Towards a Unifying View of Fault Propagation Analyses and Notations*

Marco Bozzano[1][0000−0002−4135−103X], Alessandro Cimatti[0000−0002−1315−6990], Alberto Griggio[0000−0002−3311−0893], and Fajar Haifani[0000−0001−5139−4503]

Fondazione Bruno Kessler, Trento 38123, Italy

Abstract. The design of complex systems requires a careful consideration of the possible hazards and failure conditions that may affect system functions, possibly compromising system reliability and safety. Complex systems must be able to detect components faults and isolate them before they can propagate and cause system failures. To this aim, Preliminary Safety Assessment analyzes failure conditions and allocate safety requirements to components and subsystems, based on a candidate system architecture. A modern way to conduct this analysis is via the use of fault propagation models, i.e. formal representations linking the occurrence of basic faults to their effects on other components and subsystems. Examples of such models include Timed Failure Propagation Graphs (TFPG), Finite Degradation Models (FDM) and Propagation Graphs over Finite Degradation Structures (PGFDS).

In this paper, we generalize previous models for fault propagation. We define a general formalism, called Unifying Propagation Graphs (UPG) which encompasses, and is strictly more expressive of, previous notations, and we formally define its syntax and semantics. We discuss the integration of UPG into the xSAP safety analysis platform, and the generalization of existing routines for fault propagation analysis to the complete fragment of UPG. Finally, as a first contribution, we extend the existing engine for computation of minimal cut sets of PGFDS to support interval timings, and we experimentally evaluate its performance.

Keywords: Failure Propagation Analysis · TFPG · FDM · PGFDS ·

1 Introduction

The complexity of the functions carried out by modern engineering systems, in many domains such as avionics, railways and automotive, is continuously

* This study was carried out within the Interconnected Nord-Est Innovation Ecosystem (iNEST) and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) - MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.5 - D.D. 1058 23/06/2022, ECS00000043). This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

We acknowledge the support of the MUR PNRR project FAIR - Future AI Research (PE00000013), under the NRRP MUR program funded by the NextGenerationEU.

increasing, and demands a corresponding increase in the techniques to ensure the correctness and safety of their design. Complex systems must be able to detect and handle component faults, since their propagation may cause failures, i.e. conditions whereby larger parts of the system are no longer able to perform their intended function, possibly compromising system safety and creating risk of damage, harm to humans or the environment.

Domain-specific standards [1,2,3] describe structured techniques to assess the quality and robustness of system design. Typically, the design of complex systems follows a pattern whereby a system and its functions are hierarchically decomposed and allocated to subsystems and components. In particular, the goal of safety assessment is to assess and quantify the reliability and safety of a design architecture in presence of faults. In avionics, the process is initiated with a Functional Hazard Assessment (FHA) that aims to identify failure conditions and assess their severity and impact on system functions. Then, Preliminary System Safety Assessment (PSSA) consists in the systematic evaluation of a candidate system architecture, based on the results of FHA. PSSA links failure conditions and effects to appropriate safety requirements and allocates them to components. Finally, System Safety Assessment (SSA) validates the design by systematically demonstrating that the allocated safety requirements are met.

PSSA and SSA are supported by several safety assessment techniques. PSSA analyzes fault propagation paths. A formal approach to PSSA is based on abstract propagation models, e.g. graph models where nodes represent (failures of) system components or functions, and arcs represent propagation paths that specify how the degradation of one component may lead to the degraded or failed operation of another one. Recent models include formalisms such as Timed Failure Propagation Graphs (TFPG) [4,5,6], Finite Degradation Models (FDM) [7], Propagation Graphs over Finite Degradation Structures (PGFDS) [8,9] (compare Fig. 1). TFPG are a graph-based notation, whereby arcs may be decorated with propagation timings and system modes (enabling the propagation). Nodes without incoming edges are initial failures. Inner nodes extend Boolean conjunction and disjunction by considering system mode and timing constraints. FDM is an equational notation based on variable flows, modeling the effects of component degradation, based on Finite Degradation Structures (FDS) [10], modeling degradation level beyond the pure Boolean case (e.g. $\oslash$ and $\|$ in Fig. 1). PGFDS is a hypergraph-based notation built on top of FDS and extending FDM by admitting cycle. PGFDS relaxes the notion of operators into relations between failure mode assignments.

SSA is based on classical techniques to identify links between faults and undesired (system-level) events, a.k.a. feared events or top-level events (TLE). Such techniques include Failure Mode and Effects Analysis (FMEA) and Fault Tree Analysis (FTA). The latter is carried out top-down, from the TLE to the basic faults, where former is done bottom-up. A notable problem for FTA is to identify all the minimal combinations of faults that are possible explanation for a feared event, a.k.a. Minimal Cut Sets (MCS).

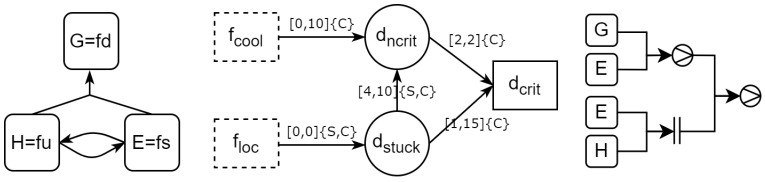

Fig. 1. Graphical representations: PGFDS (left), TFPG (center), FDM (right).

In this paper, we propose and design a novel and fully general formalism to model failure propagation, called Unifying Propagation Graphs (UPG). UPG provides a unifying view of failure propagation, in that it generalizes, and is more expressive of, existing notations such as TFPG, FDM, and PGFDS.

This paper gives the following contributions. First, we define UPG as an expressive and unifying formalism to model failure propagation. We formally define its syntax and semantics, and we show that it is strictly more expressive than TFPG, FDM and PGFDS.

