Software and Systems Modeling
https://doi.org/10.1007/s10270-026-01394-6

SPECIAL SECTION PAPER

Exploiting Assumptions for Effective Monitoring of Real-Time Properties under Partial Observability

Alessandro Cimatti[1] · Thomas M. Grosen[2] · Kim G. Larsen[2] · Stefano Tonetta[1] · Martin Zimmermann[2]

Received: 28 May 2025 / Revised: 28 April 2026 / Accepted: 18 May 2026

Abstract

Runtime verification of temporal properties is essential for ensuring the correctness and reliability of real-time systems, particularly in cyber-physical systems. A significant challenge in this domain is the effective prediction of property failure or success, especially when dealing with partially observable systems. This paper addresses these challenges by developing an Assumption-Based Runtime Verification (ABRV) approach for a continuous real-time setting. Our method exploits assumptions about the system's behavior, specified as Timed Automata, to enable monitors to predict future outcomes and handle unobservable system parts, such as internal faults. Properties to be monitored are specified using Metric Interval Temporal Logic (MITL). The approach also includes formalizing observations with data and time uncertainty using sequences of timed constraints. We present a zone-based online algorithm for computing the monitoring verdict, implemented on top of the UPPAAL tool. Experimental evaluation on proof-of-concept cases demonstrates the approach's feasibility and effectiveness, illustrating how assumptions facilitate earlier verdicts, enable monitoring of properties dependent on unobservable events, and provide insights into scalability.

Keywords Assumption-based Runtime Verification · Real-Time · MITL · Timed Automata

1 Introduction

The problem of monitoring timed properties has gained significant attention due to its crucial role in ensuring the correctness and reliability of real-time systems. The runtime verification of temporal properties over timed sequences of observations is crucial in various applications ranging from cyber-physical systems including autonomous vehicles and beyond. While different solutions for runtime verification of timed temporal properties have been presented [1–5], some challenges remain to be addressed, in particular extending these solutions with prognosis and diagnosis capabilities. More specifically, we are here interested in effectively predicting in advance the failure of properties and in handling partially observable systems.

In the discrete-time setting, these challenges have been addressed with Assumption-Based Runtime Verification (ABRV) [6–9]. ABRV uses assumptions about the behavior of the system to predict the future behavior of the system and to relate observable and non-observable variables. These assumptions can be derived, for example, from models produced during the system design, or from the data collected from the system in operation. Exploiting assumptions, the monitor can anticipate the detection of property failures. Moreover, the specification is no more limited to the interface of black box systems as in traditional runtime verification, but can be extended to constrain also the internal non-observable parts (such as, for example, internal faults).

Thus, in this setting, partial observability means that the monitor may not receive information about the occurrence

Communicated by Antonio Cerone, Alexander Knapp, and Alexandre Madeira.

✉ Thomas M. Grosen
tmgr@cs.aau.dk

Alessandro Cimatti
cimatti@fbk.eu

Kim G. Larsen
kgl@cs.aau.dk

Stefano Tonetta
tonettas@fbk.eu

Martin Zimmermann
mzi@cs.aau.dk

[1] Fondazione Bruno Kessler, Trento, Italy

[2] Aalborg University, Aalborg, Denmark

Published online: 14 July 2026

of some events (e.g., internal faults or signals not exposed at the system interface). The knowledge on the system execution is therefore underspecified: an observation constrains what must have happened, but leaves unobservable events unconstrained. Assumptions are then used to reconstruct and constrain the possible behaviors of the system compatible with the observed events. The assumption alphabet includes observable and unobservable events and, when combined with the observation, generates all executions consistent with what was seen. This allows the monitor to reason about properties that refer to unobservable events and to issue definitive verdicts even when such events were never directly observed.

In ABRV, the output of the monitor has four possible values:

- $\top$ (Satisfied): given the sequence of observations, the system satisfies the specified temporal properties under the given assumption.
- $\bot$ (Violated): this value indicates that the observed behavior of the system violates the specified temporal property, under the given assumption.
- $\times$ (Out-of-model): the observed behavior violates the assumptions, i.e., there is no run of the assumption compatible with the observations.
- ***?*** (Unknown): given the current observations and assumption, it is not possible to determine definitively whether the property is satisfied or violated.

In this work, we enhance ABRV for real-time systems, where:

- *Properties* are expressed in Metric Interval Temporal Logic (MITL).
- *Assumptions* are specified using Timed Automata.
- *Observations* incorporate data and time uncertainty, captured via sequences of timed constraints.

We formally define the semantics of monitoring under assumptions in this setting and prove that our monitoring function satisfies desirable properties: *impartiality*, once a definitive verdict is issued, it remains valid for all future observation extensions; *anticipation*, a definitive verdict is issued as early as possible when sufficient evidence exists; we introduce a partial order on the four verdicts $\{\top, \bot, \times, ?\}$, formalizing how assumptions and observations refine the verdict.

Like in the discrete-time case [9], the assumption allows the monitor to give a $\top$ or $\bot$ verdict even if the property contains future operators and non-observable events. For example, suppose we monitor the MITL property $\varphi = F_{[0,10]}a \wedge G_{[0,20]}\neg b$ (expressing that there is an "a" in the first ten units of time, but no "b" in the first 20 units of time) and we assume that the system satisfies the assumption $\psi = G_{[0,1]}\neg b \wedge G(a \rightarrow G_{[0,10]}\neg b)$ (expressing that there is no "b" in the first unit of time and no "a" is followed by a "b" within ten units of time). Then, the monitor can output a $\top$ verdict even before time 20, for instance at time 10 when "b" is false in the interval [0, 10] and "a" is true at time 10. Further, it can even give the verdict $\top$ if "b" is not observable, e.g., when "a" is true at time 0 and 10.

One of our main contributions is a rich definition of observations that take into account both data and time uncertainty. As in the discrete case [9], the observations are represented by formulas that can capture the uncertainty on data. For example, $\neg a$ means that a is not seen but "b" can be true or false. The approach is further extended to have uncertainty on time, taking into account potential errors in the timestamps with which the monitor receives data from the system. This is represented in the observations with time intervals that are associated to observation formulas. Thus, for example, we can say that "a" is seen in the interval [6, 7] but we do not know exactly when. Also, we support multiplicities to express bounds on the number of occurrences of events. Finally, we concatenate triples of formulas, time intervals, and multiplicities to form complex sequences of observations. For example, the sequence

$$(a, [0, 0], =1)\,(\neg a, [0, 7], \geq 0)\,(a, [6, 7], =1)\,(\neg a, [6, 16], \geq 0)\,(a, [15, 16], =1)$$

says that we see three a's, one at time 0, another in the interval [6, 7], and a final one in the interval [15, 16] and that we do not know anything about "b" in between the three "a" (intuitively, an observation with an $= 1$ (≥ 0) indicates exactly one occurrence (zero or more occurrences)). If the system satisfies the assumption ψ from above, we can conclude at time 16 that the property φ is true despite the uncertainty about time and b.

We propose a *zone-based online* algorithm that at any time provides a monitoring verdict saying if the property is satisfied or violated given the assumption and a sequence of observations. We implemented the algorithm on top of UPPAAL and show the feasibility of the approach. Especially, we demonstrate how the assumptions can be effective in anticipating the satisfaction/violation of timed properties and in handling properties that predicate over unobservable events. We also report on the influence of unobservable events on the response time, the time it takes to compute a verdict when given a new observation.

This paper significantly extends the work presented in [10] by providing more formal theoretical underpinnings, e.g., the results in Subsection 3.2 showing that our definition is anticipatory, impartial, and monotone in all parameters, as well as a new consistency check for observations (Lemma 3). Furthermore, it enhances the definition and handling of observations by incorporating a richer set of multiplicities, i.e., from

"exactly one" and "any number" to "exactly k", "at least k", and "at most k" for every $k \in \mathbb{N}$. Among other things, this now allows to express upper bounds. Similarly, the original monitoring algorithm has been extended to support such multiplicities. Finally, we present an additional case study in Subsection 5.3 along with more detailed experimental results for the previous case studies.

The remainder of the paper is organized as follows. Section 2 introduces the necessary preliminaries, including timed words, Timed Automata, and MITL. Section 3 defines the monitoring problem under assumptions and presents the formal semantics, including proofs of impartiality, anticipation, and the partial order on verdicts. Section 4 develops a zone-based algorithm for monitoring under assumptions, handling uncertainty in both data and time. Section 5 describes our implementation built on UPPAAL and presents experimental results that demonstrate feasibility and the benefits of assumption-based monitoring. Section 6 discusses related work, and Section 7 concludes with future research directions.

2 Preliminaries

The set of natural numbers (excluding zero) is $\mathbb{N}$, we define $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$, the set of non-negative rational numbers is $\mathbb{Q}_{\geq 0}$, and the set of non-negative real numbers is $\mathbb{R}_{\geq 0}$. The powerset of a set S is denoted by 2^S.

2.1 Timed Words

A timed word over a finite alphabet Σ is a pair $\rho = (\sigma, \tau)$ where σ is a word over Σ and τ is a sequence of non-decreasing non-negative real numbers of the same length as σ. Timed words may be finite or infinite. In the latter case, we require $\limsup \tau = \infty$, i.e., time diverges. The set of finite timed words is denoted by $T\Sigma^*$ and the set of infinite timed words by $T\Sigma^\omega$. We also represent a timed word as a sequence of pairs $(\sigma_1, \tau_1)(\sigma_2, \tau_2)\ldots$. If $\rho = (\sigma_1, \tau_1)(\sigma_2, \tau_2)\cdots(\sigma_n, \tau_n)$ is a finite timed word, we denote by $\tau(\rho)$ the total time duration of ρ, i.e., τ_n (with the convention $\tau(\varepsilon) = 0$). We lift this definition to languages $L \subseteq T\Sigma^*$ by defining $\tau(L) = \sup_{\rho \in L} \tau(\rho)$, which can be infinite.

If $\rho_1 = (\sigma_1^1, \tau_1^1)(\sigma_2^1, \tau_2^1)\ldots(\sigma_n^1, \tau_n^1)$ is a finite timed word, $\rho_2 = (\sigma_1^2, \tau_1^2)(\sigma_2^2, \tau_2^2)\ldots$ a finite or infinite timed word, and $t \in \mathbb{R}_{\geq 0}$ then the concatenation $\rho_1 \cdot_t \rho_2$ is defined if and only if $t \geq \tau(\rho_1)$. Then, we define $\rho_1 \cdot_t \rho_2 = (\sigma_1, \tau_1)(\sigma_2, \tau_2)\cdots$ such that

$$\sigma_i = \begin{cases} \sigma_i^1 & \text{if } i \leq n \\ \sigma_{i-n}^2 & \text{else} \end{cases} \quad \text{and} \quad \tau_i = \begin{cases} \tau_i^1 & \text{if } i \leq n \\ \tau_{i-n}^2 + t & \text{else.} \end{cases}$$

We lift this definition to sets $L_1 \subseteq T\Sigma^*$ and $L_2 \subseteq T\Sigma^* \cup T\Sigma^\omega$ via

$$L_1 \cdot_t L_2 = \{\rho_1 \cdot_t \rho_2 \mid \rho_1 \in L_1 \text{ and } \rho_2 \in L_2\},$$

provided we have $t \geq \tau(L_1)$.

Remark 1 The following two properties, which follow directly from the definition of concatenation, will be useful later on.[1] Let $L, L' \subseteq T\Sigma^*$.

1. If $L \subseteq L'$ and $\tau(L), \tau(L') \leq t$, then $L \cdot_t T\Sigma^\omega \subseteq L' \cdot_t T\Sigma^\omega$.
2. If $\tau(L) \leq t \leq t'$, then $L \cdot_{t'} T\Sigma^\omega \subseteq L \cdot_t T\Sigma^\omega$.

2.2 Timed Automata

A timed Büchi automaton (TBA) $\mathcal{B} = (Q, Q_0, \Sigma, C, \Delta, \mathcal{F})$ consists of a finite alphabet Σ, a finite set Q of locations, a set $Q_0 \subseteq Q$ of initial locations, a finite set C of clocks, a finite set $\Delta \subseteq Q \times Q \times \Sigma \times 2^C \times G(C)$ of transitions with $G(C)$ being the set of clock constraints over C, and a set $\mathcal{F} \subseteq Q$ of accepting locations. A transition (q, q', a, λ, g) is an edge from q to q' on input symbol a, where λ is the set of clocks to reset and g is a clock constraint over C. A clock constraint is a conjunction of atomic constraints of the form $c \sim n$, where c is a clock, $n \in \mathbb{N}_0$, and $\sim \in \{<, \leq, =, \geq, >\}$.

A state of $\mathcal{B}$ is a pair (q, v) where q is a location in Q and $v\colon C \to \mathbb{R}_{\geq 0}$ is a valuation mapping clocks to their values. For any $d \in \mathbb{R}_{\geq 0}$, $v + d$ is the valuation $x \mapsto v(x) + d$. A run of $\mathcal{B}$ from a state (q_0, v_0) over a timed word $(\sigma_1, \tau_1)(\sigma_2, \tau_2)\cdots$ is a sequence

$$(q_0, v_0) \xrightarrow{(\sigma_1, \tau_1)} (q_1, v_1) \xrightarrow{(\sigma_2, \tau_2)} (q_2, v_2) \xrightarrow{(\sigma_3, \tau_3)} \cdots$$

where for all $i \geq 1$ there is a transition $(q_{i-1}, q_i, \sigma_i, \lambda_i, g_i)$ such that $v_i(c) = 0$ for all c in λ_i and $v_i(c) = v_{i-1}(c) + (\tau_i - \tau_{i-1})$ otherwise, and g_i is satisfied by the valuation $v_{i-1} + (\tau_i - \tau_{i-1})$. Here, we use $\tau_0 = 0$. Given a run r, we denote the set of locations visited infinitely many times by r as $\mathrm{Inf}(r)$. A run r of $\mathcal{B}$ is accepting if $\mathrm{Inf}(r) \cap \mathcal{F} \neq \emptyset$. The language of $\mathcal{B}$ from a starting state (q, v), denoted $L(\mathcal{B}, (q, v))$, is the set of all timed words with an accepting run in $\mathcal{B}$ starting from (q, v). We define the language of $\mathcal{B}$, written $L(\mathcal{B})$, to be $\bigcup_q L(\mathcal{B}, (q, v_0))$, where q ranges over all locations in Q_0 and where $v_0(c) = 0$ for all $c \in C$.

Proposition 1 ([11]) *For all TBA $\mathcal{B}, \mathcal{B}'$ there is a TBA $\mathcal{B} \otimes \mathcal{B}'$ with $L(\mathcal{B} \otimes \mathcal{B}') = L(\mathcal{B}) \cap L(\mathcal{B}')$. The set of states of $\mathcal{B} \otimes \mathcal{B}'$ is $Q \times Q' \times \{0, 1\}$, where Q and Q' are the sets of states of $\mathcal{B}$ and $\mathcal{B}'$, respectively.*

[1] Note that we could formulate the properties more generally by concatenating with languages other than $T\Sigma^\omega$, but we refrain from doing so, as this is the result we need later on.

