Bounded Variability of Metric Temporal Logic

Carlo A. Furia
Department of Computer Science, ETH Zurich, Switzerland
`bugcounting.net`
Paola Spoletini
Università degli Studi dell'Insubria, Italy
`paola.spoletini@uninsubria.it`

10 June 2013*

Abstract

Previous work has shown that reasoning with real-time temporal logics is often simpler when restricted to models with *bounded variability*—where no more than v events may occur every V time units, for given v, V. When reasoning about formulas with *intrinsic* bounded variability, one can employ the simpler techniques that rely on bounded variability, without any loss of generality. What is then the complexity of algorithmically *deciding* which formulas have intrinsic bounded variability?

In this paper, we study the problem with reference to Metric Temporal Logic (MTL). We prove that deciding bounded variability of MTL formulas is undecidable over dense-time models, but with a undecidability degree lower than generic dense-time MTL satisfiability. Over discrete-time models, instead, deciding MTL bounded variability has the same exponential-space complexity as satisfiability. To complement these negative results, we also briefly discuss small fragments of MTL that are more amenable to reasoning about bounded variability.

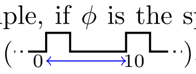
1 The Benefits of Bounding Variability

In yet another instance of the principle that "there ain't no such thing as a free lunch", expressiveness of formal languages comes with a significant cost to pay in terms of complexity—and possibly undecidability—of algorithmic analysis. The trade-off between expressiveness and complexity is particularly critical for the real-time temporal logics, which dwell on the border of intractability. A chief research challenge is, therefore, identifying expressive temporal logic fragments without letting the "dark side" of undecidability [6] prevail and abate practical usability.

Previous work by us [12, 14] and others [27, 10] has shown that the notion of *bounded variability* can help tame the complexity of real-time logics while still

*Document last updated on 4 July 2014.

retaining a reasonable expressive power. A model has variability bounded by v/V if there are at most v events every V time units. Consider a temporal logic formula ϕ: deciding whether ϕ has a model with variability bounded by some v/V is typically simpler than the more general problem of deciding whether ϕ has a model of *any* (possibly unbounded) variability (see Section 1.1 for examples). To close the gap between decidability over bounded variably models and general models, we should be able to determine if ϕ *only* has models with bounded variability. When this is the case, we lift the notion of bounded variability from models to formulas and say that ϕ has bounded variability. For formulas with bounded variability, we can apply the simpler algorithms that only consider bounded variably models, without losing generality in the analysis.

As a simple concrete example, if ϕ is the specification of a square wave of period 10 and duty cycle 30% (

), there are at most three transition events every 10 time units. Thus, all models of ϕ have variability bounded by 3/10, and we can leverage this fact to simplify the algorithmic analysis of ϕ.

This paper targets the bounded variability of formulas written in Metric Temporal Logic (MTL) [18], a popular linear-time temporal logic which extends LTL [9] with metric constraints and can be interpreted over both dense and discrete time domains. We study the complexity of the problem of determining if a generic MTL formula ϕ has bounded variability.

The bulk of the results is bad news: over dense-time models, deciding whether an MTL formula has bounded variability is undecidable; over discrete-time models, it is decidable, but with the same complexity as deciding validity[1] in general. These results are major hurdles to pursuing the idea of identifying formulas with bounded variability and then using the simpler algorithms for satisfiability on them: the complexity of the first step dominates, and nullifies the benefits of using the simplified algorithms for satisfiability under bounded variability.

As we show in Section 5.1 using reductions from undecidable problems of nondeterministic counter machines, the undecidability degree of deciding bounded variability over dense time is still lower than that of deciding validity: the former occupies the first two levels of the arithmetical hierarchy, whereas the latter belongs to Σ_1^1, the second level of the analytical hierarchy. In contrast, deciding bounded variability over discrete time is **EXPSPACE**-complete (see Section 5.2), the very same complexity as deciding validity; but the discreteness of the time domain entails that every formula has bounded variability for $v = V$.

While these results imply strong limits to reasoning about bounded variability in general, Section 6 suggests simpler cases where this may still be possible. If we identify MTL fragments that are sufficiently expressive to encode the requirement of bounded variability, yet have low complexity, we can try to establish bounded variability in special cases by considering subformulas of generic MTL formulas. We briefly illustrate two fragments, one for discrete- and one for dense-time models, that meet these requirements.

[1]Since MTL formulas are obviously closed under negation, validity and satisfiability are dual problems with the same complexity. Therefore, we indifferently use either term with reference to complexity.

1.1 Related Work

Originally introduced by Koymans [18] as a first-order real-time logic, MTL has become widespread in the propositional version popularized by Alur and Henzinger [2]. Their seminal work has also studied its complexity over dense and discrete time [2, 3], as well as interesting decidable fragments for dense time [1]. While their work basically settled the problems for discrete time, follow-up work by other authors has extended and refined the picture for dense time, such as by studying expressive completeness [16, 17], simplifying decision procedures [20], or identifying expressive decidable fragments [5, 22, 23].

Bounded variability is a natural semantic restriction over dense time, which has been applied to various formalisms including timed automata [27], duration calculus [10], and, in our previous work, MTL [12]. Recently, we also applied it to LTL over discrete time; while it is obvious that every discrete-time model has bounded variability (given by the fixed duration associated with one discrete time unit), in [14, 15] we showed how the LTL validity problem can be simplified under the assumption that only $v < V$ change events happen every V discrete time steps. Therefore, bounded variability can be a simplifying assumption also for discrete time.

The undecidability results of Section 5.1 use reductions from undecidable problems of nondeterministic n-counter machines, which we introduce in Section 4. These are a kind of Minsky's counter machines [21]; their connection with MTL was first exploited by Alur and Henzinger [2].

Section 6 discusses MTL fragments with lower complexity. Over discrete time, these fragments can be derived from similarly low-complexity fragment of LTL [9], which have been extensively studied by several authors [26, 7, 19].

2 Timed Words and Variability

We denote a generic time domain by $\mathbb{T}$. In the paper, $\mathbb{T}$ is either the *discrete* set of the nonnegative integers $\mathbb{N}$, or the *dense* (and *continuous*) set of the nonnegative reals $\mathbb{R}_{\geq 0}$.

An *interval* is a convex subset of the time domain, represented by a pair $\langle a, b\rangle$, where $\langle$ and $\rangle$ are square or round brackets to respectively denote inclusion or exclusion of the endpoint. We use the pseudo-arithmetic expressions $> s$, $\geq s$, $< s$, $\leq s$, and $= s$ as abbreviations for the intervals (s, ∞), $[s, \infty)$, $[0, s)$, $[0, s)$ and $[s, s]$. We assume a binary encoding of constants in the time domain unless explicitly stated otherwise.

Given a time domain $\mathbb{T}$ and a finite alphabet set $\mathcal{P}$ of atomic propositions, a *timed word* over $\mathbb{T}$ is a countably infinite sequence of pairs $\omega = (\sigma_0, t_0)\,(\sigma_1, t_1)\,(\sigma_2, t_2)\cdots$ such that:

1. Each integer $k \geq 0$ denotes a *position* in a timed word;
2. For each k, σ_k is a (nonempty, w.l.o.g.) subset of $\mathcal{P}$ denoting the propositions holding at position k; and $t_k \in \mathbb{T}$ is a timestamp denoting the time of the occurrence at position k;
3. The timestamps are strictly monotonic, that is $t_h > t_k$ iff $h > k$, and diverging, that is for all $t \in \mathbb{T}$ there exists k such that $t_k > t$ (divergence is subsumed by monotonicity in discrete time).

We also conventionally assume that $t_0 = 0$. The set of all timed words over $\mathbb{T}$ is denoted by $\mathcal{B}\mathbb{T}$.