Second, we design the integration of UPG into the xSAP [11,12], a state-of-the-art platform for model-based safety assessment, which provides library-based fault injection and automated model-based generation of safety artifacts such as FMEA tables and FTs. Moreover, it supports TFPGs analysis and synthesis, and TFPG validation w.r.t. to a behavioral (dynamical) model. We discuss the extension of xSAP to model UPG and the integration of the existing PGFDS engine of [8,9]. Together, the TFPG engine and the PGFDS engine enable solving problems such as TFPG analysis and synthesis, MCS computation, and validation w.r.t. a behavioral model. The integration is ongoing, our goal is to target the implementation of the existing routines for the complete fragment of UPG.

Third, as a first contribution, we solve a notable problem, namely we extend the existing engine for MCS computation of PGFDS to support interval timings. As in [8] we reduce the problem to symbolic search over a transition system, using efficient techniques for satisfiability modulo theories. We experimentally evaluate this extension on a set of benchmarks and show its effectiveness for the generation of MCS.

The rest of this paper is structured as follows. In Section 2 we present some preliminary notions. In Section 3 we present the syntax and semantics of UPG. In Section 4 we discuss how to embed UPG into xSAP as an extended version of TFPG. In Section 5 we discuss the extension of the [8] encoding with timings for MCS generation. In Section 6 we discuss the experimental evaluation. Finally, we draw some conclusions in Section 7.

Related Work Traditional formalisms for failure propagation include Failure Propagation Transformation Notation (FPTN) [13] and Hip-HOps [14]. They focus on the description of common fault types and patterns, and on the analy-

sis and synthesis of fault propagation through a system's architecture, however the specification of propagation relations is less expressive than in UPG. [1]

Timed Failure Propagation Graphs (TFPG) [4,5,6], constrain failure propagations using time bounds and system modes, however they do not consider levels of degradation, and they do not support cycles as in UPG. Moreover, UPG can constrain system modes in a more expressive way than TFPG.

Finite Degradation Models (FDM) [10] is an equational notation based on variable flows, modeling the effects of component degradation on top of Finite Degradation Structures (FDS) [10], which model levels of degradation as different values of faults organized into a semi-lattice. UPG generalize FDM by considering non-deterministic propagations, cycles and timings.

Propagation Graphs over FDS (PGFDS) [8,9] models failure propagation using a generic *canFail* relationships, similarly to UPG, and are more expressive than FDM, since they enable non-deterministic propagations and cycles. UPG extend PGFDS by supporting timings and system modes, in a more expressive way than TFPG.

UPG notation also includes as a sub-case artifacts produced by FTA, i.e. Fault Trees, consisting in propagation trees rooted at a given TLE, and extensions such as Component Fault Trees (CFT) which admit multiple roots.

Finally, [15] presents a formalism for failure propagation which enables modeling sets of failure modes using a domain specific language. It is less expressive than FDM, in that sets of failure modes cannot be related by degradation order.

2 Preliminaries

An interval $\mathcal{I}$ is a subset of real numbers contained between two points. As in previous works on TFPG, in this paper, an interval is of the form (1) $[d_1, d_2]$ (closed interval), and (2) $[d_1, +\infty]$ s.t. $d_1, d_2 \in \mathbb{R}_{\geq 0}$ and $d_1 \leq d_2$. Note that in xSAP, TFPG can have *closed infinity interval* $[d_1, +\infty)$ and *open infinity interval* $[d_1, +\infty]$ but due to our focus on finite traces, we do not distinguish between them and simply use $[d_1, +\infty]$. We denote $INTV$ as the set of all such intervals. We define $min([d_1, d_2]) = d_1$ and $max([d_1, d_2]) = d_2$.

A finite degradation structure $(FM, \leq, \bot)$ is a finite partial order with domain FM and bottom $\bot \in FM$. Every element is called a *failure mode*.

TFPG A TFPG [16,4,5,6], is a structure $G = \langle F, D, E, M, ET, EM, DC \rangle$ where (i) F is a non-empty finite set of initial components; (ii) D is a non-empty finite set of discrepancies s.t. $D \cap F = \emptyset$; (iii)$E \subseteq V \times D$ is a non-empty set of edges between the set of nodes $V = F \cup D$ and D; (iv) M is a non-empty set of system modes; (v) $ET : E \to INTV$ associates every edge in E with an interval; (vi) $EM : E \to 2^M$ associates every edge in E with some $M' \subseteq M$; (vii) $DC : D \to \{AND, OR\}$ defines a type for discrepancies. A node in F can become *active* at $t = 0$ by itself. An edge e is *active* at time t if and only if the

[1] An exception is given by propagation patterns such as $c \to d \vee e$, however note that for MCS computation this would generate the same results as $c \to d \wedge e$.

current (at ts) mode m satisfies $m \in EM(e)$. For an OR node w and an edge $e = (v, w) \in E$, once v and e become active at time t_v, w may become active at some time t_w, where $t_w - t_v \in ET(e)$, so long as e remains active from t_v to t_w (i.e. no mode switch into some $m' \notin EM(e)$). An AND node w may become active at t_w, when for all incoming edge (v, w), v may activate w at time t_w: for all (v, w), $t_w - t_v \geq min(ET((v, w)))$ and there is v s.t. $t_w - t_v \leq max(ET(v, w))$.

PGFDS Given a *finite degradation structure* $D = (FM, \leq, \bot)$, a *PGFDS* [8,9] over D is a tuple $S = (J, C, canFail)$, where C is a finite set of *system components*, $J \subseteq C$ the components that can fail initially, $canFail\colon C \times FM \to \mathcal{F}$ is a map from $C \times FM$ to *can-fail formulas* ϕ of atoms $t_1 \leq t_2$ or $t_1 = t_2$ (with terms t_1 and t_2 built over variables C and constants from FM) or the usual compound Boolean sentence $\phi_1 \vee \phi_2$, $\phi_1 \wedge \phi_2$, or $\neg\phi$. For the semantics, a (PGFDS) *state* is a map $C \to FM$ and a *propagation* is a sequence of states $s_0 \dots s_k$ s.t. $s_0(c) = fm$ implies $c \in J$; and $s_{i+1}(c) = fm$ implies either (i) $s_i(c) = fm$ or (ii) $s_i \models canFail(c, fm)$ (in which entailment for atoms are from the FDS D and entailment for Boolean formulas is the usual one). $\leq$ for FDS is generalized on states: $s \leq s'$ iff for all c it holds that $s(c) = fm$ and $s'(c) = fm'$ implies $fm \leq fm'$.