2.3 Logic

We use Metric Temporal Interval Logic (MITL) to formally express properties to be monitored in our examples. These can be translated into equivalent TBA which we use in our monitoring algorithm.

The syntax of MITL formulas over a finite alphabet Σ is defined as

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid X_I\varphi \mid \varphi\, U_I \varphi,$$

where $p \in \Sigma$ and I ranges over non-singular intervals over $\mathbb{R}_{\geq 0}$ with endpoints in $\mathbb{N}_0 \cup \{\infty\}$. Note that we often write $\sim n$ for $I = \{d \in \mathbb{R} \mid d \sim n\}$ where $\sim \in \{<, \leq, \geq, >\}$, and $n \in \mathbb{N}$. We also define the standard syntactic sugar $\texttt{true} = p \vee \neg p$, $\texttt{false} = \neg\texttt{true}$, $\varphi \wedge \psi = \neg(\neg\varphi \vee \neg\psi)$, $\varphi \rightarrow \psi = \neg\varphi \vee \psi$, $F_I\varphi = \texttt{true}\, U_I \varphi$, and $G_I\varphi = \neg F_I \neg\varphi$. Finally, we often omit the interval $[0, \infty]$ for readability.

The semantics of MITL is defined over infinite timed words. Given such a timed word $\rho = (\sigma_1, \tau_1)(\sigma_2, \tau_2)\cdots \in T\Sigma^\omega$, a position $i \geq 1$, and an MITL formula φ, we inductively define the satisfaction relation $\rho, i \models \varphi$ as follows:

- $\rho, i \models p$ if and only if $p = \sigma_i$.
- $\rho, i \models \neg\varphi$ if and only if $\rho, i \not\models \varphi$.
- $\rho, i \models \varphi \vee \psi$ if $\rho, i \models \varphi$ or $\rho, i \models \psi$.
- $\rho, i \models X_I\varphi$ if and only if $\rho, (i+1) \models \varphi$ and $\tau_{i+1} - \tau_i \in I$.
- $\rho, i \models \varphi\, U_I \psi$ if and only if there exists $k \geq i$ such that $\rho, k \models \psi$, $\tau_k - \tau_i \in I$, and $\rho, j \models \varphi$ for all $i \leq j < k$.

We write $\rho \models \varphi$ whenever $\rho, 1 \models \varphi$, and say that ρ satisfies φ. The language $L(\varphi)$ of an MITL formula φ is the set of all infinite timed words that satisfy φ.

Theorem 1 ([12, 13]) *For each MITL formula φ there exists a TBA $\mathcal{B}$ such that $L(\varphi) = L(\mathcal{B})$.*

Example 1 Figure 1 illustrates the above theorem by providing TBA for the formula $s \wedge F_{[0,10]}a \wedge G_{[0,20]}\neg b$ and its negation. The formula requires the first symbol to be an "s", that an a occurs within ten time units after the first symbol, and that no b occurs within twenty time units after the first symbol, i.e., the "s" marks the initial timestamp of the processed word.

In the following, for the sake of readability, we use φ also to denote properties, i.e., subsets of $T\Sigma^\omega$, and use MITL only to specify properties in examples.

3 Monitoring under Assumptions

Monitoring timed properties [1, 5] requires to determine whether every extension of a finite observation (a finite timed word) satisfies a given property (yielding the verdict $\top$), whether every extension violates the property (yielding the verdict $\bot$), or neither is true (yielding the verdict ***?***). In this section we introduce monitoring of real-time properties under assumptions and partial observability.

We begin in Subsection 3.1 with an informal description of monitoring of real-time properties under assumptions and partial observability. Then in Subsection 3.2 we present an abstract definition capturing this description, and prove some basic properties about it. This abstract definition employs arbitrary timed languages for observations, assumptions, and specifications, and is therefore not algorithmically feasible. Finally in Subsection 3.3 we present an effective concretisation of the abstract definition, using timed automata and finite expressions to represent assumptions, specifications, and observations, respectively.

3.1 Motivation

Monitoring under assumptions involves two changes over the classical monitoring framework.

Firstly, the assumption itself: In its most general form, it is a set $A \subseteq T\Sigma^\omega$ of infinite timed words. Intuitively, A contains the executions we assume to be possibly generated by the system we are monitoring. Hence, every execution that is not in A does not need to be taken into account when determining a verdict, i.e., the assumption refines verdicts. However, this also means that our assumption can be invalidated if we observe an execution prefix that is not consistent with our assumption. This requires a new verdict, denoted by $\times$. In this case, the assumption needs to be refined as it does not match our observation.

Secondly, we allow inexact observations: In the classical setting, we observe a finite timed word $(\sigma_1, \tau_1)\cdots(\sigma_n, \tau_n)$ and reason about its possible extensions. Hence, we implicitly presume that no other events occurred between time 0 and τ_n and that the time points are exact. In the following, we allow for some imperfect information about the observation. In the most general form, an observation is then a set $O \subseteq T\Sigma^*$ of finite timed words. Intuitively, O contains those words that are consistent with our (imperfect) observation.

Example 2 Consider the property "$s \wedge F_{[0,10]}a \wedge G_{[0,20]}\neg b$" of Example 1. Monitoring this property on a timed word starting with "s" at time 0 provides a conclusive verdict in the following cases:

- The property is false at any time in the interval $[0, 20]$ if a "b" is observed.
- The property is false after time 10 if "a" was not previously observed.

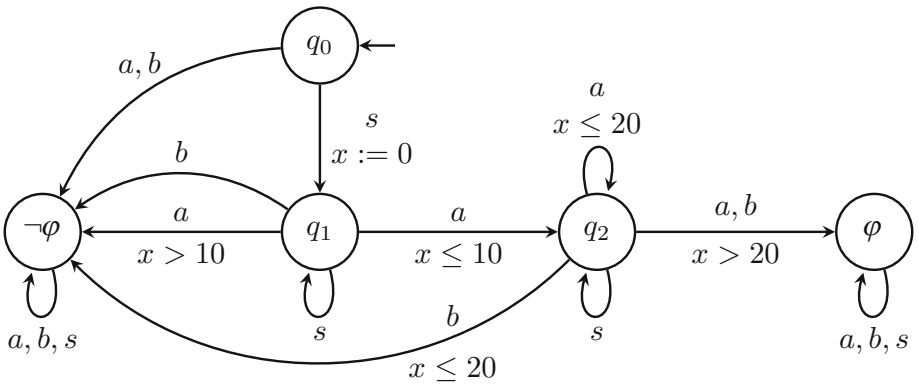


Fig. 1 A TBA for the languages of the formula $\phi = s \wedge F_{[0,10]}a \wedge G_{[0,20]}\neg b$ and its negation: If location φ ($\neg\varphi$) is accepting then it accepts $L(\varphi)$ ($L(\neg\varphi)$)

- The property is true after time 20 if "b" was not previously observed and "a" was observed in the interval [0, 10].

Note that a positive verdict can only given after time point 20, as any "b" before or at time point 20 implies that the property is violated.

Consider now the assumption "$s \wedge XG\neg s \wedge G_{[0,1]}\neg b \wedge G(a \rightarrow G_{[0,10]}\neg b)$" which corresponds to the TBA in Figure 2. Then, if "s" was observed at time 0 and "a" was observed in the interval [0, 10], as soon as we see another "a" at some time point $t \in [10, 20]$ and have not observed a "b" before t, we can conclude that the property is true already at time point t, which may be strictly smaller than 20. In this case, the assumption leads to earlier verdicts.

On the other hand, if after the two "a", a "b" is observed in the interval $[t, t + 10]$, then the observation violates the assumption, even if the property may be satisfied (i.e., if the "b" is observed in the interval $(20, t + 10]$)

Example 3 Let us consider again the property "$s \wedge F_{[0,10]}a \wedge G_{[0,20]}\neg b$" but now we observe "$a$" (possibly) with uncertainty on the timestamps and "b" is unobservable. For example, after the initial "s", we observe "a" at time 0, another time in the interval [6, 7] and a final time in the interval [15, 16], but no other s, and now is time 30. The words that are consistent with these observations have the form $(s, 0)\rho_0(a, 0)\rho_1(a, t_1)\rho_2(a, t_2)\rho_3$ where

- $t_1 \in [6, 7]$ and $t_2 \in [15, 16]$,
- ρ_0 is a (possibly empty) finite timed word $(b, 0) \cdots (b, 0)$,
- ρ_1 is a (possibly empty) finite timed word $(b, t_{1,1}) \cdots (b, t_{1,n_1})$ with $t_{1,j} \in [0, t_1]$ for all $1 \leq j \leq n_1$,
- ρ_2 is a (possibly empty) finite timed word $(b, t_{2,1}) \cdots (b, t_{2,n_2})$ with $t_{2,j} \in [t_1, t_2]$ for all $1 \leq j \leq n_2$, and
- ρ_3 is a (possibly empty) finite timed word $(b, t_{3,1}) \cdots (b, t_{3,n_3})$ with $t_{3,j} \in [t_2, 30]$ for all $1 \leq j \leq n_3$.

Without assumptions we cannot have any conclusive verdict, because we do not know if a "b" occurred before time point 20 or not. But with the assumption from the previous example, we can conclude at time 16 that the property is true:

- ρ_0 must be empty, as there cannot be a "b" within the first unit of time.
- ρ_1 must be empty, as there cannot be a "b" for ten units of time after the "a" at time point 0 and $t_1 \leq 7 \leq 10$.
- ρ_2 must be empty, as there cannot be a "b" for ten units of time after the "a" at time point t_1 and $t_2 \leq 16 \leq t_1 + 10$.
- ρ_3 cannot contain a "b" with timestamp $t_{3,j} \leq 20$, as this would imply that a "b" has occurred less than ten units of time after the "a" at t_2.

Thus, under the assumption, we can make a definitive verdict, which we could not without the assumption.

3.2 Abstract Definition and Basic Properties

In the following, we formalize this intuition. To develop the theory as general as possible, we allow real time points in the observations. Later, when we are concerned with algorithms, we will restrict ourselves to rational inputs. In the same spirit, we begin with a very abstract definition of monitoring under assumptions. Later, we will explain how to represent the property, the assumption, and the observation finitely.

Definition 1 Let $\mathbb{B}_4 = \{\top, \bot, ?, \times\}$. Given a property $\varphi \subseteq T\Sigma^\omega$ of infinite timed words, an assumption $A \subseteq T\Sigma^\omega$, an observation $O \subseteq T\Sigma^*$, and a current time point $t \geq \tau(O)$, the function $\mathcal{V}\colon (2^{T\Sigma^\omega} \times 2^{T\Sigma^\omega}) \rightarrow (2^{T\Sigma^*} \times \mathbb{R}_{\geq 0}) \rightarrow \mathbb{B}_4$ evaluates to a verdict with the following definition:

$$\mathcal{V}(\varphi, A)(O, t) = \begin{cases} \times & \text{if } O \cdot_t T\Sigma^\omega \cap A = \emptyset, \\ \top & \text{if } O \cdot_t T\Sigma^\omega \cap A \neq \emptyset \text{ and } O \cdot_t T\Sigma^\omega \cap A \subseteq \varphi, \\ \bot & \text{if } O \cdot_t T\Sigma^\omega \cap A \neq \emptyset \text{ and } O \cdot_t T\Sigma^\omega \cap A \subseteq T\Sigma^\omega \setminus \varphi, \\ ? & \text{otherwise.} \end{cases}$$

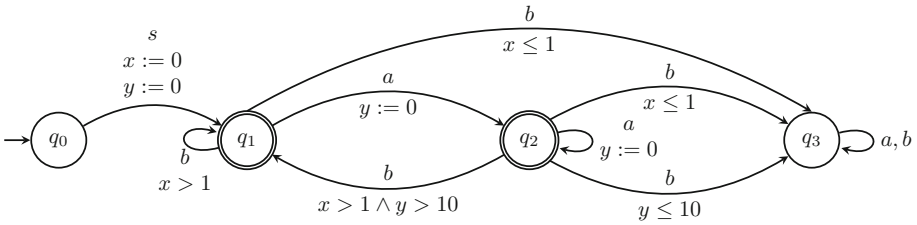


Fig. 2 A TBA for the language of the formula $s \wedge XG\neg s \wedge G_{[0,1]}\neg b \wedge G(a \rightarrow G_{[0,10]}\neg b)$ with accepting locations q_1 and q_2

$\mathcal{V}(\varphi, A)(O, t)$ is undefined when $t < \tau(O)$.

Before we study how to specify assumptions and observations, and how to compute $\mathcal{V}$, we study some properties of $\mathcal{V}$. First of all, our definition satisfies the two maxims of "impartiality" and "anticipation" [14]: Impartiality requires that if an observation yields a definitive verdict (i.e., $\top$, $\bot$, or $\times$ in our case), then every continuation of the observation also yields the same verdict. Note that impartiality is satisfied by always giving the verdict **?**. Hence, anticipation requires that a definitive verdict is given as soon as possible.

To formalize this in the setting of inexact observations, we need to define what it means for an observation to extend another one. Let $O, O' \subseteq T\Sigma^*$ and $t \geq \tau(O)$. We say that O' extends O at t, written $O \sqsubseteq_t O'$ if

- for every $o \in O$ there exists an $o^* \in T\Sigma^*$ such that $o \cdot_t o^* \in O'$ and
- for every $o' \in O'$ there exists an $o \in O$ and an $o^* \in T\Sigma^*$ such that $o' = o \cdot_t o^*$.

Also note that extending an observation that is consistent with an assumption A might lead to an observation that is no longer consistent with A, which implies that a (definitive) verdict $\top$ or $\bot$ may turn into the definitive verdict $\times$ when an observation is extended. This has to be taken into account when defining impartiality for monitoring under assumptions.

Lemma 1 *Fix a property $\varphi \subseteq T\Sigma^\omega$ and an assumption $A \subseteq T\Sigma^\omega$.*

1. *$\mathcal{V}$ is impartial, i.e., $\mathcal{V}(\varphi, A)(O, t) \in \{\top, \bot, \times\}$ implies $\mathcal{V}(\varphi, A)(O', t') \in \{\mathcal{V}(\varphi, A)(O, t), \times\}$ for all $O \sqsubseteq_t O'$, $t \geq \tau(O)$, and $t' \geq \tau(O')$, i.e., a definitive verdict is stable under further observations, as long as the observations are consistent with the assumption.*
2. *$\mathcal{V}$ is anticipatory, i.e., $\mathcal{V}(\varphi, A)(O, t) = ?$ implies that $O \cdot_t T\Sigma^\omega \cap A \cap \varphi$ and $O \cdot_t T\Sigma^\omega \cap A \cap T\Sigma \setminus \varphi$ are both nonempty, i.e., the observation O is consistent with the assumption and satisfaction of φ and the observation O is consistent with the assumption and the violation of φ.*

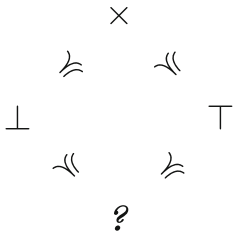


Fig. 3 The (partial) specificity order $\preccurlyeq$ on $\mathbb{B}_4$

Proof 1.) If $O \sqsubseteq_t O'$, $t \geq \tau(O)$, and $t' \geq \tau(O')$, then $O \cdot_t T\Sigma^\omega \supseteq O' \cdot_{t'} T\Sigma^\omega$. We will apply this fact in each of the cases of the following case distinction.

