Learning temporal formulas from examples is hard*

Corto Mascle corto.mascle@labri.fr
Max-Planck Institute for Software Systems, Kaiserslautern, Germany

Nathanaël Fijalkow nathanael.fijalkow@gmail.com
CNRS, LaBRI, Université de Bordeaux, France

Guillaume Lagarde guillaume.lagarde@gmail.com
LaBRI, University of Bordeaux, France

Abstract

We study the problem of learning linear temporal logic (LTL) formulas from examples, as a first step towards expressing a property separating positive and negative instances in a way that is comprehensible for humans. In this paper we give a complete picture of the computational complexity of this problem for all fragments of **LTL** built from the operators **F** (eventually), **G** (globally), **X** (next), $\wedge$ (conjunction), and $\vee$ (disjunction) — that is, without the until operator. Our main results are hardness results: we show that the LTL learning problem is NP-complete for almost all of these fragments. This motivates the search for efficient heuristics, and highlights the complexity of expressing separating properties in concise natural language.

1. Introduction

We are interested in the complexity of learning formulas of Linear Temporal Logic (**LTL**) from examples, in a passive scenario: from a set of positive and negative words, the objective is to construct a formula, as small as possible, which satisfies the positive words and does not satisfy the negative words. Passive learning of languages has a long history paved with negative results. Learning automata is notoriously difficult from a theoretical perspective, as witnessed by the original NP-hardness result of learning a Deterministic Finite Automaton (DFA) from examples (Gold, 1978). This line of hardness results culminates with the inapproximability result of (Pitt & Warmuth, 1993) stating that there is no polynomial time algorithm for learning a DFA from examples even up to a polynomial approximation of their size.

One approach to cope with such hardness results is to change representation, for instance replacing automata by logical formulas; their syntactic structures make them more amenable to principled search algorithms. There is a range of potential logical formalisms to choose from depending on the application domain. Linear Temporal Logic (Pnueli, 1977), which we abbreviate as **LTL**, is a prominent logic for specifying temporal properties over words, it has become a de facto standard in many fields such as model checking, program analysis, and motion planning for robotics. Verification of **LTL** specifications is routinely employed in industrial settings and marks one of the most successful applications of formal methods

*. This article is a long version of the article presented in the International Conference on Grammatical Inference (ICGI) in 2021 (Fijalkow & Lagarde, 2021). It includes much stronger and more general results than the extended abstract.

to real-life problems. A key property making **LTL** a strong candidate as a concept class is that its syntax does not include variables, contributing to the fact that **LTL** formulas are typically easy to interpret and therefore useful as explanations.

Over the past ten years learning temporal logics (of which **LTL** is the core) has become an active research area. There are many applications, let us only mention a few: program specification (Lemieux, Park, & Beschastnikh, 2015) and fault detection (Bombara, Vasile, Penedo Alvarez, Yasuoka, & Belta, 2016). We refer to (Camacho & McIlraith, 2019) for a longer discussion on the potential and actual applications of learning temporal logics.

Since learning temporal logics is a computationally hard problem, a number of different approaches have been explored. One of the first and probably most natural is leveraging SAT solvers (Neider & Gavran, 2018), which can then accommodate noisy data (Gaglione, Neider, Roy, Topcu, & Xu, 2022). Another line of work relies on their connections to automata (Camacho & McIlraith, 2019), and a third completely different idea approaches it from the lens of Bayesian inference (Kim, Muise, Shah, Agarwal, & Shah, 2019). Learning specifically tailored fragments of **LTL** yields the best results in practice (Raha, Roy, Fijalkow, & Neider, 2022). There are a number of temporal logics, and the ideas mentioned above have been extended to more expressive logics such as Property Specification Language (PSL) (Roy, Fisman, & Neider, 2020; Baharisangari, Gaglione, Neider, Topcu, & Xu, 2021), Computational Tree Logic (CTL) (Ehlers, Gavran, & Neider, 2020), and Metric Temporal Logic (MTL) (Raha, Roy, Fijalkow, Neider, & Pérez, 2024). Other paradigms have been explored to make learning temporal logics useful, such as sketching (Lutz, Neider, & Roy, 2023).

Despite this growing interest, very little is known about the computational complexity of the underlying problem; indeed the works cited above focused on constructing efficient algorithms for practical applications. The goal of this paper is to initiate the study of the complexity of learning **LTL** formulas from examples.

Our contributions. We present a complete picture of the complexity of learning **LTL** formulas from examples, where throughout this paper **LTL** refers to $\mathbf{LTL}(\mathbf{F}, \mathbf{G}, \mathbf{X}, \wedge, \vee)$: that is, *without* the until operator $\mathbf{U}$. Section 2 gives definitions.

- Section 3 gives two NP-hardness results that hold when the alphabet is part of the input: one for all fragments containing $\{\mathbf{F}, \vee\}$, and one for $\mathbf{LTL}(\mathbf{F}, \wedge)$. After this section the alphabet is fixed for the rest of the paper.

- Section 4 shows that all fragments have the *short formula property*, placing the learning problem in NP. Section 5 then identifies the *degenerate* fragments for which the learning problem is in P.

- Section 6 constructs a polynomial-time $\log(n)$-approximation algorithm for $\mathbf{LTL}(\mathbf{X}, \wedge)$.

- Section 7 proves NP-completeness for all fragments containing the next operator $\mathbf{X}$, and shows that the $\log(n)$-approximation ratio for $\mathbf{LTL}(\mathbf{X}, \wedge)$ is optimal under $\mathsf{P} \neq \mathsf{NP}$.

- Section 8 proves NP-completeness for all fragments without $\mathbf{X}$, over a fixed alphabet of size 3.

We conclude in Section 9.

Summary of results. Together, our results give a complete picture of the complexity of the **LTL** learning problem (without the until operator), as a function of the set Op of allowed operators and of whether the alphabet is fixed or part of the input. The picture is governed by a single dichotomy: for a fixed alphabet and $Op \subseteq \{\mathbf{F}, \mathbf{G}, \mathbf{X}, \wedge, \vee\}$, the $\mathbf{LTL}(Op)$ learning problem is NP-complete as soon as Op contains one of the four pairs $\{\mathbf{X}, \wedge\}$, $\{\mathbf{X}, \vee\}$, $\{\mathbf{F}, \wedge\}$, or $\{\mathbf{G}, \vee\}$, and is solvable in polynomial time otherwise. Membership in NP, which holds for all these fragments, is itself non-trivial and is obtained through a "short formula property". When the alphabet is part of the input the picture shifts only slightly: the fragment $\{\mathbf{F}, \vee\}$ also becomes NP-hard.

2. Preliminaries

Let us fix a (finite) alphabet Σ. We index words from position 1 (not 0) and the letter at position i in the word w is written $w(i)$, so $w = w(1) \dots w(\ell)$ where ℓ is the length of w, written $|w| = \ell$. We write $w[k \dots] = w(k) \dots w(\ell)$ and $w[1, k] = w(1) \dots w(k)$. To avoid unnecessary technical complications we only consider non-empty words, and let Σ^+ denote the set of (non-empty) words.

The syntax of Linear Temporal Logic (**LTL**) includes atomic formulas $c \in \Sigma$ and their negations, as well as $\top$ and $\bot$, the boolean operators $\wedge$ and $\vee$, and the temporal operators $\mathbf{X}$ (next), $\mathbf{F}$ (eventually), and $\mathbf{G}$ (globally). Note that as usually done, we work with **LTL** in negation normal form, meaning that negation is only used on atomic formulas. **We do not consider the until operator U in this paper**: all fragments and results concern subsets of $\{\mathbf{F}, \mathbf{G}, \mathbf{X}, \wedge, \vee\}$. The semantics of **LTL** over finite words is defined inductively over formulas, through the notation $w \models \varphi$ where $w \in \Sigma^+$ is a non-empty word, and φ is an **LTL** formula. The definition is given below for the atomic formulas and temporal operators $\mathbf{F}, \mathbf{G}$, and $\mathbf{X}$, with boolean operators interpreted as usual. For w of length ℓ:

- We have $w \models \top$ and $w \not\models \bot$.
- $w \models c$ if $w(1) = c$.
- $w \models \mathbf{X}\varphi$ if $\ell > 1$ and $w[2 \dots] \models \varphi$. It is called the next operator.
- $w \models \mathbf{F}\varphi$ if $w[i \dots] \models \varphi$ for some $i \in [1, \ell]$. It is called the eventually operator.
- $w \models \mathbf{G}\varphi$ if $w[i \dots] \models \varphi$ for all $i \in [1, \ell]$. It is called the globally operator.

We say that w satisfies φ when $w \models \varphi$ is true. It is sometimes useful to write $w, i \models \varphi$ to mean $w[i \dots] \models \varphi$. We consider fragments of **LTL** by specifying which boolean connectives and temporal operators are allowed. For instance $\mathbf{LTL}(\mathbf{X}, \wedge)$ is the set of all **LTL** formulas using only atomic formulas, conjunctions, and the next operator. In this paper, the full logic is $\mathbf{LTL} = \mathbf{LTL}(\mathbf{F}, \mathbf{G}, \mathbf{X}, \wedge, \vee)$ (without the until operator $\mathbf{U}$).

More generally, for a set of operators $Op \subseteq \{\mathbf{F}, \mathbf{G}, \mathbf{X}, \wedge, \vee\}$, we write $\mathbf{LTL}(Op)$ for the logic using operators from Op. The size of a formula is the size of its syntactic tree. We say that two formulas are equivalent if they have the same semantics, and we write $\varphi \equiv \psi$ to say that φ and ψ are equivalent.

The LTL learning problem. A sample is a pair (P, N) where $P = \{u_1, \dots, u_n\}$ is a set of positive words and $N = \{v_1, \dots, v_m\}$ a set of negative words. Without loss of generality we can assume that $n = m$ (adding duplicate identical words to have an equal number of positive and negative words). The **LTL** learning decision problem is:

INPUT:	a sample (P, N) and $k \in \mathbb{N}$,
QUESTION:	does there exist an **LTL** formula φ of size at most k such that for all $u \in P$, we have $u \models \varphi$, and for all $v \in N$, we have $v \not\models \varphi$?

In that case we say that φ separates P from N, or simply that φ is a separating formula if the sample is clear from the context. The **LTL** learning problem is analogously defined for any fragment of **LTL**.

Parameters for complexity analysis. The three important parameters for the complexity of the **LTL** learning problem are: n the number of words, ℓ the maximum length of the words, and k the desired size for the formula. As we will see, the size of the alphabet, if not fixed, is another important parameter. We will consider two settings: either the alphabet Σ is fixed, or it is part of the input. The size of the alphabet is $|\Sigma|$, the number of letters.

Representation. The words given as input are represented in a natural way: we work with the RAM model with word size $\log(|\Sigma|+n+\ell)$, which allows us to write a letter in each cell and to manipulate words and positions in a natural way. We write $|P|$ for $\sum_{j \in [1,n]} |u_j|$ and similarly $|N| = \sum_{j \in [1,n]} |v_j|$. The size of a sample is $|P| + |N|$.

We emphasise a subtlety on the representation of k: it can be given in binary (a standard assumption) or in unary. In the first case, the input size is $O(n \cdot \ell + \log(k))$, so the formula φ we are looking for may be exponential in the input size! This means that it is not clear a priori that the **LTL** learning problem is in NP. Opting for a unary encoding, the input size becomes $O(n \cdot \ell + k)$, and in that case an easy argument shows that the **LTL** learning problem is in NP. We follow the standard representation: k is given in binary, and therefore it is not immediate that the **LTL** learning problem is in NP.

A naive algorithm. Let us start our complexity analysis of the learning **LTL** problem by constructing a naive algorithm for the whole logic.

Theorem 1. *There exists an algorithm for solving the $\mathbf{LTL}(\mathbf{F}, \mathbf{G}, \mathbf{X}, \wedge, \vee)$ learning problem in time and space $O(|\Sigma| + exp(k) \cdot n \cdot \ell)$, where $exp(k)$ is exponential in k.*

Notice that the dependence of the algorithm presented in Theorem 1 is linear in n and ℓ, and it is exponential only in k, but since k is represented in binary this is a doubly-exponential algorithm.

Proof. For a formula $\varphi \in \mathbf{LTL}$, we write $\langle\varphi\rangle : P \cup N \to \{0,1\}^\ell$ for the function defined by

$$\langle\varphi\rangle(w)(i) = \begin{cases} 1 & \text{if } w[i\dots] \models \varphi, \\ 0 & \text{if } w[i\dots] \not\models \varphi, \end{cases}$$

for $w \in P \cup N$.

Note that φ is separating if and only if $\langle\varphi\rangle(u)(1) = 1$ and $\langle\varphi\rangle(v)(1) = 0$ for all $u \in P, v \in N$. The algorithm simply consists in enumerating all formulas φ of **LTL** of size at most k inductively, constructing $\langle\varphi\rangle$, and checking whether φ is separating. Initially, we construct $\langle a\rangle$ for all $a \in \Sigma$, and then once we have computed $\langle\varphi\rangle$ and $\langle\psi\rangle$, we can compute $\langle\mathbf{F}\varphi\rangle$, $\langle\mathbf{G}\varphi\rangle$, $\langle\mathbf{X}\varphi\rangle$, $\langle\varphi \wedge \psi\rangle$, and $\langle\varphi \vee \psi\rangle$ in time $O(n \cdot \ell)$. To conclude, we note that the number of formulas of **LTL** of size at most k is exponential in k. □

This approach may still be suitable for small values of k. Later, we will show that the problem is in NP by showing that if a separating formula exists then there is one of polynomial size in n and ℓ. As a consequence, we can assume without loss of generality that k is polynomial in n and ℓ, making the above algorithm singly exponential.

Subwords

Let $u = u(1)\dots u(\ell)$ and $w = w(1)\dots w(\ell')$. We say that

- u is a *subword* of w if there exist $p_1 < \dots < p_\ell$ such that $u(i) = w(p_i)$ for all $i \in [1,\ell]$.
- u is a *weak subword* of w if there exist $p_1 \leq \dots \leq p_\ell$ such that $u(i) = w(p_i)$ for all $i \in [1,\ell]$.

