Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning
Main Track

Computing All Facts Entailed By An LTL Specification

Przemysław Andrzej Wałęga[1] , **Michał Zawidzki**[1,2] , **Christoph Haase**[1]
[1]Department of Computer Science, University of Oxford
[2]Department of Logic, University of Łódź
{przemyslaw.walega, michal.zawidzki, christoph.haase}@cs.ox.ac.uk

Abstract

We study the problem of efficiently computing all (usually infinitely many) facts which are entailed by a specification written in linear temporal logic (LTL)—a standard formalism for specifying and verifying properties of computations in reactive systems. This problem can be seen as a generalisation of the standard entailment checking, but whose output provides a much wider understanding of the system's behaviour. We show that in full LTL the problem can be solved in doubly exponential time, whereas for Horn fragments of LTL, which can be seen as temporal logic programs, the problem can be solved in exponential or only quadratic time, depending on the allowed temporal operators in the input formula. Moreover, we show that these bounds are optimal. We also implement and experimentally compare two techniques for solving the problem: an automata-based algorithm for full LTL and a materialisation-based algorithm for Horn fragments. The obtained results suggest practical usefulness of our approach.

1 Introduction

Analysing temporal data is of increasing importance due to a number of emerging applications such as modelling blockchain-based smart contracts (Alharby and Van Moorsel 2017), identifying traffic anomalies (Münz and Carle 2007), algorithmic trading (Nuti et al. 2011), and IoT applications (Lécué 2017). To capture formally the evolving behaviour of dynamic domains, various temporal languages have been introduced (Vardi 2009; Abadi and Manna 1989; Demri, Goranko, and Lange 2016), including the seminal linear temporal logic (LTL) (Pnueli 1977) with its fragments and modifications, see, e.g., Alur and La Torre (2004), Artale et al. (2013), Demri and Schnoebelen (2002).

LTL-based formalisms are currently commonly used in KRR to express time-sensitive knowledge and ontologies (Artale et al. 2021; Lutz, Wolter, and Zakharyaschev 2008; Abadi and Manna 1989; Aguado et al. 2013; Ronca et al. 2018), as illustrated in the following simple example.

Example 1. *Consider a system consisting of 3 devices with IDs* $= \{1, 2, 3\}$*, all initially backed up on the same day, but each one in some unknown mode from the set* $M = \{1, \dots, 10\}$*. We express it with Formulas* (1) *for all* $id \in IDs$*, where proposition* $Bckp_m^{id}$ *being true at some day represents the fact that the device number* id *is backed up on this day in mode* m*. We assume that no device is simultaneously backed up in two different modes* m *and* m'*, as stated by Formulas* (2) *for all* $id \in IDs$ *and distinct* $m, m' \in M$*, where* G *is the 'always in the future' LTL-operator. Modes determine the frequency of backups, namely backups in a mode* $m \in M$ *are performed regularly every* m *days, as stated by Formulas* (3) *with* m *iterations of the 'next' operator* X*. The system is safe on a given day if all three devices are backed up on this day, as stated by Formulas* (4)*, for all* $m, m', m'' \in M$*.*

$$Bckp_1^{id} \vee \dots \vee Bckp_{10}^{id}, \tag{1}$$

$$\mathsf{G}\left(Bckp_m^{id} \wedge Bckp_{m'}^{id} \rightarrow \bot\right), \tag{2}$$

$$\mathsf{G}\left(Bckp_m^{id} \rightarrow \mathsf{X}^m Bckp_m^{id}\right), \tag{3}$$

$$\mathsf{G}\left(Bckp_m^1 \wedge Bckp_{m'}^2 \wedge Bckp_{m''}^3 \rightarrow Safe\right). \tag{4}$$

The main reasoning problems considered in this setting are checking consistency (satisfiability) of the input temporal knowledge and answering temporal queries mediated by a temporal knowledge (formula entailment). There exist various approaches to solve these problems, including tableau systems (Wolper 1985; Lichtenstein and Pnueli 2000), automata-based techniques (Vardi 2005; Li et al. 2014; Duret-Lutz et al. 2022), reductions to SAT (Geatti, Gigante, and Montanari 2021), and rewriting to first-order logic (Artale et al. 2021; Ryzhikov, Wałęga, and Zakharyaschev 2020), which have been implemented in a plethora of reasoning systems. Applying these methods, for example, allows an engineer working on safety requirements to check if the knowledge in Example 1 is consistent, or whether the system is safe on some particular, say 27th, day.

What if the engineer's task is more general, as they need to compute the frequency with which the system from Example 1 is guaranteed to be safe and to determine on which days the system may be vulnerable? In this case the engineer needs a more in-depth understanding of the logical consequences of the given temporal knowledge, which can be provided by presenting all of the (infinitely many) temporal facts entailed by Formulas (1)–(4); note that this cannot be obtained using standard services like consistency checking or query answering. Computing all of the entailed temporal facts is both conceptually and computationally challenging. Hence, instead of asking the engineer to come up with a case-based solution, we would prefer to provide them

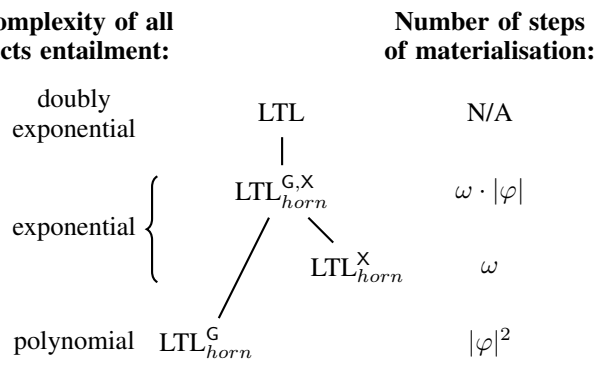


Figure 1: Cumulative results from this paper

with a general system performing reasoning automatically.

We address this task by developing optimal algorithms for computing all facts entailed by an LTL-formula. With the reasoning procedures developed in this paper, an engineer can effortlessly determine that the system from Example 1 is only guaranteed to be safe on the days of the form $2520 \cdot n$, for all $n \in \mathbb{N}$. Hence they may want to consider taking additional measures to increase the safety of the system.

Our contributions in this paper are as follows:

- After presenting preliminaries in Section 2, in Section 3 we define our problem formally as computing the intersection of all traces satisfying a given LTL-formula.
- In Section 4 we show that the problem can be solved in doubly exponential time which is worst-case unavoidable. We present an optimal algorithm for solving the problem which is based on manipulating a non-deterministic Büchi automaton corresponding to the input formula (which can be constructed using off-the-shelf systems).
- The negative complexity result from Section 4 motivates us to search for fragments of LTL in which our computational problem is easier. To this end, we study Horn fragments of LTL that were exploited in the setting of temporal ontology-based data access (Artale et al. 2013). In particular, we focus on Horn fragments $\mathrm{LTL}_{horn}^{\mathsf{G},\mathsf{X}}$, $\mathrm{LTL}_{horn}^{\mathsf{X}}$, and $\mathrm{LTL}_{horn}^{\mathsf{G}}$ which restrict temporal operators allowed in rules to the ones listed in the upper indices; the Hasse diagram of these fragments is depicted in Figure 1.
- In contrast to full LTL, each (satisfiable) formula φ in the fragments we consider has a unique least model, which can be computed via *materialisation*, that is, successive applications of Horn clauses seen as 'if-then' rules. As we show in Section 5, materialisation can take up to $\omega \cdot |\varphi|$ steps in $\mathrm{LTL}_{horn}^{\mathsf{G},\mathsf{X}}$, up to ω steps in $\mathrm{LTL}_{horn}^{\mathsf{X}}$, but only up to $|\varphi|^2$ steps in $\mathrm{LTL}_{horn}^{\mathsf{G}}$, as shown in Figure 1.
- In Section 6 we use the results on materialisation in Horn fragments to show that computing all facts entailed by $\mathrm{LTL}_{horn}^{\mathsf{G},\mathsf{X}}$- and $\mathrm{LTL}_{horn}^{\mathsf{X}}$-formulas requires exponential time. In the case of $\mathrm{LTL}_{horn}^{\mathsf{G}}$-formulas, however, the problem is tractable, in particular, it becomes solvable in time $O(|\varphi|^4)$—see Figure 1. Our algorithm for these fragments is based on performing materialisation steps, where termination is ensured by applying specific stopping conditions and completeness of the construction is obtained as a result of *unfolding* partial materialisations.
- In Section 7 we present a prototype implementation of the algorithms from Sections 4 and 6. We perform experiments for full LTL and Horn fragments on randomly generated formulas. Materialisation-based approach vastly outperforms the automata-based approach, suggesting practical usefulness of the former.

2 Preliminaries

Linear Temporal Logic. Formulas of LTL (Pnueli 1977) are generated by the following grammar, where p is a proposition from a countable set PROP:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{X}\varphi \mid \varphi \mathsf{U} \varphi.$$

We use standard abbreviations such as $\top := p \vee \neg p$, for an arbitrary $p \in \mathsf{PROP}$, $\bot := \neg\top$, $\varphi \wedge \psi := \neg(\neg\varphi \vee \neg\psi)$, $\varphi \rightarrow \psi := \neg\varphi \vee \psi$, $\varphi \leftrightarrow \psi := (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$, $\mathsf{F}\varphi := \top \mathsf{U} \varphi$, $\mathsf{G}\varphi := \neg\mathsf{F}\neg\varphi$, and $\mathsf{X}^k\varphi$ for φ preceded with $k \in \mathbb{N}$ operators X, where φ and ψ are any formulas. The *signature*, $\mathsf{PROP}(\varphi)$, of a formula φ is the set of all propositions in φ. The *size* of φ, written $|\varphi|$, is the number of symbols in φ.

