Automated Synthesis of Timed Failure Propagation Graphs

Benjamin Bittner[1,2] * and **Marco Bozzano**[1] and **Alessandro Cimatti**[1]
[1]Fondazione Bruno Kessler, [2]University of Trento
Trento, Italy - lastname@fbk.eu

Abstract

Timed Failure Propagation Graphs (TFPGs) are used in the design of safety-critical systems as a way of modeling failure propagation, and to evaluate and implement diagnostic systems. TFPGs are mostly produced manually, from a given dynamic system of greater complexity. In this paper we present a technique to automate the construction of TFPGs. It takes as input a set of failure mode and discrepancy nodes and builds the graph on top of them, based on an exhaustive analysis of all system behaviors. The result is a TFPG that accurately represents the sequences of failures and their effects as they appear in the system model. The proposed approach has been implemented on top of state-of-the-art symbolic model-checking techniques, and thoroughly evaluated on a number of synthetic and industrial benchmarks.

1 Introduction

Timed Failure Propagation Graphs [Misra *et al.*, 1992; Ofsthun and Abdelwahed, 2007; Abdelwahed *et al.*, 2009; Bozzano *et al.*, 2015b] have been studied and used in practice in the design of safety-critical systems since the early 1990s, as a way to describe the occurrence of failures and of their direct and indirect effects, and to assess the corresponding consequences over time. TFPGs are a very rich formalism: they allow to model Boolean combinations of basic faults, intermediate events, and transitions across them, possibly dependent on the operational modes of the system, and to express constraints over the delays. TFPGs have been primarily used as a way to deploy diagnosis systems [Abdelwahed *et al.*, 2009]. Their importance is now increasingly recognized in the design of autonomous systems, in particular for the definition of Fault Detection, Isolation and Recovery (FDIR) procedures [Bittner *et al.*, 2014]. TFPGs have been the object of recent invitations to tender by the European Space Agency (ESA) [European Space Agency, 2011; 2012]. In fact, compared to other techniques such as Fault Tree Analysis (FTA) [Vesely *et al.*, 1981] and Failure Modes and Effects Analysis (FMEA) [McDermott *et al.*, 1996], failure propagation analysis has substantial advantages. While Fault Tree Analysis only explores subsets of propagation paths in response to specific *feared events*, failure propagation presents a more comprehensive and integrated picture. Moreover, it gathers more fine-grained and precise information that other techniques such as FMEA do not handle, such as timing information and the AND/OR correlation between propagation causes and effects. For these reasons, TFPGs are a natural candidate to address one of the key difficulties in the implementation of FDIR, i.e. the fact that the design is based on scattered and informal analyses, which lack a coherent view on failure propagation dynamics.

In this setting, TFPGs are often *manually* derived from a given dynamic system of greater complexity, as an abstract representation of its behavior under specific faulty conditions. In [Bittner *et al.*, 2016b] a comprehensive approach was introduced to validate manually built TFPGs against the behaviors of the corresponding system. While this gives assurance on the quality of the TFPG, it requires the laborious process of manually building the underlying graph based on system documentation. In this paper we propose a framework to *automatically* synthesize the TFPG, given a behavioral model of the system and a set of nodes. The resulting TFPG satisfies by construction a number of important properties as an abstraction of the system's behavior after the occurrence of faults. The approach has been fully implemented using state-of-the-art symbolic model-checking techniques for infinite-state transition systems [Bozzano *et al.*, 2015c], and thoroughly evaluated on a number of synthetic and industrial benchmarks derived in the context of collaborations with ESA and The Boeing Company. To our knowledge, this is the first fully general method that covers TFPG synthesis for infinite-state dynamic systems. The experimental evaluation demonstrates the practicality of the approach. Positive feedback is reported from its application in an industrial setting.

Related Work [Priesterjahn *et al.*, 2013] propose a TFPG synthesis algorithm for timed automata. Following a component topology it traverses the zone graph of the automata to discover discrepancies in output signals based on failures in input signals. No formal characterization of the synthesis result and no experimental evaluation of the approach are given. The present approach instead supports generic finite-

*The first author was partially supported by the European Space Agency NPI contract No. 4000111815/14/NL/FE.

and infinite-state transition systems, defines failure modes and discrepancies as generic properties of the system state and doesn't limit them to component signals, and produces TFPGs with well-defined formal characteristics. The backend of our implementation is based on off-the-shelf technology and doesn't use custom procedures for state-space traversal. [Dubey *et al.*, 2013] also propose a TFPG synthesis approach based on structural models, but don't consider system dynamics. [Strasser and Sheppard, 2011] use historical maintenance data to improve the accuracy of a TFPG that is given. In the present work instead the goal is to build, at design-time, a TFPG from only the nodes of the graph, based on an exhaustive analysis of all system behaviors.

2 Background

Symbolic Transition Systems are, following model-checking (e.g. [Cimatti and Griggio, 2012]), tuples $S = \langle X, I, T\rangle$, where X is a finite non-empty set of state variables, $I(X)$ and $T(X, X')$ are quantifier-free first-order formulae representing the initial states and the transition relation, X' being the next-state version of X. A *state* s of S is an assignment to the variables of X. We denote with s' the corresponding assignment to the variables in X'. The domain of $x \in X$ is written $\Delta(x)$. The set of all possible states (state space) of S, denoted $SP(S)$, may be either finite or infinite (if any $x \in X$ has an infinite domain). We write $\mu \models \phi$ to indicate that the variable assignment μ *satisfies* the formula ϕ, i.e. that ϕ evaluates to *true* if its variables are assigned the values specified by μ. A *trace* of S is an infinite sequence $\pi := s_0, s_1, \ldots$ of states such that $s_0 \models I(X)$ and for all integers $k \geq 0$ we have $(s_k, s_{k+1}) \models T(X, X')$. $\pi[k]$ denotes the state s_k of trace π, and s_k is short for $\pi[k]$ if the trace is clear from the context. A state s is reachable in S iff there exists a trace π such that $s = \pi[k]$ for some $k \in \mathbb{N}$. We assume a variable $\tau \in X$ exists with $\Delta(\tau) = \mathbb{R}_{\geq 0}$, associating each state with a time stamp, and that time advances monotonically (for any state s_i of any trace π, $\tau(s_i) \leq \tau(s_{i+1})$).