For instance, **abba** is a subword of b**ab**$aaaa$**ba**, and **bba** is a weak subword of a**ba**a (using the b twice). We say that a word is non-repeating if every two consecutive letters are different. If u is non-repeating, then u is a weak subword of w if and only if it is a subword of w.

As a warm-up, let us construct simple **LTL** formulas related to subwords.

- We consider the word $u = u(1)\dots u(\ell)$ and $p_1 < p_2 < \dots < p_\ell$ positive indices. They induce the following $\mathbf{LTL}(\mathbf{X}, \wedge)$ formula, called a *pattern*:

  $$\mathcal{P} = \mathbf{X}^{p_1-1}(u(1) \wedge \mathbf{X}^{p_2-p_1}(\dots \wedge \mathbf{X}^{p_\ell-p_{\ell-1}}u(\ell))\dots).$$

  It is equivalent to the (larger in size) formula $\bigwedge_{i\in[1,\ell]} \mathbf{X}^{p_i-1}u(i)$, which states that for each $i \in [1,\ell]$, the letter in position p_i is $u(i)$. We call a pattern *grounded* if $p_1 = 1$.

- We consider the word $u = u(1)\dots u(\ell)$. It induces the following $\mathbf{LTL}(\mathbf{F}, \wedge)$ formula, called a *fattern* (pattern with an **F**):

  $$\mathcal{F} = \mathbf{F}(u(1) \wedge \mathbf{F}(\dots \wedge \mathbf{F}u(\ell))\dots).$$

  A word w satisfies $\mathcal{F}$ if and only if u is a weak subword of w. A *grounded fattern* is a formula of the form $a \wedge \mathcal{F}$ with $\mathcal{F}$ a fattern.

3. NP-hardness when the alphabet is part of the input

This section proves two NP-hardness results that hold when the alphabet is part of the input. Both reductions use a large alphabet crucially; the remaining lower bounds of the paper work with a fixed alphabet.

NP-hardness of LTL($\mathbf{F}, \vee$) when the alphabet is part of the input

Theorem 2. *For all $\{\boldsymbol{F}, \vee\} \subseteq Op$, the $\boldsymbol{LTL}(Op)$ learning problem is* NP*-hard when the alphabet is part of the input.*

Recall that the *hitting set problem* takes as input $T_1, \dots, T_n$ subsets of $[1, m]$ and $k \in \mathbb{N}$, and asks whether there exists $H \subseteq [1, m]$ of size at most k such that for every $j \in [1, n]$ we have $H \cap T_j \neq \emptyset$. It is known to be NP-complete.

Proof. We construct a reduction from the hitting set problem. Let $T_1, \dots, T_n \subseteq [1, m]$ and $k \in \mathbb{N}$ an instance of the hitting set problem. Let us define the alphabet $\Sigma = \{a_j, b_j : j \in [1, n]\}$, it has size $2n$. For $j \in [1, n]$, we define u_j of length m by

$$u_j(i) = \begin{cases} a_j \text{ if } i \in T_j, \\ b_j \text{ otherwise.} \end{cases}$$

Let $v = b_1 \cdots b_n$. We claim that there exists a hitting set H of size at most k if and only if there exists a separating formula of **LTL** of size at most $2k$.

Given a hitting set H of size k, we construct the formula $\mathbf{F}(\bigvee_{j \in H} a_j)$, it is separating by definition of H being a hitting set, and it has size $2k$. Conversely, let us consider a separating formula φ of size $2k$. Since all operators have arity one or two, the syntactic tree of φ contains at most k leaves, so the set $H = \{j \in [1, m] : a_j \text{ appears in } \varphi\}$ has size at most k. Suppose H is not a hitting set of $T_1, \dots, T_n$, there exists $j \in [1, n]$ such that $H \cap T_j = \emptyset$. We prove that for all subformulas ψ of φ and for all $i \in [1, m]$, if $u_j[i \dots] \vDash \psi$ then $v[i \dots] \vDash \psi$. We proceed by induction.

- If ψ is a letter. If $u_j[i \dots] \vDash \psi$, this implies that $u_j(i) = b_j$ because a_j cannot appear in both ψ and u_j since $H \cap T_j = \emptyset$. Remark that $v(i) = b_j$, so $v[i \dots] \vDash \psi$.
- The other cases are easily proved by applying directly the induction hypothesis.

In particular, since u_j satisfies φ, then so does v, a contradiction with φ being separating. In conclusion, H is a hitting set of size k. □

NP-hardness of LTL($\mathbf{F}, \wedge$) when the alphabet is part of the input

Theorem 3. *The $\boldsymbol{LTL}(\boldsymbol{F}, \wedge)$ learning problem is* NP*-hard when the alphabet is part of the input, and there are no $(1 - o(1)) \cdot \log(n)$ polynomial time approximation algorithms unless* P = NP*, even with a single positive word.*

The proof uses a normalisation lemma for the special case of a single positive word. The idea is that any separating formula induces a separating (possibly grounded) fattern

of lesser or equal size. For instance, if $\mathbf{F}a \wedge \mathbf{F}b$ separates u from $v_1, \ldots, v_n$ then u must satisfy either $\mathbf{F}(a \wedge \mathbf{F}b)$ or $\mathbf{F}(b \wedge \mathbf{F}a)$, while the v_i satisfy neither, making one of the two a separating fattern.

Lemma 1. *Let $u, v_1, \ldots, v_n$. If there exists $\varphi \in \boldsymbol{LTL}(\boldsymbol{F}, \wedge)$ separating u from $v_1, \ldots, v_n$, then there exists a (possibly grounded) fattern of size at most $|\varphi|$ separating u from $v_1, \ldots, v_n$.*

Proof. Recall that the fattern induced by a word x has size $3|x| - 1$, and the grounded fattern induced by x has size $3|x| - 2$; a word z satisfies the (ungrounded) fattern induced by x if and only if x is a subword of z, and satisfies the grounded fattern induced by x if and only if x is a subword of z *with* $x(1)$ *mapped to the first position of* z.

We prove the following stronger statement by induction on $\varphi \in \mathbf{LTL}(\mathbf{F}, \wedge)$: for every word u with $u \models \varphi$ there exists a fattern $\mathcal{F}$, induced by a word x which is a subword of u, such that

(a) $u \models \mathcal{F}$;

(b) for every word z, $z \models \mathcal{F}$ implies $z \models \varphi$;

(c) $3|x| \leq |\varphi| + 2$ if $\mathcal{F}$ is grounded, and $3|x| \leq |\varphi| + 1$ if $\mathcal{F}$ is not grounded.

In both cases (c) gives $|\mathcal{F}| \leq |\varphi|$. The lemma then follows by applying the statement to φ and u: the resulting $\mathcal{F}$ separates u from $v_1, \ldots, v_n$, since $u \models \mathcal{F}$ by (a), while each $v_j \not\models \varphi$ together with (b) forces $v_j \not\models \mathcal{F}$.

Atom $\varphi = a$. Then $u(1) = a$. Take $\mathcal{F}$ to be the grounded fattern induced by the one-letter word $x = a$, of size 1. Then $u \models \mathcal{F}$, and $z \models \mathcal{F}$ implies $z(1) = a$, hence $z \models \varphi$; finally $3|x| = 3 = |\varphi| + 2$.

Eventually $\varphi = \mathbf{F}\psi$. Since $u \models \mathbf{F}\psi$, some suffix $u[j \ldots]$ satisfies ψ. By induction we obtain a fattern $\mathcal{F}'$ induced by a word x' (a subword of $u[j \ldots]$) with $u[j \ldots] \models \mathcal{F}'$, $\mathcal{F}' \Rightarrow \psi$, and $3|x'| \leq |\psi| + 2$. Let $\mathcal{F}$ be the *ungrounded* fattern induced by x', of size $3|x'| - 1 \leq |\psi| + 1 = |\varphi|$. As x' is a subword of u, we have $u \models \mathcal{F}$. If $z \models \mathcal{F}$ then x' is a subword of z, so some suffix $z[i \ldots]$ starts with $x'(1)$ and contains $x'(2) \cdots x'(|x'|)$ afterwards; whether $\mathcal{F}'$ is grounded or not, this means $z[i \ldots] \models \mathcal{F}'$, hence $z[i \ldots] \models \psi$ and $z \models \mathbf{F}\psi$. Finally $3|x'| \leq |\psi| + 2 = |\varphi| + 1$, so (c) holds for the ungrounded $\mathcal{F}$.

Conjunction $\varphi = \varphi_1 \wedge \varphi_2$. Then $u \models \varphi_1$ and $u \models \varphi_2$, and by induction we obtain fatterns $\mathcal{F}_1, \mathcal{F}_2$ induced by words x_1, x_2, each a subword of u, with $u \models \mathcal{F}_i$ and $\mathcal{F}_i \Rightarrow \varphi_i$. Fix embeddings of x_1 and x_2 as subwords of u, where a grounded $\mathcal{F}_i$ embeds x_i starting at the first position of u, and let w be the subword of u read off the union of the two sets of positions. Then x_1 and x_2 are both subwords of w (with the first position preserved whenever the corresponding $\mathcal{F}_i$ is grounded), and w is a subword of u. Let $\mathcal{F}$ be induced by w, grounded if at least one of $\mathcal{F}_1, \mathcal{F}_2$ is grounded and ungrounded otherwise. Then $u \models \mathcal{F}$, and if $z \models \mathcal{F}$ then w, hence each x_i, is a subword of z (with the first position preserved when $\mathcal{F}$, and thus $\mathcal{F}_i$, is grounded), so $z \models \mathcal{F}_i$ and therefore $z \models \varphi_i$ for $i = 1, 2$, i.e. $z \models \varphi$.

For the size, set $g_i = 2$ if $\mathcal{F}_i$ is grounded and $g_i = 1$ otherwise, so $3|x_i| \leq |\varphi_i| + g_i$, and recall $|\varphi| = |\varphi_1| + |\varphi_2| + 1$.

- If $\mathcal{F}_1$ and $\mathcal{F}_2$ are both grounded, their embeddings share the first position, so $|w| \leq |x_1| + |x_2| - 1$, and $\mathcal{F}$ is grounded: $3|w| \leq 3|x_1| + 3|x_2| - 3 \leq (|\varphi_1| + 2) + (|\varphi_2| + 2) - 3 \leq |\varphi| + 2$.

- Otherwise $|w| \leq |x_1| + |x_2|$; if exactly one $\mathcal{F}_i$ is grounded then $\mathcal{F}$ is grounded and $3|w| \leq (|\varphi_1| + 2) + (|\varphi_2| + 1) = |\varphi| + 2$, and if neither is then $\mathcal{F}$ is ungrounded and $3|w| \leq (|\varphi_1| + 1) + (|\varphi_2| + 1) = |\varphi| + 1$.

In every case (c) holds, which completes the induction. □

Proof of Theorem 3. We construct a reduction from the hitting set problem. Let $C_1, \dots, C_n$ subsets of $[1, m]$ and $k \in \mathbb{N}$. Let us consider the alphabet $[0, m]$, we define the word $u = 012 \dots m$. For each $j \in [1, n]$ we let $[1, m] \setminus C_j = \{a_{j,1} < \dots < a_{j,m_j}\}$, and define $v_j = 0 a_{j,1} \dots a_{j,m_j}$.

We claim that there exists a hitting set of size at most k if and only if there exists a formula in $\mathbf{LTL}(\mathbf{F}, \wedge)$ of size at most $3k - 1$ separating u from $v_1, \dots, v_n$.

Let $H = \{c_1, \dots, c_k\}$ be a hitting set of size k with $c_1 < c_2 < \dots < c_k$, we construct the (non-grounded) fattern induced by $w = c_1 \dots c_k$, it separates u from $v_1, \dots, v_n$ and has size $3k - 1$.

Conversely, let φ be a formula in $\mathbf{LTL}(\mathbf{F}, \wedge)$ of size at most $3k - 1$ separating u from $v_1, \dots, v_n$. Thanks to Lemma 1 we can assume that φ is a fattern of size at most $3k - 1$, and we let $w = c_1 \dots c_\ell$ be the non-repeating word it induces; both for a grounded and for a non-grounded fattern its size bound forces $\ell \leq k$. Necessarily $c_1 < c_2 < \dots < c_\ell$. If φ is grounded then $c_1 = 0$ (as u starts with 0), but then the non-grounded fattern induced by $c_2 \dots c_\ell$ is also separating, so we may assume that φ is not grounded. We let $H = \{c_1, \dots, c_\ell\} \cap [1, m]$, of size at most $\ell \leq k$, and argue that H is a hitting set. Indeed, H is a hitting set if and only if for every $j \in [1, n]$ we have $H \cap C_j \neq \emptyset$, which is equivalent to: for every $j \in [1, n]$, $v_j \not\models \varphi$. Recall that $v_j = 0\, a_{j,1} \dots a_{j,m_j}$ lists $\{0\} \cup ([1, m] \setminus C_j)$ in increasing order, so a letter $c \in [0, m]$ occurs in v_j if and only if $c = 0$ or $c \notin C_j$. Since w and v_j are both strictly increasing, w is a subword of v_j if and only if every c_i occurs in v_j, that is, if and only if $H \cap C_j = \emptyset$. Hence $v_j \not\models \varphi$ (i.e. w is not a subword of v_j) if and only if $H \cap C_j \neq \emptyset$, as claimed. □

4. Membership in NP: the short formula property

The goal of this section is the upper bound: the $\mathbf{LTL}(Op)$ learning problem is in NP for every $Op \subseteq \{\mathbf{F}, \mathbf{G}, \mathbf{X}, \wedge, \vee\}$. This is not obvious, since k is encoded in binary and a smallest separating formula could a priori be exponential in the size of the sample. We rule this out by establishing a *short formula property*: whenever a separating formula exists, there is one of size polynomial in the sample. The bulk of the section proves this property; because the available simplifications depend on which operators are present, we treat several groups of operator sets separately, and together they cover all $Op \subseteq \{\mathbf{F}, \mathbf{G}, \mathbf{X}, \wedge, \vee\}$.