Formulas of LTL are interpreted over (infinite) *traces* $\sigma \in (2^{\mathsf{PROP}})^\omega$; we also exploit *finite traces* $\sigma \in (2^{\mathsf{PROP}})^*$ (we will always explicitly mention their finiteness) and denote their lengths by $|\sigma|$. *Satisfaction* of a formula φ at a time point $i \in \mathbb{N}$ of an infinite trace σ, denoted by $\sigma, i \models \varphi$, is defined below, where $\sigma(i)$ is the ith element of σ:

$$\begin{array}{lll}
\sigma, i \models p & \text{iff} & p \in \sigma(i) \\
\sigma, i \models \neg\varphi & \text{iff} & \sigma, i \not\models \varphi \\
\sigma, i \models \varphi_1 \vee \varphi_2 & \text{iff} & \sigma, i \models \varphi_1 \text{ or } \sigma, i \models \varphi_2 \\
\sigma, i \models \mathsf{X}\varphi & \text{iff} & \sigma, i+1 \models \varphi \\
\sigma, i \models \varphi_1 \mathsf{U} \varphi_2 & \text{iff} & \sigma, j \models \varphi_2 \text{ for some } j \geq i \text{ and} \\
 & & \sigma, k \models \varphi_1 \text{ for all } k \text{ with } i \leq k < j
\end{array}$$

A formula φ is *satisfied* by a trace σ if $\sigma, 0 \models \varphi$ and φ is *satisfiable* if it is satisfied by some trace. A formula φ *entails* a formula ψ if $\sigma, 0 \models \varphi$ implies $\sigma, 0 \models \psi$, for each trace σ. For $\sigma = a_0 a_1 \cdots$ and any $i, j \in \mathbb{N}$, we let $\sigma(i) = a_i$ and $\sigma[i, j] := a_i a_{i+1} \cdots a_j$. A trace σ is *contained* in a trace σ', written as $\sigma \subseteq \sigma'$, if $\sigma(i) \subseteq \sigma'(i)$ for each $i \in \mathbb{N}$. The *intersection* $\bigcap \mathsf{Tr}$ and the *union* $\bigcup \mathsf{Tr}$ of traces from a set Tr are traces such that $\bigcap \mathsf{Tr}(i) = \bigcap_{\sigma \in \mathsf{Tr}} \sigma(i)$ and $\bigcup \mathsf{Tr}(i) = \bigcup_{\sigma \in \mathsf{Tr}} \sigma(i)$ for each $i \in \mathbb{N}$.

Büchi Automata. A (non-deterministic) *Büchi automaton* (NBA) is a tuple $\mathcal{A} = (\mathcal{Q}, \Sigma, \delta, q_0, \mathcal{F})$, where $\mathcal{Q}$ is a finite set of *states*, Σ is a finite *alphabet*, $\delta : \mathcal{Q} \times \Sigma \rightarrow 2^{\mathcal{Q}}$ is a *transition function*, $q_0 \in \mathcal{Q}$ is an *initial state*, and $\mathcal{F} \subseteq \mathcal{Q}$ is a set of *final states*. An NBA $\mathcal{A}$ accepts an ω-word $a_0 a_1 \cdots \in \Sigma^\omega$ if there is a sequence $q_0, q_1, \ldots \in \mathcal{Q}^\omega$ of states, called an *accepting run*, such that $q_{i+1} \in \delta(q_i, a_i)$ for each $i \in \mathbb{N}$ and some state in $\mathcal{F}$ occurs infinitely often in $q_0, q_1, \ldots$ The language of $\mathcal{A}$ is the set $\mathcal{L}(\mathcal{A})$ of all ω-words accepted by

$\mathcal{A}$; checking emptiness of $\mathcal{L}(\mathcal{A})$ is NL-complete (Vardi and Wolper 1994). For each LTL-formula φ there is an NBA $\mathcal{A}_\varphi$ accepting exactly those infinite traces which satisfy φ; the size of such $\mathcal{A}_\varphi$ (i.e., the number of its states) is exponentially large in the size of φ (Vardi and Wolper 1994).

Horn Fragment of LTL. Following the approach of Artale et al. (2013)[1] we define Horn fragments of LTL by exploiting the clausal form of LTL-formulas, which was inspired by the *separated normal form* by Fisher (1991). To this end, we let a *temporal atom* be given by the following grammar, where $p \in \mathsf{PROP}$:

$$\lambda ::= \bot \mid p \mid \mathsf{X}\lambda \mid \mathsf{G}\lambda \mid \lambda \mathsf{U} \lambda.$$

An LTL-formula in *clausal form* is given by the grammar:

$$\varphi ::= \mathsf{X}^k p \mid \mathsf{G}(\neg\lambda_1 \vee \dots \vee \neg\lambda_n \vee \lambda_{n+1} \vee \dots \vee \lambda_{n+m}) \mid \varphi \wedge \varphi,$$

where n, m are positive integers, $k \in \mathbb{N}$, whereas λ and all λ_i are temporal atoms. Similar normal forms have been shown to preserve satisfiability of formulas (Artale et al. 2013; Fisher 1991); it is however not hard to see that they also preserve entailment, as we show below.

Proposition 2. *Each LTL-formula φ can be transformed in logarithmic space into φ' in clausal form, such that φ and φ' entail the same LTL-formulas over the signature of φ.*

Proof sketch. The transformation 'renames' subformulas ψ of φ to fresh propositions of the form a_ψ and introduces clauses which guarantee that each ψ and the corresponding a_ψ hold at the same time points. For example $\varphi = p \vee \mathsf{X}q$ is transformed into $\varphi' = a_\varphi \wedge \mathsf{G}(a_\varphi \leftrightarrow a_p \vee a_{\mathsf{X}q}) \wedge \mathsf{G}(a_{\mathsf{X}q} \leftrightarrow \mathsf{X}a_q) \wedge \mathsf{G}(a_p \leftrightarrow p) \wedge \mathsf{G}(a_q \leftrightarrow q)$, where an equivalence such as in $\mathsf{G}(a_\varphi \leftrightarrow a_p \vee a_{\mathsf{X}q})$ is an abbreviation of $\mathsf{G}(\neg a_\varphi \vee a_p \vee a_{\mathsf{X}q}) \wedge \mathsf{G}(\neg a_p \vee a_\varphi) \wedge \mathsf{G}(\neg a_{\mathsf{X}q} \vee a_\varphi)$. □

Observe that a translation in the opposite direction to the one from Proposition 2 can be obtained by simply applying abbreviations from the beginning of this section, so that $\bot$, $\wedge$, and G are eliminated from a formula in clausal form.

A formula is *Horn* when it is in clausal form with $m = 1$. Such a formula can be seen as:

- a set of *facts* of the form $\mathsf{X}^k p$, called a *dataset*, $\mathcal{D}$, and
- a set of *rules* of the form $\mathsf{G}(\neg\lambda_1 \vee \dots \vee \neg\lambda_n \vee \lambda_{n+1})$, called a *program*, Π.

We let $|\mathcal{D}|$ and $|\Pi|$ be the number of symbols in $\mathcal{D}$ and Π, respectively. Moreover, following the convention of logic programming, we will write a rule, r, as

$$\lambda_{n+1} \leftarrow \lambda_1 \wedge \dots \wedge \lambda_n,$$

where $\lambda_1 \wedge \dots \wedge \lambda_n$ is the *body* of r, in symbols $\mathsf{body}(r)$, with each λ_i, for $i \leq n$, being its *body atom*, and λ_{n+1} is its *head*. If the head of r mentions $\bot$, we call r a $\bot$*-rule*. Similarly to Artale et al. (2013), we will pay special attention to Horn formulas which allow only for G and X among temporal operators in rules, or only for X, or only for G. The corresponding classes: $\mathrm{LTL}_{horn}^{\mathsf{G},\mathsf{X}}$, $\mathrm{LTL}_{horn}^{\mathsf{X}}$, and $\mathrm{LTL}_{horn}^{\mathsf{G}}$ form a Hasse diagram depicted in Figure 1; note that these restrictions influence programs only, so the definition of datasets remains the same. It is also worth noting that there exist a number of other, yet similar, definitions of Horn temporal formulas in the literature (Chen and Lin 1993; Orgun and Ma 2005; Gabbay 1987; Abadi and Manna 1989).

Materialisation. *Materialisation* is a procedure that can be used for reasoning in $\mathrm{LTL}_{horn}^{\mathsf{G},\mathsf{X}}$, $\mathrm{LTL}_{horn}^{\mathsf{X}}$, and $\mathrm{LTL}_{horn}^{\mathsf{G}}$ (where rules do not mention disjunctions or U in their heads, and so, are 'deterministic'). Given such a formula composed of a dataset $\mathcal{D}$ and a program Π, *materialisation* consists in successively applying the *immediate consequence operator* T_Π to the least (with respect to containment) trace $\sigma_\mathcal{D}$ satisfying $\mathcal{D}$. The operator T_Π maps a trace σ to the least trace σ' containing σ and satisfying the following property for each $r \in \Pi$ which is not a $\bot$-rule: whenever σ satisfies each body atom of r at a time point i, then σ' satisfies the head of r at i; by the form of $\mathrm{LTL}_{horn}^{\mathsf{G},\mathsf{X}}$-, $\mathrm{LTL}_{horn}^{\mathsf{X}}$-, and $\mathrm{LTL}_{horn}^{\mathsf{G}}$-formulas, there exists exactly one such σ', and so, $T_\Pi(\sigma)$ is well defined. Successive applications of T_Π to $\sigma_\mathcal{D}$ constitute a transfinite sequence of traces $T_\Pi^\alpha(\sigma_\mathcal{D})$ for ordinals α:

$$\begin{aligned} T_\Pi^0(\sigma_\mathcal{D}) &= \sigma_\mathcal{D}, \\ T_\Pi^{\alpha+1}(\sigma_\mathcal{D}) &= T_\Pi(T_\Pi^\alpha(\sigma_\mathcal{D})), && \text{for } \alpha \text{ an ordinal,} \\ T_\Pi^\alpha(\sigma_\mathcal{D}) &= \bigcup_{\beta<\alpha} T_\Pi^\beta(\sigma_\mathcal{D}), && \text{for } \alpha \text{ a limit ordinal.} \end{aligned}$$

We let $\mathfrak{C}_{\Pi,\mathcal{D}} = T_\Pi^{\omega_1}(\sigma_\mathcal{D})$, where ω_1 is the first uncountable ordinal; if $\mathfrak{C}_{\Pi,\mathcal{D}}$ satisfies all $\bot$-rules in Π, then $\mathfrak{C}_{\Pi,\mathcal{D}}$ is the least trace satisfying both Π and $\mathcal{D}$ (Brandt et al. 2017). We say that materialisation of Π and $\mathcal{D}$ takes α steps if α is the least ordinal such that $T_\Pi^\alpha(\sigma_\mathcal{D}) = \mathfrak{C}_{\Pi,\mathcal{D}}$.

Complexity of Reasoning. Satisfiability of LTL-formulas over infinite traces (Sistla and Clarke 1985) as well as over finite traces (De Giacomo and Vardi 2013) is PSpace-complete. Satisfiability of Horn formulas over $\mathbb{Z}$ in fragments similar to our $\mathrm{LTL}_{horn}^{\mathsf{G},\mathsf{X}}$ and $\mathrm{LTL}_{horn}^{\mathsf{G}}$ (but with past temporal operators and over the irreflexive semantics) is PSpace- and P-complete, respectively (Artale et al. 2013). More recently, a number of problems in Horn temporal logics have been studied for their complexity, including satisfiability in interval logics (Bresolin et al. 2017; Wałęga 2023), rewritability of ontology-mediated queries with LTL and metric temporal logic (MTL) operators (Artale et al. 2021; Ryzhikov, Wałęga, and Zakharyaschev 2020), as well as fact entailment in extensions of Datalog with MTL-operators (Wałęga et al. 2019; Wałęga et al. 2020; Wałęga et al. 2023).