- If $\mathcal{V}(\varphi, A)(O, t) = \times$, then we have $O \cdot_t T\Sigma^\omega \cap A = \emptyset$ by definition. Hence, we obtain

  $$O' \cdot_{t'} T\Sigma^\omega \cap A \subseteq O \cdot_t T\Sigma^\omega \cap A = \emptyset,$$

  which implies $\mathcal{V}(\varphi, A)(O', t') = \times$, as required.
- If $\mathcal{V}(\varphi, A)(O, t) = \top$, then we have $O \cdot_t T\Sigma^\omega \cap A \neq \emptyset$ and $O \cdot_t T\Sigma^\omega \cap A \subseteq \varphi$ by definition. Now, if $O' \cdot_{t'} T\Sigma^\omega \cap A = \emptyset$, then we obtain $\mathcal{V}(\varphi, A)(O', t') = \times$, i.e., the observation O was extended to O' in a way that is not longer consistent with the assumption A. On the other hand, if $O' \cdot_{t'} T\Sigma^\omega \cap A \neq \emptyset$, then

  $$O' \cdot_{t'} T\Sigma^\omega \cap A \subseteq O \cdot_t T\Sigma^\omega \cap A \subseteq \varphi,$$

  which implies $\mathcal{V}(\varphi, A)(O', t') = \top$, as required.
- The reasoning for $\mathcal{V}(\varphi, A)(O, t) = \bot$ is dual, we just have to replace $\top$ by $\bot$ and φ by $T\Sigma^\omega \setminus \varphi$ in the previous case.

2.) This follows directly from the definition. □

Now, let us order $\mathbb{B}_4$ with the partial order shown in Figure 3, with the intuition that $\preccurlyeq$ orders verdicts by their specificity.

Then, more restrictive assumptions and more precise observations lead to more specific verdicts.

Lemma 2 *Let $A \supseteq A'$, $O \supseteq O'$, and $t \leq t'$ with $t \geq \tau(O)$ and $t' \geq \tau(O')$. Then,*

$$\mathcal{V}(\varphi, A)(O, t) \preccurlyeq \mathcal{V}(\varphi, A')(O', t').$$

Proof We proceed by case distinction.

- If $\mathcal{V}(\varphi, A)(O, t) = ?$, the claim is vacuously true, as $?$ is the $\preccurlyeq$-smallest element in $\mathbb{B}_4$.
- If $\mathcal{V}(\varphi, A)(O, t) = \top$, then we have $O \cdot_t T\Sigma^\omega \cap A \neq \emptyset$ and $O \cdot_t T\Sigma^\omega \cap A \subseteq \varphi$ by definition. Now, if we have $O' \cdot_{t'} T\Sigma^\omega \cap A' = \emptyset$, then $\mathcal{V}(\varphi, A')(O', t') = \times$, which, as $\preccurlyeq$-largest element, is more specific than $\mathcal{V}(\varphi, A)(O, t)$. On the other hand, if $O' \cdot_{t'} T\Sigma^\omega \cap A' \neq \emptyset$, then applying Remark 1 yields

  $$O' \cdot_{t'} T\Sigma^\omega \cap A' \subseteq O \cdot_t T\Sigma^\omega \cap A \subseteq \varphi.$$

  Thus,

  $$\mathcal{V}(\varphi, A')(O', t') = \top \succcurlyeq \top = \mathcal{V}(\varphi, A)(O, t)$$

  as required.
- The reasoning for $\mathcal{V}(\varphi, A)(O, t) = \bot$ is dual, we just have to replace $\top$ by $\bot$ and φ by $T\Sigma^\omega \setminus \varphi$ in the previous case.
- Finally, if $\mathcal{V}(\varphi, A)(O, t) = \times$, i.e., we have $O \cdot_t T\Sigma^\omega \cap A = \emptyset$, then applying Remark 1 yields

  $$O' \cdot_{t'} T\Sigma^\omega \cap A' \subseteq O \cdot_t T\Sigma^\omega \cap A = \emptyset,$$

  which implies

  $$\mathcal{V}(\varphi, A')(O', t') = \times \succcurlyeq \times = \mathcal{V}(\varphi, A)(O, t)$$

  as required. □

3.3 From the Abstract Definition to a Concrete Definition

Our goal is to present an algorithm computing the monitoring function $\mathcal{V}$ in the concrete setting where

- the property φ and its complement are accepted by TBA's (this covers in particular the case of φ being given in MITL due to Theorem 1),
- the assumption A is given by a TBA, and
- the observation O is given by a sequence of time-intervals and propositional formulas over the locations, the clock constraints, and the alphabet of the assumption automaton capturing the (incomplete) information observed during monitoring.

We begin by introducing the assumption and observations. The former is given by a TBA, which we typically denote by $\mathcal{A}$ to distinguish it from other TBA. Thus, let $\mathcal{A} = (Q, Q_0, \Sigma, C, \Delta, \mathcal{F})$ be a TBA, i.e., Q is the set of locations, Σ is the alphabet, and C is the set of clocks. Let $G'(C)$ denote the clock constraints over C with non-negative rational bounds, i.e., conjunctions of atomic constraints of the form $c \sim t$, where $c \in C$ is a clock, $t \in \mathbb{Q}_{\geq 0}$, and $\sim \in \{<, \leq, =, \geq, >\}$. Let ϕ be a (finite) propositional formula over the set $\Sigma \cup Q \cup G'(C)$ of propositions (which is infinite!), let $\sigma \in \Sigma$, and let (q, v) be a state of $\mathcal{A}$. We define $\sigma, (q, v) \models \phi$ as follows:

- For $\sigma' \in \Sigma$, $\sigma, (q, v) \models \sigma'$ if and only if $\sigma' = \sigma$.
- For $q' \in Q$, $\sigma, (q, v) \models q'$ if and only if $q' = q$.
- For $g \in G'(C)$, $\sigma, (q, v) \models g$ if and only if g is satisfied by v.
- The semantics of Boolean connectives is defined as usual.

An $\mathcal{A}$-observation is a finite sequence $o = (\phi_1, I_1, m_1) \cdots (\phi_n, I_n, m_n)$ where the ϕ_j are (finite) propositional formulas over $\Sigma \cup Q \cup G'(C)$, the I_j are bounded intervals of $\mathbb{R}_{\geq 0}$ with rational endpoints (which may overlap), and the multiplicities m_j are in $\{\leq e, = e, \geq e \mid e \in \mathbb{N}_0\}$. These multiplicities can describe bounds on the number of occurrences of events. For example,

- exactly k events that occur in the interval I and satisfy ϕ is captured by the triple $(\phi, I, = k)$,
- an unbounded number of events that occur in the interval I and satisfy ϕ is captured by $(\phi, I, \geq 0)$,
- at most k events that occur in the interval I and satisfy ϕ is captured by $(\phi, I, \leq k)$,
- at least k events that occur in the interval I and satisfy ϕ is captured by $(\phi, I, \geq k)$, and
- a number of events in $[\ell, u]$ that occur in the interval I and satisfy ϕ is captured by the concatenation $(\phi, I, = \ell)(\phi, I, \leq u - \ell)$.

Let $\rho = (\sigma_1, \tau_1) \cdots (\sigma_{n'}, \tau_{n'})$ be a finite timed word and let

$$r = (q_0, v_0) \xrightarrow{(\sigma_1, \tau_1)} (q_1, v_1) \xrightarrow{(\sigma_2, \tau_2)} \cdots \xrightarrow{(\sigma_{n'-1}, \tau_{n'-1})} (q_{n'-1}, v_{n'-1}) \xrightarrow{(\sigma_{n'}, \tau_{n'})} (q_{n'}, v_{n'})$$

be a prefix of a run of $\mathcal{A}$ with $q_0 \in Q_0$ and $v_0(c) = 0$ for all $c \in C$ (note that r processes ρ). Then, we say that r witnesses that ρ is consistent with an observation $o = (\phi_1, I_1, m_1) \cdots (\phi_n, I_n, m_n)$, if there is a function $h \colon \{1, 2, \ldots, n'\} \to \{1, 2, \ldots, n\}$ such that

1. $h(1) \leq h(2) \leq \cdots \leq h(n')$,
2. $\sigma_{j'}, (q_{j'}, v_{j'}) \models \phi_{h(j')}$ for all $j' \in \{1, 2, \ldots, n'\}$,
3. $\tau_{j'} \in I_{h(j')}$ for all $j' \in \{1, 2, \ldots, n'\}$, and
4. for all $j \in \{1, 2, \ldots, n\}$:
   - If m_j is equal to $\leq e$, then $|\{j' \in \{1, 2, \ldots, n'\} \mid h(j') = j\}| \leq e$.
   - If m_j is equal to $= e$, then $|\{j' \in \{1, 2, \ldots, n'\} \mid h(j') = j\}| = e$.

- If m_j is equal to $\geq e$, then $|\{j' \in \{1, 2, \ldots, n'\} \mid h(j') = j\}| \geq e$.

Intuitively, the function h maps indexes of r and ρ to indexes of o. Each index of r and ρ has to satisfy an element in o (i.e., the one it is mapped to by h), and each element of o has to be satisfied by a number of elements in r and ρ according to its multiplicity (i.e., the elements that are mapped to it by h).

Definition 2 The language $\mathcal{C}_{\mathcal{A}}(o) \subseteq T\Sigma^*$ contains all words that are consistent with o.

Example 4 Let us continue Example 3 and let $\mathcal{A}$ be the assumption automaton shown in Figure 2. Consider the $\mathcal{A}$-observation

$$o = (s, [0, 0], =1)(a, [0, 0], =1)(\neg a, [0, 7], \geq 0)(a, [6, 7], =1)$$
$$(\neg a, [6, 16], \geq 0)(a, [15, 16], =1)(\neg a, [0, 30], \geq 0).$$

Then, as argued in Example 3, $\mathcal{C}_{\mathcal{A}}(o)$ is the language

$$\{(s, 0)(a, 0)(a, t_1)(a, t_2)(b, t_{3,1}) \cdots (b, t_{3,n_3}) \mid$$
$$t_1 \in [6, 7], t_2 \in [15, 16], \text{and } t_2{+}10 < t_{3,1} \leq \cdots \leq t_{3,n_3} \leq 30\}.$$

For example, given the run prefix (we ignore the clock x as it is always equal to the timestamp on the transition leading to a state)

$$r_0 = (q_0, y = 0) \xrightarrow{(s,0)} (q_1, y = 0) \xrightarrow{(a,0)}$$
$$(q_2, y = 0) \xrightarrow{(a,6)} (q_2, y = 0) \xrightarrow{(a,15)} (q_2, y = 0)$$

we can define h as follows: $h(1) = 1, h(2) = 2, h(3) = 4$, $h(4) = 6$. Thus, r_0 witnesses that $(s, 0)(a, 0)(a, 6)(a, 15)$ is in $\mathcal{C}_{\mathcal{A}}(o)$.

For the run prefix

$$r_1 = (q_0, y = 0) \xrightarrow{(s,0)} (q_1, y = 0) \xrightarrow{(a,0)} (q_2, y = 0) \xrightarrow{(a,6)}$$
$$(q_2, y = 0) \xrightarrow{(b,15)} (q_3, y = 9) \xrightarrow{(a,16)} (q_3, y = 10)$$

we can define the function h as follows: $h(1) = 1, h(2) = 2, h(3) = 4, h(4) = 5, h(5) = 6$. Thus, r_1 witnesses that $(s, 0)(a, 0)(a, 6)(b, 15)(a, 16)$ is in $\mathcal{C}_{\mathcal{A}}(o)$. Note that $(s, 0)(a, 0)(a, 6)(b, 15)(a, 16)$ is *not* a prefix of any word that satisfies the assumption, as it contains an "a" that is followed by a "b" within nine units of time. But it is in $\mathcal{C}_{\mathcal{A}}(o)$, as membership in that set only depends on the existence of a run prefix, and not the existence of a prefix of an accepting run. The acceptance condition will be later taken care of (concretely in the first step of our algorithm in Subsection 4.4).

Finally, the run prefix

$$r_2 = (q_0, y = 0) \xrightarrow{(s,0)} (q_1, y = 0) \xrightarrow{(a,0)} (q_2, y = 0) \xrightarrow{(a,6)}$$
$$(q_2, y = 0) \xrightarrow{(a,15)} (q_2, y = 0) \xrightarrow{(a,16)} (q_2, y = 0)$$

is the prefix of a run of $\mathcal{A}$ but it is not compatible with the observation o. In fact, any h satisfying the conditions 1), 2), and 3) should assign $h(4) = 6$ and $h(5) = 6$ violating condition 4). Thus, r_2 is not a witness for any word in $\mathcal{C}_{\mathcal{A}}(o)$.

Before we continue, let us collect some useful facts about $\mathcal{A}$-observations.

Remark 2 We can restrict ourselves w.l.o.g. to multiplicities m_i from the set $\{\leq e, = e, \geq 0 \mid e \in \mathbb{N}\}$, as elements (ϕ_i, I_i, m_i) with multiplicity m_i of the form ≤ 0 or $= 0$ can be removed from the observation without changing the set of consistent words, and as each element $(\phi_i, I_i, \geq e)$, for some $e > 0$, can be replaced by $(\phi_i, I_i, = e)(\phi_i, I_i, \geq 0)$ without changing the set of consistent words. Thus, in all following proofs we only consider such restricted multiplicities.

Furthermore, let us note that there is a syntactic upper bound on the duration of finite timed words that can be consistent with an observation.

Remark 3 We have $\tau(\mathcal{C}_{\mathcal{A}}((\phi_1, I_1, m_1) \cdots (\phi_n, I_n, m_n))) \leq \sup I_n$ by definition.

Before we study the algorithmic properties of our monitoring framework we have introduced it is instructive to compare it to the abstract definition of monitoring under assumptions presented in Definition 1: In our concrete setting, the language $L(\mathcal{A})$ takes the role of the assumption A, the language $\mathcal{C}_{\mathcal{A}}(o)$ takes the role of the observation O, and the property φ is represented by two automata, one accepting φ and one accepting its complement $T\Sigma^{\omega} \setminus \varphi$. Note that here the observation depends on the TBA inducing the assumption, as observations can refer to locations and clock valuations of $\mathcal{A}$.

Next, we show that we can determine whether an observation has at least one finite word that is consistent with the assumption. If this is not the case, i.e., $\mathcal{C}_{\mathcal{A}}(o)$ is empty, then o is not consistent with the assumption $\mathcal{A}$. However, the converse is in general not true, i.e., $\mathcal{C}_{\mathcal{A}}(o)$ may be nonempty, but the words in it cannot be extended to infinite words in $L(\mathcal{A})$.

Lemma 3 *The problem "Given a TBA $\mathcal{A}$ and an $\mathcal{A}$-observation o, is $\mathcal{C}_{\mathcal{A}}(o)$ nonempty?" is decidable.*

Proof We inductively (over the length of o) construct a timed automaton $\mathcal{A} \otimes o$ (over finite timed words) that accepts $\mathcal{C}_{\mathcal{A}}(o)$. Our result follows then from non-emptiness of timed automata being in PSPACE [11].