Definition 1. *Let Op be a set of operator, we say that Op has the* short formula property *if there exists a polynomial $\boldsymbol{P}$ such that for all samples (P, N), if there exists an $\boldsymbol{LTL}(Op)$ separating formula, then there exists one of size at most $\boldsymbol{P}(|P| + |N|)$.*

Note that if Op has the short formula property then the $\mathbf{LTL}(Op)$ learning problem is in NP.

Theorem 4. *All $Op \subseteq \{\boldsymbol{F}, \boldsymbol{G}, \boldsymbol{X}, \wedge, \vee\}$ have the short formula property, and therefore the $\boldsymbol{LTL}(Op)$ learning problem is in* NP.

This theorem is the consequence of the following propositions, which establish it for various subsets of operators. The case analysis is organised around the Boolean connectives present in *Op*. When at most one of $\wedge, \vee$ occurs, the temporal operators can be normalised aggressively and the bound follows from two dedicated normalisation lemmas (Propositions 1 and 2). When both $\wedge$ and $\vee$ are available, separating formulas can instead be built explicitly word by word, and we split according to which temporal operators are present (Propositions 3, 4, and the two final propositions of the section). We start with some technical lemmas. The following lemma is easily proved by induction, it shows that **LTL** cannot detect repetitions in the last letter of a word.

Lemma 2. *Let u whose last letter is a. For all formulas $\varphi \in \boldsymbol{LTL}$, for all $k \in \mathbb{N}$, if u satisfies φ then ua^k satisfies φ.*

Lemma 3. *If a formula of the form $\varphi = \boldsymbol{F}(\psi_1 \wedge \boldsymbol{F}(\psi_2 \wedge (\cdots \wedge \boldsymbol{F}\psi_r)\dots)$ is not satisfied by some word v then there exist $i_1 < \cdots < i_k$ with $k \leq |v| + 1$ such that v does not satisfy $\psi = \boldsymbol{F}(\psi_{i_1} \wedge \boldsymbol{F}(\psi_{i_2} \wedge (\cdots \wedge \boldsymbol{F}\psi_{i_k})\dots)$ but all words satisfying φ also satisfy ψ.*

Proof. We set $v_1 = v$. For all $i \in [2, r]$ we set v_i as the largest suffix of v_{i-1} satisfying ψ_i. If this suffix does not exist, we set $v_i = \varepsilon$. As v does not satisfy φ, we easily obtain that for all $i < r$ v_i does not satisfy $\mathbf{F}(\psi_i \wedge \mathbf{F}(\psi_{i+1} \wedge \cdots \wedge \mathbf{F}\psi_r)\dots)$. Hence v_{r-1} does not satisfy $\mathbf{F}\psi_r$ and thus $v_r = \varepsilon$. We can now extract a decreasing subsequence of suffixes $v_{i_1}, \dots, v_{i_k}$ with $i_1 = 1$ and for all $j > 1$, i_j is the smallest index larger than i_{j-1} such that $v_{i_j} \neq v_{i_{j-1}}$ if it exists. Note that as this sequence is decreasing k cannot be larger than $|v| + 1$. We set $\psi = \mathbf{F}(\psi_{i_1} \wedge \mathbf{F}(\psi_{i_2} \wedge \cdots \wedge \mathbf{F}\psi_{i_k})\dots)$. We have that for all $j > 1$, v_{i_j} is the largest suffix of $v_{i_{j-1}}$ satisfying ψ_{i_j}, and $v_{i_k} = \varepsilon$. It is then easy to infer by reverse induction on j that for all j, v_{i_j} does not satisfy $\mathbf{F}(\psi_{i_j} \wedge \mathbf{F}(\psi_{i_{j+1}} \wedge \cdots \wedge \mathbf{F}\psi_{i_k})\dots)$ and thus that in particular $v_{i_1} = v$ does not satisfy ψ.

Let u be a word satisfying φ, there is a non-increasing sequence of suffixes u_i of u such that $u_1 = u$ and each u_i satisfies $\mathbf{F}(\psi_i \wedge \mathbf{F}(\psi_{i+1} \wedge \cdots \wedge \mathbf{F}\psi_r)\dots)$. It is then easy to see, by reverse induction on j, that for each $j \in [1, k]$ u_{i_j} satisfies $\mathbf{F}(\psi_{i_j} \wedge \mathbf{F}(\psi_{i_{j+1}} \wedge \cdots \wedge \mathbf{F}\psi_{i_k})\dots)$. In particular $u_1 = u$ satisfies ψ. □

We state and prove a dual version.

Lemma 4. *If a formula of the form $\varphi = \boldsymbol{G}(\psi_1 \vee \boldsymbol{G}(\psi_2 \vee \cdots \vee \boldsymbol{G}\psi_r)\dots)$ is satisfied by some word u then there exist $i_1 < \cdots < i_k$ with $k \leq |u| + 1$ such that u satisfies $\psi = \boldsymbol{G}(\psi_{i_1} \vee \boldsymbol{G}(\psi_{i_2} \vee \cdots \vee \boldsymbol{G}\psi_{i_k})\dots)$ and all words satisfying ψ also satisfy φ.*

Proof. Let us momentarily use negation. Note that the proof of Lemma 3 still holds when we allow negations. The word u does not satisfy $\varphi' = \mathbf{F}(\neg\psi_1 \wedge \mathbf{F}(\neg\psi_2 \vee \cdots \wedge \mathbf{F}\neg\psi_r)\dots)$ as it is equivalent to the negation of φ, thus there exist $i_1 < \ldots < i_k$ with $k \leq |u| + 1$ such that u does not satisfy $\psi' = \mathbf{F}(\psi_{i_1} \wedge \mathbf{F}(\psi_{i_2} \wedge \cdots \wedge \mathbf{F}\psi_{i_k})\dots)$ but all words satisfying φ' satisfy ψ'. The formula $\psi = \mathbf{G}(\psi_{i_1} \vee \mathbf{G}(\psi_{i_2} \vee \cdots \vee \mathbf{G}\psi_{i_k})\dots)$ is equivalent to the negation of ψ', thus it is satisfied by u, and all words satisfying ψ satisfy φ. □

The following fact will be useful for obtaining weak normal forms.

Fact 1. *For all formulas $\varphi, \psi, \varphi_1, \dots, \varphi_n \in \boldsymbol{LTL}$, the following equivalences hold:*

1. $\boldsymbol{G}\boldsymbol{G}\varphi \equiv \boldsymbol{G}\varphi$ *and* $\boldsymbol{F}\boldsymbol{F}\varphi \equiv \boldsymbol{F}\varphi$.
2. $\boldsymbol{X}(\varphi \wedge \psi) \equiv \boldsymbol{X}\varphi \wedge \boldsymbol{X}\psi$ *and* $\boldsymbol{X}(\varphi \vee \psi) \equiv \boldsymbol{X}\varphi \vee \boldsymbol{X}\psi$.
3. $\boldsymbol{G}\boldsymbol{X}\varphi \equiv \bot$ *and* $\boldsymbol{F}\boldsymbol{X}\varphi \equiv \boldsymbol{X}\boldsymbol{F}\varphi$.
4. $\boldsymbol{G}\boldsymbol{F}\varphi \equiv \boldsymbol{F}\boldsymbol{G}\varphi$ *(both state that φ is satisfied on the last position)*
5. $\boldsymbol{G}(\varphi \wedge \psi) \equiv \boldsymbol{G}\varphi \wedge \boldsymbol{G}\psi$ *and* $\boldsymbol{F}(\varphi \vee \psi) \equiv \boldsymbol{F}\varphi \vee \boldsymbol{F}\psi$.
6. $\boldsymbol{F}(\psi \wedge \bigwedge_{i=1}^{n} \boldsymbol{F}\varphi_i) \equiv \bigwedge_{i=1}^{n} \boldsymbol{F}(\psi \wedge \boldsymbol{F}\varphi_i)$ *and* $\boldsymbol{G}(\psi \vee \bigvee_{i=1}^{n} \boldsymbol{G}\varphi_i) \equiv \bigvee_{i=1}^{n} \boldsymbol{G}(\psi \vee \boldsymbol{G}\varphi_i)$.

Proposition 1. *All $Op \subseteq \{\boldsymbol{F}, \boldsymbol{G}, \boldsymbol{X}, \wedge\}$ have the short formula property.*

Proof. Let us consider a formula φ separating P from N in $\mathbf{LTL}(Op)$, we apply a series of transformations to φ to construct another separating formula of polynomial size.

First of all we push the $\mathbf{G}$ in front of the letters, which is possible by using repeatedly the equivalences 1, 3, 4 and 5 of Fact 1. Then we push the $\mathbf{X}$ to the bottom of the formula as well, as they commute with $\mathbf{F}$ and $\wedge$. We obtain a $\mathbf{LTL}(\mathbf{F}, \wedge)$ formula with atoms of the form either $\mathbf{X}^k\mathbf{G}a$ or $\mathbf{X}^k a$. Once that is done, we make the formula follow a normal form by repeatedly using equivalence 6 of Fact 1. While φ has a subformula of the form $\mathbf{F}(\psi \wedge \bigwedge_{i=1}^{p} \mathbf{F}\varphi_i)$ with $p > 1$, we turn it into $\bigwedge_{i=1}^{p} \mathbf{F}(\psi \wedge \mathbf{F}\varphi_i)$. In the end φ is of the form $\psi \wedge \bigwedge_{i=1}^{p} \varphi_i$ with each φ_i of the form $\mathbf{F}(\psi_1 \wedge \mathbf{F}(\psi_2 \wedge \cdots \wedge \mathbf{F}\psi_r)\dots)$ where ψ and the ψ_i are conjunctions of formulas of the form $\mathbf{X}^k\mathbf{G}a$ or $\mathbf{X}^k a$. Note that p may be exponential in $|\varphi|$.

Observe that given two formulas α and β of the form either a or $\mathbf{G}a$ for some $a \in \Sigma$, $\mathbf{X}^k\alpha \wedge \mathbf{X}^k\beta$ is equivalent to either $\mathbf{X}^k\alpha$, $\mathbf{X}^k\beta$, or $\bot$. It is then easy to see that every conjunction of formulas of the form $\mathbf{X}^k\mathbf{G}a$ or $\mathbf{X}^k a$ is equivalent over non-empty words of length at most ℓ to either $\bot$ or a conjunction of at most ℓ formulas of the form $\mathbf{X}^k a$ or $\mathbf{X}^k\mathbf{G}a$ with $k < \ell$. We can thus assume ψ and all ψ_i to be of that form, and thus to be of size polynomial in ℓ.

This formula being equivalent to φ, all u_i satisfy all φ_r and ψ, and for all v_j either v_j does not satisfy ψ or it does not satisfy some φ_r. We focus on the second case. Say v_j does not satisfy $\varphi_r = \mathbf{F}(\psi_1 \wedge \mathbf{F}(\psi_2 \wedge \cdots \wedge \mathbf{F}\psi_s)\dots)$, by Lemma 3 we can turn φ_ℓ into a short formula φ'_j that is still satisfied by all u_i but not by v_j. We set φ'_j to be as described above for all v_j that satisfy ψ and to be ψ for the others. As a result, the formula $\bigwedge_{i=1}^{m} \varphi'_j$ is a formula of polynomial size in m and ℓ separating the u_i and v_j.

Furthermore, we did not add any operators to the formula, hence the final formula uses the same set of operators as φ. □

Proposition 2. *All $Op \subseteq \{\boldsymbol{F}, \boldsymbol{G}, \boldsymbol{X}, \vee\}$ have the short formula property.*

Proof. The proof is nearly identical to the one of Proposition 1. We start by pushing the $\mathbf{F}$ and $\mathbf{X}$ to the bottom of the formula to obtain a formula of $\mathbf{LTL}(\mathbf{G}, \vee)$ with atoms of the form $\mathbf{X}^k a$ or $\mathbf{X}^k\mathbf{F}a$. We also use repeatedly equivalence 6 of Fact 1 to obtain a formula of the form $\psi \vee \bigvee_{i=1}^{p} \varphi_i$ with each φ_i of the form $\mathbf{G}(\psi_1 \vee \mathbf{G}(\psi_2 \vee \cdots \vee \mathbf{G}\psi_r)\dots)$ where ψ and the ψ_i

are disjunctions of formulas of the form $\mathbf{X}^k\mathbf{F}a$ or $\mathbf{X}^k a$. Then we note that every disjunction of formulas of the form $\mathbf{X}^k a$ and $\mathbf{X}^k\mathbf{F}a$ can be replaced by a disjunction of at most $\ell|\Sigma|$ such formulas, equivalent on non-empty words of length at most ℓ (as $\mathbf{X}^k\mathbf{F}a \vee \mathbf{X}^k a$ is equivalent to $\mathbf{X}^k\mathbf{F}a$). We can thus assume that all ψ_i and ψ mentioned above are of polynomial size in ℓ and $|\Sigma|$. No v_j satisfies either ψ or any ψ_r. For each u_i either there exists φ_r that is satisfied by u_i or u_i satisfies ψ. In the first case we use Lemma 4 to turn that φ_r into a formula φ'_i of polynomial size in ℓ and $|\Sigma|$ satisfied by u_i but not by any v_j. If u_i does not satisfy any φ_r we set $\varphi'_i = \psi$. In the end the formula $\bigvee_{i=1}^n \varphi'_i$ is satisfied by all u_i but no v_j, and is of polynomial size in ℓ, n and $|\Sigma|$. □

Proposition 3. *All $\{\boldsymbol{G}, \boldsymbol{X}, \wedge, \vee\} \subseteq Op$ have the short formula property.*

Proof. Let (P, N) a sample. Consider the formula $\varphi = \bigvee_{i=1}^n \varphi_{u_i}$ where $\varphi_{u_i} = \bigwedge_{j=1}^{m-1} \mathbf{X}^{j-1}a_j \wedge \mathbf{X}^{m-1}\mathbf{G}a_m$ with $u_i = a_1 \cdots a_m$. If there exist u_i, v_j such that $v_j \in u_i a^*$ with a the last letter of u_i, then there is no separating formula: By Lemma 2 every formula satisfied by u_i is also satisfied by v_j. Otherwise, every u_i satisfies the associated formula φ_{u_i} while no v_j satisfies any of them. Hence φ is a separating formula of polynomial size. □

Proposition 4. *All Op such that $\{\boldsymbol{X}, \wedge, \vee\} \subseteq Op \subseteq \{\boldsymbol{F}, \boldsymbol{X}, \wedge, \vee\}$ have the short formula property.*

Proof. Let (P, N) a sample. Consider the formula $\varphi = \bigvee_{i=1}^n \varphi_{u_i}$ where $\varphi_{u_i} = \bigwedge_{j=1}^{m} \mathbf{X}^{j-1}a_j$ with $u_i = a_1 \cdots a_m$. We distinguish two cases.