3 Intersection of All Traces

The main problem we consider in this paper is to efficiently compute all the atomic facts entailed by a given LTL-formula φ. One way to address this problem is to consider

[1] In contrast to Artale et al. (2013), however, we use the standard syntax of LTL with no past operators, we adapt the standard semantics of LTL which is over $\mathbb{N}$ and not over $\mathbb{Z}$, and we exploit the 'reflexive' rather than 'irreflexive' semantics of U and G.

all the traces satisfying φ and to check, for every position i, which atoms p occur in all these traces at this position. This would allow us to detect, for each i, all the atoms entailed by φ at i. In other words, this approach consists in computing the intersection of all traces satisfying φ. This intersection is a trace itself, denoted by σ_φ, and defined formally below.

Definition 3. *Let φ be an LTL-formula and let Tr_φ be the set of all traces satisfying φ at the time point 0. We define the* intersection of all traces *of φ as the trace $\sigma_\varphi = \bigcap \mathsf{Tr}_\varphi$.*

In the definition above we adapt the convention that if $\mathsf{Tr}_\varphi = \emptyset$, then $\bigcap \mathsf{Tr}_\varphi$ is the 'full trace' over the signature of φ, $(\mathsf{PROP}(\varphi))^\omega$. Thus, σ_φ is well-defined even if φ is unsatisfiable. At the same time it is worth noting that it is not guaranteed that σ_φ satisfies φ. For example if $\varphi = \mathsf{F}p$, then σ_φ is an empty trace $\emptyset^\omega$, and so, $\sigma_\varphi, 0 \not\models \varphi$. On the other hand, if φ is an $\mathrm{LTL}_{horn}^{\mathsf{G},\mathsf{X}}$-formula (which clearly also applies to more restricted $\mathrm{LTL}_{horn}^{\mathsf{X}}$- and $\mathrm{LTL}_{horn}^{\mathsf{G}}$-formulas) then, as we described in Section 2, if φ is satisfiable, it has the unique least trace. This least trace is the intersection of all traces of φ, and so, it coincides with σ_φ. Therefore, in contrast to the general case, we obtain that $\sigma_\varphi, 0 \models \varphi$.

The problem we address in this paper is to efficiently compute σ_φ. Since σ_φ is an infinite trace, we start by providing a way to represent it finitely. To this end, we observe, as stated below, that σ_φ needs to be an *ultimately-periodic* trace.

Proposition 4. *For any LTL-formula φ we have $\sigma_\varphi = uw^\omega$, where u and w are some finite traces.*

Note that it is well known that each satisfiable LTL-formula has an ultimately-periodic trace (this fact is heavily exploited in LTL reasoners), but to show that σ_φ is also ultimately periodic, additional argumentation is in place. For that purpose, we can construct a formula of S1S (monadic second-order logic of one successor) containing

$$\forall x(p_i(x) \leftrightarrow \forall X_1, \ldots, X_n(\mathsf{tr}_\varphi(X_1, \ldots, X_n) \rightarrow x \in X_i)),$$

for each of n propositions p_i in the signature of φ. It states that for each position x, the proposition p_i holds at x if and only if p_i holds at x in all traces of φ. For this, we construct a formula $\mathsf{tr}_\varphi(X_1, \ldots, X_n)$ with set variables $X_1, \ldots, X_n$, which expresses that φ holds at 0 when, for each proposition p_i in φ, p_i holds at every position from X_i and only there. Any S1S formula can be transformed into a Büchi automaton and each Büchi automaton accepting some word accepts an ultimately periodic word. In our case, the automaton accepts exactly one word σ_φ, so that σ_φ is ultimately periodic.

Next, we observe that since σ_φ is ultimately periodic, there need to exist unique u and w with the least sum of lengths which represent σ_φ. We will call such a pair (u, w) the shortest representation of σ_φ as stated formally below. It is also worth observing that the shortest representation of a periodic word can be computed from a non-shortest one in linear time (Crochemore and Rytter 1994).

Definition 5. *For an LTL-formula φ we let the* shortest representation *of σ_φ be the pair (u, w) of finite traces with minimal sum of lengths $|u| + |w|$ such that $uw^\omega = \sigma_\varphi$.*

Now, based on the above discussion on the form of σ_φ and its representation, we are ready to formulate the main computational problem considered in this paper—computing the intersection of all traces satisfying a given LTL-formula.

Definition 6. *We let* the intersection of traces *be the following computational problem:*

Input: *an LTL-formula φ,*

Output: *the shortest representation (u, w) of σ_φ.*

By the definition, the intersection of traces provides us a complete information about entailment of all propositions by a formula φ. Furthermore, we can use intersection of traces to check entailment of complex formulas by simulating them with fresh propositions; for example, to determine all the positions in which a complex formula ψ is entailed by φ, it suffices to compute the intersection of traces for the formula $\varphi' = \varphi \wedge \mathsf{G}(p_\psi \leftrightarrow \psi)$, which simulates ψ with a fresh proposition p_ψ. Then, all the positions in which p_ψ holds in σ_φ coincide with the positions in which ψ is entailed by φ.

Finally, we note that the argument (via constructing an S1S formula) for Proposition 4 implies that the intersection of traces is computable. The involved construction of a Büchi automaton from an S1S-formula is, however, non-elementary. In the next sections we will show how to compute the intersection of traces efficiently.

4 An Optimal Algorithm for LTL

In this section, we present an optimal algorithm for computing the intersection of traces of an arbitrary LTL-formula φ. We will prove, however, that for full LTL the worst-case complexity of the problem is doubly exponential.

Algorithm 1: Intersection of traces for LTL

Input: an LTL-formula φ
Output: the shortest representation (u, w) of σ_φ if φ is satsifiable, or unsat otherwise

1 Construct an NBA $\mathcal{A}_\varphi = (\mathcal{Q}, \Sigma, \delta, q_0, F)$ for φ;
2 $j \coloneqq 0$; $\mathcal{Q}_0 \coloneqq \{q_0\}$;
3 **loop**
4 $\quad \mathcal{Q}_{j+1} \coloneqq \{q' \in \mathcal{Q} \mid q' \in \delta(q, a)$ for some $q \in \mathcal{Q}_j$, $a \in \Sigma$, $\mathcal{A}_\varphi$ has an accepting run from $q'\}$;
5 $\quad$ **if** $\mathcal{Q}_{j+1} = \emptyset$ **then return** unsat;
6 $\quad a_j \coloneqq \bigcap \{a \in \Sigma \mid \delta(q, a) \in \mathcal{Q}_{j+1}, q \in \mathcal{Q}_j\}$;
7 $\quad$ **if** $\mathcal{Q}_i = \mathcal{Q}_j$ *for some* $i < j$ **then**
8 $\quad\quad (u, w) \coloneqq$ the shortest representation of $a_0 \cdots a_{i-1}(a_i \cdots a_{j-1})^\omega$;
9 $\quad\quad$ **return** (u, w);
10 $\quad j \coloneqq j + 1$;

Algorithm 1 presents our approach. In Line 1 it constructs an NBA $\mathcal{A}_\varphi$ accepting the infinite traces which satisfy φ (Vardi and Wolper 1994), where each symbol a in the alphabet $\Sigma = 2^{\mathsf{PROP}(\varphi)}$ of $\mathcal{A}_\varphi$ is a subset of propositions occurring in φ. In Line 2, it initialises counter j to 0 and $\mathcal{Q}_0$ to the singleton containing only the initial state q_0 of $\mathcal{A}_\varphi$. Then, each iteration of the loop in Lines 3–10: computes the set $\mathcal{Q}_{j+1}$ of states which can be accessed from states in $\mathcal{Q}_j$ and from which $\mathcal{A}_\varphi$ has an accepting run (Line 4), computes $a_j = \sigma_\varphi(j)$ (Line 6), and increments j by 1 (Line 10).

If some $\mathcal{Q}_j$ is empty, then φ is unsatisfiable (Line 5) and if some $\mathcal{Q}_i$ and $\mathcal{Q}_j$ have the same elements (Line 7), the trace $a_0 \cdots a_{i-1}(a_i \cdots a_{j-1})^\omega$ coincides with σ_φ. Finally, its shortest representation (u, w) is computed in Line 8 in linear time (Crochemore and Rytter 1994).

Theorem 7. *Algorithm 1 returns the shortest representation of σ_φ, or* unsat *if φ is unsatisfiable; the algorithm terminates in time $2^{2^{O(|\varphi|)}}$.*

Proof. Note that φ is unsatisfiable if and only if $\mathcal{Q}_1$ is empty, so we can focus on satisfiable formulas.

Let (u, w) be the output of the algorithm, so it is computed in Line 8 as the shortest representation of $a_0 \cdots a_{i-1}(a_i \cdots a_{j-1})^\omega$. Hence, it suffices to show that the latter coincides with σ_φ. First, we show by induction on $k \in \mathbb{N}$ that if $q_0, q_1, \cdots$ is an accepting run of $\mathcal{A}_\varphi$, then $q_k \in Q_\ell$, where $\ell = k$ if $k < j$, and $\ell = i + ((k-i) \bmod (j-i))$ if $k \geq j$. The base case (for $k = 0$) holds trivially. For the inductive step, if $k < j$ then $q_{k-1} \in Q_{k-1}$ so, by Line 4, $q_k \in Q_k$. If $k \geq j$ $k = i + ((k-i) \bmod (j-i)) + n \cdot (j-i)$, for some $n \in \mathbb{N}$, and $q_{k-1} \in Q_\ell$ for $\ell = i + ((k-1-i) \bmod (j-i))$. If $(k-i) \bmod (j-i) \neq 0$, then $q_k \in Q_{\ell+1}$. Otherwise $q_k \in Q_i$, which completes the inductive step.

By the property proved above, for any $\sigma \in L(\mathcal{A}_\varphi)$ and $k < j$, after reading $\sigma[0, k]$, the automaton $\mathcal{A}_\varphi$ is in a state from the set Q_k. Since $\sigma_\varphi(k) = \bigcap_{\sigma \in L(\mathcal{A}_\varphi)} \sigma(k)$, Line 6 implies that $\sigma_\varphi(k) = a_k$ if $k < j$. For $k \geq j$ we need to show that $\sigma_\varphi(k) = a_\ell$, where $\ell = i + ((k-i) \bmod (j-i))$. This, however, follows from the fact that after reading $\sigma[0, k]$, the automaton $\mathcal{A}_\varphi$ is in a state from the set Q_ℓ.

