General Monitorability of Totally Ordered Verdict Domains

Felipe Gorostiaga[1,2*] and César Sánchez[1*]

[1]IMDEA Software Institute, Madrid, Spain.
[2]CIFASIS, Rosario, Argentina.

*Corresponding author(s). E-mail(s): felipe.gorostiaga@imdea.org; cesar.sanchez@imdea.org;

Abstract

Online runtime verification is a formal dynamic technique that studies how to monitor formal specifications incrementally against an input trace. Often, an observed prefix of a behavior is not enough to emit a definite verdict and the monitor must wait to receive more information from the execution of the observed system. Monitorability classifies the set of properties depending on the feasibility to obtain a verdict after a finite observation. Havelund and Peled [1] classified LTL properties according to whether an observation can be extended to a definite answer. In this paper we present a framework that extends the classification of Havelund and Peled to verdict domains that are richer than Booleans, obtaining a monitorability setting under which some of the verdicts (but not others) can be discarded after a sequence of observations. We study the instance of this setting to quantitative temporal logics and we illustrate using examples the different elements of the taxonomy. Finally, we also consider how assumptions on the set of behaviors can affect monitorability.

1 Introduction

Runtime verification (RV) is a dynamic formal technique for system reliability that studies how a sequence of events—emitted dynamically from the system under study—adhere to a given formal specification. Runtime verification focuses on two main problems: (1) how to generate a monitor from a given specification, and (2) algorithms that take a monitor and process a sequence of input events produced by the system, typically in a incremental manner, attempting to produce a definite verdict. In this paper we use *event* for an individual piece of information collected at a given instant from the system under observation, *observation* to refer to a finite sequence of events that the monitor receives, and *behavior* for each of the infinite sequences of events that temporal properties can describe (and systems can produce if executed ad-infinitum).

Static formal verification techniques like model checking [2, 3] attempt to prove that every behavior of the system satisfies a given specification. In contrast, in runtime verification monitors must decide based on observations. Runtime verification sacrifices completeness to provide an applicable formal extension of testing and debugging. See [4, 5] for surveys on runtime verification and the recent book [6].

Early specification languages studied for runtime verification were based on temporal logics, typically LTL [7–9], regular expressions [10], timed regular expressions [11], rules [12], or rewriting [13]. Since monitors only see an observation and not a complete behavior, the semantics of temporal logic must be adapted for finite traces. One solution is to adapt the semantics for finite traces [8] that provide a definite answer upon the "termination" of the trace. Another solution is to give a definite answer only if all the behaviors that extend the observation satisfy the specification (declaring satisfaction), or if all such extensions violate the specification (declaring violation). Otherwise, the monitor can produce a temporary "*I don't know*" verdict [9], with the hope to later refine it into a conclusive verdict. The idea of producing an inconclusive verdict was already introduced in the context of stream runtime verification [14] and later used in variants of LTL for finite traces, like LTLf [15] and MLTL [16].

A basic *soundness* criteria for monitorability states that monitors should never give a verdict that can be later reverted by an extended observation [17]. However, sound monitors can still switch from an indecisive verdict into a definite verdict. The soundness requirement is semantic in the sense that it is based on the semantics of the logic itself by considering all possible traces that are compatible with the given observation. Monitors can be formally understood as an implementation of a computational function that maps observations into verdicts [1, 18, 19] that respects the soundness requirement. Therefore, monitoring algorithms correspond to an incremental execution of the monitor as a function. From this perspective monitorability corresponds to the question of the existence of such a computable function.

One of the first definitions of monitorability, given by Pnueli and Zaks [20], establishes that an LTL property is monitorable after an observation o if there is another observation u that extends o for which the verdict is definitely a violation or there is an observation v that is an extension of o for which the verdict is a satisfaction. There are properties that are always monitorable for violation in the sense that every violating behavior has a finite prefix (observation) that is sufficient to determine the violation. For a second class of properties this witness only exists for some behaviors, and for the rest of the properties there is never such a witness observation (these definitions are analogous replacing violation by satisfaction). Havelund and Peled present in [1] a complete taxonomy for LTL, introducing the terms AFR (always finitely refutable), SFR (sometimes finitely refutable) and NFR (never finitely refutable). Their counterparts for a satisfaction verdict are AFS, SFS and NFS. In this paper we study extensions of this taxonomy for more expressive (non-Boolean) verdicts, in particular for totally ordered domains, including numerical domains used in quantitative temporal logics.

It is useful for specification engineers to have very expressive logics to define their properties, but additional expressiveness usually comes at the price of higher complexity in the decision problems and less efficient algorithms. Since the early languages used in RV were borrowed from static verification where decidability is crucial, these languages only allowed Boolean verdicts. However, runtime verification solves a simpler problem than model-checking so some researchers have been extending the expressivity of RV specification languages. Examples include logics that can quantify over the data in the events [1, 21], extensions of automata with the ability to store and compare data [22], and quantitative semantics for temporal logics [23]. Another direction to extend the expressivity of monitors is Stream Runtime Verification [14, 24–27] that abstract the data used in the monitoring algorithms in temporal logics to arbitrary data.

The work in [28] simultaneously over- and under-approximates the verdict for a given monitor as more information is received. In comparison, [28] focuses on approximating the outcome, while we focus on proving (discarding or considering) the precise verdict. Also, they deal with omega traces of observations that are letters from a finite alphabet, while we consider (for QLTL) complex data that include an infinite alphabet of observations. In [29], the authors consider model checking for quantitative temporal logic, which allows them to handle some notions of uncertainty. However, do not study monitorability of which values can be discarded for classes of formulae. In [30] they focus on quantitative safety and liveness, attempting to provide a semantic decomposition of properties into safety and liveness, analogous to the one in the Boolean case [31]. In this paper we first generalize the Havelund-Peled notions of monitorability to the setting of richer verdicts by studying whether a subset of the possible verdicts can be discarded after witnessing a finite trace. We borrow the terminology (safety, liveness, guarantee, morbidity and quaestio) from [1] and generalize them to the quantitative (totally ordered) setting, without an aim to preserve the decomposition of formulas into their safety and liveness constituents, but focusing on whether potential verdicts can be discarded.

In [15] the monitorability necessarily refers to the ability to give a conclusive verdict after a finite observation because the logic considered in [15] is a logic for finite traces. In contrast, the logics we consider are defined over infinite traces. Also, logics that guarantee that verdicts are obtained after a finite number of steps (by the semantics of the logic or some assumption on the input trace), like MLTL [16], are immediately in AFS and AFR.

Most runtime verification works typically focus on monitors that must be correct for any system under observation, which is considered completely unknown during the generation of the monitor. That is, a sound monitor must consider all future observations as possible. However, one can often monitor more effectively for particular systems or under *assumptions* about what the system can do. For example, [32] improves LTL monitoring using a model of the system to prune the set of possible future observations, and [18] considers how to improve the monitoring of hyperproperties using approximations of the system. Similarly, [33] illustrates properties that are not monitorable but become monitorable if one assumes that the input observation satisfies a given LTL formula. In practice, the events obtained from the system may not be perfect in the sense that some events may be lost, be incorrect or only received with approximate precision, which can affect the monitoring. For example, in [34] the authors study the possibility that events or event values are unknown, so the monitor must deal with the set of possible observations, therefore emitting sets of verdicts. In [35], the authors define the concept of *trace mutations* to capture divergences between observations and behaviors, and study how different mutations affect the monitorability of a property. We present in Section 5 how assumptions can affect the monitorability of properties with richer verdicts, and instantiate the monitorability landscape for the properties monitored. A systematic analysis of monitoring of rich verdicts under assumptions and uncertainties is however, out of the scope of this paper.

In summary, the contributions of the paper are: (1) an extension of the Havelund and Peled taxonomy of monitorability to richer verdicts and in particular to totally ordered verdict domains; and (2) an instantiation of the taxonomy to quantitative temporal logics.

Our taxonomy of properties, like the one introduced by Havelund and Peled, is based on the ability of monitors to produce verdicts. Other taxonomies of properties exist. For example, [36] classifies properties based on the use of the temporal operators involved.

This paper is an extended version of the conference paper [26] published in NFM'22. In this journal version we only analyze formalisms whose verdicts are totally ordered, but provide a much finer grain study of the classes. In particular, Section 5 is a fully novel characterization by considering independently (1) the possible behaviors and (2) the possible verdicts, while in [26] these decisions were not independent. The resulting taxonomy is much richer and a novel contribution, unique for quantitative domains. This is the main contribution of this journal version. These new notions could be extended to partially ordered verdict domains, but this is out of the scope of this paper.

The rest of the paper is structured as follows. Section 2 includes the preliminaries. Section 3 introduces a generalization of the monitorability framework to expressive verdicts. This is instantiated to quantitative temporal logics in Section 4. Section 5 further generalizes the framework, allowing for a more granular classification of the properties. In this section we also show how assumptions affect the classification of properties. Finally, Section 6 contains some final remarks.

2 Preliminaries

Sequences, Streams and Data

We represent the behaviors of a system as streams, where a stream of type D is an infinite sequence of values of D. We denote the type of the streams of type D as D^ω. We use *record types* to represent the information of different aspects of the system under study. The record type $\langle p_0 :: D_0, \ldots, p_n :: D_n\rangle$ represents a record that contains a finite number of entries. A value of a record type consists of a value of type D_i to each variable p_i, for $0 \leq i \leq n$.

Example 1 For example,

$$s \overset{\text{def}}{=} (\langle p : \mathit{true}\rangle\ \langle p : \mathit{true}\rangle\ \langle p : \mathit{false}\rangle^{\omega}) \in \langle p :: \mathit{Bool}\rangle^{\omega}$$

is the stream of values of record type $\langle p :: \mathit{Bool}\rangle$ values where p starts with two *true* values and remains *false* thereafter.

