Tomorrow and All Our Yesterdays: MTL Satisfiability over the Integers

Carlo A. Furia[1] and Paola Spoletini[2]

[1] DEI, Politecnico di Milano, Milano, Italy
[2] DSCPI, Università degli Studi dell'Insubria, Como, Italy

Abstract. We investigate the satisfiability problem for metric temporal logic (MTL) with both past and future operators over linear discrete bi-infinite time models isomorphic to the integer numbers, where time is unbounded both in the future and in the past. We provide a technique to reduce satisfiability over the integers to satisfiability over the well-known mono-infinite time model of natural numbers, and we show how to implement the technique through an automata-theoretic approach. We also prove that MTL satisfiability over the integers is EXPSPACE-complete, hence the given algorithm is optimal in the worst case.

1 Introduction

Temporal logic has become a very widespread notation for the formal specification of systems, temporal properties, and requirements. Its popularity is significantly due to the fact that it provides highly effective conceptual tools to model, specify, and reason about systems [7], and it is amenable to fully automated verification techniques, the most notable being model-checking [4].

In temporal logic frameworks it is customary to model time as infinite in the future and finite in the past, i.e., with an origin; in other words, time is mono-infinite. On the contrary, models where time is infinite both in the future and in the past — i.e., it is *bi-infinite* [12] — have been routinely neglected. The reasons for this strong preference are mainly historical, as it has been pointed out by various authors [7, 13]. Namely, temporal logic has been originally introduced for the purpose of reasoning about the behavior of "ongoing concurrent programs" [7], hence a model of time with an origin is appropriate since "computation begins at an initial state" [7]. However, there are various motivations in favor of the adoption of bi-infinite time models [13] as well, and they go beyond the obvious theoretical interest.

The first of such reasons has to do with the usage of temporal logics with operators that reference to the past of the current instant. If past is bounded, we may have to deal with past operators referring to instants that are before the origin of time: this gives rise to so-called *border effects* [5]. For instance, consider *yesterday* operator Y of LTL[3]: $\mathsf{Y}p$ evaluates to true at some instant t if and only if its argument p holds at the previous instant $t-1$. Then, consider

[3] Throughout the paper we assume temporal logics with past operators.

formula `Yalarm` which models an alarm being raised at the previous instant. If we evaluate the formula at the origin, the reference to the "previous" instant of time is moot as there is no such instant, and whether the evaluation should default to true or to false depends on the role the formula plays in the whole specification. A possible solution to these problems is to introduce two variants of every past operator, one defaulting to true and the other to false [5]; however, this is often complicated and cumbersome, especially in practical applications. On the contrary, the adoption of bi-infinite time gets rid of such border effects single-handedly, in a very uniform and natural manner, because there are simply no "inaccessible" instants of time.

The second main motivation for considering bi-infinite time models is derived from a reason for adopting mono-infinite time models: the fact that ongoing non-terminating processes are considered. Similarly, when modeling processes that are "time invariant" (whose behavior does not depend on *absolute* time values) and where initialization can be abstracted away, a time model which is infinite both in the past and in the future is the most natural and terse assumption.

This paper investigates temporal logic over bi-infinite discrete-time models. More precisely, we consider a linear-time model which is isomorphic to the integer numbers. Correspondingly, Metric Temporal Logic (MTL) [1] is taken as temporal logic notation. It will be clear that, over the adopted discrete-time model, MTL boils down to LTL with a succinct encoding of constants in formulas. Hence, our results will be easily stateable in terms of LTL as well. The main contributions are as follows. First, we present a general technique to reduce the satisfiability problem for MTL over the integers to the same problem over the more familiar mono-infinite time model isomorphic to the natural numbers. Second, we show how the technique can be practically implemented with an automata-theoretic approach — derived from previous work of ours [15] — which can work on top of the Spin model-checker [10]. Third, the complexity of the MTL satisfiability problem over the integer is assessed, and it is shown that, unsurprisingly, it matches the well-known upper and lower bounds for the same problem over mono-infinite discrete time domain [1]. To the best of our knowledge, this is the first work which analyzes the complexity of MTL (and LTL) satisfiability over bi-infinite time and provides a practical algorithm for it.

For the sake of space limits, we omit some proofs and inessential details, while providing some intuitive examples. Missing details can be found in [8].

2 Definitions and Preliminaries

The symbols $\mathbb{Z}$ and $\mathbb{N}$ denote respectively the set of integer numbers and the set of nonnegative integers. For greater clarity, connectives and quantifiers of the meta-language are typeset in a bold underlined font.

2.1 Metric Temporal Logic

We define Metric Temporal Logic (MTL) [1] over mono-infinite and bi-infinite linear discrete time. We always consider the variant with both past and future operators (called $\mathrm{MTL}^{\mathrm{P}}$ by some authors [1]).

Syntax. Let $\Pi = \{p, q, \ldots\}$ be a finite set of propositions. MTL formulas are given by $\phi ::= p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \,\mathsf{U}_I\, \phi_2 \mid \phi_1 \,\mathsf{S}_I\, \phi_2$, where $p \in \Pi$, I is an interval of the naturals (possibly unbounded to the right), and the symbols U_I, S_I denote the bounded *until* and *since* operator, respectively.

Standard abbreviations are assumed such as $\top, \bot, \vee, \Rightarrow, \Leftrightarrow$. In addition, we introduce some useful derived temporal operators: *eventually* $\mathsf{F}_I\phi = \top \,\mathsf{U}_I\, \phi$; *always* $\mathsf{G}_I\phi = \neg\mathsf{F}_I\neg\phi$; *next* $\mathsf{X}\phi = \bot \,\mathsf{U}\, \phi$; *release* $\phi_1 \,\mathsf{R}_I\, \phi_2 = \neg(\neg\phi_1 \,\mathsf{U}_I\, \neg\phi_2)$. Each of these operators has its past counterpart; that is, respectively: eventually in the *past* $\mathsf{P}_I\phi = \top \,\mathsf{S}_I\, \phi$; *historically* $\mathsf{H}_I\phi = \neg\mathsf{P}_I\neg\phi$; *previous* or *yesterday* $\mathsf{Y}_k\phi = \mathsf{P}_{[k,k]}\phi$; *trigger* $\phi_1 \mathsf{T}_I\, \phi_2 = \neg(\neg\phi_1 \mathsf{S}_I\, \neg\phi_2)$. Note that, whenever no interval is specified, $I = (0, \infty)$ is assumed for all operators; also, the singleton interval $[k, k]$ is abbreviated by $= k$.

Precedence of operators is defined as follows: $\neg$ has the highest binding power, then we have the temporal modalities U_I, S_I and derived ones, then $\wedge$ and $\vee$, $\Rightarrow$, and finally $\Leftrightarrow$. $\widetilde{\phi}$ denotes the formula obtained from ϕ by switching every future operator with its past counterpart, and *vice versa*.

The size $|\phi|$ of a formula ϕ is given by the product of its number of connectives $|\phi|_{\#}$ times the size $|\phi|_{\mathrm{M}}$ of the largest constant used in its formulas, succinctly encoded in binary. A *future* formula ϕ is a formula which does not use any past operator; conversely, a *past* formula π is a formula which does not use any future operator. A formula ψ is *flat* if it does not nest temporal operators.[4] A flat formula is *propositional* if it does not use temporal operators at all.

Words and operations on them. For a finite alphabet Σ, we introduce the sets of right-infinite words (called ω-words), of left-infinite words (called $\widetilde{\omega}$-words), and of bi-infinite words (called $\mathbb{Z}$-words) over Σ, and we denote them as Σ^{ω}, ${}^{\omega}\Sigma$, and $\Sigma^{\mathbb{Z}}$, respectively. Correspondingly, an ω-language (resp. $\widetilde{\omega}$-language, $\mathbb{Z}$-language) is a subset of Σ^{ω} (resp. ${}^{\omega}\Sigma$, $\Sigma^{\mathbb{Z}}$).

