Monitoring Parametric Temporal Logic

Peter Faymonville[1], Bernd Finkbeiner[1], and Doron Peled[2]

[1] Fachrichtung Informatik
Universität des Saarlandes, Germany

[2] Department of Computer Science
Bar Ilan University, Israel

Abstract. Runtime verification techniques allow us to monitor an execution and check whether it satisfies some given property. Efficiency in runtime verification is of critical importance, because the evaluation is performed while new events are monitored. We apply runtime verification to obtain *quantitative* information about the execution, based on linear-time temporal properties: the temporal specification is extended to include parameters that are instantiated according to a measure obtained at runtime. The measure is updated in order to maintain the best values of parameters, according to their either maximizing or minimizing behavior, and priority. We provide measuring algorithms for linear-time temporal logic with parameters (PLTL). Our key result is that achieving efficient runtime verification is dependent on the determinization of the measuring semantics of PLTL. For *deterministic PLTL*, where all disjunctions are guarded by atomic propositions, online measuring requires only linear space in the size of the specification and logarithmic space in the length of the trace. For *unambiguous PLTL*, where general disjunctions are allowed, but the measuring is deterministic in the truth values of the non-parametric subformulas, the required space is exponential in the size of the specification, but still logarithmic in the length of the trace. For full PLTL, we show that online measuring is inherently hard and instead provide an efficient offline algorithm.

1 Introduction

While verifying the complete behavior of a system (e.g., using model checking) is certainly desirable, it is not always possible, as its internal structure is not always given, or its state space is prohibitively large. Runtime verification analyzes the ongoing execution of a system against a given specification, written for example in Linear Temporal Logic (LTL), based on monitoring its externally measurable events. The challenge in runtime verification is to provide an efficient algorithm that can perform its required *updates* between any two successive monitored events.

We study here the runtime monitoring of a system with respect to temporal properties, expressed using LTL, where (discrete time) duration counters are added to the subformulas. The runtime verification reports not only about the conformance between the currently monitored sequence and the specification, but also provides numerical values that bound the duration of the scope of subformulas on the checked execution from either above or below. For example, the specification $\square(r \rightarrow \diamondsuit_{\leq x} g)$ computes the maximal response time x between a request r and a response g. We call this approach for runtime verification "runtime measuring" (as "model measuring" [2] is related to

"model checking"). While runtime verification can check and alarm against unwanted situations, runtime measuring collects statistics on the system behavior.

Our monitoring approach is *declarative* as opposed to *operational*. In an operational approach [7, 8], the measures are collected by explicitly specifying the initialization and update of counters that calculate the reported values. In a declarative approach, we attach parameters to temporal formulas to specify the measure we are interested in, and the monitoring algorithm takes care of finding parameter valuations such that the formula is satisfied. Suppose, for example, that we do not want to measure all response times, but are interested only in the *last* request that was successfully answered. The formula $\Box(\bigcirc\Diamond(r \wedge \Diamond g) \vee (r \rightarrow \Diamond_{\leq x} g))$ uses the disjunction to filter out all requests before the last successful request: in every step, if the left disjunct holds, then the response time is irrelevant, because the disjunction is true for any value of x; the right disjunct thus only becomes relevant when the left disjunct is false, i.e., when there is no future request that is successfully answered.

Research on the runtime verification of LTL [6, 8, 10] distinguishes *online* algorithms that need to keep only some bounded amount of information about the trace seen so far, from *offline* algorithms that store the entire sequence for later evaluation. For runtime measuring, we also look for an online algorithm with reasonable space consumption. Since measuring necessarily entails updating counters, an algorithm that requires logarithmic space in the length of the trace is acceptable, while an algorithm that requires linear space in the length of the trace, such as an offline algorithm, is in general not practical. We show that the complexity of runtime measuring depends on the disjunctive characterization of the LTL specification; that is, when the formula contains a disjunction, or some subformula that can be satisfied in different ways (e.g., $\varphi \,\mathcal{U}\, \psi$ may be satisfied when the first ψ holds, but also when φ continues to hold until some subsequent occurrences of ψ). Due to this disjunction characteristics, counting results are not uniquely defined. Consider a sequence of length n where a, b and c happen in each event. The formula $(a \,\mathcal{U}_{\leq x}\, (b \,\mathcal{U}_{\leq y}\, c))$ can obtain any natural number for the parameters x and y such that $x + y = n$. One way to obtain unique measures is to impose a priority order among the parameters and seek for the best (minimal) results according to the lexicographic ordering. We show that for this case, an online measuring algorithm with logarithmic memory in the length of the trace is impossible.

An alternative approach to obtain unique measures is to modify the logic. We present two variations of PLTL that not only provide unique measures, but also have efficient online measuring algorithms. *Deterministic PLTL* allows, like deterministic LTL [11], only guarded disjunctions of the form $((p \wedge \psi_1) \vee (\neg p \wedge \psi_2))$, where p is an atomic proposition. Runtime measuring of deterministic PLTL indeed requires only logarithmic memory. The drawback of deterministic PLTL is, however, its limited expressiveness: deterministic LTL can only express properties in the intersection of LTL and ACTL [11]. To eliminate this drawback, we introduce *unambiguous PLTL*, which maintains the full expressiveness of LTL. Instead of syntactically restricting the possible disjunctions, unambiguous PLTL only *disambiguates* the PLTL semantics with respect to the measuring. For example, in a disjunction $(\psi_1 \vee \psi_2)$, we *only* measure ψ_2 if ψ_1 is is false under all possible parameter values.

Consider, for example, again the problem of measuring the maximal response time x between a request r and a response g. To express this measuring problem in deterministic PLTL, we use a disjunction guarded by r: $\square(\neg r \vee (r \wedge \diamondsuit_{\leq x} g))$. This encoding relies on the fact that the condition that triggers the measuring, r, is an atomic proposition. If we modify the problem, as discussed earlier, to only measure the last successful request, then this is no longer possible, because the decision whether or not to measure the current request depends on the success of *future* requests. The modified measuring problem can thus no longer be expressed in deterministic PLTL. The PLTL specification $\square(\bigcirc\diamondsuit(r \wedge \diamondsuit g) \vee (r \rightarrow \diamondsuit_{\leq x} g))$, discussed above, does, however, work for unambiguous PLTL. In unambiguous PLTL, the (unguarded) disjunction is allowed, and the right disjunct is evaluated whenever the left disjunct is false. Hence, the specification computes precisely the response time of the last successful request.

We obtain the following results: for deterministic PLTL, online measuring requires only logarithmic space in the length of the trace and linear space in the size of the specification. For unambiguous PLTL, the required space is exponential in the size of the specification, but still logarithmic in the length of the trace. For full PLTL, we provide an efficient offline algorithm, which can also be used as an online algorithm by keeping the trace seen so far in storage. This algorithm requires quasilinear space in the length of the trace. We also show that, in fact, no online measuring algorithm with logarithmic space in the length of the trace exists. Unambiguous PLTL thus appears to be the sweet spot in the trade-off between expressiveness and complexity.

Related Work. The synthesis of monitors for LTL is a well-studied problem, see [3, 5, 6, 8, 10, 13]. The offline backwards runtime algorithm of Havelund and Rosu [6] calculates with each event the truth values of the subformulas according to subformula order. Thus, with each new monitored event, the calculation would be *linearly* related to both the length of the sequence so far and the size of the checked formula. The *testers* construction by Pnueli and Zaks [13] can be used to backwards assign values to variables representing subformulas in a compositional manner. In previous work of the second author together with Sankaranarayanan and Sipma [8, 7], alternating automata are used to obtain efficient algorithms for runtime verification. The query language considered there extends LTL with functions that are executed along the trace in order to collect measures and more complicated statistics such as the average number of packet transmissions in a communication protocol. Unlike the declarative approach of this paper, the collection of measures and statistics is specified operationally, not in the form of parameters. A different type of parametric monitoring has been studied by Rosu and Chen [14]: They consider traces that contain events with parameter bindings. Such traces can be considered as several different traces merged together and the challenge for the monitoring algorithm lies in the efficient slicing of the trace.