Let $\mathcal{A} = (Q, Q_0, \Sigma, C, \Delta, \mathcal{F})$. The timed automata $\mathcal{A} \otimes o$ we construct satisfy the following invariants: The set of locations Q' of $\mathcal{A} \otimes o$ is always a subset of $Q \times S$ for some finite set S (but not necessarily equal to $Q \times S$). In the inductive proof, we assume that $S \cap \mathbb{N}$ is empty, which can always be achieved by renaming locations. This allows us to use subsets of $Q \times \mathbb{N}$ as fresh states during the induction. Further, the set of clocks of each $\mathcal{A} \otimes o$ is equal to $C \cup \{time\}$, where $time$ is a fresh clock that is never reset, i.e., it measures the time that has passed since the start of a run.

The automata $\mathcal{A} \otimes o$ we construct have rationals in their clock constraints. These can be eliminated by multiplying each one of them by the largest denominator appearing in the automaton, which results in an emptiness-equivalent timed automaton. Furthermore, we use clock constraints of the form $time \in I$, for $I = [\ell, u]$ with rational endpoints ℓ and u, as shorthands for $time \geq \ell \wedge time \leq u$.

Now, we can begin the inductive construction by considering the empty observation $o = \varepsilon$: We define $\mathcal{A} \otimes \varepsilon = (Q \times \{s\}, Q_0 \times \{s\}, \Sigma, C \cup \{time\}, \emptyset, Q_0 \times \{s\})$, where s is a fresh symbol. This timed automaton satisfies the invariants and accepts the language $\{\varepsilon\}$ if Q_0 is nonempty, otherwise it accepts the empty language. This is in both cases equal to $\mathcal{C}_{\mathcal{A}}(\varepsilon)$.

For the induction step, consider an observation $o(\phi, I, m)$, i.e., we concatenate o with (ϕ, I, m). Let $q \in Q$ and $a \in \Sigma$, and let $\phi_{(q,a)}$ be the formula obtained from ϕ by replacing each subformula a and q by `true` and each subformula $a' \neq a$ and $q' \neq q$ by `false`, i.e., the only remaining atomic subformulas are clock constraints. By bringing $\phi_{(q,a)}$ into disjunctive normal form and exploiting that clock constraints are closed under conjunction, we obtain that $\phi_{(q,a)}$ is equivalent to a formula of the form $\bigvee_{d \in D_{(q,a)}} g_{(q,a,d)}$ for some finite index set $D_{(q,a)}$, where each $g_{(q,a,d)}$ is a clock constraint.

By induction hypothesis, there is a timed automaton $\mathcal{A} \otimes o = (Q', Q'_0, \Sigma, C \cup \{time\}, \Delta', \mathcal{F}')$ with language $\mathcal{C}_{\mathcal{A}}(o)$. We construct $\mathcal{A} \otimes (o(\phi, I, m))$ by case distinction over the form of m by intuitively extending $\mathcal{A} \otimes o$ by transitions capturing (ϕ, I, m).

First, assume m is of the form "$= e$" for some $e > 0$. Intuitively, we extend $\mathcal{A} \otimes o$ by runs of $\mathcal{A}$ of length exactly e that additionally have to satisfy the requirements spelled out by ϕ and I, which can be added as clock constraints on the edges. To this end, we extend $\mathcal{A} \otimes o$ by adding e copies of $\mathcal{A}$'s locations and edges leading from $\mathcal{F}'$ to the first copy as well as edges leading from the j-th copy to the $(j+1)$-th (see Figure 4 for an illustration).

Thus, we define $\mathcal{A} \otimes (o(\phi, I, m)) = (Q'', Q'_0, \Sigma, C \cup \{time\}, \Delta'', \mathcal{F}'')$ where

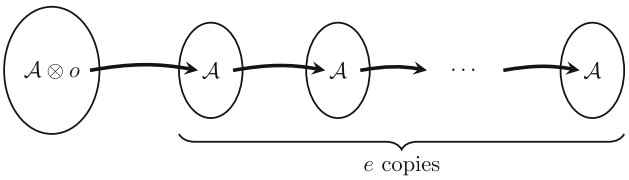


Fig. 4 Extending $\mathcal{A} \otimes o$ to obtain $\mathcal{A} \otimes (o(\phi, I, = e))$

- $Q'' = Q' \cup (Q \times \{1, 2, \ldots, e\})$ (recall that we assume w.l.o.g., that $Q' \subseteq Q \times S$ with $S \cap \mathbb{N} = \emptyset$, which implies that the union is disjoint),
- Δ'' contains
  - all transitions from Δ',
  - all transitions of the form $((q, s), (q', 1), a, \lambda, g \wedge g_{(q',a,d)} \wedge time \in I)$ for $(q, q', a, \lambda, g) \in \Delta$, $d \in D_{(q,a)}$, and $(q, s) \in \mathcal{F}'$, as well as
  - all transitions of the form $((q, j), (q', j+1), a, \lambda, g \wedge g_{(q',a,d)} \wedge time \in I)$ for $(q, q', a, \lambda, g) \in \Delta$, $d \in D_{(q,a)}$, and $1 \leq j < e$, and
- $\mathcal{F}''$ is equal to $Q \times \{e\}$.

Now, assume m is of the form "$\leq e$" for some $e > 0$. Here, we use the same transition structure as in the previous case, but runs do not have to reach $Q \times \{e\}$ to be accepting (capturing "exactly e matches of (ϕ, I, m)"), but can stop early. Formally, we define $\mathcal{A} \otimes (o(\phi, I, m)) = (Q'', Q'_0, \Sigma, C \cup \{time\}, \Delta'', \mathcal{F}'')$ where Q'', and Δ'' are as in the previous case, but $\mathcal{F}''$ is equal to $\mathcal{F}' \cup (Q \times \{1, 2, \ldots, e\})$, which captures "at most e matches of (ϕ, I, m)".

Finally, assume m is of the form "≥ 0". Here, we intuitively extend $\mathcal{A} \otimes o$ by runs of $\mathcal{A}$ of arbitrary (potentially zero) length by intuitively concatenating $\mathcal{A} \otimes o$ (at the accepting locations) and a copy of $\mathcal{A}$ with ε-transitions: from an accepting location of the form (q, s) of $\mathcal{A} \otimes o$ an ε-edge leads to the copy of q of $\mathcal{A}$. In the copy, we again have to satisfy the requirements spelled out by ϕ and I.

Since we do not allow ε-transitions in timed automata, we instead define $\mathcal{A} \otimes (o(\phi, I, m))$ to be the result of the classical construction removing such transitions: for each transition of $\mathcal{A}$ leading from q to q' and each accepting (q, s) in $\mathcal{A} \otimes o$, we add a transition from (q, s) to the copy of q'. Every location in the copy is accepting. Furthermore, the accepting locations of $\mathcal{A} \otimes o$ are also accepting in $\mathcal{A} \otimes (o(\phi, I, m))$ to allow for zero matches of (ϕ, I, m)).

Thus, we define $\mathcal{A} \otimes (o(\phi, I, m)) = (Q'', Q''_0, \Sigma, C \cup \{time\}, \Delta'', \mathcal{F}'')$ where

- $Q'' = Q' \cup (Q \times \{1\})$ (recall that we assume w.l.o.g., that $Q' \subseteq Q \times S$ with $S \cap \mathbb{N} = \emptyset$, which implies that the union is disjoint),
- $Q''_0 = Q'_0$,

- Δ'' contains
  - all transitions from Δ',
  - all transitions of the form $((q, s), (q', 1), a, \lambda, g \wedge g_{(q',a,d)} \wedge time \in I)$ for $(q, q', a, \lambda, g) \in \Delta$ with $(q, s) \in \mathcal{F}'$, and $d \in D_{(q,a)}$ and
  - all transitions of the form $((q, 1), (q', 1), a, \lambda, g \wedge g_{(q',a,d)} \wedge time \in I)$ for $(q, q', a, \lambda, g) \in \Delta$ and $d \in D_{(q,a)}$, and
- $\mathcal{F}''$ is equal to $\mathcal{F}' \cup (Q \times \{1\})$.

In all three cases, the invariants are satisfied after renaming the states. Now, an induction shows that the language of $\mathcal{A} \otimes o$ is equal to $\mathcal{C}_{\mathcal{A}}(o)$ and another induction shows that the number locations of $\mathcal{A} \otimes o$ is bounded by $n \cdot e^*$, where n is the number of locations of $\mathcal{A}$ and e^* is the sum of the constants in the multiplicities (where each multiplicity of the form ≥ 0 contributes 1 to the sum instead of 0). Note that e^* is the number of copies of $\mathcal{A}$ used to construct $\mathcal{A} \otimes o$. Furthermore, $\mathcal{A} \otimes o$ uses only one more clock than $\mathcal{A}$. However, the number of transitions may be exponential, as we turn arbitrary Boolean formulas (the ϕ in the observations) into disjunctive normal form, which can incur an exponential blowup. The number of transitions of $\mathcal{A} \times o$ in turn depends on the size of those formulas in disjunctive normal form. Finally, note that the size of the bounds used in the guards of $\mathcal{A} \times o$ depends on the bounds in $\mathcal{A}$ and the bounds in o. □

4 A Zone-Based Monitoring Algorithm

In this section, we present an algorithm computing the monitoring function $\mathcal{V}$. To this end, we first need to introduce some notation for TBA and zones to represent subsets of states of TBA, which may be uncountable.

In Subsection 4.1 we introduce zones to symbolically represent sets of states of timed automata. Then in Subsection 4.2 we present a characterization of the monitoring function using reachability sets in timed automata. These reachability sets are shown to be computable in Subsection 4.3 using a zone-based algorithm. This is the basis of our monitoring algorithm which is presented in Subsection 4.4.

4.1 Zones

For the monitoring algorithm, we use – as is standard in analysing timed automata models – symbolic states being pairs (q, Z) of locations and zones. A zone is a finite conjunction of constraints of the form $x \sim t$ and $x - x' \sim t$ for clocks x, x', constants $t \in \mathbb{Q}_{\geq 0}$, and $\sim \in \{<, \leq, =, \geq, >\}$. We write $v \models Z$ to denote that the clock valuation v satisfies all constraints of the zone Z. Furthermore, we interchangeably treat zones as conjunctions of clock constraints as well as sets of clock valuations that satisfy its constraints. Given two zones Z and Z' over a set C of clocks, and a set $\lambda \subseteq C$ of clocks, we define the following operations on zones (which can be efficiently implemented using the DBM data-structure [15]):

- $Z[\lambda] = \{v \mid \exists v' \models Z$ such that $v(x) = 0$ if $x \in \lambda$, otherwise $v(x) = v'(x)\}$
- $Z^{\nearrow} = \{v \mid \exists v' \models Z$ such that $v = v' + d$ for some $d \in \mathbb{R}_{\geq 0}\}$
- $Z \wedge Z' = \{v \mid v \models Z$ *and* $v \models Z'\}$.

To describe our algorithm, we first define the set of states of a TBA from where it is possible to reach an accepting location infinitely many times in the future, i.e., those states from which an accepting run is possible. This is useful, because if processing a finite timed word leads to such a state, then the timed word can be extended to an infinite one in the language of the automaton, a notion that underlies Definition 1. Given a TBA $\mathcal{B} = (Q, Q_0, \Sigma, C, \Delta, \mathcal{F})$, the set of states with nonempty language is

$$S^{ne}_{\mathcal{B}} = \{(q, v) \mid q \in Q, v \in C \to \mathbb{R}_{\geq 0} \text{ such that } L(\mathcal{B}, (q, v)) \neq \emptyset\}.$$

Proposition 2 ([5]) *$S^{ne}_{\mathcal{B}}$ is a finite union of zones and can be computed using a zone-based algorithm.*

4.2 Characterizing Monitoring via Reachability Sets

We continue by capturing the set of states of a TBA that can be reached by processing a finite timed word. Let $\mathcal{B} = (Q, Q_0, \Sigma, C, \Delta, \mathcal{F})$ be a TBA. In the following definition, we write $(q_0, v_0) \xrightarrow{\rho}_{\mathcal{B}} (q_n, v_n)$ for a finite timed word $\rho = (\sigma_1, \tau_1) \cdots (\sigma_n, \tau_n) \in T\Sigma^*$ to denote the existence of a finite sequence

$$(q_0, v_0) \xrightarrow{(\sigma_1,\tau_1)} (q_1, v_1) \xrightarrow{(\sigma_2,\tau_2)} \cdots \xrightarrow{(\sigma_n,\tau_n)} (q_n, v_n)$$

of states, where for all $1 \leq i \leq n$ there is a transition $(q_{i-1}, q_i, \sigma_i, \lambda_i, g_i)$ such that $v_i(c) = 0$ for all c in λ_i and $v_{i-1}(c) + (\tau_i - \tau_{i-1})$ otherwise, and g is satisfied by the valuation $v_{i-1} + (\tau_i - \tau_{i-1})$, where we use $\tau_0 = 0$. Given a TBA $\mathcal{B}$, a finite timed word $\rho \in T\Sigma^*$, and a time-point $t \in \mathbb{R}_{\geq 0}$ with $t \geq \tau(\rho)$, the set of possible states a run over ρ starting from initial states of $\mathcal{B}$ can end in after time t has passed is

$$\mathcal{T}_{\mathcal{B}}(\rho, t) = \bigcup\nolimits_{q_0 \in Q_0} \{(q, v + (t - \tau(\rho))) \mid (q_0, v_0) \xrightarrow{\rho}_{\mathcal{B}} (q, v)\},$$

where v_0 is the clock valuation mapping every clock to 0. We call $\mathcal{T}_{\mathcal{B}}(\rho, t)$ the reach-set of $\mathcal{B}$ over (ρ, t). The above definition is adapted from [5] to take into account the time that has passed since the last observation, i.e., the value $t - \tau(\rho)$.

Next, we lift this definition to sets $L \subseteq T\Sigma^*$ of finite words via

$$\mathcal{T}_{\mathcal{B}}(L, t) = \bigcup\nolimits_{\rho \in L} \mathcal{T}_{\mathcal{B}}(\rho, t),$$

assuming $t \geq \tau(L)$. Otherwise, $\mathcal{T}_{\mathcal{B}}(L, t) = \emptyset$ by convention.

First, we show that reach-sets and states with a nonempty language allow us to characterize monitoring. Then, we show how to compute reach-sets of observations.

In the following lemma, $\mathcal{C}_{\mathcal{A}}(o)$ is the observation O in Definition 1 and $L(\mathcal{A})$ is the assumption A. Furthermore, the assumption on the property φ in the lemma is satisfied for all MITL-definable properties.