Given an ω-word $w = w_0 w_1 w_2 \cdots$, $\widetilde{w}$ denotes the $\widetilde{\omega}$-word $\cdots w_{-2} w_{-1} w_0$ defined by the bijection $w_{-k} = w_k$ for $k \in \mathbb{N}$. The same notation is used for the inverse mapping from $\widetilde{\omega}$-words to ω-words. The mapping is also extended to languages as obvious, with the same notation. Given a $\mathbb{Z}$-word $x = \cdots x_{-2} x_{-1} x_0 x_1 x_2 \cdots$ and $k \in \mathbb{Z}$, x^k denotes the ω-word obtained by truncating x at x_k on the left, i.e., $x^k = x_k x_{k+1} x_{k+2} \cdots$; similarly, ${}^k x$ denotes the $\widetilde{\omega}$-word obtained by truncating x at x_k on the right, i.e., ${}^k x = \cdots x_{k-2} x_{k-1} x_k$.

The operations of intersection ($\cap$), union ($\cup$), and concatenation ($.$) for words and languages are defined as usual. Let w and $\overline{w}$ be an ω- and an $\widetilde{\omega}$-word, respectively. The $\mathbb{Z}$-word $\overline{w} \triangleright w$ (*right join*) is defined as ${}^{-1}\overline{w}.w$, and the $\mathbb{Z}$-word $\overline{w} \triangleleft w$ (*left join*) is defined as $\overline{w}.w^1$. The join operations are extended to

[4] In the literature, there exist also different definitions of flatness, e.g., [3].

languages as obvious, with the same notation. Also, $\downarrow^{\Sigma}$ denotes the projection homomorphism over Σ.

Semantics. We define the semantics of MTL formulas for infinite words over 2^{Π}, where Π is a finite set of atomic propositions. As it is standard, every letter $y_k \in 2^{\Pi}$ in such words represents the set of atomic propositions that are true at integer time instant k (also called *position*). We introduce the predicate $\mathbf{valid}(y, i)$ which holds iff i is a valid position in the infinite word y, i.e., iff y is a $\mathbb{Z}$-word and $i \in \mathbb{Z}$, or y is an ω-word and $i \in \mathbb{N}$, or y is an $\widetilde{\omega}$-word and $-i \in \mathbb{N}$.

Let ϕ be an MTL formula, y a generic infinite word over 2^{Π}, and i an integer such that $\mathbf{valid}(y, i)$. The satisfaction relation $\models$ is defined inductively as:

$$
\begin{array}{ll}
y, i \models p & \underline{\Leftrightarrow}\ p \in y_i \\
y, i \models \neg\phi & \underline{\Leftrightarrow}\ y, i \not\models \phi \\
y, i \models \phi_1 \wedge \phi_2 & \underline{\Leftrightarrow}\ y, i \models \phi_1 \ \underline{\wedge}\ y, i \models \phi_2 \\
y, i \models \phi_1 \,\mathsf{U}_I\, \phi_2 & \underline{\Leftrightarrow}\ \underline{\exists} d \in I\colon (\mathbf{valid}(y, i+d) \ \underline{\wedge} \\
 & \qquad y, i+d \models \psi_2 \underline{\wedge} \underline{\forall} 0 < u < d : y, i+u \models \psi_1) \\
y, i \models \phi_1 \,\mathsf{S}_I\, \phi_2 & \underline{\Leftrightarrow}\ \underline{\exists} d \in I\colon (\mathbf{valid}(y, i-d) \ \underline{\wedge} \\
 & \qquad y, i-d \models \psi_2 \underline{\wedge} \underline{\forall} 0 < u < d : y, i-u \models \psi_1) \\
y \models \phi & \underline{\Leftrightarrow}\ \underline{\forall} i \in \mathbb{Z} : (\mathbf{valid}(y, i) \ \underline{\Rightarrow}\ y, i \models \phi)
\end{array}
$$

Note that we defined $y \models \phi$ to denote "global satisfiability", i.e., the fact that ϕ holds at all valid positions of y. This definition is especially natural over bi-infinite words, where there is no initial instant at which to evaluate formulas. On the contrary, "initial satisfiability" is more common over mono-infinite words where an origin is unambiguously fixed. However, the global satisfiability problem is easily reducible to the initial satisfiability problem, as $\underline{\forall} i : y, i \models \phi$ iff $y, 0 \models \mathrm{Alw}(\phi)$, where $\mathrm{Alw}(\phi) \equiv \mathsf{G}\phi \wedge \phi \wedge \mathsf{H}\phi$ denotes that ϕ holds always.

For instance, consider formula $\nu = \mathsf{H}_{[0,3]} p$ and its interpretation over ω-word w^+ in Figure 1. According to the semantics defined above, ν is true at 1 because p holds for all *valid* positions between 1 and $1 - 3 = -2$. However, there may be justifications in favor of evaluating ν false at 1: there is no complete interval of size 4 where p holds continuously. This is an example of so-called *border effect*: what is a "reasonable" evaluation of formulas near the origin is influenced by the role the formulas play in a specification.

There is an interesting relation between the reverse $\widetilde{\phi}$ of a formula ϕ and the reverse $\widetilde{w}$ of ω-words w that are models of ϕ, as the following example shows. Consider formula $\theta = \mathsf{H}_{[0,3]} p \Rightarrow \mathsf{F} q$ and its reverse $\widetilde{\theta} = \mathsf{G}_{[0,3]} p \Rightarrow \mathsf{P} q$. θ asserts that whenever p held continuously for 4 time units, q must hold somewhere in the future (excluding the current instant), hence θ is true at position 4 and false at position 11 over ω-word w^+ in Figure 1. If we consider $\widetilde{\omega}$-word w^- obtained by reversing w^+ (also in Figure 1), we see that $\widetilde{\theta}$ is true at position -4 and false at position -11 over w^-.

By generalizing the example, we have the following proposition.

Proposition 1. *Let $w^+ \in \left(2^{\Sigma}\right)^{\omega}$ be an ω-word, ϕ be an MTL formula, and $i \in \mathbb{N}$. Then $w^+, i \models \phi$ iff $\widetilde{w^+}, -i \models \widetilde{\phi}$.*

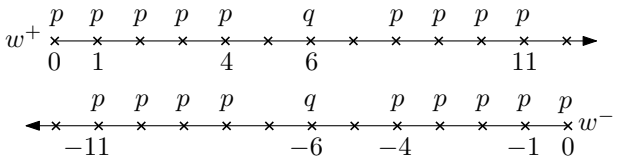

Fig. 1. ω-word w^+ (above) and its reverse $\widetilde{\omega}$-word w^- (below).

Satisfiability and language of a formula. Satisfiability is the following problem: "given a formula ϕ is there some word y such that $y \models \phi$?". It is the verification problem we consider in this paper. For an MTL formula ϕ, let $\mathcal{L}_0^\omega(\phi)$ denote the set of ω-words w such that $w, 0 \models \phi$, let $\mathcal{L}^\omega(\phi)$ denote the set of ω-words w such that $w \models \phi$, and let $\mathcal{L}^{\mathbb{Z}}(\phi)$ denote the set of $\mathbb{Z}$-words x such that $x \models \phi$. Then, the satisfiability problem for a formula ϕ is equivalent to the emptiness problem for the corresponding language.