- If there exist u_i, v_j such that u_i is a prefix of v_j, then there is no separating formula: An easy induction shows that every formula satisfied by u_i is also satisfied by v_j.
- Otherwise, every u_i satisfies the associated formula φ_{u_i} while no v_j satisfies any of them. Hence φ is a separating formula of polynomial size.

□

Fact 2. *For all $\varphi, \varphi_1, \varphi_2 \in \boldsymbol{LTL}(\boldsymbol{F}, \boldsymbol{G}, \wedge, \vee, \neg)$, the following equivalences hold:*

- $\neg(\varphi_1 \wedge \varphi_2) \equiv \neg\varphi_1 \vee \neg\varphi_2$
- $\neg(\varphi_1 \vee \varphi_2) \equiv \neg\varphi_1 \wedge \neg\varphi_2$
- $\neg\boldsymbol{F}\varphi \equiv \boldsymbol{G}\neg\varphi$
- $\neg\boldsymbol{G}\varphi \equiv \boldsymbol{F}\neg\varphi$

Proposition 5. *$\{\boldsymbol{F}, \wedge, \vee\}$ and $\{\boldsymbol{G}, \wedge, \vee\}$ have the short formula property.*

Proof. Let (P, N) a sample. Let $j \in [1, n]$, if there exists w_j a weak subword of all u_i that is not a weak subword of v_j then we set $\psi(w_j) = \mathbf{F}(a_1 \wedge \mathbf{F}(a_2 \wedge \cdots \mathbf{F}a_k))$ with $w_j = a_1 \cdots a_k$. This formula is satisfied by exactly the words which have w_j as a weak subword, hence by all u_i but not v_j. We set φ_j as $\psi(w_j)$ if it exists and a if all u_i start with an a and v_j does not for some $a \in \Sigma$.

If φ_j is defined for all j then the formula $\bigwedge_j \varphi_j$ is a separating formula. Otherwise there exists a v_j such that there are no weak subwords of all the u_i that are not weak subwords of v_j and either the u_i do not start with the same letter or v_j also starts with that letter. In that case we can easily prove by induction that all $\mathbf{LTL}(\mathbf{F}, \wedge, \vee)$ formulas of the form $\mathbf{F}\psi$ that are satisfied by all u_i are also satisfied by v_j, and as all formulas of the form a with $a \in \Sigma$ satisfied by all u_i are also satisfied by v_j, the same can be said about boolean combinations of those two types of formulas, and thus there are no separating formulas in $\mathbf{LTL}(\mathbf{F}, \wedge, \vee)$.

Concerning $\{\mathbf{G}, \wedge, \vee\}$, one can easily infer from Fact 2 that we can turn any formula of $\mathbf{LTL}(\mathbf{F}, \wedge, \vee)$ into one of $\mathbf{LTL}(\mathbf{G}, \wedge, \vee)$ equivalent to its negation, and vice-versa. Hence the short formula property of $\{\mathbf{G}, \wedge, \vee\}$ is a consequence of the one of $\{\mathbf{F}, \wedge, \vee\}$. □

Proposition 6. *$\{\boldsymbol{F}, \boldsymbol{G}, \wedge, \vee\}$ has the short formula property.*

Proof. Let (P, N) a sample. For all u_i we set $\varphi(u_i) = \mathbf{F}(a_1 \wedge \mathbf{F}(a_2 \wedge \cdots \mathbf{F}a_k))$ with $u_i = a_1 \cdots a_k$. This formula is satisfied by exactly the words with u_i as a weak subword. For all v_j we set $\varphi(v_j) = \mathbf{G}(\bar{a_1} \vee \mathbf{G}(\bar{a_2} \vee \cdots \mathbf{G}\bar{a_k}))$ with $v_j = a_1 \cdots a_k$ and $\bar{a} = \bigvee_{b \in \Sigma \setminus \{a\}} b$ for all $a \in \Sigma$. This formula is satisfied by exactly the words which do not have v_j as a weak subword. For all pairs (u_i, v_j) we set $\psi_{i,j}$ as $\varphi(u_i) \wedge \varphi(v_j)$, and finally we define ψ as $\bigvee_i \bigwedge_j \psi_{i,j}$. If this formula does not separate the u_i and v_j then there must exist u_i and v_j which are weak subwords of each other. An easy induction on the formula then shows that u_i and v_j satisfy the same formulas in $\mathbf{LTL}(\mathbf{F}, \mathbf{G}, \wedge, \vee)$, and thus there does not exist any separating formula. □

5. Degenerate cases: the shortest formula property

The previous section places the learning problem in NP for every fragment. Before turning to hardness, we identify in this section the fragments that are, on the contrary, easy: those whose operators are too weak to encode a hard problem. For each such fragment we exhibit a polynomial-time algorithm that even returns a *minimal* separating formula, a property we call the *shortest formula property*. These are the "degenerate" fragments; together with the hardness results of the following sections they pin down exactly the boundary between P and NP-completeness (see Table **??**). The fragments below are precisely those avoiding all four hard pairs $\{\mathbf{X}, \wedge\}$, $\{\mathbf{X}, \vee\}$, $\{\mathbf{F}, \wedge\}$, $\{\mathbf{G}, \vee\}$ identified later.

Definition 2. *Let Op a set of operators, we say that Op has the* shortest formula property *if there is a polynomial time algorithm solving the $\boldsymbol{LTL}(Op)$ learning problem and outputting the minimal separating formula if it exists.*

Theorem 5. *Let Op such that either $Op \subseteq \{\boldsymbol{F}, \boldsymbol{G}, \boldsymbol{X}\}$, $Op \subseteq \{\vee, \wedge\}$ or $Op = \{\boldsymbol{G}, \wedge\}$. Then Op has the shortest formula property, and therefore the $\boldsymbol{LTL}(Op)$ learning problem is in P. The same holds for $Op = \{\boldsymbol{F}, \vee\}$ if the size of the alphabet is fixed.*

This theorem is the consequence of the following propositions, which establish it for various subsets Op of operators.

Proposition 7. *All $Op \subseteq \{\vee, \wedge\}$ have the shortest formula property.*

Proof. Let (P, N) a sample. Let $A \subseteq \Sigma$ be the set of first letters of the u_i. Consider the formula $\psi = \bigvee_{a \in A} a$. We distinguish two cases.

- Either ψ does not separate the u_i and v_i, meaning that some u_i and v_j share the same first letter. In that case u_i and v_j satisfy the same formulas of $\mathbf{LTL}(\vee, \wedge)$ and thus there is no separating formula.
- Or ψ does separate the u_i and v_j. In that case it is also of minimal size: say some element a of A does not appear in a formula $\varphi \in \mathbf{LTL}(\vee, \wedge)$, then as it is satisfied by some u_i starting with a, ψ is a tautology and is also satisfied by the v_j. Hence a separating formula has to contain all letters of A, and thus also at least $|A| - 1$ binary operators. As a result, ψ is of minimal size.

□

Proposition 8. *All $Op \subseteq \{\boldsymbol{F}, \boldsymbol{G}, \boldsymbol{X}\}$ have the shortest formula property.*

Proof. Let $\ell \in \mathbb{N}$, every formula $\varphi \in \mathbf{LTL}(\mathbf{F}, \mathbf{G}, \mathbf{X})$ is equivalent over words of length at most ℓ to a formula of smaller or equal size in

$$\left\{ \top, \bot, \mathbf{X}^k a, \mathbf{X}^k \mathbf{F} a, \mathbf{X}^k \mathbf{G} a, \mathbf{X}^k \mathbf{F} \mathbf{G} a \mid a \in \Sigma, 0 \leq k < \ell \right\}.$$

This can be shown by applying Fact 1 to push all $\mathbf{X}$ operators at the top of the formula. Then, we observe that every formula of the form $\mathbf{X}^k \psi$ with $k \geq N$ is equivalent to $\bot$.

As a result we can compute a minimal separating formula by enumerating formulas from this set (of polynomial size) and checking if there is one separating the positive and negative words. □

Proposition 9. *$\{\boldsymbol{G}, \wedge\}$ has the shortest formula property.*

Proof. An easy induction on ψ shows that all $\mathbf{LTL}(\mathbf{G}, \wedge)$ formulas are equivalent over finite words to a formula of the form $\top$, $\bot$, a or $\mathbf{G}a$ with $a \in \Sigma$, i.e., to a formula of size at most 2. As those can be enumerated in polynomial time, we can compute a separating formula of minimal size or conclude that it does not exist in polynomial time. □

A corollary of our results is that the classification between P and NP mostly does not depend upon whether we consider the alphabet as part of the input or not. The only affected subcase is $Op = \{\mathbf{F}, \vee\}$, which is NP-hard when the alphabet is not fixed (Theorem 2).

Proposition 10. *$\{\boldsymbol{F}, \vee\}$ has the shortest formula property when the alphabet is fixed.*

Proof. Since $\mathbf{F}$ commutes with $\vee$, every formula can be turned into an equivalent disjunction of formulas of the form a or $\mathbf{F}a$ with $a \in \Sigma$, of polynomial size. There are only $2^{2|\Sigma|}$ such formulas, hence when given a sample (P, N) one can simply select the formulas that are not satisfied by any v_j, take their disjunction, and check that all u_i satisfy the resulting formula. If they do not there cannot exist any separating formula. If they do, then we have a separating formula of size at most $3 \cdots 2^{2|\Sigma|}$, which is constant when Σ is fixed. We can then simply enumerate the (constant number of) smaller formulas and keep the smallest separating one. □

6. An approximation algorithm for LTL($\mathbf{X}, \wedge$)

For the fragments that are NP-hard one may still hope for approximation algorithms. This section gives such a positive result for $\mathbf{LTL}(\mathbf{X}, \wedge)$: a polynomial-time $\log(n)$-approximation algorithm. The next section shows that this ratio is optimal (under $\mathsf{P} \neq \mathsf{NP}$), so the two sections together pin down the approximability of $\mathbf{LTL}(\mathbf{X}, \wedge)$. Along the way we introduce a normal form for $\mathbf{LTL}(\mathbf{X}, \wedge)$ formulas and a size measure for them, both reused in the hardness proof of the next section.

An α-approximation algorithm for learning a fragment of **LTL** does the following: the algorithm either determines that there are no separating formulas, or constructs a separating formula φ which has size at most $\alpha \cdot m$ with m the size of a minimal separating formula.

Theorem 6. *There exists a $O(n \cdot \ell^2)$ time $\log(n)$-approximation algorithm for learning $\boldsymbol{LTL}(\boldsymbol{X}, \wedge)$.*

Recall the definition of patterns: a word $u = u(1) \dots u(\ell)$ and $p_1 < p_2 < \cdots < p_\ell$ induce the following $\mathbf{LTL}(\mathbf{X}, \wedge)$ formula, called a *pattern*:

$$\mathcal{P} = \mathbf{X}^{p_1 - 1}(u(1) \wedge \mathbf{X}^{p_2 - p_1}(\cdots \wedge \mathbf{X}^{p_\ell - p_{\ell-1}} u(\ell)) \cdots).$$

It is equivalent to the (larger in size) formula $\bigwedge_{i \in [1,\ell]} \mathbf{X}^{p_i - 1} u(i)$, which states that for each $i \in [1, \ell]$, the letter in position p_i is $u(i)$. To determine the size of a pattern $\mathcal{P}$ we look at two parameters: its last position $\mathbf{last}(\mathcal{P}) = i_p$ and its width $\mathbf{width}(\mathcal{P}) = p$. The size of $\mathcal{P}$ is $\mathbf{last}(P) + 2(\mathbf{width}(P) - 1)$. The two parameters of a pattern, last position and width, hint at the key trade-off we will have to face in learning $\mathbf{LTL}(\mathbf{X}, \wedge)$ formulas: do we increase the last position, to reach further letters in the words, or the width, to further restrict the set of satisfying words?

Lemma 5. *For every formula $\varphi \in \boldsymbol{LTL}(\boldsymbol{X}, \wedge)$ there exists an equivalent pattern of size smaller than or equal to φ.*

Proof. We proceed by induction on φ.

- Atomic formulas are already a special case of patterns.
- If $\varphi = \mathbf{X}\varphi'$, by induction hypothesis we get a pattern $\mathcal{P}$ equivalent to φ', then $\mathbf{X}\mathcal{P}$ is a pattern and equivalent to φ.
- If $\varphi = \varphi_1 \wedge \varphi_2$, by induction hypothesis we get two patterns $\mathcal{P}_1$ and $\mathcal{P}_2$ equivalent to φ_1 and φ_2. We use the inductive definition for patterns to show that $\mathcal{P}_1 \wedge \mathcal{P}_2$ is equivalent to another pattern. We focus on the case $\mathcal{P}_1 = \mathbf{X}^{i_1}(c_1 \wedge \mathcal{P}_1')$ and $\mathcal{P}_2 = \mathbf{X}^{i_2}(c_2 \wedge \mathcal{P}_2')$, the other cases are simpler instances of this one.

  There are two cases: $i_1 = i_2$ or $i_1 \neq i_2$.

