Automated Verification and Tightening of Failure Propagation Models

Benjamin Bittner,[1,2]* **Marco Bozzano,**[1] **Alessandro Cimatti,**[1] **Gianni Zampedri,**[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 a very rich formalism: they allow to model Boolean combinations of faults and events, also dependent on the operational modes of the system and quantitative delays between them. TFPGs are often produced manually, from a given dynamic system of greater complexity, as abstract representations of the system behavior under specific faulty conditions.

In this paper we tackle two key difficulties in this process: first, how to make sure that no important behavior of the system is overlooked in the TFPG, and that no spurious, nonexistent behavior is introduced; second, how to devise the correct values for the delays between events. We propose a model checking approach to automatically validate the completeness and tightness of a TFPG for a given infinite-state dynamic system, and a procedure for the automated synthesis of the delay parameters. The proposed approach is evaluated on a number of synthetic and industrial benchmarks.

1 Introduction

Timed Failure Propagation Graphs (TFPGs) (Abdelwahed et al. 2009) 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, also 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. 2014b). TFPGs have been the object of recent invitations to tender by the European Space Agency (European Space Agency 2011; 2012). In fact, compared to other techniques such as FMECA (McDermott, Mikulak, and Beauregard 1996) and FTA (Bozzano, Cimatti, and Tapparo 2007), failure propagation analysis has substantial advantages. First, it allows for fine-grained and precise analyses that other techniques such as FMECA do not handle (e.g. timing information). 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. Failure propagation also addresses one of the key difficulties in the implementation of FDIR, i.e. the fact that the design is based on scattered and informal analysis, without quantitative reference that is essential for FDIR. 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 this paper we tackle two key difficulties in this process: first, how to make sure that no important behavior of the system is overlooked in the TFPG, and that no spurious, nonexistent behavior is introduced; second, how to devise the correct values for the delays between events. We propose a formal and automatic approach to validate the completeness and tightness of a TFPG for a given infinite-state dynamic system. The approach devises a sufficient set of proof obligations, in form of temporal properties, that must hold of the system for the TFPG to be complete. A model checker is used to check such proof obligations, and to produce diagnostic information if the TFPG does not accurately abstract the system.

A qualitative TFPG structure thus obtained is the input for an automated synthesis procedure, able to devise a tight set of parameter assignments. Specifically, the minimal and maximal values for the delay parameters are sought by means of runs of parametric model checking. The approach has been fully implemented, and evaluated on a number of synthetic and industrial benchmarks derived in the context of collaborations with the European Space Agency and The Boeing Company. The experimental evaluation demonstrates the practicality of the approach, and positive feedback is reported from the application in an industrial setting.

Related Work Next to considerable interest in TFPGs as tools for timed failure propagation modeling and as a basis for diagnosis implementations (Misra et al. 1992; Ofsthun and Abdelwahed 2007; Abdelwahed et al. 2009;

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

Bozzano et al. 2015b), there has also been some research in their automatic derivation and validation with respect to behavioral or structural system models.

A framework for TFPG synthesis for timed automata is presented in (Priesterjahn, Heinzemann, and Schafer 2013). The idea is to traverse the zone graph of the automata to discover discrepancies in output signals based on failures on input signals, as defined by a component model. Tight assignments to timings are also identified during the traversal. No experimental evaluation of the approach is given. The main differences to the present work are that we support generic finite and infinite transition systems as opposed to timed automata, that our approach directly accommodates both validation and tightening, and that, by mapping the problem to the model checking framework, state-of-the-art verification tools can be directly applied.

(Dubey, Karsai, and Mahadevan 2013) also proposes a TFPG synthesis approach based on component models. Only structural information is used to derive the TFPG and no behavioral model is considered. This approach is only possible when working with well-defined component models and is not suitable for the general case considered here.

In (Strasser and Sheppard 2011) an approach for TFPG maturation is presented. The goal is to use historical maintenance data to improve the model accuracy. The quality of the improvement depends on the quality of the data, and complete removal of possible errors cannot in general be guaranteed, e.g., it is not possible to remove behaviors that are allowed by the TFPG but are impossible in the application. Furthermore, the approach is only applicable when a system implementation is given and cannot be used at design time.

The rest of this paper is structured as follows. In Section 2 we present some background. In Section 3 we relate systems and TFPGs. In Section 4 we discuss the approach to abstraction validation. In Section 5 we describe our implementation of the validation algorithms. In Section 6 we analyze the experimental evaluation. In Section 7 we draw some conclusions, and discuss future work.