LTL and expressiveness. LTL is a well-known linear temporal logic based on the unique modality U. We will consider the past-enhanced variant of the logic, and call it simply LTL. For the time models we consider in this paper, MTL is simply LTL with an exponentially succinct encoding (see [8] for a translation): every MTL formula μ can be translated into an LTL formula λ_μ such that $|\lambda_\mu| = |\lambda_\mu|_\# = \exp \mathrm{O}(|\mu|_\# \, |\mu|_{\mathrm{M}})$.

2.2 Automata over Infinite Words

Languages definable in MTL can also be described as languages accepted by finite state automata such as Büchi automata (BA) [16]. The size $|\mathcal{A}|$ of a BA $\mathcal{A}$ is defined as the number of its finite states.

Alternating automata (AA, [17]) are an equally expressive but possibly more concise version of BA. AA have two kinds of transitions: nondeterministic transitions (also called existential, corresponding to $\vee$) just like vanilla BA, and parallel transitions (also called universal, corresponding to $\wedge$). Alternation can represent concisely the structure of an LTL formula [17], avoiding the exponential blow-up. In [15] we introduced Alternating Modulo Counting Automata (AMCA), an enriched variant of AA which makes use of (bounded) counters; this new feature can represent succinctly MTL formulas as well, i.e., it can encode succinctly constants used in MTL modalities.

Definition 1 (Alternating Modulo Counting Automaton (AMCA) [15]). *An Alternating Modulo Counting Automaton is a tuple $\langle \Sigma, Q, \mu, q_0, \delta, F \rangle$ where:*

- *Σ is a finite alphabet,*
- *Q is a set of states,*
- *$\mu \in \mathbb{N}_{\geq 1}$ such that $C = [0..\mu]$ denotes a modular finite counter,*
- *$q_0 \in Q$ is the initial state,*

– $\delta : Q \times C \times \Sigma \to \mathbb{B}^+(Q \times C)$ *is the transition relation,*[5]
– $F \subseteq Q$ *is a set of accepting states.*

For the sake of readability when indicating the elements in $\mathbb{B}^+(Q \times C)$ *we will use the symbol / to separate the component in* Q *from the component in* C*.*

A run of an AMCA is defined as follows.

Definition 2 (Run of an AMCA). *A run* (T, ρ) *of an AMCA* $\mathcal{A}$ *on the* ω*-word* $w = w_0 w_1 \cdots \in \Sigma^\omega$ *is a* $(Q \times C \times \mathbb{N})$*-labeled tree, where* ρ *is the labeling function defined as:* $\rho(\epsilon) = (q_0/0, 0)$*; for all* $x \in T$*,* $\rho(x) = (q/k, n)$*; and the set* $\{(q'/h, 1) \mid c \in \mathbb{N}, x.c \in T, h \in C, \rho(x.c) = (q'/h, n+1)\}$ *satisfies the formula* $\delta(q/k, w_n)$*.*

The acceptance condition for AMCA is defined similarly as for regular BA: a path is accepting iff it passes infinitely many times on at least one state in F*. Formally, for a sequence* $P \in \mathbb{N}^\omega$ *and a labeling function* ρ*, let* $\inf(\rho, P) = \{s \mid \rho(n) \in \{s\} \times \mathbb{N}$ *for infinitely many* $n \in P\}$*. A run* (T, ρ) *of an AMCA is* accepting *iff for all paths* P *of* T *it is* $\inf(\rho, P)|_Q \cap F \neq \emptyset$*.*

The size $|\mathcal{A}|$ of an AMCA $\mathcal{A}$ can be defined as the product of $|Q|$ times the size of the counter, succinctly encoded in binary: $|\mathcal{A}| = \mathrm{O}(|Q| \log \mu)$. With the usual notation, $\mathcal{L}^\omega(\mathcal{A})$ denotes the set of all ω-words accepted by an automaton $\mathcal{A}$.

3 Automata-Based MTL Satisfiability over the Naturals

A widespread approach to testing the satisfiability of an MTL (or LTL) formula over mono-infinite time models isomorphic to the natural numbers relies on the well-known tight relationship between LTL and finite state automata. In order to test the satisfiability of an MTL formula μ, one translates it into an LTL formula λ_μ, and then builds a nondeterministic BA $\mathcal{A}_{\lambda_\mu}$ that accepts precisely the models of λ_μ, hence of μ. Correspondingly, an emptiness test on $\mathcal{A}_{\lambda_\mu}$ is equivalent to a satisfiability check of μ. This procedure, very informally presented, relies on the following two well-known results.

Proposition 2 ([17]). *(1) The emptiness problem for (nondeterministic) BA of size* n *is decidable in time* $\mathrm{O}(n)$ *and space* $\mathrm{O}(\log^2 n)$*. (2) Given an LTL formula* ϕ*, one can build a (nondeterministic) BA* $\mathcal{A}_\phi$ *with* $|\mathcal{A}_\phi| = \exp \mathrm{O}(|\phi|)$ *such that* $\mathcal{L}^\omega(\mathcal{A}_\phi) = \mathcal{L}^\omega(\phi)$ *and* $\mathcal{L}_0^\omega(\mathcal{A}_\phi) = \mathcal{L}_0^\omega(\phi)$*.*

In practice, however, this unoptimized approach is inconvenient, because the BA representing an MTL formula is in general doubly-exponential in the size of the formula, hence algorithmically very inefficient. On the contrary, we would like to exploit more concise classes of automata (such as AMCA) to represent MTL formulas more efficiently in practice. With this aim, in [15] we proposed a novel approach to model-checking and satisfiability checking over discrete mono-infinite time domains for a propositional subset of the TRIO metric temporal

[5] $\mathbb{B}^+(S)$ denotes the set of all *positive* Boolean combinations of elements in S.

logic. It is clear that the subset of TRIO considered in [15] corresponds to MTL as we defined it in this paper. Hence, let us recall from [15] the following result about the translation of MTL formulas in BA and AMCA over the naturals.

Proposition 3 ([15]). *Given a past MTL formula π, one can build two deterministic BA $\mathcal{A}_{\pi^\omega}$ and $\mathcal{A}_{\pi^0}$ such that $\mathcal{L}^\omega(\mathcal{A}_{\pi^0}) = \mathcal{L}_0^\omega(\pi)$, $\mathcal{L}^\omega(\mathcal{A}_{\pi^\omega}) = \mathcal{L}^\omega(\pi)$, and the size of both $\mathcal{A}_{\pi^0}$ and $\mathcal{A}_{\pi^\omega}$ is $\exp \mathrm{O}(|\pi|)$. Given a future MTL formula φ, one can build two AMCA $\mathcal{A}_{\varphi^\omega}$ and $\mathcal{A}_{\varphi^0}$ such that $\mathcal{L}^\omega\big(\mathcal{A}_{\varphi^0}\big) = \mathcal{L}_0^\omega(\varphi)$, $\mathcal{L}^\omega(\mathcal{A}_{\varphi^\omega}) = \mathcal{L}^\omega(\varphi)$, and the size of both $\mathcal{A}_{\varphi^0}$ and $\mathcal{A}_{\varphi^\omega}$ is $\mathrm{O}(|\varphi|)$.*

More precisely, future formulas are translated into AMCA according to the following schema: the AMCA for a future formula φ over alphabet Π is $\mathcal{A}_\varphi = \langle \Sigma, Q, \mu, q_0, \delta, F \rangle$ where:

- $\Sigma = 2^\Pi$,
- $Q = \{\nu \mid \nu \text{ is a subformula of } \varphi\} \cup \{\neg\nu \mid \nu \text{ is a subformula of } \varphi\}$,
- $\mu = |\varphi|_{\mathrm{M}}$,
- $q_0 = \varphi$,
- the transition relation δ is defined as follows:
  - $\delta(\chi/0, p) = \top/0$ for $\chi \in \Pi$ and $\chi = p$,
  - $\delta(\chi/0, p) = \bot/0$ for $\chi \in \Pi$ and $\chi \neq p$,
  - $\delta(\psi \wedge \upsilon/0, p) = \delta(\psi/0, p) \wedge \delta(\upsilon/0, p)$,
  - $\delta(\neg\psi/0, p) = \mathsf{dual}(\delta(\psi/0, p))$, where $\mathsf{dual}(\phi)$ is a formula obtained from ϕ by switching $\top$ and $\bot$, $\wedge$ and $\vee$, and by complementing all subformulas of ϕ,
  - $\delta(\psi \,\mathsf{U}_{[a,b]}\, \upsilon/k, p) =$
    $$\begin{cases} \psi \,\mathsf{U}_{[a,b]}\, \upsilon/k+1 & k = 0 \\ \delta(\psi/0, p) \wedge \Big(\psi \,\mathsf{U}_{[a,b]}\, \upsilon/k+1\Big) & 0 < k < a \\ \delta(\upsilon/0, p) \vee \Big(\delta(\psi/0, p) \wedge \Big(\psi \,\mathsf{U}_{[a,b]}\, \upsilon/k+1\Big)\Big) & a \leq k \leq b \\ \bot & k > b \end{cases}$$
    for $a \leq b < \infty$,
  - $\delta(\psi \,\mathsf{U}\, \upsilon/k, p) = \delta(\upsilon/0, p) \vee (\delta(\psi/0, p) \wedge (\psi \,\mathsf{U}\, \upsilon/0))$,
- $F = \{\xi \mid \xi \in Q \text{ and } \xi \text{ has the form } \neg(\psi \,\mathsf{U}\, \upsilon)\}$

In the remainder we will show how to exploit such satisfiability checking procedures over the naturals to perform satisfiability checking over the integers.

4 Automata-Based MTL Satisfiability over the Integers

This section presents the main contribution of the paper: a technique to reduce the satisfiability problem for MTL formulas over the integers to the same problem over the naturals, and an automata-based implementation thereof.

Flat normal form. We introduce a suitable normal form where each application of temporal operators can be analyzed in isolation, and we show that any MTL formula can be rendered into this normal form by introducing auxiliary atomic proposition but without changing the asymptotic size of the formula.

An MTL formula ϕ is in *flat normal form* when it is written as:[6] $\phi' = \beta \wedge \bigwedge_{k=1}^{n} \mathrm{Alw}(p_k \Leftrightarrow \psi_k)$, where $\beta \in \mathbb{B}(\Pi)$ and ψ_k is a flat formula, for all $k = 1, \ldots, n$. In addition, if every ψ_k is a pure past formula or a pure future formula, ϕ' is named flat *separated* normal form (FSNF).

Theorem 1. *Let ϕ be an MTL formula over Π; a ϕ' in FSNF can be built efficiently such that $\mathcal{L}^{\mathbb{Z}}(\phi) = {\downarrow^{\Pi}}\mathcal{L}^{\mathbb{Z}}(\phi')$, $|\phi'|_{\mathrm{M}} = |\phi|_{\mathrm{M}}$, and $|\phi'|_{\#} = \mathrm{O}(|\phi|_{\#})$.*

For example, considering formula $\theta = \mathsf{H}_{[0,3]}p \Rightarrow \mathsf{F}q$, we can build θ' by replacing $\mathsf{H}_{[0,3]}p$ and $\mathsf{F}q$ with two new Boolean literals p' and q' respectively. Hence, $\theta' = (p' \Rightarrow q') \wedge \mathrm{Alw}\Big(p' \Leftrightarrow \mathsf{H}_{[0,3]}p\Big) \wedge \mathrm{Alw}(q' \Leftrightarrow \mathsf{F}q)$.

4.1 Splitting the Evaluation about the Origin

Let ϕ' be an MTL formula in FSNF. The satisfiability of ϕ' can be analyzed by considering each of the $n+1$ subformulas β, $p_k \Leftrightarrow \psi_k|_{1 \le k \le n}$ separately. In fact, $x \models \phi'$ iff $x \models \beta$ and $\underline{\forall} k = 1, \ldots, n : x \models p_k \Leftrightarrow \psi_k$. Hence, without loss of generality, we focus on studying the satisfiability of formulas in the form β, $p \Leftrightarrow \psi^{+}$, and $p \Leftrightarrow \psi^{-}$, where ψ^{+} and ψ^{-} are flat *until* and *since* formulas, respectively.

More precisely, let us start with the future formula: $\psi = f \Leftrightarrow p\,\mathsf{U}_I\,q \equiv (\neg f \vee p\,\mathsf{U}_I\,q) \wedge (f \vee \neg p\,\mathsf{R}_I\,\neg q)$ In turn, $x \models \psi$ iff $x \models \neg f \vee p\,\mathsf{U}_I\,q$ and $x \models f \vee \neg p\,\mathsf{R}_I\,\neg q$. Correspondingly, we now focus on studying the satisfiability of the simple formula $\neg f \vee p\,\mathsf{U}_I\,q$ over the integers. Then it will be straightforward to extend it to handle the other formula $f \vee \neg p\,\mathsf{R}_I\,\neg q$, as well as the corresponding past formula $f \Leftrightarrow p\,\mathsf{S}_I\,q$.

Behavior about the origin. Let us consider a $\mathbb{Z}$-word x such that $x \models p\,\mathsf{U}_{[l,u]}\,q$ for some $0 \le l \le u < \infty$. We aim at splitting the evaluation of $x \models p\,\mathsf{U}_{[l,u]}\,q$ into the evaluation of other related formulas over mono-infinite words x^0 and ${}^0x$.

Before introducing the formal results, let us provide some intuition about our technique, and let $l = 3, u = 7$. First of all, $x \models p\,\mathsf{U}_{[3,7]}\,q$ requires in particular that $x^0 \models p\,\mathsf{U}_{[3,7]}\,q$: *until* is a future operator, thus its evaluation over x^0 is independent of all instant before the origin, hence $x^0 \models p\,\mathsf{U}_{[3,7]}\,q$ iff $\underline{\forall} k \ge 0 : x, k \models p\,\mathsf{U}_{[3,7]}\,q$. For instance this is the case of instant 3 in Figure 2. Similarly, let us consider any position k of x such that the interval $(k, k+7] \subset (-\infty, 0]$ is contained completely to the left of the origin, such as position -8 in Figure 2. The evaluation of $p\,\mathsf{U}_{[3,7]}\,q$ at k is independent of all instants after the origin, hence $x, k \models p\,\mathsf{U}_{[3,7]}\,q$ iff ${}^0x, k \models p\,\mathsf{U}_{[3,7]}\,q$, for all $k + 7 \le 0$, i.e., $k \le -7$. Finally,

[6] $\mathbb{B}(\Pi)$ denotes the set of all Boolean combinations of elements in Π.

