Erschienen in: Formal Modeling and Analysis of Timed Systems : 18th International Conference, FORMATS 2020 : Proceedings / Bertrand, Nathalie; Jansen, Nils (Hrsg.). - Cham : Springer, 2020. - (Lecture Notes in Computer Science ; 12288). - S. 127-143. - ISBN 978-3-030-57627-1
https://dx.doi.org/10.1007/978-3-030-57628-8_8

Dynamic Causes for the Violation of Timed Reachability Properties

Martin Kölbl(✉), Stefan Leue(✉), and Robert Schmid(✉)

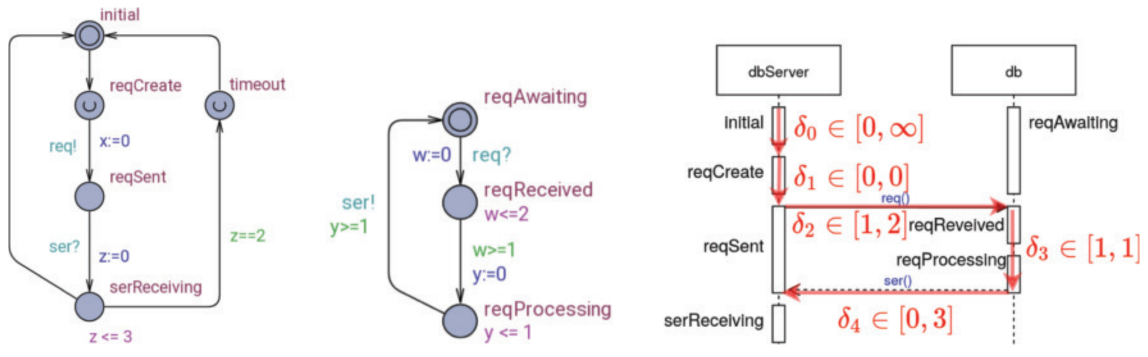
University of Konstanz, Konstanz, Germany
{Martin.Koelbl,Stefan.Leue,Robert.Schmid}@uni-konstanz.de

Abstract. When a real-time model checker detects the violation of a timed reachability property for a given Timed Automata model it returns a counterexample, here referred to as a Timed Diagnostic Trace (TDT). In this paper, we present a TDT analysis that computes actual dynamic causes in terms of delay ranges that can be considered causal for the violation of the property. The determination of actual causes can help in system analysis as well as design space exploration. The causal analysis is based on counterfactual reasoning and encoded in linear real arithmetic. We apply an implementation of the analysis in the tool CaTiRA to a number of Timed Automata models taken from the literature.

1 Introduction

The analysis of causes for the violation of a desired property has various applications in the design of systems. We are particularly interested in developing notions of causality and related analyses for models describing system computations. We have defined Causality Checking in [14] as a means to compute actual causes [10] for the violation of reachability properties, relying on a counterfactual [15] notion of causality inspired by the seminal works of Halpern and Pearl [10].

The actual causes computed in this work rely on choices made during the dynamic execution of the model, for instance, a non-deterministically chosen interleaving of concurrent events during the execution of the model, and we refer to this type of a cause as *dynamic actual causes*. In other work, we have considered the syntactic repair of timed automata models based on an analysis of timed diagnostic traces obtained in real-time model checking [12]. In that work, syntactic features of the model are considered to be actual causes for the violation of timed reachability properties, and we refer to this type of causes as *static actual causes*. Both analyses are based on the counterfactual argument and compare the alternative worlds that differ in the choice of delays or features to find minimal sets of choices that lead to the effect. The analysis of both static and dynamic actual causes can help in identifying possible modifications to design-time models, establishing safety cases for those types of models, or helping in forensic system failure analysis.



(a) TA `client`

(b) TA `db`

(c) TDT illustrated using a time-interval annotated sequence diagram

Fig. 1. Network of Timed Automata - running example

Real-time model checking [3] is a well-established design space exploration technique aiming at analyzing the real-time behavior of a system and its compliance with non-functional real-time requirements. A commonly used model of computation for real-time systems is that of Timed Automata [3]. Timed Automata (TA) describe the real-time behavior of a system in terms of states, labeled with invariant conditions referring to bounds on system clocks, and transition guards, labeled with clock constraints on the enabledness of transitions. Properties of Timed Automata are typically expressed in timed CTL [3]. In this paper, we restrict ourselves to timed reachability properties [3].

It is the objective of this paper to propose the first steps towards a framework of dynamic actual causality in the analysis of timed reachability properties of Timed Automata. We will focus on the question of whether the dynamic timing of the model during system execution can be considered an actual cause, based on a counterfactual argument, of the violation of a timed reachability property. In Timed Automata, the time that the automaton spends in a certain location can be non-deterministically chosen, as long as it complies with the timing constraints specified in the model. The question is then whether there are timing choices that can be considered causal for the violation of a desired timed reachability property.

We now illustrate the idea of our analysis on the time automata of a database request represented in Figure 1. In the model, a `client` sends a request `req` to a database `db` and expects to receive a response `ser` in the location `serReceiving` in less than 4 time units. A clock x is reset with sending the request and measures the time until the response is received with leaving location `serReceiving`. The timed model checker UPPAAL [2] finds a property violation in this model and returns a timed sequence of transitions leading to the property violation in the form of a TDT. A TDT is an alternating sequence of delay transition δ_i, which describe the time that a system remains in state i, and action transitions Θ_i. A TDT for the example in Fig. 1 is $\delta_0, \theta_0 \ldots \delta_3, \theta_3, \delta_4$ with the action transitions

$$\theta_0 = ((\texttt{initial}, \texttt{reqAwaiting}), \emptyset, \tau, \emptyset, (\texttt{reqCreate}, \texttt{reqAwaiting}))$$
$$\theta_1 = ((\texttt{reqCreate}, \texttt{reqAwaiting}), \emptyset, \tau, \{x\}, (\texttt{reqSent}, \texttt{reqReceived}))$$
$$\theta_2 = ((\texttt{reqSent}, \texttt{reqReceived}), \{x \geq 1\}, \tau, \{y\}, (\texttt{reqSent}, \texttt{reqProc.}))$$
$$\theta_3 = ((\texttt{reqSent}, \texttt{reqProc.}), \{y \geq 1\}, \tau, \{z\}, (\texttt{serReceiving}, \texttt{reqAwait.})).$$