2 Parametric Temporal Logic

Syntax. Parametric Temporal Logic (PLTL) [2] is an extension of linear-time temporal logic (LTL) [12] with parameterized operators, which measure the duration from the introduction of a temporal goal until it is satisfied. E.g., for a subformula of the form

$\varphi \,\mathcal{U}\, \psi$, we expect to measure the duration until ψ happens. We assign a parameter x together with a comparison operator to the subformula, e.g., $\varphi \,\mathcal{U}_{\leq x}\, \psi$, with the intended meaning that ψ should hold within *at most* x steps while φ holds. In contrast to LTL variants like Prompt-LTL [9], PLTL allows multiple parameters within a formula.

The *syntax* of PLTL is given, for a set of atomic propositions AP, with typical element p, and a set of parameter variables V as follows:

$$\psi ::= \mathit{true} \mid p \mid \neg\psi \mid (\psi \wedge \psi) \mid (\psi \vee \psi) \mid \bigcirc\psi \mid \Diamond\psi \mid \Box\psi \mid (\psi \,\mathcal{U}\, \psi) \mid (\psi \,\mathcal{R}\, \psi) \mid \Diamond_{\leq x}\psi \mid \Box_{\leq x}\psi$$

The operators $\mathcal{U}$-*until*, $\Diamond$-*eventually*, $\Box$-*always*, $\mathcal{R}$-release, $\bigcirc$-*next* are the usual temporal operators from LTL. In the two new parametric operators $\Diamond_{\leq x}\psi$ and $\Box_{\leq x}\psi$, x may be either a constant natural number or a parameter variable.

In addition to disjunction, conjunction, and negation, we also allow the usual derived Boolean connectives such as implication $\rightarrow$. In addition to $\Diamond_{\leq x}\psi$ and $\Box_{\leq x}\psi$, we also use the following derived parametric operators

$$\Diamond_{>x}, \Box_{>x}, \mathcal{U}_{\leq x}, \mathcal{U}_{>x}, \mathcal{R}_{\leq x}, \text{ and } \mathcal{R}_{>x},$$

where x is again a parameter or a natural number. We assume that each parameter variable occurs at most once. Let $\alpha : X \mapsto \mathbb{N} \cup \{\infty\}$ denote a value assignment for the parameters. Then $\alpha(x)$ is the integer value assigned to x by α. For simplicity, we set $\alpha(k) = k$ for $k \in \mathbb{N}$. We denote by $\alpha[k/x]$ the valuation that maps y to k if $y = x$ and to $\alpha(y)$ if $y \neq x$. Let $\alpha \setminus x$ be the valuation α, excluding the parameter x.

Semantics. In the following, we adapt the PLTL semantics to the finite traces observed during monitoring. We interpret a given PLTL formula ψ over a finite trace σ of events, numbered with nonnegative integers, where each event provides an interpretation to the Boolean propositions AP, i.e., $\sigma : \{0 \ldots |\sigma| - 1\} \rightarrow 2^{AP}$. Let the kth element of σ (starting with $k = 0$) be denoted by $\sigma[k]$.

We denote by $(\sigma, k, \alpha) \models \psi$, the fact that trace σ satisfies the formula ψ at position k with valuation α. The *satisfaction relation* $\models$ is defined recursively as follows.
For atomic propositions and Boolean connectives:

- $(\sigma, k, \alpha) \models p$ iff $p \in \sigma[k]$; $(\sigma, k, \alpha) \models \neg\psi$ iff $(\sigma, k, \alpha) \not\models \psi$
- $(\sigma, k, \alpha) \models (\psi_1 \wedge \psi_2)$ iff $(\sigma, k, \alpha) \models \psi_1$ and $(\sigma, k, \alpha) \models \psi_2$.
- $(\sigma, k, \alpha) \models (\psi_1 \vee \psi_2)$ if $(\sigma, k, \alpha) \models \psi_1$ or $(\sigma, k, \alpha) \models \psi_2$.

For the LTL operators:

- $(\sigma, k, \alpha) \models \bigcirc\psi$ iff $|\sigma| > k+1$ and $(\sigma, k+1, \alpha) \models \psi$.
- $(\sigma, k, \alpha) \models (\psi_1 \,\mathcal{U}\, \psi_2)$ if there exists i, $k < i < |\sigma|$ where $(\sigma, i, \alpha) \models \psi_2$, and for each j, $k \leq j < i$, $(\sigma, j, \alpha) \models \psi_1$.
- $(\sigma, k, \alpha) \models (\psi_1 \,\mathcal{R}\, \psi_2)$ if for each $k \leq i < |\sigma|$, it holds that either $(\sigma, i, \alpha) \models \psi_2$ or there exists j, $k \leq j < i$ such that $(\sigma, k+j, \alpha) \models \psi_1$.

We use the following standard abbreviations: $\Diamond\varphi = (\mathit{true} \,\mathcal{U}\, \varphi)$, $\Box\varphi = (\mathit{false} \,\mathcal{R}\, \varphi)$. The parametric operators are defined as follows:

- $(\sigma, k, \alpha) \models \Diamond_{\leq x} \psi$ if there exists $0 \leq i \leq \alpha(x)$, where $k + i < |\sigma|$, such that $(\sigma, k + i, \alpha) \models \psi$;
- $(\sigma, k, \alpha) \models \Box_{\leq x} \psi$ if for all $0 \leq i \leq \alpha(x)$, where $k + i < |\sigma|$, $(\sigma, k + i, \alpha) \models \psi$;

We also write $(\sigma, \alpha) \models \varphi$ for $(\sigma, 0, \alpha) \models \varphi$. We can extend our syntax and semantic definitions to allow also constants in addition to (or instead of) the variable parameters: Constants are simply parameters that have the same values under each valuation.

The semantics of the derived parametric operators is given by the following equalities (see [2], Lemma 2.2):

- $\Diamond_{>x} \psi = \Box_{\leq x} \Diamond \bigcirc \psi$;
- $\Box_{>x} \psi = \Diamond_{\leq x} \Box \bigcirc \psi$;
- $\psi_1 \, \mathcal{U}_{\leq k} \, \psi_2 = ((\psi_1 \, \mathcal{U} \, \psi_2) \wedge \Diamond_{\leq k} \psi_2)$;
- $\psi_1 \, \mathcal{R}_{\leq k} \, \psi_2 = ((\psi_1 \, \mathcal{R} \, \psi_2) \vee \Box_{\leq k} \psi_2)$;
- $\psi_1 \, \mathcal{U}_{>k} \, \psi_2 = \Box_{\leq k} (\psi_1 \wedge \bigcirc (\psi_1 \, \mathcal{U} \, \psi_2))$;
- $\psi_1 \, \mathcal{R}_{>k} \, \psi_2 = \Diamond_{\leq k} (\psi_1 \vee \bigcirc (\psi_1 \, \mathcal{R} \, \psi_2))$.

Each of the parametric operators is either upward or downward closed. $\Diamond_{\leq x}$ is *upward closed*: if $\Diamond_{\leq x}$ for some value $\alpha(x) = a$, then $\Diamond_{\leq x}$ also holds for any $\alpha(x) = b$ with $b > a$. If an operator is upward closed, we want to *minimize* the value we report. However, as this operator can hold in multiple suffixes of the measured sequence, we need to report on a value that would guarantee all of them, hence the *maximum* among these minimal values. Likewise, $\Box_{\leq x}$ is *downward closed*: if $\Box_{\leq x}$ for some value $\alpha(x) = a$, then $\Box_{\leq x}$ also holds for any $\alpha(x) = a$ with $0 \leq b < a$. If an operator is downward closed, we want to *maximize* the value we report. However, as this operator can hold in multiple suffixes of the measured sequence, we need to report on a value that would guarantee all of them, hence the *minimum* among these maximal values.

We assume that the PLTL formulas are in negation normal form (i.e., negations may only occur in front of atomic propositions). Negation normal form can be established by pushing negations inward according to the usual rewrite rules for LTL, e.g., $\neg(\psi_1 \, \mathcal{U} \, \psi_2) = (\neg\psi_1 \, \mathcal{R} \, \neg\psi_2)$, and, additionally, the following equivalence for the parametric operators: $\neg\Box_{\leq x} \psi = \Diamond_{\leq x} \neg\psi$. This transformation increases the size of the formula only by a constant factor. We also assume that the parameterized operators are only $\Box_{\leq x}$ and $\Diamond_{\leq x}$. The transformation according to the equalities for the derived parametric operators can result in an exponential explosion in the size of the formula. However, one does not need to explicitly represent such a formula: one can introduce "formula variables" to name repeating subformulas and use them repeatedly. Indeed, in all algorithms in this paper, one does not pay for the repetition of subformulas resulting from the rewriting [4].