Lemma 4 *Let $\varphi \subseteq T\Sigma^\omega$ be a property such that there are TBA's $\mathcal{B}_\varphi$ and $\mathcal{B}_{\neg\varphi}$ with $L(\mathcal{B}_\varphi) = \varphi$ and $L(\mathcal{B}_{\neg\varphi}) = T\Sigma^\omega \setminus \varphi$, let $o = (\phi_1, I_1, m_1) \cdots (\phi_n, I_n, m_n)$ be an $\mathcal{A}$-observation for some TBA $\mathcal{A}$, and let $t \geq \sup I_n$.*

1. *$\mathcal{C}_{\mathcal{A}}(o) \cdot_t T\Sigma^\omega \cap L(\mathcal{A}) \neq \emptyset$ if and only if $\mathcal{T}_{\mathcal{A}}(\mathcal{C}_{\mathcal{A}}(o), t) \cap S^{ne}_{\mathcal{A}} \neq \emptyset$.*
2. *$\mathcal{C}_{\mathcal{A}}(o) \cdot_t T\Sigma^\omega \cap L(\mathcal{A}) \subseteq \varphi$ if and only if $\mathcal{T}_{\mathcal{B}_{\neg\varphi} \otimes \mathcal{A}}(\mathcal{C}_{\mathcal{A}}(o), t) \cap S^{ne}_{\mathcal{B}_{\neg\varphi} \otimes \mathcal{A}} = \emptyset$.*
3. *$\mathcal{C}_{\mathcal{A}}(o) \cdot_t T\Sigma^\omega \cap L(\mathcal{A}) \subseteq T\Sigma^\omega \setminus \varphi$ if and only if $\mathcal{T}_{\mathcal{B}_\varphi \otimes \mathcal{A}}(\mathcal{C}_{\mathcal{A}}(o), t) \cap S^{ne}_{\mathcal{B}_\varphi \otimes \mathcal{A}} = \emptyset$.*

Proof 1.) For the left-to-right implication, let $\rho = (\sigma_1, \tau_1)(\sigma_2, \tau_2) \cdots \in \mathcal{C}_{\mathcal{A}}(o) \cdot_t T\Sigma^\omega \cap L(\mathcal{A})$. Due to $\rho \in \mathcal{C}_{\mathcal{A}}(o) \cdot_t T\Sigma^\omega$, there is an $n \geq 0$ such that $\rho' = (\sigma_1, \tau_1) \cdots (\sigma_n, \tau_n) \in \mathcal{C}_{\mathcal{A}}(o)$, $\tau_n \leq t$, and $\tau_{n+1} \geq t$. On the other hand, due to $\rho \in L(\mathcal{A})$, there is an accepting run

$$(q_0, v_0) \xrightarrow{(\sigma_1, \tau_1)} (q_1, v_1) \xrightarrow{(\sigma_2, \tau_2)} (q_2, v_2) \xrightarrow{(\sigma_3, \tau_3)} \cdots$$

of $\mathcal{A}$. Hence, we have $(q_0, v_0) \xrightarrow{\rho'}_{\mathcal{A}} (q_n, v_n)$. Thus, $(q_n, v_n + \delta) \in \mathcal{T}_{\mathcal{A}}(\rho', t) \subseteq \mathcal{T}_{\mathcal{A}}(\mathcal{C}_{\mathcal{A}}(o), t)$, where we define $\delta = t - \tau_n$. It remains to show $(q_n, v_n + \delta) \in S^{ne}_{\mathcal{A}}$. To this end, consider the run suffix

$$(q_n, v_n) \xrightarrow{(\sigma_{n+1}, \tau_{n+1})} (q_{n+1}, v_{n+2}) \xrightarrow{(\sigma_{n+2}, \tau_{n+2})} (q_{n+2}, v_{n+2}) \xrightarrow{(\sigma_{n+3}, \tau_{n+3})} \cdots,$$

which is accepting. Note that $\tau_n \leq t \leq \tau_{n+1}$ implies $\tau_{n+1} - \delta \geq \tau_n \geq 0$. Hence, $\tau_{n+1} \leq \tau_{n+2} \leq \cdots$ implies $\tau_{n+j} - \delta \geq 0$ for all $j \geq 1$. Thus, the run

$$(q_n, v_n + \delta) \xrightarrow{(\sigma_{n+1}, \tau_{n+1} - \delta)} (q_{n+1}, v_{n+2}) \xrightarrow{(\sigma_{n+2}, \tau_{n+2} - \delta)} (q_{n+2}, v_{n+2}) \xrightarrow{(\sigma_{n+3}, \tau_{n+3} - \delta)} \cdots$$

is well-defined. As it starts in $(q_n, v_n + \delta)$ and is accepting, we conclude $(q_n, v_n + \delta) \in S^{ne}_{\mathcal{A}}$ as required.

For the right-to-left implication, let $(q, v) \in \mathcal{T}_{\mathcal{A}}(\mathcal{C}_{\mathcal{A}}(o), t) \cap S^{ne}_{\mathcal{A}}$. Due to $(q, v) \in \mathcal{T}_{\mathcal{A}}(\mathcal{C}_{\mathcal{A}}(o), t)$ there is a $\rho' = (\sigma_1, \tau_1) \cdots (\sigma_n, \tau_n) \in \mathcal{C}_{\mathcal{A}}(o)$ with $\tau_n \leq t$ and there is a run prefix

$$(q_0, v_0) \xrightarrow{(\sigma_1, \tau_1)} (q_1, v_1) \xrightarrow{(\sigma_2, \tau_2)} \cdots \xrightarrow{(\sigma_n, \tau_n)} (q_n, v_n)$$

for some $q_0 \in Q_0$, where v_0 is the clock valuation mapping every clock to zero, such that $v = v_n + \delta$, where we define $\delta = (t - \tau_n)$. Furthermore, due to $(q, v) \in S^{ne}_{\mathcal{A}}$, there is a $\rho = (\sigma_{n+1}, \tau_{n+1})(\sigma_{n+2}, \tau_{n+2}) \cdots$ and an accepting run

$$(q'_n, v'_n) \xrightarrow{(\sigma_{n+1}, \tau_{n+1})} (q'_{n+1}, v'_{n+2}) \xrightarrow{(\sigma_{n+2}, \tau_{n+2})} (q'_{n+2}, v'_{n+2}) \xrightarrow{(\sigma_{n+3}, \tau_{n+3})} \cdots$$

starting in (q, v) that processes ρ.

These two runs can be combined into the accepting run

$$(q_0, v_0) \xrightarrow{(\sigma_1, \tau_1)} (q_1, v_1) \xrightarrow{(\sigma_2, \tau_2)} \cdots \xrightarrow{(\sigma_n, \tau_n)} (q_n, v_n) \xrightarrow{(\sigma_{n+1}, \tau_{n+1})} (q'_{n+1}, v'_{n+2}) \xrightarrow{(\sigma_{n+2}, \tau_{n+2} + \delta)} (q'_{n+2}, v'_{n+2}) \xrightarrow{(\sigma_{n+3}, \tau_{n+3} + \delta)} \cdots$$

processing $\rho' \cdot_t \rho$, which is therefore in $\mathcal{C}_{\mathcal{A}}(o) \cdot_t T\Sigma^\omega \cap L(\mathcal{A})$.

2.) We show that the negations of both statements are equivalent.

So, let $\mathcal{C}_{\mathcal{A}}(o) \cdot_t T\Sigma^\omega \cap L(\mathcal{A}) \not\subseteq \varphi$, i.e., $\mathcal{C}_{\mathcal{A}}(o) \cdot_t T\Sigma^\omega \cap L(\mathcal{A}) \cap (T\Sigma^\omega \setminus \varphi) \neq \emptyset$. Using arguments as in the left-to-right implication of Item 1., we can find a state $(q, v) \in \mathcal{T}_{\mathcal{B}_{\neg\varphi} \otimes \mathcal{A}}(\mathcal{C}_{\mathcal{A}}(o), t) \cap S^{ne}_{\mathcal{B}_{\neg\varphi} \otimes \mathcal{A}}$, which is thus not empty.

Now, let $\mathcal{T}_{\mathcal{B}_{\neg\varphi} \otimes \mathcal{A}}(\mathcal{C}_{\mathcal{A}}(o), t) \cap S^{ne}_{\mathcal{B}_{\neg\varphi} \otimes \mathcal{A}} \neq \emptyset$. Using arguments as in the right-to-left implication of Item 1., we can find a word of the form $\rho' \cdot_t \rho$ for $\rho' \in \mathcal{C}_{\mathcal{A}}(o)$ and $\rho \in T\Sigma^\omega$ such that $\rho' \cdot_t \rho \in L(\mathcal{A}) \cap L(\mathcal{B}_{\neg\varphi})$. Hence, $\rho' \cdot_t \rho$ is in $\mathcal{C}_{\mathcal{A}}(o) \cdot_t T\Sigma^\omega \cap L(\mathcal{A})$, but not in φ. Thus, $\mathcal{C}_{\mathcal{A}}(o) \cdot_t T\Sigma^\omega \cap L(\mathcal{A}) \not\subseteq \varphi$.

3.) The reasoning is dual to Item 2, we just have to replace φ by $T\Sigma^\omega \setminus \varphi$ and $\mathcal{B}_{\neg\varphi}$ by $\mathcal{B}_\varphi$. □

4.3 Computing Reachability Sets

We now show how to compute reach-sets using zones. First, we use the zone operations introduced above to compute the successor states of an input letter with a given target location. Fix a TBA $(Q, Q_0, \Sigma, C, \Delta, \mathcal{F})$. For a symbolic state (q, Z),

a letter $\sigma \in \Sigma$ and target location $q' \in Q$, we define

$$\begin{aligned}\text{Post}((q, Z), \sigma, q') &= \{(q', Z') \mid (q, q', \sigma, \lambda, g) \in \Delta, Z' \\ &= (Z^{\nearrow} \wedge g)[\lambda]\},\end{aligned}$$

being the set of states one can reach by taking a σ-transition at some point in the future from (q, Z) with q' as target-location. Using Post we can compute the successor states of a time-uncertain letter/target location (σ, q', I), where $\sigma \in \Sigma$, $q' \in Q$ and $I \subseteq \mathbb{R}_{\geq 0}$ is a time interval with rational endpoints. For this, we extend zones with an additional clock $time$ just recording time since system start. The successors of a symbolic state (q, Z) are

$$\begin{aligned}\text{Succ}((q, Z), (\sigma, q', I)) = \{(q', Z') \mid (q', Z'') \in \text{Post}((q, Z), \\ \sigma, q'), Z' = Z'' \wedge time \in I\}\end{aligned}$$

and the successors of a set S of symbolic states are

$$\text{Succ}(S, (\sigma, q', I)) = \bigcup\nolimits_{(q,Z)\in S} \text{Succ}((q, Z), (\sigma, q', I)).$$

Now, our main technical lemma below exploits the above to compute reach-sets. More precisely, given an $\mathcal{A}$-observation o (i.e., the TBA $\mathcal{A}$ represents the assumption), we can compute the reach-set of the set $\mathcal{C}_{\mathcal{A}}(o)$ in the product $\mathcal{B} \otimes \mathcal{A}$, for any given TBA $\mathcal{B}$, i.e., we compute the words consistent with the observation o in the TBA $\mathcal{A}$ (the assumption), while the reach-set of that language is computed in $\mathcal{B} \otimes \mathcal{A}$ (this will later be the product of the property (or its negation) and the assumption as in Lemma 4).

Lemma 5 *Fix TBA $\mathcal{A}, \mathcal{B}$. There is a zone-based online algorithm computing*

$$\mathcal{T}_{\mathcal{B}\otimes\mathcal{A}}(\mathcal{C}_{\mathcal{A}}((\phi_1, I_1, m_1) \cdots (\phi_n, I_n, m_n)), t)$$

(which is a finite union of zones) for every $\mathcal{A}$-observation $(\phi_1, I_1, m_1) \cdots (\phi_n, I_n, m_n)$, and every $t \in \mathbb{Q}_{\geq 0}$ with $t \geq \sup I_n$.

Proof Recall that the state set of $\mathcal{B} \otimes \mathcal{A}$ has the form $Q_{\mathcal{B}} \times Q_{\mathcal{A}} \times \{0, 1\}$, where $Q_{\mathcal{B}}$ and $Q_{\mathcal{A}}$ are the sets of states of $\mathcal{B}$ and $\mathcal{A}$, respectively (see Proposition 1).

Let $o^i = (\phi_1, I_1, m_1) \cdots (\phi_i, I_i, m_i)$, and let us denote by $S^i_{\mathcal{B}\otimes\mathcal{A}}$ the reach-set $\mathcal{T}_{\mathcal{B}\otimes\mathcal{A}}(\mathcal{C}_{\mathcal{A}}(o^i))$ of o^i in $\mathcal{B} \otimes \mathcal{A}$. We will show inductively in i, that $S^i_{\mathcal{B}\otimes\mathcal{A}}$ can be computed using zone operations. For the base case $i = 0$, we note that $\mathcal{C}_{\mathcal{A}}(o^0) = \{\varepsilon\}$, thus $\mathcal{T}_{\mathcal{B}\otimes\mathcal{A}}(\mathcal{C}_{\mathcal{A}}(o^0))$ is the set of initial states of $\mathcal{B} \otimes \mathcal{A}$, which is clearly effectively representable using zones.

For the inductive case, let us assume that $S^{i-1}_{\mathcal{B}\otimes\mathcal{A}}$ is computable using zone operations. Now consider (ϕ_i, I_i, m_i). Given that Σ, $Q_{\mathcal{A}}$ and $Q_{\mathcal{B}}$ are finite, ϕ_i is equivalent to a finite disjunction of simple formulas of the form $\psi_{i,j} = \sigma_{i,j} \wedge q^a_{i,j} \wedge g_{i,j}$, where $\sigma_{i,j} \in \Sigma$, $q^a_{i,j} \in Q_{\mathcal{A}}$, and $g_{i,j} \in G'(C_{\mathcal{A}})$.

Now in case m_i is equal to $= e$ for some $e > 0$, the reach-set with respect to $\psi_{i,j}$ is simply $S^{i,e}_{\mathcal{B}\otimes\mathcal{A}}$ where $S^{i,0}_{\mathcal{B}\otimes\mathcal{A}}$ is $S^{i-1}_{\mathcal{B}\otimes\mathcal{A}}$ and

$$\begin{aligned}S^{i,e'}_{\mathcal{B}\otimes\mathcal{A}} = &\bigcup\nolimits_j \bigcup\nolimits_{q^b\in Q_{\mathcal{B}}} \bigcup\nolimits_{k\in\{0,1\}} \\ &\text{Succ}(S^{i,e'-1}_{\mathcal{B}\otimes\mathcal{A}}, (\sigma_{i,j}, (q^b, q^a_{i,j}, k), I_i)) \wedge g_{i,j}\end{aligned}$$

for $0 < e' \leq e$. Here, given a set S of symbolic states, we write $S \wedge g_{i,j}$ for the set of symbolic states of the form $(q, Z \wedge q_{i,j})$ where (q, Z) is in S.