For the doubly exponential bound on the runtime, we observe that $\mathcal{A}_\varphi$ is exponentially large (Vardi and Wolper 1994), namely $|\mathcal{Q}| = O(2^{|\varphi|})$, so it suffices to show that Algorithm 1 terminates in time $O(2^{|\mathcal{Q}|})$. This holds since $\mathcal{Q}_j \subseteq \mathcal{Q}$, for each $\mathcal{Q}_j$ constructed by the algorithm (see Line 4), and so, the if-condition from Line 7 yields termination after at most $2^{|\mathcal{Q}|}$ iterations of the main loop. □

We close this section by showing that Algorithm 1 is optimal, as calculating the intersection of traces for some LTL-formulas requires doubly exponential time.

Proposition 8. *There is a family of LTL-formulas φ_n, with $n \in \mathbb{N}$, such that each φ_n is of polynomial size in n and σ_{φ_n} has the shortest representation (u, w) with $|w| = 2^{2^{\Omega(n)}}$.*

Proof. We will use φ_n to encode a binary counter which is initialised with any value $m < 2^n$ and counts up to $2^n - 1$, then sets the counter to m and repeats such computations *ad infinitum*. Formula φ_n will use propositions $0, 1, \#, \triangleleft$ and each of its (minimal) satisfying traces will be of the form

$$\big(\ \mathsf{bin}_n(m)\ \#\ \mathsf{bin}_n(m+1)\ \#\ \cdots\ \#\ \mathsf{bin}_n(2^n-1)\ \triangleleft\ \big)^\omega,$$

where $\mathsf{bin}_n(m)$ is the encoding of m using n bits $b_0 b_1 \cdots b_{n-1} \in \{0,1\}^n$ with the least-significant bit first.

The intersection of the traces above, for all $0 \leq m \leq 2^n - 1$, is the trace $(\emptyset^{(n+1)\cdot \ell - 1} \triangleleft)^\omega$, with $\ell = \mathrm{lcm}\{1, \ldots, 2^n\}$. As $\mathrm{lcm}\{1, \ldots, 2^n\} \geq 2^{2^n}$ (Nair 1982), we get the required doubly exponential bound.

It remains to show how to encode our specific counter with φ_n. To this end we let φ_n be the conjunction

$$\mathsf{Segment}_n \wedge \mathsf{Invariant}_n \wedge \mathsf{End}_n \wedge \mathsf{Propagates}_n \wedge \mathsf{Counter}_n.$$

We start by defining $\mathsf{Segment}_n$ which will hold in the time points $0, n+1, 2(n+1), \ldots$, corresponding to the positions of the least significant bits b_0:

$$\mathsf{Segment}_n = \bigwedge_{0 \leq i \leq n-1} \mathsf{X}^i(0 \vee 1) \wedge \mathsf{X}^n(\triangleleft \vee \#).$$

Formula $\mathsf{Invariant}_n$ ensures that each segment is immediately (i.e., after $n+1$ time points) followed by another one:

$$\mathsf{Invariant}_n = \mathsf{G}(\mathsf{Segment}_n \rightarrow \mathsf{X}^{n+1}\mathsf{Segment}_n).$$

We use End_n to state that $\triangleleft$ ends segments with only 1s:

$$\mathsf{End}_n = \mathsf{G}(\mathsf{X}^n \triangleleft \leftrightarrow \bigwedge_{0 \leq i \leq n-1} \mathsf{X}^i 1).$$

Formula $\mathsf{Propagate}_n$ ensures that each segment ended by $\triangleleft$ is followed by a segment encoding the initial value m:

$$\mathsf{Propagate}_n = \bigwedge_{0 \leq i \leq n-1} \bigwedge_{j \in \{0,1\}} \big(\mathsf{X}^i j \rightarrow \mathsf{G}(\triangleleft \rightarrow \mathsf{X}^{i+1} j)\big).$$

Finally, we use $\mathsf{Counter}_n$ to increment the value of the binary counter in the next segment. To this end, we introduce additional formulas $\mathsf{Equal}_{n,i}$ and $\mathsf{Inc}_{n,i}$ with $0 \leq i \leq n-1$ (for definiteness we additionally let $\mathsf{Inc}_{n,n} = \top$):

$$\begin{aligned}
\mathsf{Equal}_{n,i} &= \bigwedge_{i \leq j \leq n-1} \mathsf{X}^i 1 \leftrightarrow \mathsf{X}^{i+n+1} 1, \\
\mathsf{Inc}_{n,i} &= (\mathsf{X}^i 0 \rightarrow (\mathsf{X}^{i+n+1} 1 \wedge \mathsf{Equal}_{n,i+1})) \wedge \\
&\quad\ (\mathsf{X}^i 1 \rightarrow (\mathsf{X}^{i+n+1} 0 \wedge \mathsf{Inc}_{n,i+1})), \\
\mathsf{Counter}_n &= \mathsf{G}(\mathsf{X}^n \# \rightarrow \mathsf{Inc}_{n,0}).
\end{aligned}$$

The size of φ_n is polynomial in n, as required. □

In the next sections we address the negative result from Proposition 8 by showing Horn fragments of LTL in which computing the intersection of traces is significantly easier.

5 Materialisation in Horn Fragments

In this section, we study Horn fragments of LTL, where materialisation can be used to check satisfiability of formulas and, as we will show in Section 6, to efficiently compute the intersection of traces. In the first step of our study on Horn fragments, we determine the number of steps materialisation takes before reaching a fixpoint, which is crucial for both of the above-mentioned reasoning tasks. In particular, we will show that materialisation in $\mathrm{LTL}_{horn}^{\mathsf{G}}$, $\mathrm{LTL}_{horn}^{\mathsf{X}}$, and $\mathrm{LTL}_{horn}^{\mathsf{X},\mathsf{G}}$, takes up to $|\varphi|^2$, ω, and $\omega \cdot |\varphi|$ steps, respectively.

First we show the quadratic bound for $\mathrm{LTL}_{horn}^{\mathsf{G}}$-formulas; we recall that $\sigma_\mathcal{D}$ is the least trace satisfying $\mathcal{D}$.

Theorem 9. *For every $\mathrm{LTL}_{horn}^{\mathsf{G}}$-formula φ, composed of a program Π and dataset $\mathcal{D}$, we have that $\mathfrak{C}_{\Pi,\mathcal{D}} = T_\Pi^{|\varphi|^2}(\sigma_\mathcal{D})$.*

Proof sketch. We start by dividing the time line into segments $[0, k_1)$, $[k_1, k_1]$, (k_1, k_2), $\dots$, $[k_\ell, k_\ell]$, (k_ℓ, ∞), where $k_1 < \dots < k_\ell$ are all the time points in $\mathcal{D}$ (i.e., the numbers k from expressions $\mathsf{X}^k p$ in $\mathcal{D}$). Since Π mentions only G among temporal operators, we can show by transfinite induction on ordinals α that $T^\alpha_\Pi(\sigma_\mathcal{D})$ satisfies the same propositions in all time points belonging to the same segment and that $T^\alpha_\Pi(\sigma_\mathcal{D})$ does not satisfy any proposition in the segment $[0, k_1)$. Thus each application of T_Π (before reaching $\mathfrak{C}_{\Pi,\mathcal{D}}$) derives at least one proposition in one of the $\leq 2 \cdot |\mathcal{D}|$ segments. There are at most $\frac{|\Pi|}{3}$ propositions that can be derived (each proposition is derived by some rule and each rule has at least 3 symbols), so $\mathfrak{C}_{\Pi,\mathcal{D}} = T^\alpha_\Pi(\sigma_\mathcal{D})$, for $\alpha = 2 \cdot |\mathcal{D}| \cdot \frac{|\Pi|}{3}$, and therefore, $\alpha \leq |\varphi|^2$. □

Moreover, we can show that the bound from Theorem 9 is optimal by constructing a formula for which materialisation takes a quadratic number of steps. Indeed, consider a dataset $\mathcal{D}$ consisting of $\mathsf{X}^1 r_1$ and $\mathsf{X}^i e_i$ for each $i \in \{1, \dots, k\}$, as well as a program Π with rules:

$$\begin{aligned} p_1 &\leftarrow e_i \wedge r_i, && \text{for all } i \in \{1, \dots, k\}, \\ p_{i+1} &\leftarrow p_i, && \text{for all } i \in \{1, \dots, \ell - 1\}, \\ \mathsf{G} r_{i+1} &\leftarrow p_\ell \wedge e_i, && \text{for all } i \in \{1, \dots, k - 1\}. \end{aligned}$$

Materialisation of Π and $\mathcal{D}$ takes $k \cdot (\ell + 1)$ steps. The first $\ell+1$ steps derive facts at 1; the first step derives p_1 at 1 (using the first rule of Π), the next $\ell - 1$ steps derive $p_2, \dots, p_\ell$ at 1 (the second rule), and then $\mathsf{G} r_2$ is derived at 1 (the third rule). The next $\ell + 1$ steps derive propositions at 2, and this process repeats k times, giving a quadratic number of steps.

In the remaining part of this section we show bounds on the number of materialisation steps for $\text{LTL}^{\mathsf{X}}_{horn}$ and $\text{LTL}^{\mathsf{X},\mathsf{G}}_{horn}$. For this, the following notion will be useful.

Definition 10. *For an $\text{LTL}^{\mathsf{G},\mathsf{X}}_{horn}$ program Π, dataset $\mathcal{D}$, and an ordinal α, we let $\mathsf{rules}^\alpha_{\Pi,\mathcal{D}}$ be the set of all rules $r \in \Pi$ for which there is a time point i such that $T^\alpha_\Pi(\sigma_\mathcal{D}), i \models \mathsf{body}(r)$, but $T^\beta_\Pi(\sigma_\mathcal{D}), i \not\models \mathsf{body}(r)$ for all $\beta < \alpha$.*

Intuitively $\mathsf{rules}^\alpha_{\Pi,\mathcal{D}}$ contains all rules in Π which 'fire' in step α of materialisation at some time point i, and which did not 'fire' at this i in any previous step. We show next that if α is a limit ordinal, all such rules mention G in the body.