2 Background

2.1 Symbolic Transition Systems

A *symbolic transition system* is a tuple $S = \langle X, I, T\rangle$, where X is a finite non-empty set of state variables, $I(X)$ and $T(X, X')$ are 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 $\Sigma(S)$, 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 $(s_k, s_{k+1}) \models T(X, X')$. We denote with $\pi[k]$ the state s_k of trace π and with $x(s_k)$ the value of variable x in state s_k. We write s_k instead of $\pi[k]$ 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 $k \in \mathbb{Z}_{\geq 0}$. For enabling time-based analyses, we assume the presence of a variable $\tau \in X$ with $\Delta(\tau) = \mathbb{R}_{\geq 0}$, associating each state with a time stamp. We assume that time advances monotonically, i.e. given a trace π, $\tau(s_i) \leq \tau(s_{i+1})$ for any state s_i.

2.2 Metric Temporal Logic

Metric Temporal Logic (MTL) (Koymans 1990; Alur and Henzinger 1993; Ouaknine and Worrell 2008) is an extension of classical linear time logic LTL, where the temporal operators are augmented with timing constraints. It is interpreted over timed state sequences, e.g. the traces with time-stamps described in Section 2.1. 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: F, G, O.

Given a transition system S with timed traces, a labeling function $L : \Sigma(S) \mapsto 2^{AP}$, 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 $\forall i < j \leq k \cdot \pi[j] \models \phi_1$

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 $\pi \models \phi$.

2.3 Timed Failure Propagation Graphs

TFPGs were first introduced in (Misra et al. 1992) and (Misra 1994) to model the progression of failures in dynamic systems and to analyze diagnosability. A TFPG is a directed graph model where nodes represent *failure modes* (root events of failure propagations) and *discrepancies* (possible deviations from nominal behavior caused by failure modes). Edges model the temporal dependency between the nodes. They are labeled with propagation delays, and *system modes* indicating the system configurations in which propagation is enabled. TFPGs are formally defined as follows.

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 precisely in 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 edge in E a set of modes in M (we assume that $EM(e) \neq \emptyset$ for every edge $e \in E$);*
- *$DC : D \to \{\texttt{AND}, \texttt{OR}\}$ is a map defining the discrepancy type;*

Failure modes never have incoming edges, and 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.*

As a running example, we consider the Battery Powered Sensor System (BPSS) of (Bozzano et al. 2015b). The BPSS consists of a redundant pair of sensors powered by a redundant pair of batteries, charged by dedicated generators. A hypothetical device that depends on these sensor readings is assumed to fail if both sensors fail. A generator failure G_{Off} causes a permanent loss of power supply, and a sensor failure S_{Off} causes a permanent loss of its readings. In absence of power supply, a battery starts discharging; when depleted, the corresponding sensor stops working. Example discrepancies are the charge level of a battery below a threshold, or the absence of sensor readings. The BPSS has three modes: P, S_1, and S_2. In mode P, sensors are powered by their own batteries; in case of a battery failure, the BPSS may be reconfigured, e.g. in mode S_1 Battery 1 powers both sensors.

A TFPG for the BPSS is shown in Fig. 1. Boxes with dotted lines are failure modes, whereas discrepancies are either circles (`OR`) or boxes (`AND`) with solid lines. Edges are labeled with propagation intervals and modes ("$*$" indicates all modes). The non-determinism on the propagation time models uncertainty, e.g., the uncertainty on the propagation time from $B1_{Low}$ to $S1_{NO}$ indicates that the depletion of the battery will take longer in mode P rather than in S_1.

According to the semantics of TFPGs (Abdelwahed et al. 2009)) a node is active if the failure propagation reached it, and node activation is permanent. An edge $e = (v, w)$ is active if and only if v is active and $m \in EM(e)$, where m is the current system mode. A failure propagates through $e = (v, w)$ only if e is active throughout the propagation, that is, up to the time w activates. For an `OR` node w and an edge $e = (v, w)$, once the edge e becomes active at time t, the propagation will activate w at time t', where $e.tmin \leq t' - t \leq e.tmax$. For an `AND` w' instead, the activation period is the composition of the activation periods for each link $(v, w') \in E$ – if an edge is deactivated any time during the propagation, the propagation stops. Links are assumed memory-less, 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[1]. This is a useful over-approximation when the real t_{max} value is not available or when the propagation depends on some unconstrained input.

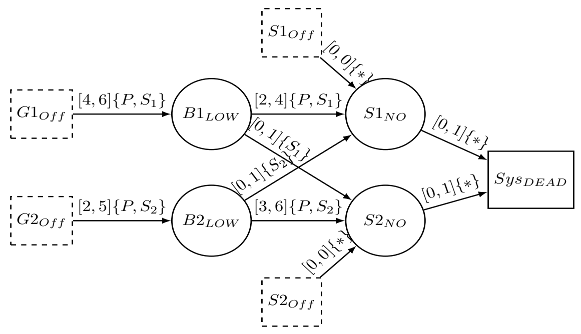

Figure 1: An Example TFPG for the BPSS.