A TDT is symbolic in that it describes a set of executions where the time delays before taking the next transition are represented by symbolic variables and constrained by symbolic constraints. The TDT of the database example and the possible time delays are depicted by the sequence diagram in Fig. 1(c). For the remainder of the paper, we use this TDT as a running example. A concrete assignment of the delay values $\delta_0 ... \delta_4$ is a realization, and represents the real-time characteristics of a concrete execution of the TDT S. A realization may, or may not, violate the considered property. An assignment in which the minimal possible values are assigned to all δ_j, in other words, the realization $\delta_0 = 0$, $\delta_1 = 0$, $\delta_2 = 1$, $\delta_3 = 1$ and $\delta_4 = 0$, leads to a trace that does not violate the considered property. Notice that the values of δ_1 and δ_3 are fixed. The values of δ_0, δ_2 and δ_4 in a concrete execution may be determined by environmental effects, or by non-deterministic internal decisions of the two involved subsystems, for instance as a result of task scheduling or memory management. If we consider an alternate execution in which $\delta_4 = 3$, while all other delay values remain as above, then this execution violates the considered property. Some assignments of values to the delays satisfy the property, while others violate the property. This indicates that the cause for the property violation is to be found in the value assignment of certain delays. We are interested in characterizing value assignments to the δ_is that inevitably lead to a property violation using linear constraints. We base the analysis on counterfactual causal reasoning [9] and call such constraints causal ranges. The causal ranges for the example TDT are $2 \leq \delta_4 \leq 3$ and $3 \leq \delta_2 + \delta_4 \leq 5$. It is the objective of this paper to present automated algorithmic ways to compute such causal ranges.

Related Work. Causal reasoning for real-time systems is considered in [7,19] and for reactive systems in [6]. In these three approaches, a system consists of several components and the analysis searches for a causal set of faulty components, whereas we are interested in constraints on a set of causal delays. There is research on system analysis based on counterfactual causal reasoning [10], for instance, in [1,8,10,13]. We are not aware of any work to compute causal time delays for a property violation in a TA.

Structure of the Paper. We exemplify our idea of timed causes in Sect. 2 and discuss in Sect. 3 the foundations of our work. In Sect. 4, we present a formal framework of dynamic trace analysis for causal delays and causal ranges, and present an algorithm to compute causal ranges in Sect. 5. We evaluate an implementation of the algorithm in the tool CaTiRA in Sect. 6 by computing causal ranges for several Timed Automata models. In Sect. 7, we draw conclusions and suggest future developments.

2 A Motivating Example

We motivate our definition of a cause for delays and exemplify our proposed causal analysis. A TDT contains all information regarding possible assignments of the delay variables, in particular the constraints determining possible value ranges for these assignments. As discussed above, these value assignments in a concretization of a TDT determine whether a property is violated.

In keeping with standard practice in engineering science, we will use a counterfactual argument [15] to establish when certain assignments of delay variables constitute a cause for a property violation. This argument says that one phenomenon is a *cause* of another phenomenon, called *effect*, if and only if

I. whenever the cause applies, the effect is observed (regularity argument),
II. when the cause does not apply, the effect will not be observed either (counterfactual argument), and
III. no true subset of the cause ensures I. and II (minimality argument).

In order to establish a causal relation between an assignment of values to the delay variables and the violation of a temporal reachability property, we develop criteria for what we understand to be a cause. For a TA, a dynamic actual cause is a constraint on delay assignments where every assignment that satisfies the cause violates a given property (condition I). Our interpretation of II is that several independent causes can result in a property violation but at least one assignment exists that is not violating the property. III is a minimality argument and removes from a cause any constraint that has no influence on whether an assignment violates the property or not.

Applying this reasoning to the running example from Fig. 1, choosing either $2 \leq \delta_4 \leq 3$ or $3 \leq \delta_2 + \delta_4 \leq 5$, with arbitrary but admissible values assigned to all other δ_js, means that the desired property is violated. Also a different choice of the delay variables value not according to these two expression exists where the property violation cannot be observed. We conclude that, following the counterfactual argument, these two constraint expressions are to be considered independent causes for the property violation.

We now illustrate the computation of a cause in the form of a *causal range* for the running example. For a range expression, to be causal, the values for all δ_js in this range have to violate the considered property (I.). Furthermore, it needs to satisfy the counterfactual argument (II.) which means that there is a different assignment of values to at least one of the δ_js such that this assignment violates the range constraint and does not lead to a property violation. This means that in order to check whether II. is satisfied overall we can test II. on every δ_j in isolation. To illustrate this point, consider the realization $\delta_0 = 1$, $\delta_1 = 0$, $\delta_2 = 1.5$, $\delta_3 = 1$, and $\delta_4 = 1.5$. The realization also violates the considered property with any other value for δ_0, while a decrease of the assigned values of δ_2 or δ_4 results in a realization that satisfies the property. The assignment of δ_0 has no impact on the property since its satisfaction solely depends on the values of δ_2, δ_3 and δ_4. The values of δ_1 and δ_3 are fixed and, hence, they have no admissible alternative assignment. We conclude that only δ_2 and δ_4 have the potential to prevent the violation of the property and, hence, satisfy II. We, therefore, call them causal delay variables. Next, we determine the range in which δ_2 and δ_4 have realizations that violate the considered property.

- A realization with an assignment of δ_4 in the range $[0, 1[$ never violates the property. On the other hand, any realization with an assignment of δ_4 in the range $[2, 3]$ leads to a violation. For every of these realizations, a realization exists that is identical, except for the value of δ_4, that does not violate the property. This means that I. and II. are satisfied and we conclude that the constraint $2 \leq \delta_4 \leq 3$ is a causal range.
- For any realization with an assignment of δ_4 in the range $[1, 2[$, the violation of the property depends on the assignment of the delay variable δ_2. Hence, we analyze which values can be assigned to δ_2 so that this assignment is admissible according to the constraints in the TDT. We then detect that for any admissible assignment of δ_2 in our running example, it conversely depends on the assignment of δ_4 whether a realization violates the property. Thus, δ_2 and δ_4 need to jointly be considered when determining causal ranges. In order to specify the interdependence of δ_2 and δ_4, we consider their sum. We, hence, analyze the range of assigned values for which the sum of δ_2 and δ_4 violates the property. We see that any realization satisfying $3 \leq \delta_2 + \delta_4 \leq 5$ violates the property and a realization not in this range exists that satisfies the property. This means that I. and II. are satisfied and $3 \leq \delta_2 + \delta_4 \leq 5$ is a second causal range.

In the sequel of this paper, we will present an algorithmic way of determining the constraints describing causal ranges.

3 Preliminaries

The Timed Automaton model that we use to represent models of timed systems is adapted from [3]. Given a set of *clocks* C, we denote by $\mathcal{B}(C)$ the finite set of all *clock constraints* over C, which are conjunctions of *atomic clock constraints* of the form $c \sim n$, where $c \in C$, $\sim \in \{<, \leq, =, \geq, >\}$ and $n \in \mathbb{N}_0^+$.