As we saw in the introduction, Fig. 1 shows alternative graphical representations used for differing formalisms. For PGFDS, it is possible to turn a can-fail formula, e.g. $canFail(G, fd)$, into DNF in which each disjunct is a source (a set of failure assignment to components) $\{H = fu, E = fs\}$, for one of the hyperedges leading to the head $G = fd$. TFPG representation is immediate from the definition. FDM [7] uses its flow variable definition where inner nodes are operators (where the domain is the failure modes of an FDS) generalizing Boolean operators[2]. Note that, as we will discuss in Sect. 4, our proposed formalism can be equivalently formulated as TFPG over FDS (at the price of introducing intermediate nodes for subformulas of $canFail$ formulas), therefore a TFPG-like graph representation is immediately possible.

3 Proposed formalism : Unifying Propagation Graph

3.1 Syntax and Semantics

Our syntactic formulation is based on PGFDS, where each combination of a component and a failure mode is assigned a formula. A formula may represent different states in a more compact manner. For the purpose of integrating TFPG, we consider temporal and system mode restrictions as separate formula decorations. As we will see later (Ex. 2), dealing with system modes and intervals separately adds another dimension of expressivity.

Definition 1 (Unifying Propagation Graph). *Given a* finite degradation structure $D = (FM, \leq, \bot)$ *and a finite set of* system modes M, *a* unifying

[2] It is noteworthy to mention here that FDM is the first formalism involving FDS, but since we focus more on PGFDS and TFPG, interested reader may refer to [17,7].

propagation graph *(UPG) over D is a tuple $S = (J, C, canFail)$, where C is a finite set of* system components*, $J \subseteq C$ the components that can fail initially,*

$$canFail\colon C \times FM \to \mathcal{F}$$

is a map from $C \times FM$ to a can fail formula $canFail(c, fm)$ over C as variables with domain FM, where $\mathcal{F}$ is of the form

$$\begin{aligned}\phi \quad := & \mid t_1 \leq t_2 \mid t_1 = t_2 \\ & \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \neg\phi \\ & \mid \mathcal{I}\phi \mid M'\phi\end{aligned}$$

with terms t_1 and t_2 built over variables C and constants from FM; $\mathcal{I}$ an interval; and $M' \subseteq M$. $\mathcal{I}\phi$ and $M'\phi$ are formulas with interval decoration *and* system mode *respectively. A* state *formula is a formula without interval decoration and a* path *formula is a formula with interval decoration.* [3]

The formalisms we unify use different notions of semantics such as activation time (TFPG), propagation graphs (PGFDS), formula evaluation (FDM). Here, we picked trace-based semantics and, to accommodate system mode and interval restriction, we bring the idea of super-dense semantics (in the context of hybrid system, see, e.g. [18]) where we combine index and timestamps. As we will see in Def. 2, we argue that this notion of semantics more closely describes the intuition of our intended operational semantics. Another alternative worth noting here is the transition system of timed automata from [19]. With this, it would be cumbersome to deal with subformulas that are *decorated* conjunction and disjunction with no explicit variables/states to be assigned.

A state is a tuple (i, μ, m, t) of an index in $\mathbb{N}$, a map $\mu : C \to FM$, a system mode $m \in M$ and a real number $t \geq 0$. A trace is a sequence of states $(0, \mu_0, m_0, t_0), \ldots, (k, \mu_k, m_k, t_k)$ with $t_0 = 0$ and $t_i \leq t_{i+1}$. A sub-trace of τ is a contiguous subsequence of τ. τ^l represents the prefix of τ of length l and $\tau[l] = (l, \mu_l, m_l, t_l)$ is the state at index l. For a UPG where system mode is not relevant we may use the notation (μ, t). Two consecutive states corresponds to either discrete steps or propagation steps. A *discrete step*[4] is between two states $(i, \mu_i, m_i, t_i).(i+1, \mu_{i+1}, m_{i+1}, t_{i+1})$ s.t. $t_i = t_{i+1}$ and either $m_i \neq m_{i+1}$ or $\mu_i \neq \mu_{i+1}$ (but not both) while a *propagation step* is when $t_i < t_{i+1}$ but $m_i = m_{i+1}$ and $\mu_i = \mu_{i+1}$. We assume *fault-persistence* for the failure modes, i.e. failure mode assignments persist: for any trace, if $\mu(c) = fm$ (noted $c \mapsto fm$ hereinafter) for $fm \neq \bot$ in some state, then it must hold in all later states.

Definition 2 (Semantics: state/path). *We define the semantics via state/path satisfaction for the can-fail formulas*[5]*.*

[3] Note that, $t_1 \leq t_2$ (or $t_1 \geq t_3$) can be translated into $=$, but can be useful when we later talk about monotonic UPG.

[4] Following that system mode switches take no delay.

[5] Note that state satisfaction is also path satisfaction.

