Abstract Monitors for Quantitative Specifications

Thomas A. Henzinger, Nicolas Mazzocchi, and N. Ege Saraç

Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria
`{tah,nmazzocc,esarac}@ist.ac.at`

Abstract. Quantitative monitoring can be universal and approximate: For every finite sequence of observations, the specification provides a value and the monitor outputs a best-effort approximation of it. The quality of the approximation may depend on the resources that are available to the monitor. By taking to the limit the sequences of specification values and monitor outputs, we obtain precision-resource trade-offs also for limit monitoring. This paper provides a formal framework for studying such trade-offs using an abstract interpretation for monitors: For each natural number n, the aggregate semantics of a monitor at time n is an equivalence relation over all sequences of at most n observations so that two equivalent sequences are indistinguishable to the monitor and thus mapped to the same output. This abstract interpretation of quantitative monitors allows us to measure the number of equivalence classes (or "resource use") that is necessary for a certain precision up to a certain time, or at any time. Our framework offers several insights. For example, we identify a family of specifications for which any resource-optimal exact limit monitor is independent of any error permitted over finite traces. Moreover, we present a specification for which any resource-optimal approximate limit monitor does not minimize its resource use at any time.

Keywords: Abstract monitor · Approximate monitoring · Quantitative monitoring · Monitor resources

1 Introduction

Online monitoring is a runtime verification (RV) technique [11] that, by sacrificing completeness, aims to lighten the burden caused by exhaustive formal methods. A monitor watches an unbounded sequence f of observations, called trace, one observation at a time. At each time $n \geq 0$, it tries to provide information about the value assigned to f by the specification. For a boolean specification P, after each trace prefix s, the monitor may output one of three values: all infinite extensions of s satisfy P, violate P, or neither [15].

Quantitative specifications [21] generalize their boolean analogs by assigning each trace f a value from some richer domain. For example, the boolean specification Resp assigns *true* to f iff every observation `req` in f is eventually followed by an observation `ack` in f, while the quantitative specification MaxRespTime assigns the least upper bound on the number of observations between each `req` and the corresponding `ack`, or ∞ if there is no such upper bound.

In the limit monitoring of a quantitative specification Φ over a trace f, a limit (e.g., $\limsup$, $\liminf$) of the infinite sequence of monitor outputs should provide information about the value $\Phi(f)$ assigned to the trace. For example, a "natural way to monitor" $\mathsf{MaxRespTime}$ is to have the monitor output, at each time, the maximum of (i) the maximal response time so far and (ii) the time since the least recent pending `req`, if there is a pending `req`. The $\limsup$ (and $\liminf$) of this infinite output sequence converges towards $\mathsf{MaxRespTime}$.

In contrast to its boolean analog, the quantitative setting naturally supports approximation. A monitor has error $\delta \geq 0$ if, for all infinite traces, the limit of the output sequence is within δ of the specification value. In particular, this leads to precision-resource trade-offs for quantitative monitors: The provisioning of additional states, registers, or operations may reduce the error, and a larger error tolerance may enable monitors that use fewer resources.

In this paper, we provide a formal framework for studying such precision-resource trade-offs for an abstract definition of quantitative monitors. This abstract framework can be instantiated, for example, by finite-state monitors or register monitors, where a finite-state monitor remembers a bounded amount of information about each trace prefix, and a register monitor remembers a bounded number of integer values [32]. For us, an *abstract monitor* partitions, at each time n, all prefixes of length up to n into a finite number of equivalence classes such that if two prefixes s_1 and s_2 are equivalent, then the monitor outputs the same value after observing s_1 and s_2. The number of equivalence classes introduced at time n provides a natural measure for the resource use of the abstract monitor after n observations.

In this setting, where the *resource use* of a monitor is measured, we also want to measure the *precision* of a monitor. To define the precision of a monitor after a finite trace prefix, we need to enrich our definition of quantitative specifications: We let a quantitative specification assign values not only to infinite traces but also to finite traces. Indeed, many specification values for infinite traces are usually defined as limits [37]. For example, what we called above the "natural way to monitor" $\mathsf{MaxRespTime}$ using two counters is, in fact, the usual formal definition of the quantitative specification $\mathsf{MaxRespTime}$.

Once both specifications and monitors assign values to all finite traces, there is a natural definition for the precision of a monitor: At each time n, the *prompt-error* is the maximal difference between the monitor output and the specification value over all finite traces of length up to n. Furthermore, the *limit-error* is the least upper bound on the difference between the limit of monitor outputs and the limit of specification values over all infinite traces. Note that if the prompt-error of a monitor is 0, then so is the limit-error, but not necessarily vice versa. An exact-value monitor (i.e., a monitor with prompt-error $\delta = 0$) implements the specification as it is defined. In contrast, an approximate monitor (i.e., a monitor with prompt-error $\delta > 0$) of the same specification may use fewer resources.

An approximate monitor may still achieve limit-error 0, which is a situation of particular interest that we study.

Given an abstract monitor, one way to obtain a new monitor that uses fewer resources use is to merge some equivalence classes, and one way to increase the precision is to split some equivalence classes. However, this naive approach toward reaching a desired precision or resource use is not always the best. For an approximate monitor with a given prompt-error and limit-error, the goal is *resource-optimality*, i.e., minimizing the resource use as much as the error threshold allows. We will see that merging the equivalence classes of a given monitor may not yield a resource-optimal one.

The limit-error of a monitor is bounded by its prompt-error. We also investigate the case where we require a certain limit-error while leaving the prompt-error potentially unbounded. We will see that allowing arbitrary prompt-error may not permit the monitor to save resources if the desired limit-error is fixed. We say that such specifications have *resource-intensive limit behavior*. In fact, MaxRespTime displays resource-intensive limit behavior. Other examples include a subclass of *reversible specifications*. Reversibility is a notion from automata theory characterized by the specification being realizable with a finite-state automaton that is both forward and backward deterministic. A similar notion, generalized to the quantitative setting, can be introduced in our framework, allowing an abstract monitor to process an infinite trace in a two-way fashion.

Overview Section 2 formalizes the framework of abstract monitors and provides insights on relations between basic notions such as resource use and precision.

Section 3 focuses on monitoring with bounded error over finite traces. First, in Subsection 3.1, we show that the exact-value monitor over finite traces is unique and resource-optimal for every specification. Additionally, for resource-optimal approximate monitors, we prove: (i) they are not unique in Subsection 3.1, (ii) they do not necessarily follow the structure of the exact-value monitor in Subsection 3.2, and (iii) they do not necessarily minimize their resource use at each time in Subsection 3.2. Then, in Subsection 3.3, we study precision-resource trade-off suitability: We exhibit (i) a specification for which we can arbitrarily improve the resource use by damaging precision, and (ii) another for which we arbitrarily improve the precision by damaging the resource use.

Section 4 focuses on monitoring without error on infinite traces. In particular, in Subsection 4.1 we provide a condition for identifying specifications with resource-intensive limit behavior, for which having zero limit-error prevents the trade-off between resource use and error on finite traces. This condition captures two paradigmatic specifications: (i) maximal response-time and (ii) average response-time. Finally, in Subsection 4.2 we investigate reversible specifications, which can be implemented in a manner both forward and backward deterministic. A subclass of reversible specifications have resource-intensive limit behavior, which we demonstrate through the average ping specification.

Section 5 concludes the paper and addresses future research directions our framework offers.

Related work In the boolean setting, several notions of monitorability have been proposed over the years [15,30,34]. Much of the theoretical efforts have focused on regular specifications [2,14,46], although some proposed more expressive models [9,12,26]. We refer the reader to [10] for coverage of these and more.

Verification of quantitative specifications [21,41] have received significant attention, especially in the probabilistic setting [17,20,33]. In the context of RV, the literature on specifications with quantitative aspects features primarily metric temporal logic and signal temporal logic [38,40,43,44,45]. Other efforts include processing data streams with a focus on deciding their properties at runtime [5,6] and an extension of weighted automata with monitor counters [22]. None of these works focus on monitoring quantitative specifications with approximate verdicts or the relation between monitorability and monitor resources.

Approximate methods have been used in verification for many years [25,39]. Beyond the boolean setting, such approaches have appeared in the context of sensor networks for approximating aggregate functions in a distributed setting [24,49,50], in approximate determinization or minimization of quantitative models of computation [7,16,35], and also in online algorithms [3].

To the best of our knowledge, the use of approximate methods in monitoring mainly concentrates on the specification rather than taking approximateness as a monitor feature and studying the quality of monitor verdicts. In predictive or assumption-based monitoring [23,54] and for monitoring hyperproperties [51], an over-approximation of the system under observation is used as an assumption to limit the set of possible traces [36]. Similarly, in runtime quantitative verification [18,47], the underlying probabilistic model of the system is approximated and continually updated. For monitoring under partial observability, [4] describes an approach to approximate the given specification for minimizing the number of undetected violations. In the branching-time setting, [1] uses a monitorable under- or over-approximation of the given specification to construct an "optimal" monitor. Nonetheless, a form of distributed and approximate limit monitoring for spatial specifications was studied in [8]. None of these works consider approximateness as a monitor property to study the relation between monitor resources and the quality of its verdicts.

Recently, [32] introduced a concrete monitor model with integer-valued registers and studied their resource needs. This model was later used for limit monitoring of statistical indicators of traces under probabilistic assumptions [31]. A general framework for approximate limit monitoring of quantitative specifications was proposed in [37]. However, that framework focuses exclusively on limit behaviors and on specific monitor models such as finite automata and register machines, thus allowing only limited precision-cost analyses. The main innovations of the present work over previous work are twofold. First, we abstract the monitor model and its resource use away from specific machine models. Second, by introducing prompt-errors, we study the resource use of monitors over time and relate this to the monitoring precision over time. This more nuanced framework enables a more fine-grained analysis and comparison of different monitors for the same specification concerning their precision and resource use.

