Abstract

We study the expressivity and complexity of model checking linear temporal logic with team semantics (TeamLTL). TeamLTL, despite being a purely modal logic, is capable of defining hyperproperties, i.e., properties which relate multiple execution traces. TeamLTL has been introduced quite recently and only few results are known regarding its expressivity and its model checking problem. We relate the expressivity of TeamLTL to logics for hyperproperties obtained by extending LTL with trace and propositional quantifiers (HyperLTL and HyperQPTL). By doing so, we obtain a number of model checking results for TeamLTL and identify its undecidability frontier. In particular, we show decidability of model checking of the so-called left-flat fragment of any downward closed TeamLTL-extension. Moreover, we establish that the model checking problem of TeamLTL with Boolean disjunction and inclusion atoms is undecidable.

1 Introduction

Linear-time temporal logic (LTL) is one of the most prominent logics for the specification and verification of reactive and concurrent systems. Practical model checking tools like SPIN, NuSMV, and many others ([29, 6, 11]) automatically verify whether a given computer system, such as a hardware circuit or a communication protocol, is correct with respect to its LTL specification. The basic principle, as introduced in 1977 by Amir Pnueli [40], is to specify the correctness of a program as a set of infinite sequences, called traces, which define the acceptable executions of the system.
Hyperproperties, i.e., properties which relate multiple execution traces, cannot be specified in LTL. Such properties are of prime interest in information flow security, where dependencies between the secret inputs and the publicly observable outputs of a system are considered potential security violations. Commonly known properties of that type are noninterference [42, 38] or observational determinism [50]. In other settings, relations between traces are explicitly desirable: robustness properties, for example, state that similar inputs lead to similar outputs. Hyperproperties are not limited to the area of information flow control. E.g., distributivity and other system properties like fault tolerance can be expressed as hyperproperties [17].

The main approach to specify hyperproperties has been to extend temporal logics like LTL, CTL, and QPTL with explicit trace and path quantification, resulting in logics like HyperLTL [7], HyperCTL* [7], and HyperQPTL [41, 9]. Most frequently used is HyperLTL, which can express noninterference as follows: \( \forall \pi. \forall \pi'. \Box (\bigwedge_{i \in I} i \pi \leftrightarrow i \pi') \rightarrow \Box (\bigwedge_{o \in O} o \pi \leftrightarrow o \pi') \). The formula states that any two traces which globally agree on the value of the public inputs \( I \) also globally agree on the public outputs \( O \). Consequently, the value of secret inputs cannot affect the value of the publicly observable outputs.

It is not clear, however, whether quantification over traces is the best way to express hyperproperties. The success of LTL over first-order logics for the specification of linear-time points in time. This allows for a much more concise and readable formulation of the same property. The natural question to ask is whether a purely modal logic for hyperproperties would have similar advantages. A candidate for such a logic is LTL with team semantics [34]. Under team semantics, LTL expresses hyperproperties without explicit references to traces. Instead, each subformula is evaluated with respect to a set of traces, called a team. Temporal operators advance time on all traces of the current team. Using the split operator \( \triangledown \), teams can be split during the evaluation of a formula, which enables us to express properties of subsets of traces.

As an example, consider the property that there is a point in time, common for all traces, after which a certain event \( a \) does not occur any more. We need a propositional and a trace quantifier to express such a property in HyperQPTL (it is not expressible in HyperLTL). The formula \( \exists p. \forall \pi. \Diamond p \land \Box (p \rightarrow \Box \neg a) \) states that there is a \( p \)-sequence \( s \in (2^{|p|})^\omega \) such that \( p \) is set at least once, and if \( p \in s[i] \), then \( a \) is not set on all traces \( \pi \) on all points in time starting from \( i \). The same property can be expressed in TeamLTL without any quantification simply as \( \Box \Diamond \neg a \). The formula exploits the synchronous semantics of TeamLTL by stating that there is a point such that for all future points all traces have \( a \) not set. As a second example, consider the case that an unknown input determines the behaviour of the system. Depending on the input, its execution traces either agree on \( a \) or on \( b \). We can express the property in HyperLTL with three trace quantifiers: \( \exists \pi_1. \exists \pi_2. \forall \pi. \Box (a_{\pi_1} \leftrightarrow a_\pi) \lor \Box (b_{\pi_2} \leftrightarrow b_\pi) \). In TeamLTL, the same property can be simply expressed as \( \Box (a \oplus \neg a) \lor \Box (b \oplus \neg b) \). The Boolean or operator \( \oplus \) expresses that in the current team, either the left side holds on all traces or the right side does.