State formula (without interval):

$$
\begin{array}{lll}
(j,\mu,m,t) \models t_1 \leq t_2 & \textit{iff} & \mu \models t_1 \leq t_2 \\
(j,\mu,m,t) \models t_1 = t_2 & \textit{iff} & \mu \models t_1 = t_2 \\
(j,\mu,m,t) \models \phi \wedge \psi & \textit{iff} & (j,\mu,m,t) \models \phi \textit{ and } (j,\mu,m,t) \models \psi \\
(j,\mu,m,t) \models \phi \vee \psi & \textit{iff} & (j,\mu,m,t) \models \phi \textit{ or } (j,\mu,m,t) \models \psi \\
(j,\mu,m,t) \models \neg\phi & \textit{iff} & (j,\mu,m,t) \not\models \phi \\
(j,\mu,m,t) \models M'\phi & \textit{iff} & (j,\mu,m,t) \models \phi \textit{ and } m \in M'
\end{array}
$$

Path formula (must fail)[6]*:*

$$
\begin{array}{ll}
\pi \models \phi \wedge \psi & \textit{iff } \pi \models \phi \textit{ and } \pi \models \psi \\
\pi \models \phi \vee \psi & \textit{iff } \pi \models \phi \textit{ or } \pi \models \psi \\
\pi \models M'\phi & \textit{iff there is } m \in M' \textit{ s.t.} \\
 & \quad \pi \models \phi \textit{ and } m \in M' \textit{ is the system mode} \\
 & \quad \textit{from the last state of } \pi \\
\pi \models \mathcal{I}\phi & \textit{iff there is } j_1, j_2, \textit{ s.t. } j_1 \leq j_2 \leq len(\pi) \textit{ and} \\
 & \quad t_{j_2} - t_{j_1} = max(\mathcal{I}) \textit{ and } \pi^{j'} \models \phi \textit{ for all } j_1 \leq j' \leq j_2
\end{array}
$$

path formula (can fail):

$$
\begin{array}{ll}
\pi \models_{cf} \mathcal{I}\phi & \textit{iff either } \pi \models \mathcal{I}\phi \textit{ or} \\
 & \quad \textit{there is } j_1, j_2, \textit{ s.t. } j_1 \leq j_2 = len(\pi) \textit{ and} \\
 & \quad t_{j_2} - t_{j_1} \in \mathcal{I} \textit{ and } \pi^{j'} \models_{cf} \phi \textit{ for all } j_1 \leq j' \leq j_2 \\
\pi \models_{cf} \phi & \textit{iff } \pi \models \phi \textit{ for other } \phi \textit{ cases.}
\end{array}
$$

System mode restricts only the last state, due to it being able to arbitrarily change. Interval decoration $\pi \models \mathcal{I}\phi$ has the 'less than' restriction $j_2 \leq len(\pi)$ to generalize fault persistence to formula. That is, once $\mathcal{I}\phi$ holds at some state, then it will hold at all later states. This makes it easier to relate to the original TFPG semantics for AND and OR discrepancies. Also, let us assume decorations bind the formulas stronger than the operators ($\{m_1\}c_1 = fm_1 \vee c_2 = fm_2$ is equivalent with $(\{m_1\}c_1 = fm_1) \vee (c_2 = fm_2)$)

Definition 3 (Semantics: UPG trace). $\pi = (0,\mu_0,m_0,t_0)\ldots(k,\mu_k,m_k,t_k)$ *is a trace of a UPG* $X = (J,C,canFail)$ *iff for all* $c \in C$ *and* $j \geq 0$

(i) (can fail) if $\mu_j(c) = fm \neq \perp$ *but* $\mu_{j-1}(c) = \perp$ *then either*

[6] For monotonic systems, negated state formula can be turned into a positive formula, but it may be used for a more concise formula (e.g. $\neg c = \perp$). We examined path formula and concluded that having negation means the introduction of a new kind of (modal) operator. For lack of space, we do not discuss this further, and assume that path formulas are without negation hereinafter.

– $c \in J$,
– $\pi^j \models_{cf} canFail(c, fm)$.

(ii) (must fail)

$$\pi_{j'} \models \bigvee_{fm \in FM \setminus \{\bot\}} canFail(c, fm) \rightarrow c = fm$$

for j' the maximum index $j \leq j'$ s.t. $t_{j'} = t_j$

In the rest of this paper, we consider traces where there is some failure in the first state $\pi[0]$, otherwise the prefix without failures can be removed. For $c \in J$ we additionally assume $j = 0$ in Def. 3.(i), which is w.l.o.g. for cut set enumeration purposes. Moreover, we rely on the notion of *FDS monotonicity* from PGFDS, i.e. $(i, \mu_i, m_i, t_i) \leq (i, \mu_i', m_i, t_i)$ iff $\mu \leq \mu_i'$, and in some examples we use an FDS called W2F $(\{\bot, fd, fu, fs\}, \leq, \bot)$ (from [20]) s.t. $\bot < fs$, $\bot < fd$, $fd < fu$ where fs, fd, fu maybe resp. called failed-safe, fail-detected, and failed-undetected. Finally, we say that a UPG is *cyclic* if there are two components and failure modes c, c' and fm, fm' s.t. $c = fm$ occurs (also transitively) in $canFail(c', fm')$ and vice versa.

Example 1 (Initial state, propagation step, and FDS-Monotonicity). With $J = \{d_1, d_2\}$, we define a UPG with the following can fail formulas and one if its traces.

$$\begin{aligned} canFail(c_1, fu) &= [5,5]\{m_2\}d_2 \geq fs \\ canFail(c_2, fu) &= [5,5]\{m_1\}d_1 \geq fd \wedge \\ &\qquad c_1 \geq fd \end{aligned}$$

$$\begin{aligned} \pi = &(0, \{d_1 \mapsto fd, d_2 \mapsto fs\}, m_2, 0) \\ &(1, \{d_1 \mapsto fd, d_2 \mapsto fs\}, m_2, 5) \\ &(2, \{d_1 \mapsto fd, d_2 \mapsto fs, c_1 \mapsto fu\}, m_2, 5) \\ &(3, \{d_1 \mapsto fd, d_2 \mapsto fs, c_1 \mapsto fu\}, m_1, 5) \\ &(4, \{d_1 \mapsto fd, d_2 \mapsto fs, c_1 \mapsto fu\}, m_1, 10) \\ &(5, \{d_1 \mapsto fd, d_2 \mapsto fs, c_1 \mapsto fu, c_2 \mapsto fu\}, m_1, 10) \end{aligned}$$

This example illustrates some properties of the trace definition. First, the initial fail $d_1 \mapsto fd$ happens in $\pi[0]$ but started propagating in $\pi[3]$ when the system mode finally became m_1. Second, the super dense semantics divides the trace into three parts: $\pi[0].\pi[1]$ and $\pi[3].\pi[4]$ are propagation steps while the others are discrete steps. Last, one can see that the UPG is FDS-monotonic due to the use of $\geq$: once $c_1 \mapsto fu$ appears in some state (index 2), this state will also satisfy $c_1 \geq fd$ in $canFail(c_2, fu)$.