A timed word ω has *variability bounded by* v/V, for $V \in \mathbb{T}$ and $v \in \mathbb{N}$, iff it has no more than v positions within any closed time interval of length V: for all $k \in \mathbb{N}$, $t_{k+v} - t_k > V$. The set of all timed words over $\mathbb{T}$ with variability bounded by v/V is denoted by $\mathcal{B}\mathbb{T}[v/V]$.

3 MTL: Metric Temporal Logic

We present the syntax and semantics of propositional MTL and recall some fundamental facts about its complexity.

Syntax. MTL formulas are defined by the grammar

$$\phi \quad ::= \quad \top \mid p \mid \neg\phi_1 \mid \phi_1 \wedge \phi_2 \mid \mathsf{U}_J(\phi_1, \phi_2)\,,$$

where p ranges over the alphabet $\mathcal{P}$, and J is an interval of the time domain with integer endpoints. We assume the standard definitions for *false*: $\bot$, and for the derived Boolean connectives: $\vee$, $\Rightarrow$, and $\Leftrightarrow$. The symbol α abbreviates the formula $\bigvee_{p\in\mathcal{P}} p$, which holds iff some proposition holds. We introduce the derived temporal operators *eventually*: $\Diamond_J(\phi) = \mathsf{U}_J(\top, \phi)$; *globally* (also, *always*): $\Box_J(\phi) = \neg\Diamond_J(\neg\phi)$; *action until*: $\widehat{\mathsf{U}}_J(\phi_1, \phi_2) = \mathsf{U}_J(\alpha \Rightarrow \phi_1, \phi_2)$; and *next*: $\bigcirc_J(\phi) = \widehat{\mathsf{U}}_J(\bot, \phi)$. Operator precedence is: $\neg$ has the highest precedence, then $\wedge$, then $\vee$, then $\Rightarrow$, then all temporal operators, and finally $\Leftrightarrow$. We may omit the parentheses around arguments when unambiguous, and drop intervals $[0, \infty)$.

Semantics. Given a timed word $\omega = (\sigma_0, t_0)\,(\sigma_1, t_1)\cdots$ and a position $k \in \mathbb{N}$, the *pointwise* satisfaction relation $\models_{\mathsf{p}}$ for an MTL formula ϕ is inductively defined as follows:

$$\begin{array}{lll}
\omega, k \models_{\mathsf{p}} \top; & & \\
\omega, k \models_{\mathsf{p}} p & \text{iff} & p \in \sigma_k\ ; \\
\omega, k \models_{\mathsf{p}} \neg\phi_1 & \text{iff} & \omega, k \not\models_{\mathsf{p}} \phi_1; \\
\omega, k \models_{\mathsf{p}} \phi_1 \wedge \phi_2 & \text{iff} & \omega, k \models_{\mathsf{p}} \phi_1 \text{ and } \omega, k \models_{\mathsf{p}} \phi_2; \\
\omega, k \models_{\mathsf{p}} \mathsf{U}_J(\phi_1, \phi_2) & \text{iff} & \text{there exists } h > k \text{ such that: } t_h - t_k \in J, \\
 & & \omega, h \models_{\mathsf{p}} \phi_2, \text{ and, for all } k < x < h,\ \omega, x \models_{\mathsf{p}} \phi_1; \\
\omega \models_{\mathsf{p}} \phi & \text{iff} & \omega, 0 \models_{\mathsf{p}} \phi\,.
\end{array}$$

The semantics of U and $\widehat{\mathsf{U}}$ coincide under the pointwise semantics, since formulas are only evaluated at positions, where α invariably holds. The (derived) semantics of next is: $\omega, k \models_{\mathsf{p}} \bigcirc_J(\phi_1)$ iff $t_{k+1} - t_k \in J$ and $\omega, k+1 \models_{\mathsf{p}} \phi_1$; that is, the next position has timestamp in J relative to the current one, and ϕ_1 holds there.

Given a timed word ω as above and a time instant $t \in \mathbb{T}$, the *continuous* satisfaction relation $\models_{\mathsf{c}}$ for an MTL formula ϕ is inductively defined as follows :

$$
\begin{array}{lll}
\omega, t \models_{\mathsf{c}} \top; & & \\
\omega, t \models_{\mathsf{c}} p & \text{iff} & \text{there exists } k \in \mathbb{N} \text{ such that: } t_k = t \text{ and } p \in \sigma_k; \\
\omega, t \models_{\mathsf{c}} \neg\phi_1 & \text{iff} & \omega, t \not\models_{\mathsf{c}} \phi_1; \\
\omega, t \models_{\mathsf{c}} \phi_1 \wedge \phi_2 & \text{iff} & \omega, t \models_{\mathsf{c}} \phi_1 \text{ and } \omega, t \models_{\mathsf{c}} \phi_2; \\
\omega, t \models_{\mathsf{c}} \mathsf{U}_J(\phi_1, \phi_2) & \text{iff} & \text{there exists } u > t \text{ such that: } u - t \in J, \\
& & \omega, u \models_{\mathsf{c}} \phi_2, \text{ and, for all } t < v < u,\ \omega, v \models_{\mathsf{c}} \phi_1; \\
\omega \models_{\mathsf{c}} \phi & \text{iff} & \omega, 0 \models_{\mathsf{c}} \phi\,.
\end{array}
$$

Over dense time, the continuous semantics generalizes the pointwise semantics in the sense that the former is strictly more expressive [8, 5]. The semantics of next under continuous semantics is, however, analogous to that over the pointwise semantics, thanks to the usage of $\widehat{\mathsf{U}}$ in its definition.

Remark 1. In the following, we assume the pointwise semantics over discrete time $\mathbb{N}$, and the continuous semantics over dense time $\mathbb{R}_{\geq 0}$. Our results for dense time are also transferable to the pointwise semantics *mutatis mutandis*, provided *past operators* are available (see Section 7).

Complexity: general models. Satisfiability of MTL formulas is highly undecidable over dense time, where it is Σ_1^1-hard [2]. It is instead decidable over discrete time, with an **EXPSPACE**-complete decidability problem [2] (which translates to doubly-exponential deterministic time). Over discrete time, the high complexity is essentially due to the succinctness of the binary encoding (the expressiveness is the same as LTL).

Complexity: bounded models. Bounded variability is a semantic restriction that reduces the complexity of MTL. In fact, we proved that satisfiability of MTL over dense-time models with variability bounded by v/V, for any given v/V, is **EXPSPACE**-complete [12], matching the complexity of MTL over discrete time, as well as that of other decidable dense-time logics [1, 16]. The following is a corollary of our previous results, which we use in this paper.

Corollary 2. *For any v, v', and V, it is decidable whether an MTL formula has some model over $\mathbb{R}_{\geq 0}$ with variability bounded by v'/V but not by v/V.*

Proof. We showed that the MTL satisfiability problem over $\mathcal{B}\mathbb{R}_{\geq 0}[v/V]$ is decidable for generic v/V [12, Corollary 1]; and that we can encode in MTL the bounded variability constrain as well as its complement [12, Section 4.3]. □

In recent work [14, 15], we showed how the notion of bounded variability can reduce the complexity of MTL over discrete time as well. While bounded variability does not affect the exponential-space worst-case complexity, since discrete-time models have inherently bounded variability, it can reduce the complexity in practice. Precisely, when studying the variability of an arbitrary LTL formula ϕ over behaviors with variability bounded by any given v/V, with $v < V$, we can consider a simplified ϕ' whose size depends on v but not on the distances encoded in ϕ through next operators. While the results of [14, 15] target LTL, it is clear that they carry over to MTL over discrete time.

4 Counter Machines

Counter machines [21] are powerful computational devices, widely used in formal language theory. We use a nondeterministic version of counter machines, and derive some complexity results which we use in the remainder.