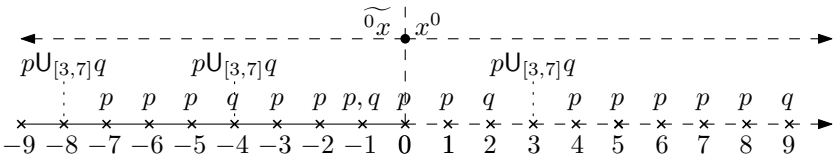

Fig. 2. Splitting the evaluation of $p\,\mathsf{U}_{[3,7]}\,q$ about the origin.

let us consider what happens to the evaluation of $p\,\mathsf{U}_{[3,7]}\,q$ at instants k such that the interval $(k, k+7] \ni 0$ contains the origin; for instance let $k = -4$ and consider again Figure 2. Hence, there exists a $h \in [-1, 3]$ such that $x, h \models q$ and for all $-4 < j < h$ it is $x, j \models p$. Here, we have to distinguish two cases and handle them differently. If $h \leq 0$ such as for $h = -1$ in Figure 2, the evaluation of $p\,\mathsf{U}_{[3,7]}\,q$ at -4 is still independent of instants after the origin, hence $x, k \models p\,\mathsf{U}_{[3,7]}\,q$ iff ${}^0x, k \models p\,\mathsf{U}_{[3,7]}\,q$. Otherwise, if $h > 0$ such as for $h = 2$ in Figure 2, we consider separately the adjacent intervals $(k, 0]$ and $(0, k+7]$. The fact that p holds throughout $(k, 0]$ is independent of instants after the origin, so $x, k \models \mathsf{G}_{(0,-k]}p$ iff ${}^0x, k \models \mathsf{G}p$. Moreover, $p\,\mathsf{U}_I\,q$ holds at the origin for the "residual" interval $(0, 3]$, thus $x, 0 \models p\,\mathsf{U}_{[1,3]}\,q$ iff $x^0, 0 \models p\,\mathsf{U}_{[1,3]}\,q$.

By generalizing the above informal reasoning, we get the following.

Lemma 1. *For $x \in (2^{\Pi})^{\mathbb{Z}}$, $0 \leq l \leq u < \infty$ such that $u \neq 0$,*[7] *$1 - u \leq i \leq -1$:*

$$x, i \models p\,\mathsf{U}_{[l,u]}\,q \quad \Leftrightarrow \quad \begin{array}{c} x, i \models p\,\mathsf{U}_{[l,-i]}\,q \\ \veebar \\ \Big(x, i \models \mathsf{G}_{[1,-i]}p \mathbin{\triangle} x, 0 \models p\,\mathsf{U}_{[\max(1,i+l),i+u]}\,q\Big) \end{array} \tag{1}$$

Proof. Let us start with the $\Rightarrow$ direction: assume $x, i \models p\,\mathsf{U}_{[l,u]}\,q$. Hence, there exists a $d \in [l, u]$ such that $x, i+d \models q$ and for all $i < j < i+d$ it is $x, j \models p$. If $i + d \leq 0$ then $0 \leq d \leq -i$, hence $x, i \models p\,\mathsf{U}_{[l,-i]}\,q$ holds. Otherwise, $i + d > 0$; in this case, p holds throughout $(i, 0]$ and thus $x, i \models \mathsf{G}_{[1,-i]}p$ holds. In addition, let $d' = i+d$; note that $1 \leq d' \leq i+u$ and also $i+l \leq d'$, so $x, 0 \models p\mathsf{U}_{[\max(1,i+l),i+u]}\,q$ holds.

Let us now consider the $\Leftarrow$ direction. If $x, i \models p\,\mathsf{U}_{[l,-i]}\,q$, from $1-u \leq i \leq -1$ we get $1 \leq -i \leq u-1$, thus $[l, -i] \subseteq [l, u]$ which entails $x, i \models p\mathsf{U}_{[l,u]}\,q$. Otherwise, let $x, i \models \mathsf{G}_{[1,-i]}p$ and $x, 0 \models p\,\mathsf{U}_{[\max(1,i+l),i+u]}\,q$. That is, p holds throughout $(i, 0]$, and there exists a $k \in [\max(1, i+l), i+u]$ such that $x, k \models q$ and p holds throughout $(0, k)$. Let $d = -i + k$; from $k \in [\max(1, i+l), i+u]$ we get $d \in [l, u]$, which establishes $x, i \models p\,\mathsf{U}_{[l,u]}\,q$. □

Lemma 1 shows how to "split" the evaluation of an *until* formula into the evaluation of two derived formulas, one to be evaluated to the left of the origin,

[7] This restriction is clearly without loss of generality, as $\phi_1\,\mathsf{U}_{[0,0]}\,\phi_2 \equiv \phi_2$.

and one to its right. Next, we use that result to express the satisfiability of a formula of the form $\neg f \vee p \,\mathsf{U}_{[l,u]}\, q$ over a bi-infinite word x as the satisfiability of several different formulas, each evaluated separately either on the whole mono-infinite word x^0 or on the whole mono-infinite word $\widetilde{{}^0x}$.[8]

Lemma 2. *Let $x \in (2^{\Pi})^{\mathbb{Z}}$ and $0 \leq l \leq u < \infty$ such that $u \neq 0$; then:*

$$x \models \neg f \vee p \,\mathsf{U}_{[l,u]}\, q \;\; \underline{\Leftrightarrow} \;\; \begin{array}{c} x^0 \models \neg f \vee p \,\mathsf{U}_{[l,u]}\, q \;\underline{\wedge}\; \widetilde{{}^0x} \models \neg f \vee p \,\mathsf{S}_{[l,u]}\, q \vee (\mathsf{H}p \wedge \mathsf{H}_{=u}\bot) \\ \underline{\wedge} \\ \underline{\forall} 1 \leq i \leq u-1 : \left(\begin{array}{c} \widetilde{{}^0x} \models \mathsf{P}_{=i}\top \wedge \mathsf{H}_{=i+1}\bot \Rightarrow \neg f \vee p \,\mathsf{S}_{[l,u]}\, q \\ \underline{\vee} \\ x^0, 0 \models p \,\mathsf{U}_{[\max(1,-i+l),-i+u]}\, q \end{array} \right) \end{array} \tag{2}$$

Let Φ_{L} and Φ_{R} be the sets of all MTL formulas appearing in left- and right-hand side of Formula (2), respectively (for all values of $1 \leq i \leq u-1$). Note that $u = \exp \mathrm{O}(|\Phi_{\mathrm{L}}|_{\mathrm{M}})$, due to the succinct encoding of constants assumption. Then, $|\Phi_{\mathrm{R}}|_{\mathrm{M}} = \mathrm{O}(|\Phi_{\mathrm{L}}|_{\mathrm{M}})$ and $|\Phi_{\mathrm{R}}|_{\#} = \mathrm{O}(u \cdot |\Phi_{\mathrm{L}}|_{\#}) = |\Phi_{\mathrm{L}}|_{\#} \exp \mathrm{O}(|\Phi_{\mathrm{L}}|_{\mathrm{M}})$.

It is not difficult to show that the equivalence of Formula (2) can be exploited to derive an equivalent formulation of the bi-infinite language $\mathcal{L}^{\mathbb{Z}}\Big(\neg f \vee p \,\mathsf{U}_{[l,u]}\, q\Big)$ in terms of mono-infinite ω-languages and composition operations on them.