2 Definitional Framework

Let $\Sigma = \{a, b, \ldots\}$ be a finite alphabet of observations. A *trace* is finite or infinite sequence of observations, which we respectively denote by $s, r, t \in \Sigma^*$ and $f, g \in \Sigma^\omega$. Given two traces $s \in \Sigma^*$ and $w \in \Sigma^* \cup \Sigma^\omega$, we denote by $s \prec w$ (resp. $s \preceq w$) that s is a strict (resp. non-strict) prefix of w. For $n \in \mathbb{N}$ we define $\Sigma^{\leq n} = \{s \in \Sigma^* \mid |s| \leq n\}$ where $|s|$ refers to the length of s. Given $a \in \Sigma$ and $s \in \Sigma^*$, we denote by $|s|_a$ the number of occurrences of a in s.

We denote by $\mathbb{N}$ the set of *non-negative integers* and by $\mathbb{R}$ the set of *real numbers*. We also consider $\overline{\mathbb{N}} = \mathbb{N} \cup \{+\infty\}$ and $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$.

A binary relation $\sim$ over Σ^* is an *equivalence relation* when it is reflexive, symmetric, and transitive. For a given equivalence relation $\sim$ over Σ^* and a finite trace $s \in \Sigma^*$, we denote by $[s]_\sim$ the equivalence class of $\sim$ in which s belongs. When $\sim$ is clear from the context, we write $[s]$ instead. A *right-monotonic* relation $\sim$ over Σ^* fulfills $s_1 \sim s_2 \Rightarrow s_1 r \sim s_2 r$ for all $s_1, s_2, r \in \Sigma^*$.

We use $\Box$ and $\Diamond$ to denote the linear temporal logic (LTL) operators *always* and *eventually*, respectively. See [48] for interpretation of LTL operators on infinite traces, and [27,19,29] on finite traces.

2.1 Quantitative specifications

A *limit-measure* is a function from $\overline{\mathbb{R}}^\omega$ to $\overline{\mathbb{R}}$. Given an infinite sequence of real numbers $x = x_1 x_2 \ldots$, we define $\liminf(x) = \lim_{n \mapsto +\infty} \inf\{x_i \mid i \geq n\}$ and $\limsup(x) = \lim_{n \mapsto +\infty} \sup\{x_i \mid i \geq n\}$. Whenever $\liminf(x) = \limsup(x)$ for a given sequence x, we simply write $\lim(x)$. A *value function* $\pi \colon \Sigma^* \to \overline{\mathbb{R}}$ associates a value to each finite trace.

Definition 1 (specification). *A* specification *extends a value function by constraining its limit behavior. Syntactically, it is a tuple $\Phi = (\pi, \ell)$ where $\pi \colon \Sigma^* \to \overline{\mathbb{R}}$ is a value function and ℓ is a limit-measure. Semantically, it is a function defined by $[\![\Phi]\!](s) = \pi(s)$ when $s \in \Sigma^*$ and $[\![\Phi]\!](f) = \ell(\pi(f))$ when $f \in \Sigma^\omega$, where $\pi(f) = (\pi(s_i))_{i \in \mathbb{N}}$ is a sequence over the prefixes $s_i \prec f$ of increasing length i.*

Together with a given specification Φ, we define the right-monotonic equivalence relation $\sim_\Phi^*$ as follows. For all $s_1, s_2 \in \Sigma^*$ we have $s_1 \sim_\Phi^* s_2$ iff $\pi(s_1 r) = \pi(s_2 r)$ holds for all $r \in \Sigma^*$.

We define below the *discounted response* specification. Throughout the section, we will use this specification as a running example.

Example 2. Let $\Sigma = \{\texttt{req}, \texttt{ack}, \texttt{other}\}$ and consider the LTL response specification $P = \Box(\texttt{req} \to \Diamond\texttt{ack})$. Let $0 < \lambda < 1$ be a discount factor. We define $\mathsf{DiscResp}(s) = 1$ if $s \in P$, and $\mathsf{DiscResp}(s) = \lambda^n$ otherwise, where $n = |s| - |r|$ and $r \prec s$ is the longest prefix of s with $r \in P$. We define $\Phi_{\mathsf{DR}} = (\mathsf{DiscResp}, \limsup)$, the *discounted response* specification. Intuitively, Φ_{DR} assigns each finite trace a value that shows how close the system behaves to P such that, at the limit, it denotes whether the infinite behavior satisfies P or not.

Now, take two traces $s, r \in \Sigma^*$. We claim that $s \sim^*_{\Phi_{\mathsf{DR}}} r$ iff either (i) both traces have no pending request or (ii) both have a request pending for the same number of steps. First, we assume $s \sim^*_{\Phi_{\mathsf{DR}}} r$ holds and note that we must have $\Phi_{\mathsf{DR}}(st) = \Phi_{\mathsf{DR}}(rt)$ for every $t \in \Sigma^*$. Then, we eliminate the cases other than (i) and (ii) as follows. If, w.l.o.g., $s \in P$ and $r \notin P$, then $\Phi_{\mathsf{DR}}(r) < \Phi_{\mathsf{DR}}(s) = 1$, thus $s \not\sim^*_{\Phi_{\mathsf{DR}}} r$. If, w.l.o.g., s has a request pending for i steps and r for $j > i$ steps, then $\Phi_{\mathsf{DR}}(r) = \lambda^j < \lambda^i = \Phi_{\mathsf{DR}}(s)$, thus $s \not\sim^*_{\Phi_{\mathsf{DR}}} r$. The other direction is similar.

2.2 Abstract monitors

We are now ready to present our abstract definition of quantitative monitors.

Definition 3 (monitor). *A* monitor $\mathcal{M} = (\sim, \gamma)$ *is a tuple consisting of a right-monotonic equivalence relation* $\sim$ *on* Σ^* *and a function* $\gamma \colon (\Sigma^*/\sim) \to \overline{\mathbb{R}}$. *Let* $\delta_{\text{fin}}, \delta_{\text{lim}} \in \overline{\mathbb{R}}$ *be error thresholds. We say that* $\mathcal{M}$ *is a* $(\delta_{\text{fin}}, \delta_{\text{lim}})$-monitor *for a given specification* $\Phi = (\pi, \ell)$ *iff*

- $|\pi(s) - \gamma([s])| \leq \delta_{\text{fin}}$ *for all* $s \in \Sigma^*$, *and*
- $|\ell(\pi(f)) - \ell(\gamma([f]))| \leq \delta_{\text{lim}}$ *for all* $f \in \Sigma^\omega$.

where $\gamma([f]) = (\gamma([s_i]))_{i \in \mathbb{N}}$ *is a sequence over the prefixes* $s_i \prec f$ *of increasing length* i. *We say that* $\mathcal{M}$ *has a* prompt-error *of* δ_{fin} *and a* limit-error *of* δ_{lim}.

We conveniently write $\mathcal{M}(s) = \gamma([s])$ when $s \in \Sigma^*$ and $\mathcal{M}(f) = \ell(\gamma([f]))$ when $f \in \Sigma^\omega$.

Observe that, for every specification, there is an obvious monitor that imitates exactly the specification, which we define as follows.

Definition 4 (exact-value monitor). *Let* $\Phi = (\pi, \ell)$ *be a specification. The* exact-value monitor *of* Φ *is defined as* $\mathcal{M}_\Phi = (\sim^*_\Phi, s \mapsto \pi(s))$.

A monitor for a given specification is *approximate* when it differs from the specification's exact-value monitor. Below we demonstrate the exact-value monitor and an approximate monitor for the discounted response specification.

Example 5. Recall from Example 2 the discounted response specification Φ_{DR}. Clearly, its exact-value monitor is $\mathcal{M}_{\Phi_{\mathsf{DR}}} = (\sim^*_{\Phi_{\mathsf{DR}}}, \gamma_{\Phi_{\mathsf{DR}}})$ where $\gamma_{\Phi_{\mathsf{DR}}}([s]) = \Phi_{\mathsf{DR}}(s)$ for all $s \in \Sigma^*$. Let us define another monitor $\mathcal{M} = (\sim, \gamma)$ such that $s \sim r$ iff either $s, r \in P$ or $s, r \notin P$ for every $s, r \in \Sigma^*$; and $\gamma([s]) = 1$ if $s \in P$, and $\gamma([s]) = 0$ if $s \notin P$. Note that for every $f \in \Sigma^\omega$ we have $f \in P$ iff infinitely many prefixes of f belong to P, therefore $\mathcal{M}$ has no limit-error. However, it yields a prompt-error of λ since it immediately outputs 0 instead of discounting on finite traces. Hence, $\mathcal{M}$ is a $(\lambda, 0)$-monitor for Φ_{DR}.

Next, we prove that our definition constrains monitors not to make two equivalent traces too distant.

Proposition 6. *Let* $\mathcal{M} = (\sim, \gamma)$ *be a* $(\delta_{\text{fin}}, \delta_{\text{lim}})$*-monitor for the specification* $\Phi = (\pi, \ell)$. *For all* $s_1, s_2 \in \Sigma^*$, *if* $s_1 \sim s_2$, *then* $|\Phi(s_1) - \Phi(s_2)| \leq 2\delta_{\text{fin}}$.

Proof. By definition of $\mathcal{M}$ we have that $-\delta_{\text{fin}} \leq \pi(s_1) - \gamma([s_1]) \leq \delta_{\text{fin}}$ as well as $\delta_{\text{fin}} \geq -\pi(s_2) + \gamma([s_2]) \geq -\delta_{\text{fin}}$. If $s_1 \sim s_2$ then $\gamma([s_1]) = \gamma([s_2])$ and thus $-2\delta_{\text{fin}} \leq \pi(s_1) - \pi(s_2) \leq 2\delta_{\text{fin}}$. □

2.3 Resource use of abstract monitors

As we demonstrated above, quantitative monitors may have different degrees of precision. A natural question is whether monitors with different error thresholds use a different amount of resources. To answer this question in its generality, we consider the following model-oblivious notions of resource use.