Definition 3. An n-counter machine executes programs consisting of a finite list of instructions with labels $\ell_0, \ell_1, \dots$ and operating on n integer counter variables $v_0, \dots, v_{n-1}$. An instruction is one of the following:

halt	terminate computation
if $v_k > 0$ **goto** ℓ_i, ℓ_j	conditional branch
inc v_k	increment counter
dec v_k	decrement counter

where the conditional branch consists in jumping to ℓ_i or ℓ_j nondeterministically if counter v_k is non-zero; and decrementing a counter with zero value is undefined. Computations start at location ℓ_0 with all counters equal to zero and proceed according to the obvious semantics of instructions. Without loss of generality, assume that instruction **halt** occurs exactly once and that the last instruction in the list is either **halt** or a branch.

For n-counter machines, with $n \geq 2$, the halting problem (deciding whether the location with **halt** is visited in some computation) is Σ_1^0-complete (**RE**-complete: undecidable but semidecidable); the non-halting problem (deciding whether some computation does not halt) is Σ_2^0-complete; the recurring computation problem (deciding whether location ℓ_0 is visited infinitely often in some computation) is Σ_1^1-hard [3].[2]

4.1 Bounded and Unbounded Counters

Consider the following decision problems for n-counter machines:

bounded counter: given an integer β, decide whether v_0 overflows β in some computation;

finite counter: decide whether there exists β such that $v_0 \leq \beta$ in all computations;

unbounded counter: decide whether v_0 is incremented infinitely often in some computation.

Theorem 4. *The bounded counter problem is Σ_1^0-complete; the finite counter problem is Σ_2^0-complete; the unbounded counter problem is Σ_1^1-hard.*

Proof. We prove hardness by reduction from, respectively, the halting, non-halting, and recurring computation problems of n-counter machines. We then report the simpler corresponding completeness proofs.

Hardness of the bounded counter problem. Given a generic n-counter machine M, we reduce halting to bounded counter for $\beta = 0$ by modifying M into M' as follows. Add one counter and injectively rename all counters in the instruction list so that the new counter is called v_0; thus, v_0 is not mentioned in the renamed instructions. Then, replace the unique halting instruction appearing at some ℓ_h in M by two instructions: ℓ_h: **inc** v_0 followed by ℓ_h^+: **halt**.

[2] [3] discusses 2-counter machines, but the generalization to n-counter machines is immediate. The other complexities follow from reduction of the same problems for Turing machines.

Since we only added deterministic instructions, there is a one-to-one correspondence between computations of M and computations of M'. A generic nondeterministic computation χ of M reaches location ℓ_h iff the unique corresponding computation χ' of M' also reaches ℓ_h. In such computations χ', v_0 overflows β before halting at ℓ_h^+. In all, some computation of M halts iff v_0 overflows in some computation of M'. Thus, the bounded counter problem is Σ_1^0-hard.

Hardness of the finite counter problem. Given a generic n-counter machine M, we reduce from the non-halting problem. Create another counter machine M' with a fresh counter v_0, which works as follows. M' simulates all computations of M deterministically: as soon as a specific computation terminates, M' backtracks the simulation and makes a different nondeterministic choice. (We omit the details of the simulation, which are straightforward.) Whenever the simulation completes a halting computation of M, it increments v_0 before continuing with the next computation. If the simulation ever comes to an end (that is, if M has only finitely many computations, all halting), M' enters an infinite loop that makes v_0 diverge. Therefore, M' has only one non-halting (because either M' enters the infinite loop or M has infinitely many computations) deterministic execution.

Consider now the finite counter problem for M'. If it has answer YES, it means that the simulation eventually executes a non-halting computation of M; from that point on, v_0 is never incremented. If it has answer NO, it means that the simulation consists of infinitely many halting computations of M, or that it reached the divergent loop and hence M had only finitely many halting computations. The answer to the non-halting problem for M is therefore the same in either case. This shows that we reduced the non-halting problem to the finite counter problem, and both are Σ_2^0-hard.

Hardness of the unbounded counter problem. Given a generic n-counter machine M, we reduce recurring computation to unbounded counter by modifying M into M' as follows. Add one counters and injectively rename all counters in the instruction list so that the new counter is called v_0. Then, replace the instruction I appearing at location ℓ_0 in M by two instructions as follows: ℓ_0: **inc** v_0; ℓ_0': I. All other instructions follow ℓ_0' as they followed ℓ_0 in M.

Also in this case we only added deterministic instructions; hence there is a one-two-one correspondence between computations of M and computations of M'. A generic nondeterministic computation χ of M visits location ℓ_0 infinitely often iff the unique corresponding computation χ' of M' also reaches the new ℓ_0 infinitely often; such computations χ' increment v_0 infinitely often when executing ℓ_0. In all, some computation of M visits ℓ_0 infinitely often iff v_0 is incremented infinitely often in some computation of M'. Thus, the unbounded counter problem is Σ_1^1-hard.

Completeness of the bounded counter problem. We reduce the bounded counter problem (for any β) to halting, thus showing that the former is in Σ_1^0 (and hence, by combining it with the hardness result, Σ_1^0-complete). The idea is to guard every increment to v_0 with a conditional of the form **if** $v_0 \geq \beta$ **goto** ℓ_h **else inc** v_0, where ℓ_h is the halting location. Since v_0 is initially zero, a computation halts iff it overflowed in the initial program. The details of how to encode such modifications using standard instructions are straightforward.

Completeness of the finite counter problem. We show that the finite counter problem is in Σ_2^0 (and hence, by combining it with the hardness result, Σ_2^0-complete) according to the definition of Σ_2^0 in the arithmetical hierarchy [25]. Let $\mathcal{O}_\beta$ be the set of all counter machines where v_0 overflows β in some computation. Previously, we have shown that $\mathcal{O}_\beta$ is Σ_1^0; hence its complement set $\overline{\mathcal{O}_\beta}$—all counter machines where $v_0 \leq \beta$ in all computations—is Π_1^0. The set $\mathcal{F}$ of all counter machines for which the finite counter problem has answer YES is defined by $M \in \mathcal{F} \iff \exists \beta : \overline{\mathcal{O}_\beta}$, and hence it is Σ_2^0. $\square$

4.2 MTL and Counter Machines

Alur and Henzinger [2] pioneered the usage of counter machines to analyze the complexity of real-time logics. Using their techniques, we show the essentials of how to encode computations of n-counter machines as MTL formulas over $\mathbb{R}_{\geq 0}$: computations are encoded as timed words; and, given a machine M, we build an MTL formula Γ_M that is satisfied precisely by the words encoding M's computations.

Consider an n-counter machine M with $m+1$ instructions $\ell_0, \ldots, \ell_m$, such that ℓ_h is the location of the unique halt instruction. We introduce the following propositions: p_k, for $0 \leq k \leq m$, which holds when M is at location ℓ_k; and z_k, for $1 \leq k \leq n$, which we use to represent the value of counter v_k: there are as many distinct occurrences of proposition z_k over a unit interval as the value of counter v_k in the corresponding configuration. A configuration is a tuple $\langle \ell_k, x_1, \ldots, x_n \rangle$ denoting that M is at location ℓ_k and the counters store the values $x_1, \ldots, x_n$. At each integer time instant: all propositions z_d's are false; and exactly one of the propositions p_k's holds, with p_0 holding initially. The p_k's are all false everywhere else:

$$p_0 \wedge \left(\begin{array}{l} \bigwedge_{1\leq k\leq m} \square \Big(p_k \Rightarrow \bigwedge_{1 \leq j \neq k \leq m} \neg p_j \ \wedge\ \bigwedge_{1 \leq d \leq n} \neg z_d \Big) \wedge \\ \bigwedge_{1\leq k\leq m} \square \Big(p_k \Rightarrow \bigvee_{1\leq j \leq m} \mathsf{U}_{=1} \Big(\bigwedge_{1 \leq i \leq m} \neg p_i, p_j \Big)\Big) \end{array} \right).$$

With similar formulas, we constrain the z_k's to occur at distinct instants: whenever z_k then $\neg z_h$ also holds simultaneously, for $h \neq k$.