Lemma 11. *If α is a limit ordinal, then each rule in $\mathsf{rules}^\alpha_{\Pi,\mathcal{D}}$ mentions G in the body, where Π is any $\text{LTL}^{\mathsf{G},\mathsf{X}}_{horn}$ program and $\mathcal{D}$ is any dataset.*

Proof. Suppose towards a contradiction that there exists $r \in \mathsf{rules}^\alpha_{\Pi,\mathcal{D}}$ which does not mention G in the body. Since $r \in \mathsf{rules}^\alpha_{\Pi,\mathcal{D}}$, by Definition 10 there exists i such that $T^\alpha_\Pi(\sigma_\mathcal{D}), i \models \mathsf{body}(r)$, but $T^\beta_\Pi(\sigma_\mathcal{D}), i \not\models \mathsf{body}(r)$, for all ordinals $\beta < \alpha$. As r does not mention G in the body, there is a unique minimal (finite) dataset $\mathcal{D}_r$ such that $\sigma_{\mathcal{D}_r} \subseteq T^\alpha_\Pi(\sigma_\mathcal{D})$ and $\sigma_{\mathcal{D}_r}, i \models \mathsf{body}(r)$. Since α is a limit ordinal, by the definition (see Section 2), $T^\alpha_\Pi(\sigma_\mathcal{D}) = \bigcup_{\beta<\alpha} T^\beta_\Pi(\sigma_\mathcal{D})$, and so, for each $\mathsf{X}^k p \in \mathcal{D}_r$ there is an ordinal $\beta < \alpha$ such that $T^\beta_\Pi(\sigma_\mathcal{D}) \models \mathsf{X}^k p$. Let γ be the largest among these β, so $\gamma < \alpha$. Moreover, as for each ordinal β we have $T^\beta_\Pi(\sigma_\mathcal{D}) \subseteq T^{\beta+1}_\Pi(\sigma_\mathcal{D})$, we obtain that $T^\gamma_\Pi(\sigma_\mathcal{D}) \models \mathcal{D}_r$. Hence, $T^\gamma_\Pi(\sigma_\mathcal{D}), i \models \mathsf{body}(r)$, raising a contradiction. □

Next, we use the above lemma to show a bound on the number of materialisation steps in $\text{LTL}^{\mathsf{X},\mathsf{G}}_{horn}$ depending on the number k of rules which mention G in the body. Note that $\text{LTL}^{\mathsf{X}}_{horn}$ is a special case of $\text{LTL}^{\mathsf{X},\mathsf{G}}_{horn}$ with $k = 0$, so the result below provides also a bound for $\text{LTL}^{\mathsf{X}}_{horn}$.

Theorem 12. *For every $\text{LTL}^{\mathsf{G},\mathsf{X}}_{horn}$ program Π and every dataset $\mathcal{D}$, we have $\mathfrak{C}_{\Pi,\mathcal{D}} = T^{\omega\cdot(k+1)}_\Pi(\sigma_\mathcal{D})$, where k is the number of rules in Π which mention G in their bodies.*

Proof sketch. To show that $\mathfrak{C}_{\Pi,\mathcal{D}} = T^{\omega\cdot(k+1)}_\Pi(\sigma_\mathcal{D})$, it suffices to prove that $\mathsf{rules}^{\omega\cdot(k+1)}_{\Pi,\mathcal{D}} = \emptyset$. To this end, we will show that for any limit ordinal α and ordinal $\beta < \alpha$, we have $\mathsf{rules}^\alpha_{\Pi,\mathcal{D}} \cap \mathsf{rules}^\beta_{\Pi,\mathcal{D}} = \emptyset$. This will finish the proof as, by Lemma 11, for each limit ordinal α the set $\mathsf{rules}^\alpha_{\Pi,\mathcal{D}}$ contains only rules with G in the body, and so, our result above implies that there are at most k limit ordinals α for which $\mathsf{rules}^\alpha_{\Pi,\mathcal{D}} \neq \emptyset$. Thus, $\mathsf{rules}^{\omega\cdot(k+1)}_{\Pi,\mathcal{D}} = \emptyset$.

To show the missing part of the proof, we suppose towards a contradiction that for some limit ordinal α and an ordinal $\beta < \alpha$, there is $r \in \mathsf{rules}^\alpha_{\Pi,\mathcal{D}} \cap \mathsf{rules}^\beta_{\Pi,\mathcal{D}}$. Since $r \in \mathsf{rules}^\alpha_{\Pi,\mathcal{D}}$, there is a time point i such that $T^\alpha_\Pi(\sigma_\mathcal{D}), i \models \mathsf{body}(r)$ but $T^\beta_\Pi(\sigma_\mathcal{D}), i \not\models \mathsf{body}(r)$. As $r \in \mathsf{rules}^\beta_{\Pi,\mathcal{D}}$, there is also a time point j such that $T^\beta_\Pi(\sigma_\mathcal{D}), j \models \mathsf{body}(r)$, and clearly $i \neq j$. Now, by the fact that $T^\beta_\Pi(\sigma_\mathcal{D}) \subseteq T^\alpha_\Pi(\sigma_\mathcal{D})$, we can show that there exists a unique minimal (finite) dataset $\mathcal{D}_r$ such that $\sigma_{\mathcal{D}_r} \subseteq T^\alpha_\Pi(\sigma_\mathcal{D})$ and $\sigma_{\mathcal{D}_r} \cup T^\beta_\Pi(\sigma_\mathcal{D}), i \models \mathsf{body}(r)$. Next, by the same argument as in the proof of Lemma 11, we can use the fact that $T^\alpha_\Pi(\sigma_\mathcal{D}) = \bigcup_{\gamma<\alpha} T^\gamma_\Pi(\sigma_\mathcal{D})$ to show that there exists $\gamma < \alpha$ such that $T^\gamma_\Pi(\sigma_\mathcal{D}), i \models \mathsf{body}(r)$. As $\gamma < \alpha$, it contradicts the assumption that $r \in \mathsf{rules}^\alpha_{\Pi,\mathcal{D}}$. □

Observe that we can replace $k+1$ in Theorem 12 with $|\varphi|$. Indeed, $k + 1 \leq |\varphi|$ unless $|\varphi| = 0$, but in this border case materialisation takes 0 steps. Note also that in $\text{LTL}^{\mathsf{X}}_{horn}$ we have $k = 0$, thus materialisation takes up to ω steps.

To show that these bounds are optimal, for any k we can construct Π and $\mathcal{D}$ whose materialisation takes $\omega \cdot (k + 1)$ steps. Indeed, let $\mathcal{D} = \{\mathsf{X}^0 p, \mathsf{X}^0 q_1\}$ and let Π have rules:

$$\begin{aligned} \mathsf{X} q_i &\leftarrow q_i, && \text{for all } i \in \{1, \dots, k + 1\}, \\ q_{i+1} &\leftarrow p \wedge \mathsf{G} q_i, && \text{for all } i \in \{1, \dots, k\}. \end{aligned}$$

After ω applications of the first rule we get $\mathsf{G} q_1$ at 0, allowing us to apply the second rule and to get q_2 at 0. This repeats $k+1$ times giving rise to $\omega \cdot (k+1)$ steps of materialisation.

We summarise the results from this sections below.

Corollary 13. *Let φ be an $\text{LTL}^{\mathsf{G},\mathsf{X}}_{horn}$-formula consisting of a program Π and a dataset $\mathcal{D}$. Then, the following hold:*

$$\begin{aligned} \mathfrak{C}_{\Pi,\mathcal{D}} &= T^{\omega\cdot|\varphi|}_\Pi(\sigma_\mathcal{D}), && \text{if } \varphi \text{ is an } \text{LTL}^{\mathsf{G},\mathsf{X}}_{horn}\text{-formula,} \\ \mathfrak{C}_{\Pi,\mathcal{D}} &= T^{\omega}_\Pi(\sigma_\mathcal{D}), && \text{if } \varphi \text{ is an } \text{LTL}^{\mathsf{X}}_{horn}\text{-formula,} \\ \mathfrak{C}_{\Pi,\mathcal{D}} &= T^{|\varphi|^2}_\Pi(\sigma_\mathcal{D}), && \text{if } \varphi \text{ is an } \text{LTL}^{\mathsf{G}}_{horn}\text{-formula.} \end{aligned}$$

Our results improve the previously established bound ω_1 (Brandt et al. 2017) and are crucial for our algorithm in the next section, which optimally computes the intersection of traces for $\mathrm{LTL}_{horn}^{\mathsf{G}}$-, $\mathrm{LTL}_{horn}^{\mathsf{X}}$-, and $\mathrm{LTL}_{horn}^{\mathsf{G},\mathsf{X}}$-formulas.

6 Algorithm for Horn Fragments

We will use results from Section 5 to compute the intersection of traces for formulas of $\mathrm{LTL}_{horn}^{\mathsf{G},\mathsf{X}}$ (as well as of $\mathrm{LTL}_{horn}^{\mathsf{G}}$ and $\mathrm{LTL}_{horn}^{\mathsf{X}}$) or return unsat if the formula is unsatisfiable. Our algorithm takes as an input a formula in *normal form* in which the program has only rules of the forms:

$$\mathsf{X}q \leftarrow p, \quad \mathsf{G}q \leftarrow p, \quad q \leftarrow \mathsf{X}p, \quad q \leftarrow \mathsf{G}p, \quad r \leftarrow p \wedge q,$$

for $p, q \in \mathsf{PROP}$ and $r \in \mathsf{PROP} \cup \{\bot\}$. Each $\mathrm{LTL}_{horn}^{\mathsf{X},\mathsf{G}}$-formula φ can be transformed in logarithmic space into a formula φ' in normal form such that $\sigma_{\varphi'}$ is a conservative extension of σ_φ, so the shortest representation (u, w) of σ_φ can be easily computed from the shortest representation (u', w') of $\sigma_{\varphi'}$. To construct φ', we introduce rules with fresh propositions, for example, $\mathsf{XG}q \leftarrow p \wedge q \wedge r$ is translated into $p_1 \leftarrow p \wedge q$, $p_2 \leftarrow p_1 \wedge r$, $\mathsf{X}p_3 \leftarrow p_2$, and $\mathsf{G}q \leftarrow p_3$; our construction guarantees also that φ' is in the same class ($\mathrm{LTL}_{horn}^{\mathsf{G},\mathsf{X}}$, $\mathrm{LTL}_{horn}^{\mathsf{G}}$, or $\mathrm{LTL}_{horn}^{\mathsf{X}}$) as φ.