Definition 7 (resource use). *Let $\mathcal{M} = (\sim, \gamma)$ be a monitor. We consider two notions of resource use for $\mathcal{M}$ defined as functions from $\mathbb{N}$ to $\mathbb{N}$. We define* step-wise resource use *as* $\mathbf{r}_n(\mathcal{M}) = |\Sigma^{\leq n}/{\sim}| - |\Sigma^{<n}/{\sim}|$, *and* total resource use *as* $\mathbf{R}_n(\mathcal{M}) = \sum_{i=0}^{n} \mathbf{r}_i(\mathcal{M}) = |\Sigma^{\leq n}/{\sim}|$.

Given two monitors $\mathcal{M}_1$ and $\mathcal{M}_2$, we compare their resource use as follows. We write $\mathbf{r}(\mathcal{M}_1) < \mathbf{r}(\mathcal{M}_2)$ when there exists $n_0 \in \mathbb{N}$ such that for every $n \geq n_0$ we have $\mathbf{r}_n(\mathcal{M}_1) < \mathbf{r}_n(\mathcal{M}_2)$. In particular, when it holds for $n_0 = 1$, we write $\mathbf{r}(\mathcal{M}_1) \ll \mathbf{r}(\mathcal{M}_2)$. We define $\mathbf{R}(\mathcal{M}_1) < \mathbf{R}(\mathcal{M}_2)$ and $\mathbf{R}(\mathcal{M}_1) \ll \mathbf{R}(\mathcal{M}_2)$ similarly. Fig. 1 shows how these notions relate. Moreover, definitions of $\mathbf{r}(\mathcal{M}_1) \propto \mathbf{r}(\mathcal{M}_2)$ and $\mathbf{R}(\mathcal{M}_1) \propto \mathbf{R}(\mathcal{M}_2)$ for $\propto \in \{\leq, \underline{\ll}, >, \gg, \geq, \underline{\gg}\}$ are as expected.

The monitor $\mathcal{M}_1$ uses *at most as many* resources as $\mathcal{M}_2$ when we have $\mathbf{r}(\mathcal{M}_1) \mathrel{\underline{\ll}} \mathbf{r}(\mathcal{M}_2)$. If we further have $\mathbf{r}_n(\mathcal{M}_1) < \mathbf{r}_n(\mathcal{M}_2)$ for some $n \geq 1$, then $\mathcal{M}_1$ uses *fewer* resources than $\mathcal{M}_2$. We similarly define the cases for using *at least as many* and *more* resources.

Given a specification Φ and a $(\delta_{\text{fin}}, \delta_{\text{lim}})$-monitor $\mathcal{M}$ for Φ, we say that $\mathcal{M}$ is *resource-optimal* for Φ when for every $(\delta_{\text{fin}}, \delta_{\text{lim}})$-monitor $\mathcal{M}'$ for Φ we have $\mathbf{r}(\mathcal{M}) \mathrel{\underline{\ll}} \mathbf{r}(\mathcal{M}')$, i.e., $\mathcal{M}$ uses at most as many resources as any other monitor $\mathcal{M}'$ with the same error thresholds.

Example 8. Recall from Examples 2 and 5 the discounted response specification Φ_{DR}, its exact-value monitor $\mathcal{M}_{\Phi_{\mathsf{DR}}}$, and the $(\lambda, 0)$-monitor $\mathcal{M}$. We claim that $\mathcal{M}$ uses fewer resources than $\mathcal{M}_{\Phi_{\mathsf{DR}}}$. To show this, we first point out that $\mathbf{r}_0(\mathcal{M}) = \mathbf{r}_1(\mathcal{M}) = 1$ and $\mathbf{r}_n(\mathcal{M}) = 0$ for every $n \geq 2$. However, $\mathbf{r}_n(\mathcal{M}_{\Phi_{\mathsf{DR}}}) \geq 1$ for every $n \geq 0$ because at each step the trace $\texttt{req}^n$ is not equivalent to any shorter trace. Therefore, while $\mathcal{M}_{\Phi_{\mathsf{DR}}}$ is an infinite-state monitor, $\mathcal{M}$ is a finite-state monitor, and $\mathbf{r}(\mathcal{M}) < \mathbf{r}(\mathcal{M}_{\Phi_{\mathsf{DR}}})$.

Finally, we conclude the description of our framework by proving the implications in Fig. 1 to establish how different ways to compare resource use of monitors relate as well as a refinement property for resource-optimal monitors.

Proposition 9. *For every monitor $\mathcal{M}_1$ and $\mathcal{M}_2$ the implications in Fig. 1 hold.*

$$\begin{tikzcd}
 & \mathbf{R}(\mathcal{M}_1) \ll \mathbf{R}(\mathcal{M}_2) \arrow[rr, Rightarrow] & & \mathbf{R}(\mathcal{M}_1) < \mathbf{R}(\mathcal{M}_2) \\
\mathbf{r}(\mathcal{M}_1) \ll \mathbf{r}(\mathcal{M}_2) \arrow[ur, Rightarrow] \arrow[rr, Rightarrow] & & \mathbf{r}(\mathcal{M}_1) < \mathbf{r}(\mathcal{M}_2) \arrow[ur, Rightarrow] &
\end{tikzcd}$$

Fig. 1. Implications between the comparisons of resource use.

Proposition 10. *Let Φ be a specification and $\delta_{\text{fin}}, \delta_{\text{lim}}$ be two error thresholds. Given $(\delta_{\text{fin}}, \delta_{\text{lim}})$-monitors $\mathcal{M}_1 = (\sim_1, \gamma_1)$ and $\mathcal{M}_2 = (\sim_2, \gamma_2)$ for Φ. If $\sim_1 \subseteq \sim_2$ and $\mathcal{M}_1$ is resource-optimal, then $\sim_1 = \sim_2$. Thus, $\mathcal{M}_2$ is also resource-optimal.*

We remark that our definitional framework can be instantiated by existing monitor models, e.g., finite state automata [15] or register monitors [32,37]. More concretely, let us consider the discounted response specification Φ_{DR} from Example 2. Its exact-value monitor $\mathcal{M}_{\Phi_{\mathsf{DR}}}$ from Example 5 can be implemented by a register monitor that stores the value n in its single register while checking for the LTL specification P using its finite-state memory. On the other hand, the monitor $\mathcal{M}$ from Example 5 can be implemented by a finite state machine.

3 Approximate Prompt Monitoring

The original purpose of a monitor is to provide continuous feedback about the system status with respect to the specification [13,30]. Focusing only on limit monitoring may allow an unbounded prompt-error and thus fail to fulfill this task. In this section, we consider *prompt monitoring*, i.e., the case where the monitor performs bounded prompt-error. First, we remark that considering a bounded prompt-error implicitly bounds the limit-error by definition.

Fact 11. *Let Φ be a specification and $\delta_{\text{fin}}, \delta_{\text{lim}} \in \overline{\mathbb{R}}$ be error thresholds. If $\mathcal{M}$ is a $(\delta_{\text{fin}}, \delta_{\text{lim}})$-monitor for Φ, then it is also a (δ_{fin}, x)-monitor for Φ where $x = \min\{\delta_{\text{fin}}, \delta_{\text{lim}}\}$.*

3.1 Uniqueness of resource-optimal prompt monitors

The exact-value monitor is arguably the most natural monitor for a given specification. In fact, it is the unique error-free monitor that is resource-optimal.

Theorem 12. *Let Φ be a specification, and $\delta \in \overline{\mathbb{R}}$ be an error threshold. Then, $\mathcal{M}_\Phi$ is the unique resource-optimal $(0, \delta)$-monitor for Φ.*

Proof. Let $\Phi = (\pi, \ell)$. Consider $\mathcal{M} = (\sim, \gamma)$ as a resource-optimal $(0, \delta)$-monitor for Φ. We get $\sim \subseteq \sim_\Phi^*$ thanks to the following implications.

$$\begin{aligned}
s_1 \sim s_2 &\implies \forall r \in \Sigma^*,\ s_1 r \sim s_2 r && \text{(right-monotonicity)}\\
&\implies \forall r \in \Sigma^*,\ \gamma([s_1 r]) = \gamma([s_2 r]) && \text{(definition)}\\
&\implies \forall r \in \Sigma^*,\ \pi(s_1 r) = \pi(s_2 r) && \text{(prompt-error 0)}\\
&\implies s_1 \sim_\Phi^* s_2 && \text{(definition)}
\end{aligned}$$

On the one hand, we have that $\sim = \sim_\Phi^*$ by Proposition 10. On the other hand, we have that $\gamma([s]) = \pi(s)$ for all $s \in \Sigma^*$ since the prompt-error threshold is 0. As a direct consequence, $\mathcal{M} = \mathcal{M}_\Phi$. □

$\Phi = (\pi, \lim)$ where:

$$\pi\colon s \mapsto \begin{cases} 0 & \text{if } s = \varepsilon \\ 3x & \text{if } s = a \\ 5x & \text{if } s = b \\ 7x & \text{if } s = c \\ 10x & \text{if } s \in \Sigma a \Sigma^* \\ 10x + y & \text{if } s \in \Sigma b \Sigma^* \\ 10x + 2y & \text{if } s \in \Sigma c \Sigma^* \end{cases}$$

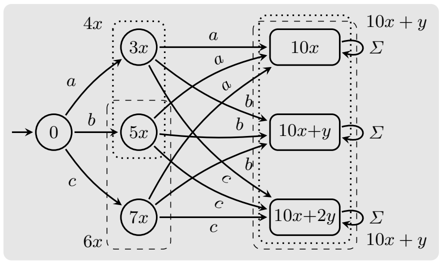


Fig. 2. A specification Φ over $\Sigma = \{a, b, c\}$ where $x > 0$ and $y \leq x$, and two resource-optimal (x, y)-monitors for Φ shown on top of the exact-value monitor $\mathcal{M}_\Phi$. As indicated by the output values on the dotted and dashed rectangles, the approximate monitors merge some equivalence classes of $\mathcal{M}_\Phi$ to save resources at the cost of losing precision.