Theorem 2. *Let $0 \leq l \leq u < \infty$ and $u \neq 0$; then:*

$$\mathcal{L}^{\mathbb{Z}}\Big(\neg f \vee p \,\mathsf{U}_{[l,u]}\, q\Big) = \begin{array}{c} \widetilde{\mathcal{L}^{\omega}}\Big(\neg f \vee p \,\mathsf{S}_{[l,u]}\, q \vee (\mathsf{H}p \wedge \mathsf{H}_{=u}\bot)\Big) \triangleright \mathcal{L}^{\omega}\Big(\neg f \vee p \,\mathsf{U}_{[l,u]}\, q\Big) \\ \cap \\ \bigcap_{i=1}^{u-1} \left(\begin{array}{c} \widetilde{\mathcal{L}^{\omega}}\Big(\mathsf{P}_{=i}\top \wedge \mathsf{H}_{=i+1}\bot \Rightarrow \neg f \vee p \,\mathsf{S}_{[l,u]}\, q\Big) \triangleright \Big(2^{\Pi}\Big)^{\omega} \\ \cup \\ {}^{\omega}\Big(2^{\Pi}\Big) \triangleright \mathcal{L}_0^{\omega}\Big(p \,\mathsf{U}_{[\max(1,-i+l),-i+u]}\, q\Big) \end{array} \right) \end{array} \tag{3}$$

Other operators. So far, we have provided a characterization of flat formulas only in the form $\neg f \vee p \,\mathsf{U}_{[l,u]}\, q$, for finite $l \leq u$. In order to handle every possible subformula in FSNF, we have to present similar characterizations for the subformulas: (1) $\neg f \vee p \,\mathsf{U}_{[l,\infty)}\, q$; (2) $f \vee p \,\mathsf{R}_I\, q$, for any interval I; (3) $f \Leftrightarrow p \,\mathsf{S}_I\, q \;\equiv\; (\neg f \vee p \,\mathsf{S}_I\, q) \wedge (f \vee \neg p \,\mathsf{T}_I\, \neg q)$, for any interval I; (4) $\beta \in \mathbb{B}(\Pi')$. Such characterizations are derivable similarly as for the bounded *until*. Hence, in the following we just collect the final results for (1) and (2), while the easily derivable results for past operators are provided only in [8].

$$\mathcal{L}^{\mathbb{Z}}(\neg f \vee p \,\mathsf{U}\, q) \;\; = \;\; \begin{array}{c} \widetilde{\mathcal{L}^{\omega}}(\neg f \vee p \,\mathsf{S}\, q \vee \mathsf{H}p) \triangleright \mathcal{L}^{\omega}(\neg f \vee p \,\mathsf{U}\, q) \\ \cap \\ \Big(\widetilde{\mathcal{L}^{\omega}}(\neg f \vee p \,\mathsf{S}\, q) \triangleleft \big(2^{\Pi}\big)^{\omega} \;\cup\; {}^{\omega}\big(2^{\Pi}\big) \triangleright \mathcal{L}_0^{\omega}(p \,\mathsf{U}\, q)\Big) \end{array} \tag{4}$$

$$\mathcal{L}^{\mathbb{Z}}\Big(f \vee p \,\mathsf{R}_{[l,u]}\, q\Big) = \begin{array}{c} \widetilde{\mathcal{L}^{\omega}}\Big(f \vee p \,\mathsf{T}_{[l,u]}\, q\Big) \triangleright \mathcal{L}^{\omega}\Big(f \vee p \,\mathsf{R}_{[l,u]}\, q\Big) \\ \cap \\ \bigcap_{i=1}^{u-1} \left(\begin{array}{c} \widetilde{\mathcal{L}^{\omega}}\big(\mathsf{P}_{=i}\top \wedge \mathsf{H}_{=i+1}\bot \Rightarrow \mathsf{P}p\big) \triangleright \Big(2^{\Pi}\Big)^{\omega} \\ \cup \\ {}^{\omega}\Big(2^{\Pi}\Big) \triangleright \mathcal{L}_0^{\omega}\Big(p \,\mathsf{R}_{[\max(1,-i+l),-i+u]}\, q\Big) \end{array} \right) \end{array} \tag{5}$$

[8] Note that $\mathsf{H}_{=k}\bot$ holds exactly at all positions $j < k$ of any ω-word.

$$\mathcal{L}^{\mathbb{Z}}(f \vee p \,\mathsf{R}\, q) \quad = \quad \begin{array}{c} \widetilde{\mathcal{L}^{\omega}}(f \vee (p \,\mathsf{T}\, q \wedge \mathsf{P}p)) \triangleright \mathcal{L}^{\omega}(f \vee p \,\mathsf{R}\, q) \\ \cup \\ \widetilde{\mathcal{L}^{\omega}}(f \vee p \,\mathsf{T}\, q) \triangleright (\mathcal{L}^{\omega}_{0}(p \,\mathsf{R}\, q) \cap \mathcal{L}^{\omega}(f \vee p \,\mathsf{R}\, q)) \end{array} \tag{6}$$

In fact, to give some intuition about the formulas for the past operators, consider the example of formula $\theta = \mathsf{H}_{[0,3]}p \Rightarrow \mathsf{F}q$. θ in separated normal form becomes $\theta' = (p' \Rightarrow q') \wedge (p' \Leftrightarrow \mathsf{H}_{[0,3]}p) \wedge (q' \Leftrightarrow \mathsf{F}q)$. Then, subformula $\lambda = p' \vee \neg\mathsf{H}_{[0,3]}p = p' \vee \mathsf{P}_{[0,3]}\neg p = p' \vee \top \,\mathsf{S}_{[0,3]}\, \neg p$ can be directly decomposed into:

$$x \models p' \vee \mathsf{P}_{[0,3]}\neg p \quad \Leftrightarrow \quad \begin{array}{c} \widetilde{x^0} \models p' \vee \mathsf{F}_{[0,3]}\neg p \triangleq x^0 \models p' \vee \mathsf{P}_{[0,3]}\neg p \vee \mathsf{H}_{=u}\bot \\ \bigwedge \\ \forall 1 \leq i \leq 2 : \left(\begin{array}{c} x^0 \models \mathsf{P}_{=i}\top \wedge \mathsf{H}_{=i+1}\bot \Rightarrow p' \vee \mathsf{P}_{[0,3]}\neg p \\ \bigvee \\ \widetilde{{}^0x}, 0 \models \mathsf{F}_{[1,-i+3]}\neg p \end{array} \right) \end{array} \tag{7}$$

4.2 From Languages to Automata (to ProMeLa)

In Section 3 we recalled that one can build an automaton that accepts any given MTL ω-language. On the other hand, in the previous section we showed how to reduce MTL satisfiability over $\mathbb{Z}$-languages to MTL satisfiability over ω-languages composed through the operations of $\triangleright$, $\triangleleft$, $\cup$, $\cap$, and $\downarrow^{\Pi}$.

Indeed, the reduction can be fully implemented. In fact, in [8] we substantiate the claim that both BA and AMCA are closed under intersection and union, in such a way that if $\mathcal{A}_1$ and $\mathcal{A}_2$ are two automata (either BA or AMCA), then $|\mathcal{A}_1 \cup \mathcal{A}_2| = \mathrm{O}(|\mathcal{A}_1| + |\mathcal{A}_2|)$ and $|\mathcal{A}_1 \cap \mathcal{A}_2| = \mathrm{O}(|\mathcal{A}_1| \cdot |\mathcal{A}_2|)$.

Moreover, consider $\triangleright$ and let L be a $\mathbb{Z}$-language defined as $\widetilde{L_1} \triangleright L_2$. Then a $\mathbb{Z}$-word x is in L iff $\widetilde{{}^{-1}x} \in L_1$ and $x^0 \in L_2$. Hence, if we have two automata $\mathcal{A}_1, \mathcal{A}_2$ such that $\mathcal{L}^{\omega}(\mathcal{A}_1) = L_1$ and $\mathcal{L}^{\omega}(\mathcal{A}_2) = L_2$ the emptiness of L can be checked noting that $L = \emptyset$ iff $\mathcal{L}^{\omega}(\mathcal{A}_1) = \emptyset$ or $\mathcal{L}^{\omega}(\mathcal{A}_2) = \emptyset$. A very similar reasoning holds for $\triangleleft$. Finally consider the projection: for any MTL formula ϕ over Π let ϕ' be an equi-satisfiable MTL formula over $\Pi' \supseteq \Pi$. Then, $\downarrow^{\Pi} \mathcal{L}^{\mathbb{Z}}(\phi') = \mathcal{L}^{\mathbb{Z}}(\phi)$, and $\mathcal{L}^{\mathbb{Z}}(\phi) = \emptyset$ iff $\mathcal{L}^{\mathbb{Z}}(\phi') = \emptyset$. Correspondingly, the technique to check the satisfiability over the extended alphabet suffices to complete the satisfiability check on the original formula.

