Explaining Safety Violations in Real-Time Systems

Thomas Mari, Thao Dang, Gregor Gössler

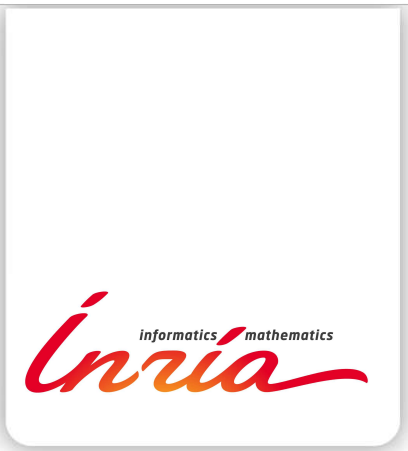
RESEARCH REPORT

N° 9420

September 2021

Project-Team SPADES

ISSN 0249-6399 ISRN INRIA/RR--9420--FR+ENG



Explaining Safety Violations in Real-Time Systems

Thomas Mari*†, Thao Dang†, Gregor Gössler*

Project-Team SPADES

Research Report n° 9420 — September 2021 — 19 pages

Abstract: We tackle the problem of explaining faults in real-time systems. Intuitively, an explanation of the violation of a safety property by an execution is a concise excerpt of the faulty execution that retains only the elements that were relevant for entailing the violation, thus exhibiting how causes accumulate over time and propagate to entail the effect. Fault explanation therefore goes beyond the well-known concepts of fault diagnosis and localization.
We provide a formal definition of causal explanations on dense-time models, based on the well-studied formalisms of timed automata and zone-based abstractions. Our approach is able to account for limited observability of the faulty execution. We propose a symbolic formalization to effectively construct such explanations, which we have implemented in a prototype tool. We illustrate our approach on several examples.

Key-words: causal explanations, real-time, safety

* Univ. Grenoble Alpes, INRIA, CNRS, Grenoble INP, LIG, 38000 Grenoble, France
† CNRS/VERIMAG, France

RESEARCH CENTRE GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Explication des violation de sûreté dans les systèmes temps réel

Résumé :

Nous nous intéressons au problème de l'explication de fautes pour des systèmes temps réel. Intuitivement, une explication de la violation d'une propriété de sûreté par une exécution est un extrait suffisamment concis de l'exécution ne contenant seulement les éléments pertinents pour entraîner la violation. Une explication permet donc d'exhiber comment les causes des violations s'accumulent et se propagent avec le temps.

L'explication de faute va au-delà des classiques diagnostique et localisation de fautes. Nous fournissons une définition formelle d'explication causale pour les modèles à temps dense. Celle-ci est basée sur le formalisme des automates temporisés et de l'abstraction basée sur les zones. Notre approche tient compte de l'observabilité partielle de l'exécution fautive. Nous proposons une formalisation symbolique pour construire en pratique ces explications. Cette construction a été implémentée dans un prototype. Nous illustrons notre approche avec des exemples.

Mots-clés : Explications causales, temps réel, sûreté

1 Introduction

Embedded real-time systems have been part of our daily lives for several decades now, and the number of such devices is growing exponentially. There is a rich body of work on software engineering, formal verification, and certification of such systems. However, when looking at the headlines about failures and casualties caused by embedded systems, one may wonder whether we have learned to fully master their growing complexity.

As a complementary approach, accountability aims at constructing systems in such a way that the responsibilities for a failure can be identified and explained *post mortem* [15]. Intuitively, an explanation is a concise excerpt of the observed execution that retains only the elements that were relevant for entailing the violation. However, formalizing this objective is far from obvious, and different communities have tackled the problem under different angles.

A first challenge consists in conveying the "right" amount of information, *e.g.*, in order to help a human expert in quickly understanding the causes of the violation, or to automatically react to the failure by swapping a component or changing parameters. In this work we are interested in *causal* explanations that allow us to track the chains from cause to effect while eliminating irrelevant events from the explanation.

A second challenge is to devise explanations that satisfy the requirements of embedded systems. First, embedded real-time systems usually have an infinite state space, requiring reasoning on an appropriate abstraction. Nevertheless, research on reasoning about causation on abstractions is only in its infancy [2, 7]. Second, explanations for embedded systems have to cope with the limited observability of state and events, which is again an issue that is hardly addressed in existing work on causation.

The work presented in this paper is part of an ongoing effort to construct explainable embedded real-time systems. We investigate how to construct, from a system model, a safety property, and an execution log that violates the property, a concise explanation of how the log brought about the violation of the property.

This paper makes the following contributions. We provide a formal definition of causal explanations on dense-time models, based on the well-studied formalism of timed automata. We propose a symbolic approach to effectively construct explanations. We illustrate our approach on several examples and a case study.

2 Related Work

Causal explanations. Our construction of choice explanations is based on effective choice explanations for discrete-event systems [6]. The latter are, in turn, inspired by [11], which leverages game theory to explain counterexample traces from model-checking by exhibiting the portions of the trace in which the system could have avoided the violation of an expected property no matter how the environment behaves.

Several authors have proposed a construction of explanations for the satisfaction of a property P by an execution trace based on sub-sequences of the trace that are sufficient to entail P, such as explanatory diagnoses [17] and causal compression [4]. However, in contrast to our approach, subsequence explanations do not convey any information about the outcome produced by alternative branches in a non-deterministic system.

Static program slicing [22] determines a part ("slice") of a program that influences a set of variables at a given point. Dynamic slicing [14] computes a slice for a given computation, yielding a smaller slice. In contrast to slicing, our approach accounts for counterfactual runs that have not been taken in the actual execution.

Fault diagnosis, localization, and repair. Counterfactual causation has been studied in many disciplines as a precise assessment of individual causes that contribute to bring about an effect, see e.g. the influential definition of *actual causality* in [9]. Some form of counterfactual reasoning has been used by many authors to diagnose, localize, and repair faults. We only cite some representative examples here. As these approaches exhibit individual causes rather than chains from cause to effect, they are less apt to explain how contributory causes accumulate over time and propagate to entail an effect. The seminal work of [19] proposes a framework of model-based fault diagnosis that defines a diagnosis as a minimal sets of components whose faults make the observations consistent with the system model. The use of a distance metric is explored in [8] to localize, based on an error trace, a possible fault as the difference between the error trace and a closest correct trace. Similarly, Delta debugging [24] starts from a failing and a passing input and finds a pair of a failing and a passing input with minimal distance. A variant of actual causality is used in [3] to over-approximate the set of causes for the first violation of an LTL formula by a trace. The goal of program repair is, given a program that violates an expected property P, to construct a syntactically close program that satisfies P, see e.g. [21] for the repair of reactive programs. Closer to our setting, [12] uses MaxSMT to repair clock bounds in a network of timed automata so as to ensure an expected property.

In contrast to the pieces of related work cited above that are based on some form of model to compare the actual execution with counterfactual traces, many recent techniques summarized under the umbrella term of *explainable AI* lack a model and hence, the possibility of counterfactual reasoning [18].

3 Preliminaries

Let C be a set of clock variables that take values in $\mathbb{R}^+$. A C-valuation is a function $C \to \mathbb{R}^+$. Let $\mathbf{0}$ denote the C-valuation assigning 0 to all clocks. An atomic constraint on C is an inequality $c \sim k$ or $c - c' \sim k$ where $c, c' \in C$, $\sim \in \{\leq, <, \geq, >\}$ and $k \in \mathbb{N}$. We say that a C-valuation v satisfies an atomic constraint $c \sim k$ (or $c - c' \sim k$) if $v(c) \sim k$ (or $v(c) - v(c') \sim k$). A C-constraint is a finite conjunction of all atomic constraints on C. Let $\mathbb{C}$ denote the set of C-constraints. By abuse of notation, we use a C-constraint interchangably with the sets of C-valuations that satisfy it.