  If $i_1 = i_2$, either $c_1 \neq c_2$ and then $\mathcal{P}_1 \wedge \mathcal{P}_2$ is equivalent to false, which is the pattern $c_1 \wedge c_2$, or $c_1 = c_2$, and then $\mathcal{P}_1 \wedge \mathcal{P}_2$ is equivalent to $\mathbf{X}^{i_1}(c_1 \wedge \mathcal{P}_1' \wedge \mathcal{P}_2')$. By induction hypothesis $\mathcal{P}_1' \wedge \mathcal{P}_2'$ is equivalent to a pattern $\mathcal{P}'$, so the pattern $\mathbf{X}^{i_1}(c_1 \wedge \mathcal{P}')$ is equivalent to $\mathcal{P}_1 \wedge \mathcal{P}_2$, hence to φ.

  If $i_1 \neq i_2$, without loss of generality $i_1 < i_2$, then $\mathcal{P}_1 \wedge \mathcal{P}_2$ is equivalent to $\mathbf{X}^{i_1}(c_1 \wedge \mathcal{P}_1' \wedge \mathbf{X}^{i_2 - i_1}(c_2 \wedge \mathcal{P}_2'))$. By induction hypothesis $\mathcal{P}_1' \wedge \mathbf{X}^{i_2 - i_1}(c_2 \wedge \mathcal{P}_2')$ is equivalent to a pattern $\mathcal{P}'$, so the pattern $\mathbf{X}^{i_1}(c_1 \wedge \mathcal{P}')$. is equivalent to $\mathcal{P}_1 \wedge \mathcal{P}_2$, hence to φ.

□

Proof. Let $u_1, \dots, u_n, v_1, \dots, v_n$ a set of $2n$ words of length at most ℓ. Thanks to Lemma 5 we are looking for a pattern. For a pattern $\mathcal{P} = \mathbf{X}^{i_1-1}(u(1) \wedge \mathbf{X}^{i_2-i_1}(\cdots \wedge \mathbf{X}^{i_m-i_{m-1}}u(m))\cdots)$ we define $I(\mathcal{P}) = \{i_q \in [1,\ell] : q \in [1,m]\}$. Note that $\mathbf{last}(\mathcal{P}) = \max I(\mathcal{P})$ and $\mathbf{width}(\mathcal{P}) = \mathrm{Card}(I(\mathcal{P}))$.

We define the set $X = \{i \in [1,\ell] : \exists c \in \Sigma, \forall j \in [1,n], u_j(i) = c\}$ of positions on which all u_i agree. Note that if $\mathcal{P}$ satisfies $u_1, \dots, u_n$ then $I(\mathcal{P}) \subseteq X$. Further, given $I = \{p_1 < \cdots < p_m\} \subseteq X$, we can construct a pattern $\mathcal{P}$ such that $I(\mathcal{P}) = I$ and $\mathcal{P}$ satisfies $u_1, \dots, u_n$: we simply choose the pattern corresponding to the word $u_i(p_1)\dots u_j(p_m)$ for some j (independent of j since $I \subseteq X$) and positions of I. We call that pattern $\mathcal{P}$.

Recall that the size of the pattern $\mathcal{P}$ is $\mathbf{last}(\mathcal{P}) + 2(\mathbf{width}(\mathcal{P}) - 1)$. This makes the task of minimising it difficult: there is a trade-off between minimising the last position $\mathbf{last}(\mathcal{P})$ and the width $\mathbf{width}(\mathcal{P})$.

Let us consider the following easier problem: construct a $\log(n)$-approximation of a minimal separating pattern with fixed last position. Assuming we have such an algorithm, we obtain a $\log(n)$-approximation of a minimal separating pattern by running the previous algorithm on prefixes of length ℓ' for each $\ell' \in [1,\ell]$.

We now focus on the question of constructing a $\log(n)$-approximation of a minimal separating pattern with fixed last position. For $i \in X$ we define $Y_i = \{j \in [1,n] \mid v_j(i) \neq u_1(i)\}$ the set of negative words which differ from the positive words at position i. For a set $I \subseteq X$, we write $C_I = \bigcup \{Y_i \mid i \in I\}$: The pattern $\mathcal{P}$ corresponding to I does not satisfy v_j if and only if $j \in C_I$. In particular, the pattern corresponding to I is separating if and only if $C_I = [1,n]$.

The algorithm constructs a set I incrementally through the sequence $(I_z)_{z \geq 0}$. For each z define $C_z = C_{I_z}$. The algorithm is greedy: I_z is augmented with $i \in X \setminus I_z$ maximising the number of words added to C_z by adding i, which is the cardinality of $Y_i \setminus C_z$.

We now prove that this yields a $\log(n)$-approximation algorithm. Let $\mathcal{P}_{\mathrm{opt}}$ a minimal separating pattern with last position ℓ, inducing $I_{\mathrm{opt}} = I(\mathcal{P}_{\mathrm{opt}}) \subseteq [1,\ell]$ of cardinal m. Note that $C_{I_{\mathrm{opt}}} = [1,n]$.

We let $n_z = n - |C_z|$ and show the following by induction on $z \geq 0$:

$$n_{z+1} \leq n_z \cdot \left(1 - \frac{1}{m}\right) = n_z \cdot \frac{m-1}{m}.$$

We claim that there exists $i \in X \setminus I_z$ such that $\mathrm{Card}(Y_i \setminus C_z) \geq \frac{n_z}{m}$. Indeed, assume towards contradiction that for all $i \in X \setminus I_z$ we have $\mathrm{Card}(Y_i \setminus C_z) < \frac{n_z}{m}$, then there are no sets I of cardinal m such that $C_I \supseteq [1,n] \setminus C_z$, contradicting the existence of I_{opt}. Thus there exists $i \in X \setminus I_z$ such that $\mathrm{Card}(Y_i \setminus C_z) \geq \frac{n_z}{m}$, implying that the algorithm chooses such an i and $n_{z+1} \leq n_z - \frac{n_z}{m} = n_z \cdot \left(1 - \frac{1}{m}\right)$.

The proved inequality implies $n_z \leq n \cdot \left(1 - \frac{1}{m}\right)^z$. This quantity is less than 1 for $z \geq \log(n) \cdot m$, implying that the algorithm stops after at most $\log(n) \cdot m$ steps. Consequently, the pattern corresponding to I has size at most $\log(n) \cdot |\mathcal{P}_{\mathrm{opt}}|$, completing the claim on approximation.

A naive complexity analysis yields an implementation of the greedy algorithm running in time $O(n \cdot \ell)$, leading to an overall complexity of $O(n \cdot \ell^2)$ by running the greedy algorithm on the prefixes of length ℓ' of $u_1, \dots, u_n, v_1, \dots, v_n$ for each $\ell' \in [1, \ell]$. □

7. Almost all fragments with the next operator are hard to approximate

We fix the alphabet for the remainder of the paper and ask where the learning problem sits in the complexity hierarchy. This section establishes hardness for all fragments containing the next operator together with a Boolean connective. We proceed in two steps. First we treat the base fragment $\mathbf{LTL}(\mathbf{X}, \wedge)$, via a reduction from the set cover problem that also yields the optimal inapproximability matching the algorithm of the previous section (Theorem 7). Then we lift this hardness to every fragment Op with $\{\mathbf{X}, \wedge\} \subseteq Op \subseteq \{\mathbf{F}, \mathbf{G}, \mathbf{X}, \wedge, \vee\}$, by transforming the words so that the $\mathbf{F}$, $\mathbf{G}$, and $\vee$ operators become useless and a minimal $\mathbf{LTL}(Op)$ separator collapses to an $\mathbf{LTL}(\mathbf{X}, \wedge)$ one (Proposition 11); a symmetric argument handles the fragments with $\{\mathbf{X}, \vee\}$ (Proposition 12).

Theorem 7. *The $\boldsymbol{LTL}(\boldsymbol{X}, \wedge)$ learning problem is* NP*-hard, and there are no $(1-o(1))\cdot\log(n)$ polynomial time approximation algorithms unless* P = NP*, even for a single positive word.*

Note that Theorem 6 and Theorem 7 yield matching upper and lower bounds on approximation algorithms for learning $\mathbf{LTL}(\mathbf{X}, \wedge)$.

The hardness result stated in Theorem 7 follows from a reduction to the set cover problem, that we define now. The set cover decision problem is: given $S_1, \dots, S_\ell$ subsets of $[1, n]$ and $k \in \mathbb{N}$, does there exists $I \subseteq [1, \ell]$ of size at most k such that $\bigcup_{i \in I} S_i = [1, n]$? In that case we say that I is a cover. This problem is in fact essentially the same as hitting set, but presented differently, which makes the following proofs easier. An α-approximation algorithm returns a cover of size at most $\alpha \cdot k$ where k is the size of a minimal cover. The following results form the state of the art for solving exact and approximate variants of the set cover problem.

Theorem 8 ((Dinur & Steurer, 2014))**.** *The set cover problem is* NP*-complete, and there are no $(1 - o(1)) \cdot \log(n)$ polynomial time approximation algorithms unless* P = NP*.*

Proof of Theorem 7. We construct a reduction from set cover. Let $S_1, \dots, S_\ell$ subsets of $[1, n]$ and $k \in \mathbb{N}$.

Let us consider the word $u = a^{\ell+1}$, and for each $j \in [1, n]$ and $i \in [1, \ell]$, writing $v_j(i)$ for the i^{th} letter of v_j:

$$v_j(i) = \begin{cases} b \text{ if } j \in S_i, \\ a \text{ if } j \notin S_i, \end{cases}$$

and we set $v_j(\ell + 1) = a$ for any $j \in [1, n]$. We also add $v_{n+1} = a^\ell b$.

We claim that there is a cover of size k if and only if there is a formula of size $\ell + 2k + 1$ separating u from $v_1, \dots, v_{n+1}$.

Thanks to Lemma 5 we can restrict our attention to patterns, *i.e* formulas of the form (we adjust the indexing for technical convenience)

$$\varphi = \mathbf{X}^{i_1 - 1}(c_1 \wedge \mathbf{X}^{i_2 - i_1}(\cdots \wedge \mathbf{X}^{i_{p+1} - i_p} c_{p+1}) \cdots),$$

for some positions $i_1 \leq \cdots \leq i_{p+1}$ and letters $c_1, \ldots, c_{p+1} \in \Sigma$. If u satisfies φ, then necessarily $c_1 = \cdots = c_{p+1} = a$. This implies that if v_{n+1} does not satisfy φ, then necessarily $i_{p+1} = \ell + 1$.

We associate to φ the set $I = \{i_1, \ldots, i_p\}$. It is easy to see that φ is equivalent to $\bigwedge_{q \in [1,p]} \mathbf{X}^{i_q - 1} a \wedge \mathbf{X}^{\ell} a$, which is the pattern on the positions $I \cup \{\ell + 1\}$, all checking the letter a. Its last position is $\ell + 1$ and its width is $|I| + 1$, so following the size convention $\mathbf{last} + 2(\mathbf{width} - 1)$ its size is $(\ell + 1) + 2|I|$.

By construction, φ separates u from $v_1, \ldots, v_{n+1}$ if and only if I is a cover. Indeed, I is a cover if and only if for every $j \in [1, n]$ there exists $i \in I$ such that $j \in S_i$, which is equivalent to for every $j \in [1, n]$ we have $v_j \not\models \varphi$. □

We can extend the previous hardness result to all sets of operators Op such that $\{\mathbf{X}, \wedge\} \subseteq Op \subseteq \{\mathbf{F}, \mathbf{G}, \mathbf{X}, \wedge, \vee\}$, by reducing their learning problems to the previous one. The reduction consists in transforming the input words so that the $\mathbf{F}$ and $\mathbf{G}$ operators are essentially useless. The $\vee$ operator is also useless since we only have one positive word, implying that the minimal separating formulas in $\mathbf{LTL}(Op)$ and $\mathbf{LTL}(\mathbf{X}, \wedge)$ are in fact the same.

The first step is a reduction lemma for disjunctions.

Lemma 6. *For all $\varphi \in \boldsymbol{LTL}(\boldsymbol{X}, \wedge, \vee)$, for all $u, v_1, \ldots, v_n$, if φ separates u from $v_1, \ldots, v_n$, then there exists $\psi \in \boldsymbol{LTL}(\boldsymbol{X}, \wedge)$ such that $|\psi| \leq |\varphi|$ which separates u from $v_1, \ldots, v_n$.*

Proof. We define $D(\varphi) \subseteq \mathbf{LTL}(\mathbf{X}, \wedge)$ by induction:

- If $\varphi = c$ then $D(\varphi) = \{c\}$.
- If $\varphi = \varphi_1 \wedge \varphi_2$ then $D(\varphi) = \{\psi_1 \wedge \psi_2 : \psi_1 \in D(\varphi_1), \psi_2 \in D(\varphi_2)\}$.
- If $\varphi = \varphi_1 \vee \varphi_2$ then $D(\varphi) = D(\varphi_1) \cup D(\varphi_2)$.
- If $\varphi = \mathbf{X}\varphi'$ then $D(\varphi) = \{\mathbf{X}\psi : \psi \in D(\varphi')\}$.

Observe that all formulas of $D(\varphi)$ are of size at most $|\varphi|$, and are in $\mathbf{LTL}(\mathbf{X}, \wedge)$. We now show that for all φ, for all $u, v_1, \ldots, v_n$, if φ separates u from $v_1, \ldots, v_n$ then there exists $\psi \in D(\varphi)$ separating them, which proves the lemma.

We proceed by induction on φ.

- If $\varphi = c$ this is clear.
- If $\varphi = \varphi_1 \wedge \varphi_2$ then $D(\varphi) = \{\psi_1 \wedge \psi_2 : \psi_1 \in D(\varphi_1), \psi_2 \in D(\varphi_2)\}$. Since φ separates u from $v_1, \ldots, v_n$, there exists $I_1, I_2 \subseteq [1, n]$ such that $I_1 \cup I_2 = [1, n]$, φ_1 separates u from $\{v_i : i \in I_1\}$, and φ_2 separates u from $\{v_i : i \in I_2\}$. By induction hypothesis applied to both φ_1 and φ_2 there exists $\psi_1 \in D(\varphi_1)$ separating u from $\{v_i : i \in I_1\}$ and $\psi_2 \in D(\varphi_2)$ separating u from $\{v_i : i \in I_2\}$. It follows that $\psi_1 \wedge \psi_2$ separates u from $v_1, \ldots, v_n$, and $\psi_1 \wedge \psi_2 \in D(\varphi)$.
- If $\varphi = \varphi_1 \vee \varphi_2$ then $D(\varphi) = D(\varphi_1) \cup D(\varphi_2)$. Since φ separates u from $v_1, \ldots, v_n$, either φ_1 or φ_2 does as well; without loss of generality let us say that φ_1 separates u from $v_1, \ldots, v_n$. The induction hypothesis implies that $\psi_1 \in D(\varphi_1)$ separates u from $v_1, \ldots, v_n$, and $\psi_1 \in D(\varphi)$.