Metric Temporal Logic (MTL) is an extension of linear time logic (LTL), where the temporal operators are augmented with timing constraints (for an overview see [Ouaknine and Worrell, 2008]). It is interpreted over timed state sequences. Given a set of atomic propositions AP, including the symbols $\top$ (true) and $\bot$ (false), MTL formulae are defined as follows, with $p \in AP$: $\phi ::= p | \neg\phi | \phi_1 \wedge \phi_2 | \phi_1 \mathsf{U}^I \phi_2 | \phi_1 \mathsf{S}^I \phi_2$. The intervals I can be (partially) open or closed, $[a, b]$, (a, b), $(a, b]$, $[a, b)$, with $a, b \in \{\mathbb{R}_{\geq 0} \cup +\infty\}$ and $a \leq b$. I is omitted if $I = [0, +\infty)$, and the resulting simplified operators correspond to their standard LTL versions. Other operators can be defined as syntactic sugar: G, O. Given a transition system S with timed traces, a labeling function $L : SP(S) \mapsto 2^{AP}$, a trace π of S and a trace index k, we say that an MTL formula ϕ is satisfied at $\pi[k]$, written $\pi[k] \models \phi$, if the following holds: $\pi[k] \models p$ iff $p \in L(\pi[k])$; $\pi[k] \models \neg\phi$ iff not $\pi[k] \models \phi$; $\pi[k] \models \phi_1 \wedge \phi_2$ iff $\pi[k] \models \phi_1$ and $\pi[k] \models \phi_2$; $\pi[k] \models \phi_1 \mathsf{U}^I \phi_2$ iff $\exists i \geq k \cdot \tau_i - \tau_k \in I$ and $\pi[i] \models \phi_2$ and $\forall k \leq j < i \cdot \pi[j] \models \phi_1$; $\pi[k] \models \phi_1 \mathsf{S}^I \phi_2$ iff $\exists i \leq k \cdot \tau_k - \tau_i \in I$ and $\pi[i] \models \phi_2$ and $\forall i < j \leq k \cdot \pi[j] \models \phi_1$; $\pi[k] \models \mathsf{G}\phi$ iff $\pi[k] \models \neg(\top \mathsf{U} \neg\phi)$; $\pi[k] \models \mathsf{O}\phi$ iff $\pi[k] \models \top \mathsf{S} \phi$. We write $\pi \models \phi$ for $\pi[0] \models \phi$, and $S \models \phi$ to indicate that for all traces π of S we have $\pi \models \phi$.

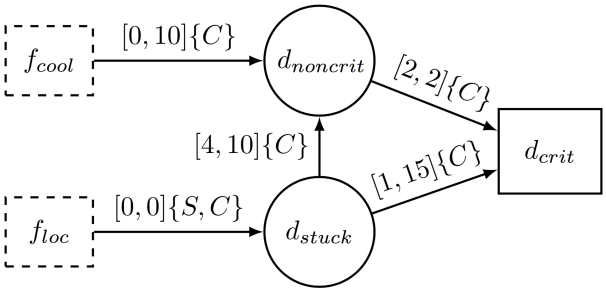

Figure 1: TFPG for the ForgeRobot example. Dotted boxes are failure mode nodes, solid boxes `AND` nodes, and circles `OR` nodes.

Timed Failure Propagation Graphs are directed graph models where nodes represent *failure modes* and *discrepancies* (failure effects). Edges model the temporal dependency between the nodes. They are labeled with lower and upper bounds on the propagation delay, and with labels indicating in what system modes the propagations are possible. TFPGs are defined as follows, based on [Abdelwahed *et al.*, 2009].

Definition 1 (TFPG)**.** *A* TFPG *is a structure* $G = \langle F, D, E, M, ET, EM, DC\rangle$*, where:* F *is a non-empty set of failure modes;* D *is a non-empty set of discrepancies;* $E \subseteq V \times V$ *is a non-empty set of edges connecting the set of nodes* $V = F \cup D$*;* M *is a non-empty set of system modes (we assume that at each time instant the system is in precisely one mode);* $ET : E \to I$ *is a map that associates every edge in* E *with a time interval* $[t_{min}, t_{max}] \in I$ *indicating the minimum and maximum propagation time on the edge (where* $I \in \mathbb{R}_{\geq 0} \times (\mathbb{R}_{\geq 0} \cup \{+\infty\})$ *and* $t_{min} \leq t_{max}$*);* $EM : E \to 2^M$ *is a map that associates to every* $e \in E$ *a set of modes in* M *(we assume* $EM(e) \neq \emptyset$*);* $DC : D \to \{\texttt{AND}, \texttt{OR}\}$ *is a map defining the discrepancy type. Failure modes never have incoming edges. All discrepancies must have at least one incoming edge and be reachable from a failure mode node. Circular paths (but not self-loops) are possible. We use* `OR`*(*G*) and* `AND`*(*G*) to indicate the set of OR nodes and AND nodes of a TFPG* G*, respectively,* $D(G)$ *to indicate all discrepancies, and* $F(G)$ *to indicate all failure modes.*

The running example *ForgeRobot* describes a robot working in a hypothetical industrial forge. The robot is either in standby in a safe area, or performs work in a critical area that has high heat levels. It moves around using its locomotion facilities. To prevent overheating in the critical area, a cooling system is used. The TFPG in Figure 1 shows possible failures of the robot and their effects over time. The locomotion drive of the robot can fail (f_{loc}), causing the robot to be stuck (d_{stuck}). The cooling system can fail (f_{cool}), decreasing the performance of heat protection. f_{cool} and d_{stuck} can both independently cause a non-critical overheating of the robot ($d_{noncrit}$) in mode C. In case both happen, they cause a critical overheating (d_{crit}). Two modes are used to differentiate

the operational context: S for safe area, and C for critical area. The time ranges on the propagation edges represent the different propagation speeds influenced by variable amount of workload and of heat in the critical area.