Definition 1 (Timed Automaton)**.** *A timed automaton (TA) $\mathcal{A}$ is a tuple $\mathcal{A} = \langle \Sigma, L, L_0, C, F, \mathcal{I}, E \rangle$ where*

- Σ *is a finite set of* events*;*
- L *is a finite set of* locations*;*
- $L_0 \subseteq L$ *is the set of* initial locations*;*
- C *is a set of* clock variables*;*
- $F \subseteq L$ *is a set of* accepting locations*;*
- $\mathcal{I} : L \to \mathbb{C}$ *specifies for each location an* invariant*;*
- $E \subseteq L \times \mathbb{C} \times \Sigma \times 2^C \times L$ *is a set of edges of the form $e = \langle \ell, g, \sigma, X, \ell' \rangle$ where ℓ and ℓ' are respectively* source *and* target locations*; σ is an* event*; g is the* guard *of e; and X is a set of clocks to be* reset *when the edge is traversed.*

To account for the fact that some events are not observable, the set Σ of events is partitioned into two subsets of observable and unobservable events.

We formalize the semantics of timed automata using labeled transition systems, or LTS. An LTS is a tuple $\langle \Sigma, \mathcal{V}, \mathcal{E}, \mathcal{V}^0, \mathcal{V}^F \rangle$ where Σ is the alphabet, $\mathcal{V}$ is a set of nodes, $\mathcal{E} \subseteq \mathcal{V} \times \Sigma \times \mathcal{V}$ is the set of labeled transitions, and $\mathcal{V}^0 \subseteq \mathcal{V}$ and $\mathcal{V}^F \subseteq \mathcal{V}$ are the sets of initial and accepting nodes, respectively. For $\nu \in \mathcal{V}$ let $\nu^\bullet = \{\nu' \in \mathcal{V} \mid \exists \sigma \in \Sigma : (\nu, \sigma, \nu') \in \mathcal{E}\}$ be the postset of ν. In the following we use the terms LTS and graph interchangeably.

Definition 2 (Semantic LTS)**.** *The* semantic LTS *of a timed automaton $\mathcal{A} = \langle \Sigma, L, L_0, C, F, \mathcal{I}, E \rangle$ is the LTS $sem(\mathcal{A}) = \langle \Sigma', \mathcal{V}, \mathcal{E}, \mathcal{V}^0, \mathcal{V}^F \rangle$ where*

- $\Sigma' = \Sigma \cup \mathbb{R}^+$ *is the set of labels;*
- $\mathcal{V} = \{(\ell, v) \mid \ell \in L \wedge v \in \mathcal{I}(\ell)\}$*, i.e. the states of $\mathcal{A}$ are the pairs (ℓ, v) where $v \in \mathcal{I}(\ell)$ is a clock valuation that satisfies the invariant of the location ℓ;*
- $\mathcal{V}^0 = \{(\ell, v) \in \mathcal{V} \mid \ell \in L_0 \wedge v = \mathbf{0}\}$*;*
- $\mathcal{V}^F = \mathcal{V}$ *(indeed the accepting states do not matter);*
- *the set of transitions are of two types, discrete and time transitions:*

$$\begin{aligned}\mathcal{E} = &\big\{((\ell, v), \sigma, (\ell', v')) \mid \exists g, X : \langle \ell, g, \sigma, X, \ell' \rangle \in E \wedge v' = v[X := 0] \wedge v' \in \mathcal{I}(\ell')\big\} \\ &\cup \big\{\big((\ell, v), \delta, (\ell, v')\big) \mid \delta > 0 \wedge v' = v + \delta \wedge v' \in \mathcal{I}(\ell)\big\}\end{aligned}$$

The states (ℓ', v') and (ℓ, v') are respectively called e-successor δ-successor of (ℓ, v).

As usual we write $(\ell, v) \xrightarrow{\cdot} (\ell', v')$ for $\big((\ell, v), \cdot, (\ell', v')\big) \in \mathcal{E}$. By abuse of notation we omit the curly braces in $\{\nu^0\}$ when there is a single initial state. Note that since δ is a real number, a time transition $(\ell, v) \xrightarrow{\delta} (\ell, v')$ can be split into an arbitrary number k of time transitions, that is, $(\ell, v) \xrightarrow{\delta_1} (\ell, v_1) \xrightarrow{\delta_2} (\ell, v_2) \ldots \xrightarrow{\delta_k} (\ell, v_k)$ such that $\delta = \delta_1 + \delta_2 + \ldots + \delta_k$.

Definition 3 (Runs and Traces)**.** *Let $G = \langle \Sigma, \mathcal{V}, \mathcal{E}, \mathcal{V}^0, \mathcal{V}^F \rangle$ be an LTS.*

- *A* run *of G starting from $\nu_1 \in \mathcal{V}$ is a (finite or infinite) sequence of states and transitions: $\rho = \nu_1 \xrightarrow{e_1} \nu_2 \xrightarrow{e_2} \ldots$. We denote by $\Upsilon(G)$ the set of all runs of G. A state ν is* reachable *if there is a run from an initial state to ν.*
- *Given a run $\rho \in \Upsilon(G)$, the sequence of labels in ρ is called* trace*. We denote by $\Psi(G)$ the set of all traces of G.*

Definition 4 (Timed Log)**.** *A timed log is a one-clock, deterministic and acyclic TA. In addition, all the events of the automaton are observable.*

An example of timed log is a timed automaton roughly depicted as follows: $l_0 \xrightarrow{x=t_0,a,\emptyset} l_1 \xrightarrow{x=t_1,b,\emptyset} l_2 \xrightarrow{x=t_2,a,\emptyset} l_3$, with l_3 as an accepting location and l_0 an initial location. The edge from the location l_0 to the location l_1 has the guard $x = t_0$; its event is a; and its set of clocks to be reset is empty.

As mentioned in the introduction, we are interested in explaining why a system, modeled as a timed automaton $\mathcal{A}$, violates a safety property. To this end, we define a *safety property observer* as a timed automaton with one sink state, to model the property violation of a timed log. In addition, this observer is required to be receptive with respect to all observable events of $\mathcal{A}$, as

defined in the following. An LTS $\langle \Sigma, \mathcal{V}, \mathcal{E}, \mathcal{V}^0, \mathcal{V}^F \rangle$ is *receptive* over $\Sigma_1 \subseteq \Sigma$ if at every reachable state ν, all events in Σ_1 are enabled, that is, $\forall \sigma \in \Sigma_1 \; \exists \nu' : \nu \xrightarrow{\sigma} \nu'$. Given a timed log $\mathcal{L}$ and a safety property observer $\mathcal{P}$, we say that $\mathcal{L}$ violates the safety property at hand if each run of $\mathcal{L}$ is a run of $\mathcal{P}$ that reaches the sink state.

The sets of states and transitions of a timed automaton are infinite, and therefore, as in verification, finite abstractions can be used to construct explanations. In this work, we use time-abstracting bisimulations to abstract away the quantitative information about time lapses in the execution of a timed automaton. This leads to finite discrete abstractions of the original timed automata from which we can compute choice-based explanations [6]. The following definition is adapted from [20] so as to distingluish events, and accepting vs. non-accepting states.

Definition 5 (Strong bisimulations)**.** *A binary relation $\sim$ on an LTS $G = \langle \Sigma, \mathcal{V}, \mathcal{E}, \mathcal{V}^0, \mathcal{V}^F \rangle$ is a* strong bisimulation *if for any pair of nodes p and q of G such that $p \sim q$, the following conditions hold:*

- $\forall \sigma \in \Sigma \; \forall p' \in \mathcal{V} : \big(p \xrightarrow{\sigma} p' \implies \exists q' : q \xrightarrow{\sigma} q' \wedge p' \sim q'\big)$;
- *the above condition also holds when p and q are swapped;*
- $p \in \mathcal{V}^F \iff q \in \mathcal{V}^F$.

When G is the semantic LTS of a TA, the relation $\sim$ is a strong time-abstracting bisimulation (STAB) *if for any pair of nodes $p = \langle \ell_1, v_1 \rangle$ and $q = \langle \ell_2, v_2 \rangle$ of G such that $p \sim q$, the following conditions hold:*

- $\ell_1 = \ell_2$;
- $\forall \sigma \in \Sigma \setminus \mathbb{R}^+ \;\; \forall p' \in \mathcal{V} : \big(p \xrightarrow{\sigma} p' \implies \exists q' : q \xrightarrow{\sigma} q' \wedge p' \sim q'\big)$;
- $\forall \delta > 0 \; \forall p' \in \mathcal{V} : \big(p \xrightarrow{\delta} p' \implies \exists \delta' > 0 \; \exists q' : q \xrightarrow{\delta'} q' \wedge p' \sim q'\big)$;
- *the above conditions also hold when p and q are swapped;*
- $p \in \mathcal{V}^F \iff q \in \mathcal{V}^F$.