The use of the \( \oplus \) operator reveals another strength of TeamLTL: its modularity. The research on team semantics (see related work section) has a rich tradition of studying extensions of team logics with new atomic statements and operators. They constitute a well-defined way to increase a logic’s expressiveness in a step-by-step manner. Besides \( \oplus \), examples are Boolean negation \( \sim \), the inclusion atom \( \subseteq \), and universal subteam quantifiers \( A \) and \( \dot{A} \). Inclusion atoms have been found to be fascinating for their ability to express recursion in the first-order setting; the expressivity of \( \text{FO}(\subseteq) \) coincides with greatest fixed point logic and hence PTIME [20]. In turn, all LTL-definable properties can be expressed by
TeamLTL-formulæ of the form $A\varphi$. With the introduction of generalised atoms, TeamLTL even permits custom extensions. Possibly most interesting in the context of hyperproperties are dependence atoms. A dependence atom $\text{dep}(x_1, \ldots, x_n)$ is satisfied by a team $X$ if any two assignments assigning the same values to the variables $x_1, \ldots, x_{n-1}$ also assign the same value to $x_n$. For example, the TeamLTL formula $(\Box \text{dep}(i_1, i_2, o)) \lor (\Box \text{dep}(i_2, i_3, o))$ states that the executions of the system can be decomposed into two parts; in the first part, the output $o$ is determined by the inputs $i_1$ and $i_2$, and in the second part, $o$ is determined by the inputs $i_2$ and $i_3$.

Temporal team logics constitute a new, fundamentally different approach to specify hyperproperties. While HyperLTL and other quantification-based hyperlogics have been studied extensively (see section on related work), only few results are known about the expressive power and complexity of TeamLTL and its variants. In particular, we know very little about how the expressivity of the two approaches compares. What is known is that HyperLTL and TeamLTL are incomparable in expressivity [34] and that the model checking problem of TeamLTL without splitjunctions $\lor$ (what makes the logic significantly weaker) is in PSPACE [34]. On the other hand, it was recently shown that the complexity of satisfiability and model checking of TeamLTL with Boolean negation $\sim$ is equivalent to the decision problem of third-order arithmetic [36] and hence highly undecidable.

**Our contribution.** We advance the understanding of team-based logics for hyperproperties by exploring the relative expressivity of TeamLTL and temporal hyperlogics like HyperLTL, as well as the decidability frontier of the model checking problem of TeamLTL. Our expressivity and model checking results are summarized in Table 1 and Table 2. We identify expressively complete extensions of TeamLTL (displayed on the left of Table 1) that can express all (all downward closed, resp.) Boolean relations on LTL-properties of teams, and present several translations from team logics to hyperlogics. We begin by approaching the decidability frontier of TeamLTL from above, and tackle a question posed in [36]: *Does some sensible restriction to the use of Boolean negation in TeamLTL($\sim$) yield a decidable logic?* We show that already a very restricted access to $\sim$ leads to high undecidability, whereas already the use of inclusion atoms $\subseteq$ together with Boolean disjunctions $\lor$ suffices for undecidable model checking. Furthermore, we establish that these complexity results transfer to the satisfiability problem of the related logics. Next, regarding the expressivity of TeamLTL, we show that its extensions with all (all downward closed, resp.) atomic LTL-properties of teams translate to simple fragments of HyperQPTL. Consequently, known decidability results for quantification-based hyperlogics enable us to approach the decidability frontier of TeamLTL extensions from below. We establish an efficient translation from the so-called $k$-*coherent fragment* of TeamLTL($\sim$) to the universal fragment of HyperLTL (for which model checking is PSPACE-complete [19]) and thereby obtain EXPSPACE model checking for the fragment. Finally, we show that the so-called left-flat fragment of TeamLTL($\varnothing, A$) enjoys decidable model checking via a translation to $\exists^* \mathcal{X}^* \exists^* \mathcal{Y}^* \mathcal{Z}^* \mathcal{W}^* \mathcal{P}^*$ HyperQPTL.