A TFPG node is activated when a failure propagation has reached it. An edge $e = (v, d)$ is active iff v is active and $m \in EM(e)$, where m is the current system mode. A failure propagates through $e = (v, d)$ only if e is active throughout the propagation, that is, up to the time d activates. For an OR node d and an edge $e = (v, d)$, once e becomes active at time t, the propagation will activate d at time t', where $tmin(e) \leq t' - t \leq tmax(e)$. Activation of an AND node d will occur at time t' if every edge $e = (v, d)$ has been activated at some time t, with $tmin(e) \leq t' - t$, and for at least one such edge e we also have $t' - t \leq tmax(e)$. If an edge is deactivated any time during the propagation, due to mode switching, the propagation stops. Links are assumed memoryless, thus failure propagations are independent of any (incomplete) previous propagation. A maximum propagation time of $t_{max} = +\infty$ indicates that a propagation can be delayed indefinitely, i.e. it can occur but not necessarily will.

Behavioral Validation of TFPGs In [Bittner *et al.*, 2016b] the notion of TFPG completeness is defined. Informally, it asks whether some failure propagation pattern exists on some system trace that is not captured by the TFPG. In the running example such a trace would be, e.g., one where first f_{loc} happens and then $d_{noncrit}$, without any other node activating, which violates the TFPG constraints. In a certain sense completeness asks whether the behaviors allowed by the TFPG are an over-approximation of the failure propagation behaviors that are possible in the system model.

The definition of completeness and its validation are based on *TFPG association maps* Γ, which w.l.o.g. are injective functions associating all elements of F, D, and M of a TFPG G to Boolean expressions γ over the state vector X of a system model S. For a node $v \in F \cup D$ we write γ_v to indicate the corresponding expression in Γ. Given an edge $e \in E$, the short form $\gamma_{\mu(e)}$ is used instead of $\bigvee_{m \in EM(e)} \gamma_m$.

These maps are specified by the user and are used to map system traces to traces of *TFPG transition systems*, which are defined as follows: $X = F \cup D \cup M \cup \tau$, with $\Delta(x) = \{\top, \bot\}$ for $x \in F \cup D \cup M$ and $\Delta(\tau) = \mathbb{R}_{\geq 0}$; $I(X) = \phi_{modes}(M) \wedge \tau = 0$; $T(X, X') = \phi_{modes}(M') \wedge \bigwedge_{x \in \{F \cup D\}}(x \rightarrow x') \wedge (\tau \leq \tau') \wedge ((\bigvee_{x \in \{F \cup D \cup M\}}(x \neq x')) \rightarrow (\tau = \tau'))$, where $\phi_{modes}(M) \equiv \bigwedge_{m \in M}(m \leftrightarrow \bigwedge_{n \in \{M \setminus m\}} \neg n)$. $TS(G)$ is used to indicate the TFPG transition system derived from the nodes $F \cup D$ and modes M of the TFPG G. The traces of $TS(G)$ that satisfy the TFPG constraints are represented by $\Pi(G)$. A trace $\pi \in TS(G)$ satisfies the Boolean constraints as encoded by the graph iff $\forall k \in \mathbb{N} \cdot \pi[k] \models \phi_{bool}(G)$, where $\phi_{bool}(G) := \bigwedge_{d \in \mathtt{OR}(\mathsf{G})}(\mathbf{d} \rightarrow \bigvee_{(v,d) \in E} \mathbf{v}) \wedge \bigwedge_{d \in \mathtt{AND}(\mathsf{G})}(\mathbf{d} \rightarrow \bigwedge_{(v,d) \in E} \mathbf{v})$ (bold-face letters are Boolean activation-state variables of the corresponding TFPG nodes).

Given a system model S, an association map Γ relating S to a given TFPG G, the verification of TFPG completeness is expressed as a set of model-checking problems on S, following the semantics of TFPGs. For OR nodes, the following proof obligations are formulated:

- $\psi_{\mathtt{OR} \cdot A}(d, \Gamma) := \mathsf{G}((\mathsf{O}\gamma_d) \rightarrow \mathsf{O}((\mathsf{O}\gamma_d) \wedge \bigvee_{e=(v,d) \in E}((\mathsf{O}\gamma_v) \wedge \gamma_{\mu(e)} \mathsf{S}^{\geq tmin(e)}(\mathsf{O}\gamma_v) \wedge \gamma_{\mu(e)})))$
- $\psi_{\mathtt{OR} \cdot B}(d, \Gamma) := \mathsf{G}\neg(\bigvee_{e=(v,d) \in E}((\mathsf{O}\gamma_v) \wedge \gamma_{\mu(e)} \wedge \neg(\mathsf{O}\gamma_d) \mathsf{S}^{> tmax(e)}((\mathsf{O}\gamma_v) \wedge \gamma_{\mu(e)} \wedge \neg(\mathsf{O}\gamma_d)))$