A *Timed Automaton (TA)* T is a tuple $T = (L, l_0, C, \Sigma, \Theta, I)$ where L is a finite set of locations, $l_0 \in L$ is an initial location, C is a finite set of clocks, Σ is a set of action labels, $\Theta \subseteq L \times \mathcal{B}(C) \times \Sigma \times 2^C \times L$ denotes the *transition relation*, and $I : L \to \mathcal{B}(C)$ denotes a labeling of locations with clock constraints, referred to as location invariants. For $\theta \in \Theta$ with $\theta = (l, g, a, r, l')$, we refer to g as the *guard* of θ, to a as the *action label* and to r as its *clock resets*. An urgent location is a location that has to be left again without any delay in time [4]. Urgent locations are syntactic sugar of Uppaal and can be expressed as an additional clock p which is reset with entering the location and a location invariant $p = 0$.

The operational semantics of TAs [3] is given via the definition of *action* and *delay* transitions. Action transitions take the TA from a location l to a location l', execute an action from Σ, reset a subset of the clocks in C while the clock assignments comply with the clock constraints on transition guards and location invariants. Delay transitions only advance the value of all clocks in C by a non-deterministically chosen delay satisfying the invariant condition in the location in which they occur.

The type of properties that we are interested in are time bounded reachability properties, i.e., properties that state that a certain state will (or will not) be reached while a certain clock is satisfying a given bound. When a real-time model checker such as UPPAAL is noticing a violation of such a property, it produces a TDT which we represent symbolically as a *symbolic timed trace* (STT) [12]. A STT is a sequence of actions $S = \theta_0, \ldots, \theta_{n-1}$. A *realization* of S is a sequence of delay values $\delta_0, \ldots, \delta_n$ such that there exist states $s_0, \ldots, s_n, s_{n+1}$ with $s_i \xrightarrow{\delta_i}\xrightarrow{\theta_i} s_{i+1}$ for all $i \in [0, n)$ and $s_n \xrightarrow{\delta_n} s_{n+1}$.

We encode the symbolic semantics of the TDT in linear real arithmetic as a *timed diagnostic trace constraint system* (TDTCS) [12]. A TDTCS $\mathcal{T}$ encodes every transition $\theta_j = (l_j, g_j, a, r_j, l_{j+1})$ in the TDT and is a conjunction of the following constraints:

$$\mathcal{C}_0 \equiv \bigwedge_{c \in C} c_0 = 0 \qquad \text{(clock initialization)}$$

$$\mathcal{A} \equiv \bigwedge_{j \in [0,n]} \delta_j \geq 0 \qquad \text{(time advancement)}$$

$$\mathcal{R} \equiv \bigwedge_{c \in r_j,} c_{j+1} = 0 \qquad \text{(clock resets)}$$

$$\mathcal{D} \equiv \bigwedge_{c \notin r_j} c_{j+1} = c_j + \delta_j \qquad \text{(sojourn time)}$$

$$\mathcal{I} \equiv \bigwedge_{(c \sim \beta) \in I(l_j)} c \sim \beta \wedge c + \delta_j \sim \beta \qquad \text{(location invariants)}$$

$$\mathcal{G} \equiv \bigwedge_{(c \sim \beta) \in g_j} c + \delta_j \sim \beta \qquad \text{(transition guards)}$$

$$\mathcal{L} \equiv @l_n \wedge \bigwedge_{l \neq l_n} \neg @l \qquad \text{(location predicates)}$$

A model satisfies the TDTCS iff the sequence of the delay values $\delta_0, \ldots, \delta_n$ in the model is a realization of the STT [12]. We denote a realization by $\mathcal{T}[\delta_0 \ldots \delta_n]$. A TDTCS is convex since it is a conjunction of constraints [17]. The clock variables are syntactic sugar and can be removed from the TDTCS by replacing all occurrences of a clock variable c_j with $\Sigma_{j' \leq i \leq j} \delta_i$ where $j' = 0$ or the index of the last transition with a reset of clock c before c_j. A partial realization $\delta = \delta_0 \ldots \delta_j$ of $\mathcal{T}$ with $0 \leq j \leq n$ is a realization of a TDTCS $\mathcal{T}_j$, where $\mathcal{T}_j$ encodes only the first j transitions of a given TDT. A suffix blocking partial realization $\delta'_0 \ldots \delta'_j$ is a partial realization that satisfies $\mathcal{T}_j[\delta'_0 \ldots \delta'_j]$ while $\mathcal{T}[\delta'_0 \ldots \delta'_j]$ is unsatisfiable.

We also logically encode a given timed safety property Π as a property constraint system ϕ. The original property Π contains location constraints and time constraints for a clock set C. A location predicate $@l \in \mathcal{L}$ is satisfied when the Timed Automaton is in location l. Let $\phi \equiv \mathit{\Pi}[\mathbf{c}_{n+1}/\mathbf{c}]$ where $\mathit{\Pi}[\mathbf{c}_{n+1}/\mathbf{c}]$ is obtained from property $\mathit{\Pi}$ by substituting the location constraints by location predicates and all occurrences of clocks $c \in C$ by c_{n+1}. This substitution replaces a clock c referred to in property $\mathit{\Pi}$ by the variable referring to the clock c_{n+1} in the final location n of the TDT. The logical encoding of the property in Fig. 1 is $\neg @\mathit{client.serReceiving} \vee x_5 < 4$.

4 Formalizations to Compute Causal Ranges

We first introduce the notion of a *delay set*, which is a subset of the indices of the delay variables $\delta_0, \ldots, \delta_n$ occurring in the STT. For instance, the set $\{\delta_2, \delta_4\}$ is represented by the delay set $\{2, 4\}$. The aim of this section is to define a causal range as a range of values for a given delay set D where any value satisfies the regularity argument (I.), the counterfactual argument (II.) and the minimality argument (III.).

Before we define a causal range, we need to determine a delay set D. Any delay variable δ_j in D shall be causal, thus, a realization δ exists where the value assignment of δ_j matters whether δ violates or satisfies the property. In case δ exists, we call δ_j a *causal delay variable*. Whether δ satisfies or violates the property can interdepend on the value assignment of several causal delay variables as we have seen before. In this case, a realization δ exists such that a different value assignment of any δ_j in D can results in a realization that satisfies the property. We formally define the existence of δ for a delay set D in Definition 1. CV1 in Definition 1 ensures that a realization δ exists for a value v where the sum of the delay assignments with an index in D is equivalent to $v = \sum_{j \in D} \delta_j$. This δ violates a given property ϕ and this satisfies the regularity argument (I.). CV2 ensures II. by requiring the existence of an alternate assignment for every delay variable with an index in D, resulting in a realization δ' that does not violate the property. δ' can also be a suffix blocking partial realization $\delta_0 \ldots \delta_{j-1}\delta'_j$ which cannot be completed to a full realization in a way that would violate the property. For instance, consider a TDT with a guard on a transition that leads to an immediate property violation and where the guard is enabled for an assignment $\delta_j < 2$ and disabled for an assignment $\delta_j \geq 2$. Thus, assigning $\delta_j = 2$ prevents the property violation to be reachable. In conclusion, a causal value satisfies I. and II. for a realization δ, and witnesses that any δ_j in D is a causal delay variable.