We use STAB, implemented in Minim [20], to construct a finite symbolic abstraction of TA. We then reduce the latter with respect to strong bisimulation so as to merge bisimilar states involving different locations.

4 Explanations

Our goal is to explain, for a TA $\mathcal{A}$ modelling a system, an observer $\mathcal{P}$ of a safety property, and a timed log $\mathcal{L}$ of observable events of $\mathcal{A}$ that violates $\mathcal{P}$, how the violation came to happen. More precisely, in this work, we focus on *non-deterministic choices* in the execution that entailed the failure, where different choices would have helped to avoid it. Prominent of such failures are "concurrency bugs" that only occur in certain interleavings of threads, and deadline misses due to bad scheduling decisions in real-time systems.

To construct such explanations for real-time systems, we lift the *effective choice explanations* formalized in [6] to timed automata. The main steps of our construction, as shown in Figure 1, are the following.

1. Construct a timed *log observer* $\mathcal{L}^c$ from $\mathcal{L}$ that tracks, for any run ρ of $\mathcal{L}^c$, whether the observable behavior of ρ produces $\mathcal{L}$.

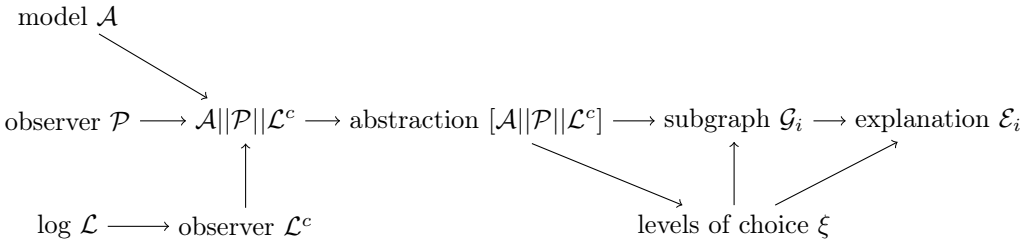


Figure 1: Overview of the approach.

2. Compose $\mathcal{A}$, $\mathcal{P}$, and $\mathcal{L}^c$ to form a timed automaton $\mathcal{A}||\mathcal{P}||\mathcal{L}^c$, where $\|$ is the standard parallel composition of timed automata, see *e.g.* [23].

3. Construct a discrete abstraction $[\mathcal{A}||\mathcal{P}||\mathcal{L}^c]$ of the continuous-time semantics of $\mathcal{A}||\mathcal{P}||\mathcal{L}^c$, using a time-abstracting bisimulation.

4. Compute the *levels of choice* on $[\mathcal{A}||\mathcal{P}||\mathcal{L}^c]$ that, intuitively, represent, for each equivalence class $q \in [\mathcal{A}||\mathcal{P}||\mathcal{L}^c]$, the number of *bad choices* $\xi(q)$ left before violating $\mathcal{P}$.

5. Extract, from $[\mathcal{A}||\mathcal{P}||\mathcal{L}^c]$, a sequence of subgraphs $\mathcal{G}_i$ representing the traces that will be condensed to an explanation of length i.

6. Abstract away, from each $\mathcal{G}_i$, all transitions that do not decrease ξ, in order to obtain the explanation $\mathcal{E}_i$ that retains only those (discrete or timed) transitions that contributed to the failure.

We will now discuss each of these steps.

Log observer.

Given an alphabet Σ and a timed log $\mathcal{L}$ over the observable alphabet $\Sigma^{obs} \subseteq \Sigma$, we construct a *log observer* that accepts all runs over Σ and enters an accepting *sink* state whenever an observed behavior is inconsistent with the log.

Definition 6 (Log observer)**.** *The observer of a log* $\mathcal{L} = \langle \Sigma, L, L_0, C, F, \mathcal{I}, E \rangle$ *is the TA* $\mathcal{L}^c := \langle \Sigma, L', L_0, C, F, \mathcal{I}', E' \rangle$ *where*

- $L' = L \cup \{sink\}$ *where* $sink \notin L$ *is a fresh location;*
- $\mathcal{I}' = \mathcal{I} \cup \{sink \mapsto true\}$*;*
- $E' = E \cup E_1 \cup E_2$ *where*
  - $E_1 = \{\ell \xrightarrow{\mathcal{C}(\ell,\sigma),\sigma,\emptyset} sink \mid \ell \in L \wedge \sigma \in \Sigma\}$ *where* $\mathcal{C}(\ell,\sigma) = \neg \bigvee \{g \mid \exists \ell' \in L : \ell \xrightarrow{g,\sigma,\cdot} \ell'\}$.
  - $E_2 = \{sink \xrightarrow{true,\sigma,\emptyset} sink \mid \sigma \in \Sigma\}$.

Intuitively, E_1 is the set of edges from a location in $\mathcal{L}$ that are not consistent with $\mathcal{L}$, used to make the log observer receptive with respect to Σ^{obs}.

Discrete abstraction.

To obtain discrete abstractions for the timed automaton $\mathcal{A}||\mathcal{P}||\mathcal{L}^c$, we use the STAB of Definition 5. Let us briefly recall the notion of time-abstracting quotient graph [20]. Given a TA over alphabet Σ with semantic LTS $G = \langle \Sigma \cup \mathbb{R}^+, \mathcal{V}, \mathcal{E}, \mathcal{V}^0, \mathcal{V}^F \rangle$ and a partition $\tilde{\mathcal{V}}$ of $\mathcal{V}$, the quotient of G with respect to $\tilde{\mathcal{V}}$ is the LTS $G/\tilde{\mathcal{V}} = \langle \Sigma', \tilde{\mathcal{V}}, \mathcal{E}', \tilde{\mathcal{V}}^0, \tilde{\mathcal{V}}^F \rangle$ where

- $\Sigma' = \Sigma \cup \{\delta\}$ where δ is a fresh symbol;
- $\mathcal{E}' = \{\tilde{\nu} \xrightarrow{\sigma} \tilde{\nu}' \mid (\nu, \sigma, \nu') \in \mathcal{E} \wedge \sigma \in \Sigma\} \cup \{\tilde{\nu} \xrightarrow{\delta} \tilde{\nu}' \mid \exists t > 0 : (\nu, t, \nu') \in \mathcal{E}\}$;
- $\tilde{\mathcal{V}}^0 = \{\tilde{\nu} \mid \nu \in \mathcal{V}^0\}$ and $\tilde{\mathcal{V}}^F = \{\tilde{\nu} \mid \nu \in \mathcal{V}^F\}$

and for $\nu \in \mathcal{V}$, $\tilde{\nu}$ denotes the element of $\tilde{\mathcal{V}}$ for which $\nu \in \tilde{\nu}$.

In particular, we are interested in the quotient with respect to the equivalence classes of states, called symbolic states, induced by STAB. Let $[\mathcal{A}||\mathcal{P}||\mathcal{L}^c] := \mathcal{A}||\mathcal{P}||\mathcal{L}^c/\sim$ be the quotient graph with respect to STAB. This quotient graph can be computed by the existing timed automata model-checkers, such as UPPAAL and Kronos [23, 16]. In this work, we use the tool Minim [20] integrated in Kronos.

Levels of choice.

In the following, we compute the level of choice on $[\mathcal{A}||\mathcal{P}||\mathcal{L}^c]$. Intuitively, the level of choice measures how close the system is to violating the required safety property.

Definition 7 (Level of choice)**.** *Given an LTS $G = \langle \Sigma, \mathcal{V}, \mathcal{E}, \mathcal{V}^0, \mathcal{V}^F \rangle$, we say that $\xi : \mathcal{V} \to \mathbb{N} \cup \{\infty\}$ is the level of choice function if:*

1. $\forall \nu \in \mathcal{V}^F : \xi(\nu) = 0$
2. *$\forall \nu \in \mathcal{V}$ such that ν is not co-reachable, $\xi(\nu) = \infty$*
3. *for all other states $\nu \in \mathcal{V}$ let*

$$\xi(\nu) = \min^+\big(\{\ell \mid \exists \nu' : \xi(\nu') = \ell \wedge \exists e \in \Sigma : \nu \xrightarrow{e} \nu'\}\big)$$

where

$$\min^+(G) = \begin{cases} \min(G) & \text{if } |G| \leq 1 \\ 1 + \min(G) & \text{otherwise} \end{cases}$$

and we set $\min \emptyset = \infty$.