Each time interval $[t, t+1)$, for $t \in \mathbb{N}$, encodes the $(t+1)$-th configuration reached during a valid computation: p_k holding at t means that M is at location ℓ_k; and, for $1 \leq j \leq n$, z_j holds over $[t, t+1)$ exactly as many times as the integer value stored in counter v_j. The initial configuration $\langle \ell_0, 0, \ldots, 0 \rangle$ is encoded by

$$\bigwedge_{1 \leq j \leq n} \square_{[0,1]} (\neg z_j) \ .$$

The encoding of any instruction refers to a current time $t \in \mathbb{N}$ and defines the state over $[t+1, t+2)$ as a modification of the state over $[t, t+1)$. The most significant operation is the increment: ℓ_k: **inc** v_c, whose MTL encoding declares that the state in the next interval has exactly one more occurrence of

z_c than it has in the current interval:

$$\Box\left(p_k \Rightarrow \left(\begin{array}{l} \Diamond_{=1}\, p_{k+1} \\ \wedge\ \bigwedge_{1 \leq d \neq c \leq n} \Box_{(0,1)}(z_d \Leftrightarrow \Diamond_{=1}\, z_d) \\ \wedge\ \Box_{(0,1)}(z_c \Rightarrow \Diamond_{=1}\, z_c) \\ \wedge\, \mathsf{U}_{(0,1)} \left(\begin{array}{l} \Diamond_{=1}\, z_c \Rightarrow z_c, \\ \neg z_c \wedge \Diamond_{=1} z_c \wedge \\ \mathsf{U}_{>0}(\neg z_c \wedge \Diamond_{=1}(\neg z_c)\,, p_{k+1}) \end{array} \right) \end{array} \right)\right). \tag{1}$$

In (1)'s consequent, the first conjunct states that ℓ_{k+1} is the next location visited (since this is not a branch instruction). The second conjunct states that the values of all counters other than v_c are unchanged: for every occurrence of some z_d in the current interval, there is an occurrence exactly one time unit later in the next interval and vice versa; hence occurrences of z_d are "copied" from the current to the next interval. Similarly, the third conjunct declares that v_c does not decrease (z_c's occurrences in the current interval are copied into the next one). The fourth conjunct asserts that there exists an instant, after the last occurrence of z_c in the current interval and before the next occurrence of p_{k+1} at the beginning of the next interval, such that z_c occurs exactly once at the corresponding instant in the next interval. This new distinct occurrence of z_c is always possible thanks to the density of the temporal domain; thus any value of counters can be stored in a unit time interval. The encoding of other instructions is similar, with the halting instruction determining an indefinite repetition of the final configuration in the future.

Remark 5. Over pointwise semantics, we can express a behavior analogous to (1) using past operators. The key observation [22] is that the "copy" of a counter v_d can be expressed as $\Box_{(0,1)}(z_d \Rightarrow \Diamond_{=1}\, z_d)$ and $\Box_{(1,2)}(z_d \Rightarrow \overleftarrow{\Diamond}_{=1}\, z_d)$, where $\overleftarrow{\Diamond}_{=1}(\phi)$ holds iff its arguments held one time unit in the past.

5 The Complexity of Bounded Variability

Given a time domain $\mathbb{T}$ and a formula ϕ, we define two decision problems—the second is a generalization of the first—that deal with ϕ's bounded variability. We write $\mathcal{B}(\phi)$ to denote the subset of a set $\mathcal{B}$ of timed words that satisfy ϕ.

$BV_{\mathbb{T}}(v, V)$: Determine whether every model of ϕ over $\mathbb{T}$ has variability bounded by v/V: does $\mathcal{B}\mathbb{T}(\phi) \subseteq \mathcal{B}\mathbb{T}[v/V](\phi)$?

$BV_{\mathbb{T}}$: Determine whether there exist v, V such that the answer to $BV_{\mathbb{T}}(v, V)$ is YES: does $\exists\, v, V : \mathcal{B}\mathbb{T}(\phi) \subseteq \mathcal{B}\mathbb{T}[v/V](\phi)$?

A bar denotes the corresponding *complement* problems: $\overline{BV}_{\mathbb{T}}(v, V)$ asks whether some model of ϕ has variability not bounded by v/V (bounded by v'/V for some $v' > v$, or unbounded); $\overline{BV}_{\mathbb{T}}$ asks whether, for every v, V, some model of ϕ has variability not bounded by v/V. Notice that the latter is not the same as asking if some model of ϕ has unbounded variability: it may as well be that every model of ϕ has bounded variability, but no variability bounds all of the models.

This section establishes the complexity of the decision problems for $\mathbb{T} = \mathbb{R}_{\geq 0}$ (Section 5.1) and $\mathbb{T} = \mathbb{N}$ (Section 5.2).

5.1 Complexity of Bounded Variability over Continuous Time

Both variants of the bounded variability problems just introduced are undecidable over continuous time, but with different undecidability degrees in the arithmetical hierarchy; in both cases, however, the undecidability degree is lesser than MTL satisfiability, which is highly undecidable (Σ^1_1-hard [2]).

Theorem 6. *$BV_{\mathbb{R}_{\geq 0}}(v, V)$ is $\Pi^0_1 = \mathsf{coRE}$-complete; $BV_{\mathbb{R}_{\geq 0}}$ is Σ^0_2-complete.*

Proof. The completeness result for $BV_{\mathbb{R}_{\geq 0}}(v, V)$ is proved in Lemmas 7 and 8. The completeness result for $BV_{\mathbb{R}_{\geq 0}}$ is proved in Lemmas 9 and 10. $\square$

Lemma 7. *$BV_{\mathbb{R}_{\geq 0}}(v, V)$ is in $\Pi^0_1 = \mathsf{coRE}$.*

Proof. We give a procedure to semi-decide $\overline{BV}_{\mathbb{R}_{\geq 0}}(v, V)$; this establishes that $\overline{BV}_{\mathbb{R}_{\geq 0}}(v, V) \in \mathsf{RE}$ and thus $BV_{\mathbb{R}_{\geq 0}}(v, V) \in \mathsf{coRE}$ by complement.

Consider a generic MTL formula ϕ. Some model of ϕ has variability not bounded by v/V iff: (a) some model of ϕ has variability bounded by v'/V but not by v/V, for some $v' > v$; or (b) some model of ϕ has unbounded variability. Since we are dealing with divergent models only (see Section 2), (b) can only occur with models where the variability is bounded up to any finite time t, but the variability bound increases indefinitely over time.[3]

For any finite time T, let $\phi[T]$ denote the MTL formula which restricts the evaluation of ϕ to the finite time interval $[0, T]$. This can be constructed as follows: add a fresh proposition e constrained by $\phi_e = \mathsf{U}_{=T}(e, e \wedge \Box_{>0}\neg e)$. Rewrite ϕ in negation normal form, and replace every atom q by $e \Rightarrow q$. Postulate that, if e is false, all other propositions in $\mathcal{P}$ are false as well: $\phi_{\mathcal{P}} = \Box(\neg e \Rightarrow \bigwedge_{p \in \mathcal{P}} \neg p)$. Finally, $\phi[T]$ is $\phi_e \wedge \phi \wedge \phi_{\mathcal{P}}$. Since no event occurs after finite time T, all models of $\phi[T]$ have variability bounded by x/T, for some finite (possibly very large) x. Therefore, some model of $\phi[T]$ has variability not bounded by t/T iff some model of $\phi[T]$ has variability bounded by t'/T but not by t/T, for some $x \geq t' > t$.

We can now describe a procedure P_1 that semi-decides $\overline{BV}_{\mathbb{R}_{\geq 0}}(v, V)$; it consists of the following steps:

1. Initially, $\delta := v + 1$ and $\Delta := V + 1$;
2. Using Corollary 2, decide whether $\phi[\Delta]$ has some model with variability bounded by δ/V but not by v/V;
3. If it does, stop and return YES;
4. Otherwise $\delta := \delta + 1$, $\Delta := \Delta + 1$, and go to (*2*).