Given a record value $r \overset{\text{def}}{=} \langle p_0 : v_0, \ldots, p_n : v_n\rangle$ we use $r(p_i)$ to refer to v_i for $0 \leq i \leq n$. Given a stream $\sigma \in D^{\omega}$ and a natural number $i \in \mathbb{N}_0$ we use $\sigma(i)$ to refer to the element of type D at position i in σ. Similarly, we use σ^i to refer to the stream $(\sigma(i)\ \sigma(i{+}1)\ \ldots)$. Given a stream σ of values of record type $\langle p_0 : D_0, \ldots, p_n : D_n\rangle$ we abuse notation and use $\sigma(p_i)$ as the stream (of type D_i) that corresponds to $(\sigma(0)(p_i), \sigma(1)(p_i)\ldots)$. Also, we use $\sigma(j)(p_i)$ and $\sigma(p_i)(j)$ interchangeably.

Example 2 Let s be as in the previous example. Then, $s(0)(p) = \mathit{true}$, $s(50)(p) = \mathit{false}$, and $s^1 = (\langle p : \mathit{true}\rangle\ \langle p : \mathit{false}\rangle^{\omega})$.

We use finite sequences to represent *observations* from the system under analysis. A sequence of type D is a finite sequence of values of D, and we denote the type of the sequences of type D as D^*. We use ϵ for the empty sequence.

Example 3 The following

$$\mathit{ls} \overset{\text{def}}{=} [\langle p : \mathit{true}\rangle\ \langle p : \mathit{true}\rangle\ \langle p : \mathit{false}\rangle\ \langle p : \mathit{false}\rangle\ \langle p : \mathit{false}\rangle]$$

is the stream of assignments of Boolean values to p, which starts with two *true* values and is succeeded by three *false* values.

We say that a sequence $\mathit{ls} \in D^*$ of n elements is a prefix of a stream $s \in D^{\omega}$ and write $\mathit{ls} \prec s$ if the first n elements of s coincide with (the n elements of) ls. We also say that s is a continuation of ls. We say that $\mathit{ls} \in D^*$ is a subsequence of a stream $s \in D^{\omega}$ and write $\mathit{ls} \sqsubset s$ if there is an index i such that $\mathit{ls} \prec s^i$. We also say that s is an expansion of ls. For instance, considering ls as defined in Example 3, $\mathit{ls} \prec s$ (that is, s is a continuation of ls), and obviously $\mathit{ls} \sqsubset s$ (that is, s is an expansion of ls). The sequence $[\langle p : \mathit{false}\rangle\ \langle p : \mathit{false}\rangle\ \langle p : \mathit{false}\rangle]$ is also a subsequence of s, because it is a prefix of s^2.

Semantics Monitors

We borrow the definition of a semantic monitor from [37]. Consider a formalism whose semantics $[\![\cdot]\!]$ is defined over behaviors of type I^{ω} and that assigns verdicts of type D. The semantic monitor of a specification φ in that formalism is a function

$$M_{\varphi}(s) = \{[\![sw]\!](\varphi) \mid w \in I^{\omega}\}$$

The semantic monitor provides the set of possible outcomes $[\![sw]\!](\varphi)$ that are compatible with the observed finite prefix s. If the set of potential verdicts D is the Boolean domain, we call the formalism a *Boolean formalism*. For a Boolean formalism, the semantic monitor of a property given a finite prefix of a trace will return either $\{\mathit{true}\}$, $\{\mathit{false}\}$ or $\{\mathit{true}, \mathit{false}\}$. Note that a semantic monitor can provide information about a property statically if invoked with input prefix ϵ. We say that the values in $D \setminus M_{\varphi}(\epsilon)$ are *statically dismissible*.

Linear Temporal Logic

Linear Temporal Logic (LTL) is a formalism defined over records of Boolean values and whose semantics assigns Boolean verdicts. Let $\mathsf{AP} = \{p_0, \ldots, p_n\}$ be a finite set of atomic propositions and $R \overset{\text{def}}{=} \langle p_0 :: \mathit{Bool}, \ldots, p_n :: \mathit{Bool}\rangle$ the record type that assigns a Boolean value to each atomic proposition in AP. The syntax of LTL is:

$$\varphi ::= T \mid a \mid \varphi \vee \varphi \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi\,\mathcal{U}\,\varphi$$

where $a \in \mathsf{AP}$ is an atomic proposition, $\vee$ and $\neg$ are the usual Boolean disjunction and negation, and $\bigcirc$ and $\mathcal{U}$ are the next and until temporal operators. The semantics of LTL associate behaviors $\sigma \in R^{\omega}$ with formulas as follows:

$$\begin{array}{ll}
[\![T]\!](\sigma) & \overset{\text{def}}{=} \mathit{true} \\
[\![\varphi \vee \psi]\!](\sigma) & \overset{\text{def}}{=} [\![\varphi]\!](\sigma) \vee [\![\psi]\!](\sigma) \\
[\![a]\!](\sigma) & \overset{\text{def}}{=} \sigma(0)(a) \\
[\![\neg\varphi]\!](\sigma) & \overset{\text{def}}{=} \neg[\![\varphi]\!](\sigma) \\
[\![\bigcirc\varphi]\!](\sigma) & \overset{\text{def}}{=} [\![\varphi]\!](\sigma^1) \\
[\![\varphi\,\mathcal{U}\,\psi]\!](\sigma) & \overset{\text{def}}{=} \bigvee_{i \geq 0} \{[\![\psi]\!](\sigma^i) \wedge \bigwedge_{0 \leq j < i} \{[\![\varphi]\!](\sigma^j)\}\}
\end{array}$$

Common derived operators are
$\varphi_1 \wedge \varphi_2 \overset{\text{def}}{=} \neg(\varphi_1 \vee \varphi_2)$, $\Diamond\varphi \overset{\text{def}}{=} T\,\mathcal{U}\,\varphi$ and

$\Box\varphi \stackrel{\text{def}}{=} \neg\Diamond\neg\varphi$. In this paper we use a map $[\![\cdot]\!]$ for the semantics, as opposed to a relation $\models$, for uniformity with Section 4 below.

2.1 LTL Property Classification

In [1], Havelund and Peled introduce a property classification according to the capability of a (semantic) monitor to return a single value witnessing a finite prefix of a trace. The original definitions are the following. Given a property φ:

- **Safety**, or Always Finitely Refutable (AFR): When φ does not hold for a behavior, a failed verdict can always be identified after a sufficiently long finite prefix.
- **Guarantee**, or Always Finitely Satisfiable (AFS): When φ is satisfied on a behavior, a satisfied verdict can be identified after a finite prefix.
- **Liveness**, or Never Finitely Refutable (NFR): When φ does not hold on a behavior, a refutation can not be identified after a finite prefix.
- **Morbidity**, or Never Finitely Satisfiable (NFS): When φ is satisfied on a behavior, satisfaction can not be identified after a finite prefix.

Havelund and Peled also define two extra property classes that are not given a name:

- Sometimes Finitely Refutable (SFR): For some behaviors that violate φ, a refutation can be identified after a finite prefix; while for other behaviors violating φ, a refutation cannot be identified with a finite prefix.
- Sometimes Finitely Satisfiable (SFS): For some behaviors that satisfy φ, satisfaction can be identified after a finite prefix; while for other behaviors satisfying the property, satisfaction cannot be identified with a finite prefix.

Fig. 1 shows the landscape of property classes along with an example LTL property for every class.

We can see, for example, that $\Diamond p$ belongs to **Guarantee** and **Liveness**. This property is NFR because given any finite prefix of a trace where the property does not hold, we can construct an alternative continuation where it does hold, simply making the next value of p be *true*. The property is also AFS because we can find the first index

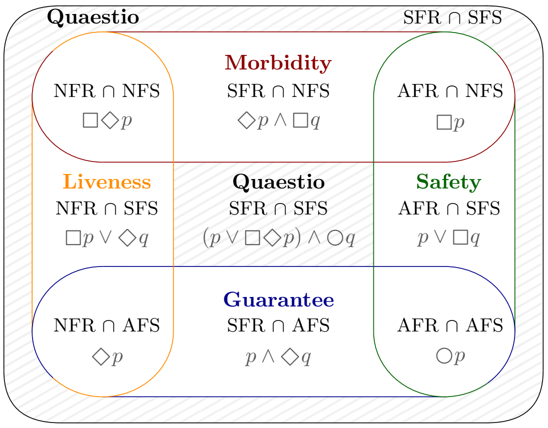

Fig. 1 Landscape of property classes according to [1].

when p becomes *true* and any continuation of that prefix makes the property *true*.

One final remark is that [1] implicitly ignores tautologies and contradictions. Strictly according to the definitions above, contradictions are at the same time AFR, AFS and NFS; while tautologies are at the same time AFS, AFR and NFR. We have made the decision to consider only monitorability with respect to verdicts that are not statically dismissible, as monitoring is intrinsically the activity of obtaining information at runtime. In this paper we use AFR, NFR, SFR, AFS, NFS, and SFS to classify properties in any Boolean formalism (in [1] these classes are introduced for LTL).

3 A Richer View of Monitorability

In this section we generalize the framework of Havelund and Peled [1] to consider richer verdicts by considering domains other than Booleans. Similar to the approach in [1], we base our work on the ability of a semantic monitor to reach a verdict after only witnessing a finite observation.

The main intuition of our approach is to focus on the dismissibility of verdict values. The finite satisfiability of a property means that after a finite observation a semantic monitor can *dismiss* the value *false* as the result, and the finite refutability of a property means that with a finite observation a semantic monitor can *dismiss* the value *true* as the result.

Let us consider a formalism that assigns verdicts of type D to behaviors I^ω, with a semantic assignment $[\![\cdot]\!]$. We say a value $v \in D$, a potential verdict, is *Finitely Dismissible* for a formula φ and a behavior $s \in I^\omega$ if there is an observation $ls \in I^*$, $ls \prec s$ such that $v \notin M_\varphi(ls)$, where M_φ is the semantic monitor for φ. We say a value $v \in D$ is *Finitely Admissible* for a formula φ and a behavior $s \in I^\omega$ if there is an observation $ls \in I^*$, $ls \prec s$ such that that $M_\varphi(ls) = \{v\}$. Note that the only value that can be Finitely Admissible for φ over s is $[\![\varphi]\!](s)$.