Implementing automata. In [2] we presented TRIO2ProMeLa, a tool that translates TRIO formulas (or, equivalently, MTL formulas) into a ProMeLa representation of the automata presented in Section 3. ProMeLa is the input language to the Spin model-checker [10], hence the tool allows one to check the satisfiability of an MTL formula on top of Spin. This approach is very efficient in practice, since it translates directly compositions (through union and intersection) of BA and AMCA to ProMeLa, obtaining a code of the same size as the original automata composition description. In a nutshell, every state of an AMCA is implemented with a ProMeLa process, existential transitions are implemented

as nondeterministic choices, and universal transitions as the parallel run of concurrent processes. The tool also introduces some useful optimizations, such as merging processes when possible. When Spin is run on the automata described in ProMeLa, it unfolds them on-the-fly. This unfolding may lead to a blow-up in the dimension of the automata but it is performed by the model-checker only when needed. This approach is convenient, since in many practical cases, when the original formulas are large, the direct translation to BA and then to ProMeLa is simply unfeasible. We refer the reader to [2, 15] for a detailed description of the translation from AMCA and BA to ProMeLa code.

TRIO2ProMeLa can be reused to provide an implementation of our satisfiability checking procedure over the integers. Once a formula is decomposed as explained in the previous sections, each component is translated into the ProMeLa process that represents the equivalent automaton. All the obtained processes are then suitably composed and coordinated by starting them together at time 0. The results of the various emptiness checks are then combined to have a response about the satisfiability of the original formula.

4.3 Summary and Complexity

Let us briefly summarize the satisfiability checking technique we presented in this section and let us analyze its worst-case asymptotic complexity.

Summary of the satisfiability checking algorithm. Given an MTL formula ϕ over Π, the satisfiability over $\mathbb{Z}$-words is checked according to the following steps.

1. From ϕ, build a formula ϕ' in FSNF such that $\mathcal{L}^{\mathbb{Z}}(\phi) = \downarrow^{\Pi} \mathcal{L}^{\mathbb{Z}}(\phi')$ (Theorem 1).
2. For each subformula ϕ'_i of ϕ', build a set of formulas $\{\phi'_{i,j}\}_j$, whose combined satisfiability over ω-words is equivalent to the satisfiability of ϕ'_i over $\mathbb{Z}$-words (e.g., according to (3) for the bounded *until*). Let $\phi''_i = \bigcup_j \{\phi'_{i,j}\}$.
3. Translate each subformula $\phi'_{i,j}$ into an automaton $\mathcal{A}_{i,j}$ according to what is described in Section 3.
4. For each i, compose the various automata $\mathcal{A}_{i,j}$ according to the structure of the corresponding language equivalences (e.g., according to (3) for the bounded *until*). In practice, for every i we can assume to have two automata $\mathcal{A}_i^+, \mathcal{A}_i^-$ such that $\widetilde{\mathcal{L}^{\omega}}(\mathcal{A}_i^-) \sim \mathcal{L}^{\omega}(\mathcal{A}_i^+) = \mathcal{L}^{\mathbb{Z}}(\phi'_i)$, where $\sim$ is $\triangleright$ or $\triangleleft$.
5. Let $\mathcal{A}^+, \mathcal{A}^-$ be the automata resulting from the intersection of the various $\mathcal{A}_i^{\pm}$'s according to the structure of $\mathcal{L}^{\mathbb{Z}}(\phi')$.
6. Since the equivalence $\downarrow^{\Pi} \mathcal{L}^{\mathbb{Z}}(\phi') = \mathcal{L}^{\mathbb{Z}}(\phi)$ holds by construction, the emptiness test on $\mathcal{L}^{\omega}(\mathcal{A}^+)$ and on $\mathcal{L}^{\omega}(\mathcal{A}^-)$ is equivalent to the satisfiability check of ϕ over $\mathbb{Z}$-words.

Let us go back to our previous example of $\theta = \mathsf{H}_{[0,3]}p \Rightarrow \mathsf{F}q$, and let θ' be θ in FSNF. One of the subformulas in θ' is $\lambda = p' \vee \mathsf{P}_{[0,3]}p$, which can be decomposed according to the left-hand side of 2. Correspondingly, we would build the following automata: $\mathcal{A}_1$ for $p' \vee \mathsf{P}_{[0,3]}\neg p$; $\mathcal{A}_2$ for $p' \vee \mathsf{P}_{[0,3]}\neg p \vee \mathsf{H}_{=u}\bot$;

$\mathcal{A}_3^j$ for $\mathsf{P}_{=j}\top \wedge \mathsf{H}_{=j+1}\bot \Rightarrow p' \vee \mathsf{P}_{[0,3]}\neg p$, $j = 1, 2$; $\mathcal{A}_4^j$ for $\mathsf{F}_{[1,-j+3]}\neg p$, $j = 1, 2$. The automata would then be composed into: $\mathcal{A}_\lambda^- = \mathcal{A}_1$; $\mathcal{A}_\lambda^+ = \mathcal{A}_2 \cap \bigcap_{j=1}^{2} \left(\mathcal{A}_3^j \cup \mathcal{A}_4^j\right)$. Overall, we build two such automata $\mathcal{A}_i^-$ and $\mathcal{A}_i^+$ for each of the 5 subformulas θ' can be decomposed into. Let $\mathcal{A}^+ = \bigcap_{i=1}^{5} \mathcal{A}_i^+$ and $\mathcal{A}^- = \bigcap_{i=1}^{5} \mathcal{A}_i^-$. We conclude that θ is satisfiable iff $\mathcal{L}^\omega(\mathcal{A}^+)$ is non-empty and $\mathcal{L}^\omega(\mathcal{A}^-)$ is non-empty.

Complexity of satisfiability checking over the integers. Let us now evaluate an upper bound on the complexity of the above procedure. The worst-case occurs when overall automata $\mathcal{A}^\pm$ are expanded entirely into nondeterministic BA, thus losing entirely the conciseness of AMCA and the implicit representation of intersections.

First of all, let us estimate the size of every $\mathcal{A}_{i,j}$ with respect to the size of ϕ'_i. In Proposition 2 we recalled that the size $|\mathcal{B}|$ of a Büchi automaton $\mathcal{B}$ encoding an LTL formula θ of size $|\theta|$ is $\exp \mathrm{O}(|\theta|)$. Also, every MTL formula η can be translated into an equivalent LTL formula of size $\exp \mathrm{O}(|\eta|_\# \, |\eta|_\mathrm{M})$. In our case, every formula $\phi'_{i,j}$ is translated into an automaton of size:

$$|\mathcal{A}_{i,j}| = \exp\exp \mathrm{O}\left(\left|\phi'_{i,j}\right|_\# \left|\phi'_{i,j}\right|_\mathrm{M}\right) = \exp\exp \mathrm{O}\left(\left|\phi'_{i,j}\right|_\mathrm{M}\right)$$

because every subformula $\phi'_{i,j}$ has a constant (i.e., independent of $|\phi|$) number of connectives. Also, we noted that $\left|\phi'_{i,j}\right|_\mathrm{M} = |\phi'_i|_\mathrm{M}$, so:

$$|\mathcal{A}_{i,j}| = \exp\exp \mathrm{O}\left(|\phi'_i|_\mathrm{M}\right)$$

Next, let us estimate the size of $\mathcal{A}_i^\pm$. Roughly, $\mathcal{A}_i^\pm$ is the intersection $\bigcap_j \mathcal{A}_{i,j}$, hence its size is upper-bounded by the product of the sizes $|\mathcal{A}_{i,j}|$:

$$\left|\mathcal{A}_i^\pm\right| = \prod_j |\mathcal{A}_{i,j}| \leq \left(\max_j |\mathcal{A}_{i,j}|\right)^{\left|\phi''_i\right|} = \left(\exp\exp \mathrm{O}(|\phi'_i|_\mathrm{M})\right)^{\exp \mathrm{O}\left(\left|\phi'_i\right|_\mathrm{M}\right)}$$

where the equivalence between $|\phi''_i|$ and $\exp \mathrm{O}(|\phi'_i|_\mathrm{M})$ was highlighted in Section 4.1. After some manipulation, we get:

$$\left|\mathcal{A}_i^\pm\right| = \exp\Big(\left(\exp \mathrm{O}(|\phi'_i|_\mathrm{M})\right)\left(\exp \mathrm{O}(|\phi'_i|_\mathrm{M})\right)\Big) = \exp\exp \mathrm{O}\left(|\phi'_i|_\mathrm{M}\right)$$

Then, the overall size of $\mathcal{A}^+$ and $\mathcal{A}^-$ can be computed as:

$$\left|\mathcal{A}^+\right| + \left|\mathcal{A}^-\right| = \mathrm{O}(\left|\mathcal{A}^\pm\right|) = \prod_i \left|\mathcal{A}_i^\pm\right| \leq \left(\max_i |\mathcal{A}_i|\right)^{\left|\phi'\right|_\#} = \exp\left(\left|\phi'\right|_\# \exp \mathrm{O}(|\phi'|_\mathrm{M})\right)$$

thanks to the equivalence between $|\phi'_i|_\mathrm{M}$ and $\mathrm{O}(|\phi'|_\mathrm{M})$ stated in Remark 4.1.

Finally, Theorem 1 relates the size of ϕ' to that of the original formula ϕ, so we have:

$$\left|\mathcal{A}^+\right| + \left|\mathcal{A}^-\right| = \exp\left(\left|\phi\right|_\# \exp \mathrm{O}(|\phi|_\mathrm{M})\right)$$

From the well-known result that emptiness check of a Büchi automaton takes time polynomial (actually, linear) in the size of the automaton (see Proposition 2), we have established the following.

Theorem 3 (Upper-bound complexity). *The verification algorithm of this paper can check the satisfiability of an MTL formula ϕ over $\mathbb{Z}$-words in time doubly-exponential in the size $|\phi|$ of ϕ.*

The doubly-exponential time performance is worst-case optimal, because the satisfiability problem for MTL over the integers is an EXPSPACE-complete problem, as it is over the naturals [1].

Theorem 4 (Complexity of MTL over the integers). *The satisfiability problem for MTL over the integers is* EXPSPACE*-complete.*

Proof (sketch). From the upper-bound analysis and Proposition 2 it follows also that the problem is is decidable in nondeterministic (singly) exponential space, hence it is in EXPSPACE. For the EXPSPACE-hardness proof, one reduces from the satisfiability of future-MTL over integer-timed ω-words, which is also EXPSPACE-complete [1]. See [8] for details. □

5 Discussion

As we discussed in the Introduction, bi-infinite time models for temporal logic have been studied very rarely. Let us briefly consider a few noticeable exceptions.

On the more practical side, Pradella et al. [13] recently developed a tool-supported technique for bounded model-checking of temporal logic specifications over the integers. Bounded model-checking is a verification technique based on reduction to the propositional satisfiability (SAT) problem, for which very efficient off-the-shelf tools exist. The technique is however incomplete, as it only looks for words of length up to a given bound k, where k is a parameter of the verification problem instance. [13] describes a direct encoding of MTL bounded satisfiability as a SAT instance and reports on some interesting experimental results with an implementation. [13] also discusses the appeal of bi-infinite time from a system modeling perspective; some of its considerations are also discussed in the Introduction of the present paper.

In the area of automata theory and formal languages, there exist a few works considering bi-infinite time models. For instance Perrin and Pin [12] introduce bi-infinite words and automata on them, and extend some classical results for mono-infinite words to these new models. In the same vein, Muller et al. [11] establish the decidability of LTL over the integers. However, to the best of our knowledge the complexity of temporal logic over bi-infinite time has never been investigated in previous work.

On the contrary, temporal logic over mono-infinite time models has been extensively studied, and it has been the object of an impressive amount of both practical and theoretical research (e.g., [7, 9, 17, 1, 6]). Satisfiability of both LTL [14] and MTL [1] — also with past operators — over mono-infinite discrete time models has been thoroughly investigated. Sistla and Clarke [14] proved that LTL satisfiability over the naturals is PSPACE-complete, with a (singly) exponential time algorithm. Correspondingly, Alur and Henzinger [1] proved

that MTL satisfiability over mono-infinite integer timed words is EXPSPACE-complete, and provided a doubly-exponential time algorithm. The same holds for bi-infinite discrete time, as we showed in this paper.

In the future, we plan to work on the implementation of an automated translator from integer-time MTL specifications to Spin models, and to experiment with it to assess the practical feasibility of the approach, also in comparison with similar tools for mono-infinite time models. Also, the related MTL model-checking problem over integer time will be investigated.

References

1. R. Alur and T. A. Henzinger. Real-time logics: Complexity and expressiveness. *Information and Computation*, 104(1):35–77, 1993.
2. D. Bianculli, A. Morzenti, M. Pradella, P. San Pietro, and P. Spoletini. Trio2Promela: A model checker for temporal metric specifications. In *ICSE Companion*, pages 61–62, 2007.
3. P. Bouyer, N. Markey, J. Ouaknine, and J. Worrell. The cost of punctuality. In *Proceedings of LICS'07*, pages 109–120. IEEE Computer Society, 2007.
4. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 2000.
5. A. Coen-Porisini, M. Pradella, and P. San Pietro. A finite-domain semantics for testing temporal logic specifications. In *Proceedings of FTRTFT'98*, volume 1486 of *LNCS*, pages 41–54, 1998.
6. S. Demri and P. Schnoebelen. The complexity of propositional linear temporal logics in simple cases. *Information and Computation*, 174(1):84–103, 2002.
7. E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 996–1072. Elsevier Science, 1990.
8. C. A. Furia and P. Spoletini. MTL satisfiability over the integers. Technical Report 2008.2, DEI, Politecnico di Milano, 2008.
9. D. M. Gabbay, I. Hodkinson, and M. Reynolds. *Temporal Logic (vol. 1): mathematical foundations and computational aspects*. Oxford University Press, 1994.
10. G. J. Holzmann. *The SPIN Model Checker: Primer and Reference Manual*. 2003.
11. D. E. Muller, P. E. Schupp, and A. Saoudi. On the decidability of the linear Z-temporal logic and the monadic second order theory. In *Proc. of ICCI'92*, pages 2–5, 1992.
12. D. Perrin and J.-É. Pin. *Infinite Words*, volume 141 of *Pure and Applied Mathematics*. Elsevier, 2004.
13. M. Pradella, A. Morzenti, and P. San Pietro. The symmetry of the past and of the future. In *Proceedings of ESEC/FSE'07*, pages 312–320, 2007.
14. A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985.
15. P. Spoletini. *Verification of Temporal Logic Specification via Model Checking*. PhD thesis, DEI, Politecnico di Milano, May 2005.
16. W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 133–164. Elsevier Science, 1990.
17. M. Y. Vardi. *Handbook of Modal Logic*, chapter Automata-Theoretic Techniques for Temporal Reasoning, pages 971–990. 2006.