Interval If $\phi = M'\phi$ then M' restricts the system mode only during propagation and can change afterwards. Moreover, during the propagation, e.g. for $[1,2]c = f$, failure propagation from a state with $c \mapsto f$ from time 0 can only be completed at the earliest time 1 and have definitely completed at time 2. Mode switching within the allowed system mode decoration e.g. $[1,2]\{m_1, m_2\}\phi'$ is possible. The state of index j_1 (compare Def. 2) represents the first state ϕ becomes true, j'

during a propagation, j_2 the first time $\mathcal{I}\phi$ becomes true, and $t > j_2$ the states where faults are persistent but there is no more restriction on the system mode. Notice that for $\mathcal{I} = \emptyset$, $\mathcal{I}\phi$ is simply **false** by definition ($\pi \models_{cf} \mathcal{I}\phi$).

Example 2 (Basic Example: TFPG with formulas). The following are various cases of decoration:

$$canFail^{(1)}(c', fm') = [1,2]\{m_1\}c = fm$$
$$canFail^{(2)}(c', fm') = [1,2][0,0]\{m_1\}c = fm$$
$$canFail^{(3)}(c', fm') = \{m_1\}[1,2]c = fm$$
$$canFail^{(4)}(c', fm') = [1,2](\{m_1\}c_1 = fm_1) \vee (\{m_2\}c_2 = fm_2))$$
$$canFail^{(5)}(c', fm') = [1,1](\{m_1\}[1,1](\{m_2\}[1,1]\{m_1\}c_1 = fm_1))$$

Here, (1) is the usual edge in TFPG, (2) shows an instantaneous propagation in mode m_1, followed by another propagation in any mode, (3) is a propagation ending in system mode m_1, (4) enforces a similar propagation time for alternative TFPG edges, and (5) shows a sequence of system mode changes where the failure mode mapping in a state does not change.

Example 3 (Max delay). Given $canFail(d, g) = [0,5]\{m\}c = f$, the following are both accepted traces.

$$\begin{aligned}\pi =&(0, \{c \mapsto f\}, m, 0).(1, \{c \mapsto f\}, m, 5).(2, c \mapsto f, d \mapsto g, m, 5).\\ &(3, c \mapsto f, d \mapsto g, m', 5)\\ \pi' =&(0, \{c \mapsto f\}, m, 0).(1, \{c \mapsto f\}, m, 5).(2, c \mapsto f, m', 5)\end{aligned}$$

This example illustrates that at maximal delay, two cases can happen depending on which discrete step (failure mode assignment or system mode switch) happens first. In the first case, the propagation is finished then its system mode changes, while in the second one, mode changes before the propagation is completed.

Interval Decoration Sequence An intermediate formula may have a satisfying trace where in some consecutive states time passes, without any change in the failure/system mode mapping, but they can be summed/partitioned.

Lemma 1 (Interval Sum/Partition). *Given $M'\phi$, it holds that*

$$[d_1, d_2]M'[d_1', d_2']M'\phi \quad \textit{iff} \quad [d_1 + d_1', d_2 + d_2']M'\phi$$

Example 4 (Summing/partitioning of temporal decoration). For

$$\{m_3\}([1,1]c_1 = fm_1 \wedge [0.5, 0.5]\{m_2\}[0.5, 0.5]\{m_1\}c_2 = fm_2),$$

notice that the following is still a trace of $\{m_3\}[1,1]c_1 = fm_1$ despite there are states with timestamp 0.5 in between.

$$\begin{aligned}&(0, \{c_1 \mapsto fm_1, c_2 \mapsto fm_2\}, m_1, \quad 0).(1, \{c_1 \mapsto fm_1, c_2 \mapsto fm_2\}, m_1, 0.5)\\ &(2, \{c_1 \mapsto fm_1, c_2 \mapsto fm_2\}, m_2, 0.5).(3, \{c_1 \mapsto fm_1, c_2 \mapsto fm_2\}, m_2, \quad 1)\\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (4, \{c_1 \mapsto fm_1, c_2 \mapsto fm_2\}, m_3, \quad 1)\end{aligned}$$

System Mode Sequence For consecutive mode decoration, a satisfying trace must also obey the order of the modes in the decorations. This can more specifically expressed as the following lemma.

Lemma 2 (System Mode Sequence). *$M_2 M_1 \phi$ is equivalent to $M_2 \cap M_1 \phi$*

System modes in different conjuncts do not have to obey any ordering restriction, but (in conjunction with temporal restrictions) may influence *when* a future failure can happen. An example is as follows.

Example 5 (Permutation by system mode). For the following

$$\begin{aligned} canFail(e_2, g_2) &= [1,1]e_1 = g_1 & canFail(e_1, g_1) &= [1,1]\{m_2\}d_2 = f_2 \\ canFail(tle, f) &= e_2 = g_2 \wedge c = f & canFail(c, f) &= [1,1]\{m_1\}d_1 = f_1 \wedge \\ & & & \quad [1,1]\{m_2\}d_2 = f_2 \end{aligned}$$

with $J = \{d_1, d_2\}$, we have the following trace ($tle \mapsto \bot$ and $e_2 \mapsto \bot$ are omitted):