In case m_i is equal to ≥ 0, $S^i_{\mathcal{B}\otimes\mathcal{A}}$ is the least fixed-point $\mathbf{X}$ satisfying the equality

$$\begin{aligned}\mathbf{X} = &S^{i-1}_{\mathcal{B}\otimes\mathcal{A}} \cup \bigcup\nolimits_j \bigcup\nolimits_{q^b\in Q_{\mathcal{B}}} \bigcup\nolimits_{k\in\{0,1\}} \\ &\left[\text{Succ}(\mathbf{X}, (\sigma_{i,j}, (q^b, q^a_{i,j}, k), I_i)) \wedge g_{i,j}\right].\end{aligned}$$

Given the upper bounds of the interval I_i, the least fixed-point will be found in a finite number of iterations of the right-hand-side of the above equation (starting from the empty set).

Finally, in case m_i is equal to $\leq e$ for some $e > 0$, $S^i_{\mathcal{B}\otimes\mathcal{A}}$ is obtained by unraveling the fixed point above exactly e times.

The above inductive proof provides in an obvious manner the basis for an online construction of the sets $\mathcal{T}_{\mathcal{B}\otimes\mathcal{A}}(\mathcal{C}_{\mathcal{A}}(o^i))$. □

Remark 4 For the special case of $\mathcal{B} = \mathcal{A}$ of Lemma 5, we do not have to work on the product $\mathcal{A} \times \mathcal{A}$, but can simplify the construction to work on $\mathcal{A}$ directly. Thus, there is also a zone-based algorithm computing

$$\mathcal{T}_{\mathcal{A}}(\mathcal{C}_{\mathcal{A}}((\phi_1, I_1, m_1) \cdots (\phi_n, I_n, m_n)), t)$$

(which is again a finite union of zones) for every $\mathcal{A}$-observation $(\phi_1, I_1, m_1) \cdots (\phi_n, I_n, m_n)$, and every $t \in \mathbb{Q}_{\geq 0}$ with $t \geq \sup I_n$.

4.4 Monitoring Algorithm

Now, we are able to present our algorithm to compute $\mathcal{V}(\varphi, A)$ for a property $\varphi \subseteq T\Sigma^\omega$ (given by two TBA $\mathcal{B}_\varphi$ and $\mathcal{B}_{\neg\varphi}$ such that $L(\mathcal{B}_\varphi) = \varphi$ and $L(\mathcal{B}_{\neg\varphi}) = T\Sigma^\omega \setminus \varphi$) and an assumption A (given by a TBA $\mathcal{A}$): Given $o = (\phi_1, I_1, m_1) \cdots (\phi_n, I_n, m_n)$ (an observation) and $t \geq \sup I_n$, do the following:

1. Compute $\mathcal{T}_{\mathcal{A}}(\mathcal{C}_{\mathcal{A}}(o), t)$. If it has an empty intersection with $S^{ne}_{\mathcal{A}}$, then return $\times$. This checks whether there is some some finite word that is consistent with the observation and can be extended to satisfy the assumption. If this is not the case, then the assumption was wrong.

2. Compute $\mathcal{T}_{\mathcal{B}_{\neg\varphi}\otimes\mathcal{A}}(\mathcal{C}_{\mathcal{A}}(o), t)$. If it has an empty intersection with $S^{ne}_{\mathcal{B}_{\neg\varphi}\otimes\mathcal{A}}$, then return $\top$: If there is a finite word consistent with the observation that can be extended to satisfy the assumption, but no such extension satisfies the complement of the property, then every such extension must satisfy the property. Hence, we can return $\top$.
3. Compute $\mathcal{T}_{\mathcal{B}_{\varphi}\otimes\mathcal{A}}(\mathcal{C}_{\mathcal{A}}(o), t)$. If it has an empty intersection with $S^{ne}_{\mathcal{B}_{\varphi}\otimes\mathcal{A}}$, then return $\bot$: If there is a finite word consistent with the observation that can be extended to satisfy the assumption, but no such extension satisfies the property, then every such extension must satisfy the complement of the property. Hence, we can return $\bot$.
4. Return *?*. Otherwise, there is both a finite word that is consistent with the observation that can be extended to satisfy the property and a finite word that is consistent with the observation that can be extended to satisfy the complement of the property. Consequently, we return *?*.

Theorem 2 *The algorithm described above can be implemented using zones and computes* $\mathcal{V}(\varphi, A)$.

Proof This follows immediately from Lemma 4, Lemma 5, and the fact that non-emptiness of intersections of sets of symbolic states can be solved algorithmically [16]. □

Note that our algorithm is online in the following sense: The set of nonempty states only needs to be computed once for each of the three automata and the symbolic states capturing

$$\mathcal{T}_{\mathcal{A}}(\mathcal{C}_{\mathcal{A}}((\phi_1, I_1, m_1)\cdots(\phi_n, I_n, m_n)(\phi_{n+1}, I_{n+1}, m_{n+1})), t')$$

can be computed from the symbolic states capturing

$$\mathcal{T}_{\mathcal{A}}(\mathcal{C}_{\mathcal{A}}((\phi_1, I_1, m_1)\cdots(\phi_n, I_n, m_n)), t),$$

as evident from the proof of Lemma 5. The same is true for the reach-sets in the other two automata $\mathcal{B}_{\neg\varphi}\otimes\mathcal{A}$ and $\mathcal{B}_{\varphi}\otimes\mathcal{A}$.

Remark 5 Our algorithm requires TBA both for the property *and* its complement, but TBA are in general not closed under complementation [11]. For the important case of MITL properties, such automata always exist, as MITL is closed under negation and can be translated into equivalent TBA (Theorem 1). In general, one needs both automata for the property and the complement of the property for monitoring of real-time properties [17].

Note that online monitoring as described above is not a decision problem per se. Instead, our algorithm continually updates reach-sets and intersects them with precomputed sets of states with non-empty language to determine verdicts. Furthermore, the number of zones making up the reach-sets can grow exponentially in the number of observations, even if all events are fully observable. Also, the number of zones constituting the reach-sets depends not only on the automata for the property being monitored, but also on the timestamps appearing in the observations.

Hence, we need an experimental evaluation to determine the true efficiency of our algorithm. To this end, we report in the next section on a prototype implementation, with a particular focus on the time it takes to compute the verdict after a new observation and the growth rate of the zones constituting the reach-sets.

5 Evaluation

We implemented our assumption-based online monitoring algorithm described in Section 4 by extending the UPPAAL tool component MoniTAal[2], thereby demonstrating how the use of assumptions and unobservable events can enhance monitoring capabilities. The tool can be utilized as a C++ library to monitor properties described by TBA. It supports intersection, so that given a TBA for the property, one for the negation of the property, as well as one for the assumption, one can create monitors for the assumption, and for the intersection of the assumption and the two property automata, respectively. The tool can then be used to update the reach-set of each automaton (triggered by (possibly inexact) observations) and detect if there is a non-empty intersection with the non-empty language states. In the following, we report on three proof-of-concept cases, that each demonstrate unique features of monitoring with assumptions.

- Task Sequence: This example demonstrates how utilizing assumptions can result in earlier verdicts. Furthermore, it shows the effect of unobservable events on the response time, the time between receiving an event and outputting a verdict.
- Conveyor Belt: This example shows that it is possible to monitor properties of unobservable propositions using assumptions.
- Jobshop Scheduling: This example demonstrates how the size of the automata (and the amount of nondeterministic choices in them) affects the monitoring response time and the number of zones constituting the reach-sets.

5.1 Task Sequence

We first experiment with a system under monitoring that produces a finite sequence of events $a_1, \ldots, a_k$. Each a_i, with $1 \leq i < k$, is followed by a_{i+1} with a time within the interval $[l_i, u_i]$. The assumption is formalized by the TBA shown

[2] https://github.com/DEIS-Tools/MoniTAal

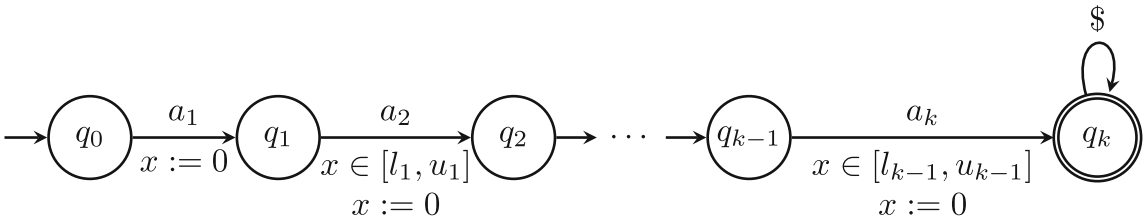


Fig. 5 A TBA representing the assumption for the bounded response example

in Figure 5. The domain is parameterized on k, and the l_i and u_i. Further, depending on the experiment, not all the a_i will be observable. We consider the bounded response property $G(a_1 \rightarrow F_{[0,B]}a_k)$. Suppose that we observe a timed word $(a_1, t_1) \cdots (a_k, t_k)$. If for some j in the range $1 < j \leq k$, we have $t_j + \sum_{j \leq i < k} u_i \leq B + t_1$, then the verdict at time t_j is $\top$. On the other hand, if $t_j + \sum_{j \leq i < k} l_i > B + t_1$, then the verdict at time t_j is $\bot$. As a corner case, if $\sum_{1 \leq i < k} u_i \leq B$ or $\sum_{1 \leq i < k} l_i > B$, the verdict is respectively $\top$ and $\bot$ at time 0, since all words of the assumption respectively satisfy and violate the property. We run several experiments to show the effect of the assumption and study the scalability under a sequence of unobservable events.

First we show how unobservable events can affect the response time, the time between receiving an event and outputting a verdict. We pick $k = 100$ and $l_i = 50$ and $u_i = 100$ for all i. The events $\{a_i \mid i \in \{21, \ldots, 40\} \cup \{61, \ldots, 80\}\}$ are unobservable within the interval [0, 10000]. In Figure 6 we see that for each consecutive unobservable event, the response time grows linearly. This is due to the reach-set growing. Nevertheless, the reach-set shrinks when an observable event is received. The minimum response time is 5 microseconds (μs), the maximum is 116 μs and the average is 25 μs. For reference, if we monitor 5000 consecutive unobservable events, the maximum response time is 32 milliseconds.

To show the effect of the assumption, we monitor the bounded response property $G(a_1 \rightarrow F_{[0,675]}a_{10})$ a thousand times, with and without the assumption where $l_i = 50$ and $u_i = 100$ for all i. The observed words are random, but within the assumption. The results in Table 1 show that verdicts are computed earlier with the assumption than without. Without the assumption the earliest verdicts were in 33 cases $\bot$ after observing a_9, while with the assumption we saw a $\top$ or $\bot$ verdict in 522 cases before observing a_{10}.

Thus, when monitoring a live system in an online setting (compared to evaluating a log history), a verdict can be reached earlier, because of the restrictions the assumption inhibits. Furthermore, we demonstrate in this case how unobservable events can affect the size of the reach-set, as the number of words that are consistent with an observation can increase with the number of consecutive unobservable events. This in turn affects the response time.

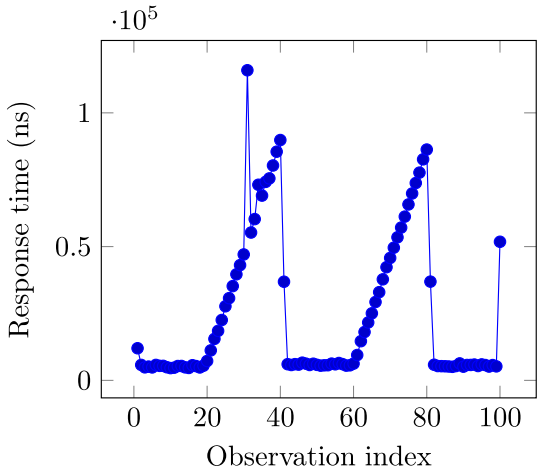


Fig. 6 Response time of the monitoring implementation, when monitoring the task sequence example with $\{a_i \mid i \in \{21, \ldots, 40\} \cup \{61, \ldots, 80\}\}$ being unobservable

Table 1 Verdict distribution for monitoring the task sequence example a thousand times with $k = 10$, $B = 675$, $l_i = 50$ and and $u_i = 100$ for all i, with and without assumption. For the definitive verdicts $\top$ and $\bot$, the entry in the row labeled a_i specifies the number of new verdicts given after the i-th observation; for the inconclusive verdict **?**, the entry specifies how many of the thousand example observations still yield that verdict

Observation	Verdicts					
	No Assumption			With Assumption		
	$\top$	$\bot$	**?**	$\top$	$\bot$	**?**
a_5	0	0	1000	0	0	1000
a_6	0	0	1000	0	1	999
a_7	0	0	1000	17	15	967
a_8	0	0	1000	81	90	796
a_9	0	33	967	165	153	478
a_{10}	0	457	510	246	232	0

5.2 Conveyor Belt

This example represents a conveyor belt that moves an item through different stations, where the item is processed according to some task. The task in the nominal case takes between 8 and 10 times units. However, if the process is faulty, it finishes earlier and takes between 7 and 9 time units:

it may sometimes complete correctly on time, but it may in other cases stop too early. The fault can happen at any time and is permanent. Our assumption automaton is shown in Figure 7. Our monitoring property is simply $G\neg fault$.

Consider that we observe the events $start$, $stop$, and $move$ with precise information on the time. If the $stop$ signal happens less than 8 time units after $start$, we detect a violation of the property. If instead $stop$ happens between 8 and 9 time units, we cannot say if there was a fault or not.

Suppose now that we have uncertainty on the time of the observations like in the following observation sequence:

$(start, [1, 1], = 1)(fault, [1, 11], \geq 0)(stop, [8, 10], = 1)$
$(fault, [8, 11], \geq 0)(move, [9, 11], = 1)$
$(fault, [9, 12], \geq 0)(start, [10, 12], = 1)$
$(fault, [11, 22], \geq 0)(stop, [16, 18], = 1)$

The first $stop$ happens at a time between 8 and 10, thus between 7 and 9 time units after the first $start$. This is compatible with both a nominal (with $stop$ occurring between times 9 and 10) and a faulty execution (with $stop$ occurring between times 8 and 9): after the first stop, we do not know if there was a fault.

The second $start$ happens in the time interval [10, 12] and has the same uncertainty: it is consistent with the nominal behavior if $start$ actually occurred in [11, 12] and with a faulty behavior if $start$ occurred in [10, 11]. The second stop happens in the time interval [16, 18]. Thus, the difference with the previous start is between 4 and 8 time units. This seems compatible with a nominal delay ([8, 10]). However, from the reasoning done above, if there were no fault the second start would have occurred in the interval [11, 12] and the second $stop$ would have occurred in the interval [19, 22], which is not compatible with the observation. Thus we can conclude there was a fault.

Using MoniTAAL, we monitored the property $G\neg fault$ with the assumption from Figure 7 by simulating a faulty conveyor belt. The observed trace is generated according to the simulated faulty behavior of the assumption, and the monitor observes only *start*, *stop*, and *move* signals, with a time uncertainty of 2. For example, if a *stop* occurs at time t in the simulation, then the monitor would observe $(stop, [l, u], = 1)$ for some randomly selected l and u such that $u - l = 2$ and $l \leq t \leq u$. As our observations are generated from the assumption, it is never violated. Furthermore, for the property $G\neg$*fault*, a monitor can never give the verdict $\top$. Thus, the only conclusive verdict we report is $\bot$ i.e., that the property does not hold.