4. *Maximality: ξ is maximal among the functions fulfilling the preceding conditions.*

Definition 8 (Effective choice)**.** *Given an LTS $G = \langle \Sigma, \mathcal{V}, \mathcal{E}, \nu^0, \mathcal{V}^F \rangle$ and a level of choice function $\xi : \mathcal{V} \to \mathbb{N} \cup \{\infty\}$, a transition $s \xrightarrow{e} s' \in \mathcal{E}$ is an effective choice transition iff $\xi(s) = 1 + \xi(s')$ and $\exists \rho = s' \xrightarrow{e_0} s_1 \xrightarrow{e_1} ...s_n \in run(G)$ such that $\xi(s_n) = 0$ and $(\max_{s \in \rho} \xi(s)) = \xi(s')$. When such a transition exists, the state s is called an effective choice state.*

Intuitively, an effective choice transition is a transition that decrements the level of choice and is prefix of a run ρ violating $\mathcal{P}$ along which the level of choice no longer exceeds $\xi(s')$.

Lemma 1. *Let $G = \langle \Sigma, \mathcal{V}, \mathcal{E}, \mathcal{V}^0, \mathcal{V}^F \rangle$ and ξ the level of choice of G. Let $\mathcal{V}^C = \big\{\nu \in \mathcal{V} \mid \xi(\nu) \in \mathbb{N} \setminus \{0\} \wedge \xi(\nu) = 1 + min\{\xi(\nu') \mid \nu' \in \nu^\bullet\}\big\}$ and $\mathcal{V}^{NC} = \big\{\nu \in \mathcal{V} \mid \xi(\nu) \in \mathbb{N} \setminus \{0\}\big\} \setminus \mathcal{V}^C$. Then $\mathcal{E} \cap (\mathcal{V}^{NC} \times \Sigma \times \mathcal{V}^{NC})$ is acyclic.*

That is, $\mathcal{V}^C$ is inevitable from $\mathcal{V}^{NC}$.

Proof. If there is a cycle such that all the states have the same finite level of choice, then from all those states there exists a path to $\mathcal{V}^F$, but there exists also a path that avoid indefinitely $\mathcal{V}^F$ by repeating the cycle. Leaving the cycle should decrease the level of choice in the path to $\mathcal{V}^F$. Hence at least one of those states is in $\mathcal{V}^C$. □

Theorem 1. *Given an LTS $G = \langle \Sigma, \mathcal{V}, \mathcal{E}, \mathcal{V}^0, \mathcal{V}^F \rangle$, a level of choice function ξ on G, and a strong bisimulation $\sim$ on $\mathcal{V}$, we have that $s \sim s' \implies \xi(s) = \xi(s')$.*

Proof. Toward contradiction: Let us suppose that we have $G = \langle \Sigma, \mathcal{V}, \mathcal{E}, \mathcal{V}^0, \mathcal{V}^F \rangle$, a level of choice function ξ on G, and a strong bisimulation $\sim$ on $\mathcal{V}$ and

$$\exists \nu, \nu' \in \mathcal{V}, \nu \sim \nu' \wedge \xi(\nu) \neq \xi(\nu') \tag{H}$$

Let $n \in \mathbb{N}$ be the smallest level of choice such that there exists one state at level n which is bisimilar with a state at level $n' > n$. Formally, n is the smallest integer such that $P(n)$ with:

$$P(n) := \exists \nu, \nu' \in \mathcal{V}, (\nu \sim \nu') \wedge \xi(\nu) = n \wedge \xi(\nu) < \xi(\nu') \tag{P(n)}$$

The existence of n is implied by the hypothesis (H).

Case $n = 0$: Let $\nu \in \mathcal{V}$ such that $\xi(\nu) = 0$ and $\exists \nu' \in \mathcal{V}, \nu \sim \nu' \wedge \xi(\nu') > 0$. $\xi(\nu') > 0$ implies $\nu' \notin \mathcal{V}^F$ by definition of the level of choice and therefore $\nu \notin \mathcal{V}^F$ because $\nu \sim \nu'$ by definition of $\sim$.

$\xi(\nu') > 0$ implies that there exists a successors ν'' of ν' such that $\xi(\nu'') > 0$. Therefore from ν' we can build a run where $\xi^{-1}(0)$ can be avoided indefinitely (or reach a state non-coreachable wrt $\mathcal{V}^F$) while from ν, all paths reach $\mathcal{V}^F$ within a bounded number of transitions. We can build two sequentially bisimilar paths, from ν and from ν' where the destinations are bisimilar but one is in $\mathcal{V}^F$ while the other destination is not.

Case $n > 0$: Because n is minimal we know that for lower level of choice, bisimilarity implies same level of choice :

$$\forall n' \in \mathbb{N}, n' < n \implies \forall \nu, \nu' \in \mathcal{V}, \nu \sim \nu' \wedge \xi(\nu) = n' \implies \xi(\nu) = \xi(\nu') \tag{1}$$

Let $\mathcal{V}_n^C = \{\nu \in \mathcal{V} \mid \xi(\nu) = n \wedge \xi(\nu) = 1 + min\{\xi(\nu') \mid \nu' \in \nu^\bullet\}$

Let $\nu \in \mathcal{V}$ be the closest (wrt. d) to $\mathcal{V}_n^C$ such that

$$\xi(\nu) = n \wedge \exists \nu' \in \mathcal{V}, (\nu \sim \nu') \wedge \xi(\nu') > n \tag{2}$$

ν exists because $P(n)$ is true.

Subcase $\nu \in \mathcal{V}_n^C$:

By definition of the level of choice, $\xi(\nu)$, ν has one successor ν_{min} such that $\xi(\nu_{min}) = n - 1$. Let $\nu'_{min} \in \nu'^\bullet$ such that $\nu_{min} \sim \nu'_{min}$ a successor of ν', it exists because $\nu \sim \nu'$. $\xi(\nu_{min}) < n$ therefore by induction hypothesis $\xi(\nu'_{min}) = \xi(\nu_{min}) = n - 1$.

By definition of ξ, we have $\xi(\nu') \leq 1 + min\{\xi(v'') \mid v'' \in \nu'^\bullet\} \leq 1 + \xi(\nu'_{min}) = n$. We have a contradiction with $\xi(\nu') > \xi(\nu)$.

Subcase $\nu \notin \mathcal{V}_n^C$:

All successors of ν are also at level n, and at least one, let us call it ν_{min}, is strictly closer to $\mathcal{V}_n^C$ wrt d, than ν. We have either $\nu_{min} \in \mathcal{V}_n^C$ or $(\forall \nu'_{min} \in \mathcal{V}, \nu_{min} \sim \nu'_{min} \implies \xi(\nu'_{min}) = n)$. Let ν'_{min} the successor of ν' such that $\nu_{min} \sim \nu'_{min}$, it exists because $\nu \sim \nu'$.

If $\nu_{min} \in \mathcal{V}_n^C$ then $\xi(\nu'_{min}) \leq \xi(\nu_{min})$, therefore $\xi(\nu') \leq n+1$, therefore $\xi(\nu') = n+1$ because $\xi(\nu') > n$.

If $\nu_{min} \notin \mathcal{V}_n^C$ then $\xi(\nu'_{min}) = \xi(\nu_{min}) = n$, therefore $\xi(\nu') \leq n+1$, therefore $\xi(\nu') = n+1$.