Given a formula φ in normal form, which consists of a program Π and a dataset $\mathcal{D}$, Algorithm 2 shows how to efficiently compute the shortest representation of σ_φ. In each iteration of the main loop (Lines 1–9), the algorithm keeps extending $\mathcal{D}$. In particular, $\mathcal{D}$ is extended in Line 5 using $\mathsf{ApplyRules}(\Pi, \mathcal{D})$ which implements a one-step-application of T_Π to $\sigma_\mathcal{D}$. If some $\bot$-rules 'fires', the input formula is unsatisfiable, so the algorithm returns unsat in Line 4. This, however, does not prevent from applying rules infinitely many times. Hence we keep applying them only until $\mathcal{D}$ enjoys a specific *saturation* property (see Definition 14), which allows us to compute the *period* $[i, j]$ of $\mathcal{D}$ (Line 6). Next, in Line 7, the crucial part of the algorithm occurs, where $\mathcal{D}$ and $[i, j]$ are used to compute an additional set $\mathcal{D}_\infty$ of formulas which follow from Π and $\mathcal{D}$, but which may not be derivable in any finite number of $\mathsf{ApplyRules}$ applications. After adding $\mathcal{D}_\infty$ to $\mathcal{D}$ (Line 8) we may be able to derive further formulas, and so, we repeat the main loop until $\mathcal{D}_\infty$ is empty. Once the loop terminates, we can show that σ_φ coincides with the trace $\sigma_\mathcal{D}[0, i-1]\,(\sigma_\mathcal{D}[i, j])^\omega$, so it remains to compute its shortest representation (Line 10).

The crucial part of the algorithm is the definition of a $(\Pi, k_{\max})$-saturated set and its period, which are used to compute $\mathcal{D}_\infty$.

Definition 14. *Let $\mathcal{D}$ be a set of formulas of the forms $\mathsf{X}^k p$ and $\mathsf{X}^k \mathsf{G}p$, let Π be a program, and let $k_{\max}$ be a time point. We say that $\mathcal{D}$ is* $(\Pi, k_{\max})$-saturated *if there is an interval $[i, j]$ with $k_{\max} < i$ such that*

- *$\sigma_\mathcal{D}, i \models \lambda$ if and only if $\sigma_\mathcal{D}, j+1 \models \lambda$, for each temporal atom λ occurring in Π, and*
- *$\sigma_\mathcal{D}, k \models \lambda_1 \wedge \cdots \wedge \lambda_n \rightarrow \lambda_{n+1}$, for each $k \in [0, j]$ and each rule $\lambda_{n+1} \leftarrow \lambda_1 \wedge \cdots \wedge \lambda_n$ in Π.*

We let the period *of $\mathcal{D}$ be the interval with the least j among $[i, j]$'s satisfying the properties above.*

Algorithm 2: Intersection of traces for $\mathrm{LTL}_{horn}^{\mathsf{G},\mathsf{X}}$

Input: an $\mathrm{LTL}_{horn}^{\mathsf{G},\mathsf{X}}$-formula φ in normal form consisting of a program Π and a dataset $\mathcal{D}$
Output: the shortest representation (u, w) of σ_φ if φ is satisfiable, or unsat otherwise

1 **repeat**
2 $\quad k_{\max} \coloneqq$ maximal k such that X^k occurs in $\mathcal{D}$;
3 $\quad$ **while** $\mathcal{D}$ *is not* $(\Pi, k_{\max})$*-saturated* **do**
4 $\quad\quad$ **if** $\sigma_\mathcal{D}, i \models \mathsf{body}(r)$*, for some $i \in \mathbb{N}$ and a $\bot$-rule $r \in \Pi$* **then return** unsat;
5 $\quad\quad$ **else** $\mathcal{D} \coloneqq \mathsf{ApplyRules}(\Pi, \mathcal{D})$;
6 $\quad [i, j] \coloneqq$ period of $\mathcal{D}$;
7 $\quad \mathcal{D}_\infty \coloneqq \{\mathsf{X}^i \mathsf{G}p \mid \mathsf{G}p$ occurs in the body of Π, $\sigma_\mathcal{D}, k \models p$ for all $k \in [i, j]$, and $\sigma_\mathcal{D}, i \not\models \mathsf{G}p\}$;
8 $\quad \mathcal{D} \coloneqq \mathcal{D} \cup \mathcal{D}_\infty$;
9 **until** $\mathcal{D}_\infty = \emptyset$;
10 $(u, w) \coloneqq$ the shortest representation of $\sigma_\mathcal{D}[0, i-1]\,(\sigma_\mathcal{D}[i, j])^\omega$;
11 **return** (u, w);

Note that the period in Definition 14 is uniquely defined; indeed, if there were two distinct periods $[i, j]$ and $[i', j]$ with $i < i'$, then, by the first item in the definition, the same temporal atoms λ from Π would be satisfied at i and i'. Hence, $[i, i'-1]$, with $i'-1 < j$, would satisfy the properties of a period, which contradicts the minimality of j.

In the next theorems we show that Algorithm 2 is correct and worst-case optimal. For any $s \in \{k_{\max}, i, j, \mathcal{D}_\infty\}$, and any iteration $\ell \in \mathbb{N}$ of the main loop, we will use s^ℓ for the value of s at the end of the ℓth iteration. We also let $\mathcal{D}_b^\ell$ and $\mathcal{D}_e^\ell$ be the values of $\mathcal{D}$ in the ℓth iteration before and after executing the 'while' loop, respectively.

Theorem 15. *Algorithm 2 terminates in time $O(2^{|\varphi|})$ in general and in time $O(|\varphi|^4)$ if φ is an $\mathrm{LTL}_{horn}^{\mathsf{G}}$-formula.*

Proof. The most significant component in the computation is the number A of applications of $\mathsf{ApplyRules}$ (taking place in Line 5). Each such application adds to $\mathcal{D}$ at least one new formula of the form $\mathsf{X}^k p$ or $\mathsf{X}^k \mathsf{G}p$, which did not follow from $\mathcal{D}$ so far. Thus, $A \leq k_{\max}^{f+1} \cdot 2|\varphi|$, where $k_{\max}^f$ is the value of $k_{\max}$ in the final iteration f of the main loop, whereas we let $k_{\max}^{f+1}$ be the value as if it was computed after the final iteration. We will show that $k_{\max}^{f+1} = O(2^{|\varphi|})$. For this, we observe that $k_{\max}^{\ell+1} \leq k_{\max}^\ell + (k_{\max}^\ell + 2^{|\varphi|} + 1) \cdot 2|\varphi|$ for each iteration ℓ, since $(k_{\max}^\ell + 2^{|\varphi|} + 1) \cdot 2|\varphi|$ is the maximal number of applications of $\mathsf{ApplyRules}$ till saturation is reached, and each such application can increase the maximal number in $\mathcal{D}$ by at most 1. We can find a constant $c \in \mathbb{N}$ such that, for any ℓ, we have $2^{|\varphi|} + 1 \leq c \cdot k_{\max}^\ell$, so $k_{\max}^{\ell+1} \leq k_{\max}^\ell \cdot (1 + 4c^2 k_{\max}^\ell) \leq 5(c^2 k_{\max}^\ell)^2$. Since $k_{\max}^1 \leq |\varphi|$ we obtain that $k_{\max}^{f+1} = O(|\varphi|^{f+1})$. It remains to show that $f = O(|\varphi|)$. In particular, $f < k + 1$, where $k \leq |\varphi|$ is the number of body atoms $\mathsf{G}p$ in Π which mention distinct propositions p. This is because there can be at most k distinct and non-empty sets $\mathcal{D}_\infty$ in the run; otherwise we

had $\mathsf{X}^i\mathsf{G}p \in \mathcal{D}^\ell_\infty$ and $\mathsf{X}^{i'}\mathsf{G}p \in \mathcal{D}^{\ell'}_\infty$ for some $\ell' > \ell$ and $i, i' \in \mathbb{N}$. However, $\ell' > \ell$ implies $i' \geq i$, and so $\mathsf{X}^i\mathsf{G}p$ entails $\mathsf{X}^{i'}\mathsf{G}p$ contradicting $\mathsf{X}^{i'}\mathsf{G}p \in \mathcal{D}^{\ell'}_\infty$ (see Line 7).

We showed an exponential bound on $k^\ell_{\max}$, so each $\mathcal{D}^\ell_b$ (and $\mathcal{D}^\ell_e$) is also exponentially bounded. We observe that each application of $\mathsf{ApplyRules}(\Pi, \mathcal{D})$ is polynomial in the size of Π and current $\mathcal{D}$, and so is checking saturation of $\mathcal{D}$ (Line 3) and computing its period (Line 6). This allows us to conclude that the algorithm runs in $O(2^{|\varphi|})$ time.

If φ is an $\mathrm{LTL}^{\mathsf{G}}_{horn}$-formula, there is only one iteration of the algorithm's main loop and, as we have shown in Corollary 13, the number of $\mathsf{ApplyRules}$ calls in this iteration is in $O(|\varphi|^2)$. Moreover, it follows from the proof of Theorem 9 that each $\mathsf{X}^k p$ and $\mathsf{X}^k\mathsf{G}p$ in $\mathcal{D}$ has k that occurs in the initial $\mathcal{D}^1_b$. Thus, there are linearly many such k's and linearly many rules that can be applied, so each call of $\mathsf{ApplyRules}$ requires $O(|\varphi|^2)$ time. Each saturation check and computing the period (which are actually not needed in $\mathrm{LTL}^{\mathsf{G}}_{horn}$) is also in $O(|\varphi|^2)$. Thus, the overall runtime is $O(|\varphi|^4)$. □

Theorem 16. *Algorithm 2 returns the shortest representation of σ_φ, or unsat if φ is unsatisfiable.*

Proof sketch. First, we assume that φ has no $\bot$-rules, so it is satisfiable and the algorithm returns some (u, w). We will show that $\sigma_\varphi = uw^\omega$. For any iteration ℓ we let σ^ℓ be $\sigma_{\mathcal{D}^\ell_e}[0, i^\ell{-}1](\sigma_{\mathcal{D}^\ell_e}[i^\ell, j^\ell])^\omega$, thus $uw^\omega = \sigma^f$, for f being the final iteration. Hence, we want to show that $\sigma_\varphi = \sigma^f$.