If the answer to $\overline{BV}_{\mathbb{R}_{\geq 0}}(v, V)$ is YES, then either (a) or (b) above holds; let us show that, in both cases, P_1 terminates with the correct answer.

If (a) is the case, let ω_a be a model with variability bounded by v'/V but not by v/V for some $v' > v$; that is, ω_a has $\overline{v}$ events, for $v < \overline{v} \leq v'$, over some time interval $[x, x + V]$. In this case, P_1 terminates with YES as soon as $\delta \geq \overline{v}$ and $\Delta \geq x + V$.

If (b) is the case, let ω_b be a model with unbounded variability; since variability is unbounded, there exists a time T such that: ω_b has $v' > v$ events over

[3] In related work, we called similar behaviors "Berkeley" [13, 11].

some time window $[x, x+V]$, for $0 \leq x < x+V \leq T$. In this case, P_1 terminates with YES as soon as $\delta \geq v'$ and $\Delta \geq T$. $\square$

Lemma 8. *$BV_{\mathbb{R}_{\geq 0}}(v, V)$ is* **coRE**-*hard.*

Proof. We reduce the bounded counter problem (Section 4.1) of 2-counter machines to $\overline{BV}_{\mathbb{R}_{\geq 0}}(v, V)$; the lemma follows by Theorem 4 through complement problems.

Consider a generic 2-counter machine M with counters v_0 and v_1. We construct an MTL formula Γ_M that encodes the computations of M along the lines of Section 4.2, but with some modifications. For $t \in \mathbb{N}$, the $(t+1)$-th configuration $\langle \ell_k, x_0, x_1 \rangle$ is encoded over the time interval $[4t, 4t+4)$ as follows: p_k holds at $4t$, z_0 holds x_0 times over $(4t+1, 4t+2)$, z_1 holds x_1 times over $(4t+3, 4t+4)$, and no propositions hold elsewhere over the whole $[4t, 4t+4)$. With this spacing of counter events, we can see that the models of Γ_M are such that any interval of length 1 includes at most as many events as the largest value held by a counter during some computation. Thus, Γ_M has some model with variability not bounded by $\beta/1$, which is an instance of $\overline{BV}_{\mathbb{R}_{\geq 0}}(v, V)$, iff a counter overflows β in some computation of M.

Now we have only established whether *some* counter overflows in M, whereas the bounded counter problem specifically targets overflows of v_0. To close the gap, we encode the overflowing of v_0 in M as an MTL formula $\Xi_\beta^{v_0}$:

$$\Diamond \left(\left(\bigvee_{0 \leq k \leq m} p_k \right) \wedge \overbrace{\Diamond_{(0,1)} \Big(z_0 \wedge \Diamond_{(0,1)} (z_0 \wedge \cdots) \Big)}^{\beta+1 \text{ nested diamonds}} \right) .$$

Thanks to the padding, the nested diamonds evaluate to true iff there are at least $\beta + 1$ distinct occurrences of z_0 in the slot corresponding to one configuration. Thus, v_0 overflows β in M iff $\Gamma_M \wedge \Xi_\beta^{v_0}$ has some model with variability not bounded by $\beta/1$. $\square$

Lemma 9. *$BV_{\mathbb{R}_{\geq 0}}$ is in Σ_2^0.*

Proof. Given the definition of Σ_2^0 in the arithmetical hierarchy [25], it is sufficient to provide an enumeration of all MTL formulas ϕ for which the answer to $BV_{\mathbb{R}_{\geq 0}}$ is YES, relative to an oracle for $BV_{\mathbb{R}_{\geq 0}}(v, V)$, which is in Π_1^0 by Lemma 7. To this end, we dovetail [24, Chap. 3] through all pairs (v, ϕ) of nonnegative integers $v \in \mathbb{N}$ and MTL formulas ϕ. For each pair, if the answer to $BV_{\mathbb{R}_{\geq 0}}(v, 1)$ is YES for ϕ, then the answer to $BV_{\mathbb{R}_{\geq 0}}$ also is YES for ϕ. It is clear that this enumeration eventually finds all formulas for which the answer to $BV_{\mathbb{R}_{\geq 0}}$ is YES. $\square$

Lemma 10. *$BV_{\mathbb{R}_{\geq 0}}$ is Σ_2^0-hard.*

Proof. We reduce the finite counter problem (Section 4.1) of n-counter machines to $BV_{\mathbb{R}_{\geq 0}}$; the lemma follows by Theorem 4.

This reduction is the trickiest among those in this paper. The difficulty lies in the fact that, while the finite counter problem refers a specific counter v_0, $BV_{\mathbb{R}_{\geq 0}}$ considers variability of all propositions; while it is easy to reduce from general to specific, here we need to build a reduction in the opposite direction.

The proof of Lemma 8 involves a similar mismatch, but things are simpler there, thanks to the existence of a known bound β, which we can monitor explicitly; now, instead, the bound is existentially quantified. An easy solution would be to change the definition of $BV_{\mathbb{R}_{\geq 0}}$ to refer a specific proposition that varies, but that would weaken the result proved. Instead, we leverage nondeterminism to "guess" the bound.

Build a counter machine M_x which simulates computations of M as follows. Every computation of M_x starts by nondeterministically storing a positive integer x in a fresh counter v_x. This is achieved by the instructions:

$$\begin{aligned} \ell_0 &: \ \textbf{inc}\ v_x \\ \ell_1 &: \ \textbf{if}\ v_x > 0\ \textbf{goto}\ \ell_0, \ell_2 \end{aligned} \tag{2}$$

If ℓ_1's nondeterministic branch eventually jumps to ℓ_2, the rest of M_x's program simulates all computations of M by dovetailing [24, Chap. 3], so that the simulation does not get stuck in non-terminating computations of M. Additionally, whenever v_0 overflows $x-1$, the simulation halts; and if M had only finitely many computations, the simulation concludes with an infinite idle loop (unless it has previously halted upon v_0 overflowing).

Consider now the MTL formula $\Gamma = \Gamma_{M_x} \wedge \Diamond p_h$, where Γ_{M_x} encodes M_x's computations as in Section 4.2, and ℓ_h is the unique halting location of M_x. Thus, the models of Γ describe all valid computations of M_x that halt (and hence, in particular, that do not get stuck forever in the initial loop (2) that increments v_x—something we get for free given that we are reducing between undecidable problems).

Consider now $BV_{\mathbb{R}_{\geq 0}}$ for Γ. If its answer is YES, then there must be only finitely many models that satisfy Γ; otherwise, they would include simulations for all values of x, which would entail that, for every possible bound x, there exists a model where v_0 overflows x, against the hypothesis that the answer is YES (i.e., all models are bounded). Therefore, there is a finite bound on v_0 in all computations of M, given by one plus the maximum of values reached by v_0 in all finitely many models. Conversely, if the answer to $BV_{\mathbb{R}_{\geq 0}}$ for Γ is NO, then there must be infinitely many models that satisfy Γ, that is one for every value of x; in fact, all such models are halting, and hence if they are finitely many the maximum of all counters in all such models would be well defined and finite, against the hypothesis that the answer is NO (i.e., there always is an unbounded model). The existence of halting models for all values of x entails that v_0 overflows any finite value in some computation. In summary, the answer to $BV_{\mathbb{R}_{\geq 0}}$ for Γ is YES iff the answer to the finite counter problem for M is YES. This concludes the reduction. $\square$

5.2 Complexity of Bounded Variability over Discrete Time

The complexity of bounded variability over discrete time paints a picture quite different from that over continuous time. Lemmas 11–12 prove that $BV_{\mathbb{N}}(v, V)$ is **EXPSPACE**-complete, which is the same complexity as MTL satisfiability over $\mathbb{N}$. On the other hand, it is clear that $BV_{\mathbb{N}}$ is decidable in constant time, since every discrete-time MTL formula has variability bounded by x/x for any integer $x > 0$, precisely because the time domain is discrete, and hence there is a hard upper bound on the variability of events.