Intuitively, $\psi_{\mathtt{OR} \cdot A}(d, \Gamma)$ requires at least one edge $e = (v, d)$ to be active for at least $tmin(e)$ time units at the point where d activates, and $\psi_{\mathtt{OR} \cdot B}(d, \Gamma)$ requires that a propagation along some edge $e = (v, d)$ cannot be delayed for more than $tmax(e)$ time units. The corresponding proof obligations for AND nodes ($\psi_{\mathtt{AND} \cdot A}(d, \Gamma)$ and $\psi_{\mathtt{AND} \cdot B}(d, \Gamma)$) are symmetrical, having $\bigwedge_{e \in E}$ instead of $\bigvee_{e \in E}$. The proof obligation for TFPG completeness is $\Psi(G, \Gamma) := \bigwedge_{d \in \mathtt{OR}(\mathsf{G})}(\psi_{\mathtt{OR} \cdot A}(d, \Gamma) \wedge \psi_{\mathtt{OR} \cdot B}(d, \Gamma)) \wedge \bigwedge_{d \in \mathtt{AND}(\mathsf{G})}(\psi_{\mathtt{AND} \cdot A}(d, \Gamma) \wedge \psi_{\mathtt{AND} \cdot B}(d, \Gamma))$. Then, G is complete w.r.t. S *iff* $S \models \Psi(G, \Gamma)$.

3 Problem Description

This section describes in detail the problem the present work addresses. Specifically, we are interested in synthesizing the graph of a TFPG. The idea is to provide the synthesis procedure a system model, a list of nodes, and an association map defining them. The procedure then should return a TFPG that includes all input nodes, that is complete and whose graph accurately represents the precedence constraints among node activations in correspondence to the system behaviors. Synthesis of edge constraints is covered in [Bittner *et al.*, 2016b] and not included here.

Informally, given a discrepancy d, the precedence constraints of d describe which other nodes must activate before d itself can become active. In the running example, for instance, it means that $d_{noncrit}$ is always preceded by either f_{cool} or, alternatively, by f_{loc} and d_{stuck}. The formal definition of precedence constraints proposed here is based on the notion of *cut-sets* from Fault Tree Analysis [Vesely *et al.*, 1981], a classical technique for safety analysis. We report here the definition from [Bozzano *et al.*, 2015c], adapted to our setting.

Definition 2 (Cut-Set)**.** *Given are a transition system S and a set of events $EV = \langle e_1 \dots e_n \rangle$, where each event $e_i \in EV$ is defined by a Boolean formula interpreted over the state vector of S. Given an event $e \in EV$, a set $cs \subseteq EV \setminus e$ is a* cut-set *of e iff there exists a trace π of S for which $\exists k \in \mathbb{N}$ such that $\pi[k] \models e$ and $\forall e' \in EV \setminus e \cdot e' \in cs \Leftrightarrow \exists i \leq k \cdot \pi[i] \models e'$. A cut-set cs of e is* minimal *iff no proper subset of cs is a cut-set of e. We use $MCS(e, EV, S)$ as a short form to indicate the set of all minimal cut-sets of event e in S w.r.t. EV.*

In a TFPG transition system the events are the node activations. In a system model they are the Boolean events associated to the nodes by the association map. In the TFPG G of the running example, $\{f_{loc}, d_{stuck}\}$ is a minimal cut-set of $d_{noncrit}$, considering the traces in $\Pi(G)$. This means that, after the activations of f_{loc} and d_{stuck}, the node $d_{noncrit}$ can become active before any other node does, e.g., f_{cool} doesn't have to occur before $d_{noncrit}$. Note that satisfaction

of these conditions does not imply that a propagation always *will* reach a certain target discrepancy, as completion of a propagation can depend on dynamics within the model that are invisible from the point of view of the TFPG.

Based on the notion of minimal cut-sets, precedence constraints are defined as follows.

Definition 3 (Precedence Constraints)**.** *Given are a transition system S, a set of events EV, and some $e \in EV$. The* precedence constraints *of e are the set of all of its minimal cut-sets w.r.t. EV and S. The precedence constraints of e are satisfied on some trace and state $\pi[k]$ iff all events of one of e's minimal cut-sets have occurred at some state j, with $0 \leq j \leq k$.*

The precedence constraints of e describe which events of EV must happen on the traces of S before e itself will occur. We now define the property of *graph correctness* which the synthesized TFPG must have. We rely on Γ to map sets of elements to the respective domain, i.e., to map sets of nodes $cs \subseteq F \cup D$ to the equivalent set $\{\gamma_x | x \in cs\}$, and vice-versa.

Definition 4 (Graph Correctness)**.** *Given are a system model S, a TFPG G, and an association map Γ. Also, let $EV \subseteq D(G) \cup F(G)$ and let $\Sigma(G)$ be the set of traces of $TS(G)$ that satisfy the Boolean constraints of the graph. We say that the graph of G is correct w.r.t. S, Γ and EV iff, for every discrepancy $d \in D(G) \cap EV$, the precedence constraints of d w.r.t. EV and $\Sigma(G)$ are equivalent, based on the mapping defined by Γ, to the precedence constraints of the corresponding expression γ_d w.r.t. $\{\gamma_v | v \in EV\}$ and S.*

This property guarantees an accurate graphical representation of the event orders of failure propagations possible in the system. In the running example, ignoring d_{crit}, we could obtain a complete TFPG by simply connecting f_{loc} to d_{stuck} and $d_{noncrit}$, and f_{cool} to $d_{noncrit}$. The graph however would not be correct, since in S, assuming the absence of f_{cool}, $d_{noncrit}$ is always preceded by both f_{loc} and d_{stuck}, which is not required by that graph. This piece of information can be crucial for instance in diagnosis applications. Assume both f_{cool} and $d_{noncrit}$ are observable, that f_{cool} is not activated and that $d_{noncrit}$ is activated. From this, based on the correct graph, we can deduce the occurrence of d_{stuck}, which is not possible in the other TFPG.

We now formally define the TFPG synthesis problem.