Definition 1 (Causal Value). *Assume a TDTCS $\mathcal{T}$ for a TDT of length n, a delay set D of $\mathcal{T}$ and a property constraint system ϕ. A causal value is a value v in a delay set constraint $v = \sum_{j \in D} \delta_j$ where $\delta_j \in \mathbb{R}_0^+$ that satisfies:*

CV1 *There exists a realization $\delta = \delta_0 ... \delta_n$ with delay values $\delta_j \in \mathbb{R}_0^+$ for $0 \leq j \leq n$, $v = \sum_{j \in D} \delta_j$ is satisfied and ϕ is violated.*

CV2 *For every delay value δ_j with $j \in D$, a different delay value δ'_j with $\delta_j \neq \delta'_j$ exists that either $\delta_0 \ldots \delta_{j-1}\delta'_j$ is a suffix blocking partial realization or $\delta_0 \ldots \delta_{j-1}\delta'_j\delta_{j+1} \ldots \delta_n$ satisfies $\mathcal{T}$ and ϕ.*

In the TDT in the example from Fig. 1(c), a causal value of the delay set $\{4\}$ is every value in the range $[1, 3]$, of the delay set $\{2\}$ every value in the range $[1, 2]$ and of the delay set $\{2, 4\}$ is every value in the range $[3, 4]$.

Not all values from a causal range are causal values. For instance, in the TDT in Fig. 1(c), the value $v = 5$ is part of the causal range $3 \leq \delta_2 + \delta_4 \leq 5$, and has a realization with the assignments $\delta_2 = 2$ and $\delta_4 = 3$ that violates the property ϕ, thus satisfying conditions CV1 and I. CV2 requires that an alternative assignment for each of δ_2 and δ_4 exists that prevents the property violation. Such an assignment does not exist for δ_2 but for δ_4 with the assignment $\delta_4 = 0$. Thus, $v = 5$ is not a causal value since it violates CV2, even though it satisfies II. The purpose of a causal value is different from a causal range. A causal value ensures that the assignment of every delay variable in D influences for at least one realization whether the realization violates the property. In difference, a causal range is a cause and ensures that for every realization that satisfies the cause a different assignment exists that satisfies the property.

Next, we define a causal range for a delay set D in Definition 2. A causal range is a constraint of the form $l \sim \sum_{j \in D} \delta_j \sim u$ with a lower bound l and an upper bound u. Every value v in a causal range has to satisfy the causal arguments I and II. Condition CR1 in Definition 2 claims that every value v in the causal range has a realization with $v = \sum_{j \in D} \delta_j$. CR2 ensures the regularity argument I that any realization which satisfies the causal range violates the property. Also, a partial realization $\delta_0 \ldots \delta_m$ can satisfy the causal range constraints when all delays in D are assigned a value $m \geq \max(D)$ where $\max(D)$ returns the maximal value of all elements in D. In order to satisfy causal condition I, this partial realization is not allowed to be a suffix blocking partial realization that prevents the property violation. CR2 refers to all partial realizations. Notice that in particular a partial realization with $m = n$ satisfies $\mathcal{T}$ and violates the property ϕ. We conclude that any realization that satisfies the causal range constraint violates the property, and CR2 actually ensures I. Condition CR3 ensures that a causal value v_c in the range exists that is not a causal value for a true subset of D. We interpret the causal minimality argument III such that we require the number of delay variables that are part of a causal range to be minimal. When v_c is already part of a causal range r' for a true subset of the variables in D, then we conclude that r' is a more concise cause for the violation of property ϕ, thus satisfying III. The existence of v_c, as required by CR3, will be used in the proof of Theorem 1. Condition CR4 claims that the causal range r is maximal in the sense that there is no truly larger range encompassing r which satisfies CR1 to CR3. We include this constraint on the assumption that an analysis result consisting of fewer causal ranges is easier to interpret than one with more causal ranges.

Definition 2 (Causal Range). *Assume a TDTCS $\mathcal{T}$ for a TDT of length n and a property constraint system ϕ. A causal range r is a constraint for a delay set D of the form $l \sim \sum_{j \in D} \delta_j \sim u$ and $\sim \in \{<, \leq\}$, where $l, u \in \mathbb{R}_{\geq 0}$, delay values $\delta_j \in \mathbb{R}_0^+$ and the following conditions hold:*

CR1 *Every value $v \in [l, u]$ has a realization $\delta_0 \ldots \delta_n$ with $v = \sum_{j \in D} \delta_j$.*

CR2 *For every partial realization $\delta_0 \ldots \delta_m$ where $max(D) \leq m \leq n$ and the value $v = \sum_{j \in D} \delta_j$ is in $[l, u]$, there exists a realization $\delta_0 \ldots \delta_m \ldots \delta_n$ that violates ϕ.*

CR3 *At least one value v_c in the causal range is a causal value of D, and v_c is not a causal value for a true subset of D.*

CR4 *The range r is maximal, that means any lower bound $l' < l$ and any higher bound $u' > u$ will not satisfy CR1 to CR3.*

As an example, the constraint $2 \leq \delta_4 \leq 3$ and the constraint $3 \leq \delta_2 + \delta 4 \leq 5$ satisfy the definition of a causal range for the example in Fig. 1(c).

A causal range is only a cause when it satisfies the counterfactual argument (II.) and is ensured by Theorem 1. CR2 ensures that every realization r with value v violates a given property ϕ. The counterfactual argument is satisfied for v when for every r a different value assignment only of the delay variables in D exists such that the resulting realization satisfies ϕ.

Theorem 1. *For every value in a causal range, the counterfactual argument II. is satisfied.*

We have argued above that CR2 ensures the regularity argument I. and CR3 ensures the minimality argument III. In combination with Theorem 1, we conclude that a causal range represents a cause for the property violation according to the definition in Sect. 2.

