalent to A'' and thus to A, where equivalence here means both the preservation of the language $L(A)$ and of the safe language $L_N(A)$. This leads to the following result.

Theorem 23. A DBP-FM automaton A is resilient iff its SD-segment A'' contains no faulty state. Consequently, verifying the resilience of a DBP-FM automaton is in PTIME.

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

[Lafortune 2018] Stéphane Lafortune, 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/s10703-012-0141-9

[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.1-23. https://hal.archives-ouvertes.fr/hal-00678155/document

[Kuperberg 2015] Denis Kuperberg, Michal 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(4), Nov. 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: 10.4230/LIPIcs.MFCS.2021.85, https://drops.dagstuhl.de/opus/volltexte/2021/14525/

[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.