Unfortunately, the uniqueness of resource-optimal monitors does not necessarily hold once we allow erroneous monitor verdicts. For instance, Fig. 2 shows on the left a specification Φ parameterized by x and y, together with its exact-value monitor $\mathcal{M}_\Phi$ on the right. In addition, the figure highlights two ways to make $\sim_\Phi$ coarser to obtain distinct resource-optimal (x, y)-monitors for Φ.

Proposition 13. *For all $x > 0$ and $y \leq x$ there exists a specification Φ that admits multiple resource-optimal (x, y)-monitors.*

3.2 Structure of resource-optimal prompt monitors

Regardless of the uniqueness, one can ask whether making $\sim_\Phi$ coarser always yields a resource-optimal approximate monitor. Here, we answer this question negatively. In particular, Fig. 3 shows on the left a specification Φ and on the right a resource-optimal $(1, 0)$-monitor $\mathcal{M} = (\sim, \gamma)$ for Φ with $ab \nsim ba$, although $ab \sim_\Phi^* ba$.

Proposition 14. *There exists a $(1, 0)$-monitor $\mathcal{M} = (\sim, \gamma)$ for some specification Φ such that for every other $(1, 0)$-monitor $\mathcal{M}' = (\sim', \gamma')$ we have that $\sim_\Phi \subseteq \sim'$ implies $\mathbf{r}(\mathcal{M}) \ll \mathbf{r}(\mathcal{M}')$.*

$\Phi = (\pi, \lim)$ where:

$$\pi\colon s \mapsto \begin{cases} 0 & \text{if } s = \varepsilon \\ 3 & \text{if } s = c \\ 6 & \text{if } s = a \text{ or } s = ca \\ 9 & \text{if } s = b \text{ or } s = cb \\ 12 & \text{if } s = cab \\ 14 & \text{if } s = ab \text{ or } s = ba \\ 16 & \text{if } s = cba \\ 19 & \text{otherwise} \end{cases}$$

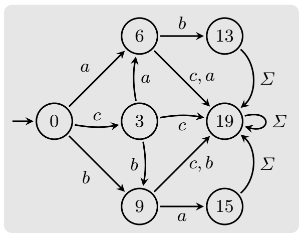
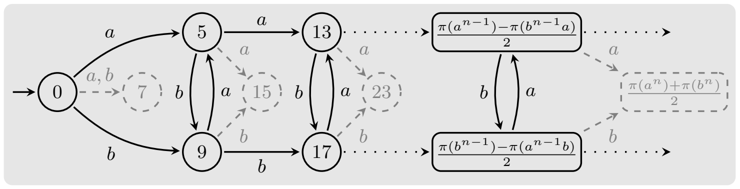


Fig. 3. A specification for which no $(1, 0)$-monitor that $\mathcal{M}_\Phi$ refines is resource-optimal, and the witnessing resource-optimal approximate monitor that splits an equivalence class of the specification.



0
5
9
13
17
7
15
23
a
b
a, b
$\frac{\pi(a^{n-1})-\pi(b^{n-1}a)}{2}$
$\frac{\pi(b^{n-1})-\pi(a^{n-1}b)}{2}$
$\frac{\pi(a^{n})+\pi(b^{n})}{2}$

Fig. 4. A resource-optimal (1,1)-monitor for the specification Φ of Proposition 15 that never minimizes its step-wise resource use $\mathbf{r}_n$ (black). Attempting to minimize $\mathbf{r}_n$ at each step n results in taking a^n and b^n as equivalent, but breaking the equivalence at step $n+1$ as the prompt-error bound would be violated otherwise (gray).

We established that the structure of the exact-value monitor does not necessarily provide insights into finding a resource-optimal approximate monitor. In fact, as we demonstrate in Fig. 4, there exist a specification such that its resource-optimal $(1,1)$-monitor $\mathcal{M}$ never minimizes the resource use $\mathbf{r}_i(\mathcal{M})$.

Proposition 15. *There exists a specification Φ admitting a $(1,1)$-monitor $\mathcal{M} = (\sim, \gamma)$ such that for all equivalence relations $\approx$ over Σ^* and $n \in \mathbb{N}$ we have that $|\Sigma^{\leq n}/{\sim}|$ is strictly greater than*

$$\min\left\{ |\Sigma^{\leq n}/{\approx}| \;\middle|\; \forall s_1, s_2 \in \Sigma^{\leq n} : s_1 \approx s_2 \Rightarrow \bigwedge \begin{matrix} \forall r \in \Sigma^* : s_1 r \approx s_2 r \\ |\Phi(s_1) - \Phi(s_2)| \leq 1 \end{matrix} \right\}.$$

Proof. Let $\Phi = (\pi, \limsup)$ be a specification from $\Sigma = \{a, b\}$ to $\mathbb{N}$ where π is defined as follows.

$$\pi : s \mapsto \begin{cases} 8|s| & \text{if } s \in b^* \\ 8|s| - 16k + 4 & \text{if } s \in (b^+a^+)^k \text{ for some } k \geq 1 \\ 8|s| - 16k + 2 & \text{if } s \in (b^+a^+)^k b^+ \text{ for some } k \geq 1 \\ 8|s| - 2 & \text{if } s \in a^+ \\ 8|s| - 16k + 10 & \text{if } s \in (a^+b^+)^k \text{ for some } k \geq 1 \\ 8|s| - 16k - 4 & \text{if } s \in (a^+b^+)^k a^+ \text{ for some } k \geq 1 \end{cases}$$

Let $n \in \mathbb{N}$. The key argument is that it is beneficial to put a^n and b^n in the same equivalence class for minimizing $\mathbf{r}_n$ since $|\Phi(a^n) - \Phi(b^n)| = 2$ and since no other trace in $\Sigma^{\leq n}$ admits a value close to either $\Phi(a^n)$ or $\Phi(b^n)$. However, once we consider traces of length $n+1$, we introduce several values close to $\Phi(a^n)$ as well as $\Phi(b^n)$, but not both at the same time. Therefore, to minimize the resource use $\mathbf{r}_{n+1}$ while maintaining the prompt-error bound of 1, it becomes beneficial to put a^n and b^n in distinct equivalence classes. □

3.3 Unbounded precision-resource trade-offs for prompt monitors

In this subsection, we exhibit specifications admitting an infinite sequence of monitors that trade precision and resource use. First, we investigate the *maximal* *response-time* specification by demonstrating how a monitor can save more and more resources by increasing both its prompt- and limit-error.

Example 16. Let $\Sigma = \{\texttt{req}, \texttt{ack}, \texttt{other}\}$ and consider the usual LTL response specification $P = \Box(\texttt{req} \to \Diamond\texttt{ack})$. We define $\mathsf{CurResp}(s) = 0$ if $s \in P$, and $\mathsf{CurResp}(s) = |s| - |r|$ otherwise, where $r \prec s$ is the longest prefix with $r \in P$. Now, let $\mathsf{MaxResp}(s) = \sup_{r \preceq s} \mathsf{CurResp}(r)$ and define $\Phi_{\mathsf{MR}} = (\mathsf{MaxResp}, \lim)$, which we call the *maximal response-time* specification. Note that $\mathsf{CurResp}$ outputs the current response-time for a finite trace, and $\mathsf{MaxResp}$ outputs the maximum response-time so far.

Consider the monitor $\mathcal{M} = (\sim, \gamma)$ that counts the response time when there is an open $\texttt{req}$, but only stores an approximation of the maximum when an $\texttt{ack}$ occurs. More explicitly, let $\sim$ and γ be such that we have the following: $\mathcal{M}(s) = 5k + 2$ if $s \in P$, where $k \in \mathbb{N}$ satisfies $5k \leq \mathsf{MaxResp}(s) < 5(k+1)$; and $\mathcal{M}(s) = \max\{\mathcal{M}(r), \mathsf{CurResp(s)}\}$ otherwise, where $r \prec s$ is the longest prefix with $r \in P$. We claim that $\mathcal{M}$ is a $(2, 2)$-monitor for Φ_{MR}. First, observe that whenever there is no pending request, i.e., $s \in P$, the monitor has a prompt-error of at most 2 by construction. Indeed, $\mathsf{MaxResp}(s) \in \{5k+i \mid i \in \{0, 1, 2, 3, 4\}\}$. In the case of a pending request, i.e., $s \notin P$, there is a prompt-error only when the monitor's approximation of the maximum-so-far is not replaced by the current response time. Again, by construction, we can bound this error by 2. Intuitively, $\mathcal{M}$ achieves this approximation by merging in $\sim$ some equivalence classes of $\sim^*_{\Phi_{\mathsf{MR}}}$ where there are no pending requests. One can thus verify that $\mathbf{r}(\mathcal{M}) < \mathbf{r}(\mathcal{M}_{\Phi_{\mathsf{MR}}})$.

The construction described in Example 16 can be generalized to identify a precision-resource trade-off with an infinite hierarchy of approximate monitors.