- The case $\varphi = \mathbf{X}\varphi'$ follows directly by induction hypothesis.

□

Proposition 11. *For all $\{\boldsymbol{X}, \wedge\} \subseteq Op \subseteq \{\boldsymbol{F}, \boldsymbol{G}, \boldsymbol{X}, \wedge, \vee\}$, there is a polynomial-time reduction from the $\boldsymbol{LTL}(X, \wedge)$ learning problem for a single positive word to the $\boldsymbol{LTL}(Op)$ one over the same alphabet.*

We recall two (folklore) facts about **LTL**. They can easily be proven by induction on the formula.

Fact 3. *For all $w \in \Sigma^{\geq m}, m \in \mathbb{N}$, and $\varphi \in \boldsymbol{LTL}(\boldsymbol{X}, \wedge)$ with $|\varphi| \leq m$, we have $w \vDash \varphi$ if and only if $w[1, m] \vDash \varphi$.*

Fact 4. *For all $w_1 \in \Sigma^*, w_2 \in \Sigma^+, m \in \mathbb{N}$, and $\varphi \in \boldsymbol{LTL}$ with $|\varphi| \leq m$, then $w_1 w_2^m \vDash \varphi$ if and only if $w_1 w_2^{m+1} \vDash \varphi$.*

We now proceed to the proof of Proposition 11.

Proof. We construct a polynomial-time reduction from the $\mathbf{LTL}(\mathbf{X}, \wedge)$ learning problem for a single positive word to the same problem for $\mathbf{LTL}(Op)$. Let $u, v_1, \ldots, v_n \in \Sigma^*$ be words, all of length ℓ. Without loss of generality we can assume that u is different from all v_i's, since we can detect this case in polynomial time and outputs that there are no separating formula then.

Let us consider the formula $\psi_u = \bigwedge_{i=1}^{\ell} \mathbf{X}^{i-1} u(i)$, it separates u and the v_i's (as each $v_i \neq u$ differs from u at some position). Its size is $M = O(\ell^2)$, which is therefore an upper bound on the size of a separating formula. Let $a \in \Sigma$, we define the following:

- $u' = ua^M$ and for all $i \in [1, n]$, $v_i' = v_i a^M$.
- $\mathbf{u} = (u' v_1' \cdots v_n')^{M+1}$ and for all $i \in [1, n]$, $\mathbf{v}_i = v_i' \cdots v_n' (u' v_1' \cdots v_n')^M$.

We write $z = u' v_1' \cdots v_n'$ for one block, so that $\mathbf{u} = z^{M+1}$ and $\mathbf{v}_i = v_i' \cdots v_n' \, z^M$. Note that $\mathbf{v}_i$ is a suffix of $\mathbf{u}$, since $\mathbf{u} = u' v_1' \cdots v_{i-1}' \, \mathbf{v}_i$. We argue that the following two properties hold:

1. The minimal size of a formula of $\mathbf{LTL}(\mathbf{X}, \wedge)$ separating $\mathbf{u}$ and the $\mathbf{v}_i$'s is the same as the minimal size of a formula of $\mathbf{LTL}(\mathbf{X}, \wedge)$ separating u and the v_i's.
2. The minimal size of a formula of $\mathbf{LTL}(Op)$ separating $\mathbf{u}$ and the $\mathbf{v}_i$ is the same as the minimal size of a formula of $\mathbf{LTL}(\mathbf{X}, \wedge)$ separating $\mathbf{u}$ and the $\mathbf{v}_i$'s.

Let us prove the first property. The formula ψ_u separates both instances (its positions all lie in $[1, \ell]$, where u differs from each v_i, and where likewise $\mathbf{u}$ differs from each $\mathbf{v}_i$ on their first ℓ letters), so both minimal sizes are well defined and at most M.

By Lemma 5 a minimal separator can be taken to be a pattern, and since its size is at most M, all its positions lie in $[1, M]$. Recall that $u' = ua^M$ and $v_i' = v_i a^M$; by construction $\mathbf{u}$ and u' share their first $\ell + M$ letters, as do $\mathbf{v}_i$ and v_i'. In particular, for $\ell < q \leq M$ we have $\mathbf{u}(q) = a = \mathbf{v}_i(q)$ for all $i \in [1, n]$. Hence a position $q \in (\ell, M]$ of a separating pattern is satisfied by $\mathbf{u}$ and by every $\mathbf{v}_i$ alike (it checks the letter a, since $\mathbf{u} \vDash \varphi$),

so removing it preserves separation; by minimality no such position occurs. Therefore a minimal separating pattern for $(\mathbf{u}, (\mathbf{v}_i)_i)$ uses only positions in $[1, \ell]$, and the same holds for $(u, (v_i)_i)$ since $|u| = \ell$ already forces its positions to be in $[1, \ell]$. Such a pattern depends only on the first ℓ letters of a word; as $\mathbf{u}$ and u agree on these letters, and so do $\mathbf{v}_i$ and v_i, it separates $\mathbf{u}$ from the $\mathbf{v}_i$'s if and only if it separates u from the v_i's. The two minimal sizes therefore coincide, which proves the first property.

We now prove the second property: the minimal size of a formula of $\mathbf{LTL}(Op)$ separating $\mathbf{u}$ and the $\mathbf{v}_i$ is the same as the minimal size of a formula of $\mathbf{LTL}(\mathbf{X}, \wedge)$ separating $\mathbf{u}$ and the $\mathbf{v}_i$'s. Since $\{\mathbf{X}, \wedge\} \subseteq Op$, one inequality is clear. To show the converse inequality, we consider $\varphi \in \mathbf{LTL}(Op)$ a formula of minimal size separating $\mathbf{u}$ and the $\mathbf{v}_i$'s. Since ψ_u is such a formula, we know that $|\varphi| \leq |\psi_u| = M$. Our goal is to show that φ does not contain any $\mathbf{F}$ or $\mathbf{G}$. Towards contradiction, let us assume that it does, and write $\varphi = C[\psi_1 \to x_1, \dots, \psi_m \to x_m]$ where $C \in \mathbf{LTL}(\mathbf{X}, \wedge, \vee)$ is a "context" (a formula with some free variables $x_1, \dots, x_m$) and $\psi_j \in \mathbf{LTL}(Op)$ are of the form either $\mathbf{F}\psi'$ or $\mathbf{G}\psi'$.

Since the $\mathbf{X}$ operator commutes with $\wedge$ and $\vee$, we can push the $\mathbf{X}$ to the variables in C. This yields a Boolean context B such that $B[\mathbf{X}^{j_1}x_1 \to y_1, \dots, \mathbf{X}^{j_m}x_m \to y_m]$ is equivalent to C. Note that B may be larger than C in size (since for instance $\mathbf{X}(\varphi \wedge \psi) = \mathbf{X}\varphi \wedge \mathbf{X}\psi$), but they have the same depth. Since the size of C is at most M, we have $j_p \leq M$ for all $p \in [1, m]$.

We claim that:

- $\mathbf{u}$ satisfies $\mathbf{X}^{j_p}\psi_p$ for all $p \in [1, m]$;
- $\mathbf{v}_i$ satisfies $\mathbf{X}^{j_p}\psi_p$ for all $p \in [1, m]$ and all $i \in [1, n]$.

We first establish the first bullet. Note that B does not contain any negation, thus if $\mathbf{u}$ did not satisfy some $\mathbf{X}^{j_p}\psi_p$, then $\mathbf{u}$ would still satisfy $B[\mathbf{X}^{j_1}\psi_1 \to y_1, \dots, \mathbf{X}^{j_p}\bot \to y_p, \dots, \mathbf{X}^{j_m}\psi_m \to y_m]$, while no $\mathbf{v}_i$ satisfies it, since the $\mathbf{v}_i$ do not even satisfy $B[\mathbf{X}^{j_1}\psi_1 \to y_1, \dots, \mathbf{X}^{j_m}\psi_m \to y_m]$. As a result, $C[\psi_1 \to x_1, \dots, \bot \to x_p, \dots, \psi_m \to x_m]$ would be satisfied by $\mathbf{u}$ but no $\mathbf{v}_i$, contradicting the minimality of φ.

Let $1 \leq p \leq m$, let $1 \leq i \leq n$, we show that $\mathbf{v}_i$ satisfies $\mathbf{X}^{j_p}\psi_p$. We distinguish two cases:

- $\psi_p = \mathbf{G}\psi'$. Since $\mathbf{v}_i$ is a suffix of $\mathbf{u}$ of length greater than M, $\mathbf{v}_i[j_p \dots]$ is not empty and is a suffix of $\mathbf{u}[j_p \dots]$. As the latter satisfies $\mathbf{G}\psi'$, so does the former. Hence $\mathbf{v}_i \vDash \mathbf{X}^{j_p}\psi_p$.
- $\psi_p = \mathbf{F}\psi'$. By the first claim, $\mathbf{u}[j_p \dots] \vDash \mathbf{F}\psi'$, so some suffix s of $\mathbf{u}[j_p \dots]$ satisfies ψ'. As s is a suffix of $\mathbf{u} = z^{M+1}$, we can write $s = z[r \dots] z^t$ for some $r \in [1, |z|]$ and $t \in [0, M]$. Since ψ' is a strict subformula of ψ_p, we have $|\psi'| \leq |\psi_p| - 1 \leq M - 1$. Hence, by Fact 4 applied with $m = M - 1$, we may decrease t down to $\min(t, M-1)$ while preserving satisfaction of ψ': the word $s' = z[r \dots] z^{\min(t, M-1)}$ also satisfies ψ', and $|s'| \leq |z| + (M-1)|z| = M|z|$. Now $\mathbf{v}_i[j_p \dots]$ is a suffix of $\mathbf{u}$ of length at least $|z^M| = M|z|$, since $\mathbf{v}_i$ ends with z^M and $j_p \leq M < |v'_i \cdots v'_n|$; in particular $|\mathbf{v}_i[j_p \dots]| \geq |s'|$. As two suffixes of the same word $\mathbf{u}$ are nested by length, s' is a suffix of $\mathbf{v}_i[j_p \dots]$, so $\mathbf{v}_i[j_p \dots] \vDash \mathbf{F}\psi' = \psi_p$. Hence $\mathbf{v}_i \vDash \mathbf{X}^{j_p}\psi_p$.

Now consider the formula $\varphi' = C[\top \to x_1, \dots, \top \to x_m]$, which is equivalent to $B[\mathbf{X}^{j_1}\top \to y_1, \dots, \mathbf{X}^{j_m}\top \to y_m]$. As all $\mathbf{v}_i$ satisfy all ψ_p but not φ, no $\mathbf{v}_i$ satisfies φ'. Further, as $\mathbf{u}$ satisfies φ and all ψ_p, it also satisfies φ'. This contradicts the minimality of φ, unless $m = 0$. As a result, φ does not contain any $\mathbf{F}$ or $\mathbf{G}$, hence $\varphi \in \mathbf{LTL}(\mathbf{X}, \wedge, \vee)$. Since there is a single positive word $\mathbf{u}$, Lemma 6 yields a formula of $\mathbf{LTL}(\mathbf{X}, \wedge)$ of size at most $|\varphi|$ separating $\mathbf{u}$ from the $\mathbf{v}_i$'s. Together with the converse inequality $\mathbf{LTL}(\mathbf{X}, \wedge) \subseteq \mathbf{LTL}(Op)$, this proves the second property.

Combining the two properties, the minimal size of a formula of $\mathbf{LTL}(Op)$ separating $\mathbf{u}$ from the $\mathbf{v}_i$'s equals the minimal size of a formula of $\mathbf{LTL}(\mathbf{X}, \wedge)$ separating u from the v_i's. Finally, the words $\mathbf{u}$ and $\mathbf{v}_i$ have length $O(n\ell^4)$ and are computable in polynomial time, so $(u, v_1, \dots, v_n) \mapsto (\mathbf{u}, \mathbf{v}_1, \dots, \mathbf{v}_n)$ together with the unchanged size bound k is a polynomial-time reduction from the $\mathbf{LTL}(\mathbf{X}, \wedge)$ learning problem (for a single positive word) to the $\mathbf{LTL}(Op)$ one. □

Proposition 12. *For all $\{\boldsymbol{X}, \vee\} \subseteq Op \subseteq \{\boldsymbol{F}, \boldsymbol{G}, \boldsymbol{X}, \wedge, \vee\}$, there is a polynomial-time reduction from the $\boldsymbol{LTL}(\boldsymbol{X}, \vee)$ learning problem for a single negative word to the $\boldsymbol{LTL}(Op)$ one over the same alphabet.*

Proof. The proof is identical to the one of Proposition 11, with the roles of positive and negative words reversed and disjunctions and conjunctions reversed. □

8. Almost all fragments without the next operator are hard to approximate

This section contains the most technical results of the paper: hardness for the fragments *without* the next operator, for a fixed alphabet of size 3. The difficulty is that, lacking $\mathbf{X}$, a formula can no longer point at a specific position, so the reductions and especially the lower bounds require substantially more work. The section is organised in two movements. First we prove hardness for $\mathbf{LTL}(\mathbf{F}, \wedge)$ for a fixed alphabet of size 3 (Theorem 9); the heart of the argument is a lower bound established through a formula-size game in the style of Adler and Immerman. Then allowing negation throughout lets us derive a dual hardness result for fragments built from $\mathbf{G}$ and $\vee$ (Theorem 10) almost for free.

Hardness result for fixed alphabet

We now come to the central result of the section, which strengthens Theorem 3 by restricting to an alphabet of size 3.