Problem 1 (Graph Synthesis)**.** *Given are a system model S, a set of failure mode nodes F, a set of discrepancy nodes D, a set of modes M, and an association map Γ defining the nodes $x \in F \cup D$ w.r.t. S.* Graph Synthesis *consists in finding a TFPG G that satisfies the following properties: 1. $F = F(G)$ 2. $D \subseteq D(G)$ 3. G is complete w.r.t. S and Γ 4. the graph of G is correct w.r.t. S, Γ, and $EV = F \cup D$.*

Property 2 does not require D to be identical to $D(G)$, as it might be necessary to introduce additional nodes to model the precedence constraints, as described in Section 4.

4 TFPG Synthesis

A methodology to automatically synthesize and refine TFPGs is now introduced. First a TFPG is built according to Problem 1. The resulting TFPG is then simplified in a second step to make it more amenable to manual inspection.

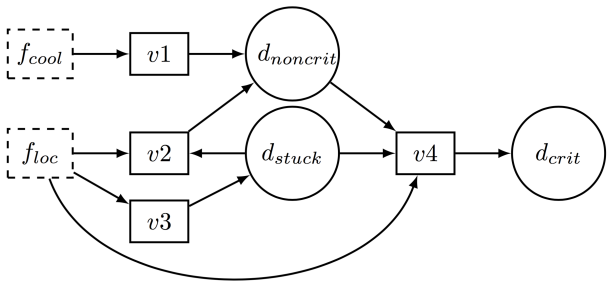

Figure 2: TFPG as produced by Algorithm 1 for running example, including user-defined and virtual nodes.

Graph Synthesis The synthesis algorithm makes use of *virtual discrepancies*, which are auxiliary TFPG nodes used to express the temporal relationships among the user-defined nodes when instantiating the graph.

Definition 5 (Virtual Discrepancy)**.** *Given a TFPG G, a system model S, and an association map Γ that is total w.r.t. $F(G)$ but partial w.r.t. $D(G)$, a* virtual discrepancy *is a node $d \in D(G)$ that is not in the domain of Γ. Instead of being associated, w.r.t. S, to a Boolean expression γ_d, such discrepancies are associated to temporal expressions over the traces of S, following the structure of G. For* OR *nodes the associated expression is $\gamma_d := \bigvee_{(v,d) \in E} \mathsf{O}\gamma_v$, and for* AND *nodes it is $\gamma_d := \bigwedge_{(v,d) \in E} \mathsf{O}\gamma_v$. It is assumed that no $v \in D(G)$, where $(v, d) \in E$, is itself a virtual discrepancy.*

Intuitively, w.r.t. system traces, a virtual discrepancy activates at the same instant as one or all of its predecessors in the graph, depending on the node type. Note that the proof obligations for TFPG completeness do not change, only the specific expressions associated to virtual discrepancies do.

Algorithm 1 TFPG-by-FTA

Inputs: system model S; set of failure modes F; set of discrepancies D; set of modes M; association map Γ.
Steps

```
1: instantiate each discrepancy d ∈ D as an OR node;
2: instantiate each failure mode f ∈ F as an FM node;
3: for all d ∈ D do
4:     for all mcs ∈ MCS(γ_d, {γ_d' | d' ∈ F ∪ D}, S) do
5:         instantiate a fresh virtual AND node v
6:         create unconstrained edge (v, d)
7:         for all γ_v' ∈ mcs do
8:             create unconstrained edge (v', v)
9:         end for
10:    end for
11: end for
```

The procedure to create a TFPG according to Problem Definition 1 is defined in Algorithm 1. It computes the minimal cut-sets of all discrepancies in terms of all other TFPG nodes and merges the results in a single graph[1]. Virtual AND

[1]It is assumed that no discrepancy can have a minimal cut-set that doesn't contain a failure mode event, i.e. that discrepancies can occur independently of failure mode events.

nodes are introduced (line 5) to represent the activation of all nodes of a minimal cut-set, which then enable the activation of the given target node. The edges are unconstrained in the sense that they are labeled with maximally permissive constraints, i.e. with $tmin(e)$ set to 0, $tmax(e)$ set to $+\infty$, and $EM(e)$ set to M. This labeling ensures TFPG completeness. The graph produced for the running example can be seen in Figure 2.

The synthesized TFPG is complete w.r.t. S and Γ, as shown in the following theorem.

Theorem 1. *A TFPG G built using the TFPG-by-FTA procedure, based on a system model S and an association map Γ, is complete w.r.t. S and Γ.*

Proof. In a TFPG where $\forall e \in E$ we have $tmin(e) = 0$, $tmax(e) = +\infty$, and $EM(e) = M$, the proof obligations can be simplified. $\psi_{\texttt{OR}\cdot B}$ and $\psi_{\texttt{AND}\cdot B}$ trivially hold for all discrepancies since no delay value can be greater than $+\infty$. $\psi_{\texttt{OR}\cdot A}$ can be simplified to $\mathsf{G}(\mathsf{O}\gamma_d \rightarrow \mathsf{O}(\mathsf{O}\gamma_d \wedge \bigvee_{(v,d)\in E}(\mathsf{O}\gamma_v)))$, and further to $\mathsf{G}(\mathsf{O}\gamma_d \rightarrow (\mathsf{O}\gamma_d \wedge \bigvee_{(v,d)\in E}(\mathsf{O}\gamma_v)))$. The $\mathsf{O}\gamma_d$ in the consequence can be dropped: $\mathsf{G}(\mathsf{O}\gamma_d \rightarrow \bigvee_{(v,d)\in E}(\mathsf{O}\gamma_v))$. Finally, since every AND node v is a virtual discrepancy, we obtain $\mathsf{G}(\mathsf{O}\gamma_d \rightarrow \bigvee_{(v,d)\in E}\bigwedge_{(v',v)\in E}(\mathsf{O}\gamma_{v'}))$, which is ensured by construction of the graph from the minimal cut-sets of γ_d. $\psi_{\texttt{AND}\cdot A}(d,\Gamma)$ can symmetrically be reduced to $\mathsf{G}(\mathsf{O}\gamma_d \rightarrow \bigwedge_{(v,d)\in E}(\mathsf{O}\gamma_v))$ and rewritten as $\mathsf{G}(\bigwedge_{(v,d)\in E}(\mathsf{O}\gamma_v) \rightarrow \bigwedge_{(v,d)\in E}(\mathsf{O}\gamma_v))$, which trivially holds. □

In addition to completeness, the graph of the synthesized TFPG also satisfies the correctness property.