Theorem 17. *For all $\delta \in \mathbb{N}$, there exists a (δ, δ)-monitor $\mathcal{M}_\delta$ for the maximal response-time specification. Furthermore, $\mathbf{r}(\mathcal{M}_i) < \mathbf{r}(\mathcal{M}_j)$ for all $i > j$, and $\mathcal{M}_0$ is the exact-value monitor.*

Proof. Let $\Phi_{\mathsf{MR}} = (\mathsf{MaxResp}, \lim)$ be the maximal response-time specification as introduced in Example 16. Let $\delta \in \mathbb{N}$ and $s \in \Sigma^*$. If s does not have a pending request, we define $\mathcal{M}_\delta(s) = k(2\delta + 1) + \delta$ where $k \in \mathbb{N}$ satisfies $k(2\delta+1) \leq \mathsf{MaxResp}(s) < (k+1)(2\delta+1)$. Otherwise, if s has a pending request, we define $\mathcal{M}_\delta(s) = \max\{\mathcal{M}_\delta(r), \mathsf{CurResp(s)}\}$ where $r \prec s$ is the longest prefix with no pending request. We construct the (δ, δ)-monitor $\mathcal{M}_\delta$ for Φ_{MR} as in Example 16. In particular, $\mathcal{M}_0$ is the exact-value monitor. Indeed, $\delta = 0$ implies $\mathcal{M}_\delta(s) = k = \mathsf{MaxResp}(s)$ when s does not have a pending request, and otherwise $\mathcal{M}_\delta(s) = \sup_{r \preceq s} \mathsf{CurResp(r)} = \mathsf{MaxResp}(s)$ by definition. For all $i > j$, the monitor $\mathcal{M}_i$ partitions the traces with no pending requests into sets of cardinality $2i+1$ while $\mathcal{M}_j$ does so using sets of cardinality $2j+1$. Then, the equivalence relation used by $\mathcal{M}_i$ is coarser than that of $\mathcal{M}_j$, and thus $\mathbf{r}(\mathcal{M}_i) < \mathbf{r}(\mathcal{M}_j)$. $\square$

Note that, except $\mathcal{M}_0$, the monitors given by Theorem 17 have non-zero limit-error. We explore in Section 4 the specifications for which having fewer resources than the exact-value monitor forces a non-zero limit-error. Moreover, we show in Example 25 that the maximal response-time is one of these specifications.

Next, we investigate the *server/client* specification by demonstrating how a monitor can be more and more precise by increasing its resource use.

Example 18. Consider a server that receives requests and issues acknowledgments. The number of simultaneous requests the system can handle is determined at runtime through a preprocessing computation. We describe a specification that, at its core, requires that every request is acknowledged and the server never has more open requests than it can handle. In particular, until the server is turned off, the specification assigns a value to each finite trace, denoting the likelihood and criticality of a potential immediate violation.

Let $\Sigma = \{\texttt{req}, \texttt{ack}, \texttt{other}, \texttt{off}\}$ be an alphabet, $\lambda \in (0, 1)$ be a discount factor, and $\Lambda > 0$ be an integer denoting the request threshold. For every $s \in \Sigma^*$ we denote by $\mathsf{NumReq}(s)$ the number of pending requests in s. We define the *server/client specification* $\Phi_{\mathsf{SC}} = (\pi, \lim)$ where π is defined as follows.

- $\pi(s) = 0$ if s contains an occurrence of $\texttt{off}$,
- $\pi(s) = \mathsf{NumReq}(s)\lambda^{|s|}$ if $\mathsf{NumReq}(r) \leq \Lambda$ for all $r \preceq s$, and otherwise
- $\pi(s) = \mathsf{NumReq}(r)\lambda^{|r|}$ where $r \preceq s$ is the shortest with $\mathsf{NumReq}(r) > \Lambda$.

Theorem 19. *For every positive integer Λ and real number $0 < \delta \leq \Lambda$, there exists a (δ, δ)-monitor $\mathcal{M}_\delta$ for the server/client specification Φ_{SC}. Furthermore, $\mathcal{M}_\delta$ uses finitely many resources.*

Proof. Let Λ and δ be as above, and consider the set X we define as follows: $X = \{s \in \Sigma^* \mid \sup_{r_1 \in \Sigma^*}\{\pi(sr_1)\} - \inf_{r_2 \in \Sigma^*}\{\pi(sr_2)\} \geq \delta\}$. Note that X is finite. On the one hand, only a finite number of prefixes of a trace admitting an occurrence of $\texttt{off}$ can belong to X since $\delta > 0$ and by definition of Φ_{SC}. On the other hand, only a finite number of prefixes of a trace in which no $\texttt{off}$ occurs can belong to X since the discounting forces the value of Φ_{SC} to converge to 0. We construct $\mathcal{M}_\delta$ such that, if the trace belongs to X, it outputs the value given by the specification, otherwise it outputs the value of the shortest prefix that does not belong to X. In other words, $\mathcal{M}_\delta$ does not distinguish traces with the same prefix not belonging to X and thus admits at most $2|X|$ equivalence classes. $\square$

4 Approximate Limit Monitoring

In contrast to Section 3 where we tackle the limit monitoring problem indirectly with a bounded prompt-error, here we bound the limit-error directly and allow arbitrary prompt-error.

Example 20. Let $\Phi = (\pi, \liminf)$ be a specification over $\Sigma = \{\texttt{safe}, \texttt{danger}, \texttt{off}\}$ such that $\pi(s) = 2^{|r|}$ if s does not contain $\texttt{off}$, where r is the longest suffix of s of the form $\texttt{safe}^*$, and $\pi(s) = |s|_{\texttt{danger}}$ otherwise. Intuitively, Φ assigns each trace a confidence value while the system is on and how many times the system was in danger otherwise. We describe an approximate monitor with unbounded prompt-error and bounded but non-zero limit-error.

Let $\sim$ be a right-monotonic equivalence relation and γ an output function such that $\mathcal{M} = (\sim, \gamma)$ satisfies the following: $\mathcal{M}(s) = \infty$ when s has no `off` and ends with `safe`, $\mathcal{M}(s) = 0$ when s has no `off` and ends with `danger`, and $\mathcal{M}(s) = 9k + 4$ otherwise, where $k \in \mathbb{N}$ satisfies $9k \leq |s|_{\texttt{danger}} < 9(k+1)$. Notice that the monitor partitions $\mathbb{N}$ into intervals and takes traces with a "close enough" number of `danger`'s equivalent – as in Example 16. It is easy to see that $\mathcal{M}$ is a $(\infty, 4)$-monitor for Φ.

At its core, the limit-error threshold of a monitor is a theoretical guarantee since we cannot compute arbitrary limit-measures at runtime. Then, as a starting point, we insist that the monitor has zero limit-error, which is a reasonable requirement given that we allow unbounded prompt-error. In this case, the monitoring is still potentially approximate since we allow any error on finite traces. To talk about specifications for which saving resources by allowing prompt-error is not possible, we define the following notion.

Definition 21 (resource-intensive limit behavior). *A specification Φ has* resource-intensive limit behavior *iff its exact-value monitor $\mathcal{M}_\Phi$ is a resource-optimal $(\delta, 0)$-monitor for any $\delta \geq 0$.*

First, we identify a sufficient condition for a specification to be resource-intensive limit behavior. Then, we present *reversible specifications* and show a subclass of them that satisfy our condition.

4.1 Specifications with resource-intensive limit behavior

Let $\Phi = (\pi, \ell)$ be a specification and recall the equivalence $\sim^*_\Phi$ that, for every $s_1, s_2 \in \Sigma^*$, is defined as $s_1 \sim^*_\Phi s_2$ iff $\pi(s_1 r) = \pi(s_2 r)$ holds for all $r \in \Sigma^*$. To investigate the limit behavior of a specification, we define the following equivalence relation: for every $s_1, s_2 \in \Sigma^*$ we have $s_1 \sim^\omega_\Phi s_2$ iff $\ell(\pi(s_1 f)) = \ell(\pi(s_2 f))$ holds for all $f \in \Sigma^\omega$. Intuitively, traces with indistinguishable limit behavior are equivalent according to this relation. As a direct consequence of Fact 11, the following holds.

Fact 22. *For every specification Φ, we have that $\sim^*_\Phi \subseteq \sim^\omega_\Phi$.*

However, the converse does not necessarily hold, as we demonstrate with Example 23 below. We will show later that, when it holds, the specification has resource-intensive limit behavior.

Example 23. Recall the discounted response specification Φ_{DR} in Example 2, and that for all $s, r \in \Sigma^*$, we have $s \sim^*_{\Phi_{\mathsf{DR}}} r$ iff either (i) both traces have no pending `req` or (ii) both have a `req` pending for the same number of steps.

Let $s, r \in \Sigma^*$. We claim $s \sim^\omega_{\Phi_{\mathsf{DR}}} r$ iff either both traces have a pending request or both do not. Indeed, if s has a pending request and r does not, then we have $\Phi(s.\texttt{other}^\omega) = 0$ but $\Phi(r.\texttt{other}^\omega) = 1$. For the other direction, simply observe that if $s \sim^\omega_{\Phi_{\mathsf{DR}}} r$ then $\Phi(s.\texttt{other}^\omega) = \Phi(r.\texttt{other}^\omega)$, but the equality does not hold if s has a pending request and r does not (or vice versa). Having these characterizations at hand, we immediately observe that $s \sim^*_{\Phi_{\mathsf{DR}}} r$ implies $s \sim^\omega_{\Phi_{\mathsf{DR}}} r$.

Notice that the approximate monitor $\mathcal{M}$ for Φ_{DR} we constructed in Example 5 follows exactly the limit behavior of the specification. We were able to take advantage of the fact that $\sim^{\omega}_{\Phi_{\mathsf{DR}}}$ is coarser than $\sim^{*}_{\Phi_{\mathsf{DR}}}$ and design $\mathcal{M}$ such that it saves resources by allowing some prompt-error but no limit-error. We generalize this observation by showing that we could not have designed such a monitor if these equivalences had overlapped.