Unique Measures. An attractive feature of PLTL is that the logic permits more than one parameter in the same formula. As discussed in the introduction, this means, however, that a trace can satisfy the formula with multiple incomparable value assignments: for example, the formula $\Diamond_{\leq x} \Diamond_{\leq y} p$ is satisfied on a trace where p is false in the first position and true in the second position both for the value assignment $\alpha : x \mapsto 1, y \mapsto 0$ and for the value assignment $\alpha' : x \mapsto 0, y \mapsto 1$.

To avoid such ambiguities, we introduce a total (i.e., linear) *priority* order $\gg$ on the parameters in X. Let $\max(X)$ be the maximum element of X according to $\gg$. The priority order induces a total order $\sqsupset$ on value assignments where $\alpha_1 \sqsupset \alpha_2$ if, for $x = \max(X)$,

- $(\alpha_1(x) - \alpha_2(x)) > 0$ and we *maximize* x (i.e., the operator of x is downward closed) or
- $(\alpha_1(x) - \alpha_2(x)) < 0$ and we *minimize* x (i.e., the operator of x is upward closed) or
- $\alpha_1(x) = \alpha_2(x)$ and $\alpha_1 \setminus x \sqsupseteq \alpha_2 \setminus x$.

The *measure* of a PLTL formula φ over an infinite trace σ is the optimal (with respect to $\sqsupset$) value assignment α such that $(\sigma, 0, \alpha) \models \varphi$.

3 Offline Measuring

We present a first algorithm for measuring a given finite trace. We call the algorithm *offline*, because it requires access to the trace positions in reverse chronological order; this type of access is possible if the trace has been stored before its analysis. The algorithm is less appropriate for the online setting of monitoring, where the trace becomes available one position at a time. We will study online measuring in Sections 4 and 5.

Intuitively, we focus first on the parameter with highest priority, setting up the other variables to a default value of 0 for a maximizing parameter, and $|\sigma| + 1$ for a minimizing parameter. We perform a binary search on the value of this variable, hence are left with a formula with constant parameters. After finding the optimal (minimal or maximal) value for this parameter in this way, we fix it, and move to optimize the next highest priority parameter and so forth.

Checking Formulas with Constant Parameters. We begin with the simple case of PLTL formulas *without* parameters that may still contain parametric operators that refer to constants. In this case, we are only interested in the truth value, not in an actual measurement.

Let σ be a finite trace and let φ be a PLTL formula in normal form with constants instead of variable parameters. We check the satisfaction of φ in a backward traversal of σ. During the traversal, we maintain for every subformula ψ the truth value b_ψ, which indicates whether ψ is satisfied on the suffix from the currently considered position. For every subformula ψ that starts with a parametric operator (referring to some constant k) we additionally maintain a counter c_ψ, which indicates for $\psi = \Diamond_{\leq k}\mu$ the number of steps until μ is satisfied, and for $\psi = \Box_{\leq k}\mu$ the number of steps until μ is falsified, respectively; in case μ (respectively, $\neg\mu$) never become true, we set $c_\psi = \bot$.

Before we process the last event in the trace we set up the values as follows:

- $b_{\Diamond_{\leq k}\psi} = \mathit{false}$, $c_{\Diamond_{\leq k}\psi} = \bot$,
- $b_{\Box_{\leq k}\psi} = \mathit{true}$, $c_{\Box_{\leq k}\psi} = \bot$,
- $b_{\bigcirc\psi} = \mathit{false}$.
- $b_{(\psi_1 \mathcal{U} \psi_2)} = \mathit{false}$.
- $b_{(\psi_1 \mathcal{R} \psi_2)} = \mathit{true}$.

For the Boolean connectives and the non-parametric operators, the backward propagation proceeds as in a standard backward update algorithm for LTL (c.f., [6]). We

denote the values for the current level i with b_ψ and c_ψ and the values for the previously considered level $i+1$ with b'_ψ and c'_ψ. For Boolean combinations we evaluate bottom-up as follows: $b_{(\varphi\wedge\psi)} = (b_\varphi \wedge b_\psi)$ and $b_{(\varphi\vee\psi)} = (b_\varphi \vee b_\psi)$.

For the non-parametric temporal operators, we propagate the truth values as follows:

- $b_{\bigcirc\psi} = b'_\psi$
- $b_{\psi_1\vee\psi_2} = b_{\psi_1} \vee b_{\psi_2}$
- $b_{(\psi_1\,\mathcal{U}\psi_2)} = (b_{\psi_2} \vee (b_{\psi_1} \wedge b'_{\psi_1\,\mathcal{U}\psi_2}))$
- $b_{(\psi_1\,\mathcal{R}\psi_2)} = (b_{\psi_2} \wedge (b_{\psi_1} \vee b'_{\psi_1\,\mathcal{R}\psi_2}))$

For the parametric operators, we need to update the counters. In the following we assume $\bot + 1 = 0$ and $\bot \not\leq i$ for any $i \in \mathbb{N}$.

- $c_{\Diamond\leq k\psi} :=$ if b_ψ then 0
  else if $c'_{\Diamond\leq k\psi} < k$ then $c'_{\Diamond\leq k\psi} + 1$ else $\bot$.

- $c_{\Box\leq k\psi} :=$ if $\neg b_\psi$ then $\bot$
  else if $c'_{\Box\leq k\psi} < k$ then $c'_{\Box\leq k\psi} + 1$ else 0.

The truth values of the parametric operators can then be derived from the counter values: $b_{\Diamond\leq k\psi} = (c_{\Diamond\leq k\psi} \leq k)$; $b_{\Box\leq k\psi} = (c_{\Box\leq k\psi} \geq k)$.

In order to check φ on σ, we thus update two values for every subformula and trace position. The running time of our algorithm is therefore in $O(2^{|\varphi|} \times |\sigma|)$.

Measuring Formulas with at least one Parameter. In the case that φ contains a single parameter x, we know that the possible values of x are bounded by the length of the trace. We carry out a binary search to find the best value. The running time of the algorithm thus increases by a logarithmic factor in the length of the trace: $O(2^{|\varphi|} \times |\sigma| \times log|\sigma|)$. In case that φ contains $n > 1$ parameters, we focus on the parameter x with the highest priority by replacing all other parameters by their 'weakest' values, i.e., 0 if we maximize and $|\sigma| + 1$ if we minimize that parameter. This clearly does not affect the value of x in the measure. Once the value of x is obtained, we replace x with its value, and continue with the parameter with the next-highest priority.

The algorithm from the single-parameter case is therefore applied n times, where n is bounded by $|\varphi|$. We therefore obtain the following complexities.

Theorem 1. *Let φ be a PLTL formula and σ be a finite trace. With direct access to all trace positions, the measure of φ on σ can be computed in space $O(|\varphi| \times |\sigma| \times \log|\sigma|)$ and time $O(|\varphi| \times 2^{|\varphi|} \times |\sigma| \times \log|\sigma|)$.*

4 Online Measuring is Hard

The algorithms from the previous section assume direct access to the full trace. Since huge traces are common in practice for runtime verification, in fact, their size may not be a priori bounded, one would like to avoid storing the full trace, and instead only work with a logarithmic representation, such as the current value of a fixed number of counters. The following theorem shows that, unfortunately, such a monitoring algorithm cannot exist.

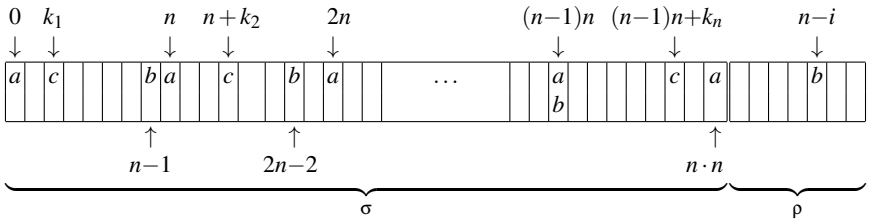

Fig. 1. Sequences σ and ρ in the proof of Theorem 2.