($\subseteq$) As φ is a satisfiable $\mathrm{LTL}^{\mathsf{X},\mathsf{G}}_{horn}$-formula, σ_φ is the least trace of φ. Hence, to prove that $\sigma_\varphi \subseteq \sigma^f$, it suffices to show that σ^f is a trace of φ. By the structure of the algorithm, no formula is ever removed from $\mathcal{D}$, so $\sigma^f, 0 \models \mathcal{D}$. Thus, it remains to prove that $\sigma^f, 0 \models \Pi$, that is, for each time point t and each rule $\lambda_3 \leftarrow \lambda_1 \wedge \lambda_2$ in Π, if $\sigma^f, t \models \lambda_1 \wedge \lambda_2$ then $\sigma^f, t \models \lambda_3$. We observe that $\sigma_{\mathcal{D}^f_e}$ satisfies this property for all $t \in [0, j^f]$, since it is $(\Pi, k^f_{\max})$-saturated. Hence, it suffices to show that for each t and atom λ occurring in Π the following statements hold:

(i) if $t < i^f$, then $\sigma^f, t \models \lambda$ iff $\sigma_{\mathcal{D}^f_e}, t \models \lambda$, and

(ii) if $t \geq i^f$, then $\sigma^f, t \models \lambda$ iff $\sigma_{\mathcal{D}^f_e}, t' \models \lambda$, where t' is the unique point in $[i^f, j^f]$ such that $j^f{+}1{-}i^f$ divides $t{-}t'$.

As φ is in normal form, each λ is of the form p, $\mathsf{G}p$, or $\mathsf{X}p$, so we conduct a proof by cases. If $\lambda = p$, Statements (i) and (ii) hold since $\sigma^f = \sigma_{\mathcal{D}^f_e}[0, i^f{-}1](\sigma_{\mathcal{D}^f_e}[i^f, j^f])^\omega$. If $\lambda = \mathsf{X}p$, Statement (i) holds for the same reason, and so does Statement (ii) for $t' \neq j^f$. For $t' = j^f$, the following are equivalent: (1) $\sigma^f, t \models \mathsf{X}p$, (2) $\sigma^f, t{+}1 \models p$, (3) $\sigma_{\mathcal{D}^f_e}, i^f \models p$ (as $(t{+}1)' = i^f$), (4) $\sigma_{\mathcal{D}^f_e}, j^f{+}1 \models p$ (as $[i^f, j^f]$ is the period of $\mathcal{D}^f_e$), and (5) $\sigma_{\mathcal{D}^f_e}, j^f \models \mathsf{X}p$. If $\lambda = \mathsf{G}p$, the proof is similar, but uses the fact that $\mathcal{D}^f_\infty = \emptyset$.

($\supseteq$) Next we prove that $\sigma^f \subseteq \sigma_\varphi$. We claim that it suffices to show that ($*$) for any $\ell \leq f$ if $\sigma_{\mathcal{D}^\ell_b} \subseteq \sigma_\varphi$, then $\sigma^\ell \subseteq \sigma_\varphi$. Statement ($*$) indeed implies that $\sigma^f \subseteq \sigma_\varphi$; since $\sigma_{\mathcal{D}^1_b} \subseteq \sigma_\varphi$ and, moreover, $\sigma^\ell \subseteq \sigma_\varphi$ implies that $\sigma_{\mathcal{D}^{\ell+1}_b} \subseteq \sigma_\varphi$ (by the definition of $\mathcal{D}^\ell_\infty$, which is used to construct $\mathcal{D}^{\ell+1}_b$), by Statement ($*$) we obtain that $\sigma_{\mathcal{D}^f_b} \subseteq \sigma_\varphi$ and also $\sigma^f \subseteq \sigma_\varphi$.

To prove Statement ($*$), we fix $\ell \leq f$ and assume that $\sigma_{\mathcal{D}^\ell_b} \subseteq \sigma_\varphi$. To show that $\sigma^\ell \subseteq \sigma_\varphi$, recall that σ^ℓ is the concatenation of $\sigma_{\mathcal{D}^\ell_e}[0, i^\ell{-}1]$ and $(\sigma_{\mathcal{D}^\ell_e}[i^\ell, j^\ell])^\omega$. As $\sigma_{\mathcal{D}^\ell_b} \subseteq \sigma_\varphi$ and $\mathcal{D}^\ell_e$ is obtained by applying (several times) $\mathsf{ApplyRules}$ to $\mathcal{D}^\ell_b$, clearly $\sigma_{\mathcal{D}^\ell_e} \subseteq \sigma_\varphi$. Thus $\sigma_{\mathcal{D}^\ell_e}[0, i^\ell{-}1]$ is contained in σ_φ. To show the inclusion for the second part of σ^ℓ, we will prove inductively on $n \in \mathbb{N}$ that $\sigma_{\mathcal{D}^\ell_e}, t \models p$ implies that $\sigma_\varphi, (t{+}n{\cdot}\mathsf{per}) \models p$, for any $t \geq i^\ell$ (in particular for all $t \in [i^\ell, j^\ell]$), and $p \in \mathsf{PROP}$, where per is the number of points in the period $[i^\ell, j^\ell]$. The base (for $n = 0$) holds as $\sigma_{\mathcal{D}^\ell_e} \subseteq \sigma_\varphi$. The inductive step for $t = i^\ell$ holds by the inductive assumption and saturation of $\mathcal{D}^\ell_e$ (implying that $\sigma_{\mathcal{D}^\ell_e}$ satisfies the same propositions at i^ℓ and $j^\ell{+}1$). For $t > i^\ell$, we recall that $\mathcal{D}^\ell_e$ is obtained by applying (several times) $\mathsf{ApplyRules}$ to $\mathcal{D}^\ell_b$, so we prove inductively on $m \in \mathbb{N}$ that $T^m_\Pi(\sigma_{\mathcal{D}^\ell_b}), t' \models p'$, implies that $\sigma_\varphi, (t'{+}n{\cdot}\mathsf{per}) \models p'$, for all $t' > i^\ell$ and $p' \in \mathsf{PROP}$. For the base ($m = 0$), since $t' > k^\ell_{\max}$, we get $\sigma_{\mathcal{D}^\ell_b}, t' \models \mathsf{G}p'$, and so $\sigma_{\mathcal{D}^\ell_b} \subseteq \sigma_\varphi$ implies that $\sigma_\varphi, (t'{+}n{\cdot}\mathsf{per}) \models p'$. In the inductive step ($m > 0$) we assume without loss of generality that $T^{m-1}_\Pi(\sigma_{\mathcal{D}^\ell_b}), t' \not\models p'$, so $T^m_\Pi(\sigma_{\mathcal{D}^\ell_b}), t' \models p'$ by an application of a rule of one of the five types occurring in the normal-form Π. We can show that in each case the inductive assumption implies that $\sigma_\varphi, (t'{+}n{\cdot}\mathsf{per}) \models p'$, as required.

Now, we consider Algorithm 2 running on φ (consisting of Π and $\mathcal{D}$) which can have $\bot$-rules and we let φ' be obtained by deleting all $\bot$-rules, so $\sigma_{\varphi'} = \mathfrak{C}_{\Pi,\mathcal{D}}$.

If φ is satisfiable, then $\mathfrak{C}_{\Pi,\mathcal{D}}$ satisfies all $\bot$-rules in Π. By the proof of Inclusion ($\supseteq$), $\mathfrak{C}_{\Pi,\mathcal{D}} \models \mathcal{D}^\ell_b$ for each ℓ. Hence, no $\bot$-rule applies to any $\mathcal{D}^\ell_b$, and the algorithm never returns unsat in Line 4. Thus the algorithm behaves as if it was running on φ', and so, it returns the shortest representation of $\sigma_{\varphi'}$. Since φ is satisfiable, we get $\sigma_\varphi = \mathfrak{C}_{\Pi,\mathcal{D}} = \sigma_{\varphi'}$, so the output is also the shortest representation of σ_φ.

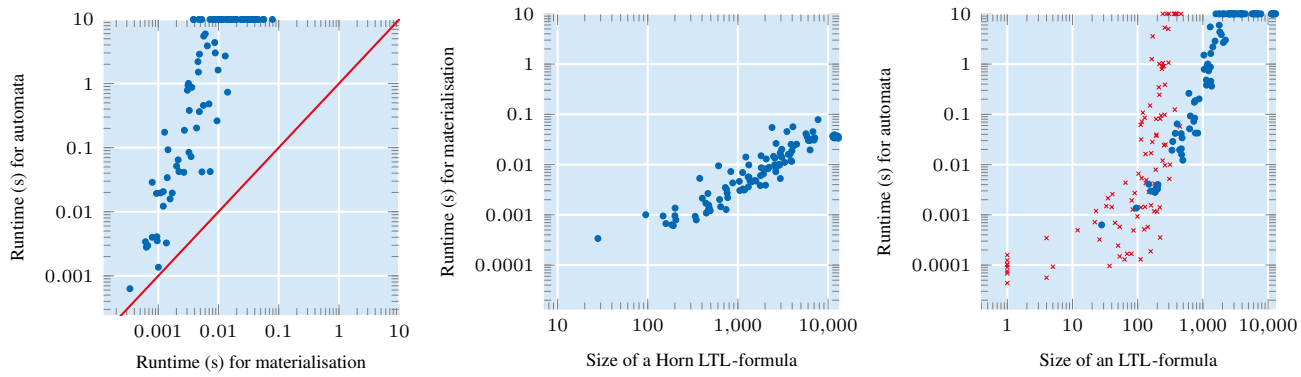
If φ is unsatisfiable, we need to show that unsat is returned. Towards a contradiction suppose that some (u, w) is returned. Thus, the same (u, w) is returned if the algorithm runs on φ', and so $uw^\omega = \sigma_{\varphi'} = \mathfrak{C}_{\Pi,\mathcal{D}}$. Since φ is unsatisfiable, some of its $\bot$-rules is not satisfied in $\mathfrak{C}_{\Pi,\mathcal{D}}$. Thus, in the last iteration the algorithm (running on φ) outputs unsat in Line 4. This, however, contradicts the assumption that (u, w) is returned. □

We observe that $\mathrm{LTL}^{\mathsf{G},\mathsf{X}}_{horn}$, and even $\mathrm{LTL}^{\mathsf{X}}_{horn}$, allows us to simulate a binary counter (similarly to the proof of Proposition 8), so the exponential blow-up is unavoidable. In the next section we experimentally evaluate of our algorithms.

7 Experimental Evaluation

In this section, we will discuss the implementation of our algorithms and their experimental evaluation.