Theorem 24. *Let Φ be a specification. If $\sim^{*}_{\Phi} = \sim^{\omega}_{\Phi}$ then Φ has resource-intensive limit behavior.*

Proof. Let $\mathcal{M} = (\sim, \gamma)$ be a resource-optimal $(\delta, 0)$-monitor for Φ. Suppose towards contradiction that $\sim^{*}_{\Phi} = \sim^{\omega}_{\Phi}$ and $\mathcal{M}_{\Phi}$ is not resource-optimal for Φ. In particular $\sim \neq \sim^{*}_{\Phi}$. Since the limit-error threshold is 0, we get $\sim \subseteq \sim^{*}_{\Phi}$ by the following.

$$\begin{aligned} s_1 \sim s_2 &\implies \forall f \in \Sigma^{\omega},\, \ell(\gamma([s_1 f])) = \ell(\gamma([s_2 f])) && \text{(right-monotonicity)} \\ &\iff \forall f \in \Sigma^{\omega},\, \ell(\pi(s_1 f)) = \ell(\pi(s_2 f)) && \text{(limit-error 0)} \\ &\iff s_1 \sim^{\omega}_{\Phi} s_2 && \text{(definition)} \\ &\iff s_1 \sim^{*}_{\Phi} s_2 && \text{(hypothesis)} \end{aligned}$$

The contradiction is then raised by Proposition 10 implying that $\sim = \sim^{*}_{\Phi}$. $\square$

As demonstrated in Example 5 and discussed above, the discounted response specification does not display resource-intensive limit behavior. We give below two examples of specifications with resource-intensive limit behavior. Let us start with the *maximal response-time* specification.

Example 25. Consider the maximal response-time specification $\Phi_{\mathsf{MR}} = (\mathsf{MaxResp}, \lim)$ from Example 16. We argue that $\sim^{*}_{\Phi_{\mathsf{MR}}}$ and $\sim^{\omega}_{\Phi_{\mathsf{MR}}}$ overlap.

Suppose towards contradiction that there exist $s, r \in \Sigma^*$ such that $s \sim^{\omega}_{\Phi_{\mathsf{MR}}} r$ and $s \not\sim^{*}_{\Phi_{\mathsf{MR}}} r$. Then, there is $t \in \Sigma^*$ with $\Phi_{\mathsf{MR}}(st) \neq \Phi_{\mathsf{MR}}(rt)$. If at least one of st or rt has no pending request, take the continuation $\mathtt{other}^{\omega}$ to reach a contradiction to $s \sim^{\omega}_{\Phi_{\mathsf{MR}}} r$. Otherwise, if in both st and rt the current response time is smaller than the maximum among granted requests, then the continuation $\mathtt{ack}^{\omega}$ yields a contradiction. The same continuation covers the case when both current response times are greater. Finally, assume w.l.o.g. that the current response time is smaller than the maximum among granted requests in st and greater in rt. In this case, $\mathtt{ack}^{\omega}$ yields a contradiction again because their outputs stay the same as $\Phi_{\mathsf{MR}}(st)$ and $\Phi_{\mathsf{MR}}(rt)$, respectively. Therefore, we have $s \sim^{*}_{\Phi_{\mathsf{MR}}} r$, and thus $\sim^{*}_{\Phi_{\mathsf{MR}}}$ and $\sim^{\omega}_{\Phi_{\mathsf{MR}}}$ overlap.

Next, we describe the *average response-time* specification and argue that it displays resource-intensive limit behavior.

Example 26. Let $\Sigma = \{\mathtt{req}, \mathtt{ack}, \mathtt{other}\}$ and consider the usual LTL response specification $P = \square(\mathtt{req} \rightarrow \lozenge\mathtt{ack})$. For $s \in \Sigma^*$, we denote by $\mathsf{RespTime}(s)$ the total number of letters between the matching $\mathtt{req}$-$\mathtt{ack}$ pairs in s, and by

$\mathsf{NumReq}(s)$ the number of valid `req`'s in s. For all $s \in \Sigma^*$, we fix $p(s) = 1$ if $s \in P$, and $p(s) = 0$ otherwise. Then, we define $\mathsf{RespTime}(s) = \sum_{r \preceq s} 1 - p(r)$ and $\mathsf{NumReq}(s) = |P_s|$ where $P_s = \{r \preceq s \mid \exists t \in \Sigma^*,\ r = t.\mathtt{req} \wedge p(t) = 1\}$ is the set of valid requests in s. We define the *average response-time* specification as $\Phi_{\mathsf{AR}} = (\mathsf{AvgResp}, \liminf)$ where we let $\mathsf{AvgResp}(s) = \frac{\mathsf{RespTime}(s)}{\mathsf{NumReq}(s)}$ for all $s \in \Sigma^*$.

We claim that $\sim^*_{\Phi_{\mathsf{AR}}}$ and $\sim^\omega_{\Phi_{\mathsf{AR}}}$ overlap. To show this, one can proceed similarly as in Example 25. The cases with no pending requests are similar. When both traces have a pending request and their output values differ, extend both with $\mathtt{ack}^\omega$ to get a contradiction.

4.2 Reversible specifications

The *reversible* subclass of specifications enjoys the ability to move between computation steps forward and backward deterministically. Such specifications received particular interest in the literature since they can be implemented on hardware without energy dissipation [42,52]. Since it imitates the specification, the exact-value monitor of a reversible specification can roll back its computation, if allowed, without needing additional memory. From an automata-theoretic perspective, reversibility can be seen as the automaton being both forward and backward deterministic. Algebraically, this is captured by the syntactic monoid being a group.

Definition 27 (reversible specification). *A specification Φ is* reversible *iff $(\Sigma^*/\sim^*_\Phi, \cdot, \varepsilon)$ is a group.*

First, we describe the *average ping* specification – a variant of the average response-time specification where a single `ping` event captures `req` and `ack` events, and time proceeds through clock `tick` events. We then show that this specification is reversible.

Example 28. Let $\Sigma = \{\mathtt{ping}, \mathtt{tick}, \mathtt{other}\}$. Let $\mathsf{ValidTick}(s) = |s|_{\mathtt{tick}} - |r|_{\mathtt{tick}}$ where $r \preceq s$ is the longest prefix with no `ping`, and let $\mathsf{NumPing}(s) = |s|_{\mathtt{ping}}$. The *average ping* specification is defined as $\Phi_{\mathsf{AP}} = (\mathsf{AvgPing}, \liminf)$ where, for all $s \in \Sigma^*$, we let $\mathsf{AvgPing}(s) = \frac{\mathsf{ValidTick}(s)}{\mathsf{NumPing}(s)}$ if $\mathsf{NumPing}(s) > 0$; and $\mathsf{AvgPing}(s) = -1$ otherwise.

We argue that this specification is reversible. To see why, first observe for all $s, r \in \Sigma^*$ that we have $s \sim^*_{\Phi_{\mathsf{AP}}} r$ iff (i) $\mathsf{NumPing}(s) = \mathsf{NumPing}(r)$ and (ii) $\mathsf{ValidTick}(s) = \mathsf{ValidTick}(r)$. We particularly show for every $s, r, t \in \Sigma^*$ that if $s \not\sim^*_{\Phi_{\mathsf{AP}}} r$ then $st \not\sim^*_{\Phi_{\mathsf{AP}}} rt$, therefore $\sim^*_{\Phi_{\mathsf{AP}}}$ yields a group. Let $s, r \in \Sigma^*$ be such that $s \not\sim^*_{\Phi_{\mathsf{AP}}} r$ and let $t \in \Sigma^*$ be arbitrary. Suppose the condition (i) above does not hold. Since the $\mathsf{NumPing}$ values increase monotonically with every `ping`, we get $\mathsf{NumPing}(st) - \mathsf{NumPing}(rt) = \mathsf{NumPing}(s) - \mathsf{NumPing}(r)$, which is non-zero by supposition. If (ii) does not hold, it does not hold for st and rt either by a similar reasoning. Hence we have $st \not\sim^*_{\Phi_{\mathsf{AP}}} rt$.

Intuitively, we can backtrack the information on these functions: The value of $\mathsf{NumPing}$ is decremented with each preceding `ping`, while $\mathsf{ValidTick}$ is decremented with each preceding `tick` until it hits 0. It means that $\sim^*_{\Phi_{\mathsf{AP}}}$ can be seen as an automaton that is both forward and backward deterministic.

We identify below a well-behaved subclass of reversible specifications with resource-intensive limit behavior.

Theorem 29. *Let Φ be a reversible specification. If for every $s, r \in \Sigma^*$ with $s \sim_\Phi^\omega r$ there exists $t \in \Sigma^*$ with $st \sim_\Phi^* rt$, then Φ has resource-intensive limit behavior.*

Proof. We show that the reversibility of Φ, together with the above assumption, implies $\sim_\Phi^* = \sim_\Phi^\omega$. Note that the inclusion $\sim_\Phi^* \subseteq \sim_\Phi^\omega$ always holds as stated by Fact 22. Assuming $(\Sigma^*/\sim_\Phi^*, \cdot, \varepsilon)$ is a group, we have $s_1 r \sim_\Phi^* s_2 r \Rightarrow s_1 \sim_\Phi^* s_2$ for all $s_1, s_2, r \in \Sigma^*$. The inclusion $\sim_\Phi^\omega \subseteq \sim_\Phi^*$ holds since having $s_1 \not\sim_\Phi^* s_2$ implies for all $r \in \Sigma^*$ that $s_1 r \not\sim_\Phi^* s_2 r$, which in turn implies $s_1 \not\sim_\Phi^\omega s_2$ by our initial assumption. Finally, by Theorem 24, we obtain that Φ has resource-intensive limit behavior. $\square$

Recall the average ping specification from Example 28. It is reversible, as discussed earlier, and satisfies the condition in Theorem 29, therefore it has resource-intensive limit behavior. Finally, we present the *maximal ping* – a similarly simple variant of the maximal response-time specification. We demonstrate that this specification is not reversible, although it has resource-intensive limit behavior.