3 Mapping Systems and TFPGs

3.1 TFPGs as Symbolic Transition Systems

To enable the mapping of system traces to TFPG traces, we define TFPGs as transition systems, whose paths describe failure propagations as timed sequences of failure mode and discrepancy occurrences and mode switches. We first define a TFPG transition system over TFPG nodes, modes, and time delays. Second, given a TFPG G defined over the same nodes and modes, we enforce the propagation constraints represented by the edges and the discrepancy classes.

We remark that in this model it is not always possible to precisely determine along which path a propagation occurred, e.g. in Fig. 1 it might not be clear along which path $S1_{NO}$ was activated, if the preconditions of all incoming edges are satisfied. However, it is always possible to check whether a trace satisfies the propagation constraints.

Definition 2 (TFPG Transition System)**.** *A set of failure mode variables F, a set of discrepancy variables D, and a set of system modes M are given. A* TFPG Transition System *is a tuple $S_{tfpg} = \langle X, I, T \rangle$ such that:*

- *$X = F \cup D \cup M \cup \{\tau\}$, with $\Delta(x) = \{\top, \bot\}$ for $x \in F \cup D \cup M$ and $\Delta(\tau) = \mathbb{R}_{\geq 0}$;*
- *$I(X) = \phi_{modes}(M) \wedge \tau = 0$;*
- *$T(X, X') = \phi_{modes}(M') \wedge \bigwedge_{x \in \{F \cup D\}} (x \to x') \wedge (\tau \leq \tau') \wedge ((\bigvee_{x \in \{F \cup D \cup M\}} (x \neq x')) \to (\tau = \tau'))$*

where $\phi_{modes}(M) \equiv \bigwedge_{m \in M} (m \leftrightarrow \bigwedge_{n \in \{M \setminus m\}} \neg n)$. A trace π of a TFPG transition system is called a TFPG trace*. We write $TS(G)$ to indicate the TFPG transition system derived from the nodes $F \cup D$ and modes M of the TFPG G.*

For $x \in \{F \cup D\}$, $x = \top$ indicates that the node is active in the current state, whereas for $x \in M$ it means the system is currently in mode x. Formula $\phi_{modes}(M)$ states that

[1]We are mainly interested in systems with a bounded operation time, hence we do not adopt the alternative interpretation prescribing that the propagation will eventually occur (Bozzano et al. 2015b)

the system is in precisely one mode at any time. The transition relation enforces that: TFPG nodes stay active once activated; time advances monotonically; time does not pass during discrete switches (by similarity with timed automata).

We define now under which conditions a trace of a TFPG transition system S_{tfpg} satisfies the constraints of a given TFPG. We use the notation $\mu(e) = \bigvee_{m \in EM(e)} m$ to indicate the system modes that are supported by an edge $e \in E$.

Definition 3 (OR-node satisfaction)**.** *Given a TFPG G, we say that a trace π of $TS(G)$ satisfies the constraints of an* OR *node $d \in D$ of G iff for any state $\pi[k]$:*

A. $(\pi[k] \models d) \rightarrow \exists j \leq k \cdot ((\pi[j] \models d) \wedge \exists e = (v,d) \in E\ \exists i \leq j \cdot ((\tau_j - \tau_i \geq tmin(e)) \wedge \forall i \leq l \leq j \cdot (\pi[l] \models (v \wedge \mu(e))))).$

B. $\forall e = (v,d) \in E \cdot (\neg \exists i \leq k \cdot ((\tau_k - \tau_i > tmax(e)) \wedge \forall i \leq j \leq k \cdot \pi[j] \models (v \wedge \mu(e) \wedge \neg d))).$

Condition A of Def. 3 states that if d is active in $\pi[k]$ then it must have been activated at some previous point $\pi[j]$ after some edge leading to d was active for at least the respective t_{min}, starting from $\pi[i]$, up to $\pi[j]$ where d became active. Condition B instead states that for no edge e the propagation can be delayed for more than the respective t_{max}.