5 Causal Range Algorithm

We present the `Causal Range Algorithm` to compute a set of causal ranges for a given TDT T and a given property ϕ. The input of the algorithm is a TDTCS $\mathcal{T}$ derived from T and a property constraint ϕ created for the considered property. The output of the algorithm is a set of causal ranges, where any causal range is characterized by a real valued lower bound, a real valued upper bound and a delay set taken from the power set of the TDT delay variables. The algorithm performs the search for causal ranges by solving three satisfiability problems. These problems are depicted in the control flow diagram given in Fig. 2. By solving the problem $P1$ the algorithm starts to iteratively compute the causal delay variables for $\mathcal{T}$. For every computed causal delay variable it creates a delay set. Next, the algorithm computes for every delay set D a causal range by solving the problem $P2$. After the range computation of a delay set D, the algorithm solves problem $P3$ to check whether another causal delay variable δ_k depends on D. In case δ_k exists, the algorithm found a new delay set $D' = D \cup \{k\}$. The algorithm solves $P2$ and $P3$ for every delay set. We encode the problems $P1$ to $P3$ in linear real arithmetic as follows:

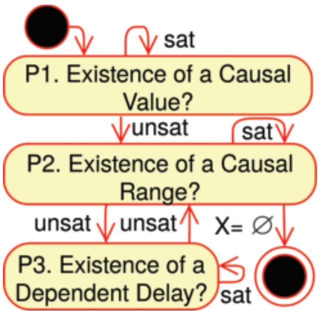

sat
P1. Existence of a Causal Value?
unsat
sat
P2. Existence of a Causal Range?
unsat
unsat
X= ∅
P3. Existence of a Dependent Delay?
sat

Fig. 2. Control flow diagram of Causal Range Algorithm

Problem P1: Existence of a Causal Value. The algorithm first checks for every delay variable δ_j in the TDT whether it is a causal delay variable. A delay variable δ_j is causal when a causal value for $D = \{\delta_j\}$ exists. We encode the conditions that CV1 and CV2 in Definition 1 specify for a causal value in the constraint $\mathcal{C}_j$ (1). The constraint ensures that a realization $\delta = \delta_0, \ldots, \delta_n$ with δ_j and value $v = \delta_j$ exists. δ is a realization when it satisfies $\mathcal{T}$, and has to violate ϕ in order to satisfy CV1 in Definition 1. The constraint $\mathcal{C}_j$ also ensures that CV2 is satisfied. It is satisfiable when an assignment δ'_j that is different from δ_j exists, such that either $\delta' = \delta_0 \ldots \delta_{j-1}\delta'_j$ is a suffix blocking partial realization, or $\delta[\delta_j/\delta'_j]$ is a realization that satisfies ϕ. When δ' is a suffix blocking partial realization, it satisfies $\mathcal{T}^*$ (2). T^* ensures that δ' satisfies the constraint $\mathcal{T}_j$ for a partial realization and no assignment of δ'_{j+1} to δ'_n exists such that $\delta_0 \ldots \delta_{j-1}\delta'_j\delta'_{j+1}, \ldots, \delta'_n$ is a realization that satisfies $\mathcal{T}$.

$$\mathcal{C}_j \equiv (\exists \delta_0, \ldots, \delta_n, \delta'_j)(\mathcal{T} \wedge \neg\phi \wedge (\mathcal{T}^*[\delta_0 \ldots \delta_{j-1}\delta'_j] \vee (\mathcal{T}[\delta_j/\delta'_j] \wedge \phi[\delta_j/\delta'_j]))) \quad (1)$$

$$\mathcal{T}^*[\delta_0 \ldots \delta_{j-1}\delta'_j] \equiv \mathcal{T}_j[\delta_0 \ldots \delta_{j-1}\delta'_j] \wedge \neg(\exists \delta'_{j+1}, \ldots \delta'_n)(\mathcal{T}[\delta_0 \ldots \delta_{j-1}\delta'_j \ldots \delta'_n]) \quad (2)$$

By iteratively determining the satisfiability of the constraint $\mathcal{C}_j$ for all delay variables δ_j occurring in $\mathcal{T}$, the algorithm checks whether these δ_js actually are causal delay variables. In the implementation of the algorithm, we combine all $\mathcal{C}_j$ into one linear constraint system $\mathcal{C}$ of the form $\bigwedge_{0 \leq j \leq n} \neg c_j \vee C_j$ in which we add a fresh Boolean variable c_j for every occurring δ_j. These c_j are defined such that if some $\mathcal{C}_j$ is unsatisfiable, then $\neg c_j$ holds. We use a MaxSMT solver in order to determine the minimum number of $c_j =$ true assertions that need to be violated in order to render $\mathcal{C}$ satisfiable. A delay variable δ_j is causal if and only if c_j is true in the solution to the MaxSMT problem. Subsequently, for every computed causal delay variable δ_j the algorithm adds a delay set $\{\delta_j\}$ to a first-in-first-out queue X. This queue stores every computed delay set and will be handed over to the algorithm addressing problem P2, which computes a causal range for every element of X. For the example in Fig. 1(c), the queue passed on to P2 is $X = \{\{4\}, \{2\}\}$.

Problem P2: Existence of a Causal Range. The algorithm solving problem P2 removes a delay set D from queue X and computes the causal ranges of D. We encode the computation of a causal range as a satisfiability problem. We formalize the conditions CR1 to CR3 in Definition 2 as individual constraints $\mathcal{R}_1$ to $\mathcal{R}_3$. The satisfiability of this conjunction yields an answer to problem P2 and computes causal ranges if they exist. CR1 claims that every value t in the causal range $[l, u]$ has a realization. If $\mathcal{R}_1^b$ is satisfiable, then there exists a realization of $\mathcal{T}$ with a value $b = \sum_{j \in D} \delta_j$. We now check whether $\mathcal{R}_1^b[b/l]$ and $\mathcal{R}_1^b[b/u]$ are satisfiable, respectively. If both are satisfiable, due to the convexity of $\mathcal{T}$ we can conclude that $\mathcal{R}_1^b[b/t]$ is satisfiable for any value $t \in [l, u]$.

$$\mathcal{R}_1^b \equiv (\exists \delta_0 \ldots \delta_n)(\mathcal{T} \wedge b = \sum\nolimits_{j \in D} \delta_j) \quad (3)$$

CR2 claims that for every partial realization $\delta_0 \ldots \delta_j$ which satisfy $l \leq \sum_{j \in D} \delta_j \leq u$ there exists $\delta_0 \ldots \delta_j \delta_{j+1} \ldots \delta_n$ that violates the property ϕ. We formalize CR2 as follows:

$$\mathcal{R}_2 \equiv \bigwedge\nolimits_{max(D) \leq j \leq n} (\forall \delta_0 \ldots \delta_j)(\mathcal{T}_j[\delta_0 \ldots \delta_j] \wedge (l \leq \sum\nolimits_{j \in D} \delta_j \leq u) \\ \Rightarrow (\exists \delta_{j+1} \ldots \delta_n)(\mathcal{T} \wedge \neg\phi)) \quad (4)$$