Theorem 2. *There is no online measuring algorithm for PLTL that uses only logarithmic space in the length of the trace.*

Proof. Suppose that such an online algorithm exists. We run the algorithm on the formula $\Box(a \rightarrow ((\Diamond_{\leq x} b) \vee (\Diamond_{\leq y} c)))$, where $x \gg y$. We will show that there is a sequence σ of length $O(n^2)$, such that the memory of the monitor must, after processing σ, contain all elements of an arbitrary chosen set $K = \{k_1, k_2, \ldots, k_n\}$ of n natural numbers.

Assume, without loss of generality, that $k_1 < k_2 < \ldots < k_n$. The sequence σ is constructed as follows. There is an a in the first position and then again after n steps, after $2n$ steps, and so on, for a total of $n+1$ times. In the first interval between two occurrences of a, there is a b after $n-1$ steps following the first a, and a c after k_1 steps following the first a, in the second interval, there is a b after $n-2$ steps following the first a and a c after k_2 steps, and so on. The construction of σ is illustrated on the left in Figure 1.

The monitor must keep the entire set K in memory after processing σ, because we can force the monitor to retrieve $k_i \in K$ for any $i = 1, \ldots n$, by extending σ with a suitable sequence ρ such that the y-measurement of $\sigma \cdot \rho$ is k_i. The extension, after the last a, consists of another $n-1$ steps with a b in the $(n-i)$th step and no further or c (or a). The construction is illustrated on the right in Figure 1. With only logarithmic memory, it is impossible to store K. Logarithmic space can only distinguish $O(n)$ cases; however, there can be 2^n different sets K.

The measurement of the higher priority variable x in $\sigma \cdot \rho$ cannot be smaller than $x = n - i + 1$, since the last a is followed by a b after that distance, but not with any c. For the purposes of measuring y, we only need to consider the first $i-1$ occurrences of a, because for all other occurrences, the left disjunct, with the chosen measure of x being at least $x = n - i + 1$, is always satisfied. In order not to increase the value of x beyond $n-i+1$, we need to satisfy the disjunction for each of the first $i-1$ occurrences of a (which follow by a b at a larger distance, namely, $n-i+2, n-i+3 \ldots n$) using its righthand side, i.e., through the first occurrence of a c after each a. In order to guarantee that all these distances from a to the first subsequent c are satisfying the formula with the measurement of y, we must choose y as the maximal value of them. As the distances appear in ascending order, we must have $y = k_{i-1}$. □

5 Online Measuring in Logarithmic Space

In this section, we present online measuring algorithms that only need logarithmic space in the length of the trace. Since we know from Theorem 2 that disjunctions between subformulas with parameters make this impossible for PLTL, we must look at syntactic or semantic variations of PLTL that "determinize" such disjunctions. We start with a syntactic fragment based on deterministic LTL [11]: in deterministic LTL, the only allowed disjunctions are of the form $((p \wedge \psi_1) \vee (\neg p \wedge \psi_2))$, where the subformulas ψ_1 and ψ_2 are guarded by a proposition p; since the value of p is immediately available, the choice of the disjunct is deterministic. Indeed, as we show in Section 5.1, runtime measuring of deterministic PLTL can be done with logarithmic cost in the length of the trace. Deterministic PLTL is, however, not completely satisfying as a logic for runtime measuring, because it is less expressive than full LTL: deterministic LTL can express exactly the properties in the intersection of LTL and ACTL [11].

We solve this problem in Section 5.2 by introducing *unambiguous* PLTL, which maintains the full expressiveness of LTL. Instead of syntactically restricting the possible disjunctions, we only *disambiguate* the PLTL semantics with respect to the measuring. For example, in a disjunction $(\psi_1 \vee \psi_2)$, we *only* measure ψ_2 if ψ_1 is false for all possible instantiations of the parameters. Again, the complexity of online measuring drops from linear to logarithmic in the length of the trace.

5.1 Deterministic PLTL

We define deterministic PLTL in analogy to deterministic LTL [11] by restricting the syntax of PLTL such that disjunctions and eventualities are always guarded by atomic propositions.

Syntax and Semantics. The *syntax* of PLTL^{det} is given, for a set of atomic propositions AP and a set of parameter variables V as follows:

$$\psi ::= \mathit{true} \mid p \mid \neg\psi \mid \psi \wedge \psi \mid (p \wedge \psi) \vee (\neg p \wedge \psi) \mid \bigcirc\psi \mid \Diamond p \mid (p \wedge \psi)\ \mathcal{U}\ (\neg p \wedge \psi) \mid \Box_{\leq x} p \mid \Diamond_{\leq x} p$$

PLTL^{det} is a sublogic of PLTL; the semantics remains the same.

Measuring Automata. We construct a monitor in the form of an extended finite-state automaton, which maintains the current measurements in a fixed number of integer variables. We begin with a formal definition of measuring automata.

A measuring automaton is a deterministic finite-state automaton extended with a set of variables, which are used to store data needed to compute the measure. The variables are initialized with either 0 or ∞. In each step, the automaton may update the integer variables with a reset to 0, an increment by 1, or by computing the minimum or maximum of two values. When the automaton reaches a final state it accepts the input word and outputs its measurement based on the state and the values of the integer variables.

Definition 1. *A* measuring automaton $(\Sigma, \Omega, Q, q_0, X, \theta, \delta, \gamma, F, \omega)$ *consists of an input alphabet* Σ*, an output domain* Ω*, a finite set of states* Q*, an initial state* q_0*, a finite set of variables* X*, an initial assignment* $\theta : X \to \{0, \infty\}$*, a transition function* $\delta : Q \times \Sigma \to (Q \cup \{\bot\})$*, an update function* $\gamma : Q \times \Sigma \to (X \to \mathbb{N}) \to (X \to \mathbb{N})$*, a set of final states* F*, and an output function* $\omega : F \times (X \to \mathbb{N}) \to \Omega$*. The update function* γ *is restricted to one of the following operations for each variable* $x \in X$*:* reset $x := 0$*, an* increment $x := y + 1$*, or with the* maximum *or* minimum *of two values:* $x := \min(x, y)$ *or* $x := \max(x, y)$*, where* y *is some other variable* $y \in X$*.*

A *run* of a measuring automaton $\mathcal{A} = (\Sigma, \Omega, Q, q_0, X, \theta, \delta, \gamma, F, \omega)$ on an input sequence $\sigma = \sigma_0 \sigma_1 \ldots \sigma_n \in \Sigma^*$ is a sequence $(s_0, \eta_0)(s_1, \eta_1) \ldots (s_n, \eta_n)$ of configurations, where each configuration is a pair (s_i, η_i) of a state s_i and a valuation $\eta_i : X \to \mathbb{N}$ of the integer variables, such that

- $s_0 = q_0$
- $\eta_0 = \theta$
- $s_{i+1} = \delta(s_i, \sigma_i)$ for $i = 0 \ldots (n-1)$
- $\eta_{i+1} = \gamma(s_i, \sigma_i)(\eta_i)$ for $i = 0 \ldots (n-1)$
- $s_n \in F$.

The *result* of the run is $\omega(s_n, \eta_n)$. For every input sequence, $\mathcal{A}$ has either no run at all or a unique run. If $\mathcal{A}$ has a run on σ, we say that $\mathcal{A}$ *accepts* σ with result $\omega(s_n, \eta_n)$.

Since the only allowed update operations are reset, increment, maximum, and minimum, the values of the variables are always either ∞ or bounded by the length of the input sequence. These values can therefore be represented in logarithmic space in the length of the input.

Lemma 1. *The configuration of a measuring automaton can be represented in logarithmic space in the length of the input sequence.*

From Formulas to Automata. We measure formulas of PLTL^{det} with a measuring automaton with input alphabet $\Sigma = 2^{AP}$ and output domain $\Omega : V \to \mathbb{N}$, consisting of the evaluations of the parameters. The state space of the automaton is based, as in classic LTL-to-automata translations, on the *closure* of the formula.

Definition 2. *The* closure φ*, denoted by* $cl(\varphi)$*, of a PLTL formula* φ *is the set of PLTL formulas that includes all the subformulas of* φ *and the negations of the non-parametric subformulas of* φ*.*

The states of the measuring automaton consist (in addition to a unique initial state q_0) of subsets of the closure called *atoms*. Intuitively, an atom represents the state of the temporal specification after processing a prefix of the trace.