Definition 4 (AND-node satisfaction)**.** *Given a TFPG G, we say that a trace π of $TS(G)$ satisfies the constraints of an* AND *node $d \in D$ of G iff for any state $\pi[k]$:*

A. $(\pi[k] \models d) \rightarrow \exists j \leq k \cdot ((\pi[j] \models d) \wedge \forall e = (v,d) \in E\ \exists i \leq j \cdot ((\tau_j - \tau_i \geq tmin(e)) \wedge \forall i \leq l \leq j \cdot (\pi[l] \models (v \wedge \mu(e))))).$

B. $\exists e = (v,d) \in E \cdot (\neg \exists i \leq k \cdot ((\tau_k - \tau_i > tmax(e)) \wedge \forall i \leq j \leq k \cdot \pi[j] \models (v \wedge \mu(e) \wedge \neg d))).$

Condition A of Def. 4 states that if d is active in $\pi[k]$ then it has been activated at some previous point $\pi[j]$ after all edges leading to d were active for at least the respective t_{min}, each starting from some individual $\pi[i]$, up to $\pi[j]$ where d became active. Condition B instead states that at least for one edge e the propagation must respect the respective t_{max} bound.

Definition 5 (TFPG satisfaction)**.** *Given a TFPG G, we say that a trace π of $TS(G)$* satisfies *G iff π satisfies, for all $d \in D$, the conditions of Def. 3 or Def. 4, depending on whether d is an* OR *node or* AND *node, respectively.*

We remark that TFPG satisfaction is based on local node activation constraints. According to Def. 5, a TFPG trace satisfies the TFPG constraints if all individual nodes are activated according to the respective local constraints, and if no node activation is delayed beyond the respective local upper bounds on propagation delay. Note that failure mode nodes $fm \in F$ need not be considered, since their activation is completely unconstrained w.r.t. the other TFPG elements.

This trace-based semantics for TFPGs closely follows the original semantics of TFPGs, as described earlier, and has been validated in various projects with the European Space Agency and The Boeing Company.

In the following sections we use $\Pi^*(G)$ to indicate all possible traces of $TS(G)$, and we use $\Pi(G) \subseteq \Pi^*(G)$ to indicate all traces of $TS(G)$ that satisfy G as per Def. 5.

3.2 System Abstraction via TFPGs

We are interested in abstracting systems using TFPGs, by associate system traces with TFPG traces. To this aim, we define TFPG elements of interest (failure modes, discrepancies and modes) in terms of system properties, as follows.

Definition 6 (TFPG Association Map)**.** *Given a set of failure mode variables F, a set of discrepancy variables D, a set of system modes M, a time-stamp variable τ, and a system model S_{sys}, a* TFPG Association Map *is a map Γ that associates every variable $x \in \{F \cup D \cup M\}$ with a Boolean predicate γ over the state variables X of S_{sys}, written $\gamma_x(X)$, or simply γ_x when the reference to X is clear from the context, and τ with a variable $x \in X$, representing the state time-stamps in the system, with $\Delta(x) = \mathbb{R}_{\geq 0}$. Given an edge $e \in E$, we use the short form $\gamma_{\mu(e)}$ for $\bigvee_{m \in EM(e)} \gamma_m$.*

For instance, in the running example the discrepancy $B1_{Low}$ may be defined by the expression $psu1.battery.charge < 40$.

When interpreting a system trace in terms of TFPG primitives we are interested in the points in the trace where failure modes occur, discrepancies become true or the system mode changes, and in the order and time delay between these events. Def. 7 defines a mapping from system traces to TFPG traces that guarantees that the order and timing of TFPG events is the same as in the system trace.

Definition 7 (Trace Abstraction)**.** *Given a system model $S_{sys} = \langle X, I, T \rangle$, a TFPG transition system $S_{tfpg} = \langle X_G, I_G, T_G \rangle$ with $X_G = \{F \cup D \cup M \cup \tau\}$, and a TFPG association map Γ defining the symbols in X_G based on predicates interpreted over X, we define the* trace abstraction *ζ_Γ of a system trace π producing an abstract TFPG trace π' of S_{tfpg}, written $\pi' = \zeta_\Gamma(\pi)$, as follows:*

- $\forall x \in \{F \cup D\} \forall k \in \mathbb{Z}_{\geq 0} \cdot ((\pi'[k] \models x) \leftrightarrow \exists i \leq k \cdot (\pi[i] \models \gamma_x))$
- $\forall x \in \{M \cup \tau\} \forall k \in \mathbb{Z}_{\geq 0} \cdot (x(\pi'[k]) = \gamma_x(\pi[k]))$

Given a system trace π, we assume that for every $x \in \{F \cup D \cup M\}$ and every point k it holds that $\gamma_x(\pi[k]) \neq \gamma_x(\pi[k+1]) \rightarrow \gamma_\tau(\pi[k]) = \gamma_\tau(\pi[k+1])$, i.e. time does not pass when the truth value of the predicate defining x changes. We also assume that the system is at each time instant in precisely one mode. These assumptions guarantee that the abstract traces of Def. 7 satisfy the constraints of Def. 2.