We project a realization $\delta = \delta_0 \ldots \delta_n$ to a single value v with the formula $v = \sum_{j \in D} \delta_j$. If $v \in [l, u]$ we say that δ is contained in $[l, u]$. Constraint $\mathcal{R}_3$ (5) ensures that a realization δ contained in $[l, u]$ exists and δ is not contained in any causal range $[l_i, u_i]$ of a subset D_i of D. When this δ exists, CR3 is fulfilled. Notice that the subset D_i is not necessarily a true subset of D and therefore ignores previously found causal ranges contained in D. If further causal ranges are contained in D these will also be computed by the algorithm solving P2.

$$\mathcal{R}_3 \equiv \exists \delta_0 \ldots \delta_n . \mathcal{T} \wedge l \leq \sum_{j \in D} \delta_j \leq u \wedge \bigwedge_{D_i \subseteq D} (\sum_{i \in D_i} \delta_i < l_i) \vee (u_i < \sum_{i \in D_i} \delta_i) \quad (5)$$

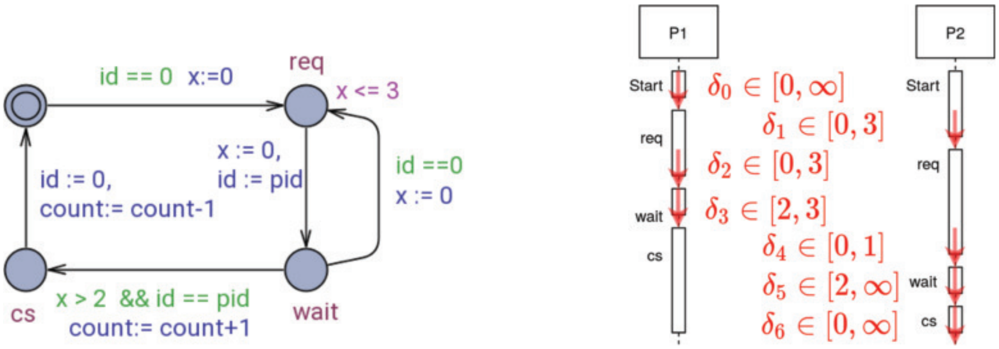
A satisfying assignment for the conjunction of $\mathcal{R}_1^l$, $\mathcal{R}_1^u$, $\mathcal{R}_2$, and $\mathcal{R}_3$ contains a causal range $[l, u]$ which is not necessarily maximal. The algorithm takes advantage of the optimization possibilities of the SMT solver Z3 [16] to minimize l and to maximize u in the conjunction in order to ensure a maximal range (c.f. CR4). In the example of Fig. 1(c) for the delay set $D = \{4\}$, the algorithm computes the causal range $2 \leq \delta_4 \leq 3$. Since no further causal range is found, then the algorithm proceeds with extending D by solving problem P3.

Problem P3: Existence of a Dependent Delay. A delay variable δ_k is dependent on a delay set D when the delay set $D' = D \cup \{k\}$ has a causal value. The algorithm solving P3 extends a delay set D with the index of a delay variable δ_k depending on D. We encode the question whether a causal value for D' exists as the problem whether constraint $\mathcal{C}_k$ (6) is satisfiable. The satisfiability of $\mathcal{C}_k$ yields an answer to problem P3. A satisfying model then yields the dependent delays δ_k, if any exist. $\mathcal{C}_k$ ensures that there exists a realization $\delta = \delta_0 \ldots \delta_n$ with value v that violates the property ϕ. For every index j in the delay set D' a realization exists that differs from δ only in the assignment of δ_j and does not violate ϕ, thus satisfying CV2 in Definition 1.

$$\mathcal{C}_k \equiv \exists \delta_0, \ldots, \delta_n . \mathcal{T} \wedge \neg\phi \wedge \bigwedge_{j \in D'} \exists \delta'_j . \mathcal{T}^*[\delta_0 \ldots \delta_{j-1} \delta'_j] \vee (\mathcal{T}[\delta_j / \delta'_j] \wedge \phi[\delta_j / \delta'_j]) \quad (6)$$

For every causal delay variable δ_k with $k \notin D$ for which $\mathcal{C}_k$ is satisfiable, the algorithm adds $D' = D \cup \{k\}$ to the queue X. To illustrate this step, for the TDT in Fig. 1(c) and the delay set $D = \{4\}$, $\mathcal{C}_k$ is satisfiable for $k = 2$ and the queue $X = \{\{2\}\}$ is extended to $X = \{\{2\}, \{4, 2\}\}$. The algorithm proceeds with removing the next delay set from X, solving problems P2 and P3 for this delay set, and terminates when X is empty.

Correctness of the Algorithm. The theorems below show that the algorithm to compute causal ranges is correct with respect to soundness and completeness.



(a) TA (b) Sequence Diagram

Fig. 3. Fischer's protocol

Theorem 2 (Soundness of Causal Range Computation). *Every causal range returned by the Causal Range Algorithm for a TDTCS $\mathcal{T}$ and a delay set D satisfies CR1 to CR4.*

Theorem 3 (Completeness of Causal Range Computation). *The Causal Range Algorithm returns every maximal causal range for any given TDT.*

6 Evaluation

We implemented the Causal Range Algorithm in a tool that we call *Causal Timed Range Analyser* (CaTiRA) and evaluated the tool by computing causal ranges for several case studies taken from the literature.

Evaluation Methodology. CaTiRA is intended to be used at design time to support design space exploration based on causal information regarding the dynamic timing behavior. We foresee a usage of CaTiRA at an intermediate design stage when a considered preliminary model does not yet satisfy all required properties. The objective of the analysis is to point the designer to variations in the delay timings at certain locations in a TDT, which may motivate changes in, for instance, timing bounds in the model. Such preliminary design models are not available for experimentation. Therefore, it is necessary that we use published, correct models and revert them to a preliminary state by seeding syntactic code variations. We emphasize that the objective is not to locate these code variations, or even propose repairs to syntactic elements, but to illustrate to the designer what ranges of delay variations contribute to avoiding the observed property violation. The designer can then decide to perform syntactic changes to the model in order to constrain the timing of the system in such a way that property violations will be avoided.