**Related work.** The development of team semantics began with the introduction of Dependence Logic [46], which adds the concept of functional dependence to first-order logic by means of new atomic dependence formulæ. During the past decade, team semantics has been generalised to propositional [49], modal [47], temporal [33], and probabilistic [13] frameworks, and fascinating connections to fields such as database theory [23], statistics [12], real valued computation [24], and quantum information theory [30] has been identified. In the modal team semantics setting, model checking and satisfiability problems have been shown to be decidable, see [26, page 627] for an overview of the complexity landscape. Expressivity and definability of related logics is also well understood, see, e.g. [27, 32, 43]. The study
4 LTL with Team Semantics: Expressivity and Complexity

(assuming left-flatness)

TeamLTL(∅, 1, A) ≤ Thm. 14 3≤u∃∀HyperQPTL+

∧†

TeamLTL(∅, ∼⊥, A) ≤ Thm. 6 ∃pQ∗p∀∀HyperQPTL+

|∀ [36] (assuming k-coherence)

TeamLTL(∼) ≤ Thm. 9 ∀HyperLTL

Table 1 Expressivity results. The logics TeamLTL(∅, ∼⊥, A) and TeamLTL(1, ∅) can express all/all downward closed atomic LTL-properties of teams (see the discussion at the end of Section 2). † holds since TeamLTL(1, ∅) is downward closed.

<table>
<thead>
<tr>
<th>Logic</th>
<th>Model Checking Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>TeamLTL without ∨</td>
<td>in PSPACE [34]</td>
</tr>
<tr>
<td>k-coherent TeamLTL(∼)</td>
<td>in EXPSPACE [Thm. 10]</td>
</tr>
<tr>
<td>left-flat TeamLTL(∅, 1, A)</td>
<td>in EXPSPACE [Thm. 15]</td>
</tr>
<tr>
<td>TeamLTL(1, ∅)</td>
<td>Σ11-hard [Thm. 2]</td>
</tr>
<tr>
<td>TeamLTL(1, A)</td>
<td>Σ11-hard [Thm. 3]</td>
</tr>
<tr>
<td>TeamLTL(∼)</td>
<td>complete for third-order arithmetic [36]</td>
</tr>
</tbody>
</table>

Table 2 Complexity results.

of temporal logics with team semantics, was initiated in [33], where team semantics for computational tree logic CTL was given. The idea to develop team-based logics for hyperproperties was coined in [34], where TeamLTL was first introduced and shown incomparable to HyperLTL. The interest on logics for hyperproperties, so-called hyperlogics, was sparked by the introduction of HyperLTL and HyperCTL* [7]. Many temporal logics have since been extended with trace and path quantification to obtain various hyperlogics, e.g., to express asynchronous hyperproperties [22, 4], hyperproperties on finite traces [21], probabilistic hyperproperties [1], or timed hyperproperties [28]. Model checking HyperLTL and the strictly more expressive HyperQPTL is decidable, though k-EXPSPACE-complete, where k is the number of quantifier alternations in the formula [19, 41]. Model checking HyperQPTL+, on the other hand, is undecidable [16]. The expressivity of HyperLTL, HyperCTL*, and HyperQPTL has been compared to first-order and second-order hyperlogics resulting in a hierarchy of hyperlogics [9]. Beyond model checking and expressivity questions, especially HyperLTL has been studied extensively. This includes its satisfiability [15, 37], runtime monitoring [18, 2] and enforcement problems [10], as well as synthesis [17].

2 Basics of TeamLTL

Let us start by recalling the syntax of LTL from the literature. Fix a set AP of atomic propositions. The set of formulae of LTL (over AP) is generated by the following grammar:

φ ::= p | ¬p | φ ∨ φ | φ ∧ φ | φOφ | φUφ | φWφ, where p ∈ AP.

We adopt, as is common in studies on team logics, the convention that formulae are given in negation normal form. The logical constants ⊤, ⊥ and connectives →, ↔ are defined as
usual (e.g., $\bot := p \land \neg p$ and $\top := p \lor \neg p$), and
$\Diamond \varphi := \exists t \forall u \varphi$ and $\Box \varphi := \varphi W \bot$.

A trace $t$ over $AP$ is an infinite sequence from $(2^{2AP})^\omega$. For a natural number $i \in \mathbb{N}$, we
denote by $[t][i]$ the $i$th element of $t$ and by $[t][i, \infty]$ the postfix $([t][j])_{j \geq i}$ of $t$. The satisfaction
relation $(t, i) \models \varphi$, for LTL formulae $\varphi$, is defined as usual, see e.g., [39]. We use $[\varphi]_{[t][i]} \in \{0, 1\}$
to denote the truth value of $\varphi$ on $(t, i)$. A (temporal) team is a pair $(T, i)$ consisting of a set of traces $T \subseteq (2^{2AP})^\omega$ and a natural number $i \in \mathbb{N}$ representing the time step. We write $T[i]$ and $T[i, \infty]$ to denote the sets $\{[t][i] | t \in T\}$ and $\{[t][i, \infty] | t \in T\}$, respectively.