Lemma 11. *$BV_{\mathbb{N}}(v, V)$ is* **EXPSPACE***-hard.*

Proof. We polynomial-time reduce MTL satisfiability to $\overline{BV}_{\mathbb{N}}(v, V)$; the lemma follows since **EXPSPACE** is closed under complement.

The decision procedure for discrete-time TPTL is based on the following fundamental property [3, Lemma 5]: a TPTL formula ψ is satisfiable iff it has a model where the difference between any pair of consecutive timestamps is always less than or equal to the product δ_ψ of all constants appearing in ψ. The same property holds of MTL formulas over the integers [2].

Since we are considering timed ω-words, which have infinitely many events, the property entails that an MTL formula ϕ is satisfiable iff there exists a timed word ω such that: ω has at least one event with timestamp $t \leq \delta_\phi$ and $\omega \models \phi$. Therefore, a generic MTL formula ϕ is satisfiable iff some of its models have variability not bounded by $0/\delta_\phi$, that is iff the answer to $\overline{BV}_{\mathbb{N}}(0, \delta_\phi)$ is YES. Assuming a binary encoding of constants, as it is customary, δ_ϕ is polynomial in the size of ϕ (because the product of n constants has size $O(n^2)$ in binary), thus the reduction is done in polynomial time. □

Lemma 12. *$BV_{\mathbb{N}}(v, V)$ is in* **EXPSPACE***.*

Proof. We show how to encode the requirement that a model has variability bounded by v/V as an MTL formula $B_{v,V}$.

If $v = 0$, then $B_{v,V} = \Box\bot$. Otherwise, we can adapt the techniques we introduced for LTL [14]. Consider $v > 0$ fresh propositions p_i, for $i = 1, \ldots, v$. Proposition p_1 holds initially, followed by $p_2, \ldots, p_v$ in sequence; the sequence repeats indefinitely:

$$B_v = p_1 \wedge \bigwedge_{1 \leq k \leq v} \left(\Box(p_1 \Leftrightarrow \bigcirc p_{k \oplus 1}) \wedge \Box \left(p_k \Rightarrow \bigwedge_{1 \leq h \neq k \leq v} \neg p_h \right) \right)$$

where $a \oplus b$ is a shorthand for $1 + ((a + b) \bmod v)$. Since every p_k holds in a different position, we can express bounded variability by requiring that the timestamp of the next $(v+1)$-th position in the future be greater than V with respect to the current position's (and note that $k \oplus v = k$):

$$B_{v,V} \quad = \quad B_v \wedge \bigwedge_{1 \leq k \leq v} \Box(p_k \Rightarrow \mathsf{U}_{>V}(\neg p_k, p_k)) \ .$$

Thus, $\phi \Rightarrow B_{v,V}$ is valid iff the answer to $BV_{\mathbb{N}}(v, V)$ for ϕ is YES.

The only problem with this reduction is that $B_{v,V}$ has size exponential in the size of the instance of $BV_{\mathbb{N}}(v, V)$ assuming a binary encoding of constants. Precisely, the blow-up occurs because B_v has size polynomial in v, which is exponential in the size of a binary encoding of v. Encoding the modulo-v counter in binary (using $n = \lfloor \log_2 v \rfloor + 1$ propositions) would not help: while updates to the counter itself can be done with formulas of size polynomial in n, there is no easy way to express in MTL the fact that the timestamp of the "next" occurrence is greater than V (with respect to the current position's) without enumerating all $2^n = v$ values for the counter.

Let us illustrates the problem, assuming for simplicity, but without loss of generality, that $v = 2^n$ for some integer n. Consider n propositions $b_1, \ldots, b_n$ such that a b_k represent the k-th bit of a counter spanning the 2^n values from

0^n to 1^n; and b_n is the most significant bit. To simplify the notation, we write $\neg b_k$ as $\overline{b}_k$, and string such as $b_n \cdots b_1$ represent propositional formulas such as $b_n \wedge \cdots \wedge b_1$. From one position to the next, the counter gets incremented by one. In binary, this is expressed as follows: starting from the least significant bit, flip all 1s until you reach the first 0; flip the 0 as well, and leave all other more significant bits unchanged:

$$\bigwedge_{1 \leq k \leq n} \left(\overline{b}_k b_{k-1} \cdots b_1 \Rightarrow \bigcirc \big(b_k \overline{b}_{k-1} \cdots \overline{b}_1 \big) \wedge \bigwedge_{k < j \leq n} (b_j \Leftrightarrow \bigcirc b_j) \right)$$

plus the special case $b_n \cdots b_1 \Rightarrow \overline{b}_n \cdots \overline{b}_1$ specified separately. This formulas has size $\mathrm{O}(n^2)$, but expressing bounded variability also requires a formula:

$$\Box(x_n \cdots x_1 \Rightarrow \Box_{>V}(x_n \cdots x_1))$$

for each of the 2^n values $x_n \cdots x_1$ of the bits $b_1, \ldots, b_n$.

The blow-up is, however, inessential and only due to the fact that MTL operators do not include compact "counting" modalities. We omit the details for brevity, but it is clear that one can extend the standard decision procedures for MTL [2] to handle counting modalities without affecting the complexity of the logic. Specifically, we could introduce an operator $\mathsf{K}_J^n \psi$ with the semantics: $\omega, k \models \mathsf{K}_J^n \psi$ iff $t_{k+n} - t_k \in J$ and $\omega, k+n \models \psi$. $B_{v,V}$ for $v > 0$ is then equivalent to $\Box\big(\mathsf{K}_{>V}^v \top\big)$; assuming a binary encoding of constants, this has size linear in the size of the encodings of v and V. $\Box$

6 Bounded Variability in Simple Cases

The complexity results of Section 5 pose some major limitations to deciding bounded variability for *generic* MTL formulas. However, the outlook may be better if we target *fragments* of MTL that are still sufficiently expressive but for which reasoning about bounded variability is simpler than in the general case. We call such fragments "bounded friendly". We give two examples of non-trivial bounded-friendly fragments, one for discrete and one for dense time.

Definition 13. An MTL fragment $\mathcal{F}$ is *bounded friendly* over $\mathbb{T}$ iff three conditions hold:

1. We can express in $\mathcal{F}$ a sufficient condition for bounded variability; that is, for any v, V, there exists a computable formula $B_{v,V} \in \mathcal{F}$ such that all models of $B_{v,V}$ have variability bounded by v/V.

Given a generic MTL formula ϕ, we can construct two formulae ψ and ϕ' such that:[4]

2. ϕ is satisfiable iff $\psi \wedge \phi'$ is.

3. There exists a formula $\psi' \in \mathcal{F}$ equivalent to $\psi \Rightarrow B_{v,V}$ and constructable in $\mathrm{O}(b(|\phi|))$.

[4]For consistency, assume all complexities are time complexities.

4. Deciding validity $\psi \Rightarrow B_{v,V}$ is simpler than deciding bounded variability for ϕ; that is, if validity for $\gamma \in \mathcal{F}$ is decidable in $\mathrm{O}(f(|\gamma|))$, and $\psi \Rightarrow B_{v,V}$ is constructable in , then $f(b(x))$ is $\mathrm{o}(m(x))$, where $m(x)$ bounds the complexity of deciding $BV_{\mathbb{T}}(v, V)$.