Definition 3. *An* atom *of a PLTL formula* φ *is subset of formulas from* $cl(\varphi)$ *that is consistent with respect to propositional logic, locally consistent with respect to the until, release, globally, and parametric globally operators, and maximal.*

- *A subset* $A \subseteq cl(\varphi)$ *of the closure is* consistent with respect to propositional logic *if the following conditions hold:* $(\psi_1 \wedge \psi_2) \in A$ *iff* $\psi_1 \in A$ *and* $\psi_2 \in A$*;* $\psi \in A$ *implies that* $\neg\psi \notin A$*; and true* $\in cl(\varphi)$ *implies that true* $\in A$*.*

- *A subset* $A \subseteq cl(\varphi)$ *of the closure is* locally consistent with respect to the until operator *if for all* $(\psi_1 \,\mathcal{U}\, \psi_2) \in cl(\varphi)$ *the following conditions hold:* $\psi_2 \in A$ *implies that* $(\psi_1 \,\mathcal{U}\, \psi_2) \in A$*; and* $(\psi_1 \,\mathcal{U}\, \psi_2) \in A$ *and* $\psi_2 \notin A$ *implies that* $\psi_1 \in A$*;*
- *A subset* $A \subseteq cl(\varphi)$ *of the closure is* locally consistent with respect to the release operator *if for all* $(\psi_1 \,\mathcal{R}\, \psi_2) \in cl(\varphi)$ *the following conditions hold:* $\psi_1 \in A$ *and* $\psi_2 \in A$ *implies that* $(\psi_1 \,\mathcal{R}\, \psi_2) \in A$*; and* $(\psi_1 \,\mathcal{R}\, \psi_2) \in A$ *implies that* $\psi_2 \in A$*;*
- *A subset* $A \subseteq cl(\varphi)$ *of the closure is* locally consistent with respect to the globally operator *if for all* $\Box\psi \in cl(\varphi)$ *it holds that if* $\Box\psi \in A$ *then* $\psi \in A$*;*
- *A subset* $A \subseteq cl(\varphi)$ *of the closure is* locally consistent with respect to the parametric globally operator *if for all* $\Box_{\leq x}\psi \in cl(\varphi)$ *it holds that if* $\Box_{\leq x}\psi \in A$ *then* $\psi \in A$*.*
- *A subset* $A \subseteq cl(\varphi)$ *of the closure is* maximal *if, for all non-parametric subformulas* $\psi \in cl(\varphi)$*, we have that either* $\psi \in A$ *or* $\neg\psi \in A$*.*

Let At_φ denote the set of atoms of φ. We consider the following successor relation on atoms:

Definition 4. *Let* $\rightarrow \subseteq At_\varphi \times 2^{AP} \times At_\varphi$ *be a* successor relation *between atoms of* φ *and for each* t*,* $e \subseteq AP$*,* t'*, we have* $t \xrightarrow{e} t'$ *if* t' *is the smallest set s.t.*

- $t' \cap AP = e$*.*
- *If* $\bigcirc\psi \in t$ *then* $\psi \in t'$*.*
- *If* $\Box\psi \in t$*, then* $\Box\psi \in t'$*.*
- *If* $(\psi_1 \,\mathcal{U}\, \psi_2) \in t$ *then either* $\psi_2 \in t$ *or* $\psi_1 \in t$ *and* $(\psi_1 \,\mathcal{U}\, \psi_2) \in t'$*.*
- *If* $\Diamond\psi \in t$ *then* $\psi \in t$ *or* $\Diamond\psi \in t'$*.*
- *If* $\Diamond_{\leq x}\psi \in t$ *then* $\psi \in t$ *or* $\Diamond_{\leq x}\psi \in t'$*.*
- *If* $\Box_{\leq x}\psi \in t$ *then* $\psi \in t$ *and* $\psi \notin t'$ *or* $\Box_{\leq x}\psi \in t'$*.*

For PLTL$^{\text{det}}$, this successor relation leads to a deterministic automaton.

Lemma 2. *For every atom* $t \in At(\varphi)$ *of a PLTL*^{det} *formula* φ *and every event* e *there is at most one atom* $t' \in At(\varphi)$ *such that* $t \xrightarrow{e} t'$*.*

The transition function δ of the measuring automaton is therefore directly based on the successor relation:

- For the initial state q_0, the successor $\delta(q_0, e)$ is the unique atom that contains φ and is consistent with e, i.e., the atom $t \in At_\varphi$ with $\varphi \in t$ and $t \cap AP = e$, or $\perp$ if no such atom exists.
- For every other state $t \in At_\varphi$, the successor $\delta(t, e)$ is, whenever it exists, the unique atom t' with $t \xrightarrow{e} t'$. If no successor atom exists, then $\delta(t, e) = \perp$.

The set X of variables of the measuring automaton contains two variables n_v and m_v for each parameter $v \in V$. The variable n_v is a counter, similar to the counter used in the offline algorithm: n_v indicates the number of steps since the formula that starts with the operator that is parametric in v has existed in the atom. Since the same formula may be generated several times along the trace, e.g., the subformula $\Diamond_{\leq x}q$ of the formula $\Box(p \rightarrow \Diamond_{\leq x}q)$ (written in negation normal form) is generated whenever p is true, we

additionally keep track of the worst (maximal, if we need to minimize, or minimal, if we need to maximize the measure) value of n_v seen so far (thus ensuring that the final measurement will cover all occurrences). This is the purpose of the second variable m_v: for the upward-closed operator $\Diamond_{\leq x}$, m_x contains the greatest value of n_x seen so far, for the downward-closed operator $\Box_{\leq x}$, m_x contains the smallest value of n_x seen so far. In θ, we initialize n_x with 0 and m_x with ∞ for $\Box_{\leq x}$ and with 0 for $\Diamond_{\leq x}$.

For the update function $\gamma(s,e)$, which maps a state and an input to a mapping between successive valuations, we need to distinguish situations where a parametric formula is freshly generated from situations where the formula is present in order to continue a measurement started earlier.

Definition 5. *A formula $\psi \in t$ is* generated *in a pair of atoms $t,t' \in At_\varphi$, denoted by $generated(\psi,t,t')$, if one of the following conditions is true:*

- *ψ is the direct subformula of some other formula in t';*
- *$\bigcirc\psi$ is a formula in t.*

For simplicity of notation, we extend $generated(\psi,s,s')$ to pairs of states $s,s' \in Q$ of the measuring automaton, where we set $generated(\psi,q_0,t') = true$ for all $t' \in At_\varphi, \psi \in t'$ to also cover the designated initial state. The updates of m_x and n_x take into account whether we want to maximize or minimize a subformula. Also, the updates take into account the fact that a measured subformula may be regenerated while predecessor atoms already include that subformula, hence it is already in the process of being measured. For example, when monitoring $\Box(p \rightarrow \Diamond_{\leq x} q)$, the subformula $\Diamond_{\leq x} q$ may be generated, or a result of it appearing in the previous atom, or both. We need to measure an overall worst *minimal* value (i.e., maximum among minimal values from the time $\Diamond_{\leq x} p$ is generated in an atom until p it holds a subsequent atom). In this case, we ignore any new generation of the subformula in the current atom. The situation is reversed when we have an $\Box_{\leq x} q$ subformula: Since we need the worst among *maximal* measurements, we ignore the occurrence of the subformula that is propagated from a predecessor atom in favor of a current generation, and start to count from fresh.

For $\Diamond_{\leq x}\eta \in \delta(s,e)$: if $\eta \in t'$ then $m_x := \max(m_x,n_x); n_x := 0$
else $m_x := m_x$; $n_x := n_x + 1$.

For $\Box_{\leq x}\eta \in \delta(s,e)$: if $\eta \in \delta(s,e)$ then if $generated(\Box_{\leq x}\eta, s, \delta(s,e))$
then $m_x := m_x; n_x := 0$
else $m_x := m_x; n_x := n_x + 1$;
else $m_x := \min(m_x,n_x); n_x := 0$.

If x does not occur in the new atom, then m_x and n_x remain unchanged: $m_x := m_x$ and $n_x := n_x$. The final states of the measuring automaton are the atoms without unfulfilled obligations, i.e., $t \in F$, iff for all $\eta\, \mathcal{U} \mu \in t$ also $\mu \in t$, for all $\Diamond_{\leq x}\mu$ also $\mu \in t$, and there is no formula $\bigcirc\psi$ in t. For such states, ω reports the parameter assignment $v \mapsto m_v$ for all parameters, with the exception of remaining formulas of type $\Box_{\leq x}\mu$, where it reports $\min(m_x,n_x)$ to take care of a new possible minimum on the last event on the trace.