Recall that a special case of finite dismissibility of a value v for a property φ is when $v \notin M_\varphi(\epsilon)$, that is, when v is statically dismissible. In this paper we focus on dynamically dismissing the values that cannot be discarded statically, and exclusively study the monitorability of verdicts that are, in principle, possible outcomes of the monitoring process. We are interested in determining whether a monitor can conclude that the verdict is discarded after a finite observation.

Example 4 For example, consider the property $\varphi(s) = \min\limits_{0 \leq i}(s(i)(p))$ for an infinite sequence s of non-negative integer values of p for which $[\![\varphi]\!](s) = 0$. In this case, the value -1 can be statically dismissed because all values are non-negative, the value $s(0)(p) + 1$ is finitely dismissible because after the first step $s(0)(p)$ is observed and the final verdict must be at most $s(0)(p) < s(0)(p)+1$. Note that 0 is finitely admissible because $[\![\varphi]\!](s) = 0$ and after 0 is observed the verdict is determined.

We say that a set of values $D' \subseteq D$ is *None Finitely Dismissible* (NFD) for a formula φ and a behavior s if every $v \in D'$ that is not statically dismissible for φ is also not Finitely Dismissible for φ and s. Analogously, we say that a set of values $D' \subset D$ is *All Finitely Dismissible* (AFD) for a formula φ and a behavior s if every $v \in D'$ is Finitely Dismissible for φ and s. Considering the property from Ex. 4, the set of values in the interval $(s(0)(p), \infty)$ are AFD, and the set of values $(0, [\![\varphi]\!](s))$ are NFD. Note that the empty set is both NFD and AFD, as are the sets that contain only statically dismissible values. We say that a set of values $D' \subset D$ is *Some Finitely Dismissible* (SFD) if it is neither AFD nor NFD. We can extend the definition of Finite Admissibility to sets of values but they are of little use in our work.

Proposition 1 *If v is Finitely Admissible for a formula φ and a behavior s then $D \setminus \{v\}$ is* AFD *for φ and s.*

Proof Since v is Finitely Admissible for φ and s, there is a finite sequence $ls \prec s$ such that $M_\varphi(ls) = \{v\}$. We can therefore dismiss any value in $D \setminus \{v\}$ witnessing the finite prefix ls. $\square$

The converse of Proposition 1 holds for finite domains.

Proposition 2 *If $D \setminus \{v\}$ is* AFD *for a formula φ and a behavior s and D is finite, then v is Finitely Admissible for φ and s.*

Proof For every element v' in $D \setminus \{v\}$ there is a prefix $ls_{v'}$ such that $v' \notin M_\varphi(ls_{v'})$. Let ls be the prefix of maximum length among those. After witnessing ls, every element in $D \setminus \{v\}$ is not in $M_\varphi(ls)$, so $M_\varphi(ls) = \{v\}$. $\square$

However, if D is infinite, Proposition 2 does not hold.

Proposition 3 *If $D \setminus \{v\}$ is* AFD *for a formula φ and a behavior s and D is infinite, then it is not necessarily the case that v is Finitely Admissible for φ and s.*

Proof Let φ be a property that assigns the maximum value of the field p (of type $\mathbb{N}$) in the behavior if it exists, and ∞ otherwise. The verdict is of type $\mathbb{N} \cup \{\infty\}$ and for the behavior $s \overset{\text{def}}{=} (\langle p : 1\rangle\ \langle p : 2\rangle\ \langle p : 3\rangle\ \ldots)$, the semantics of φ is $[\![\varphi]\!](s) = \infty$, any natural number is finitely dismissible and yet ∞ is not finitely admissible: we can simply continue the observation by repeating the last value ad-infinitum, creating a behavior whose semantics, according to φ, is any natural number (and not ∞). $\square$

We will later show two more counter-examples for the converse of Proposition 1 for bounded, dense verdict domains in Sections 4.2 and 4.3.

4 Boolean and Quantitative Totally Ordered Domains

In this section we instantiate the framework in the previous section to totally ordered sets, according to the dismissibility of values with respect to the result, and then apply to two well-known quantitative temporal logics.

4.1 Property classes

If the type D of the verdicts of a formalism is a totally ordered set equipped with an order relation $(D, \leq)$, we can classify the properties according to their value-dismissibility as follows. Note that we only require that the domain of verdicts be totally ordered, so the domain of observations can be different to the domain of verdicts, and potentially not an ordered set. Let $v = [\![\varphi]\!](\sigma)$ be the semantics of the property φ for behavior σ. We use $v_<$ for the set of values lower than v and and $v_>$ for the set of values greater than v, that is $v_< \overset{\text{def}}{=} \{v' \mid v' < v\}$ and $v_> \overset{\text{def}}{=} \{v' \mid v' > v\}$. Note that $D = v_< \cup \{v\} \cup v_>$. Recall that we only consider the set of potential verdicts $M_\varphi(\epsilon)$ that cannot be dismissed statically for φ. That is, a value v is only interesting to study if a semantic monitor cannot dismiss it as a potential verdict before the monitoring starts.

We say a property is $\mathsf{AFD}_>$ if the set of values greater than its verdict for any behavior is AFD, that is, if all the values greater than the verdict that are not statically dismissible, are finitely dismissible. We define $\mathsf{AFD}_<$, $\mathsf{NFD}_>$ and $\mathsf{NFD}_<$ analogously. A property is $\mathsf{SFD}_>$ if for some executions, some values greater than its verdict are finitely dismissible while other are not. The definition of $\mathsf{SFD}_<$ is analogous.

For example, the property

$$\varphi(s) = \min_{0 \leq i}(s(i)(p))$$

from Ex. 4 is $\mathsf{AFD}_>$ and also $\mathsf{NFD}_<$. If the verdict is 1, the empty set of values greater than the verdict is finitely dismissible. If the verdict is $v < 1$, for any v' greater than v, at some point we will witness a value in the range $[v, v')$ (otherwise, v' would be the minimum), and then we can dismiss v' as a potential verdict. Thus, the property is $\mathsf{AFD}_>$. For any value v' lower than the verdict we can create an alternative continuation that consists of v' forever after any prefix, making the verdict v', which is why the property is $\mathsf{NFD}_<$.

On the other hand, the property

$$\varphi_1(s) = \max(s(0)(q), \min_{0 \leq i}(s(i)(p)))$$

is $\mathsf{AFD}_>$ and $\mathsf{SFD}_<$. The property is not $\mathsf{NFD}_<$ because if $s(0)(q) = 1$ we can dismiss every value lower than the verdict finitely. Similarly, the property is not $\mathsf{AFD}_<$ because if $s(0)(q) = 0$ we cannot dismiss any value lower than the verdict finitely. Thus, the property is $\mathsf{SFD}_<$.

The property

$$\varphi_2(s) = s(0)(p)$$

is trivially in $\mathsf{AFD}_>$ and $\mathsf{AFD}_<$. The verdict is known at the first instant, so any other value can be finitely dismissed. Finally,

$$\varphi_3(s) = \min_{0 \leq i}(\max_{j \leq i}(s(i)(p)))$$

is in $\mathsf{NFD}_>$ and $\mathsf{NFD}_<$, since we can take any prefix and value v and create a continuation that consists of v indefinitely, making v the verdict of the property over such trace.

With these definitions we can redefine the property classes for rich, totally ordered domains as follows[1]:

- **Safety**/$\mathsf{AFD}_>$: We say that a property is a *Safety* property if the set $v_>$ is All Finitely Dismissible for any behavior σ (that is, a semantic monitor can dismiss every value greater than the result after a finite prefix). In other words, setting a threshold t that happens to be above the actual verdict guarantees that after a prefix all values above the threshold will be dismissed.
- **Guarantee**/$\mathsf{AFD}_<$: We say that a property is a *Guarantee* property if the set $v_<$ is All Finitely Dismissible for any behavior σ (that is, a semantic monitor can dismiss every value lower than the verdict after a sufficiently long prefix). In other words, if you set a minimum score t and the result is higher than it, you will know it with a finite prefix.
- **Liveness**/$\mathsf{NFD}_>$: We say a property is a *Liveness* property if the set $v_>$ is None Finitely Dismissible for any behavior σ (that is, a semantic monitor can never dismiss any value greater than the result processing any prefix). In other words, if you set a maximum

[1]Note that, even though we borrow the names (safety, liveness, ...) from [1] and [38], our notions do not preserve the decomposition of formulas into safety and liveness from the Boolean domain (as opposed to the notions in [30] which preserve the classical decompositions).

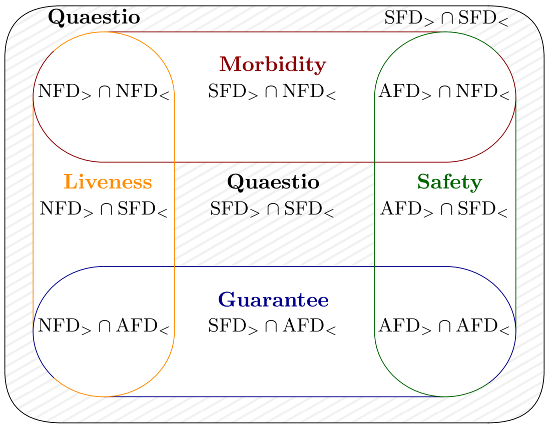

Fig. 2 Landscape of property classes for totally ordered domains

tolerable threshold t and the result is below t, then you will not know that the value is below after any finite prefix.

- **Morbidity**/NFD$_<$: We say a property is a *Morbidity* property if the set $v_<$ is None Finitely Dismissible for any behavior σ (that is, a semantic monitor can never dismiss any value lower than the result with a prefix). In other words, if you set a minimum score t and the result is higher than t, you will not know after a finite prefix.

Fig. 2 shows the resulting landscape of property classes for totally ordered domains. Note that the definitions of **Safety** and **Liveness** are incompatible for verdict domains with more than one element, and so are the definitions of **Guarantee** and **Morbidity**, which means that no property can be both a **Safety** and a **Liveness** property, or can be both a **Guarantee** and a **Morbidity** property. However, it is possible that a property belongs to two classes, and also that a property does not belong to any of the classes described above.