For a bounded-friendly MTL fragment, we can proceed as follows. Rewrite ϕ into $\psi \wedge \phi'$; construct $\psi \Rightarrow B_{v,V}$ and determine if it is valid; if it is, then all models of ϕ have variability bounded by v/V, since $\phi \Rightarrow \psi$, but we determined it with less computational resources than by analyzing ϕ directly. The challenge in making this process practical is finding sufficiently expressive fragments $\mathcal{F}$, which can represent a "large part" of ϕ, as well as bounded variability itself. The following subsections discuss non-trivial MTL fragments that are also bounded friendly over the integers (Section 6.1) and over the reals (Section 6.2).

6.1 Simpler Bounded Variability over Discrete Time

Over discrete time, MTL essentially boils down to an exponentially succinct version of LTL. Therefore, we can try to lift some complexity results about simpler fragments of LTL [26, 7, 4] to MTL over $\mathbb{N}$, and use them to identify bounded-friendly fragments.

Consider the two dual MTL fragments $\mathcal{F}^{+}_{\Diamond,\bigcirc}$ and $\mathcal{F}^{+}_{\Box,\bigcirc}$: $\mathcal{F}^{+}_{\Diamond,\bigcirc}$ (respectively, $\mathcal{F}^{+}_{\Box,\bigcirc}$) denotes the MTL fragment using only the $\Diamond_J$ (respectively, $\Box_J$) and $\bigcirc_J$ modalities (which we now regard as primitive), the propositional connectives $\wedge$ and $\vee$, and where negations only appear on atomic propositions. Satisfiability for these fragments is decidable in exponential *time*.

Lemma 14. *Satisfiability of $\mathcal{F}^{+}_{\Diamond,\bigcirc}$ and of $\mathcal{F}^{+}_{\Box,\bigcirc}$ over $\mathbb{N}$ is* **EXP***-complete.*

Proof. Consider the LTL fragment $\mathcal{L}^{+}_{\mathsf{F},\mathsf{X}}$ which only uses the eventually and next LTL modalities, the propositional connectives $\wedge$ and $\vee$, and where negations only appear on atomic propositions; [26, Th. 3.7] proves that satisfiability for $\mathcal{L}^{+}_{\mathsf{F},\mathsf{X}}$ is **NP**-complete. We outline how to transform a generic $\mu \in \mathcal{F}^{+}_{\Diamond,\bigcirc}$ into a $\lambda \in \mathcal{L}^{+}_{\mathsf{F},\mathsf{X}}$ such that μ and λ are equisatisfiable; the converse transformation is also derivable along the same lines. In general, the size of λ is exponential in the size of μ due to the fact that metric constraints are encoded in binary in μ. The lemma follows as a manifestation of the "succinctness phenomenon" [24, Chap. 20].

The models of $\mathcal{L}^{+}_{\mathsf{F},\mathsf{X}}$ are denumerable sequences $w = w_0\, w_1 \cdots$ such that w_k is the set of propositions that hold at step k. We conventionally assume that a step in w corresponds to one discrete time instant; thus, a generic $\mathbb{N}$-timed word $\omega = (\sigma_0, t_0)\,(\sigma_1, t_1) \cdots$ uniquely corresponds to a sequence $w = w_0\, w_1 \cdots$ such that: for all $k \in \mathbb{N}$, $w_{t_k} = \sigma_k$; and, for all h's that are not timestamps of ω, $w_h = \{\epsilon\}$, where ϵ is a special proposition denoting absence of a reading.

We define a translation τ from $\mathcal{F}^{+}_{\Diamond,\bigcirc}$ to $\mathcal{L}^{+}_{\mathsf{F},\mathsf{X}}$ inductively as follows, for

$a, b \in \mathbb{N}$ and $c \in \mathbb{N} \cup \{\infty\}$:

$$\begin{aligned}
\tau\left(\Diamond_{[a,b]}(\pi)\right) &= \mathsf{X}_a(\pi_\tau \vee \overbrace{\mathsf{X}(\pi_\tau \vee \cdots)}^{b-a \text{ nested } \mathsf{X}s}) \\
\tau\left(\Diamond_{[a,\infty)}(\pi)\right) &= \mathsf{X}_a\mathsf{F}(\pi_\tau) \\
\tau\left(\bigcirc_{[a,c]}(\pi)\right) &= \overbrace{\mathsf{X}(\epsilon \wedge \mathsf{X}(\epsilon \wedge \cdots))}^{a-1 \text{ nested } \mathsf{X}s} \wedge \tau\left(\Diamond_{[a,c]}(\pi)\right)
\end{aligned}$$

where $\tau(\pi) = \pi_\tau$ and X_k is a shorthand for k nested applications of X. τ does not otherwise change the propositional structure of formulas.

It should be clear that a generic μ is satisfiable over timed words over $\mathbb{N}$ iff $\tau(\mu)$ is satisfiable: models of $\tau(\mu)$ are obtained from models of μ according to the mapping of timed words described above. Furthermore, the size of $\tau(\mu)$ is $O(2^{|\mu|})$, since τ "unrolls" the constants succinctly represented in μ, which results in a possible exponential blow-up. This establishes the lemma for $\mathcal{F}^+_{\Diamond,\bigcirc}$. The same complexity result for $\mathcal{F}^+_{\Box,\bigcirc}$ follows by duality of $\Box$ and $\Diamond$. $\Box$

We can leverage Lemma 14 to show that $\mathcal{F}^+_{\Diamond,\bigcirc}$ is *bounded friendly*. Let v, V be any variability bounds, with $v > 0$ w.l.o.g. First, note that the equivalent $\mathcal{F}^+_{\Box,\bigcirc}$ formulas $\Box\Box_{(0,\nu]}(\bot)$ and $\Box\bigcirc_{>\nu}(\top)$, for $\nu = \lceil V/v \rceil$, hold only for models with variability bounded by v/V (specifically, they are stricter than the definition of bounded variability). Consider now a generic MTL formula ϕ written as $\phi' \wedge \psi$, where $\psi \in \mathcal{F}^+_{\Diamond,\bigcirc}$. The implication $\psi \Rightarrow B_{v,V} \equiv \neg\psi \vee B_{v,V}$, where $B_{v,V}$ is one of the two just defined $\mathcal{F}^+_{\Box,\bigcirc}$ formulas implying bounded variability, is an $\mathcal{F}^+_{\Box,\bigcirc}$ formula: push in the outermost negation $\neg\psi$, and use the duality between $\Diamond$ and $\Box$. The validity of $\psi \Rightarrow B_{v,V}$ can thus be decided in singly exponential time (Lemma 14), as opposed to $BV_{\mathbb{N}}(v, V)$ or the validity of ϕ which are **EXPSPACE**-complete: solving them for a generic ϕ takes *time* doubly exponential in $|\phi|$. We can thus decide whether $\psi \Rightarrow B_{v,V}$ is valid in singly exponential time; if it is, ϕ has bounded variability a fortiori, and hence we can study its validity with the simplified algorithms [14, 15].

We can show by duality that $\mathcal{F}^+_{\Diamond,\bigcirc}$ is also bounded friendly; for example, instead of the validity of $\psi \Rightarrow B_{v,V}$, we equivalently consider the unsatisfiability of $\psi \wedge \neg B_{v,V}$.

6.2 Simpler Bounded Variability over Continuous Time

While MTL is highly undecidable over dense time, a number of expressive yet decidable fragments thereof have been identified. MITL is the fragment of MTL where intervals are non-punctual, that is non-singular; MITL is fully decidable with **EXPSPACE**-complete complexity [1, 16]. More recently, other decidable expressive fragments have been identified that allow singular intervals [23]; BMTL and SMTL, in particular, are interesting because their expressive power is incomparable with MITL's.

From the point of view of deciding bounded variability, however, MITL remains the most suitable choice. SMTL validity has a non-elementary decision problem; while this is still better than the undecidable $BV_{\mathbb{R}_{\geq 0}}(v, V)$, it makes it intractable in practice. BMTL validity, in contrast, is decidable in **EXPSPACE**;

however, BMTL cannot express invariance properties since only finite intervals are allowed, and it is clear that bounded variability is a form of invariance property since it involves whole timed words.