Theorem 3. *For every PLTL^{det} formula φ there exists a measuring automaton $\mathcal{A}_\varphi = (\Sigma, \Omega, Q, q_0, X, \theta, \delta, \gamma, F, \omega)$ with a linear number of states Q in $|\varphi|$ and a linear number of variables X in $|\varphi|$ such that for every sequence $\sigma \in (2^{AP})^*$, σ is accepted by $\mathcal{A}_\varphi$ with result r iff r is the measure of φ on σ.*

Soundness. We split the correctness argument of Theorem 3 into two lemmata:

Lemma 3. *If there exists a run of $\mathcal{A}_\varphi$ on the trace $\sigma = e_0 e_1 \dots e_{n-1} \in (2^{AP})^*$ with result r, then $(\sigma, 0, r) \models \varphi$.*

Lemma 4. *For a trace $\sigma = e_0 e_1 \dots e_{n-1} \in (2^{AP})^*$, if $(\sigma, 0, r) \models \varphi$, then there exists a run $\pi = (t_0, \eta_0), (t_1, \eta_1) \dots (t_n, \eta_n)$ of $\mathcal{A}_\varphi$ on σ with result r', where $r' \sqsubseteq r$.*

Because $\mathcal{A}_\varphi$ has at most one run on a given trace and therefore a unique result, provided that some run exists, Lemmata 3 and 4 imply that the result of $\mathcal{A}_\varphi$ is the measure of φ on the given trace. To prove Lemma 3, we first establish that the formulas in the atoms are satisfied for the respective suffixes of the trace.

Lemma 5. *If there exists a run of φ in $\mathcal{A}_\varphi$ with atom sequence $t_0 \dots t_n$, then we have that for all subformulas ψ, for all positions i, if $\psi \in t_i$ then $(\sigma, i-1, r) \models \psi$.*

The proof of Lemma 5 is by induction on the length of the trace, progressing backwards from the last position. It remains to show that the result of the run satisfies φ.

Lemma 6. *If there exists a run $\pi = (t_0, \eta_0), (t_1, \eta_1) \dots (t_n, \eta_n)$ in $\mathcal{A}_\varphi$ with result r, we have that $(\sigma, 0, r) \models \varphi$.*

To prove Lemma 6, we show, inductively, that for a subformula $\Diamond_{\leq x}\psi$ in atom t_i, the distance to the next atom with $[\psi]$ is at most $r(x) - \eta_i(n_x)$ steps; and, likewise, that for a subformula $\Box_{\leq x}\psi$ in atom t_i, the distance to the next atom with $\neg[\psi]$ is at least $r(x) - \eta_i(n_x)$ steps.

For the reverse direction, stated as Lemma 4, we first construct the sequence of atoms corresponding to the given trace. Based on the semantics definition, we show inductively that the subformulas in the atoms hold over the respective suffices.

Lemma 7. *For a trace $\sigma = e_0 e_1 \dots e_{n-1}$ and a deterministic PLTL formula φ, if $(\sigma, 0, r) \models \varphi$, then there exists an atom sequence $t_0 \dots t_n$ such that t_0 is the unique atom that contains φ and is consistent with e_0, $t_i \xrightarrow{s_i} t_{i+1}$ for all $i = 0 \dots n-2$, and for every subformulas ψ and position $i = 0 \dots n-1$, if $\psi \in t_i$, $(\sigma, i-1, r) \models \psi$.*

We complete the atom sequence of Lemma 7 into a complete run by computing the values of n_x and m_x for each parameter x and each trace position according to the definition of the automaton.

Lemma 8. *For a trace $\sigma = e_0 e_1, \dots e_{n-1}$ and a deterministic PLTL formula φ, if $(\sigma, 0, r) \models \varphi$, then there exists a run of $\mathcal{A}_\varphi$, $\pi = (t_0, \eta_0), (t_1, \eta_1) \dots (t_n, \eta_n)$ where $t_i \xrightarrow{e_i} t_{i+1}$ and for all subformulas and positions i, if $\psi \in t_i$, $(\sigma, i-1, r) \models \psi$, with result $\omega(e_n, \eta_n) \sqsubseteq r$.*

To prove the claim in Lemma 8, that the result of the run is at least as good as r, we show, inductively, that for each subformula $\Diamond_{\leq x}\psi$ and each trace position i, $\eta_i(n_x)$ is less than or equal to the difference of $r(x)$ and the distance of the closest atom that contains $[\psi]$; and that for each subformula $\Box_{\leq x}\psi$ and each trace position i, the sum of $\eta_i(n_x)$ and the distance of the closest atom that contains $\neg[\psi]$ is greater than or equal to $r(x)$. Since m_x maintains for $\Diamond_{\leq x}\psi$ and for $\Box_{\leq x}\psi$, the maximum and minimum measure, respectively, the claim of Lemma 8 follows.

From Theorem 3 and Lemma 1 it follows that the space required by the online monitor is linear in the size of the specification and logarithmic in the length of the trace.

Theorem 4. *A PLTL^{det} formula φ can be measured in linear space in the size of φ and logarithmic space in the length of the trace.*

5.2 Unambiguous PLTL

As discussed in the introduction of Section 5, the disadvantage of the syntactic restriction in deterministic PLTL is that it affects the expressiveness of the logic. In *unambiguous PLTL*, we modify the semantics of PLTL, rather than its syntax, in order to determinize the measuring behavior. Because the change does not affect the truth value of the non-parametric subformulas, we maintain the full expressiveness of LTL. In particular, under the unambiguous interpretation we give priority in $(\varphi \vee \psi)$ to φ, and measure according to ψ only when $[\varphi]$ does not hold. Similarly, for $\Diamond_{\leq x}\varphi$, we measure x to the *first* occurrence where φ holds.

Syntax and Semantics. We use the full PLTL syntax as defined in Section 2. The changes in the semantics (we do not redefine the cases that remain the same) are as follows:

- $(\rho,k,\alpha) \models (\psi \vee \eta)$ if $(\rho,k,\alpha) \models \psi$ or $(\rho,k,\alpha) \models (\neg[\psi] \wedge \eta)$;
- $(\rho,k,\alpha) \models (\psi \,\mathcal{U}\, \eta)$ if there exists i where $|\sigma| < k+i$, such that $(\rho,k+i,\alpha) \models \eta$, and for each j, $0 \leq j < i$, $(\rho,k+j,\alpha) \models \psi \wedge [\neg\eta]$;
- $(\sigma,k,\alpha) \models \Diamond_{\leq x}\eta$ if there exists $0 \leq i \leq \alpha(x)$, where $k+i < |\sigma|$, such that $(\sigma,k+i,\alpha) \models \eta$ and for each j, $0 \leq j < i$, $(\rho,k+j,\alpha) \models [\neg\eta]$;
- $(\sigma,k,\alpha) \models (\psi \,\mathcal{R}\, \eta)$ if for each $k \leq i < |\sigma|$, either $(\sigma,i,\alpha) \models \eta \wedge [\neg\psi]$ or there exists j, $k \leq j < i$ such that $(\sigma,k+j,\alpha) \models \psi$.

The definition uses the LTL abstraction $[\psi]$ of a PLTL formula ψ, which is the LTL formula that is satisfied exactly if the PLTL formula is satisfiable for some instance of the parameters, i.e.,

- $[\Diamond_{\leq x}\psi] = \Diamond[\psi]$
- $[\Box_{\leq x}\psi] = [\psi]$
- $[p] = p; [\neg p] = \neg p; [(\psi_1 \wedge \psi_2)] = ([\psi_1] \wedge [\psi_2]); [(\psi_1 \vee \psi_2)] = ([\psi_1] \vee [\psi_2])$;
- $[\bigcirc\psi] = \bigcirc[\psi]; [\Diamond\psi] = \Diamond[\psi]; [\Box\psi] = \Box[\psi]$;
- $[(\psi_1 \,\mathcal{U}\, \psi_2)] = ([\psi_1] \,\mathcal{U}\, [\psi_2]); [(\psi_1 \,\mathcal{R}\, \psi_2)] = ([\psi_1] \,\mathcal{R}\, [\psi_2])$.