The histogram in Figure 8 shows the distribution of the number of *start*, *stop*, and *move* signals that were observed before a fault was detected for 1000 runs. We see that the fault is often detected relatively early, even though there is a time uncertainty in every observation. The trace always begins with a *start* signal. Since the constraints on *start* are the same in the faulty behavior, a fault is never detected after just one observation, but a fault was detected after the second observation in 93 cases. The longest trace had 77 observations before a fault was detected.

With this example, we see how an assumption makes it possible to monitor properties over unobservable events (recall what we do not observe the signal *fault* referred to in the property). Without an assumption, reasoning about unobservable behavior would not be possible for such a property.

5.3 Jobshop Scheduling

In this example, we test the capability of the assumption-based online monitoring of scaling with the number of non-deterministic choices that the monitored system may take.

The assumption is intuitively the product composition of $n + 1$ automata depicted in Figure 9. These automata represent some processes that, within n time units, must lock a resource (either A or B) and take some time to complete the task. Thus, a process can use a resource only if no other process is already using it (this is intuitively represented by the guard $\neg A$ or $\neg B$). Process 0 takes n time units to complete, while the others only 1 time unit. We monitor the property $F_{[0,n]}done$, where *done* is the global state in which all tasks are completed.

In practice, the assumption is defined as a single timed automaton, which intuitively corresponds to the product of the automata depicted in Figure 9. This is shown in Figure 10 for the simple case of $n = 1$.

More precisely, the assumption automaton has $4^{(n+1)}$ states represented by a tuple $\langle s_0, s_1, \ldots, s_n\rangle$, where every $s_i \in \{I, A, B, D\}$, where I stands for idle, A for processing the task with resource A, B for processing with resource B, and D for done (corresponding to the locations p_i^I, p_i^A, p_i^B, and p_i^D in Figure 9). Thus, *done* is the state in which $s_i = D$, for all i, while the initial state is the state in which $s_i = I$ for all i. The automaton has $n + 1$ clocks x_i. The transitions are defined as follows:

- $\langle s_0, s_1, \ldots, I, \ldots, s_n\rangle \xrightarrow{\tau, x_i \leq n, \{x_i\}} \langle s_0, s_1, \ldots, A, \ldots, s_n\rangle$ switching s_i from I to A if $s_j \neq A$ for all j, with $0 \leq j \leq n$ and $j \neq i$;
- $\langle s_0, s_1, \ldots, I, \ldots, s_n\rangle \xrightarrow{\tau, x_i \leq n, \{x_i\}} \langle s_0, s_1, \ldots, B, \ldots, s_n\rangle$ switching s_i from I to B if $s_j \neq B$ for all j, with $0 \leq j \leq n$ and $j \neq i$;
- $\langle s_0, s_1, \ldots, s_n\rangle \xrightarrow{d_0, x_i \geq n} \langle D, s_1, \ldots, s_n\rangle$ switching s_0 to D if $s_0 \in \{A, B\}$;
- $\langle s_0, s_1, \ldots, s_n\rangle \xrightarrow{d_i, x_i \geq 1} \langle s_0, s_1, \ldots, D, \ldots, s_n\rangle$ switching s_i to D if $s_i \in \{A, B\}$ and $1 \leq i \leq n$.

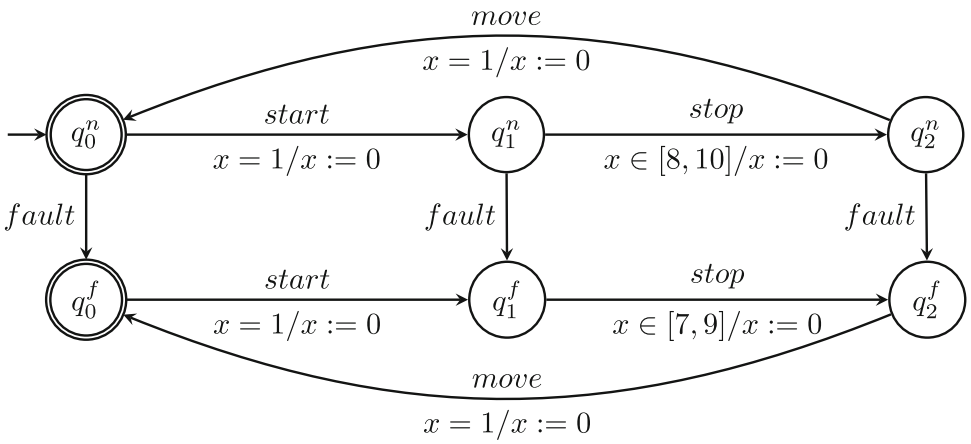


Fig. 7 A TBA representing the assumption for the conveyor belt example

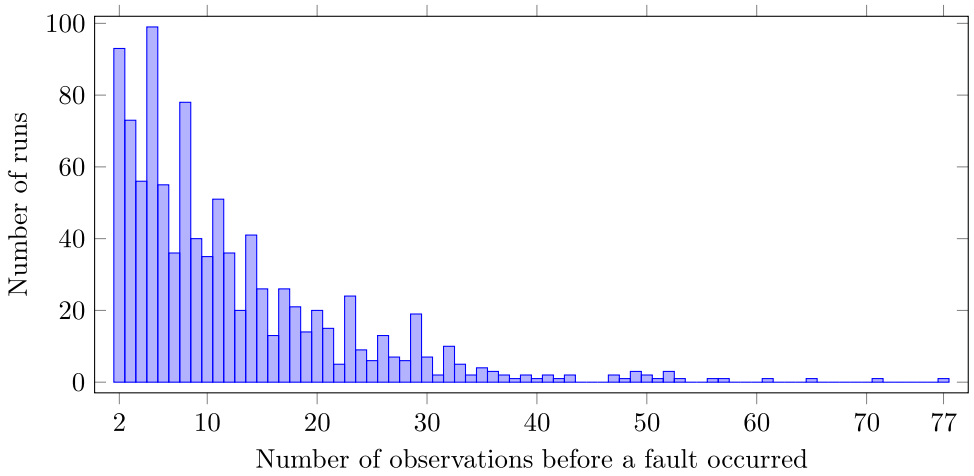


Fig. 8 Histogram of 1000 runs showing how many *start*, *stop*, and *move* events are observed before a fault is detected while monitoring a faulty conveyor belt

Let us suppose that τ is not observable and d_i is observable for all i, $0 \leq i \leq n$. Then $F_{[0,n]}done$ is monitorable as the monitor's output must be $\top$ after observing all d_i, one after the other, in any order, if the last process completes within n time units. The output must be $\bot$ if after n time units some d_i has not been observed.

Given the above assumption, we can detect a violation of the property in advance. In fact, note that the property can be satisfied only if process p_0 starts at time 0 choosing a resource and the other processes one after the other, starting at time 0, complete the task choosing the other resource. For example, with $n = 2$, a satisfying run would be

$$(\langle I, I, I\rangle, x_0 = 0, x_1 = 0, x_2 = 0) \xrightarrow{(\tau,0)}$$
$$(\langle I, A, I\rangle, x_0 = 0, x_1 = 0, x_2 = 0) \xrightarrow{(\tau,0)}$$
$$(\langle B, A, I\rangle, x_0 = 0, x_1 = 0, x_2 = 0) \xrightarrow{(d_1,1)}$$
$$(\langle B, D, I\rangle, x_0 = 1, x_1 = 1, x_2 = 1) \xrightarrow{(\tau,1)}$$
$$(\langle B, D, A\rangle, x_0 = 1, x_1 = 1, x_2 = 0) \xrightarrow{(d_2,2)}$$
$$(\langle B, D, D\rangle, x_0 = 2, x_1 = 2, x_2 = 1) \xrightarrow{(d_0,2)}$$
$$(\langle D, D, D\rangle, x_0 = 2, x_1 = 2, x_2 = 1).$$

Instead, if we observe d_1 and d_2 to occur before time 2, from the assumption, we can deduce that p_1 and p_2 used different resources. Thus, p_0 cannot complete within time n and the property will be violated.

For the experiments, we will monitor a satisfying observation of the form

$$(\tau, [0, n], \geq 0), (d_1, [1, 1], = 1),$$
$$(\tau, [0, n], \geq 0), (d_2, [2, 2], = 1), ..., (d_0, [n, n], = 1),$$

where $n+1$ is the number of jobs. Table 2 shows the results of monitoring such a satisfying observation for 2 to 9 jobs. The size of the assumption is the number of locations, the maximum response time is the longest time it took to compute a verdict after receiving an observation, and the maximum number of symbolic states refers to the size of the representation of the combined reach-sets for the property and assumption. We see that the increasing size of the assumption

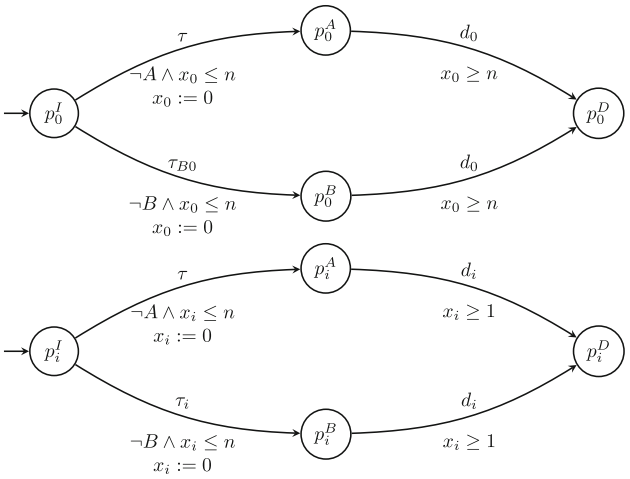


Fig. 9 A network of automata representing the assumption automaton of the jobshop scheduling problem. The upper automaton represents process 0, while the lower automaton represents process i, for $1 \leq i \leq n$

slows the response time significantly around 9 concurrent jobs, where the response time goes above a second.

6 Related Work

Our automata-based monitoring of finite words against specifications over infinite words follows the seminal work of Bauer et al. [1], who presented monitoring algorithms for LTL and timed LTL. Their algorithm for timed LTL is based on clock regions [11], while we follow the approach of Grosen et al. [5] and use clock zones [15], whose performance is an order of magnitude faster. Also, they translated timed LTL into event-clock automata, which are less expressive than the timed Büchi automata (TBA) used both by Grosen et al. [5] and here. This approach has also been applied to monitoring under delayed observations [18].

As our algorithms work with TBA, we also support MITL specifications, as these can be compiled into TBA. The monitoring problem for MITL has been investigated before. Baldor et al. showed how to construct a monitor for dense-time MITL formulas by constructing a tree of timed transducers [3]. Ho et al. split unbounded and bounded parts of MITL formulas for monitoring, using traditional LTL monitoring for the unbounded parts and permitting a simpler construction for the (finite-word) bounded parts [4].

There is also a large body of work on monitoring with finite-word semantics. Roşu et al. focussed on discrete-time finite-word MTL [19], while Basin et al. proposed algorithms for monitoring real-time finite-word properties [2] and compared different time models. Donzé et al. [20] focussed on monitoring a quantitative semantics for STL, a variant of MTL with predicates over real-valued signals. André et al.

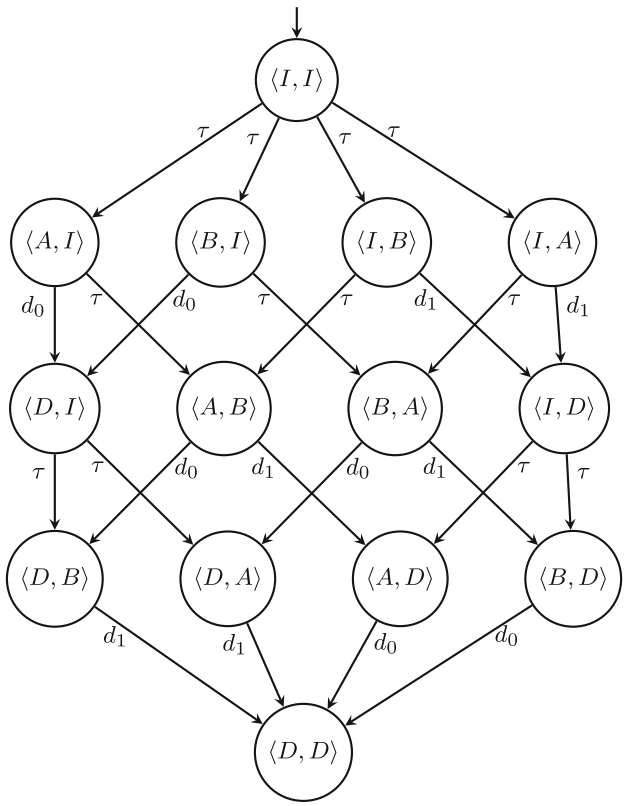


Fig. 10 Jobshop assumption for $n = 1$. Clocks guards, resets, and unreachable locations are omitted to improve readability

consider monitoring finite logs of parameterized timed and hybrid systems [21]. Finally, Ulus et al. described monitoring timed regular expressions over finite words using unions of two-dimensional zones [22, 23].

The contribution of this paper is focused on extending the monitoring of timed properties with assumptions, framing the problem as defined in [6–9] for the discrete-time setting. Assumptions were first used in [24] for extending the monitoring of LTL with predictive capabilities. In [25], the assumption for predictive RV is computed applying static analysis to the monitored program. Pinisetty et al. further extend the predictive RV idea to support RV of timed properties [26], where the *a priori knowledge* is also expressed as a timed property. As in [9], we adopt a four-valued semantics for timed properties and we support partial observability. Besides the complexity of moving from discrete to dense time semantics, the ABRV framework is extended with a rich notion of observations that take into account uncertainty on the time.

The research of partial observability in Discrete-Event Systems is usually connected with diagnosability [27] and predictability [28, 29]. These notions have been extended to timed systems (see, e.g., [30, 31]). Moreover, they are related to monitorability, an important topic in RV and other related fields [32–34], which has been studied taking into account

Table 2 Experimental results for monitoring a satisfying word of the jobshop scheduling example with 2 - 9 jobs. Response time is the maximal time it takes to compute a verdict after an observation. The number of symbolic states refers to the combined size of the reach-set of the property and assumption automata

Jobs	Size of Assumption	Max. resp. Time	Max. # Symbolic States
2	14	0.1 ms	9
3	44	0.3 ms	15
4	128	1.1 ms	21
5	352	2.8 ms	27
6	928	9.7 ms	33
7	2.368	38.3 ms	39
8	5.888	198.6 ms	45
9	14.336	1.294.6 ms	51

assumptions in [35]. Recently, the monitorability problem for real-time properties has been studied [17, 36].

7 Conclusion

We presented an approach to assumption-based runtime verification for real-time systems under partial observability. The method builds on Timed Automata to represent assumptions and uses Metric Interval Temporal Logic (or Timed Automata) to specify properties. A key element of the approach is the formalization of observations with both data and time uncertainty, allowing the monitor to reason about incomplete inputs. We proved that the monitoring function satisfies the properties of impartiality and anticipation, and we introduced a partial order on the verdicts to formalize their refinement. We developed a zone-based online monitoring algorithm and implemented it on top of UPPAAL. Experimental results illustrate how assumptions can enable earlier verdicts and allow monitoring of properties involving unobservable events.