Theorem 9. *For all $\{\boldsymbol{F}, \wedge\} \subseteq Op \subseteq \{\boldsymbol{F}, \boldsymbol{G}, \wedge, \vee, \neg\}$, the learning problem for $\boldsymbol{LTL}(Op)$ is* NP*-hard even for an alphabet of size* 3*.*

Its main difficulty stems from the use of a fixed alphabet, combined with the absence of $\mathbf{X}$, which forbids us from pointing at a specific position. To circumvent this problem we will adapt the previous reductions from hitting set: we again use a single positive example u against several negative ones $v_1, \dots, v_n$. Each of those words is a sequence of factors $abab\cdots$ (one for each subset) separated by a letter c. We introduce small differences between the factors in the v_i and in u. We ensure that detecting those differences is costly in terms or

operators, and thus that a minimal formula has to "talk about" as few of them as possible. The main difficulty in the proof is showing that we do not have more efficient ways to separate the words. Much like in the previous hardness proofs, the minimal formula has to select a minimal amount of positions (here factors) in the words that are enough to distinguish u from the v_i. Those positions answer the hitting set problem. We temporarily allow the negation operator as it does not make the proof more difficult and allows us to obtain a dual version of Theorem 9 easily.

Let us consider an instance of the hitting set problem: $T_1, \dots, T_n \subseteq [1, m]$. We set $M = 3m + 2$, $u = (c\,(ab)^{M+1})^m c = c\,(ab)^{M+1} c\,(ab)^{M+1} c \cdots (ab)^{M+1} c$ and for all $i \in [1, n]$, we define $v_i = c w_{i,1} c w_{i,2} c \cdots w_{i,m} c$ with

$$w_{i,j} = \begin{cases} (ab)^M & \text{if } j \in T_i, \\ (ab)^{M+1} & \text{otherwise.} \end{cases} \tag{1}$$

We will prove that there exists a hitting set of $(T_j)_{1 \le j \le n}$ of size k if and only if there exists a separating formula in $\mathbf{LTL}(\mathbf{F}, \wedge)$ of size at most $6kM + 9m + 2$

Claim 1. *If there exists a hitting set of $(T_j)_{1 \le j \le n}$ of size k, then there exists a separating formula in $\boldsymbol{LTL}(\boldsymbol{F}, \wedge)$ of size $6kM + 9m + 2$.*

Proof. Let H be a hitting set of size k. We define for $j \in [1, m]$,

$$z_j = \begin{cases} (ab)^{M+1} & \text{if } j \in H, \\ ab & \text{otherwise.} \end{cases}$$

and $w = c\ z_1\ c\ z_2\ c\ \cdots z_m\ c$. Let us write $w = x_1 \cdots x_p$ where $x_1, \dots, x_p$ are letters, and define the formula

$$\psi = \mathbf{F}(x_1 \wedge \mathbf{F}(x_2 \wedge \mathbf{F}(x_3 \wedge \mathbf{F}(\cdots \wedge \mathbf{F} x_p)))).$$

This formula has size

$$3p - 1 = 3\Bigg(\underbrace{m+1}_{\text{number of } c\text{'s}} + \underbrace{2(M+1)\cdot k}_{\text{letters in } (ab)^{M+1}\text{'s}} + \underbrace{2\cdot(m-k)}_{\text{letters in } ab\text{'s}}\Bigg) - 1 = 6kM + 9m + 2.$$

We claim that ψ is a separating formula. First, note that since all letters in w are non-repeating, ψ is satisfied by exactly the words which have w as a subword.

- Since w is a subword of u, we have that u satisfies ψ.
- Let $i \in [1, n]$, we show that v_i does not satisfy ψ. Since v_i and w contain the same number of c's, for w to be a subword of v_i, z_j needs to be a subword of $w_{i,j}$ for all $j \in [1, m]$. However, as H is a hitting set of $(T_j)_{1 \le j \le n}$, there exists $j \in H$ such that $i \in T_j$. Thus $w_{i,j} = (ab)^M$ and $z_j = (ab)^{M+1}$, implying that w is not a subword of v_i, so v_i does not satisfy ψ.

□

We turn to the lower bound. For $I \subseteq [1,n]$ and $j \in [1, m+1]$, write

$$h_j(I) = \min\big\{|H| : H \subseteq [j,m],\ H \cap T_i \neq \varnothing \text{ for all } i \in I\big\},$$

with value $+\infty$ if no such H exists. We use $h_{m+1}(\varnothing) = 0$. Observe that for all $I \subseteq I'$, I_1, I_2 and $j \leq j'$,

$$h_j(I) \leq h_j(I'), \qquad h_j(I) \leq h_{j'}(I), \qquad h_j(I_1 \cup I_2) \leq h_j(I_1) + h_j(I_2). \tag{1}$$

Furthermore, with $K_j(I) = \{i \in I : j \in T_i\}$,

$$h_j(I) \leq 1 + h_{j+1}\big(I \setminus K_j(I)\big). \tag{2}$$

These follow by taking unions of hitting sets and, in the last case, adding position j.

Suffix configurations. For $j \in [1, m+1]$ put

$$U_j = \big(c(ab)^{M+1}\big)^{m-j+1}c, \qquad V_{i,j} = cw_{i,j}c\cdots cw_{i,m}c,$$

where $U_{m+1} = V_{i,m+1} = c$. A positive suffix has the form $u' = b^y(ab)^\ell U_j$ ($y \in \{0,1\}$). A negative suffix *matches* it when it equals $b^y(ab)^\ell V_{i,j}$, and *follows* it when it equals $b^y(ab)^{\ell-1}V_{i,j}$; the latter is defined only for $\ell \geq 1$. The names refer to the remaining alternating prefix before the next c.

We use three families of configurations (A, B), with A positive and B negative suffixes:

Type 1. $A = \{u'\}$, where $u' = b^y(ab)^\ell U_j$, and $B = \{v'_i : i \in I\}$, with $2\ell + y \geq 3$. For $i \in K_j(I)$ the negative suffix v'_i follows u', and for $i \notin K_j(I)$ it matches u'.

Type 2. The configuration is obtained from type 1 by deleting the first letter of the positive suffix and of every matching negative suffix.

Type 3. $A = \{u'_i : i \in I\}$ and $B = \{v'_i : i \in I\}$, where each v'_i matches u'_i.

We define the cost of an instance of type 1 as follows.

$$C_1(j, \ell, y, I) = \min\Big\{6(M-1)h_j(I),\ 6(M-1)h_j\big(I \setminus K_j(I)\big) + 3(2\ell + y - 3)\Big\}. \tag{3}$$

Type 2 inherits the cost of the corresponding type 1 configuration. For type 3, with current indices j_i, we assign the cost

$$C_3((j_i)_{i\in I}) = 6(M-1)h_{\min_{i\in I} j_i}(I).$$

The intuition behind the cost is that it estimates the size of a separating *fattern*. Specifically, it estimates the size of the formula resulting from the strategy applied for the upper bound: we select some minimal hitting set H and write a formula expressing that the word contains $(ab)^{M+1}$ on blocks corresponding to H. To show that this is the best we can do, we extend this to suffixes. We associate suffixes of u with very similar suffixes of the negative words and show that they are hard to separate. For now say we have only **F** and $\wedge$. To separate u from the v_i, we have to start the formula with one of those operators. Using a $\wedge$ means partitioning the v_i in two sets, depending on the side of the conjunction they do

not satisfy. This amounts to building a hitting set for both parts, which brings no benefit: the hitting set size is sub-additive. We will see that $\wedge$ is only useful when some of the v_j suffixes start with a different letter than the suffix of u: then a formula $x \wedge \psi$ with x the starting letter of the suffix of u eliminates some of the negative suffixes.

Using an **F** means the rest of the formula must be satisfied by some suffix $u' = b^y(ab)^\ell(c(ab)^{M+1})^j$ of u. For each v_i, we select a suffix that is as close as possible: same block, matching if possible, otherwise following u'. The choice of u' amounts to either starting to consume the first block (i.e. putting it in the hitting set) or jumping to a later block.

Example 1. *Consider the hitting set instance $T_1 = \{1, 3\}$ and $T_2 = \{2, 3\}$. The translation into a learning instance is the following, with $M = 3m + 2 = 11$.*

$$\begin{aligned} u &= c(ab)^{M+1}c(ab)^{M+1}c(ab)^{M+1}c \\ v_1 &= c(ab)^{M+1}c(ab)^{M}c(ab)^{M}c \\ v_2 &= c(ab)^{M}c(ab)^{M+1}c(ab)^{M}c \end{aligned}$$

An example of an configuration of type 1 induced by this instance is the following.

$$\begin{aligned} u' &= ababc(ab)^{M+1}c \\ v'_1 &= abc(ab)^{M}c \\ v'_2 &= ababc(ab)^{M}c \end{aligned}$$

Here v'_1 follows u' while v'_2 matches it. Adding a b to the left of v'_1 yields a configuration of type 2 while setting $u'_2 = u'$ and $u'_1 = abc(ab)^{M+1}c$ yields a configuration of type 3.

Lemma 7. *Assume that $\ell > 1$ or $\ell = y = 1$. For all I_1, I_2 with $I = I_1 \cup I_2$,*

$$C_1(j, \ell, y, I) \leq C_1(j, \ell, y, I_1) + C_1(j, \ell, y, I_2).$$

Proof. Suppose first that $C_1(j, \ell, y, I_s) = 6(M-1)h_j(I_s)$ for both $s \in \{1, 2\}$. By (1),

$$\begin{aligned} C_1(j, \ell, y, I) &\leq 6(M-1)h_j(I) \\ &\leq 6(M-1)\big(h_j(I_1) + h_j(I_2)\big) \\ &= C_1(j, \ell, y, I_1) + C_1(j, \ell, y, I_2). \end{aligned}$$

Otherwise, by symmetry, we may assume that

$$C_1(j, \ell, y, I_2) = 6(M-1)h_j\big(I_2 \setminus K_j(I_2)\big) + 3(2\ell + y - 3).$$

Observe that

$$I \setminus K_j(I) = \big(I_1 \setminus K_j(I_1)\big) \cup \big(I_2 \setminus K_j(I_2)\big).$$

Hence, again by (1),

$$\begin{aligned} C_1(j, \ell, y, I) &\leq 6(M-1)h_j\big(I \setminus K_j(I)\big) + 3(2\ell + y - 3) \\ &\leq 6(M-1)h_j\big(I_1 \setminus K_j(I_1)\big) \\ &\qquad + 6(M-1)h_j\big(I_2 \setminus K_j(I_2)\big) + 3(2\ell + y - 3) \\ &= 6(M-1)h_j\big(I_1 \setminus K_j(I_1)\big) + C_1(j, \ell, y, I_2) \\ &\leq C_1(j, \ell, y, I_1) + C_1(j, \ell, y, I_2). \end{aligned}$$

The last inequality follows from $h_j(I_1 \setminus K_j(I_1)) \leq h_j(I_1)$ and the definition of C_1. □

Lemma 8. *For a type-1 or type-2 configuration, every separating Boolean combination of formulas beginning with* $\boldsymbol{F}$ *or* $\boldsymbol{G}$ *has size at least its assigned cost. For a type-3 configuration, every separating formula has size at least its assigned cost.*

Proof. We proceed by structural induction on a separating formula φ. Let (A, B) a configuration.

Boolean connectives. For types 1 and 2 and $\varphi = \psi_1 \vee \psi_2$, every negative suffix must dissatisfy both ψ_1 and ψ_2, while the positive word u' satisfies one of the two, say ψ_1. Then ψ_1 separates (A, B), and has size at least its cost by induction hypothesis.

For types 1 and 2 and $\varphi = \psi_1 \wedge \psi_2$, every negative suffix must dissatisfy either ψ_1 or ψ_2. We split the indices according to this: $I = I_1 \cup I_2$ depending on which side v_i dissatisfies. Then ψ_s separates $(A, \{v'_i \mid i \in I_s\})$ for both $s \in \{1, 2\}$. By induction hypothesis and Lemma 7, we obtain the result.

For type 3, $\wedge$ and $\vee$ behave symmetrically, so we only treat the $\vee$ case. For each $i \in I$, u_i must satisfy either ψ_1 or ψ_2 while v_i must satisfy neither. We split the indices according to this: $I = I_1 \cup I_2$ depending on which side u_i satisfies. By induction hypothesis and (1), we get the result.

Always. If $\varphi = \mathbf{G}\psi$, choose for each negative suffix v'_i a suffix v''_i rejecting ψ. Choose a suffix of one of the positive suffix u''_i matching v''_i (it always exists since v'_i always matches a suffix of one of the positive suffixes). Since the original positive suffixes satisfy $\mathbf{G}\psi$, each u''_i satisfies ψ. This produces a type-3 configuration.

We show that for all three types this cannot decrease the cost. For type 3 this is a direct consequence of the monotonicity of hitting set with respect to j (1). For types 1 and 2 the original cost was at most $6(M-1)h_j(I)$, and is now $6(M-1)h_{j''}(I)$ for some $j'' \geq j$: we again apply (1).

Eventually. Suppose $\varphi = \mathbf{F}\psi$, so that ψ is a Boolean combination of formulas starting with $\mathbf{F}$ or $\mathbf{G}$. The proof amounts to a large case distinction. For type 1 or 2 let u'' a suffix of u' satisfying ψ. For type 3 u'' is set as a suffix of the shortest u'_i (hence of every u'_i) satisfying ψ.

The idea is that we will pick for each v'_i a suffix as close to u'' as possible: same block, and as close as possible in this block. If we can find a matching suffix then we pick it. If not, we make it as close to matching as possible. We distinguish two cases, depending on whether ψ is a Boolean combination of formulas starting with temporal operators (e.g. $\psi = \mathbf{F}\psi_1 \wedge \mathbf{G}\psi_2$) or has a top-level literal (e.g. $\psi = \mathbf{F}a \wedge \mathbf{F}\psi_1$). In the first case we don't mind picking a suffix of v'_i starting with a different letter than u'', while in the second case we make sure that they start the same. The first case will often result in instances of type 2 and the second in instances of type one.