$(0, \{d_1 \mapsto f_1, d_2 \mapsto f_2, c \mapsto \bot, e_1 \mapsto \bot\}, m_1, 0)$
$(1, \{d_1 \mapsto f_1, d_2 \mapsto f_2, c \mapsto \bot, e_1 \mapsto \bot\}, m_1, 1).(2, \{d_1 \mapsto f_1, d_2 \mapsto f_2, c \mapsto \bot, e_1 \mapsto \bot\}, m_2, 1)$
$(3, \{d_1 \mapsto f_1, d_2 \mapsto f_2, c \mapsto \bot, e_1 \mapsto \bot\}, m_2, 2).(4, \{d_1 \mapsto f_1, d_2 \mapsto f_2, c \mapsto f, e_1 \mapsto \bot\}, m_2, 2)$
$(5, \{d_1 \mapsto f_1, d_2 \mapsto f_2, c \mapsto f, e_1 \mapsto g_1\}, m_2, 2)$

Here, $e_1 \mapsto g_1$ happens at timestamp 2. We can continue extending the trace, to reach TLE at timestamp 3. However, if m_2 holds between timestamp 0 and 1 and then m_1 afterwards, $e_1 \mapsto g_1$ can then happen earlier at timestamp 1 and thus also the TLE can be reached at timestamp 2.

UPG as a unifying formalism The following shows how UPG relates to PGFDS and TFPG. Note that, if we also include other existing notions, UPG actually unifies other safety analysis formalisms such as FDM, static fault trees, and cyclic propagation graphs. We state the following relations (leaving the proofs to the reader, for space reasons) while the others can be checked in the literature.

Lemma 3 (From restrictive UPG to PGFDS and TFPG). *It holds that,*

(i) there is a translation from a UPG X without interval decoration into a PGFDS X' s.t. a trace of X can be projected to a trace of X'; and
(ii) there is a translation from a negation-free Boolean UPG X into a TFPG X' s.t. for a trace of X, its activation time for every component corresponds to a possible sequence of activation times for F of X'.

4 Implementation: UPG in xSAP

For the practical contribution, we would like xSAP to support UPG. To this aim, we extend TFPG with FDS, and show how to express UPG on top of this. A

practical consideration for this is that xSAP already supports TFPG and that from theoretical side, by Lemma 3, Boolean UPG shares a close relationship with TFPG. So, we need to accommodate FDS into TFPG. First, we simply expand the definition of TFPG nodes to be some assignment of a failure mode to a component. So the set of nodes V become $F \times FM \uplus D \times FM$. Second, conjunctions and disjunctions in UPG amount to discrepancies with an explicit variable in D that will be assigned to true or false. We will now present its formalization.

TFPG over FDS as a variant of UPG A TFPG $G = \langle F, D, E, M, ET, EM, DC \rangle$ over FDS $S = \langle FM_S, \leq_S, \bot_S \rangle$ is a structure where[7]: (i) F is a non-empty finite set of initial components; (ii) D is a non-empty finite set of discrepancies s.t. $D \cap F = \emptyset$; (iii)$E \subseteq V \times (D \times FM)$ is a non-empty set of edges between the set of nodes $V = (F \times FM) \cup (D \times FM)$. For readability, we may write an edge as $\langle c, f \rangle \rightarrow_E \langle c', fm' \rangle$; (iv) M is a non-empty set of system modes (v) $ET : E \rightarrow INTV$ is a map that associates every edge in E with an interval; (vi) $EM : E \rightarrow 2^M$ is a map associating edges in E with a set of modes in M; (vii) $DC : D \rightarrow \{AND, OR\}$ is a map defining the discrepancy type.

For the semantics, we add the restriction that, for each component/discrepancy $d \in F \uplus D$, node (d, fm) can become active only for one $fm \in FM$.

We can translate a UPG into a TFPG over FDS by giving a unique label to all occurrences of subformulas in canFail and make them a unique discrepancy. For convenience we consider them without failure modes (and implicitly treat them as if they have Boolean failure modes (failing/not failing)). From this discrepancy and all of its subformulas, for intervals, if $\phi = \mathcal{I}\phi'$, then we make an edge from the discrepancy label of ϕ' into ϕ (the case for system mode is similar). Two syntactically similar subformulas, e.g. $canFail(c, f) = [0, 1]d = f$ and $canFail(c', f') = [0, 1]d = f$ need different discrepancies. This is because from state $(0, \{d \mapsto f\}, 0)$, we can have $(0, \{d \mapsto f\}, 0).(1, \{d \mapsto f, c' \mapsto f'\}, 0.5)(2, \{d \mapsto f, c' \mapsto f', c \mapsto f\}, 1)$ which is not possible if the subformula $[0, 1]d = f$ is given a single label used for both $c = f$ and $c' = f'$.

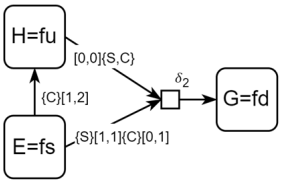

Fig. 2. UPG in TFPG-like graphical format

[7] The changes from the usual TFPG are in (iii) and its semantics that would have to accommodate FDS monotonicity.

With this formalization, UPG can then alternatively take advantage of the visual format of TFPG (compare example in Fig. 2). Rounded boxes are atomic formulas, a circle is a disjunction, a box is a conjunction, a label on an edge is a decoration on a formula represented by its source node. The edge labeling, the potentially unlabeled discrepancies, and the nodes being from $C \times FM$ are the primary difference w.r.t. the original TFPG.

xSAP-supported format We can easily extend the concrete syntax for TFPG in xSAP to handle FDS (and hence UPG). E.g., we extend the existing format by introducing the declaration of the list of failure modes of an FDS and their order, whereas edge definition declares components and failure modes. For backward compatibility, the declaration of a Boolean FDS may be omitted and in this case, node and edge declarations contain only component names.

5 SMT-based approach for Safety Analysis

As another further contribution on the practical side, we demonstrate cut set enumeration for UPG, by extending the SMT encoding from [8] with temporal constraints. A *cut set* for some decoration-free canfail formula TLE potentially with minimal-time *mintt* restriction and maximal-time restriction *maxtt* is an initial state (at timestamp 0) for which there is a trace to a state τ s.t. $\tau \models TLE$ at the earliest timestamp *mintt* and at the latest timestamp *maxtt*.