Let us next introduce the logic LTL interpreted with team semantics (denoted TeamLTL).

TeamLTL was first studied in [34], where it was called LTL with synchronous team semantics. The
satisfaction relation $(T, i) \models \varphi$ for TeamLTL is defined as follows:

$(T, i) \models p$ if and only if $\forall t \in T : p \in [t][i]$
$(T, i) \models \neg p$ if and only if $\forall t \in T : p \notin [t][i]$
$(T, i) \models \varphi \land \psi$ if $(T, i) \models \varphi$ and $(T, i) \models \psi$
$(T, i) \models \Box \varphi$ if $(T, i + 1) \models \varphi$
$(T, i) \models \varphi \lor \psi$ if $(T, i) \models \varphi$ or $(T, i) \models \psi$
$(T, i) \models p W \psi$ if $\exists k \geq i$ such that $(T, k) \models \psi$ and $\forall m : i \leq m < k \Rightarrow (T, m) \models \varphi$

Note that $(T, i) \models \bot$ if $T = \emptyset$. Two formulae $\varphi$ and $\psi$ are equivalent (written $\varphi \equiv \psi$), if
the equivalence $(T, i) \models \varphi$ if and only if $(T, i) \models \varphi$ holds for every $(T, i)$. We say that a logic $L_2$ is at
least as expressive as a logic $L_1$ (written $L_1 \leq L_2$) if for every $L_1$-formula $\varphi$, there exists an
$L_2$-formula $\psi$ such that $\varphi \equiv \psi$. We write $L_1 \equiv L_2$ if both $L_1 \leq L_2$ and $L_2 \leq L_1$ hold. The
following are important semantic properties of formulae from the team semantics literature:

(Downward closure) If $(T, i) \models \varphi$ and $S \subseteq T$, then $(S, i) \models \varphi$.
(Empty team property) $(\emptyset, i) \models \varphi$.
(Flatness) $(T, i) \models \varphi$ if $(\{t\}, i) \models \varphi$ for all $t \in T$.
(Singleton equivalence) $(\{t\}, i) \models \varphi$ if $(t, i) \models \varphi$.

A logic has one of the above properties if every formula of the logic has the property. TeamLTL
satisfies downward closure, singleton equivalence, and the empty team property [34]. However, it
does not satisfy flatness; for instance, the formula $\Diamond p$ is not flat.

The power of team semantics comes with the ability to enrich logics with novel atomic
statements describing properties of teams. We thereby easily get a hierarchy of team
logics of different expressiveness. The most prominent examples of such atoms are
dependence atoms $\text{dep}(\varphi_1, \ldots, \varphi_n, \psi)$ and inclusion atoms $\varphi_1, \ldots, \varphi_n \subseteq \psi_1, \ldots, \psi_n$, with
$\varphi_1, \ldots, \varphi_n, \psi_1, \ldots, \psi_n$ being LTL-formulæ. The dependence atom states that the truth
value of $\psi$ is functionally determined by that of $\varphi_1, \ldots, \varphi_n$. The inclusion atom states that each
value combination of $\varphi_1, \ldots, \varphi_n$ must also occur as a value combination for $\psi_1, \ldots, \psi_n$.

Their formal semantics is defined as:

$(T, i) \models \text{dep}(\varphi_1, \ldots, \varphi_n, \psi)$ if $\forall t, t' \in T : \bigwedge_{1 \leq j \leq n} [\varphi_j]_{[t][i]} = [\varphi_j]_{[t'][i]} \Rightarrow [\psi]_{[t][i]} = [\psi]_{[t'][i]}

(T, i) \models \varphi_1, \ldots, \varphi_n \subseteq \psi_1, \ldots, \psi_n$ if $\forall t \in T \exists t' \in T : \bigwedge_{1 \leq j \leq n} [\varphi_j]_{[t][i]} = [\psi_j]_{[t'][i]}$
As an example, let $o_1, \ldots, o_n$ be some observable outputs and $s$ be a secret. The atom $(o_1, \ldots, o_n, s) \subseteq (o_1, \ldots, o_n, \neg s)$ expresses a form of non-inference by stating that an observer cannot infer the current value of the secret from the outputs. We also consider other connectives known in the team semantics literature: Boolean disjunction $\lor$, Boolean negation $\neg$, and universal subteam quantifiers $A$ and $\hat{A}$, with their semantics defined as:

\[
\begin{align*}
(T, i) &= T 
\end{align*}
\]

If $A$ is a collection of atoms and connectives, we let $\text{TeamLTL}(A)$ denote the extension of TeamLTL with the atoms and connectives in $A$. For any atom or connective $c$, we write simply $\text{TeamLTL}(A, c)$ instead of TeamLTL($A \cup \{c\}$).