There are several directions for future development. One is to improve scalability of the algorithm, especially when the assumption is specified as a network of automata. Another is to extend the output of the monitor to include richer diagnostic information. It would also be useful to investigate an epistemic characterization of the monitor specification, capturing what can be known from the observations. Finally, applying the approach in real-world settings would help assess its practical relevance and identify further challenges.

Supplementary information

The tool implementing the algorithms presented here is freely available at https://github.com/DEIS-Tools/MoniTAal.

Acknowledgements T.M. Grosen and K.G. Larsen have been funded by the Villum Investigator Grant S4OS. T.M. Grosen, K.G. Larsen, and M. Zimmermann have been supported by DIREC - Digital Research Centre Denmark. A. Cimatti and S. Tonetta have been supported by the PNRR project FAIR - Future AI Research (PE00000013), under the Italian NRRP MUR program funded by the NextGenerationEU.

Funding Open access funding provided by Aalborg University

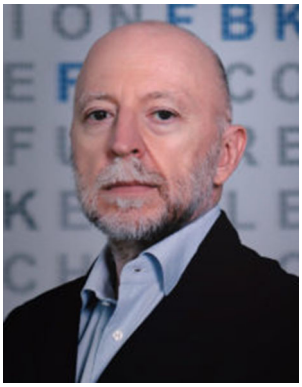
Declarations

Competing interests The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Bauer, A., Leucker, M., Schallhart, C.: Monitoring of real-time properties. In: Arun-Kumar, S., Garg, N. (eds.) FSTTCS 2006. LNCS, vol. 4337, pp. 260–272. Springer, (2006). https://doi.org/10.1007/11944836_25
2. Basin, D.A., Klaedtke, F., Zalinescu, E.: Algorithms for monitoring real-time properties. Acta Informatica **55**(4), 309–338 (2018). https://doi.org/10.1007/S00236-017-0295-4
3. Baldor, K., Niu, J.: Monitoring dense-time, continuous-semantics, metric temporal logic. In: Qadeer, S., Tasiran, S. (eds.) RV 2012. LNCS, vol. 7687, pp. 245–259. Springer, (2012). https://doi.org/10.1007/978-3-642-35632-2_24
4. Ho, H., Ouaknine, J., Worrell, J.: Online monitoring of metric temporal logic. In: Bonakdarpour, B., Smolka, S.A. (eds.) RV 2014. LNCS, vol. 8734, pp. 178–192. Springer, (2014). https://doi.org/10.1007/978-3-319-11164-3_15
5. Grosen, T.M., Kauffman, S., Larsen, K.G., Zimmermann, M.: Monitoring timed properties (revisited). In: Bogomolov, S., Parker, D. (eds.) FORMATS 2022. LNCS, vol. 13465, pp. 43–62. Springer, (2022). https://doi.org/10.1007/978-3-031-15839-1_3
6. Cimatti, A., Tian, C., Tonetta, S.: Assumption-based runtime verification with partial observability and resets. In: Finkbeiner, B.,

Mariani, L. (eds.) RV 2019. LNCS, vol. 11757, pp. 165–184. Springer, (2019). https://doi.org/10.1007/978-3-030-32079-9_10
7. Cimatti, A., Tian, C., Tonetta, S.: NuRV: A nuXmv extension for runtime verification. In: Finkbeiner, B., Mariani, L. (eds.) RV 2019. LNCS, vol. 11757, pp. 382–392. Springer, (2019). https://doi.org/10.1007/978-3-030-32079-9_23
8. Cimatti, A., Tian, C., Tonetta, S.: Assumption-based runtime verification of infinite-state systems. In: Feng, L., Fisman, D. (eds.) RV 2021. LNCS, vol. 12974, pp. 207–227. Springer, (2021). https://doi.org/10.1007/978-3-030-88494-9_11
9. Cimatti, A., Tian, C., Tonetta, S.: Assumption-based runtime verification. Formal Methods Syst. Des. **60**(2), 277–324 (2022). https://doi.org/10.1007/S10703-023-00416-Z
10. Cimatti, A., Grosen, T.M., Larsen, K.G., Tonetta, S., Zimmermann, M.: Exploiting assumptions for effective monitoring of real-time properties under partial observability. In: Madeira, A., Knapp, A. (eds.) SEFM 2024. LNCS, vol. 15280, pp. 70–88. Springer, (2024). https://doi.org/10.1007/978-3-031-77382-2_5
11. Alur, R., Dill, D.L.: A theory of timed automata. Theor. Comput. Sci. **126**(2), 183–235 (1994). https://doi.org/10.1016/0304-3975(94)90010-8
12. Alur, R., Feder, T., Henzinger, T.A.: The benefits of relaxing punctuality. J. ACM **43**(1), 116–146 (1996). https://doi.org/10.1145/227595.227602
13. Brihaye, T., Geeraerts, G., Ho, H., Monmege, B.: MightyL: A compositional translation from MITL to timed automata. In: Majumdar, R., Kuncak, V. (eds.) CAV 2017 (Part I). LNCS, vol. 10426, pp. 421–440. Springer, (2017). https://doi.org/10.1007/978-3-319-63387-9_21
14. Decker, N., Leucker, M., Thoma, D.: Impartiality and anticipation for monitoring of visibly context-free properties. In: Legay, A., Bensalem, S. (eds.) RV 2013. LNCS, vol. 8174, pp. 183–200. Springer, (2013). https://doi.org/10.1007/978-3-642-40787-1_11
15. Bengtsson, J., Yi, W.: Timed automata: Semantics, algorithms and tools. In: Desel, J., Reisig, W., Rozenberg, G. (eds.) Lectures on Concurrency and Petri Nets, Advances in Petri Nets. LNCS, vol. 3098, pp. 87–124. Springer, (2003). https://doi.org/10.1007/978-3-540-27755-2_3
16. Bellman, R.: Dynamic Programming. Princeton University Press, Princeton, NJ, USA (1957)
17. Grosen, T.M., Kauffman, S., Larsen, K.G., Zimmermann, M.: Time for timed monitorability. In: Bouyer, P., Pol, J. (eds.) CONCUR 2025. LIPIcs, vol. 348, pp. 19–11920. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, (2025). https://doi.org/10.4230/LIPICS.CONCUR.2025.19
18. Fränzle, M., Grosen, T.M., Larsen, K.G., Zimmermann, M.: Monitoring real-time systems under parametric delay. In: Kosmatov, N., Kovács, L. (eds.) IFM 2024. LNCS, vol. 15234, pp. 194–213. Springer, (2024). https://doi.org/10.1007/978-3-031-76554-4_11
19. Thati, P., Rosu, G.: Monitoring algorithms for metric temporal logic specifications. In: Havelund, K., Rosu, G. (eds.) RV@ETAPS 2004. ENTCS, vol. 113, pp. 145–162. Elsevier, (2004). https://doi.org/10.1016/J.ENTCS.2004.01.029
20. Donzé, A., Ferrère, T., Maler, O.: Efficient robust monitoring for STL. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 264–279. Springer, (2013). https://doi.org/10.1007/978-3-642-39799-8_19
21. Waga, M., André, É., Hasuo, I.: Model-bounded monitoring of hybrid systems. ACM Trans. Cyber Phys. Syst. **6**(4), 30–13026 (2022). https://doi.org/10.1145/3529095
22. Ulus, D., Ferrère, T., Asarin, E., Maler, O.: Timed pattern matching. In: Legay, A., Bozga, M. (eds.) FORMATS 2014. LNCS, vol. 8711, pp. 222–236. Springer, (2014). https://doi.org/10.1007/978-3-319-10512-3_16
23. Ulus, D., Ferrère, T., Asarin, E., Maler, O.: Online timed pattern matching using derivatives. In: Chechik, M., Raskin, J. (eds.) TACAS 2016. LNCS, vol. 9636, pp. 736–751. Springer, (2016). https://doi.org/10.1007/978-3-662-49674-9_47
24. Leucker, M.: Sliding between model checking and runtime verification. In: Qadeer, S., Tasiran, S. (eds.) RV 2012. LNCS, vol. 7687, pp. 82–87. Springer, (2012). https://doi.org/10.1007/978-3-642-35632-2_10
25. Zhang, X., Leucker, M., Dong, W.: Runtime verification with predictive semantics. In: Goodloe, A., Person, S. (eds.) NFM 2012. LNCS, vol. 7226, pp. 418–432. Springer, (2012). https://doi.org/10.1007/978-3-642-28891-3_37
26. Pinisetty, S., Jéron, T., Tripakis, S., Falcone, Y., Marchand, H., Preoteasa, V.: Predictive runtime verification of timed properties. J. Syst. Softw. **132**, 353–365 (2017). https://doi.org/10.1016/J.JSS.2017.06.060
27. Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., Teneketzis, D.: Diagnosability of discrete-event systems. IEEE Trans. Autom. Control **40**(9), 1555–1575 (1995). https://doi.org/10.1109/9.412626
28. Genc, S., Lafortune, S.: Predictability of event occurrences in partially-observed discrete-event systems. Autom. **45**(2), 301–311 (2009). https://doi.org/10.1016/J.AUTOMATICA.2008.06.022
29. Genc, S., Lafortune, S.: Predictability in discrete-event systems under partial observation. In: Zhang, H.-Y. (ed.) Fault Detection, Supervision and Safety of Technical Processes 2006, pp. 1461–1466. Elsevier Science Ltd, Oxford (2007). https://doi.org/10.1016/B978-008044485-7/50245-1
30. Cassez, F., Grastien, A.: Predictability of event occurrences in timed systems. In: Braberman, V.A., Fribourg, L. (eds.) FORMATS 2013. LNCS, vol. 8053, pp. 62–76. Springer, (2013). https://doi.org/10.1007/978-3-642-40229-6_5
31. Cassez, F., Tripakis, S.: Fault diagnosis of timed systems. In: Jard, C., Roux, O.H. (eds.) Communicating Embedded Systems, pp. 107–138. Wiley, (2013). https://doi.org/10.1002/9781118558188.CH4
32. Aceto, L., Achilleos, A., Francalanza, A., Ingólfsdóttir, A., Lehtinen, K.: An operational guide to monitorability. In: Ölveczky, P.C., Salaün, G. (eds.) SEFM 2019. LNCS, vol. 11724, pp. 433–453. Springer, (2019). https://doi.org/10.1007/978-3-030-30446-1_23
33. Sistla, A.P., Zefran, M., Feng, Y.: Monitorability of stochastic dynamical systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 720–736. Springer, (2011). https://doi.org/10.1007/978-3-642-22110-1_58
34. Peled, D., Havelund, K.: Refining the safety-liveness classification of temporal properties according to monitorability. In: Margaria, T., Graf, S., Larsen, K.G. (eds.) Models, Mindsets, Meta: The What, the How, and the Why Not? - Essays Dedicated to Bernhard Steffen on the Occasion of His 60th Birthday. LNCS, vol. 11200, pp. 218–234. Springer, (2018). https://doi.org/10.1007/978-3-030-22348-9_14
35. Henzinger, T.A., Saraç, N.E.: Monitorability under assumptions. In: Deshmukh, J., Nickovic, D. (eds.) RV 2020. LNCS, vol. 12399, pp. 3–18. Springer, (2020). https://doi.org/10.1007/978-3-030-60508-7_1
36. Amara, M., Bernardi, G., Foughali, M.A., Francalanza, A.: A theory of (linear-time) timed monitors. In: Aldrich, J., Silva, A. (eds.) ECOOP 2025. LIPIcs, vol. 333, pp. 1–1130. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, (2025). https://doi.org/10.4230/LIPICS.ECOOP.2025.1



Alessandro Cimatti is the director of the Center for Digital Industry at Fondazione Bruno Kessler in Trento, Italy. His research interests include formal verification, model checking, temporal logics, runtime verification, safety assessment, diagnosis and planning. Cimatti authored more than 250 papers in the fields of formal methods and artificial intelligence. For his fundamental works on Bounded Model Checking and on Satisfiability Modulo Theories, Cimatti has received the TACAS 2014 Most Influential Paper award, the ETAPS 2017 Test of Time award and the CAV Award in 2018 and 2021. He has been the leader of several technology transfer projects, including research projects funded by the European Union, the European Space Agency and the European Railway Agency, as well as of industrial collaborations with RFI, SAIPEM and Boeing.

Thomas M. Grosen is a postdoc at Aalborg University. His interests are in verification and monitoring of real-time systems, automata theory and semantics, as well as implementing it in practice. During his PhD study, under the supervision of Kim G. Larsen and Martin Zimmermann, he made several contributions in runtime verification of real-time systems both in theory and practice. During this time he recieved a Mitacs globalink research award to visit Queens University in Canada, under the supervision of Sean Kauffman, to study timed monitorability.

Kim G. Larsen is a Professor in Computer Science at Aalborg University since 1993. He obtained his PhD in Computer Science at Edinburgh University 1986 under the supervision of Professor Robin Milner. His research interests span semantics, verification and logic, concurrency theory, performance analysis of real-time, embedded and cyber-physical systems, model checking and machine learning, as well as quantum systems. He is the prime investigator of the model checking tool UPPAAL. He is the founding member of the conference TACAS and the conference series ETAPS. He is life-long member of the Royal Danish Academy of Sciences and Letters, Copenhagen, is member of the Danish Academy of Technical Sciences. He became Honorary Doctor (Honoris causa) at Uppsala University, Sweden, in 1999 for his contributions to the popular verification tool UPPAAL. In 2007 he became Knight of the Order of the Dannebrog. In 2007 he became Honorary Doctor (Honoris causa) at ENS Cachan, France. In 2012 he became Honary Member of Academia Europaea. In 2013 he was the recipient of the CAV Award for his work UPPAAL "the foremost model checker for real-time Systems". In 2015 he won an ERC Advanced Grant (LASSO). He received the Grundfos prize 2016, became Foreign Expert of China, Distinguished Professor, Northeastern University, 2018, and received the Test-Of-Time Award, 2022. In 2021 he won the VILLUM Investigator Grant S4OS.

Stefano Tonetta is currently the head of the Unit on Formal Methods for Systems and Software Design of the Fondazione Bruno Kessler, which develops tools and techniques on formal methods and model-based design, including the tools nuXmv, OCRA, and xSAP. With a PhD in Information and Telecommunication from the University of Trento, Tonetta has focused most of his research on the development of formal methods and automated verification techniques. He has authored more than hundred papers, with contributions on model checking, temporal logics, and safety-critical systems. He participated in several European projects and lead different studies such as OthelloPlay, awarded by Microsoft Software Engineering Innovations Foundation in 2012. He co-chaired various conferences such as SAFECOMP23, TIME22, and VSTTE22, and participated in several program committees.

Martin Zimmermann is an associate professor at the Department of Computer Science of Aalborg University. He is interested in temporal logics, automata theory, and infinite games as well as their application to verification problems like monitoring, model-checking, and reactive synthesis. He has published about 80 papers on these topics and serves currently in the steering committee of the International Symposium on Games, Automata, Logics, and Formal Verification and the summer school series Modelling and Verification of Parallel Processes.