Implementation. We implemented both of our algorithms for computing the intersection of traces, namely



(a) (b) (c)

Figure 2: Evaluation of our automata-based Algorithm 1 and materialisation-based Algorithm 2; Figure 2(a) compares the two implementations running on Horn formulas; Figure 2(b) displays runtimes of the materialisation-based approach on Horn formulas; Figure 2(c) shows runtimes of the automata-based approach running on formulas from both the LTL (red crosses) and the Horn benchmarks (blue dots)

the automata-based Algorithm 1 for full LTL and the materialisation-based Algorithm 2 designed for $\text{LTL}_{horn}^{\mathsf{G},\mathsf{X}}$. The automata-based implementation uses the state-of-the-art `spot` platform (Duret-Lutz et al. 2022) for constructing a Büchi automata corresponding to LTL-formulas (required in Line 1 of Algorithm 1). We wrote the implementations in Python and made them, together with all the benchmarks used in the experiments, available in open access.[2]

Benchmarks. Since computing the intersection of traces is a new task introduced in this paper, there were no benchmarks designed for this setting. Moreover, to the best of our knowledge, there are no benchmarks for formulas in the Horn forms we consider in the paper. Therefore, we randomly generated formulas required for testing our implementations. Our benchmarks consist of 100 LTL-formulas and another 100 $\text{LTL}_{horn}^{\mathsf{G},\mathsf{X}}$-formulas, as described below.

For LTL-formulas we used the random generator `randltl` provided in `spot` (Duret-Lutz et al. 2022). We set the parameter for the maximal number of propositions mentioned in a formula to 5 and the parameter 'random tree size', restricting the depth of the syntax tree of a formula, to values between 150 and 250. This allowed us to generate formulas of size varying from 1 to 461.

For $\text{LTL}_{horn}^{\mathsf{G},\mathsf{X}}$-formulas we had to implement a dedicated generator which provides pairs consisting of a dataset and a program. We generated such formulas randomly based on the following parameters: the number of propositions (ranging from 1 to 50), the number of facts in the dataset for each proposition (ranging from 1 to 12), the maximal nesting of X operator in the dataset (ranging from 0 to 22), and the number of rules in the program (ranging from 1 to 100). The obtained formulas have size varying from 28 to 13014.

Experiments. We ran our automata-based implementation on all of the benchmark formulas (so we used both LTL- and $\text{LTL}_{\mathsf{G},\mathsf{X}}^{horn}$-formulas), whereas the materialisation-based implementation could be ran only on the $\text{LTL}_{\mathsf{G},\mathsf{X}}^{horn}$ benchmark. We run the implementations 10 times for each formula and recorded the average runtime. Times exceeding 10 seconds were considered timeouts. All experiments were performed on a machine with an Intel® Core™ i7-8665U CPU@1.90Ghz and 16Gb RAM under Ubuntu Linux running inside a Windows 11 Subsystem for Linux.

The obtained results are presented in Figure 2. The scatter plot in Figure 2(a) shows that the materialisation-based implementation is significantly faster on every formula and always terminates within 0.1s, whereas the automata-based implementation times out on around half of the formulas (the time out often occurs already at the stage of constructing the automaton). Therefore, it is highly beneficial to use our dedicated materialisation-based implementation whenever the input formula falls into the Horn classes we considered. Figure 2(b) shows that the materialisation-based implementation scales well in practice and does not exhibit the exponential behaviour, which is in a sharp contrast with the results in Figure 2(c) for the automata-based approach.

8 Conclusions and Future Work

Motivated by applications in temporal knowledge representation, we introduced the problem of computing all facts entailed by an LTL-formula. We showed that the problem is doubly exponential, but becomes significantly easier in Horn fragments of LTL—in some cases it is tractable. We showed two optimal algorithms solving the problem for full LTL and its Horn fragments, respectively. We have implemented our algorithms and evaluated their performance. The results show a particularly good performance of the materialisation-based approach and suggest its practical usefulness.

In future, we plan to extend our materialisation-based approach to other classes of Horn formulas (e.g., allowing for past operators and the U operator in rule bodies). It would also be interesting to try to apply our approach to other temporal logics, for instance, to real time logics, metric temporal logics, and temporal description logics.

[2]The code and all benchmarks can be obtained in open access from https://github.com/chalwidz/LTLIntersectionOfTraces.

Acknowledgments

This research was supported in whole or in part by the EPSRC projects OASIS (EP/S032347/1), ConCuR (EP/V050869/1) and UK FIRES (EP/S019111/1), the SIRIUS Centre for Scalable Data Access, and Samsung Research UK. The third author is supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 852769, ARiAT). For the purpose of Open Access, the authors have applied a CC BY public copyright licence to any Author Accepted Manuscript (AAM) version arising from this submission.

References

Abadi, M., and Manna, Z. 1989. Temporal logic programming. *J. Symb. Comput.* 8(3):277–295.

Aguado, F.; Cabalar, P.; Diéguez, M.; Pérez, G.; and Vidal, C. 2013. Temporal equilibrium logic: A survey. *J. Appl. Non-Class. Log.* 23(1-2):2–24.

Alharby, M., and Van Moorsel, A. 2017. Blockchain-based smart contracts: A systematic mapping study. arXiv:1710.06372.

Alur, R., and La Torre, S. 2004. Deterministic generators and games for LTL fragments. *ACM Trans. Comput. Log.* 5(1):1–25.

Artale, A.; Kontchakov, R.; Ryzhikov, V.; and Zakharyaschev, M. 2013. The complexity of clausal fragments of LTL. In *Proc. of LPAR*, 35–52.

Artale, A.; Kontchakov, R.; Kovtunova, A.; Ryzhikov, V.; Wolter, F.; and Zakharyaschev, M. 2021. First-order rewritability of ontology-mediated queries in linear temporal logic. *Artif. Intell.* 299:103536.

Brandt, S.; Kalayci, E. G.; Kontchakov, R.; Ryzhikov, V.; Xiao, G.; and Zakharyaschev, M. 2017. Ontology-based data access with a Horn fragment of metric temporal logic. In *Proc. of AAAI*, 1070–1076.

Bresolin, D.; Kurucz, A.; Muñoz-Velasco, E.; Ryzhikov, V.; Sciavicco, G.; and Zakharyaschev, M. 2017. Horn fragments of the Halpern-Shoham interval temporal logic. *ACM Trans. Comput. Log.* 18(3):1–39.

Chen, C.-C., and Lin, I.-P. 1993. The computational complexity of satisfiability of temporal Horn formulas in propositional linear-time temporal logic. *Inf. Process. Lett.* 45(3):131–136.

Crochemore, M., and Rytter, W. 1994. *Text algorithms*. Oxford University Press.

De Giacomo, G., and Vardi, M. Y. 2013. Linear temporal logic and linear dynamic logic on finite traces. In *Proc. of IJCAI*, 854–860.

Demri, S., and Schnoebelen, P. 2002. The complexity of propositional linear temporal logics in simple cases. *Inf. Comput.* 174(1):84–103.

Demri, S.; Goranko, V.; and Lange, M. 2016. *Temporal logics in computer science: Finite-state systems*. Cambridge University Press.

Duret-Lutz, A.; Renault, E.; Colange, M.; Renkin, F.; Aisse, A. G.; Schlehuber-Caissier, P.; Medioni, T.; Martin, A.; Dubois, J.; Gillard, C.; and Lauko, H. 2022. From Spot 2.0 to Spot 2.10: What's new? In *Proc. of CAV*, 174–187.

Fisher, M. 1991. A resolution method for temporal logic. In *Proc. of IJCAI*, volume 91, 99–104.

Gabbay, D. 1987. Modal and temporal logic programming. In Galton, A., ed., *Temporal logics and their applications*. Academic Press. 197–237.

Geatti, L.; Gigante, N.; and Montanari, A. 2021. Black: A fast, flexible and reliable LTL satisfiability checker. In *Proc. of OVERLAY*, volume 2987, 7–12.

Lécué, F. 2017. Deep dive on smart cities by scaling reasoning and interpreting the semantics of IoT. In *Proc. of EGC*, 7–8.

Li, J.; Yao, Y.; Pu, G.; Zhang, L.; and He, J. 2014. Aalta: an LTL satisfiability checker over infinite/finite traces. In *Proc. of SIGSOFT*, 731–734.

Lichtenstein, O., and Pnueli, A. 2000. Propositional temporal logics: Decidability and completeness. *Log. J. IGPL* 8(1):55–85.

Lutz, C.; Wolter, F.; and Zakharyaschev, M. 2008. Temporal description logics: A survey. In *Proc. of TIME*, 3–14.

Münz, G., and Carle, G. 2007. Real-time analysis of flow data for network attack detection. In *Proc. of IFIP/IEEE IM*, 100–108.

Nair, M. 1982. On chebyshev-type inequalities for primes. *Am. Math. Mon.* 89(2):126–129.

Nuti, G.; Mirghaemi, M.; Treleaven, P. C.; and Yingsaeree, C. 2011. Algorithmic trading. *IEEE Comput.* 44(11):61–69.

Orgun, M. A., and Ma, W. 2005. An overview of temporal and modal logic programming. In *Proc. of ICTL*, 445–479.

Pnueli, A. 1977. The temporal logic of programs. In *Proc. of FOCS*, 46–57.

Ronca, A.; Kaminski, M.; Cuenca Grau, B.; Motik, B.; and Horrocks, I. 2018. Stream reasoning in temporal Datalog. In *Proc. of AAAI*, 1941–1948.

Ryzhikov, V.; Wałęga, P. A.; and Zakharyaschev, M. 2020. Temporal ontology-mediated queries and first-order rewritability: A short course. In *Proc. of Reasoning Web. Declarative AI*, 109–148.

Sistla, A. P., and Clarke, E. M. 1985. The complexity of propositional linear temporal logics. *J. ACM* 32(3):733–749.

Vardi, M. Y., and Wolper, P. 1994. Reasoning about infinite computations. *Inf. Comput.* 115(1):1–37.

Vardi, M. Y. 2005. An automata-theoretic approach to linear temporal logic. In Möller, F., and Birtwistle, G., eds., *Logics for concurrency: Structure versus automata*. Springer. 238–266.

Vardi, M. Y. 2009. From philosophical to industrial logics. In *Proc. of ICLA*, 89–115.

Wałęga, P. A. 2023. Computational complexity of hybrid interval temporal logics. *Ann. Pure Appl. Log.* 174(1):103165.

Wałęga, P. A.; Cuenca Grau, B.; Kaminski, M.; and Kostylev, E. V. 2019. DatalogMTL: Computational complexity and expressive power. In *Proc. of IJCAI*, 1886–1892.

Wałęga, P. A.; Cuenca Grau, B.; Kaminski, M.; and Kostylev, E. V. 2020. Tractable fragments of Datalog with metric temporal operators. In *Proc. of IJCAI*, 1919–1925.

Wałęga, P. A.; Zawidzki, M.; Wang, D.; ; and Cuenca Grau, B. 2023. Materialisation-based reasoning in datalogmtl with bounded intervals. In *Proc. of AAAI*.

Wolper, P. 1985. The tableau method for temporal logic: An overview. *Log. et Anal.* 119–136.