$\xi(\nu') = n+1 \wedge \xi(\nu'_{min}) \leq n \implies \exists \nu'_{max} \in \nu'^{\bullet}, \xi(\nu'_{max}) \geq n'$ and because $\nu \sim \nu'$, $\exists \nu_{max} \in \nu'^{\bullet}_{max}, \nu_{max} \sim \nu'_{max}$ and because $\nu \notin \mathcal{V}_n^C$, $\xi(\nu_{max}) = \xi(\nu) = n$

For $x, y \in \mathcal{V}$, let $P(n, x, y) := x \sim y \wedge \xi(x) = n \wedge \xi(x) < \xi(y)$

We proved that $\exists \nu, \nu' \in \mathcal{V}^2, P(n, \nu, \nu') \implies \exists \nu_{max}, \nu'_{max} \in \nu^{\bullet} \times \nu'^{\bullet}, P(n, \nu_{max}, \nu'_{max})$

We can iterate until we have $P(n, x, y)$ with $x \in \mathcal{V}_n^C$. There is a bounded number of iterations because of the lemma 1 there are no cycles in $\mathcal{V}^{NC}$, especially in $\xi^{-}1(n) \cap (\mathcal{V}^{NC})$. We can then conclude with the same contradiction of the previous subcase. □

This theorem allows us to work on the LTS further reduced with respect to strong bisimulation instead of $[\mathcal{A}||\mathcal{P}||\mathcal{L}^c]$, as we know that bisimulation preserves the level of choice.

Once we have the abstraction $[\mathcal{A}||\mathcal{P}||\mathcal{L}^c]$ and labeling with levels of choice, we are ready to extract the explanations. The basic idea is to extract, from $[\mathcal{A}||\mathcal{P}||\mathcal{L}^c]$, a graph that retains only

- the states of $[\mathcal{A}||\mathcal{P}||\mathcal{L}^c]$ that are co-reachable from the final location of $\mathcal{L}^c$, *i.e.*, the states that are consistent with the observed log, and
- the edges along which the level of choice decreases, *i.e.*, the ones that bring the system closer to a violation of $\mathcal{P}$.

To this end we proceed as follows.

Graph splitting.

Given an acyclic graph $G = \langle \Sigma, \mathcal{V}, \mathcal{E}, \mathcal{V}^0, \mathcal{V}^F \rangle$ equipped with a level of choice function $\xi : \mathcal{V} \to \mathbb{N} \cup \{\infty\}$, we compute a split graph as follows. For any $\nu \in \mathcal{V}$ let

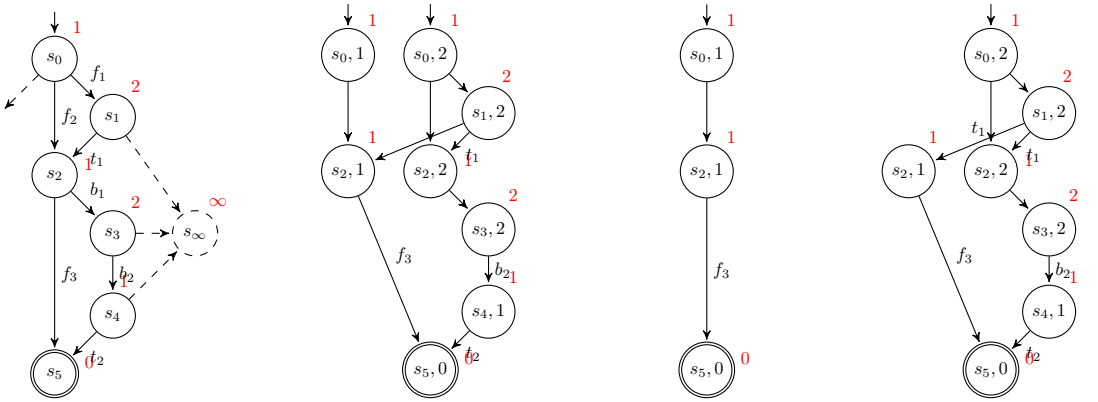
$$bounds(\nu) = \begin{cases} \{ \max\{\xi(\nu), bounds(\nu')\} \mid \nu' \in \nu^{\bullet} \} & \text{if } \nu^{\bullet} \neq \emptyset \\ \{\xi(\nu)\} & \text{otherwise} \end{cases}$$

We define the split graph $\mathcal{G} = \langle \Sigma, \mathcal{V}', \mathcal{E}', (\mathcal{V}')^0, (\mathcal{V}')^F \rangle$ where

$$\begin{aligned} \mathcal{V}' &= \big\{ (\nu, b) \mid \nu \in \mathcal{V} \wedge b \in bounds(\nu) \big\} \\ \mathcal{E}' &= \big\{ \big((\nu, b), e, (\nu', b')\big) \mid (\nu, e, \nu') \in \mathcal{E} \wedge b = \max\{b', \xi(\nu)\} \big\} \\ (\mathcal{V}')^0 &= \big\{ (\nu, b) \in \mathcal{V}' \mid \nu \in \mathcal{V}^0 \big\} \\ (\mathcal{V}')^F &= \{ (\nu, b) \mid \nu \in \mathcal{V}^F \} \end{aligned}$$

That is, we duplicate the states according to the maximum levels of choice that may be encountered in the future, and update the edges so as to point to the matching copy. We extend ξ to the split graph by putting $\xi\big((\nu, b)\big) := \xi(\nu)$. Intuitively, $\mathcal{G}$ accepts the same traces as G, while ensuring that each state is either effective choice or not, independent of the future behavior.

Example 1. *Consider the discrete graph shown in Figure 2(a). From s_0 a fault f_1 or f_2 may occur, and f_1 may be coped with until a timeout t_1 occurs. From s_2, a second fault f_3 will entail a system failure. However, the initial fault can be handled by primary and secondary fallback mechanisms b_1 and b_2, until a timeout t_2 occurs. The split graph is shown in Figure 2(b).*

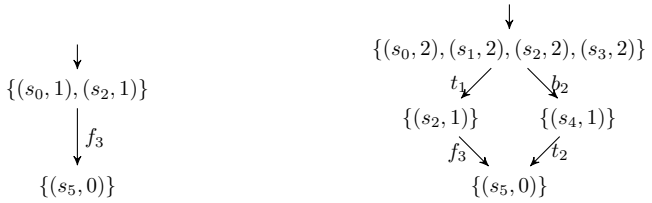


(a) Discrete graph with levels of choice.

(b) Split graph. Only labels of effective choice transitions are shown.

(c) Subgraph 1

(d) Subgraph 2



(e) Explanation $\mathcal{E}_1$. The shortest explanation of the system failure is the occurrence of event f_3.

(f) Explanation $\mathcal{E}_2$. The events that ultimately entailed the system failure are either $t_1; f_3$ or $b_2; t_2$.

Figure 2: Splitting, sub-graph extraction, and explanations obtained after determinization. Dashed transitions are in the model but not consistent with the log.

Subgraph extraction.

The explanations in the split graph are bounded in length by $[\xi(\nu_0), \max \xi]$, where $\max \xi$ is the maximum finite level of choice in $\mathcal{G}$. Our experiments suggest that explanations are easier to grasp when only explanations of the same length are presented simultaneously. We therefore extract, from the split graph $\mathcal{G} = \langle \Sigma, \mathcal{V}', \mathcal{E}', (\mathcal{V}')^0, (\mathcal{V}')^F \rangle$, for $l \in [\xi(\nu_0), \max \xi]$, the graph $\mathcal{G}_l$ of explanations of length l by restricting the LTS $\langle \Sigma, \mathcal{V}'', \mathcal{E}'', (\mathcal{V}'')^0, (\mathcal{V}'')^F \rangle$ where

$$\begin{aligned} \mathcal{V}'' &= \{(\nu, b) \in \mathcal{V}' \mid b \leq l\} \\ \mathcal{E}'' &= \mathcal{E}' \cap (\mathcal{V}'' \times \Sigma \times \mathcal{V}'') \\ (\mathcal{V}'')^0 &= \{(\nu, b) \in (\mathcal{V}')^0 \mid b = l\} \\ (\mathcal{V}'')^F &= (\mathcal{V}')^F \cap \mathcal{V}'' \end{aligned}$$

to the states that are reachable from $(\mathcal{V}'')^0$, and from which some state in $(\mathcal{V}'')^F$ is reachable. Notice that $\mathcal{G}_l$ is the empty graph when there is no explanation of length l. We then construct, for each (non-empty) subgraph, an explanation by applying the standard determinization (τ-elimination) algorithm based on subset construction [1] to "collapse" the non-pertinent parts of the graph.