We see that our definitions maintain the classification of the original LTL properties presented in [1] if we consider the Boolean domain with the usual order relation *false* < *true*. Recall that according to our definitions, a **Safety** property is one such that a semantic monitor can always dismiss the values greater than the result with a finite prefix. This is equivalent to saying, in the Boolean ordered set, that if the result is *false* then a semantic monitor can always dismiss the set $\{true\}$ with a prefix. Since the domain is finite, Proposition 2 implies that the value *false* is always Finitely Admissible, and thus, a failed verdict can be identified after a finite prefix. A similar analysis can be made for the rest of the classes.

In the following sections we will give a witness for every class and sensible multiclass for two extensions of LTL to quantitative domains.

4.2 Quantitative LTL

In [23] the authors define quantitative semantics for LTL, which generalize the semantics from Boolean to a numeric domain. Input streams are streams of real numbers in the range $[0, 1]$. The intuition is that the numeric value for atomic variable p at a given instant is the extent to which p is "true" (1 being absolutely true and 0 being absolutely false). The syntax is the same as for the classic LTL. The semantics is given recursively over the terms and assigns a value in the range $[0, 1]$ for every term with respect to a behavior that assigns a real number in the range $[0, 1]$ to every proposition, this is, in QLTL, $R \stackrel{\text{def}}{=} \langle p_0 :: \mathbb{R}_{[0,1]}, \ldots, p_n :: \mathbb{R}_{[0,1]}\rangle$.

$$\begin{aligned}
[\![T]\!](\sigma) &\stackrel{\text{def}}{=} 1 \\
[\![\varphi \vee \psi]\!](\sigma) &\stackrel{\text{def}}{=} \max\{[\![\varphi]\!](\sigma), [\![\psi]\!](\sigma)\} \\
[\![a]\!](\sigma) &\stackrel{\text{def}}{=} \sigma(0)(a) \\
[\![\neg\varphi]\!](\sigma) &\stackrel{\text{def}}{=} 1 - [\![\varphi]\!](\sigma) \\
[\![\bigcirc\varphi]\!](\sigma) &\stackrel{\text{def}}{=} [\![\varphi]\!](\sigma^1) \\
[\![\varphi\,\mathcal{U}\,\psi]\!](\sigma) &\stackrel{\text{def}}{=} \sup_{i\geq 0}\{\min\{[\![\psi]\!](\sigma^i), \\
&\qquad\qquad \min_{0\leq j<i}\{[\![\varphi]\!](\sigma^j)\}\}\}
\end{aligned}$$

Following the syntax of the derived operators, their semantics in QLTL are

$$\begin{aligned}
[\![\varphi \wedge \psi]\!](\sigma) &\stackrel{\text{def}}{=} \min\{[\![\varphi]\!](\sigma), [\![\psi]\!](\sigma)\} \\
[\![\Diamond\varphi]\!](\sigma) &\stackrel{\text{def}}{=} \sup_{i\geq 0}\{[\![\varphi]\!](\sigma^i)\} \\
[\![\Box\varphi]\!](\sigma) &\stackrel{\text{def}}{=} \inf_{i\geq 0}\{[\![\varphi]\!](\sigma^i)\}
\end{aligned}$$

Note that under the assumption that the values of the atomic propositions are always 0 or 1, a semantic monitor can statically dismiss the values in the open interval $(0, 1)$ as the possible verdicts of a QLTL property and hence the classification of QLTL under this assumption coincides with that of LTL.

The formulae presented in Section 2.1 for LTL belong to the same classes in QLTL with no assumptions.

Proposition 4 *The following hold:*

- $\Diamond p$ *belongs to* **Guarantee** *and* **Liveness***,*
- $\Box p$ *belongs to* **Safety** *and* **Morbidity***,*
- $\bigcirc p$ *belongs to* **Safety** *and* **Guarantee***,*
- $\Box\Diamond p$ *belongs to* **Morbidity** *and* **Liveness***,*
- $p \wedge \Diamond q$ *only belongs to* **Guarantee***,*
- $\Box p \vee \Diamond q$ *only belongs to* **Liveness***,*
- $p \vee \Box q$ *only belongs to* **Safety***, and*
- $\Diamond p \wedge \Box q$ *only belongs to* **Morbidity***.*

Proof We show that $\Diamond p$ belongs to **Guarantee** and **Liveness**. $\Diamond p$ is $\mathsf{NFD}_>$ because given any finite prefix of a trace where the supremum is $v \neq 1$, we can construct an alternative continuation where it is greater than v simply making the next value 1. If the verdict is $v = 1$, the (empty) set of values $v_>$ would be trivially NFD. It is $\mathsf{AFD}_<$ because the verdict v is the minimum element greater or equal than the infinite values of p throughout the trace. Let $v' < v$. If no element in $(v', v]$ occur in the trace, then the result would be v'. Otherwise, the occurrence of such value would dismiss v' as a possible result. The proof that $\Box p$ belongs to **Safety** and **Morbidity** is dual. The verdict of the property $\bigcirc p$ is Finitely Admissible, which means that any other value is fd, and the property belongs to **Safety** and **Guarantee**.

We show that the property $\Box\Diamond p$ belongs to **Morbidity** and **Liveness**. It is $\mathsf{NFD}_>$ because given any finite prefix of a trace where the result is $v \neq 1$, we can construct an alternative continuation where the result is greater than v, simply making $p = 1$ forever. Analogously, the property is $\mathsf{NFD}_<$ because given any finite prefix of a trace where the result is $v \neq 0$, we can construct an alternative continuation where the result is lower than v, simply making $v = 0$ forever. □

In general and without assumptions, the verdict of a QLTL property is a real number in the range $[0, 1]$, that is an infinite set. Therefore, it is not covered by Proposition 2. The following propositions show that in fact, in some cases, the set of values different from the result is AFD but the result itself is not Finitely Admissible.

Proposition 5 *There is a QLTL property φ and a behavior s such that $v = [\![\varphi]\!](s)$ is not Finitely Admissible, but $[0, 1] \setminus \{v\}$ is AFD.*

Proof Consider the property $(\Diamond p \wedge q)$ and a behavior s such that, at every instant i, the value of q is $\frac{1}{2}$ and the value of p is $\sum_{n=0}^{i} \frac{1}{4\times 2^n}$, that is, p produces values closer to $\frac{1}{2}$, but never $\frac{1}{2}$ exactly. Then, $[\![\Diamond p]\!](s) = \frac{1}{2}$, the set $[0, \frac{1}{2}) \cup (\frac{1}{2}, 1]$ is All Finitely Dismissible, but the result $\frac{1}{2}$ is not Finitely Admissible. □

4.3 Discounting in LTL

We now study, $\mathrm{Disc}^{\mathrm{LTL}}[\mathcal{D}]$, a temporal logic that generalizes LTL by adding discounting temporal operators [39]. According [39], $\mathrm{Disc}^{\mathrm{LTL}}[\mathcal{D}]$ is in fact a family of logics, each parameterized by a set $\mathcal{D}$ of discounting functions. A function $\eta : \mathbb{N} \to [0, 1]$ is a *discounting function* if $\lim_{i\to\infty} \eta(i) = 0$, and η is strictly decreasing. Input streams are Boolean, as in classic LTL, but verdicts are real numbers in the range $[0, 1]$. Note that in this case, the domain of observations and the domain of verdicts are different. We reason about the potential verdicts, that is, the real numbers in $[0, 1]$.

For a given a set of discounting functions $\mathcal{D}$, the logic $\mathrm{Disc}^{\mathrm{LTL}}[\mathcal{D}]$ adds to LTL the binary operator $\mathcal{U}_\eta$. The semantics of this logic is given recursively over the terms and assigns a value in the range $[0, 1]$ for every term with respect to a behavior, assigning 0 to an input value of *false* and 1 to an input value of *true*. The semantics of $\mathrm{Disc}^{\mathrm{LTL}}[\mathcal{D}]$ is:

$$\begin{aligned}
[\![T]\!](\sigma) &\overset{\mathrm{def}}{=} 1\\
[\![\varphi \vee \psi]\!](\sigma) &\overset{\mathrm{def}}{=} \max\{[\![\varphi]\!](\sigma), [\![\psi]\!](\sigma)\}\\
[\![a]\!](\sigma) &\overset{\mathrm{def}}{=} \begin{cases} 1 & \text{if } \sigma(0)(a) = \mathit{true}\\ 0 & \text{otherwise}\end{cases}\\
[\![\neg\varphi]\!](\sigma) &\overset{\mathrm{def}}{=} 1 - [\![\varphi]\!](\sigma)\\
[\![\bigcirc\varphi]\!](\sigma) &\overset{\mathrm{def}}{=} [\![\varphi]\!](\sigma^1)\\
[\![\varphi\,\mathcal{U}\,\psi]\!](\sigma) &\overset{\mathrm{def}}{=} \sup_{i\geq 0}\{\min\{[\![\psi]\!](\sigma^i),\\
&\qquad \min_{0\leq j<i}\{[\![\varphi]\!](\sigma^j)\}\}\}\\
[\![\varphi\,\mathcal{U}_\eta\,\psi]\!](\sigma) &\overset{\mathrm{def}}{=} \sup_{i\geq 0}\{\min\{\eta(i)\cdot[\![\psi]\!](\sigma^i),\\
&\qquad \min_{0\leq j<i}\{\eta(j)\cdot[\![\varphi]\!](\sigma^j)\}\}\}
\end{aligned}$$

All properties in Section 4.2 belong to the same categories (without assumptions) which is reasonable because they do not use discounting functions, and the semantics guarantee that the outcome of the formulas is the same as for LTL. For the same reason, and since the observations are Boolean values for this logic, the only possible verdicts are $\{0,1\}$ and thus the semantics and the property classes coincide with those of classic LTL if $\mathcal{U}_\eta$ is not used.

Proposition 6 *Properties of the form $\varphi \, \mathcal{U}_\eta \, \psi$ belong to* **Safety** *and* **Guarantee**.