Completeness The notion of completeness of a TFPG G reflects the fact that all possible failure propagations in S are also modeled by G, in other words, whether the abstraction of every system trace satisfies the constraints of G.

Definition 8 (TFPG Completeness)**.** *Given a system model S, a TFPG G, and a TFPG association map Γ connecting the two, we say that G is* complete *w.r.t. S* iff *for every trace π of S, its abstraction $\zeta_\Gamma(\pi)$ satisfies G, i.e. $\zeta_\Gamma(\pi) \in \Pi(G)$.*

In the BPSS example we may be interested in whether, for instance, the propagation from $G1_{Off}$ to $B1_{Low}$ cannot happen in mode S_2, or take more than 6 time units.

Tightness Conversely to the notion of completeness, it is legitimate to ask whether each failure propagation modeled by a TFPG can actually take place in the system (correctness property). This question is misleading, since the TFPG is naturally an over-approximation, e.g. system modes are completely unconstrained in the TFPG, but in realistic cases this is not true in the system, and similarly for timing correlations. Instead we propose to study the property of tightness, i.e., whether certain parameters of the TFPG can be reduced without breaking its completeness. Specifically we are interested in tighten the propagation intervals and the modes.

Definition 9 (Edge Tightness)**.** *Given are a system model S, a TFPG G, an association map Γ, and an edge $e \in E$ of G. We say that $tmin(e)$ is tight* iff *there is no $r > tmin(e)$ such that G is complete w.r.t. S with $tmin(e) := r$ and all other parameters of G remaining the same. We say that $tmax(e)$ is tight* iff *there is no $r < tmax(e)$ such that G is complete w.r.t. S with $tmax(e) := r$ and all other parameters of G remaining the same. We say that $EM(e)$ is tight* iff *there exists no $m \in EM(e)$ such that G is complete w.r.t. S with $EM(e) := EM(e) \setminus m$ and all other parameters of G remaining the same. Finally, we say that the edge e is tight* iff *$tmin(e)$, $tmax(e)$, and $EM(e)$ are tight.*

We remark that this definition checks for the effect of single parameter changes. As an example, for the BPSS we could verify whether some behavior really exists where $S1_{NO}$ occurs only 4 time units after $B1_{Low}$, or whether that propagation can indeed occur in mode S_1. Completeness might be preserved when changing multiple parameters simultaneously, e.g. if a mode m on an edge e is dropped in which the propagation cannot occur at all and $tmax(e)$ is set to a finite value instead of $+\infty$ that is a correct bound for the propagation in the remaining modes, completeness is preserved, while just reducing $tmax(e)$ would break it.

4 Behavioral Validation of TFPGs

In this section we describe how to check whether a TFPG is a complete and tight abstraction of a system.

TFPG completeness can be reduced to a model checking problem of an MTL formula over the original system. TFPG trace validity is expressed directly w.r.t. system traces, using an association map, hence no composition of the system model and the TFPG transition system is necessary. Theorem 1 and Theorem 2 below provide partial proof obligations (for `OR` and `AND` nodes, respectively) to check whether the constraints of individual nodes are satisfied on a system trace.

Theorem 1. *Given a system model S, an association map Γ relating S to a given TFPG G, and an `OR` node d of G, we define the following proof obligations:*

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

For a trace π of S, $\zeta_\Gamma(\pi)$ satisfies the constraints of d, as per Def. 3, iff *$\pi \models \psi_{\texttt{OR}\cdot A}(d,\Gamma)$ and $\pi \models \psi_{\texttt{OR}\cdot B}(d,\Gamma)$.*

Theorem 2. *Given a system model S, an association map Γ relating S to a given TFPG G, and an `AND` node d of G, we define the following proof obligations:*

1. $\psi_{\texttt{AND}\cdot A}(d,\Gamma) := \mathsf{G}((\mathsf{O}\gamma_d) \rightarrow \mathsf{O}((\mathsf{O}\gamma_d) \wedge \bigwedge_{e=(v,d)\in E}((\mathsf{O}\gamma_v) \wedge \gamma_{\mu(e)}\mathsf{S}^{\geq tmin(e)}(\mathsf{O}\gamma_v) \wedge \gamma_{\mu(e)})))$
2. $\psi_{\texttt{AND}\cdot B}(d,\Gamma) := \mathsf{G}\neg(\bigwedge_{e=(v,d)\in E}((\mathsf{O}\gamma_v) \wedge \gamma_{\mu(e)} \wedge \neg(\mathsf{O}\gamma_d)\mathsf{S}^{>tmax(e)}((\mathsf{O}\gamma_v) \wedge \gamma_{\mu(e)} \wedge \neg(\mathsf{O}\gamma_d)))$

For a trace π of S, $\zeta_\Gamma(\pi)$ satisfies the constraints of d, as per Def. 4, iff *$\pi \models \psi_{\texttt{AND}\cdot A}(d,\Gamma)$ and $\pi \models \psi_{\texttt{AND}\cdot B}(d,\Gamma)$.*

Based on Theorems 1 and 2, Theorem 3 formulates the proof obligation that a system trace must satisfy in order for the corresponding TFPG trace to satisfy a given TFPG.