Theorem 2. *Given are a system model S, an association map Γ, and a TFPG G as produced by Algorithm 1 for failure mode nodes F and discrepancies D. As per Definition 4, the graph of G is correct w.r.t. S, Γ, and $EV = F \cup D$.*

Proof. For every discrepancy $d \in D$ the following holds. The set of all sets of nodes $cs \subset EV$ connected to d via some respective virtual AND node v (i.e., where $(v,d) \in E$, and $v' \in cs$ iff $(v',v) \in E$) represents by construction, when mapped to S, the set of all minimal cut-sets of γ_d. It also represents all minimal cut-sets of d w.r.t. $\Sigma(G)$. Otherwise, one such set of nodes cs would exist that is not a minimal cut-set for d in $\Sigma(G)$. If cs is a cut-set, it is automatically minimal by the semantics of the nodes. If it is not a cut-set, then for at least one $d' \in D \cap (cs \cup d)$ there are not sufficient nodes in $(cs \cup d)$ such that one virtual AND node v, with $(v,d') \in E$, can be activated. This however is not possible, since from the FTA step we know that $\{\gamma_v | v \in cs \cup d\}$ contains a minimal cut-set for every $\gamma_{d'}$, with $d' \in D \cap (cs \cup d)$. □

Graph Simplification Even though a TFPG synthesized by the procedure above satisfies certain relevant properties, its graph structure might be too verbose for some applications. For instance, manual inspection by a safety engineer is often impractical with the full graph and requires a simpler version that still maintains completeness and correctness. In Figure 2, for instance, the edge $(f_{loc}, v2)$ is redundant, and all virtual AND nodes except $v4$ are also not essential to encode the precedence constraints among the user-defined nodes.

For the purpose of TFPG simplification we introduce the following theorem, based on which a procedure to remove edges while preserving completeness and correctness can be derived. It uses the formula $\phi_{prec}(G) := \bigwedge_{d\in D}(\mathbf{d} \rightarrow \bigvee_{(v,d)\in E(G)}\bigwedge_{(v',v)\in E(G)}\mathbf{v'})$, which encodes the Boolean constraints of G, similarly to $\phi_{bool}(G)$, but factors out the virtual AND nodes, as the correctness property only regards the user-defined nodes. We write $\overline{cs}$ for the valuations of $\phi_{prec}(G)$ and $\phi_{bool}(G)$ that assign $\top$ to any $v \in cs$, and $\bot$ to all other variables.

Theorem 3. *Given is a TFPG G as produced by Algorithm 1 for failure mode nodes F and discrepancies D, map Γ and system model S. Also, given is a second TFPG G' that has the same nodes as G, the same edges towards OR nodes, but a subset of the edges towards the AND nodes. Then, G' is complete w.r.t. Γ and S; also, if $\phi_{prec}(G) \equiv \phi_{prec}(G')$, the graph of G' is correct w.r.t. Γ, S, and $EV = F \cup D$.*

Proof. Due to maximally permissive edge constraints and the expressions associated to virtual discrepancies, we can ignore the proof obligations $\psi_{\texttt{OR}\cdot B}$, $\psi_{\texttt{AND}\cdot A}$, and $\psi_{\texttt{AND}\cdot B}$, as shown in the proof of Theorem 1. Removing edges towards virtual AND nodes effectively weakens the proof obligations $\psi_{\texttt{OR}\cdot A}$, thus completeness is preserved.

Then, for both G and G' we have that $cs \subset F \cup D$ is a cut-set of $d \in D$ in $\Sigma(G)$ *iff* $\overline{cs \cup d} \models \phi_{prec}(G)$. ($\Rightarrow$) Assume cs is a cut-set of d in $\Sigma(G)$; then there exists $V \subseteq \texttt{AND}(G)$ such that $\overline{V \cup cs \cup d}$ is reachable on some $\pi \in \Sigma(G)$, and hence $\overline{cs \cup d} \models \phi_{prec}(G)$, since $\phi_{bool}(G) \rightarrow \phi_{prec}(G)$. ($\Leftarrow$) Assume $\overline{cs \cup d} \models \phi_{prec}(G)$, then $\overline{V \cup cs \cup d} \models \phi_{bool}(G)$, with $V \subseteq \texttt{AND}(G)$ and $v \in V$ iff $\overline{cs \cup d} \models \bigwedge_{(v',v)\in E(G)} v'$, of which there is at least one for every $d' \in D \cap (cs \cup d)$ as guaranteed by $\phi_{prec}(G)$, from which it follows that $\overline{V \cup cs \cup d}$ is reachable on some $\pi \in \Sigma(G)$ at $\pi[0]$, as it is a state of $TS(G)$ that satisfies $\phi_{bool}(G)$, thus qualifying as an initial state of a trace $\pi \in \Sigma(G)$, and hence cs is a cut-set of d in $\Sigma(G)$.