Let us show that MITL is bounded friendly. Let v, V be any variability bounds, with $v > 0$ w.l.o.g. (the limit case $v = 0$ can be handled separately). Note that the MITL formula $B_{v,V} = \Box\Box_{(0,\nu]}(\bot)$, for $\nu = \lceil V/v \rceil$, subsumes variability bounded by v/V. Consider now a generic MTL formula ϕ written as $\phi' \wedge \psi$, where ψ is an MITL formula. The implication $\psi \Rightarrow B_{v,V}$ obviously also is an MITL formula. The validity of $\psi \Rightarrow B_{v,V}$ is thus decidable; if $\psi \Rightarrow B_{v,V}$ is valid, ϕ has bounded variability a fortiori, and hence its validity is decidable [12].

As a final remark, notice how leveraging MITL's bounded friendliness can still be useful to determine the satisfiability of MTL specifications not entirely expressed in MITL: as long as the part ψ expressible in MITL entails bounded variability, the rest ϕ' of the actual specification can use any MTL operator, including singular intervals.

7 Discussion: Other MTL Semantics

Remark 1 clarified that the results of this paper assume infinite timed words, and the continuous semantics over dense time. While these are perfectly common assumptions (and the naturalness of the pointwise semantics over dense time has been questioned [16]), it is still interesting to get an idea of how our results would change under a different semantics.

Over discrete time, it is straightforward to notice that all complexity results proved in 5.2 carry over to the finite-word semantics (where decidability also has the same **EXPSPACE** complexity). Over dense time, it is possible to extend the results of Section 5.1 to the *signal* semantics of [1, 13]—which can be seen as a variant of the continuous semantics—by reusing some of the constructions and definitions of [12].

The situation over dense timed words (both finite and infinite) under the pointwise semantics (and no past operators) is different. For both finite and infinite words, $BV_{\mathbb{R}_{\geq 0}}(v, V)$ is no more difficult than validity, because one can encode the bounded variability requirement as in Lemma 12. Therefore, $BV_{\mathbb{R}_{\geq 0}}(v, V)$ is decidable (nonprimitive recursive) over finite words [22], and is **RE** over infinite words [?]. One can prove matching lower bounds along the lines of the proof of Lemma 11: the abstraction of clock valuations into clock regions used in the construction of [22] is such that the time difference between any pair of consecutive timestamps in any word that satisfies a formula ϕ is bounded above by a finite constant δ_ϕ that depends only on ϕ. Therefore, $\overline{BV}_{\mathbb{R}_{\geq 0}}(0, \delta_\phi)$ for ϕ reduces to non-satisfiability of ϕ, giving the matching lower bounds through complement. Finally, $BV_{\mathbb{R}_{\geq 0}}$ is **RE** under the pointwise semantics: by definition of **RE** as existential quantification over a recursive relation (finite words); and by dovetailing through the possible values v, V and enumerating $BV_{\mathbb{R}_{\geq 0}}(v, V)$ for them (infinite words). Finding matching lower bounds belongs to future work, which may exploit the connection between MTL over infinite words under the pointwise semantics and channel machines with insertion errors [?].

8 Conclusions

The strong negative results of the paper need not be the deathblow to leveraging bounded variability to simplify temporal reasoning. From a broader perspective, we can still look at the glass as half-full: while deciding bounded variability is intractable in general, there are situations where the physical requirements of a system include a notion of *finite speed*, which bounded variability naturally embodies. For such systems, there is still hope of using the expressiveness of MTL without succumbing to the dark side of intractability.

Acknowledgements. Thanks to the anonymous reviewers of TIME 2014 for useful comments; in particular, reviewer 4's insightful observations helped simplify some of the proofs.

References

[1] Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43(1):116–146, 1996.

[2] Rajeev Alur and Thomas A. Henzinger. Real-time logics: Complexity and expressiveness. *Inf. Comp.*, 104(1):35–77, 1993.

[3] Rajeev Alur and Thomas A. Henzinger. A really temporal logic. *Journal of the ACM*, 41(1):181–204, 1994.

[4] Michael Bauland, Thomas Schneider, Henning Schnoor, Ilka Schnoor, and Heribert Vollmer. The complexity of generalized satisfiability for linear temporal logic. *Logical Methods in Computer Science*, 5(1), 2009.

[5] Patricia Bouyer, Nicolas Markey, Joël Ouaknine, and James Worrell. The cost of punctuality. In *LICS*, pages 109–120, 2007.

[6] Davide Bresolin, Dario Della Monica, Valentin Goranko, Angelo Montanari, and Guido Sciavicco. The dark side of interval temporal logic: Sharpening the undecidability border. In *TIME*, pages 131–138. IEEE, 2011.

[7] Stéphane Demri and Ph. Schnoebelen. The complexity of propositional linear temporal logics in simple cases. *Inf. Comput.*, 174(1):84–103, 2002.

[8] Deepak D'Souza and Pavithra Prabhakar. On the expressiveness of MTL in the pointwise and continuous semantics. *STTT*, 9(1):1–4, 2007.

[9] E. Allen Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science*, pages 996–1072. Elsevier, 1990.

[10] Martin Fränzle. Model-checking dense-time duration calculus. *Formal Asp. Comput.*, 16(2):121–139, 2004.

[11] Carlo A. Furia, Dino Mandrioli, Angelo Morzenti, and Matteo Rossi. *Modeling Time in Computing.* Monographs in Theoretical Computer Science. An EATCS series. Springer, 2012.

[12] Carlo A. Furia and Matteo Rossi. MTL with bounded variability: Decidability and complexity. In *FORMATS*, volume 5215 of *LNCS*, pages 109–123. Springer, 2008.

[13] Carlo A. Furia and Matteo Rossi. A theory of sampling for continuous-time metric temporal logic. *ACM Transactions on Computational Logic*, 12(1):1–40, 2010. Article 8.

[14] Carlo A. Furia and Paola Spoletini. On relaxing metric information in linear temporal logic. In *TIME*, pages 72–79. IEEE, 2011.

[15] Carlo A. Furia and Paola Spoletini. Automata-based verification of linear temporal logic models with bounded variability. In *TIME*, pages 89–96. IEEE, 2012.

[16] Yoram Hirshfeld and Alexander Moshe Rabinovich. Logics for real time: Decidability and complexity. *Fundam. Inform.*, 62(1):1–28, 2004.

[17] Paul Hunter, Joël Ouaknine, and James Worrell. Expressive completeness for metric temporal logic. In *Proceedings of LICS*. IEEE, 2013.

[18] Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

[19] Carsten Lutz, Dirk Walther, and Frank Wolter. Quantitative temporal logics over the reals: PSPACE and below. *Inf. Comput.*, 205(1):99–123, 2007.

[20] Oded Maler, Dejan Nickovic, and Amir Pnueli. Real time temporal logic: Past, present, future. In *FORMATS*, volume 3829 of *LNCS*, pages 2–16. Springer, 2005.

[21] Marvin Lee Minsky. *Computation: Finite and Infinite Machines.* Prentice Hall, 1967.

[22] Joël Ouaknine and James Worrell. On the decidability and complexity of metric temporal logic over finite words. *Logical Methods in Computer Science*, 3(1), 2007.

[23] Joël Ouaknine and James Worrell. Some recent results in metric temporal logic. In *FORMATS*, volume 5215 of *LNCS*, pages 1–13. Springer, 2008.

[24] Christos Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[25] Hartley Rogers, Jr. *Theory of Recursive Functions and Effective Computability.* MIT Press, 1987.

[26] A. Prasad Sistla and Edmund M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985.

[27] Thomas Wilke. Specifying timed state sequences in powerful decidable logics and timed automata. In *FTRTFT*, volume 863 of *LNCS*, pages 694–715. Springer, 1994.