Proof Let v be $[\![\varphi \mathcal{U}_\eta \psi]\!](\sigma)$. If $v > 0$, there is an index k such that $\eta(k) < v$. After this index, the successive values of the expression $(\min\{\eta(i) \cdot [\![\psi]\!](\sigma(i)), \min_{0 \le j < i}\{\eta(j) \cdot [\![\varphi]\!](\sigma(j))\}\})$ will always be strictly lower than v, which means that (1) v has already occurred, and (2) it is guaranteed to be the supremum. This means that if $v > 0$, at index k we know the verdict and the semantic monitor can discard every other value.

In the special case of $v = 0$, the semantic monitor can trivially dismiss the (empty) set of values lower than it. Let $v' \in (0,1]$. There is an index k such that $\eta(k) < v'$. The expression $\eta(i) \cdot [\![\psi]\!](\sigma(i))$ will be lower than v' for every $i \ge k$, which means that after index k, the possible successive values of the whole expression $(\min\{\eta(i) \cdot [\![\psi]\!](\sigma(i)), \min_{0 \le j < i}\{\eta(j) \cdot [\![\varphi]\!](\sigma(j))\}\})$ can only be lower than v', meaning that the monitor can discard v' as a potential verdict after index k. □

$\mathrm{Disc}^{\mathrm{LTL}}[\mathcal{D}]$ provides another example where the set of non-verdicts is All Finitely Dismissible but the correct result is not Finitely Admissible.

Proposition 7 *There is a $Disc^{LTL}[\mathcal{D}]$ property φ and a behavior s such that $v = [\![\varphi]\!](s)$ is not Finitely Admissible, but $[0,1] \setminus \{v\}$ is* AFD.

Proof Consider a behavior s such that p is always *false*, and the $\mathrm{Disc}^{\mathrm{LTL}}[\mathcal{D}]$ property $\varphi = \Diamond_\eta p(s)$. The temporal operator $\Diamond_\eta$ is defined as $\Diamond_\eta \varphi \overset{\text{def}}{=} T \, \mathcal{U}_\eta \, \varphi$. From the semantics of $\mathcal{U}$ and T, we see that

$$\begin{aligned} [\![\varphi]\!](s) &= \sup_{i \ge 0}\{\min\{\eta(i) \cdot \sigma(i)(p), \min_{0 \le j < i}\{\eta(j) \cdot 1\}\}\} \\ &= \sup_{i \ge 0}\{\min\{\eta(i) \cdot 0, \min_{0 \le j < i}\{\eta(j)\}\}\} \\ &= \sup_{i \ge 0}\{\min_{0 \le j < i}\{\eta(j)\}\} \end{aligned}$$

Then, $[\![\varphi]\!](s) = 0$, the set $(0,1]$ is All Finitely Dismissible, but the result 0 is not Finitely Admissible. □

5 More General Monitorability Classes

In this section we refine the taxonomy in the previous section exploiting the following observation. The definition of SFD as the complement of the other classes is somewhat unsatisfying as it collapses several different cases. We now split SFD into two different classes, which in turn generates dualities with respect to the complements of the classes and the finite dismissibility of values.

5.1 A Finer Classification of SFD

For a given a formula φ, we can classify the finite-dismissibility (fd) of a set of values D analyzing, for each of the following four definitions, whether φ meets the criteria or not:

- $\mathsf{A\forall fd}_D$. For all the behaviors, all the values in D are finitely dismissible.
- $\mathsf{A\exists fd}_D$. For all the behaviors, there is a value in D which is finitely dismissible.
- $\mathsf{S\forall fd}_D$. For some behaviors, all the values in D are finitely dismissible.
- $\mathsf{S\exists fd}_D$. For some behaviors, there is a value in D which is finitely dismissible.

We use $\overline{c}$ to denote the complement of a class c and we say that a property is in $\overline{c}$ iff it is not in c. For example, a property is in $\overline{\mathsf{A\forall fd}_D}$ iff it is not in $\mathsf{A\forall fd}_D$. We also introduce the equivalent definitions for non-dismissibility ($\overline{\mathsf{fd}}$) of a value as follows:

- $\mathsf{A\forall\overline{fd}}_D$. For all the behaviors, all the values in D are *not* finitely dismissible. Equivalently, there is no behavior that makes any value in D finitely dismissible, which means that $\mathsf{A\forall\overline{fd}}_D = \overline{\mathsf{S\exists fd}_D}$.
- $\mathsf{A\exists\overline{fd}}_D$. For all the behaviors, there is a value in D which is *not* finitely dismissible. Equivalently, there is no behavior for which all the values in D are finitely dismissible, meaning that $\mathsf{A\exists\overline{fd}}_D = \overline{\mathsf{S\forall fd}_D}$.
- $\mathsf{S\forall\overline{fd}}_D$. For some behaviors, all the values in D are *not* finitely dismissible. Equivalently, it is not the case that all the behaviors contain some value in D which is finitely dismissible, meaning that $\mathsf{S\forall\overline{fd}}_D = \overline{\mathsf{A\exists fd}_D}$.
- $\mathsf{S\exists\overline{fd}}_D$. For some behaviors, there is a value in D which is *not* finitely dismissible. Equivalently, it is not the case that for all the

behaviors, all the values in D are finitely dismissible, which means that $\mathsf{S\exists\overline{fd}}_D = \overline{\mathsf{A\forall fd}_D}$. Therefore, in principle, a property can be classified using only four dimensions ($\mathsf{S\exists fd}$, $\mathsf{S\forall fd}$, $\mathsf{A\exists fd}$, $\mathsf{A\forall fd}$), which gives an upper-bound of 16 different categories. We now analyze what we can expect when we aim to monitor a property, considering each dimension:

- When we monitor a property in $\mathsf{A\forall fd}_D$, it is guaranteed that, for any value d in D and after a finite prefix, a semantic monitor can dismiss d as the verdict. In particular, this could imply that the verdict is finitely admissible.

  On the other hand, if the property is in $\overline{\mathsf{A\forall fd}_D} = \mathsf{S\exists\overline{fd}}_D$, it means that it may be the case that in the monitored trace our value of interest $d \in D$ is $\overline{\mathsf{fd}}$; but it still makes sense to monitor the property in case that d is fd.
- When a property is in $\mathsf{A\exists fd}_D$, there is at least one value in D that can be finitely dismissible by a semantic monitor, that is, that we can always gain information about the dismissibility of at least one value in D when we monitor the property.

  When a property is in $\overline{\mathsf{A\exists fd}_D} = \mathsf{S\forall\overline{fd}}_D$, it could be the case that for the specific trace we are witnessing, all the values in D are not finitely dismissible. It still makes sense to monitor the property, in case we are monitoring a more friendly trace.
- When we monitor a property in $\mathsf{S\forall fd}_D$, it could be the case that we receive a trace where every value in D is fd, so it makes sense to monitor the property.

  If instead the property is in $\overline{\mathsf{S\forall fd}_D} = \mathsf{A\exists\overline{fd}}_D$, it is guaranteed that at least one value in D will not be fd, although others may be, so it makes sense to monitor a property with the hope that we will dismiss a specific value.
- When a property is in $\mathsf{S\exists fd}_D$, we can hope that our value of interest $d \in D$ is finitely dismissible in the specific trace we are monitoring.

  Instead, if the property is in $\overline{\mathsf{S\exists fd}_D} = \mathsf{A\forall\overline{fd}}_D$, it is guaranteed that the monitor will never be able to dismiss any value in D.

We now refine this result even further as not all the definitions intersect with each other. In particular, since we are assuming the existence of behaviors

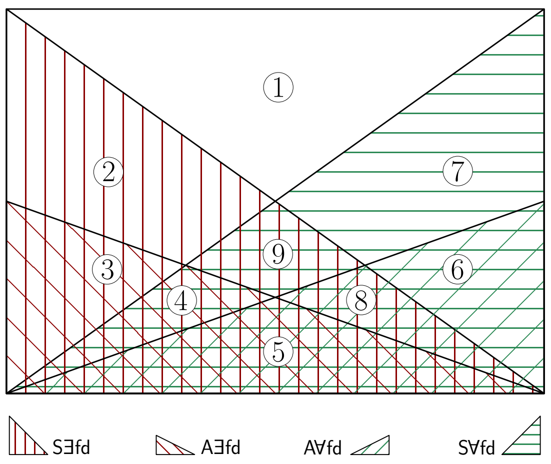

Fig. 3 General landscape of property classes

in the properties we analyze, if for a given property a characteristic holds for all behaviors then it follows that there is a behavior for which the characteristic holds. Formally:

$$\begin{array}{ll} \mathsf{A\forall fd}_D \subseteq \mathsf{S\forall fd}_D & \mathsf{A\forall\overline{fd}}_D \subseteq \mathsf{S\forall\overline{fd}}_D \\ \mathsf{A\exists fd}_D \subseteq \mathsf{S\exists fd}_D & \mathsf{A\exists\overline{fd}}_D \subseteq \mathsf{S\exists\overline{fd}}_D \end{array}$$

This means that we do not need to consider properties with $\mathsf{A\forall fd}_D$ and $\overline{\mathsf{S\forall fd}_D}$, and with $\mathsf{A\exists fd}_D$ and $\overline{\mathsf{S\exists fd}_D}$. This leaves 9 potential classes.

We can see that the definitions from Section 4 of NFD, AFD and SFD can be rewritten using these new classifications. We say that a set of values D is *None Finitely Dismissible* (NFD) if it is $\mathsf{A\forall\overline{fd}}_D$. Analogously, we say that a set of values D is *All Finitely Dismissible* (AFD) if it is $\mathsf{A\forall fd}_D$. Note that the empty set is both NFD and AFD. We say that a set of values D is *Some Finitely Dismissible* (SFD) if it is neither AFD nor NFD. Note that AFD and NFD can overlap, in particular when the set of values D is empty: always all the values in an empty set can be finitely dismissed, but also always all the values in an empty set cannot be finitely dismissed.