Example 2. *From the split graph of Figure 2(b), two sub-graphs of constant height, in terms of effective choice transitions, are extracted (Figures 2(c) and 2(d)). By determinization we obtain the explanations $\mathcal{E}_1$ and $\mathcal{E}_2$ of Figures 2(e) and 2(f) that highlight the decisive events for disjoint scenarios, with increasing complexity of the explanation.*

Theorem 2. *For each trace $w \in \Psi(\mathcal{G})$ there exists $l \in \left[\xi(\nu_0), \max \xi\right]$ such that $w \in \mathcal{G}_l$.*

Proof. Let $n \in \mathbb{N}$ and $w = e_1e_2e_3...e_n \in \Psi(\mathcal{G})$. Let $\rho = \nu_0 \xrightarrow{e_1} \nu_1 \xrightarrow{e_2} ... \xrightarrow{e_n} \nu_n \in \Upsilon(\mathcal{G})$. Let $\langle b_0, b_1, ..., b_n\rangle \in \mathbb{N}^{n+1}$ such that $\forall i \in [0,n], b_i = \max\{\xi(\nu_j) \mid j \in [i,n]\}$. Let $l = \max\{\xi(\nu_i) \mid i \in [0,n]\}$. We prove that $\rho' = (\nu_0, b_0) \xrightarrow{e_1} (\nu_1, b_1) \xrightarrow{e_1} ... \xrightarrow{e_n} (\nu_n, b_n) \in \Upsilon(\mathcal{G}_l)$.

First of all, let us prove that the states in the run ρ' are in the split graph $\mathcal{G}' = \langle \Sigma, \mathcal{V}', \mathcal{E}', (\mathcal{V}')^0, (\mathcal{V}')^F\rangle$. We know that $\forall i \in [0,n], \nu_i \in \mathcal{V}$, and therefore $\big(\forall i \in [0,n], b_i \in \mathit{bounds}(\nu_i)\big) \implies \big(\forall i \in [0,n], (\nu_i, b_i) \in \mathcal{V}'\big)$.

By induction on $n - i$, we prove that $\forall i \in [0,n], b_i \in \mathit{bounds}(\nu_i)$.

For $i = n$, $\rho \in \Upsilon(\mathcal{G}) \implies \nu_n \in \mathcal{V}^f \implies \mathit{bounds}(\nu_n) = \{0\}$ and $\nu_n \in \mathcal{V}^f \implies \xi(\nu_n) = 0$, hence $b_n = 0 \in \mathit{bounds}(\nu_n)$.

For $i < n$, $b_i = max\{\xi(\nu_i), \xi(\nu_{i+1}), ..., \xi(\nu_n)\} = max(\xi(\nu_i), max\{\xi(\nu_{i+1}), ..., \xi(\nu_n)\}) = max(\xi(\nu_i), b_{i+1})$. By induction hypothesis we know that $b_{i+1} \in \mathit{bounds}(\nu_{i+1})$. Because $\nu_{i+1} \in \nu_i^\bullet$ then by construction $b_i = max(\xi(\nu_i), b_{i+1})$. We have that $\forall i \in [0,n], b_i \in \mathit{bounds}(\nu_i)$ and therefore the states in ρ' are state of the split graph.

Now let us prove that the transitions of ρ' are also in the split graph. $\forall i \in [1,n], (\nu_{i-1}, b_{i-1}) \xrightarrow{e_i} (\nu_i, b_i) \in \mathcal{E}'$ because $\nu_{i-1} \xrightarrow{e_i} \nu_i \in \mathcal{E}$ and $b_i = max(b_{i+1}, \xi(\nu_i))$

Let $\mathcal{G}_l = \langle \Sigma, \mathcal{V}'', \mathcal{E}'', (\mathcal{V}'')^0, (\mathcal{V}'')^F\rangle$ be the given a sub-graph for the explanation of length l.

$\forall i \in [1,n], (\nu_{i-1}, b_{i-1}) \xrightarrow{e_i} (\nu_i, b_i) \in \mathcal{E}'' = \mathcal{E}' \cap (\mathcal{V}'' \times \Sigma \times \mathcal{V}'')$ because $\nu_{i-1} \xrightarrow{e_i} \nu_i \in \mathcal{E}'$ and $b_i \leq l$ by definition of l and b_i.

The run ρ' is in $\Upsilon(\mathcal{G}_l)$ and therefore $w \in \Psi(\mathcal{G}_l)$. □

This result means that any log-consistent violation is contained in some subgraph after split and extraction.

4.1 Further Improvements

4.1.1 Compressing δ-sequences and estimating time delays.

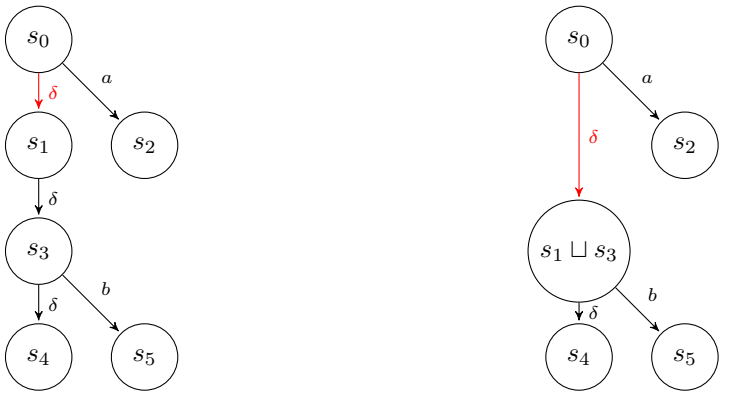
Our explanations may still encompass sequences of discretized time delays whose intermediate states are distinguished by the bisimulation. Our motivation for eliminating such sequences of delays is twofold. First, to construct a more concise explanation. In the case study, compressing δ-sequences allows us to reduce the number of transitions in the explanation from 21 to 7. Second, compressing sequences of time delays allows us to quantitatively estimate the possible time delays of the concrete runs that are summarized by the explanation.

Given a sub-graph $\mathcal{G}_l = \langle \Sigma, \mathcal{V}, \mathcal{E}, \mathcal{V}^0, \mathcal{V}^F\rangle$, a *$\delta^+$-sequence* is an atomic sequence of transitions $\sigma = \nu_1 \xrightarrow{\delta} \nu_2 \xrightarrow{\delta} ... \xrightarrow{\delta} \nu_{n+1}$, that is,

$$\forall i \in \{1, ..., n\}\ \forall e \in \Sigma\ \forall \nu' \in \mathcal{V} : \big(\nu_i \xrightarrow{e} \nu' \implies e = \delta \wedge \nu' = \nu_{i+1}\big)$$

such that $\nu_1 \xrightarrow{\delta} \nu_2$ is an effective choice transition.

As illustrated in Figure 3, we replace each maximal δ^+-sequence σ with a single transition $\nu_1 \xrightarrow{\delta} \nu_{n+1}$. The second condition requires the first transition of the sequence to be effective choice, which ensures the information about effective choice states, used in the sequel, to be preserved.



(a) Effective choice transition in red, a and b are discrete events, the transitions in black are either effective choice or not.

(b) δ^+-compression.

Figure 3: Compression of sequences of discretized time delays.

In order for the explanation to convey quantitative timing information, we estimate, for each abstract time transition $\nu \xrightarrow{\delta} \nu'$, the range of concrete delays represented by the transition, given the location invariants in ν and ν'. For each state s and clock c, there exists the constants $\inf_c^s, \sup_c^s \in \mathbb{N} \cup \{\infty\}$ such that in s, $\inf_c^s \leq c \leq \sup_c^s$. If s' is a time successor of s then we can estimate the delay to be between $\delta_{\text{inf}} = \max_{c \in C}(\inf_c^{s'} - \sup_c^s)$ and $\delta_{\text{sup}} = \min_{c \in C}(\sup_c^{s'} - \inf_c^s)$. In the case where δ_{inf} is negative it is set to 0.

4.1.2 Safe alternatives.

The rationale of our construction of explanations is to highlight the events that contributed to the violation of a safety property. Complementary to this information, and equally crucial for understanding how the property was violated, is the question "how could the outcome have been avoided?". Providing this information is the goal of *safe alternatives*.