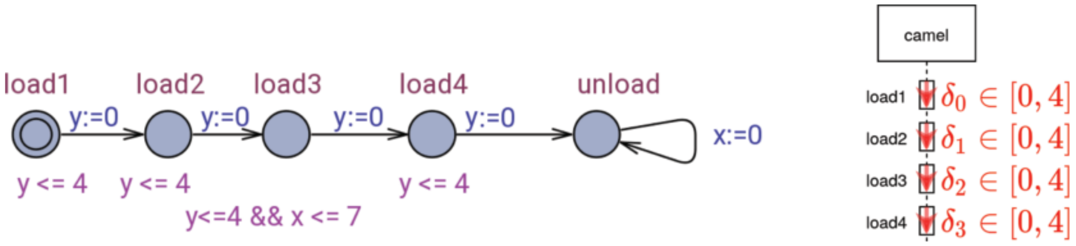
Database Example [12]. For the running example in Fig. 1, the analysis found the causal delays δ_2 and δ_4, and the causal ranges $2 \leq \delta_4 \leq 3$ and $3 \leq \delta_2 + \delta_4 \leq 5$.

Fischer's Protocol [18]*.* The purpose of Fischer's protocol is to ensure mutual exclusion. The TA in Fig. 3 is a process of the protocol with a unique id *pid* and requests for the critical section *cs.* The global variable *id* controls the access to the critical section and allows a process only to enter the critical location *cs* when id = pid. In order to request access to the critical section, a process checks the transition guard $id == 0$ to check that no other process currently requests access to the critical section, and subsequently enters the location *req.* Afterwards the process enters the location *wait* within 3 time units for the critical section and overwrites *id* with its unique *pid.* The process has to wait for 2 time units to ensure that when another process also requests access to the critical section, then this process will have left the location *req.* In this model, the timing is not correct and it is possible that a process enters location *cs* and the other process is in location *req.* We reverted the model to a preliminary state by replacing the invariant $x \leq 2$ as in [18] by $x \leq 3$. We checked mutual exclusion of this model with UPPAAL and obtained a TDT depicted by the sequence diagram in Fig. 3(b).

For this TDT, the causal range analysis by CaTiRA finds the causal delays δ_1 to δ_3, and the causal ranges $2 < \delta_1 \leq 3$, $0 \leq \delta_2 < 1$, $2 < \delta_3 \leq 3$, and $0 \leq \delta_1 + \delta_2 < 1$. Notice that the TDT of Fischer's protocol has no realization that satisfies the property. The causal ranges were computed because the Causal Range Algorithm also considers suffix blocking partial realization to satisfy the counterfactual argument (II.).

The causal ranges with the causal delay variables δ_1, δ_2 and δ_3 are reasonable since during these time delays the TDT is in the location *req*, labeled with the seeded constraint $x \leq 3$. During the time delay δ_4, the TDT is also in location *req.* However, δ_4 is not a causal delay variable since no different delay assignment exists for it that prevents the property violation. The interval of the causal ranges $2 < \delta_1 \leq 3$ and $2 < \delta_3 \leq 3$ is identical to the seeded faulty extension of the constraint. The choice of delay assignments that satisfies $0 \leq \delta_2 < 1$ and $0 \leq \delta_1 + \delta_2 < 1$ ensures that TA *P1* leaves location *req* early enough that the extension of the constraint comes into effect and the property violation will be reached. We see that the causal ranges actually express the choices of delay assignments that will lead to a property violation.

Camel Transporter (Adapted from [5]*).* In this model, in every location *load1* to *load4* a worker loads a bag on a camel. The weight of a bag is between 1 and 4 units and is modeled by the time that the worker stays in a location. The camel will only arrive at the destination when the weight is not more than 7 units. The worker checks the payload of the camel with loading the third bag on the camel but is in a rush and does not check the payload after loading the fourth bag. A verification of the model with UPPAAL results in a TDT depicted in the Fig. 4(b). We manually computed the possible assignments of the delay variables and added them in red to the diagram. For this TDT, CaTiRA computes the causal delays δ_0 to δ_3, and the causal ranges $7 < \delta_0 + \delta_3 \leq 8$, $7 < \delta_1 + \delta_3 \leq 8$, $7 < \delta_2 + \delta_3 \leq 8$, $7 < \delta_0 + \delta_1 + \delta_3 \leq 11$, $7 < \delta_0 + \delta_2 + \delta_3 \leq 11$, $7 < \delta_1 + \delta_2 + \delta_3 \leq 11$, and $7 < \delta_0 + \delta_1 + \delta_2 + \delta_3 \leq 11$.



(a) TA

(b) Sequence Diagram

Fig. 4. Camel transporter (Color figure online)

All causal ranges contains the delay variable δ_3 of the location *load4*. This is reasonable since an overload of the camel is checked in location *load3*. The load limit of 7 can only be exceeded in a combination of at least two delay assignments since the maximal delay assignment is 4 for every delay variable. Also, even two variable assignments can already result in an overload of the camel, every delay assignment has an impact on whether the camel is overloaded. This becomes obvious by the realization $\delta_0 = 2$, $\delta_1 = 2$, $\delta_3 = 2$, $\delta_4 = 2$ that is contained only by the causal range $7 < \delta_0 + \delta_1 + \delta_2 + \delta_3 \leq 11$. In this realization, every delay assignment contributes to the overload of the camel but no subset of the delay assignments can overload the camel. We see that the causal ranges express the dynamic timing behavior that leads to the property violation.

The Pacemaker Model [11] that we consider originally satisfies all properties. We analyze this model since it is a realistic model and of a reasonable size. A modified version of the model, which contains a property violation, is analyzed in [12]. The violated property expresses that the time delay between two ventricular heartbeats is not too high. For the TDT illustrating the property violation, CaTiRA computes the causal delays δ_0 and δ_6, and the causal range $150 < \delta_6 < 350$.

The results can be interpreted as follows. After the time delay of δ_0, the first heartbeat happens and a timer starts to measure the time delay until the next ventricular heartbeat. For some realization of the TDT that violate the property, an increase of the value assignment of δ_0 can prevent the property violation, thus, δ_0 is a causal delay variable. However, no causal range with δ_0 exists since each of these realizations is already contained by the causal range $150.0 < \delta_6 < 350.0$. Only during the time delay δ_6, the TDT is in the location in which a constraint was altered when reverting the model. The modification of the constraint corresponded to an increase of 1200 time units of a bound. However, the possible assignments in this location that lead to a property violation are in the range from $[150, 350]$. Thus, only the increase of the constraint bound by the first 200 time units has an impact on the possible execution. We see that the causal range shows the erroneous timing behavior of the TDT.

Quantitative Results. The quantitative results of every model are represented in Table 1. For every model, we indicate the time T_{UP} that UPPAAL needed to

Table 1. Quantitative experimental results.