Example. Consider the PLTL formula $\varphi = a \,\mathcal{U} \Diamond_{\leq x} b$ on the trace $\sigma = \{a\}\{a,b\}\emptyset$. According to the standard semantics from Section 2, we have both $(\sigma, 0, x \mapsto 1) \models \varphi$, because $(\sigma, 0, x \mapsto 1) \models \Diamond_{\leq x} b$, and $(\sigma, 0, x \mapsto 0) \models \varphi$ because $(\sigma, 0, x \mapsto 0) \models a$ and $(\sigma, 1, x \mapsto 0) \models \Diamond_{\leq x} b$. The result according to the standard semantics is therefore 0. According to the unambiguous semantics, we only have $(\sigma, 0, x \mapsto 1) \models \varphi$, because $(\sigma, 0, x \mapsto 1) \models [\Diamond_{\leq x} b] = \Diamond b$, and $(\sigma, 0, x \mapsto 1) \models \Diamond_{\leq x} b$. The result is therefore 1.

Note that for formulas in the syntax of deterministic PLTL, the unambiguous and the standard semantics agree. Unambiguous PLTL is thus a strict generalization of deterministic PLTL.

From Formulas to Automata. Measuring unambiguous PLTL is more difficult than deterministic PLTL, since the disjuncts are no longer guarded by atomic propositions, we generally do not know the truth value of a disjunct until the end of the trace. Our construction exploits the fact that the counting that is needed to compute the measure of the trace is deterministic in the truth value of the temporal subformulas. While the actual value of the future formulas is not known during monitoring, it is possible to split the analysis into a fixed set of cases based on the possible truth values of the temporal subformulas.

The states of the measuring automaton are *sets* of atoms. The intuitive idea is that each atom represents a possible future behavior. The sets of atoms correspond to a determinization of the possible futures. As the execution unrolls, some of the future possibilities are ruled out. As before, an atom is a subset of the formulas of the closure. We extend the closure of the PLTL formula φ with the subformulas of $[\varphi]$ and the negations of the subformulas of $[\varphi]$. Under the unambiguous semantics, we additionally require unambiguity with respect to disjunction and until:

Definition 6. *An* atom *of an unambiguous PLTL formula φ is subset of formulas from $cl(\varphi)$ that is consistent with respect to propositional logic, locally consistent with respect to the until, release, globally, and parametric globally operators, maximal, and unambiguous with respect to disjunction and until.*

- *A subset $A \subseteq cl(\varphi)$ of the closure is* unambiguous with respect to disjunction *if, for every $\eta \vee \mu \in A$ either $[\eta] \in A$ and $\eta \in A$, or $\neg[\eta] \in A$ and $\mu \in A$.*
- *A subset $A \subseteq cl(\varphi)$ of the closure is* unambiguous with respect to the until operator *if, for every $\eta \,\mathcal{U}\, \mu \in A$, either $\eta \in A$ and $\neg[\mu] \in A$, or $\mu \in A$.*

The transition function δ of the measuring automaton computes the set of successor atoms analogously to the construction for deterministic PLTL; the difference is that the successor atoms are no longer unique and we maintain a set of atoms.

- For the initial state q_0, the successor $\delta(q_0, e)$ is the set of atoms that contain φ and are consistent with e, i.e., the atoms $t \in At_\varphi$ with φ and $t \cap AP = e$.
- For every other state $t = s \subseteq At_\varphi$, the successor $\delta(s, e)$ is the set of atoms t' with $t \xrightarrow{e} t'$ for some $t \in s$.

We observe that the only nondeterminism in the successor relation $\xrightarrow{e}$ is in the selection of the non-parametric subformulas; once the non-parametric formulas have been chosen, the parametric formulas to be included in an atom are determined.

Lemma 9. *Let t,t' be two atoms in a state of the measuring automaton reached after reading some trace $\sigma = e_0e_1e_2\dots e_n$, such that there exists a sequence $t_1t_2\dots t_{n+1}$ of atoms with $t_i \in s_i$ for $i = 1\dots n+1$ and $t_i \xrightarrow{e_i} t_{i+1}$ for $i = 1\dots n$ such that $t_{n+1} = t$, and a sequence $t'_1t'_2\dots t'_{n+1}$ of atoms with $t'_i \in s_i$ for $i = 1\dots n+1$ and $t'_i \xrightarrow{e_i} t'_{i+1}$ for $i = 1\dots n$ such that $t'_{n+1} = t'$. If t_i and t'_i agree on the non-parametric formulas for all $i = 1\dots n+1$, then $t = t'$.*

The set X of variables contains now the variables $n_{x,t}$ and $m_{x,t}$ for each parameter $x \in V$ and each atom $t \in At_\varphi$, where the intended meaning is the same as in the construction for deterministic PLTL: the variable $n_{x,t}$ is the counter, the variable $m_{x,t}$ maintains the greatest counter value reached so far if x is an upward-closed parameter, and the smallest counter value reached so far if x is a downward-closed parameter.

As before, θ initializes n_x with 0, and m_x with ∞ for $\Box_{\leq x}$ and with 0 for $\Diamond_{\leq x}$. The key observation that allows us to define the update function γ is that every atom has a unique predecessor: while, for a pair of successor states s,s', each atom $t \in s$ may have multiple atoms $t' \in s'$ such that $t \xrightarrow{e} t'$, we have that, reversely, each atom $t' \in s'$ has exactly one atom in s with $t \xrightarrow{e} t'$; we denote this unique atom by $pre(s,s',t')$.

Lemma 10. *Let t_1,t_2 be two atoms in a state of the measuring automaton reached after reading some trace σ, and let t' be an atom in the state reached after reading the additional event e, such that $t_1 \xrightarrow{e} t'$ and $t_2 \xrightarrow{e} t'$. Then $t_1 = t_2$.*

The update function γ computes the new values of $n_{x,t}$ and $m_{x,t}$ based on the values of $n_{x,pre(s,s',t')}$ and $m_{x,pre(s,s',t')}$, i.e.,

$$\begin{aligned}\text{for } \Diamond_{\leq x}\eta \in t'\text{: if } \eta \in t' \text{ then } & m_{x,t'} := \max(m_{x,pre(s,s',t')}, n_{x,pre(s,s',t')}); n_{x,t'} := 0\\ \text{else } & m_{x,t'} := m_{x,pre(s,s',t')};\ n_{x,t'} := n_{x,pre(s,s',t')} + 1;\end{aligned}$$

$$\begin{aligned}\text{for } \Box_{\leq x}\eta \in t'\text{: if } \eta \in t' \text{ then if } & generated(\Box_{\leq x}\eta, pre(s,s',t'),t')\\ & \text{then } m_{x,t'} := m_{x,pre(s,s',t')}; n_{x,t'} := 0\\ & \text{else } m_{x,t'} := m_{x,pre(s,s',t')}; n_{x,t'} := n_{x,pre(s,s',t')} + 1;\\ \text{else } & m_{x,t'} := \min(m_{x,pre(s,s',t')}, n_{x,pre(s,s',t')}); n_{x,t'} := 0.\end{aligned}$$

If x does not occur in the atom, then $m_{x,t'} := m_{x,pre(s,s',t')}$ and $n_{x,t'} := n_{x,pre(s,s',t')}$. The final states of the measuring automaton are the sets that contain an atom without unfulfilled obligations, i.e., $f \in F$ iff there is a $t \in f$ such that for all $\eta\,\mathcal{U}\mu \in t$ also $\mu \in t$, for all $\eta\,\mathcal{R}\mu \in t$ also $\eta \in t$, and there is no formula $\bigcirc\psi$ in t. As the following lemma clarifies, every reachable final state $f \in F$ contains in fact exactly one such atom.

Lemma 11. *Every final state $f \in F$ reached by the measuring automaton on some trace contains exactly one atom $t \in f$, without unfulfilled obligations, i.e., where for all $\eta\,\mathcal{U}\mu \in t$ also $\mu \in t$, for all $\eta\,\mathcal{R}\mu \in t$ also $\eta \in t$, and there is no formula $\bigcirc\psi$ in t.*

The output is based on this unique atom $t \in f$: ω reports the parameter assignment $x \mapsto m_{x,t}$ for all parameters, with the exception of remaining formulas of type $\Box_{\leq x}\mu$, where it reports $\min(m_{x,t}, n_{x,t})$ to take care of a new possible minimum on the last event on the trace.