Theorem 3. *Given a system model S, a TFPG G, and an association map Γ relating S to G, let $\Psi(G,\Gamma) := \bigwedge_{d\in\texttt{OR(G)}}(\psi_{\texttt{OR}\cdot A}(d,\Gamma) \wedge \psi_{\texttt{OR}\cdot B}(d,\Gamma)) \wedge \bigwedge_{d\in\texttt{AND(G)}}(\psi_{\texttt{AND}\cdot A}(d,\Gamma) \wedge \psi_{\texttt{AND}\cdot B}(d,\Gamma))$. Then, G is complete w.r.t. S* iff *$S \models \Psi(G,\Gamma)$.*

The intuition behind the Theorems is that the proof obligations can be derived from the definitions of OR-node satisfaction and AND-node satisfaction via the semantics of temporal operators and the mappings of Definition 7.

Note that, given an edge $e = (v,d)$, the subclauses relative to e in $\psi_{\texttt{OR}\cdot B}(d,\Gamma)$ and $\psi_{\texttt{AND}\cdot B}(d,\Gamma)$ are trivially false when $tmax(e) = +\infty$ and can be simplified accordingly. Furthermore, the fact that the proof obligations consist of a number of local checks makes it easier to pinpoint the source of any TFPG constraint violation.

Edge tightness can be reduced to a number of completeness checks. As per Def. 9, checking the tightness of an edge $e \in E$ amounts to verifying, individually for each parameter $tmin(e)$, $tmax(e)$, and $EM(e)$, if there exists a tighter assignment such that the accordingly modified TFPG G' is still complete w.r.t. S: $S \models \Psi(G',\Gamma)$. This can be done by searching over the range of possible tighter parameter assignments. Note that tightness checks for edge e only require to evaluate the proof obligations affected by the parameter change, e.g., if d is an `OR` node and we check the tightness of $tmin(e)$, then only $\psi_{\texttt{OR}\cdot A}(d,\Gamma)$ needs to be evaluated.

5 Implementation

The prototype created for the present work is implemented on top of the safety analysis platform `xSAP` (Bittner et al. 2015), which in turn is based on `nuXmv` (Cavada et al. 2014), a symbolic model checker for infinite state transition systems modeled in the SMV language.

At the core of the implementation we use a reduction of the completeness check to a reachability problem. For expressions of the type $\mathsf{O}\gamma_d$ we extend the system model with corresponding history monitors. Furthermore we introduce one stopwatch per edge e that measures the duration for which the corresponding expression $(\mathsf{O}\gamma_v) \wedge \gamma_{\mu(e)}$ has been true on the current path; the stopwatch is disabled when the edge is not active and frozen when the target discrepancy is activated. Based on this, the MTL proof obligations are expressed as invariance proof obligations, which can be solved with standard reachability algorithms. For instance, to verify the proof obligation $\psi_{\texttt{OR}\cdot A}(d, \Gamma)$, we check whether the clock value of at least one edge reaching the respective OR discrepancy is greater or equal to the corresponding t_{min} value when the discrepancy is activated. We remark that the reachability problem for infinite-state transition systems is in general undecidable, and plan to adress the computational complexity of decidable subclasses in future work.

In order to compute tight assignments to the time bounds of transitions we rely on recent developments in parameter synthesis (Bittner et al. 2014a) and the symbolic model checking algorithm IC3 (Bradley 2011) as implemented in xSAP. The idea is to explore the lattice of solutions top-down and to stop when a tight solution has been found. We perform the search for tighter time bounds according to the highest precision of any time constant in the original TFPG; the rationale is that in practice the precision of interest is always finite. Key to an efficient implementation is to reuse the inductive invariant returned by IC3, representing an overapproximation of the reachable state space, to bootstrap subsequent calls to the model checker and enable searching for tighter assignments without the necessity of unrolling the transition relation.

Our current implementation does not support tightening of t_{max} bounds set to ∞. To check for the existence of a value (in an infinite domain) for some t_{max} that guarantees completeness, more advanced proving techniques are required that go beyond simple model checking. The implementation also does not yet support tightening of modes and focuses on the tightening of time bounds only, even though the proposed framework enables it. We plan to extend in future work the implementation to cover both issues, based on a tight integration with more advance proving techniques.