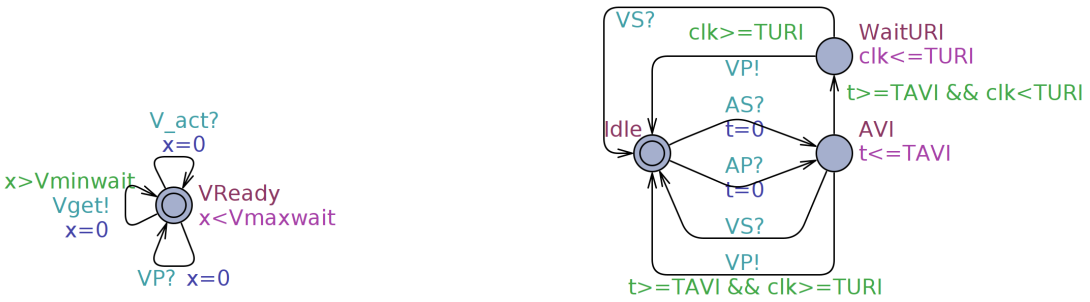
Definition 9 (Safe alternative)**.**

Definition 10 (Safe alternative)**.** *A transition $\nu \xrightarrow{e} \nu'$ of an LTS $\mathcal{G}_l$ is a safe alternative iff ν is an effective choice state and $\xi(\nu') \geq \xi(\nu)$.*

Intuitively, given a choice state, a safe alternative is a transition that would have contributed to avoiding the violation by not decreasing the level of choice.

4.1.3 State constraints.

So far we have focused our attention on the *events* of a failing run. The complementary information crucial for understanding the outcome, is in which *states* some relevant event took place. In this section, let us assume the TA to be equipped with a function $\pi : L \to 2^{\Pi}$ that labels each location with a set of atomic propositions. A straight-forward approach for displaying the states in the explanation would be to compute, for each aggregate state of the determinized graph consisting of a set Q of locations, the disjunction of the invariants (resp. of the atomic proposition) of the locations in L. This would, however, lead to unreadably complex expressions.



(a) Heart ventricular component.

(b) Pacemaker ventricular component *AVI*.

Figure 4: Two components of the model.

We therefore make the design choice to label each state $q = \{\nu_1, ..., \nu_k\} \in Q$ returned by determinization, with a *convex* predicate of the form $\nu_1 \sqcup \cdots \sqcup \nu_k := \hat{C}(q) \wedge SP(q)$, where $\hat{C}(q)$ is the weakest convex clock constraint that is implied by the invariants of the locations of all effective choice states in q. It is straight-forward to compute this clock constraint from the DBMs of the involved location invariants.

Similarly, for the function SP that aggregates, for a state q, atomic propositions of the states in q, multiple definitions are possible. We settle for the conjunction of the atomic propositions that hold in all effective choice states in q. This set is therefore obtained as:

$$SP(q) = \bigcap_{\nu_i \in q:\ \nu_i \text{ is an effective choice state}} \ \bigcap_{\ell \in \nu_i} \pi(\ell)$$

An example of the obtained state predicates is shown in Figure 7.

5 Implementation and Case Study

We have implemented our results in a tool written in Python. It relies on Kronos [23] for the composition of timed automata, Minim for the generation of the quotient graph, and CADP [5] for reductions up to bisimulation.

We illustrate our approach on the dual-chamber implantable pacemaker model of [10]. It is a multi-component system where components are timed automata communicating over channels [16]. We use the model of [13] that we have translated into the Kronos format. The model consists of 5 timed automata for the components of the pacemaker and 2 timed automata that model its environment, that is, the atrial and ventricular behavior of a heart. Whenever delays between sensed atrial or ventricular events exceed a threshold, the pacemaker produces an AP (atricular pace) or VP (ventricular pace) event.

Among the safety properties discussed in [10] we focus on the requirement that the time between two ventricular events (sensed or paced) never exceeds 1000 ms. The safety property observer is shown in Figure 5. If the heart model is safe then the system is also safe. We have therefore modified the parameters of the pacemaker so as to allow for unsafe behaviors.

Figure 4(a) shows the model of the ventricular behavior, with a frequency between $Vminwait = 500$ ms and $Vmaxwait = 1100$ ms, so as to allow a for fault. On the pacemaker model we increase the upper rate interval $TURI$ from 400 ms to 1600 ms. This parameter determines the minimum delay between two ventricular events VP in the AVI component shown in Figure 4(b). In order to compare our results, we have taken the same parameters as in [13].

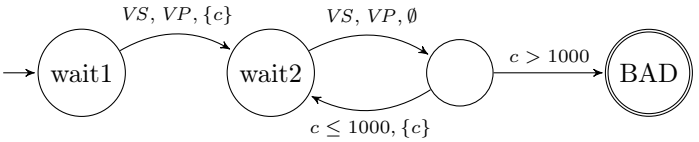


Figure 5: Safety property observer.

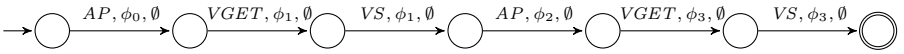


Figure 6: Timed log. All location invariant are *true*, $\phi_0 = (x = 850)$, $\phi_1 = (850 \leq x \leq 1000)$, $\phi_2 = (1700 \leq x \leq 1800)$, $\phi_3 = (1850 \leq x \leq 2200)$.

We fix the set of observable events as the set $\Sigma^{obs} = \{VP, VS, AS, AGET, AP, VGET\}$ of signals exchanged between components, whereas we consider all internal events of the components as unobservable. We use Uppaal to obtain a witness trace for the violation of the property, from whose projection on the observable events we construct the timed log shown in Figure 5.

Let us now apply our approach to explain the causes of the violation. The sizes of the timed automata, discrete abstractions, and explanation are shown in Table 1.

1. The first step is to generate the log observer, which consists of 8 locations and 60 transitions.
2. We invoke Kronos to compose the components, the safety property observer, and the log observer.
3. We use Minim to compute the quotient graph with respect to strong time-abstracting bisimulation.
4. The levels of choice and safe alternatives are computed.
5. We remove the states that are not consistent with the log or the property violation. The difference in size is important because the observed behavior is only a small part of the behavior of the model.
6. In this case study the effective choice states are not ambiguous, therefore splitting does not change the graph. Similarly, the extracted sub-graph amounts to the full graph here.
7. Two maximal δ^+-sequences of length 11 (resp. 6) are found, encompassing 10 (resp. 6) effective choice transitions. After δ^+-compression, the quantitative time delays along the abstract delay transitions (called "time_succ" in our implementation) are computed.
8. After determinization we obtain the explanation shown in Figure 7.

In our case, the explanation is a sequence of discrete and δ-transitions. Each of them moves the system closer to the violation of the safety property, in spite of safe alternatives (shown as transitions without a target state) that would have avoided the violation. In our case the safe alternatives are mostly events sent by the heart model. This is logical because we know that *VGET* from the heart induces immediately a ventricular event in the *PVRP* component (not shown here) that would have avoided the violation. In order to understand why the pacemaker failed to adjust the ventricular events rate, we need to look at the state predicates. If we focus on the last three transitions, we see that the pacemaker waits 150 ms and takes an internal transition

Automaton	#states	#transitions
$\mathcal{A}\|\|\mathcal{P}$	96	296
$\mathcal{L}$	7	6
$\mathcal{L}^c$	8	60
$\mathcal{A}\|\|\mathcal{P}\|\|\mathcal{L}^c$	117	332
$[\mathcal{A}\|\|\mathcal{P}\|\|\mathcal{L}^c]$	1817	5041
$[\mathcal{A}\|\|\mathcal{P}\|\|\mathcal{L}^c]/\sim$	1763	4931
log-consistent	50	52
split	50	49
δ^+-compression	35	34
explanation	8	7

Table 1: Sizes of the timed automata (number of locations), discrete abstractions, and resulting explanation.

I_AVI. If we look at the model of the AVI component, we see that this event could only occur because we increased $TURI$. From this point the violation becomes unavoidable after a further delay of 400 ms.

Comparison with TarTar [13]. We use the same model as in [13], up to the fact that we have increased the parameters *Aminwait* and *Vminwait* (the minimal atrial and ventricular rate of the heart model, respectively) from 1 ms to 500 ms in order to reduce the size of the quotient graph to a size that can be managed by Minim. This modification does not impact the safety violation we are interested in.

TarTar focuses on fixing time delay parameters in order to repair safety violations, and proposes a repair of the bounds on $TURI$. Our approach is more general in the sense that it does not restrict its attention to time delays. On the other hand, it does not propose a repair. In particular, our explanations are useful to explain failures caused by nondeterministic behavior, or when there are no admissible repairs but one still wants to understand the causes of a violation.