Example 30. Let $\Sigma = \{\texttt{ping}, \texttt{other}\}$ and consider the boolean specification $P = \Box\Diamond\texttt{ping}$. Let $\mathsf{CurPing}(s)$ and $\mathsf{MaxPing}(s)$ be defined similarly as for the maximal response-time specification in Example 16. We fix $\Phi_{\mathsf{MP}} = (\mathsf{MaxPing}, \lim)$ which we call the *maximal ping* specification. Consider $s = \texttt{ping.other}$ and $r = \texttt{ping.other.other}$. While $s \not\sim_{\Phi_{\mathsf{MP}}}^* r$, we have $sr \sim_{\Phi_{\mathsf{MP}}}^* rr$, therefore $\sim_{\Phi_{\mathsf{MP}}}^*$ does not yield a group. Intuitively, this is because we cannot backtrack the information on the running maximum. However, similarly as for the maximal-response time specification in Example 16, one can verify that $\sim_{\Phi_{\mathsf{MP}}}^* = \sim_{\Phi_{\mathsf{MP}}}^\omega$.

Note that a notion of reversibility exists for abstract monitors as well: A monitor $\mathcal{M} = (\sim, \gamma)$ where $\sim$ yields a group enjoys reversibility. In particular, this ability allows the monitor to return to a previous computation step without using additional resources and thus consider a different trace suffix.

5 Conclusion and Future Work

We formalize a framework that supports reasoning about precision-resource trade-offs for the approximate and exact monitoring of quantitative specifications. Unlike previous results, which analyze trade-offs for specific machine models such as register monitors [32,37], the framework presented in this paper studies for the first time an abstract notion of monitors, independent of the representation model, and separates the monitor errors on finite traces from those at the limit. These innovations allow us to design and study monitors that keep the focus on the resources needed for the approximate monitoring of quantitative specifications with a given precision. We provide several examples of when approximate monitoring can save resources and investigate when it fails to achieve this goal.

An expected future work is to provide a procedure for constructing a *concrete* (exact or approximate) monitor from an abstract description. Monitors having finitely many equivalence classes can be naturally mapped to finite-state automata. For a monitor with infinitely many equivalence classes, the model must be an infinite-state transition system. Yet, there are different levels of infinite state space. It can be generated, for example, by a finite collection of registers [32] or by a pushdown system [28]. Even when two abstract monitors are mapped to register automata with the same number of registers, they may differ in the type of operations used or the run-time needed per observation. It is also worth emphasizing that saving a single register may save infinitely many resources. Our current results do not provide such performance, so it is a natural future direction. To this end, we can consider alternative approaches to evaluate a monitor based on the number of violations of the error-threshold.

Another direction is on the relevance of resources through time. Our notion of resource use covers the number of equivalence classes added at time n, but an assumption that the monitor can release resources would trigger more possibilities. We can extend our framework to *dynamic abstract monitors* in a way that is related to existing works on dynamic programming for model checking [53]. Intuitively, a dynamic abstract monitor keeps track of the equivalence classes that can be reused in the future and prunes all the others to reduce resource use.

Acknowledgments

We thank the anonymous reviewers for their helpful comments. This work was supported in part by the ERC-2020-AdG 101020093.

References

1. Aceto, L., Achilleos, A., Francalanza, A., Ingólfsdóttir, A., Lehtinen, K.: The best a monitor can do. In: Baier, C., Goubault-Larrecq, J. (eds.) 29th EACSL Annual Conference on Computer Science Logic, CSL 2021, January 25-28, 2021, Ljubljana, Slovenia (Virtual Conference). LIPIcs, vol. 183, pp. 7:1–7:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021). https://doi.org/10.4230/LIPIcs.CSL.2021.7, https://doi.org/10.4230/LIPIcs.CSL.2021.7
2. Aceto, L., Achilleos, A., Francalanza, A., Ingólfsdóttir, A., Lehtinen, K.: An operational guide to monitorability with applications to regular properties. Softw. Syst. Model. **20**(2), 335–361 (2021). https://doi.org/10.1007/s10270-020-00860-z, https://doi.org/10.1007/s10270-020-00860-z
3. Albers, S.: Online algorithms: a survey. Mathematical Programming **97**(1), 3–26 (2003)
4. Alechina, N., Dastani, M., Logan, B.: Norm approximation for imperfect monitors. In: Bazzan, A.L.C., Huhns, M.N., Lomuscio, A., Scerri, P. (eds.) International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14, Paris, France, May 5-9, 2014. pp. 117–124. IFAAMAS/ACM (2014), http://dl.acm.org/citation.cfm?id=2615753
5. Alur, R., Mamouras, K., Stanford, C.: Automata-based stream processing. In: 44th International Colloquium on Automata, Languages, and Programming (ICALP 2017). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2017)
6. Alur, R., Mamouras, K., Stanford, C.: Modular quantitative monitoring. Proc. ACM Program. Lang. **3**(POPL) (Jan 2019). https://doi.org/10.1145/3290363, https://doi.org/10.1145/3290363
7. Aminof, B., Kupferman, O., Lampert, R.: Rigorous approximated determinization of weighted automata. Theor. Comput. Sci. **480**, 104–117 (2013). https://doi.org/10.1016/j.tcs.2013.02.005, https://doi.org/10.1016/j.tcs.2013.02.005
8. Audrito, G., Casadei, R., Damiani, F., Stolz, V., Viroli, M.: Adaptive distributed monitors of spatial properties for cyber-physical systems. J. Syst. Softw. **175**, 110908 (2021). https://doi.org/10.1016/j.jss.2021.110908, https://doi.org/10.1016/j.jss.2021.110908
9. Barringer, H., Falcone, Y., Havelund, K., Reger, G., Rydeheard, D.: Quantified event automata: Towards expressive and efficient runtime monitors. In: Giannakopoulou, D., Méry, D. (eds.) FM 2012: Formal Methods. pp. 68–84. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
10. Bartocci, E., Falcone, Y.: Lectures on Runtime Verification. Springer (2018)
11. Bartocci, E., Falcone, Y., Francalanza, A., Reger, G.: Introduction to runtime verification. In: Bartocci, E., Falcone, Y. (eds.) Lectures on Runtime Verification - Introductory and Advanced Topics, Lecture Notes in Computer Science, vol. 10457, pp. 1–33. Springer (2018). https://doi.org/10.1007/978-3-319-75632-5_1, https://doi.org/10.1007/978-3-319-75632-5_1
12. Basin, D., Klaedtke, F., Müller, S., Zălinescu, E.: Monitoring metric first-order temporal properties. Journal of the ACM (JACM) **62**(2), 1–45 (2015)
13. Bauer, A., Leucker, M., Schallhart, C.: The good, the bad, and the ugly, but how ugly is ugly? In: International Workshop on Runtime Verification. pp. 126–138. Springer (2007)
14. Bauer, A., Leucker, M., Schallhart, C.: Comparing LTL semantics for runtime verification. J. Log. Comput. **20**(3), 651–674 (2010). https://doi.org/10.1093/logcom/exn075, https://doi.org/10.1093/logcom/exn075

15. Bauer, A., Leucker, M., Schallhart, C.: Runtime verification for ltl and tltl. ACM Trans. Softw. Eng. Methodol. **20**(4) (Sep 2011). `https://doi.org/10.1145/2000799.2000800`, `https://doi.org/10.1145/2000799.2000800`
16. Boker, U., Henzinger, T.A.: Approximate determinization of quantitative automata. In: D'Souza, D., Kavitha, T., Radhakrishnan, J. (eds.) IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012, December 15-17, 2012, Hyderabad, India. LIPIcs, vol. 18, pp. 362–373. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2012). `https://doi.org/10.4230/LIPIcs.FSTTCS.2012.362`, `https://doi.org/10.4230/LIPIcs.FSTTCS.2012.362`
17. Brázdil, T., Chatterjee, K., Forejt, V., Kučera, A.: Multigain: A controller synthesis tool for mdps with multiple mean-payoff objectives. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 181–187. Springer (2015)
18. Calinescu, R., Gerasimou, S., Johnson, K., Paterson, C.: Using runtime quantitative verification to provide assurance evidence for self-adaptive software - advances, applications and research challenges. In: de Lemos, R., Garlan, D., Ghezzi, C., Giese, H. (eds.) Software Engineering for Self-Adaptive Systems III. Assurances - International Seminar, Dagstuhl Castle, Germany, December 15-19, 2013, Revised Selected and Invited Papers. Lecture Notes in Computer Science, vol. 9640, pp. 223–248. Springer (2013). `https://doi.org/10.1007/978-3-319-74183-3_8`, `https://doi.org/10.1007/978-3-319-74183-3_8`
19. Chang, E., Manna, Z., Pnueli, A.: The safety-progress classification. In: Logic and Algebra of Specification, pp. 143–202. Springer (1993)
20. Chatterjee, K., Doyen, L.: Energy and mean-payoff parity markov decision processes. In: International Symposium on Mathematical Foundations of Computer Science. pp. 206–218. Springer (2011)
21. Chatterjee, K., Doyen, L., Henzinger, T.A.: Quantitative languages. ACM Trans. Comput. Logic **11**(4) (Jul 2010). `https://doi.org/10.1145/1805950.1805953`, `https://doi.org/10.1145/1805950.1805953`
22. Chatterjee, K., Henzinger, T.A., Otop, J.: Quantitative monitor automata. In: International Static Analysis Symposium. pp. 23–38. Springer (2016)
23. Cimatti, A., Tian, C., Tonetta, S.: Assumption-based runtime verification of infinite-state systems. In: Feng, L., Fisman, D. (eds.) Runtime Verification - 21st International Conference, RV 2021, Virtual Event, October 11-14, 2021, Proceedings. Lecture Notes in Computer Science, vol. 12974, pp. 207–227. Springer (2021). `https://doi.org/10.1007/978-3-030-88494-9_11`, `https://doi.org/10.1007/978-3-030-88494-9_11`
24. Considine, J., Li, F., Kollios, G., Byers, J.W.: Approximate aggregation techniques for sensor databases. In: Özsoyoglu, Z.M., Zdonik, S.B. (eds.) Proceedings of the 20th International Conference on Data Engineering, ICDE 2004, 30 March - 2 April 2004, Boston, MA, USA. pp. 449–460. IEEE Computer Society (2004). `https://doi.org/10.1109/ICDE.2004.1320018`, `https://doi.org/10.1109/ICDE.2004.1320018`
25. Cousot, P.: Abstract interpretation. ACM Computing Surveys (CSUR) **28**(2), 324–328 (1996)
26. d'Angelo, B., Sankaranarayanan, S., Sánchez, C., Robinson, W., Finkbeiner, B., Sipma, H.B., Mehrotra, S., Manna, Z.: Lola: runtime monitoring of synchronous systems. In: 12th International Symposium on Temporal Representation and Reasoning (TIME'05). pp. 166–174. IEEE (2005)