6 Experiments

In this section we provide an experimental evaluation of the developed algorithms for behavioral validation and tightening of TFPGs.

For the experimental evaluation we use the following use cases. ACEX and AUTOGEN are artificial models based on a state space derived from partially random graphs, containing discrete clocks. BATTERY SENSOR is the model described in Section 2.3. It is also mainly discrete with real-valued clocks. CASSINI are models of the spacecraft propulsion system described in (Williams and Nayak 1996). SHUTTLE GUIDANCE is a purely discrete model with discrete clocks of the Space Shuttle engines contingency guidance requirements. FORGEROBOT describes a robot in a hazardous environment and how its protection mechanisms can fail. POWERDIST describes the fault protection logic of a power distribution application. Additionally we ran our implementation on two discrete untimed industrial models, 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.[2] In total we created 72 tight TFPGs. For all usecases except for the untimed WBS and X34, we created two "relaxed" versions of the TFPGs, one by putting all t_{min} values to 0, and one by putting all t_{min} values to 0 and multiplying by 2 all t_{max} values, resulting in a overall number of 212 use cases. We ran both the completeness check and the tightening procedure on each use case.

[2]Benchmark files and theorem proofs are downloadable at es.fbk.eu/people/bittner/aaai16.tar.gz.

model	bool	real	TFPGs	avg. FM	avg. D	avg. E
acex-10	31	0	5	2	15	16
acex-12	35	0	11	2	17	18
autogen	99	0	22	8	15	23
battery	43	5	4	4	6	11
cassini2	241	6	5	10	6	11
cassini4	301	10	13	16	10	27
forge-B	16	0	1	2	3	5
forge-R1	10	3	1	2	3	5
forge-R2	17	5	1	4	8	14
forge-R3	25	7	1	6	13	23
guidance	98	0	4	6	6	13
pdist	84	0	2	7	7	19
wbs	1179	0	1	9	8	19
x34	553	0	1	9	18	32

Figure 2: Usecase statistics with number of Boolean and real variables per model, number of tight TFPGs, average numbers of failure modes, discrepancies, and edges.

All tests were run on a dedicated 64bit Linux computer with a 12 core Intel Xeon CPU at 2.67 GHz and 100GB of RAM. 4 cores were reserved for each test run to limit time skew due to parallel executions of the tests. Each test was executed on a single core with a time limit of 3600 seconds and a memory limit of 4GB.

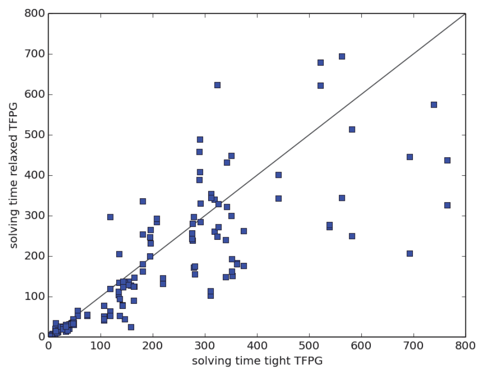

Figure 3: Solving time (seconds) of tight vs. relaxed TFPGs

All completeness checks terminated within the timeout (all except four within 800s) and IC3 was able to prove completeness in all cases, which shows the feasibility of the approach. Figure 3 shows the results for these checks, comparing the solving times for the tight TFPGs vs. their relaxed variants. From the plot it can also be seen that proving completeness is slightly easier for relaxed TFPGs. Furthermore, the check for WBS terminated after 67s, and the one for X34 after 21s. Also most tightening runs terminated within the timeout bound; 10 went out-of-time, and 5 out-of-memory. Results are shown in Figure 4, plotting them against the solving times of the respective completeness check. Not surprisingly the problem is much harder

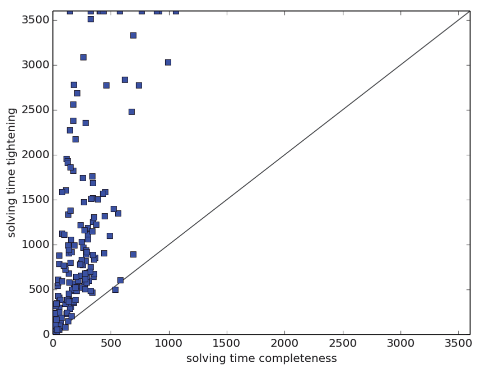

Figure 4: Solving time (seconds) of completeness check vs. tightening of time bounds

than verification of completeness. However, most still terminate successfully within the timeout, which shows the efficiency of current parameter synthesis techniques in solving the problem.