From now on, we omit the subscript D for clarity, as it is fixed in the context. Fig. 3 shows the general landscape of property classes for a particular set of values. Note that the region of $\mathsf{A\exists fd}$ is contained in $\mathsf{S\exists fd}$ and $\mathsf{A\forall fd}$ is contained in $\mathsf{S\forall fd}$ as mentioned above. Therefore, a given property belongs to one of the following nine classes (the gray characteristics are implied by other characteristics):

1. $\overline{\mathsf{S\exists fd}} \cap \overline{\mathsf{S\forall fd}} \cap [\overline{\mathsf{A\exists fd}}] \cap [\overline{\mathsf{A\forall fd}}]$
2. $\mathsf{S\exists fd} \cap \overline{\mathsf{S\forall fd}} \cap \overline{\mathsf{A\exists fd}} \cap [\overline{\mathsf{A\forall fd}}]$
3. $[\mathsf{S\exists fd}] \cap \overline{\mathsf{S\forall fd}} \cap \mathsf{A\exists fd} \cap [\overline{\mathsf{A\forall fd}}]$
4. $[\mathsf{S\exists fd}] \cap \mathsf{S\forall fd} \cap \mathsf{A\exists fd} \cap \overline{\mathsf{A\forall fd}}$
5. $[\mathsf{S\exists fd}] \cap [\mathsf{S\forall fd}] \cap \mathsf{A\exists fd} \cap \mathsf{A\forall fd}$
6. $\overline{\mathsf{S\exists fd}} \cap [\mathsf{S\forall fd}] \cap [\overline{\mathsf{A\exists fd}}] \cap \mathsf{A\forall fd}$
7. $\overline{\mathsf{S\exists fd}} \cap \mathsf{S\forall fd} \cap [\overline{\mathsf{A\exists fd}}] \cap \overline{\mathsf{A\forall fd}}$
8. $\mathsf{S\exists fd} \cap [\mathsf{S\forall fd}] \cap \overline{\mathsf{A\exists fd}} \cap \mathsf{A\forall fd}$
9. $\mathsf{S\exists fd} \cap \mathsf{S\forall fd} \cap \overline{\mathsf{A\exists fd}} \cap \overline{\mathsf{A\forall fd}}$

5.2 Example Classifications

As we have done in Section 4, we analyze the finite dismissibility of values greater or lower than the verdict for logics where the set of potential verdicts is totally ordered.

Boolean Formalisms.

First, consider finite dismissibility of the values greater than the verdict in a Boolean formalism, that is, one where the domain of potential verdicts is the Boolean domain. We use the subscript $>$ to refer to the set of values greater than the verdict. Regions 1 to 5 contain no properties; region number 6 contains all the tautologies and contradictions; region number 7 contains all the properties that are NFR, but are not tautologies; region number 8 contains all the properties that are AFR, but are neither tautologies nor contradictions; and region number 9 contains all the properties that are SFR.

We will prove that region 6 contains all the tautologies and contradictions.

Proposition 8 *Let φ be an property in a Boolean formalism. Then, φ belongs to $\overline{\mathsf{S\exists fd}_>} \cap \mathsf{A\forall fd}_>$ if and only if φ is a tautology or a contradiction.*

Proof Let φ in $\overline{\mathsf{S\exists fd}_>} \cap \mathsf{A\forall fd}_>$. Since $\overline{\mathsf{S\exists fd}_>} = \mathsf{A\forall}\overline{\mathsf{fd}}_>$, and $\varphi \in \mathsf{A\forall fd}_>$, then all the values greater than the verdict have to be both fd and $\overline{\mathsf{fd}}$ at the same time. Thus, the set of values greater than the verdict which are not statically dismissible has to be empty. If there is a trace that admits *false* as the verdict and there is a trace that admits *true* as a verdict, when the verdict is *false* the set of values greater than the result is the nonempty set $\{true\}$. Consequently, φ can only admit one of the Boolean values as its verdict for all the traces, that is, φ is either a tautology or a contradiction.

Let φ be a tautology or a contradiction. In that case, the verdict for every trace is the same and every other values is statically dismissible. Since in particular the set of values greater than the verdict and not statically dismissible is empty, the property belongs to both $\mathsf{A\forall fd}_>$ and $\mathsf{A\forall}\overline{\mathsf{fd}}_> = \overline{\mathsf{S\exists fd}_>}$. □

The proof that regions 1 to 5 are empty in Boolean formalisms follows from the following proposition:

Proposition 9 *In Boolean formalisms, the sets $\mathsf{A\exists}\overline{\mathsf{fd}}_>$ (that is, $\overline{\mathsf{S\forall fd}_>}$) and $\mathsf{A\exists fd}_>$ are empty.*

Proof Let φ be a property in $\mathsf{A\exists}\overline{\mathsf{fd}}_>$ or $\mathsf{A\exists fd}_>$. Since there has to be a value greater than the verdict, the verdict cannot be *true* for any trace for φ. Thus, φ is a contradiction and belongs to region 6, as shown in Proposition 8. □

We prove now that region 7 contains all the properties that are NFR.

Proposition 10 *Let φ be an property in a Boolean formalism. Then, φ belongs to $\overline{\mathsf{S\exists fd}_>} \cap \mathsf{S\forall fd}_> \cap \overline{\mathsf{A\forall fd}_>}$ if and only if φ is NFR and is not a tautology.*

Proof Let φ in $\overline{\mathsf{S\exists fd}_>} \cap \mathsf{S\forall fd}_> \cap \overline{\mathsf{A\forall fd}_>}$. Since φ belongs to $\overline{\mathsf{S\exists fd}_>} = \mathsf{A\forall}\overline{\mathsf{fd}}_>$, then for the traces for which the verdict is *false*, the set $\{true\}$ is $\overline{\mathsf{fd}}$, and thus we can never accept the verdict *false* (i.e. refute the property) with a finite prefix. This means that φ is NFR, and it is not a tautology because it would belong to region 6.

Let φ be an NFR property, but not a tautology. When the verdict is *false*, the set $\{true\}$ is $\overline{\mathsf{fd}}$ because φ is is NFR. When the verdict is *true*, the empty set of values greater than the verdict is $\overline{\mathsf{fd}}$. Thus, the property is in $\mathsf{A\forall}\overline{\mathsf{fd}}_> = \overline{\mathsf{S\exists fd}_>}$. There are traces for which the verdict is *true* (because φ is in NFR and thus not a contradiction). For those traces the set of values greater than the verdict is empty, and thus the property is in $\mathsf{S\forall fd}_>$. There are also traces for which the verdict is *false* (because φ is not a tautology). For those traces the value $true > false$ is not finitely dismissible because otherwise we would be able to refute the property with a finite prefix, contradicting the fact that φ is NFR. Therefore, the property is in $\mathsf{S\exists}\overline{\mathsf{fd}}_> = \overline{\mathsf{A\forall fd}_>}$. □

We now prove that region 8 contains all the properties that are AFR but are neither tautologies nor contradictions.

Proposition 11 *Let φ be an property in a Boolean formalism. Then, φ belongs to $\mathsf{S\exists fd}_{>} \cap \overline{\mathsf{A\exists fd}_{>}} \cap \mathsf{A\forall fd}_{>}$ if and only if φ is AFR and is neither a tautology nor a contradiction.*

Proof Let φ a property in $\mathsf{S\exists fd}_{>} \cap \overline{\mathsf{A\exists fd}_{>}} \cap \mathsf{A\forall fd}_{>}$. The property is not a tautology or a contradiction because it would belong to region 6, as proved by Proposition 8. When the verdict is *true* for a trace, the (empty) set of values greater than *true* is finitely dismissible. When the verdict is *false* for a trace, since φ belongs to $\mathsf{A\forall fd}_{>}$, the value $true > false$ is fd, and thus we can always accept the verdict *false* (i.e. refute the property) with a finite prefix, which means that φ is AFR.

Let φ be an AFR property, but not a tautology or a contradiction. When the verdict is *false*, the set $\{true\}$ is fd because φ is AFR. When the verdict is *true*, the (empty) set of values greater than the verdict is fd. Thus, the property is in $\mathsf{A\forall fd}_{>}$. Since there are traces for which the verdict is *false* (because φ is not a tautology), there are some traces where there is a value greater than the verdict (i.e. *true*) which is fd (because φ is AFR), meaning that the property is in $\mathsf{S\exists fd}_{>}$. However, since there are traces for which the verdict is *true* (because φ is not a contradiction), there are some traces where there is not a value greater than the verdict (i.e. *true*) which is fd, meaning that the property is in $\mathsf{S\forall\overline{fd}}_{>} = \overline{\mathsf{A\exists fd}_{>}}$. □

We now prove that region 9 contains all the properties that are SFR.

Proposition 12 *Let φ be an property in a Boolean formalism. Then, φ belongs to $\mathsf{S\exists fd}_{>} \cap \mathsf{S\forall fd}_{>} \cap \overline{\mathsf{A\exists fd}_{>}} \cap \overline{\mathsf{A\forall fd}_{>}}$ if and only if φ is SFR.*

Proof In the original definitions, SFR is defined as a property that is not AFR or NFR. We have shown that regions 1 to 5 contain no properties. We have also shown that AFR and NFR properties belong to regions 6, 7 and 8, including tautologies and contradictions. Region 9 contains all the properties that are not AFR or NFR, and only those. □

A similar analysis can be made with respect to the finite dismissibility of the values lower than the verdict in Boolean formalisms. In this case, regions 1 to 5 contain no properties; region number 6 contains all the contradictions and tautologies; region number 7 contains the properties that are NFS, but are not contradictions; region number 8 contains the properties that are AFS, but are not tautologies or contradictions; and region number 9 contains the properties that are SFS. The proofs are similar to their counterparts for values greater than the verdict.

LTL.

Since LTL is a Boolean formalism, the regions 1 to 5 are empty for LTL, and tautologies and contradictions belong to region 6 in both cases, leaving us with only three sensible regions (7, 8 and 9). The overlapping of these regions for values lower and greater than the verdict form the original diagrams of [1] by Havelund and Peled and also the diagrams of Figs. 1 and 2 from Sections 2 and 4.

QLTL.