Model	T_{UP}	Ln	$\#CD$	$\#CR$	T	M	$\#Cn$	$\#Vr$	T_{Z3}	M_{Z3}
Database [12]	0.010	4	2	2	0.420	13.2	66	19	0.020	6.7
Fischer's protocol [18]	0.010	6	3	4	0.626	26.5	78	23	0.040	6.9
Camel Transporter [5]	0.008	3	4	7	4.724	35.9	138	105	0.350	7.1
Pacemaker [11]	0.015	7	2	1	0.587	33.5	226	114	0.080	7.3

compute a TDT for the model and the length Ln of the TDT. For a given TDT, CaTiRA computes a number $\#CD$ of causal delay variables and a number $\#CR$ of causal ranges. The computation of the causal ranges takes in total a time T in seconds and consumes at most an amount M of memory in megabytes. Z3 solves constraint systems with at most a count $\#Cn$ of clauses and at most a number $\#Vr$ of variables. Z3 needs at most the time T_{Z3} in seconds to solve a constraint system with a maximal memory usage of M_{Z3} in megabytes.

All experiments were performed with the SMT-solver Z3 (Version 4.8.3) on a computer with an i7-6700K CPU (4.00 GHz), 60 GB of RAM and a Linux operating system. For the considered models, we found a total of 11 causal delay variables and 14 causal ranges. The highest computation effort for a causal range computation can be observed in the camel transporter TDT with 35.9 MB memory consumption and 4.724 s computation time. In line with this, Z3 has the highest computation effort in time (0.350 s) with this model. The intrinsic complexity of this TDT seems to be high since with 138 clauses it has fewer clauses than the TDT of the Pacemaker model with 226 clauses.

The most complex model is the Pacemaker model since it takes the most time (0.015) for UPPAAL to compute the TDT. Also, its TDT is the longest with 7 transitions. With 226 the encoding of the analysis has the most clauses and with 114 the most variables. Even so, the computation effort of the causal ranges is moderate with 0.587s and 33.5 MB. In conclusion, the analyses results show that the causal range analysis requires a reasonable computation effort.

7 Conclusion

We have presented the Causal Range Algorithm and its implementation in the tool CaTiRA. Based on a counterfactual causality argument, the Causal Range Algorithm performs an analysis to determine dynamic causes for timed reachability property violations in the timing behavior of a timed system as documented by TDTs. Using various case studies we have shown that the analysis is both efficient and effective. In particular, our work shows that using interpretations of counterfactual causal reasoning can lead to precise and intuitive explanations for dynamic timing behaviors.

In future work, we plan to generalize our findings to the analysis of hybrid systems. Another direction of research is to develop causal analyses that do not just rely on a single execution, as given by a TDT, but on the full structure of a Timed Automaton model.

References

1. Beer, I., Ben-David, S., Chockler, H., Orni, A., Trefler, R.J.: Explaining counterexamples using causality. Formal Methods Syst. Des. **40**(1), 20–40 (2012). https://doi.org/10.1007/s10703-011-0132-2
2. Bengtsson, J., Larsen, K., Larsson, F., Pettersson, P., Yi, W.: UPPAAL—A tool suite for automatic verification of real-time systems. In: Alur, R., Henzinger, T.A., Sontag, E.D. (eds.) HS 1995. LNCS, vol. 1066, pp. 232–243. Springer, Heidelberg (1996). https://doi.org/10.1007/BFb0020949
3. Bengtsson, J., Yi, W.: Timed automata: semantics, algorithms and tools. In: Desel, J., Reisig, W., Rozenberg, G. (eds.) ACPN 2003. LNCS, vol. 3098, pp. 87–124. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27755-2_3
4. Bouyer, P., Fahrenberg, U., Larsen, K.G., Markey, N., Ouaknine, J., Worrell, J.: Model checking real-time systems. In: Clarke, E., Henzinger, T., Veith, H., Bloem, R. (eds.) Handbook of Model Checking, pp. 1001–1046. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-10575-8_29
5. Chapman, A.: Camels, diamonds and counterfactuals: a model for teaching causal reasoning. In: Teaching History, pp. 46–53 (2003)
6. Dimitrova, R., Majumdar, R., Prabhu, V.S.: Causality analysis for concurrent reactive systems (extended abstract). In: CREST@ETAPS, EPTCS, vol. 286, pp. 31–33 (2018)
7. Goessler, G., Astefanoaei, L.: Blaming in component-based real-time systems. In: EMSOFT, pp. 7:1–7:10. ACM (2014)
8. Groce, A., Chaki, S., Kroening, D., Strichman, O.: Error explanation with distance metrics. STTT **8**(3), 229–247 (2006)
9. Halpern, J.Y., Pearl, J.: Causes and explanations: a structural-model approach - part II: explanations. Br. J. Philos. Sci. **56**, 889–911 (2005)
10. Halpern, J.Y., Pearl, J.: Causes and explanations: a structural-model approach. Part I: causes. Br. J. Philos. Sci. **56**, 843–887 (2005)
11. Jiang, Z., Pajic, M., Moarref, S., Alur, R., Mangharam, R.: Modeling and verification of a dual chamber implantable pacemaker. In: Flanagan, C., König, B. (eds.) TACAS 2012. LNCS, vol. 7214, pp. 188–203. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28756-5_14
12. Kölbl, M., Leue, S., Wies, T.: Clock bound repair for timed systems. In: Dillig, I., Tasiran, S. (eds.) CAV 2019. LNCS, vol. 11561, pp. 79–96. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25540-4_5
13. Kumazawa, T., Tamai, T.: Counterexample-based error localization of behavior models. In: Bobaru, M., Havelund, K., Holzmann, G.J., Joshi, R. (eds.) NFM 2011. LNCS, vol. 6617, pp. 222–236. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20398-5_17
14. Leitner-Fischer, F., Leue, S.: Causality checking for complex system models. In: Giacobazzi, R., Berdine, J., Mastroeni, I. (eds.) VMCAI 2013. LNCS, vol. 7737, pp. 248–267. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35873-9_16
15. Lewis, D.: Counterfactuals. Wiley-Blackwell, Hoboken (2001)
16. de Moura, L., Bjørner, N.: Z3: an efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78800-3_24
17. Oppen, D.C.: Complexity, convexity and combinations of theories. Theor. Comput. Sci. **12**, 291–302 (1980)

18. Uppaal: Uppaal benchmarks (2017). http://www.it.uu.se/research/group/darts/uppaal/benchmarks/#benchmarks. Accessed 23 Jan 2020
19. Wang, S., Ayoub, A., Kim, B.G., Gössler, G., Sokolsky, O., Lee, I.: A causality analysis framework for component-based real-time systems. In: Legay, A., Bensalem, S. (eds.) RV 2013. LNCS, vol. 8174, pp. 285–303. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40787-1_17