We first take care of type 3. Recall that u'' is a suffix of each u'_i, which v'_i matches. Let j' be the block in which the shortest u'_i is.

- If u'' starts on the first a of a block j'', then $j'' > j'$ and for each $i \in I$ we set v''_i as the suffix of v'_i matching u'' if it exists, and the suffix following u'' otherwise. This yields an instance of type 1, of cost $6(M-1)H^I_{\geq j''}$.

- If u'' starts on the first b of a block j'', then $j'' > j'$ and for each $i \in I$ we set v_i'' as the suffix of v_i' matching u'' if it exists, and the suffix matching u'' minus its first letter otherwise. This yields an instance of type 2, of cost $6(M-1)H^I_{\geq j''}$.
- If u'' starts on a later position of a block j'', then $j'' \geq j'$ and for each $j \in I$ we set v_i'' as the suffix of v_i' matching u'', which always exists. This yields an instance of type 3 (with $u_i' = u''$ for all j), of cost $6(M-1)H^I_{\geq j''+1}$.

In all three cases the resulting cost is at least $6(M-1)H^I_{\geq j'+1}$, which is greater or equal to the cost of (A, B).

Now suppose (A, B) is of type 1, type 2 is similar. We set $K = \{i \in I \mid v_i' \text{ follows } u'\} = I \cap \{i \mid j \in T_i\}$.

- Suppose $|u''| \leq |u'| - 2$. We know that v_i'' matches u' or u' minus one or two letters. If $|u''| \leq |u'| - 2$ then we can always select v_i'' the same way as for type 3. We do so, and the proof goes through the same way.
- If $u'' = u'$ then ψ must be separating for (A, B): we apply the induction hypothesis.

For the case $|u''| = |u'| - 1$ we need to know whether ψ has a top-level literal or not.

If ψ is a boolean combination of formulas starting with $\mathbf{F}$ or $\mathbf{G}$, then for each $i \in I$ we set v_i'' as either v_i' minus the first letter if v_i' matches u' or simply v_i' if v_i' follows u'. Then ψ must be separating for $(\{u''\}, \{v_i'' \mid j \in I\})$, which is of type 2, with the same cost as (A, B).

If ψ is a boolean combination of formulas starting with $\mathbf{F}$ or $\mathbf{G}$ and at least one literal, let $\bar{\psi}$ the formula by replacing each literal in that boolean combination by $\top$ if u'' satisfies it and $\bot$ otherwise, and eliminating them. The size of $\bar{\psi}$ is at most the one of $\mathbf{F}(\psi)$ minus 3 (we removed the leading $\mathbf{F}$, at least one literal and thus a binary operator as well). Observe that $\bar{\psi}$ must be satisfied by u'' but not by any suffix of the v_i' starting with the same letter. Also observe that since $2\ell + y \geq 3$, all v_i' start with the same letter as u'. We set v_i'' as v_i' minus the first letter for all $i \in I$. We remove from I all i such that v_i'' starts with c, which does not change the cost. Then $\bar{\psi}$ must be separating for $(\{u''\}, \{v_i'' \mid j \in I\})$, which is of type 1, with the same cost as (A, B) minus 3.

We have gone through all cases and shown that they preserve the lower bound, proving the induction. □

Proposition 13. *Any formula over $\boldsymbol{F}, \boldsymbol{G}, \wedge, \vee$ and literals (including negated literals) separating u from all v_i has size at least $6(M-1)h_1([1, n])$.*

Proof. All the examples begin with c. Thus any literals occurring in the outermost Boolean combination have the same truth value on every example and can be simplified away. The remaining formula is a Boolean combination of temporal formulas. The initial configuration is the one associated with u and all v_i; its cost is $6(M-1)h_1([1, n])$. Apply Lemma 8. □

Assuming the lower bound, the reduction is complete. Indeed, if there is no hitting set of size at most k, then every separating formula has size at least $6(k+1)(M-1)$, and

$$6(k+1)(M-1) - (6kM + 9m + 2) = 9m + 4 - 6k > 0$$

because $k \leq m$. Together with Claim 1, this gives the required equivalence and hence NP-hardness.

Dual hardness result

We now show hardness for fragments with operators **G** and $\vee$. As we allowed negation in the previous result, we can infer that one almost directly.

Theorem 10. *For all $\{\boldsymbol{G}, \vee\} \subseteq Op \subseteq \{\boldsymbol{F}, \boldsymbol{G}, \wedge, \vee\}$, the learning problem for $\boldsymbol{LTL}(Op)$ is* NP*-hard even for an alphabet of size* 3*.*

We take the same instance of the hitting problem, but this time we consider the $\mathbf{LTL}(Op)$ learning problem with the v_i as positive words and u as the only negative one.

Corollary 1. *Let φ be an $\boldsymbol{LTL}(\boldsymbol{F}, \boldsymbol{G}, \wedge, \vee)$ formula separating $v_1, \dots, v_n$ (positive) from u (negative), then $6k(M-1) \leq |\varphi|$.*

Proof. Let φ be such a formula, and let $\overline{\varphi}$ be its negation, with the negations pushed to the bottom. Formally, $\overline{\varphi}$ is defined by induction on φ:

- $\overline{a} = \neg a$
- $\overline{\varphi_1 \wedge \varphi_2} = \overline{\varphi_1} \vee \overline{\varphi_2}$
- $\overline{\varphi_1 \vee \varphi_2} = \overline{\varphi_1} \wedge \overline{\varphi_2}$
- $\overline{\mathbf{F}\varphi_1} = \mathbf{G}\overline{\varphi_1}$
- $\overline{\mathbf{G}\varphi_1} = \mathbf{F}\overline{\varphi_1}$

A clear induction shows that a word satisfies one if and only if it does not satisfy the other (note that this would not be true if we allowed the operator **X**). Hence $\overline{\varphi}$ is satisfied by u but not by the v_i, that is, $\overline{\varphi}$ separates u from $v_1, \dots, v_n$ as in Theorem 9; since negation normal form preserves the number of nodes, $|\overline{\varphi}| = |\varphi|$, and Proposition 13 gives $6k(M-1) \leq |\overline{\varphi}| = |\varphi|$. □

Lemma 9. *There exists a formula ψ of $\boldsymbol{LTL}(\boldsymbol{G}, \vee)$ separating $v_1, \dots, v_n$ (positive) from u (negative) with $|\psi| \leq 6kM + 11m + 4$.*

Proof. Let H be a hitting set of $(T_i)_{1 \leq i \leq n}$ of size k. We set, for all $j \in [1, m]$,

$$z_j = \begin{cases} (ab)^{M+1} & \text{if } j \in H, \\ ab & \text{otherwise,} \end{cases}$$

and $w = cz_1cz_2c \cdots z_mc$. We use the formula $\psi = \mathbf{G}(\overline{x_1} \vee \mathbf{G}(\overline{x_2} \vee \mathbf{G}(\overline{x_3} \vee \mathbf{G}(\cdots \vee \mathbf{G}\overline{x_p}))))$, where $x_1, \dots, x_p$ are the letters such that $x_1 \cdots x_p = w$, $\overline{a} = b$, $\overline{b} = a$ and $\overline{c} = a \vee b$. This formula has size $3p - 1 + 2(m+1) = 3(2kM + 3m + 1) + 2m + 1 \leq 6kM + 11m + 4$.

Observe that ψ is satisfied by exactly the words which do not have w' as a weak subword, where $w' \in (2^\Sigma)^*$ is obtained from w by replacing each c by $\{c\}$, each a by $\{a, c\}$ and each b by $\{b, c\}$. As w is a weak subword of u, so is w', hence u does not satisfy ψ. Let $i \in [1, n]$; since v_i and w contain the same number of c's, for v_i to fail ψ (i.e. to contain w' as a weak subword), z_j would need to be a subword of $w_{i,j}$ for all $j \in [1, m]$, which in particular requires $w_{i,j} = (ab)^{M+1}$ for all $j \in H$. However, as H is a hitting set, there exists $j \in H$ such that $i \in T_j$, and thus $w_{i,j} = (ab)^M$. Therefore every v_i satisfies ψ. □

Let φ be a formula of minimal size satisfied by all v_i but not by u. By Corollary 1 and Lemma 9, we have $6kM - 6k \leq |\varphi| \leq 6kM + 11m + 4$.

We set $K = 6k'M + 11m + 4$ and show that $k' \geq k$ if and only if $|\varphi| \leq K$.

- Suppose $k' \geq k$, then $K \geq 6kM + 11m + 4 \geq |\varphi|$.
- Suppose $k' \leq k-1$, then, using $M = 3m+2$ and $k \leq m$, $K \leq 6(k-1)M + 11m + 4 = 6kM - 6M + 11m + 4 = 6kM - 7m - 8 < 6kM - 6k \leq |\varphi|$, the strict inequality holding because $6k \leq 6m < 7m + 8$.

As a result, the hitting set problem has a positive answer on instance $S, T_1, \ldots, T_n, k'$ if and only if so does the $\mathbf{LTL}(Op)$ learning problem on instance $u, v_1, \ldots, v_n, K$. As the latter instance is constructible in polynomial time from the former, Theorem 10 is proven.

9. Perspectives and open problems

In this paper, we showed NP-completeness of the $\mathbf{LTL}(\mathbf{F}, \mathbf{G}, \mathbf{X}, \wedge, \vee)$ learning problem for all fragments, giving a complete picture of its complexity. This paper does not consider the until operator $\mathbf{U}$; extending the picture to **LTL** with $\mathbf{U}$ remains open. In particular the following question is open:

Open problem 1. *Is the learning problem* NP*-complete for* $\boldsymbol{LTL}(\boldsymbol{U}, \boldsymbol{F}, \boldsymbol{G}, \boldsymbol{X}, \wedge, \vee)$ *with a fixed constant-size alphabet?*

We consider **LTL** over finite traces; some of the related works we discussed in the introduction actually considered infinite traces as common in the verification research community. All our hardness results easily transfer to the infinite trace case, simply by extending finite traces into infinite ones with a new symbol. In particular, we obtain as corollary that the **LTL** learning problem is NP-hard for infinite traces when the alphabet is part of the input.

The negative results in this paper suggest looking for approximation algorithms. Beyond $\mathbf{LTL}(\mathbf{X}, \wedge)$, nothing is known in terms of upper and lower bounds on polynomial time approximation algorithms, leaving an open field of exciting research directions. In the same vein, one could wonder about the parameterized complexity of the **LTL** learning problem, in particular when fixing the number of words. We leave this question open and hope to inspire further studies!

References

Baharisangari, N., Gaglione, J., Neider, D., Topcu, U., & Xu, Z. (2021). Uncertainty-aware signal temporal logic inference. In Bloem, R., Dimitrova, R., Fan, C., & Sharygina, N. (Eds.), *International Conference on Software Verification, VSTTE*, Vol. 13124 of *Lecture Notes in Computer Science*, pp. 61–85. Springer.

Bombara, G., Vasile, C. I., Penedo Alvarez, F., Yasuoka, H., & Belta, C. (2016). A Decision Tree Approach to Data Classification using Signal Temporal Logic. In *Hybrid Systems: Computation and Control, HSCC*.

Camacho, A., & McIlraith, S. A. (2019). Learning interpretable models expressed in linear temporal logic. *International Conference on Automated Planning and Scheduling, ICAPS, 29*.

Dinur, I., & Steurer, D. (2014). Analytical approach to parallel repetition. In *Symposium on Theory of Computing, STOC*, pp. 624–633.

Ehlers, R., Gavran, I., & Neider, D. (2020). Learning properties in LTL ∩ ACTL from positive examples only. In *Formal Methods in Computer Aided Design, FMCAD*.

Fijalkow, N., & Lagarde, G. (2021). The complexity of learning linear temporal formulas from examples. In Chandlee, J., Eyraud, R., Heinz, J., Jardine, A., & van Zaanen, M. (Eds.), *International Conference on Grammatical Inference, ICGI*, Vol. 153 of *Proceedings of Machine Learning Research*, pp. 237–250. PMLR.

Gaglione, J., Neider, D., Roy, R., Topcu, U., & Xu, Z. (2022). Maxsat-based temporal logic inference from noisy data. *Innovations in Systems and Software Engineering*, *18*(3), 427–442.

Gold, E. M. (1978). Complexity of automaton identification from given data. *Information and Control*, *37*(3), 302–320.

Kim, J., Muise, C., Shah, A., Agarwal, S., & Shah, J. (2019). Bayesian inference of linear temporal logic specifications for contrastive explanations. In *International Joint Conference on Artificial Intelligence, IJCAI*.

Lemieux, C., Park, D., & Beschastnikh, I. (2015). General LTL specification mining. In *International Conference on Automated Software Engineering, (ASE)*.

Lutz, S., Neider, D., & Roy, R. (2023). Specification sketching for linear temporal logic. In André, É., & Sun, J. (Eds.), *International Symposium on Automated Technology for Verification and Analysis, ATVA*, Vol. 14216 of *Lecture Notes in Computer Science*, pp. 26–48. Springer.

Neider, D., & Gavran, I. (2018). Learning linear temporal properties. In *Formal Methods in Computer Aided Design, FMCAD*, pp. 1–10.

Pitt, L., & Warmuth, M. K. (1993). The minimum consistent DFA problem cannot be approximated within any polynomial. *Journal of the ACM*, *40*(1), 95–142.

Pnueli, A. (1977). The temporal logic of programs. In *Symposium on Foundations of Computer Science, SFCS*.

Raha, R., Roy, R., Fijalkow, N., & Neider, D. (2022). Scalable anytime algorithms for learning fragments of linear temporal logic. In Fisman, D., & Rosu, G. (Eds.), *International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS*, Vol. 13243 of *Lecture Notes in Computer Science*, pp. 263–280. Springer.

Raha, R., Roy, R., Fijalkow, N., Neider, D., & Pérez, G. A. (2024). Synthesizing efficiently monitorable formulas in metric temporal logic. In *International Conference on Verification, Model Checking, and Abstract Interpretation, VMCAI*.

Roy, R., Fisman, D., & Neider, D. (2020). Learning interpretable models in the property specification language. In *International Joint Conference on Artificial Intelligence, IJCAI*.