Just like in Section 4, the classification of QLTL properties coincides with that of LTL if we assume that the values of the atomic propositions are always 0 or 1. Without this assumption, some properties belong to other classes. Consider for example the LTL property $\Box p \wedge \Box\neg p$. This property is a contradiction in LTL and belongs to region 6. However, in QLTL if p is not assumed to be always 0 or 1, the property belongs to region 8 (considering the finite dismissibility of values greater than the verdict).

Proposition 13 *The QLTL property $\Box p \wedge \Box\neg p$ belongs to $\mathsf{S\exists fd}_{>} \cap \overline{\mathsf{A\exists fd}_{>}} \cap \mathsf{A\forall fd}_{>}$.*

Proof The range of values $(0.5, 1]$ can be statically dismissed. For the behavior for which p is always 0.5, the verdict is 0.5 and there are no values greater than the verdict that are fd. Thus, φ belongs to $\mathsf{S\forall\overline{fd}}_{>} = \overline{\mathsf{A\exists fd}_{>}}$.

Let v be the verdict of φ over a trace, and let $d > v$. At some point, either p or $1 - p$ has to become lower than d. So, d is fd, and φ is in $\mathsf{A\forall fd}_{>}$.

Consider a trace where p is 0 at the first instant. The verdict of the property is 0 and every other value is fd. In particular, value 1 is finitely dismissible in such trace. Thus, φ belongs to $\mathsf{S\exists fd}_{>}$. Note that this also implies that the property is not a tautology or a contradiction. □

Other assumptions make QLTL properties populate regions that are originally empty for LTL. We say that a proposition p is "fairly lower" than a value d in a given behavior if the supremum of the values of p is strictly less than d. This assumption allows dismissing some values finitely and statically.

Proposition 14 *Assuming that p is fairly lower than 1, the property $\Diamond p$ belongs to region 1 ($\overline{\mathsf{S\exists fd}_>} \cap \overline{\mathsf{S\forall fd}_>}$).*

Proof We can statically dismiss the value 1 as the verdict because the supremum of p is strictly less than 1. Let v be the verdict of φ for a certain trace and let d be a value in $(v, 1)$. We cannot finitely dismiss d as the verdict because for any given prefix, we can create a continuation where the next value of p is d, and the verdict would be d. Thus, φ belongs to $\mathsf{A\forall}\overline{\mathsf{fd}}_> = \overline{\mathsf{S\exists fd}_>}$.

On the other hand, we can always find such value because the verdict is strictly lower than 1. Thus, φ belongs to $\mathsf{A\exists}\overline{\mathsf{fd}}_> = \overline{\mathsf{S\forall fd}_>}$. □

Proposition 15 *Assuming that p is fairly lower than the first value of q, the property $\varphi = \Diamond p \wedge q$ belongs to region 2 ($\mathsf{S\exists fd}_> \cap \overline{\mathsf{S\forall fd}_>} \cap \overline{\mathsf{A\exists fd}_>}$).*

Proof Let d be the value of q at the first instant. Again, we can statically dismiss the value 1 as the verdict because d is at most 1, and p is fairly lower than d.

If $d < 1$, we can dismiss the values in $(d, 1)$ finitely. Thus, φ belongs to $\mathsf{S\exists fd}_>$.

If $d = 1$, the property behaves as $\Diamond p$ and all the values greater than the verdict are $\overline{\mathsf{fd}}$. Thus, φ is in $\mathsf{S\forall}\overline{\mathsf{fd}}_> = \overline{\mathsf{A\exists fd}_>}$.

Finally, since the verdict is strictly lower than 1, there are always values between the verdict and 1 which are $\overline{\mathsf{fd}}$, making the property belong to $\mathsf{A\exists}\overline{\mathsf{fd}} = \overline{\mathsf{S\forall fd}_>}$. □

Proposition 16 *Assuming that p is fairly lower than the first value of q, and that the first value of q is lower than 1, the property $\varphi = \Diamond p \wedge q$ belongs to region 3 ($\overline{\mathsf{S\forall fd}_>} \cap \mathsf{A\exists fd}_>$).*

Proof Let d be the value of q at the first instant. Since $d < 1$, we can statically dismiss 1 as a potential verdict.

We can dismiss the values in $(d, 1)$ finitely. Since $d < 1$, the set $(d, 1)$ is not empty, and φ belongs to $\mathsf{A\exists fd}_>$.

Also, since the verdict v is lower than d, the set (v, d) is not empty and φ belongs to $\mathsf{A\exists}\overline{\mathsf{fd}}_> = \overline{\mathsf{S\forall fd}_>}$. □

Proposition 17 *Assuming that the first value of q is lower than 1, the property $\varphi = \Diamond p \wedge q$ belongs to region 4 ($\mathsf{S\forall fd}_> \cap \mathsf{A\exists fd}_> \cap \overline{\mathsf{A\forall fd}_>}$).*

Proof Let d be the value of q at the first instant. Since $d < 1$, we can statically dismiss 1 as a potential verdict. We can dismiss the values in $(d, 1)$ finitely. Since $d < 1$, the set $(d, 1)$ is not empty, and φ belongs to $\mathsf{A\exists fd}_>$. If $d = 0$, the verdict is 0 and all the values greater than it are fd. Thus, the property belongs to $\mathsf{S\forall fd}_>$.

If the verdict v is lower than d, the values in (v, d) are $\overline{\mathsf{fd}}$, making φ belong to $\mathsf{S\exists}\overline{\mathsf{fd}}_> = \overline{\mathsf{A\forall fd}_>}$. □

Proposition 18 *Assuming that the first value of q is lower than 1, the property q belongs to region 5 ($\mathsf{A\exists fd}_> \cap \mathsf{A\forall fd}_>$).*

Proof Let d be the value of q at the first instant. Since $d < 1$, we can statically dismiss 1 as a potential verdict. We can always dismiss the values in the non-empty set $(d, 1)$ finitely. Thus, the property is in $\mathsf{A\exists fd}_>$ and also in $\mathsf{A\forall fd}_>$. □

Regions 1 and 2 are empty when we analyze the finite dismissibility of values greater (resp. lower) than the verdict and the totally ordered set of potential verdicts contains a maximum (resp. minimum) value. This is the case for LTL (where the maximum value is *true* and the minimum value is *false*), QLTL, and $\mathrm{Disc}^{\mathrm{LTL}}[\mathcal{D}]$ (where the maximum value is 1 and the minimum value is 0). The proof follows from the following proposition (for the dismissibility analysis of values greater than the verdict).

Proposition 19 *If the set of potential verdicts D contains a maximum value $\top$, then $\overline{\mathsf{S\forall fd}_>} \subseteq \mathsf{A\exists fd}_>$.*

Proof Let φ be a property in $\overline{\mathsf{S}\forall\mathsf{fd}_{>}} = \mathsf{A}\exists\overline{\mathsf{fd}}_{>}$. This means that for every trace there is a value greater than the verdict which is $\overline{\mathsf{fd}}$. If the verdict is $\top$ for φ for any trace, then for that trace we will not be able to find a value greater than the verdict at all, which means that the verdict can never be $\top$ for φ for any trace, and we can dismiss it right away. The value $\top$ is a value greater than the verdict that can always be finitely dismissed, and thus φ belongs to $\mathsf{A}\exists\mathsf{fd}_{>}$. □

The previous propositions show how assumptions can affect the classification of properties. Moreover, using these assumptions we have populated the nine regions of Fig. 3 with QLTL properties.

As stated in Section 4, all the properties of $\mathrm{Disc}^{\mathrm{LTL}}[\mathcal{D}]$ that do not use the operator $\mathcal{U}_\eta$ belong to the same classes as their LTL counterpart; meaning that some $\mathrm{Disc}^{\mathrm{LTL}}[\mathcal{D}]$ properties populate the regions 6 to 9 of the general landscape shown in Fig. 3 considering the finite dismissibility of values greater or lower than the verdict.

6 Conclusion

In this paper we have presented a generalization of the classification of Havelund and Peled [1] to expressive verdicts. We have introduced general definitions for admissibility and dismissibility of verdicts and instantianted these to totally ordered domains. Then we have illustrated the taxonomy to two quantitative logics: quantitative LTL and discounting LTL. Finally, we have introduced a generalization of the framework in [38] further breaking up one of the original classes, which introduces duality and limits all monitorability candidates to 9 possible cases. For LTL, only a few of these cases are populated (apart from contradictions and tautologies), which leads to the known taxonomy of Havelund-Peled. With the use of assumptions, we managed to populate all the 9 categories for QLTL.

Many questions remain open, including (surprisingly) whether—without assumptions—the empty categories in LTL remain empty in QLTL. Another important problem is to study how assumptions can affect precisely the monitorability of a property. It is possible that a property φ that belongs to a class without assumptions, belongs to a different class with an assumption. As a trivial example, if one assumes φ itself, then φ becomes a tautology.

We also want to study the impact of composing properties from different regions using operators from the logic, and how adding or removing assumptions affect the classification of a property. Future work also includes studying other quantitative logics like *Counting LTL* [40], where the semantics distinguish the steps necessary until satisfactions, and *Robust LTL* [41]. We also plan to extend our framework to general verdicts for other classes of domains beyond totally ordered domains.