7 Conclusion

In this paper, we investigated the application of Timed Failure Propagation Graphs (TFPGs) as abstractions of dynamic systems with continuous time. We presented an approach, based on model checking, to validate the completeness and tightness of a TFPG, and to synthesize a set of tight time bounds. The experiments carried out on a set of industrial benchmarks demonstrate the practicality of the approach.

In the future, we will investigate the application of TFPGs as ways to present sets of counterexamples in model checking. We also want to synthesize the underlying graph using only failure mode and discrepancy definitions, and investigate scalability improvements obtained by reduction to the discrete case.

References

Abdelwahed, S.; Karsai, G.; Mahadevan, N.; and Ofsthun, S. 2009. Practical implementation of diagnosis systems using timed failure propagation graph models. *Instrumentation and Measurement, IEEE Transactions on* 58(2):240–247.

Alur, R., and Henzinger, T. A. 1993. Real-time logics: complexity and expressiveness. *Information and Computation* 104(1):35–77.

Bajwa, A., and Sweet, A. 2003. The livingstone model of a main propulsion system. In *Proceedings of the IEEE Aerospace Conference*, 63–74.

Bittner, B.; Bozzano, M.; Cimatti, A.; Gario, M.; and Griggio, A. 2014a. Towards pareto-optimal parameter synthesis for monotonic cost functions. In *Proceedings of the 14th Conference on Formal Methods in Computer-Aided Design*, 23–30. FMCAD Inc.

Bittner, B.; Bozzano, M.; Cimatti, A.; De Ferluc, R.; Gario, M.; Guiotto, A.; and Yushtein, Y. 2014b. An integrated process for FDIR design in aerospace. In *Model-Based Safety and Assessment*. Springer. 82–95.

Bittner, B.; Bozzano, M.; Cavada, R.; Cimatti, A.; Gario, M.; Griggio, A.; Mattarei, C.; Micheli, A.; and Zampedri, G. 2015. The xSAP safety analysis platform. *arXiv preprint arXiv:1504.07513*.

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

Bozzano, M.; Cimatti, A.; Gario, M.; and Micheli, A. 2015b. Smt-based validation of timed failure propagation graphs. In *Twenty-ninth AAAI Conference on Artificial Intelligence*.

Bozzano, M.; Cimatti, A.; and Tapparo, F. 2007. Symbolic fault tree analysis for reactive systems. In *Automated Technology for Verification and Analysis*. Springer. 162–176.

Bradley, A. R. 2011. Sat-based model checking without unrolling. In *Verification, Model Checking, and Abstract Interpretation*, 70–87. Springer.

Cavada, R.; Cimatti, A.; Dorigatti, M.; Griggio, A.; Mariotti, A.; Micheli, A.; Mover, S.; Roveri, M.; and Tonetta, S. 2014. The nuxmv symbolic model checker. In *Computer Aided Verification*, 334–342. Springer.

Dubey, A.; Karsai, G.; and Mahadevan, N. 2013. Fault-adaptivity in hard real-time component-based software systems. In *Software engineering for self-adaptive systems II*. Springer. 294–323.

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

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

Koymans, R. 1990. Specifying real-time properties with metric temporal logic. *Real-time systems* 2(4):255–299.

McDermott, R.; Mikulak, R. J.; and Beauregard, M. 1996. *The basics of FMEA*. SteinerBooks.

Misra, A.; Sztipanovits, J.; Underbrink, A.; Carnes, R.; and Purves, B. 1992. Diagnosability of dynamical systems. In *Third International Workshop on Principles of Diagnosis*.

Misra, A. 1994. *Senor-based diagnosis of dynamical systems*. Ph.D. Dissertation, Vanderbilt University.

Ofsthun, S. C., and Abdelwahed, S. 2007. Practical applications of timed failure propagation graphs for vehicle diagnosis. In *Autotestcon, 2007 IEEE*, 250–259. IEEE.

Ouaknine, J., and Worrell, J. 2008. Some recent results in metric temporal logic. In *Formal Modeling and Analysis of Timed Systems*. Springer. 1–13.

Priesterjahn, C.; Heinzemann, C.; and Schafer, W. 2013. From timed automata to timed failure propagation graphs. In *Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), 2013 IEEE 16th International Symposium on*, 1–8. IEEE.

Strasser, S., and Sheppard, J. 2011. Diagnostic alarm sequence maturation in timed failure propagation graphs. In *AUTOTEST-CON, 2011 IEEE*, 158–165. IEEE.

Williams, B. C., and Nayak, P. P. 1996. A model-based approach to reactive self-configuring systems. In *Proceedings of the National Conference on Artificial Intelligence*, 971–978.