Finally, from $\phi_{prec}(G) \equiv \phi_{prec}(G')$ it follows that every $d \in D$ has the same cut-sets in $\Sigma(G)$ and $\Sigma(G')$ w.r.t. EV, and thus also the same minimal cut-sets. □

After removing edges of G, guided by $\phi_{prec}(G)$, it might also be possible to eliminate some virtual AND nodes without affecting the completeness of G and the correctness of its graph. A virtual AND node v with a single incoming edge (v',v) and a single outgoing edge (v,d), e.g. $v1$ in Figure 2, can be removed by replacing the two edges involving v with a new edge (v',d). If two AND nodes exist that have identical incoming and outgoing edges, one of them can be simply dropped. An OR node d with a single incoming edge from a virtual AND node v can be redeclared as an AND node; then for every edge (v',v) a new edge (v',d) is introduced and v is dropped from G; in Figure 2 this applies to d_{crit}. By applying all simplification strategies to the TFPG in Figure 2 we obtain the one in Figure 1.

Edge Constraint Tightening The procedures of [Bittner *et al.*, 2016b] may be used to tighten the time bounds and mode constraints of the edges, in order to identify more precise constraints on propagation delays and contexts. Note that edge tightening can be performed only after simplification, as the latter ignores propagation delays and assumes maximally permissive edges. In future work we will investigate a closer integration between graph simplification and tightening, for applications that need a more fine-grained modeling of propagation delays.

5 Experimental Evaluation

This section describes the experimental evaluation of our implementation of the proposed algorithms. For the FTA part we use the techniques of [Bozzano *et al.*, 2015c] off-the-shelf, as implemented in the safety analysis toolkit xSAP [Bittner *et al.*, 2016a], which itself is based on nuXmv [Cavada *et al.*, 2014], a symbolic model checker for infinite-state transition systems. The FTA implementation is based on reachability analysis, which for infinite-state systems is in principle undecidable; termination thus depends on the engine and the dynamics of the system model. The simplification step was implemented based on the SMT solver MathSAT5 [Cimatti *et al.*, 2013]. It uses two copies of $\phi_{prec}(G)$, one parameterized on all edges towards AND nodes, and the other one not. The lattice of edges that could be dropped is then searched guided by SAT calls and accelerated with UNSAT cores, until no more edges can be removed.

For the evaluation we use the following use cases. BATTERY SENSOR describes a timed generator-battery system that powers a hypothetical device. CASSINI are variants of the spacecraft propulsion system described in [Williams and Nayak, 1996], enriched with timed aspects. GUIDANCE is a model of the Space Shuttle engines contingency guidance requirements. FORGEROBOT are variations of the running example. POWERDIST describes the fault protection logic of a power distribution application. ACEX and AUTOGEN are hand-crafted models based on a state space derived from partially random graphs. We also ran our implementation on two industrial models of realistic complexity, WBS [Bozzano *et al.*, 2015a] describing an aircraft wheel-braking system, and X34 [Bajwa and Sweet, 2003] describing the propulsion system of an experimental spacecraft. In total we used 14 system models and 82 problem instances, with varying amounts of failure mode and discrepancy nodes[2].

All tests were run on a dedicated 64bit Linux computer with a 12 core CPU at 2.67 GHz and 100GB of RAM. Four cores were reserved for each test run to limit potential time skew. Each test was executed on a single core with a time limit of 900 seconds and a memory limit of 4GB. For every instance we ran graph synthesis and simplification. All benchmarks terminated well within the timeout. From Figure 4 an exponential behavior can be recognized, which is more or less pronounced depending on the use case. Interestingly, X34 turned out to have a maximum run-time of 9s, and WBS of 355s. The time needed for the simplification step is negligible w.r.t. the FTA step. Simplification removed on average 88% of the generated AND nodes and 67% of the edges.

[2]Benchmark files can be downloaded at es.fbk.eu/people/bittner/ijcai16.tar.gz

use case	max. bool	max. real	instances	max. FM	max. D
acex	35	0	24	2	25
autogen	99	0	11	8	20
battery	43	5	4	4	9
cassini	301	10	18	16	16
forgerobot	25	7	7	6	9
guidance	98	0	4	6	7
pdist	84	0	2	7	4
wbs	1179	0	6	12	11
x34	553	0	6	9	14

Figure 3: Use-case statistics with maximum number of Boolean and real variables among all use-case models, number of instances, maximum number of failure modes and discrepancies among all instances of each use case.

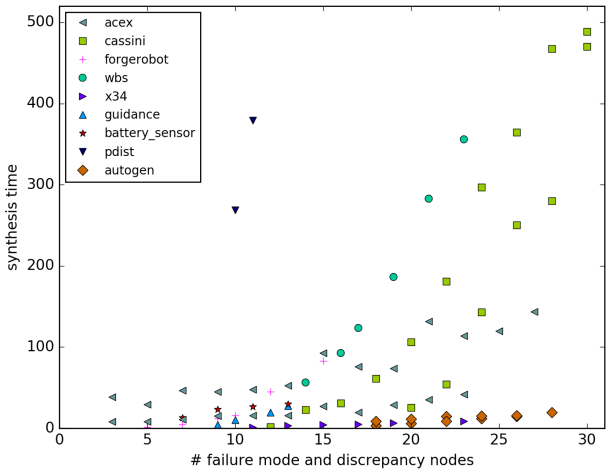

Figure 4: Number of nodes vs. synthesis time (in seconds).

These results show the feasibility of the approach, since, on models with a complexity comparable to industrial unit-to-subsystem-level problems, it terminates in reasonable time. For models with higher complexity we plan to investigate hierarchical/compositional synthesis approaches.

6 Conclusion

An approach to the automated synthesis of timed failure propagation graphs was introduced, based on symbolic model-checking. Given a system model and a set of nodes, the algorithm exhaustively analyzes the system behaviors and builds a TFPG that satisfies completeness and correctness properties. Evaluation on realistic models shows the feasibility of the approach. In future work we plan to extend the algorithm to filter out propagation links representing purely temporal correlations as opposed to causal ones, to investigate synthesis that explicitly considers diagnosability under the resulting TFPG, to use hierarchical approaches to deal with more complex instances, and to generalize the approach for synthesizing and representing sets of counterexamples for model-checking.