References

[1] Havelund, K., Peled, D.: Runtime verification: From propositional to first-order temporal logic. In: Proc. of the 18th Int'l Conf. on Runtime Verification (RV'18). LNCS, vol. 11237, pp. 90–112. Springer, Berlin, Heidelberg (2018)

[2] Emerson, E.A., Clarke, E.M.: Characterizing correctness properties of parallel programs using fixpoints. In: Proc. of the 7th Colloquium on Automata, Languages and Programming (ICALP'80). LNCS, vol. 85, pp. 169–181. Springer, Berlin, Heidelberg (1980)

[3] Queille, J.-P., Sifakis, J.: Specification and verification of concurrent systems in CESAR. In: Symposium on Programming. LNCS, vol. 137, pp. 337–351. Springer, Berlin, Heidelberg (1982)

[4] Havelund, K., Goldberg, A.: Verify your runs. In: Proc. of the First IFIP TC 2/WG 2.3 Conference on Verified Software: Theories, Tools, Experiments (VSTTE'05). LNCS, vol. 4171, pp. 374–383. Springer, Berlin, Heidelberg (2005)

[5] Leucker, M., Schallhart, C.: A brief account of runtime verification. Journal of Logic and Algebraic Programming **78**(5), 293–303 (2009)

[6] Bartocci, E., Falcone, Y. (eds.): Lectures on Runtime Verification - Introductory and Advanced Topics. LNCS, vol. 10457. Springer, Berlin, Heidelberg (2018)

[7] Havelund, K., Roşu, G.: Synthesizing monitors for safety properties. In: Proc. of the 8th Int'l Conf on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'02). LNCS, vol. 2280, pp. 342–356. Springer, Berlin, Heidelberg (2002)

[8] Eisner, C., Fisman, D., Havlicek, J., Lustig, Y., McIsaac, A., Campenhout, D.V.: Reasoning with temporal logic on truncated paths. In: Proc. of the 15th Int'l Conf on Computer Aided Verification (CAV'03). LNCS, vol. 2725, pp. 27–39. Springer, Berlin, Heidelberg (2003)

[9] Bauer, A., Leucker, M., Schallhart, C.: Runtime verification for LTL and TLTL. ACM Transactions on Software Engingeering and Methodology **20**(4), 14 (2011)

[10] Sen, K., Roşu, G.: Generating optimal monitors for extended regular expressions. ENTCS **89**(2), 226–245 (2003)

[11] Asarin, E., Caspi, P., Maler, O.: Timed regular expressions. Journal of the ACM **49**(2), 172–206 (2002)

[12] Barringer, H., Goldberg, A., Havelund, K., Sen, K.: Rule-based runtime verification. In: Proc. of the 5th Int'l Conf on Verification, Model Checking and Abstract Interpretation (VMCAI'04). LNCS, vol. 2937, pp. 44–57. Springer, Berlin, Heidelberg (2004)

[13] Roşu, G., Havelund, K.: Rewriting-based techniques for runtime verification. Automated Software Engineering **12**(2), 151–197 (2005)

[14] 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: Proc. of the 12th Int'l Symp. of Temporal Representation and Reasoning (TIME'05), pp. 166–174. IEEE CS Press, Burlington, VT, USA (2005)

[15] De Giacomo, G., Vardi, M.Y.: Linear temporal logic and linear dynamic logic on finite traces. In: Proc. of the 23rd Int'l Joint Conf. on Artificial Intelligence (IJCAI'14), pp. 854–860. AAAI Press, Palo Alto, California (2013)

[16] Reinbacher, T., Rozier, K.Y., Schumann, J.: Temporal-logic based runtime observer pairs for system health management of real-time systems. In: Proc. of the 20th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'14). LNCS, vol. 8413, pp. 357–372. Springer, Berlin, Heidelberg (2014)

[17] Bauer, A., Leucker, M., Schallhart, C.: The good, the bad, and the ugly—but how ugly is ugly? In: Proc. of the 7th Int'l Workshop on Runtime Verification (RV'07). LNCS, vol. 4839, pp. 126–138. Springer, Berlin, Heidelberg (2007)

[18] Stucki, S., Sánchez, C., Schneider, G., Bonakdarpour, B.: Gray-box monitoring of hyperproperties. In: ter Beek, M.H., McIver, A., Oliveira, J.N. (eds.) Formal Methods - The Next 30 Years - Third World Congress, FM 2019. Lecture Notes in Computer Science, vol. 11800, pp. 406–424. Springer, Berlin, Heidelberg (2019). https://doi.org/10.1007/978-3-030-30942-8_25

[19] Stucki, S., Sánchez, C., Schneider, G., Bonarkdarpour, B.: Gray-box monitoring of hyperproperties with an application to privacy. Formal Methods in Systems Desing, 1–36 (2020). https://doi.org/10.1007/s10703-020-00358-w

[20] Pnueli, A., Zaks, A.: PSL model checking and run-time verification via testers. In: Proc. of the 14th Int'l Symp. on Formal Methods (FM'06). LNCS, vol. 4085, pp. 573–586. Springer, Berlin, Heidelberg (2006)

[21] Basin, D.A., Klaedtke, F., Müller, S., Zalinescu, E.: Monitoring metric first-order temporal properties. Journal of the ACM **62**(2) (2015)

[22] Colombo, C., Pace, G.J., Schneider, G.: Dynamic event-based runtime monitoring of real-time and contextual properties. In:

Proc. of the 13th Int'l Workshop on Formal Methods for Industrial Critical Systems (FMICS'08). LNCS, vol. 5596, pp. 135–149. Springer, Berlin, Heidelberg (2008)

[23] Faella, M., Legay, A., Stoelinga, M.: Model checking quantitative linear time logic. Electronic Notes in Theoretical Computer Science **220**(3), 61–77 (2008). https://doi.org/10.1016/j.entcs.2008.11.019. Proc. of the Sixth Workshop on Quantitative Aspects of Programming Languages (QAPL 2008)

[24] Sánchez, C.: Online and offline stream runtime verification of synchronous systems. In: Proc. of the 18th Int'l Conf. on Runtime Verification (RV'18). LNCS, vol. 11237, pp. 138–163. Springer, Berlin, Heidelberg (2018). https://doi.org/10.1007/978-3-030-03769-7_9

[25] Faymonville, P., Finkbeiner, B., Schirmer, S., Torfah, H.: A stream-based specification language for network monitoring. In: Proc. of the 16th Int'l Conf. on Runtime Verification (RV'16). LNCS, vol. 10012, pp. 152–168. Springer, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-319-46982-9_10

[26] Gorostiaga, F., Sánchez, C.: Striver: Stream runtime verification for real-time event-streams. In: Proc. of the 18th Int'l Conf. on Runtime Verification (RV'18). LNCS, vol. 11237, pp. 282–298. Springer, Berlin, Heidelberg (2018)

[27] Convent, L., Hungerecker, S., Leucker, M., Scheffel, T., Schmitz, M., Thoma, D.: TeSSLa: Temporal stream-based specification language. In: Proc. of the 21st. Brazilian Symp. on Formal Methods (SBMF'18). LNCS, vol. 11254. Springer, Berlin, Heidelberg (2018)

[28] Henzinger, T.A., Saraç, N.E.: Quantitative and approximate monitoring. In: Proceedings of the 36th Annual ACM/IEEE Symposium on Logic in Computer Science. LICS '21. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1109/LICS52264.2021.9470547. https://doi.org/10.1109/LICS52264.2021.9470547

[29] Li, Y., Droste, M., Lei, L.: Model checking of linear-time properties in multi-valued systems. Information Sciences **377**, 51–74 (2017). https://doi.org/10.1016/j.ins.2016.10.030

[30] Henzinger, T.A., Mazzocchi, N., , E.N.S.: Quantitative safety and liveness. In: Proc. of the 26th Int'l Conf. on Foundations of Software Science and Computation Structures (FoSSaCS'23). LNCS, vol. 13992, pp. 349–370. Springer, Cham (2023)

[31] Alpern, B., Schneider, F.B.: Defining liveness. Information Processing Letters **21**(4), 181–185 (1985). https://doi.org/10.1016/0020-0190(85)90056-0

[32] Zhang, X., Leucker, M., Dong, W.: Runtime verification with predictive semantics. In: Proc. of the 4th Int'l Symp NASA Formal Methods (NFM'12). LNCS, pp. 418–432. Springer, Berlin, Heidelberg (2012)

[33] Henzinger, T.A., Saraç, N.E.: Monitorability under assumptions. In: Proc. of the 20th Int'l Conf. on Runtime Verification (RV'20). LNCS, vol. 12399, pp. 3–18. Springer, Berlin, Heidelberg (2020)

[34] Leucker, M., Sánchez, C., Scheffel, T., Schmitz, M., Thoma, D.: Runtime verification for timed event streams with partial information. In: Proc. of the 19th Int'l Conf. on Runtime Verification (RV'19). LNCS, vol. 11757, pp. 273–291. Springer, Berlin, Heidelberg (2019)

[35] Kauffman, S., Havelund, K., Fischmeister, S.: What can we monitor over unreliable channels? International Journal on Software Tools for Technology Transfer, 1–24 (2020)

[36] Chang, E., Manna, Z., Pnueli, A.: Characterization of temporal property classes. In: Kuich, W. (ed.) Automata, Languages and Programming, pp. 474–486. Springer, Berlin, Heidelberg (1992)

[37] Kallwies, H., Leucker, M., Sánchez, C., Scheffel, T.: Anticipatory recurrent monitoring with uncertainty and assumptions.

In: Proc. of the 22nd Int'l Conference on Runtime Verification (RV'22). LNCS, vol. 13498, pp. 181–199. Springer, Berlin, Heidelberg (2022). https://doi.org/10.1007/978-3-031-17196-3_10. https://doi.org/10.1007/978-3-031-17196-3_10

[38] Gorostiaga, F., Sánchez, C.: Monitorability of expressive verdicts. In: Proc. of the 14th Int'l Symp. on NASA Formal Methods (NFM'22). LNCS, vol. 13260, pp. 693–712. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-06773-0_37. https://doi.org/10.1007/978-3-031-06773-0_37

[39] Almagor, S., Boker, U., Kupferman, O.: Discounting in LTL. In: Proc. of the 20th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'14). LNCS, vol. 8413, pp. 424–439. Springer, Berlin, Heidelberg (2014)

[40] Laroussinie, F., Meyer, A., Petonnet, E.: Counting LTL. In: Proc. of the 2010 17th Int'l Symp. on Temporal Representation and Reasoning (TIME'10), pp. 51–58. IEEE, Burlington, VT, USA (2010). https://doi.org/10.1109/TIME.2010.20

[41] Tabuada, P., Neider, D.: Robust linear temporal logic. In: Proc. of the 25th EACSL Annual Conference on Computer Science Logic (CSL'16). LIPIcs, vol. 62, pp. 10–11021. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Marseille, France (2016)