Definition 4 (SMT-Encoding of TFPG (With timing but no system mode)). *Given a TFPG X over an FDS S, its SMT encoding is defined as*

$$\varphi^t_{cs} = \varphi_{once} \wedge \varphi_{init} \wedge \varphi^t_{next} \wedge \varphi_{tle}$$

where $\varphi_{once} = \bigwedge_{c=fm,c=fm' \in D} \left(\neg F_{c,fm} \vee \neg F_{c,fm'}\right)$ *is for unique mode assignment;* $\varphi_{init} = \bigwedge_{(c,fm)\in F}(I_{c,fm} \rightarrow F_{c,fm})$ *is for relating initial and propagation failure;*

$$\varphi^t_{next} = \Big(\bigwedge_{(c,fm)\in D\cap F} \big(F_{c,fm} \rightarrow \big(I_{c,fm} \vee \mathit{canFail}^t_{smt}(c,fm)\big)\big)\Big) \wedge \Big(\bigwedge_{(c,fm)\in D\setminus F} \big(F_{c,fm} \rightarrow \big(\mathit{canFail}^t_{smt}(c,fm)\big)\big)\Big)$$

for propagation relation; and

$$\mathit{canFail}^t_{smt}(c,fm) = \begin{cases} \bigvee_{e=\langle d,fm'\rangle \rightarrow_E \langle c,fm\rangle} (F_{d,fm} \wedge o_d < o_c \wedge \quad //DC(c,fm) = OR \\ \quad t_d + ptime_{dc} = t_c \wedge \\ \quad min(ET(e)) \leq ptime_{dc} \leq max(ET(e))) \\ \\ \bigwedge_{e=\langle d,fm'\rangle \rightarrow_E \langle c,fm\rangle} (F_{d,fm} \wedge o_d < o_c) \wedge \quad //DC(c,fm) = AND \\ \quad t_d + ptime_{dc} = t_c \wedge min(ET(e)) \leq ptime_{dc} \wedge \\ \bigvee_{e=\langle d,fm'\rangle \rightarrow_E \langle c,fm\rangle} \quad ptime_{dc} \leq max(ET(e)) \end{cases}$$

The formula ϕ_{tle} can be seen as a canfail formula $canFail^t_{smt}(tle) \wedge mintt \leq t_{tle} \wedge t_{tle} \leq maxtt$ for a decoration-free formula $canFail(tle)$.

FDS monotonicity is from [8]. We replace $F_{d,fm}$ in $canFail^t_{smt}(c, fm)$ with $\bigvee_{fm \leq \hat{fm}} F_{d,\hat{fm}}$. For the implementation generating only FDS-minimal cut-sets, we remove ϕ_{once} and replace $I_{c,fm}$ in φ^t_{next} with $\bigwedge_{fm' \leq fm} I_{c,fm} \wedge \bigwedge_{fm' \not\leq fm} \neg I_{c,fm}$. Let us call the resulting encoding $\phi^{FM,t}_{cs}$. With this, we generate subset-minimal models w.r.t. I-vars $I_{c,fm}$ assigned to $true$. We get a cut set from such a model by collecting $c \mapsto fm$ s.t. fm is FDS-maximal among all fm' s.t. $I_{c,fm'} \mapsto true$ in the subset-minimal model (we call this extraction $modelToState^{FM}$ [8]). Its correctness builds upon the fact that the temporal restriction in the encoding already follows TFPG temporal restriction (see Sect. 2).

Theorem 1 (Cut set enumeration). *Given UPG X and a top level event TLE and its encoding $\phi^{FM,t}_{cs}$, we have*

$$cutset(X, TLE) \subseteq \{modelToState^{FM}(\mu) | \mu \models \phi^{FM,t}_{cs}\}$$

They coincide for FDS-monotone UPG.

6 Experimental Evaluation

We intend to have the full UPG formalism supported in xSAP. As an initial contribution, we extended the SMT-based encoding from [8] to accommodate temporal constraints, and we design an experiment to evaluate how temporal constraints affect performance.

Benchmarks and Experimental Setup We use cyclic benchmarks *ladder* and *radiator* from [21]. The benchmarks have two types of components: modules and voters. In order to focus and accentuate the effect of temporal restrictions, we make the following simplification. We only use the ones with one voter, because it was shown [21] that the cyclicity for two and three voters rendered the problems much more difficult and cycle handling is not our focus here. For FDS, we add failure modes using the FDS with order $\perp < L < H$ without randomization and in such a way that the number of cut sets do not grow too large. We assign J to only contain the modules and not voters. We randomize temporal decoration between $[0, 1]$ and $[1, 2]$ only for modules in the can-fail formulas. The number of components (representing the size of the benchmark instances) are multiple of three in $\{3, \ldots, 15\}$. For each instance, we try to generate cut sets that cause the top level event within 2 time-span up to 40 time-span (in multiples of two).

Experimental Results Fig. 6 shows the runtime and the number of cut sets w.r.t. the maximum TLE time. The main results are that (1) the runtime tends to be small for small and large maximum TLE time, (2) increases as the number of admissible cut sets (that can cause the given TLE within the time-bound) increase, (3) decreases when the admissible cut sets are the same as all cut sets (but may first have some increase when the max TLE time is tight). For

each of the number-of-component parameter, the number of all minimal cut sets for *ladder* and *radiator* benchmarks are 12, 24 36, 48, and 60. For *ladder* benchmark, they are found for the maximum TLE times of 2, 2, 4, 6, and 10, while, for *radiator* benchmark, they are found for maximum TLE times of 2, 4, 6, 10, and 10.