References

[Abdelwahed *et al.*, 2009] S. Abdelwahed, G. Karsai, N. Mahadevan, and S.C. Ofsthun. Practical Implementation of Diagnosis Systems Using Timed Failure Propagation Graph Models. *Instrumentation and Measurement, IEEE Transactions on*, 58(2):240–247, 2009.

[Bajwa and Sweet, 2003] Anupa Bajwa and Adam Sweet. The Livingstone Model of a Main Propulsion System. In *Proceedings of the IEEE Aerospace Conference*, pages 63–74, 2003.

[Bittner *et al.*, 2014] Benjamin Bittner, Marco Bozzano, Alessandro Cimatti, Regis De Ferluc, Marco Gario, Andrea Guiotto, and Yuri Yushtein. An Integrated Process for FDIR Design in Aerospace. In *Model-Based Safety and Assessment*, pages 82–95. Springer, 2014.

[Bittner *et al.*, 2016a] 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 *Tools and Algorithms for the Construction and Analysis of Systems*, pages 533–539. Springer, 2016.

[Bittner *et al.*, 2016b] Benjamin Bittner, Marco Bozzano, Alessandro Cimatti, and Gianni Zampedri. Automated verification and tightening of failure propagation models. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI 2016)*, 2016.

[Bozzano *et al.*, 2015a] M. Bozzano, A. Cimatti, A. Fernandes Pires, D. Jones, G. Kimberly, T. Petri, R. Robinson, and S. Tonetta. Formal Design and Safety Analysis of AIR6110 Wheel Brake System. In *Proc. CAV 2015*, pages 518–535, 2015.

[Bozzano *et al.*, 2015b] Marco Bozzano, Alessandro Cimatti, Marco Gario, and Andrea Micheli. SMT-based Validation of Timed Failure Propagation Graphs. In *Twenty-ninth AAAI Conference on Artificial Intelligence*, 2015.

[Bozzano *et al.*, 2015c] Marco Bozzano, Alessandro Cimatti, Alberto Griggio, and Cristian Mattarei. Efficient Anytime Techniques for Model-Based Safety Analysis. In *Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part I*, pages 603–621, 2015.

[Cavada *et al.*, 2014] Roberto Cavada, Alessandro Cimatti, Michele Dorigatti, Alberto Griggio, Alessandro Mariotti, Andrea Micheli, Sergio Mover, Marco Roveri, and Stefano Tonetta. The nuXmv Symbolic Model Checker. In *Computer Aided Verification*, pages 334–342. Springer, 2014.

[Cimatti and Griggio, 2012] Alessandro Cimatti and Alberto Griggio. Software Model Checking via IC3. In *Computer Aided Verification*, pages 277–293. Springer, 2012.

[Cimatti *et al.*, 2013] Alessandro Cimatti, Alberto Griggio, Bastiaan Joost Schaafsma, and Roberto Sebastiani. The MathSAT5 SMT Solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 93–107. Springer, 2013.

[Dubey *et al.*, 2013] Abhishek Dubey, Gabor Karsai, and Nagabhushan Mahadevan. Fault-Adaptivity in Hard Real-Time Component-Based Software Systems. In *Software engineering for self-adaptive systems II*, pages 294–323. Springer, 2013.

[European Space Agency, 2011] European Space Agency. Statement of Work: FDIR Development and Verification & Validation Process, 2011. Appendix to ESTEC ITT AO/1-6992/11/NL/JK.

[European Space Agency, 2012] European Space Agency. Statement of Work: Hardware-Software Dependability for Launchers, 2012. Appendix to ESTEC ITT AO/1-7263/12/NL/AK.

[McDermott *et al.*, 1996] Robin McDermott, Raymond J Mikulak, and Michael Beauregard. *The Basics of FMEA*. SteinerBooks, 1996.

[Misra *et al.*, 1992] Amit Misra, Janos Sztipanovits, Al Underbrink, Ray Carnes, and Byron Purves. Diagnosability of Dynamical Systems. In *Third International Workshop on Principles of Diagnosis*, 1992.

[Ofsthun and Abdelwahed, 2007] Stanley C Ofsthun and Sherif Abdelwahed. Practical Applications of Timed Failure Propagation Graphs for Vehicle Diagnosis. In *Autotestcon, 2007 IEEE*, pages 250–259. IEEE, 2007.

[Ouaknine and Worrell, 2008] Joël Ouaknine and James Worrell. Some Recent Results in Metric Temporal Logic. In *Formal Modeling and Analysis of Timed Systems*, pages 1–13. Springer, 2008.

[Priesterjahn *et al.*, 2013] Claudia Priesterjahn, Christian Heinzemann, and Wilhelm Schafer. From Timed Automata to Timed Failure Propagation Graphs. In *Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), 2013 IEEE 16th International Symposium on*, pages 1–8. IEEE, 2013.

[Strasser and Sheppard, 2011] Shane Strasser and John Sheppard. Diagnostic Alarm Sequence Maturation in Timed Failure Propagation Graphs. In *AUTOTESTCON, 2011 IEEE*, pages 158–165. IEEE, 2011.

[Vesely *et al.*, 1981] W Vesely, F Goldberg, N Roberts, and D Haasl. Fault Tree Handbook (NUREG-0492). *Washington, DC: Division of Systems and Reliability Research, Office of Nuclear Regulatory Research, US Nuclear Regulatory Commission*, 1981.

[Williams and Nayak, 1996] Brian C Williams and P Pandurang Nayak. A Model-Based Approach to Reactive Self-Configuring Systems. In *Proceedings of the National Conference on Artificial Intelligence*, pages 971–978, 1996.