6 Discussion

We have proposed a novel approach to explain the violation of a safety property by a real-time system. In essence, an explanation is formed by the parts of the behaviors that are consistent with the log, that jointly contributed to move the system into the failure state.

This work is a first step in constructing explainable cyber-physical systems. Much work remains to be done though towards this long-term goal. In particular, we need to establish formal properties of our explanations, such as soundness and completeness. We will also investigate how to construct explanations online, and how to extend our analysis to more general classes of hybrid systems.

Acknowledgment. The authors thanks Stavros Tripakis and Sergio Yovine for their help in using Kronos.

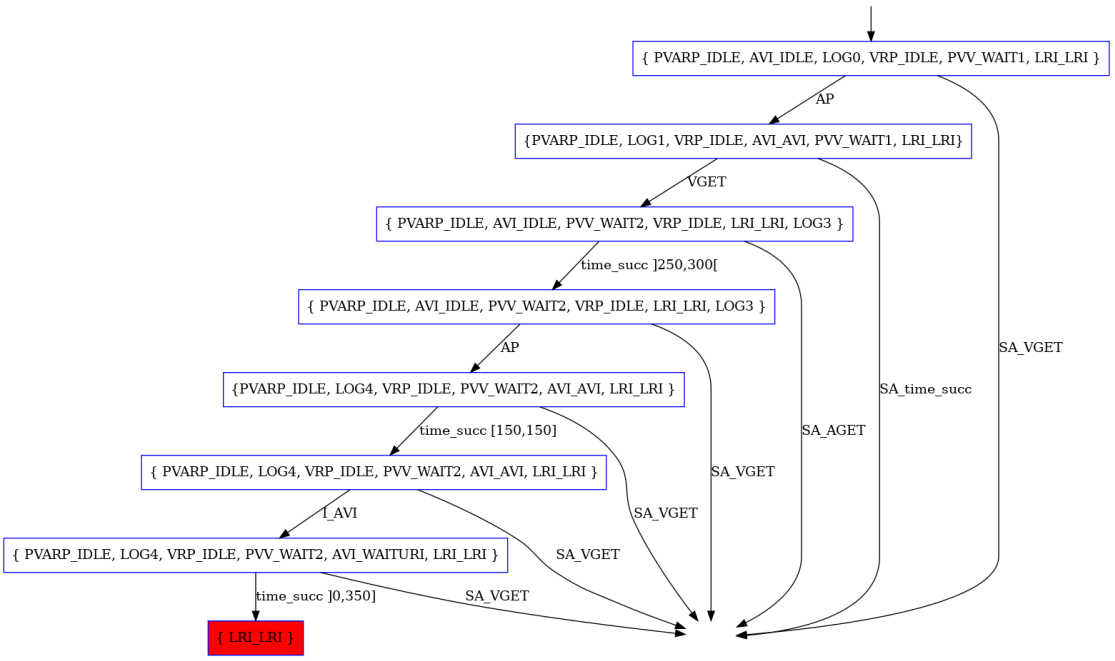


Figure 7: Pacemaker explanation

References

[1] A.V. Aho, R. Sethi, and J.D. Ullman. *Compilers – Principles, Techniques, and Tools*. Addison Wesley, 1986.

[2] S. Beckers, F. Eberhardt, and J.Y. Halpern. Approximate causal abstractions. In Ryan P. Adams and Vibhav Gogate, editors, *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pages 606–615. PMLR, 22–25 Jul 2020.

[3] I. Beer, S. Ben-David, H. Chockler, A. Orni, and R.J. Trefler. Explaining counterexamples using causality. *Formal Methods in System Design*, 40(1):20–40, 2012.

[4] V. Danos, J. Feret, W. Fontana, R. Harmer, J. Hayman, J. Krivine, C. Thompson-Walsh, and G. Winskel. Graphs, Rewriting and Pathway Reconstruction for Rule-Based Models. In D. D'Souza, T. Kavitha, and J. Radhakrishnan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2012)*, volume 18 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 276–288. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2012.

[5] H. Garavel, F. Lang, R. Mateescu, and W. Serwe. CADP 2011: a toolbox for the construction and analysis of distributed processes. *Int. J. Softw. Tools Technol. Transf.*, 15(2):89–107, 2013.

[6] G. Gössler, T. Mari, Y. Pencolé, and L. Travé-Massuyès. Towards Causal Explanations of Property Violations in Discrete Event Systems. In *DX'19 - 30th International Workshop on Principles of Diagnosis*, pages 1–8, November 2019.

[7] G. Gössler and J.-B. Stefani. Causality analysis and fault ascription in component-based systems. *Theoretical Computer Science*, 837:158–180, 2020.

[8] A. Groce, S. Chaki, D. Kroening, and O. Strichman. Error explanation with distance metrics. *STTT*, 8(3):229–247, 2006.

[9] J.Y. Halpern and J. Pearl. Causes and explanations: A structural-model approach. part I: Causes. *British Journal for the Philosophy of Science*, 56(4):843–887, 2005.

[10] Z. Jiang, M. Pajic, S. Moarref, R. Alur, and R. Mangharam. Modeling and verification of a dual chamber implantable pacemaker. In C. Flanagan and B. König, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 188–203, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[11] H. Jin, K. Ravi, and F. Somenzi. Fate and free will in error traces. *STTT*, 6(2):102–116, 2004.

[12] M. Kölbl, S. Leue, and T. Wies. Clock bound repair for timed systems. In I. Dillig and S. Tasiran, editors, *Computer Aided Verification - 31st International Conference, CAV 2019*, volume 11561 of *LNCS*, pages 79–96. Springer, 2019.

[13] M. Kölbl, S. Leue, and T. Wies. Tartar: A timed automata repair tool. In S.K. Lahiri and C. Wang, editors, *Computer Aided Verification - 32nd International Conference, CAV 2020*, volume 12224 of *LNCS*, pages 529–540. Springer, 2020.

[14] B. Korel and J. Laski. Dynamic program slicing. *IPL*, 29(3):155–163, 1988.

[15] R. Küsters, T. Truderung, and A. Vogt. Accountability: definition and relationship to verifiability. In *ACM Conference on Computer and Communications Security*, pages 526–535, 2010.

[16] K.G. Larsen, P. Pettersson, and W. Yi. Uppaal in a Nutshell. *Int. Journal on Software Tools for Technology Transfer*, 1(1–2):134–152, October 1997.

[17] S.A. Mcllraith. *Explanatory diagnosis: Conjecturing actions to explain observations*, pages 155–172. Springer Berlin Heidelberg, 1999.

[18] J. Pearl. Theoretical impediments to machine learning with seven sparks from the causal revolution. In *Proc. Eleventh ACM International Conference on Web Search and Data Mining (WSDM '18)*, pages 3–3. ACM, 2018.

[19] R. Reiter. A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95, 1987.

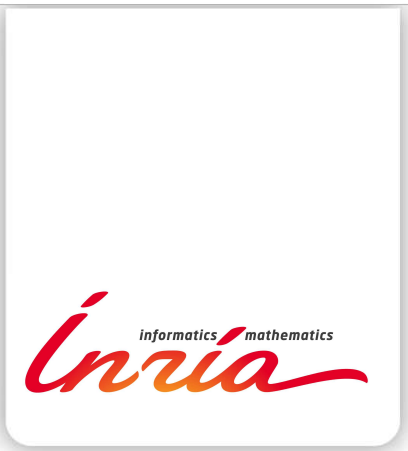
[20] S. Tripakis and S. Yovine. Analysis of timed systems using time-abstracting bisimulations. *Formal Methods Syst. Des.*, 18(1):25–68, 2001.

[21] C. von Essen and B. Jobstmann. Program repair without regret. *Formal Methods in System Design*, 47(1):26–50, 2015.

[22] M. Weiser. Program slicing. *IEEE Transactions on Software Engineering*, 10(4), 7 1984.

[23] S. Yovine. KRONOS: A verification tool for real-time systems. *Software Tools for Technology Transfer*, 1(1+2):123–133, 1997.

[24] A. Zeller. *Why Programs Fail.* Elsevier, 2009.



RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399