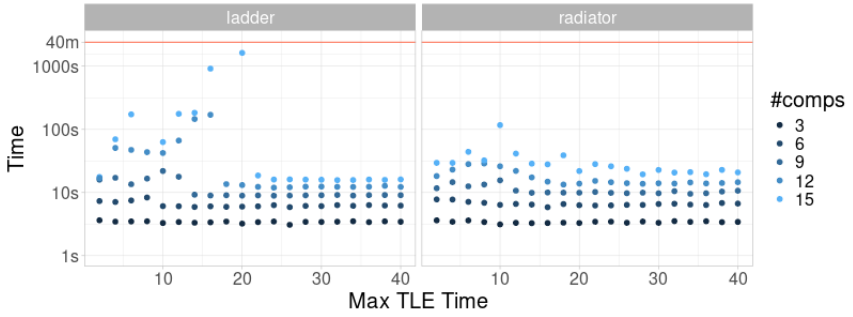

Fig. 3. Experimental Results

7 Conclusion and Future Work

In this work, we have proposed a new formalism for failure propagation called UPG, unifying and extending previous notations such as TFPG, FDM, PGFDS. We have designed the integration of UPG into the xSAP safety analysis platform, which will serve as an integrated platform for extended versions of the existing analysis and synthesis routines available for TFPG and PGFDS. As a first contribution in this direction, we have extended the encoding and the engine of [8,9] for MCS computation to support propagation timings, and demonstrated its performance on a set of benchmarks. We plan to make the integrated platform available in time for demonstration at the conference.

As part of our future work, we are considering further extensions of UPG. In particular, we want to investigate modeling aspects that are available in Dynamic Fault Trees, e.g. dynamic gates, spares and more complex forms of dependencies. Finally, we want to integrate probabilistic and observability aspects into UPG, with applications in monitoring and diagnosis.

References

1. IEC. Railway applications - Specification and demonstration of reliability, availability, maintainability and safety (RAMS), 2002.
2. ISO. Road vehicles - Functional safety, 2011.
3. SAE. ARP4761A Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment, 2022.

4. Marco Bozzano, Alessandro Cimatti, Marco Gario, and Andrea Micheli. SMT-based validation of timed failure propagation graphs. In *Proc. AAAI*, pages 3724–3730. AAAI Press, 2015.
5. B. Bittner, M. Bozzano, A. Cimatti, and G. Zampedri. Automated verification and tightening of failure propagation models. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI 2016)*, pages 907–913, 2016.
6. B. Bittner, M. Bozzano, and A. Cimatti. Automated Synthesis of Timed Failure Propagation Graphs. In *Proc. IJCAI*, pages 972–978, 2016.
7. Liu Yang and Antoine Rauzy. FDS-ML: A new modeling formalism for probabilistic risk and safety analyses. In *Proc. IMBSA*, volume 11842 of *Lecture Notes in Computer Science*, pages 78–92. Springer, 2019.
8. Marco Bozzano, Alessandro Cimatti, Anthony Fernandes Pires, Alberto Griggio, Martin Jonáš, and Greg Kimberly. Efficient SMT-Based Analysis of Failure Propagation. In *CAV*, volume 12760 of *LNCS*, pages 209–230. Springer, 2021.
9. Marco Bozzano, Alessandro Cimatti, Alberto Griggio, Martin Jonás, and Greg Kimberly. Analysis of cyclic fault propagation via ASP. In *Proc. LPNMR*, volume 13416 of *Lecture Notes in Computer Science*, pages 470–483. Springer, 2022.
10. Antoine Rauzy and Liu Yang. Finite degradation structures. *FLAP*, 6(6):1447–1474, 2019.
11. Benjamin Bittner, Marco Bozzano, Roberto Cavada, Alessandro Cimatti, Marco Gario, Alberto Griggio, Cristian Mattarei, Andrea Micheli, and Gianni Zampedri. The xSAP safety analysis platform. In *Proc. TACAS*, volume 9636 of *Lecture Notes in Computer Science*, pages 533–539. Springer, 2016.
12. M. Bozzano, A. Cimatti, M. Gario, D. Jones, and C. Mattarei. Model-based Safety Assessment of a Triple Modular Generator with xSAP. *Formal Aspects of Computing*, 33(2):251–295, 2021.
13. Peter Fenelon and John A. McDermid. An integrated tool set for software safety analysis. *J. Syst. Softw.*, 21(3):279–290, 1993.
14. Yiannis Papadopoulos and John A. McDermid. Hierarchically performed hazard origin and propagation studies. In *Proc. SAFECOMP*, volume 1698 of *Lecture Notes in Computer Science*, pages 139–152. Springer, 1999.
15. Kevin Delmas, Remi Delmas, and Claire Pagetti. SMT-based architecture modelling for safety assessment. In *12th IEEE International Symposium on Industrial Embedded Systems, SIES 2017*, pages 1–8. IEEE, 2017.
16. Sherif Abdelwahed, Gabor Karsai, Nagabhushan Mahadevan, and Stanley C. Ofsthun. Practical implementation of diagnosis systems using timed failure propagation graph models. *IEEE Trans. Instrum. Meas.*, 58(2):240–247, 2009.
17. Liu Yang, Antoine Rauzy, and Cecilia Haskins. Finite Degradation Structures: a Formal Framework to Support the Interface between MBSE and MBSA. In *Proc. ISSE*, pages 1–6, Rome, Italy, 2018.
18. Zohar Manna and Amir Pnueli. Verifying hybrid systems. In Robert L. Grossman, Anil Nerode, Anders P. Ravn, and Hans Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 4–35. Springer, 1992.
19. Johan Bengtsson and Wang Yi. Timed automata: Semantics, algorithms and tools. In *Lectures on Concurrency and Petri Nets, Advances in Petri Nets*, volume 3098 of *LNCS*, pages 87–124. Springer, 2003.
20. Antoine Rauzy. Mathematical foundations of minimal cutsets. *IEEE Trans. Reliab.*, 50(4):389–396, 2001.
21. Marco Bozzano, Alessandro Cimatti, Alberto Griggio, and Martin Jonáš. Efficient analysis of cyclic redundancy architectures via boolean fault propagation. In *TACAS*, volume 13244 of *LNCS*, pages 273–291. Springer, 2022.