27. De Giacomo, G., Vardi, M.Y.: Linear temporal logic and linear dynamic logic on finite traces. In: IJCAI'13 Proceedings of the Twenty-Third international joint conference on Artificial Intelligence. pp. 854–860. Association for Computing Machinery (2013)
28. Decker, N., Leucker, M., Thoma, D.: Impartiality and anticipation for monitoring of visibly context-free properties. In: Legay, A., Bensalem, S. (eds.) Runtime Verification. pp. 183–200. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
29. Eisner, C., Fisman, D., Havlicek, J., Lustig, Y., McIsaac, A., Van Campenhout, D.: Reasoning with temporal logic on truncated paths. In: Hunt, W.A., Somenzi, F. (eds.) Computer Aided Verification. pp. 27–39. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)
30. Falcone, Y., Fernandez, J.C., Mounier, L.: What can you verify and enforce at runtime? International Journal on Software Tools for Technology Transfer **14**(3), 349–382 (2012)
31. Ferrère, T., Henzinger, T.A., Kragl, B.: Monitoring event frequencies. In: 28th EACSL Annual Conference on Computer Science Logic (CSL 2020). Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2020)
32. Ferrère, T., Henzinger, T.A., Saraç, N.E.: A theory of register monitors. In: Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science. pp. 394–403 (2018)
33. Forejt, V., Kwiatkowska, M., Norman, G., Parker, D., Qu, H.: Quantitative multi-objective verification for probabilistic systems. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 112–127. Springer (2011)
34. Francalanza, A., Aceto, L., Achilleos, A., Attard, D.P., Cassar, I., Della Monica, D., Ingólfsdóttir, A.: A foundation for runtime monitoring. In: Lahiri, S., Reger, G. (eds.) Runtime Verification. pp. 8–29. Springer International Publishing, Cham (2017)
35. Halamish, S., Kupferman, O.: Approximating deterministic lattice automata. In: Chakraborty, S., Mukund, M. (eds.) Automated Technology for Verification and Analysis - 10th International Symposium, ATVA 2012, Thiruvananthapuram, India, October 3-6, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7561, pp. 27–41. Springer (2012). `https://doi.org/10.1007/978-3-642-33386-6_4`, `https://doi.org/10.1007/978-3-642-33386-6_4`
36. Henzinger, T.A., Saraç, N.E.: Monitorability under assumptions. In: Deshmukh, J., Ničković, D. (eds.) Runtime Verification. pp. 3–18. Springer International Publishing, Cham (2020)
37. Henzinger, T.A., Saraç, N.E.: Quantitative and approximate monitoring. In: 2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS). pp. 1–14. IEEE (2021)
38. Ho, H., Ouaknine, J., Worrell, J.: Online monitoring of metric temporal logic. In: Bonakdarpour, B., Smolka, S.A. (eds.) Runtime Verification - 5th International Conference, RV 2014, Toronto, ON, Canada, September 22-25, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8734, pp. 178–192. Springer (2014). `https://doi.org/10.1007/978-3-319-11164-3_15`, `https://doi.org/10.1007/978-3-319-11164-3_15`
39. Holzmann, G.J., Smith, M.H.: Automating software feature verification. Bell Labs Tech. J. **5**(2), 72–87 (2000). `https://doi.org/10.1002/bltj.2223`, `https://doi.org/10.1002/bltj.2223`

40. Jakšić, S., Bartocci, E., Grosu, R., Nguyen, T., Ničković, D.: Quantitative monitoring of stl with edit distance. Formal methods in system design **53**(1), 83–112 (2018)
41. Kwiatkowska, M.: Quantitative verification: Models techniques and tools. In: Proceedings of the the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering. p. 449–458. ESEC-FSE '07, Association for Computing Machinery, New York, NY, USA (2007). https://doi.org/10.1145/1287624.1287688, https://doi.org/10.1145/1287624.1287688
42. Landauer, R.: Irreversibility and heat generation in the computing process. IBM J. Res. Dev. **5**(3), 183–191 (1961). https://doi.org/10.1147/rd.53.0183, https://doi.org/10.1147/rd.53.0183
43. Maler, O., Nickovic, D.: Monitoring temporal properties of continuous signals. In: Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems, pp. 152–166. Springer (2004)
44. Mamouras, K., Chattopadhyay, A., Wang, Z.: Algebraic quantitative semantics for efficient online temporal monitoring. In: Groote, J.F., Larsen, K.G. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 27th International Conference, TACAS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12651, pp. 330–348. Springer (2021). https://doi.org/10.1007/978-3-030-72016-2_18, https://doi.org/10.1007/978-3-030-72016-2_18
45. Mamouras, K., Chattopadhyay, A., Wang, Z.: A compositional framework for quantitative online monitoring over continuous-time signals. In: Feng, L., Fisman, D. (eds.) Runtime Verification - 21st International Conference, RV 2021, Virtual Event, October 11-14, 2021, Proceedings. Lecture Notes in Computer Science, vol. 12974, pp. 142–163. Springer (2021). https://doi.org/10.1007/978-3-030-88494-9_8, https://doi.org/10.1007/978-3-030-88494-9_8
46. Mostafa, M., Bonakdarpour, B.: Decentralized runtime verification of LTL specifications in distributed systems. In: 2015 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2015, Hyderabad, India, May 25-29, 2015. pp. 494–503. IEEE Computer Society (2015). https://doi.org/10.1109/IPDPS.2015.95, https://doi.org/10.1109/IPDPS.2015.95
47. Nia, M.A., Kargahi, M., Faghih, F.: Probabilistic approximation of runtime quantitative verification in self-adaptive systems. Microprocess. Microsystems **72** (2020). https://doi.org/10.1016/j.micpro.2019.102943, https://doi.org/10.1016/j.micpro.2019.102943
48. Piterman, N., Pnueli, A.: Temporal Logic and Fair Discrete Systems, pp. 27–73. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-10575-8_2, https://doi.org/10.1007/978-3-319-10575-8_2
49. Shrivastava, N., Buragohain, C., Agrawal, D., Suri, S.: Medians and beyond: new aggregation techniques for sensor networks. In: Stankovic, J.A., Arora, A., Govindan, R. (eds.) Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, SenSys 2004, Baltimore, MD, USA, November 3-5, 2004. pp. 239–249. ACM (2004). https://doi.org/10.1145/1031495.1031524, https://doi.org/10.1145/1031495.1031524
50. Silberstein, A., Braynard, R., Yang, J.: Constraint chaining: on energy-efficient continuous monitoring in sensor networks. In: Chaudhuri, S., Hristidis, V., Polyzotis, N. (eds.) Proceedings of the ACM SIGMOD International Conference

on Management of Data, Chicago, Illinois, USA, June 27-29, 2006. pp. 157–168. ACM (2006). `https://doi.org/10.1145/1142473.1142492`, `https://doi.org/10.1145/1142473.1142492`

51. Stucki, S., Sánchez, C., Schneider, G., Bonakdarpour, B.: Gray-box monitoring of hyperproperties with an application to privacy. Formal Methods Syst. Des. **58**(1), 126–159 (2021). `https://doi.org/10.1007/s10703-020-00358-w`, `https://doi.org/10.1007/s10703-020-00358-w`
52. Toffoli, T.: Reversible computing. In: de Bakker, J.W., van Leeuwen, J. (eds.) Automata, Languages and Programming, 7th Colloquium, Noordweijkerhout, The Netherlands, July 14-18, 1980, Proceedings. Lecture Notes in Computer Science, vol. 85, pp. 632–644. Springer (1980). `https://doi.org/10.1007/3-540-10003-2_104`, `https://doi.org/10.1007/3-540-10003-2_104`
53. Wang, C., Yang, Y., Gupta, A., Gopalakrishnan, G.: Dynamic model checking with property driven pruning to detect race conditions. In: Cha, S.D., Choi, J., Kim, M., Lee, I., Viswanathan, M. (eds.) Automated Technology for Verification and Analysis, 6th International Symposium, ATVA 2008, Seoul, Korea, October 20-23, 2008. Proceedings. Lecture Notes in Computer Science, vol. 5311, pp. 126–140. Springer (2008). `https://doi.org/10.1007/978-3-540-88387-6_11`, `https://doi.org/10.1007/978-3-540-88387-6_11`
54. Zhang, X., Leucker, M., Dong, W.: Runtime verification with predictive semantics. In: Goodloe, A., Person, S. (eds.) NASA Formal Methods - 4th International Symposium, NFM 2012, Norfolk, VA, USA, April 3-5, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7226, pp. 418–432. Springer (2012). `https://doi.org/10.1007/978-3-642-28891-3_37`, `https://doi.org/10.1007/978-3-642-28891-3_37`