Theorem 5. *For every PLTL formula* φ *there exists a measuring automaton* $\mathcal{A}_\varphi = (\Sigma, \Omega, Q, q_0, X, \theta, \delta, \gamma, F, \omega)$ *with an exponential number of states* Q *in* $|\varphi|$ *and a linear number of variables* X *in* $|\varphi|$ *such that for every sequence* $\sigma \in (2^{AP})^*$, σ *is accepted by* $\mathcal{A}_\varphi$ *with result* r *iff* r *is the measure of* φ *on* σ *under the unambiguous semantics.*

Soundness. The proof of the correctness of the construction of $\mathcal{A}_\varphi$ in Theorem 5 follows the structure of the proof of Theorem 3 for the corresponding construction for deterministic PLTL. The key difference is in the proof of Lemma 7, where we claim that for every trace σ and formula φ, if $(\sigma, 0, r) \models \varphi$, then there exists an atom sequence that satisfies the successor relation. For deterministic PLTL, this sequence can be constructed in a simple induction, progressing from the first position forwards, because the semantics is deterministic, i.e., the subformulas of the successor atom are uniquely determined by the present atom and the next event. For unambiguous PLTL, the parametric subformulas are chosen based on the truth value of the non-parametric subformulas. We therefore construct the sequence of atoms in two steps. In the first step, we compute, progressing backwards from the final position, precisely the set of non-parametric formulas that are satisfied in each position. In the second step, we add, progressing forwards from the initial position, the parametric subformulas according to the (now deterministic) semantics of unambiguous PLTL.

From Theorem 5 and Lemma 1 it follows that the space required by the online monitor is exponential in the size of the specification and logarithmic in the length of the trace.

Theorem 6. *Under the unambiguous semantics, a PLTL formula* φ *can be measured in exponential space in the size of* φ *and logarithmic space in the length of the trace.*

6 Experiments

We have implemented the offline measuring algorithm for PLTL from Section 3 and the online algorithm for unambiguous PLTL from Section 5. (Since unambiguous PLTL is a generalization of deterministic PLTL, the online algorithm handles the deterministic case as well.) The offline algorithm traverses the trace several times in backwards direction from the last event to the first event; the online algorithm traverses the trace just once in forward direction. Table 1 shows data from two experiments carried out with our Java implementation on a Intel Core i7 processor with 2.6 GHz and 8 GB main memory. The traces were generated based on simulation runs from two applications, a bus arbiter and a memory controller, and stored on a solid-state disk drive. We tested traces of varying length, between 10 000 and 10 000 000 events, and report the running time of both algorithms in milliseconds.

Bus arbiter. In this benchmark, we measure traces generated from an implementation of a synchronous bus arbiter for three clients. The parameters measure the duration between the occurrence of a request until a grant is given to the client.

Memory controller. Our second benchmark is a memory controller. The memory controller provides a bus interface to a memory module. We measure the retention period of a memory cell over a trace.

	bus arbiter		memory controller	
trace length	offline	online	offline	online
10k events	2027 ms	74 ms	444 ms	84 ms
100k events	6679 ms	263 ms	752 ms	246 ms
1M events	63711 ms	1484 ms	6275 ms	727 ms
10M events	180281 ms	13642 ms	65209 ms	15606 ms

Table 1. Running times for the offline and online measuring algorithms.

The online algorithm is significantly faster than the offline algorithm. In part, this can be explained by the fact that the offline algorithm has to perform several passes over the data, which is read anew from the disk every time. Additionally, the algorithm needs to evaluate all subformulas for every event, because a subformula might be satisfied from an earlier position onwards. The online algorithm only needs to evaluate the reachable sets of atoms and thus typically performs less work per event.

7 Conclusions

We have presented a logical and algorithmic framework for *runtime measuring*: a way to provide quantitative results for a trace that needs to conform with an LTL specification, based on duration parameters attached to the LTL subformulas. This is a declarative approach, where minimal/maximal results need to be reported, as opposed to an operational approach, where counters are spawned and updated explicitly while the LTL formula is being verified. The complexity of runtime measuring depends on the disjunctive nature of a formula. Due to explicit disjunctions and temporal operators with implicit disjunctive semantics, such as *until*, there can be multiple answers. Disambiguating the results using priority among the measured variables resulted in an algorithm that requires quasilinear space in the length of the trace.

We showed that an efficient online algorithm is possible if one determinizes the measuring semantics of PLTL. The completely deterministic semantics of deterministic PLTL leads to a logarithmic space requirement in the length of the trace and a linear space requirement in the size of the specification; the combination of standard LTL semantics for non-parametric formulas with deterministic measuring in unambiguous PLTL increases the space requirement to exponential in the size of the specification, but still maintains the logarithmic dependency on the length of the trace.

Our interpretation is based on finite sequences, and thus differs a bit from the usual LTL interpretation. Indeed, in runtime verification, only a finite part of the sequence is always revealed. This is consistent with the formation of runtime verification algorithms for LTL [8, 6]. A promising direction for future work is to consider an interpretation over infinite sequences. Then, we obtain three possible result values instead of the two Boolean values: *true*, *false* and *maybe* [10]. In the last position of the trace, we need to check whether the formulas in any of the closures are still satisfiable. Such a satisfiability algorithm would be similar to the one presented in [2].

Acknowledgments. This work was partly supported by the German Research Council (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS, www.avacs.org). The first author was supported by an IMPRS-CS PhD Scholarship. The third author was supported by ISF grant 126/12 "Efficient Synthesis Method of Control for Concurrent Systems".

References

1. Ben D'Angelo, Sriram Sankaranarayanan, César Sánchez, Will Robinson, Bernd Finkbeiner, Henny B. Sipma, Sandeep Mehrotra, Zohar Manna, Lola: Runtime Monitoring of Synchronous Systems. TIME 2005: 166-174.
2. Rajeev Alur, Kousha Etessami, Salvatore La Torre, Doron Peled, Parametric temporal logic for "model measuring". ACM Trans. Comput. Log. 2(3): 388-407 (2001).
3. Andreas Bauer, Martin Leucker, Christian Schallhart, Runtime Verification for LTL and TLTL. ACM Trans. Software Engineering Methodologies 20(4): 14 (2011)
4. Kousha Etessami, A note on a question of Peled and Wilke regarding stutter-invariant LTL, Information Processing Letters 75, Volume 200, 261–263.
5. Rob Gerth, Doron Peled, Moshe Y. Vardi, Pierre Wolper, Simple on-the-fly automatic verification of linear temporal logic. PSTV 1995: 3-18.
6. Klaus Havelund, Grigore Rosu, Synthesizing Monitors for Safety Properties. TACAS 2002, Grenoble, France, 342-356.
7. Bernd Finkbeiner, Sriram Sankaranarayanan, Henny Sipma, Collecting statistics over runtime executions. Proceedings of Runtime Verification (RV02), Electronic Notes in Theoretical Computer Science, Volume 70, Issue 4, 36-54 (2002).
8. Bernd Finkbeiner, Henny Sipma, Checking Finite Traces Using Alternating Automata. Formal Methods in System Design 24(2): 101-127 (2004).
9. Orna Kupferman, Nir Piterman, Moshe Y. Vardi, From Liveness to Promptness, Formal Methods in System Design 34(2): 83-103 (2009).
10. Orna Kupferman, Moshe Y. Vardi, Model Checking of Safety Properties, Formal Methods in System Design 19(3): 291-314 (2001).
11. Monika Maidl: The Common Fragment of CTL and LTL. FOCS 2000, Rodendo Beach, CA, USA, 643-652.
12. Zohar Manna, Amir Pnueli, Completing the Temporal Picture, Theoretical Computer Science 83, 91-130, 1991.
13. Amir Pnueli, Aleksandr Zaks: PSL Model Checking and Run-Time Verification Via Testers. FM 2006: 573-586.
14. Grigore Rosu, Feng Chen, Semantics and Algorithms for Parametric Monitoring, Logical Methods in Computer Science 8:1, 2012.
15. M. Vardi, P. Wolper, An Automata-Theoretic Approach to Automatic Program Verification, LICS 1986, Cambridge, MA, 332-344.