TeamLTL($\sim$) is a very expressive logic; all of the above connectives and atoms, as well as many others, have been shown to be definable in TeamLTL($\sim$) [25, 36]. To systematically explore less expressive variants of TeamLTL, we introduce two representative logics of different expressiveness, namely TeamLTL($\lor, \hat{A}$) and TeamLTL($\lor, \sim \bot, \hat{A}$). The expression $\sim \bot$ can be used to enforce non-emptiness of a team. What makes these logics good representatives is their semantic property that they can express a general class of Boolean relations. Let $B$ be a set of $n$-ary Boolean relations. We define the property $[\varphi_1, \ldots, \varphi_n]_B$ for an $n$-tuple $(\varphi_1, \ldots, \varphi_n)$ of LTL-formulae:

\[
(T, i) = [\varphi_1, \ldots, \varphi_n]_B \iff [(\varphi_1)_{(t,i)}, \ldots, (\varphi_n)_{(t,i)}] \mid t \in T \in B.
\]

The logic TeamLTL($\lor, \sim \bot, \hat{A}$) is expressively complete with respect to all $[\varphi_1, \ldots, \varphi_n]_B$. That is, for every set of Boolean relations $B$ and LTL-formulae $\varphi_1, \ldots, \varphi_n$, the property $[\varphi_1, \ldots, \varphi_n]_B$ is expressible in TeamLTL($\lor, \sim \bot, \hat{A}$). Furthermore, TeamLTL($\lor, \hat{A}$) can express all downward closed $(S_1 \in B$ and $S_2 \subseteq S_1$ imply $S_2 \in B$) $B$. These results are reformulated and proved using so-called generalised atoms in Appendix A. Note that, e.g., $k$-ary inclusion and dependence atoms can be defined using suitable Boolean relations $B$. Indeed it follows that, from the expressivity point-of-view, TeamLTL($\lor, \hat{A}$) and TeamLTL($\lor, \sim \bot, \hat{A}$) subsume all extensions of TeamLTL with downward closed (resp. all) atomic notions of dependence, i.e., atoms which state some sort of functional (in)dependence, like the dependence atom (which is downward closed) or the inclusion atom (which is not).

## 3 Undecidable Extensions of TeamLTL

In [36], Lück established that the model checking problem for TeamLTL($\sim$) is highly undecidable. The proof heavily utilises the interplay between Boolean negation $\sim$ and disjunction $\lor$; it was left as an open problem whether some sensible restrictions on the use of the Boolean negation would lead toward discovering decidable logics. We show that, on the contrary, the decidability bounds are much tighter. Already TeamLTL($\subseteq, \lor$) (which is subsumed by TeamLTL($\lor, \sim \bot, \hat{A}$)) is undecidable, and already very restricted access to $\sim$ (namely, a single use of the $A$ quantifier) leads to high undecidability.

We define the model checking problem based on Kripke structures $K = (W, R, \eta, w_0)$, where $W$ is a finite set of states, $R \subseteq W^2$ the transition relation, $\eta: W \to 2^{\mathbb{A}_P}$ a labelling function, and $w_0 \in W$ an initial state of $W$. A path $\sigma$ through $K$ is an infinite sequence $\sigma \in W^\omega$ such that $\sigma[0] = w_0$ and $(\sigma[i], \sigma[i + 1]) \in R$ for every $i \geq 0$. The trace of $\sigma$ is
defined as \( t(\sigma) := \eta(\sigma[0])\eta(\sigma[1]) \cdots \in (2^{AP})^\omega \). A Kripke structure \( K \) induces a set of traces \( \text{Traces}(K) = \{ t(\sigma) \mid \sigma \text{ is a path through } K \} \).

**Definition 1.** The model checking problem of a logic \( \mathcal{L} \) is the following decision problem: Given a formula \( \varphi \in \mathcal{L} \) and a Kripke structure \( K \) over AP, determine whether \( (\text{Traces}(K),0) \models \varphi \).

Our undecidability results are obtained by reductions from non-deterministic 3-counter machines. A non-deterministic 3-counter machine \( M \) consists of a list \( I \) of \( n \) instructions that manipulate three counters \( C_l, C_m, \) and \( C_r \). All instructions are of the following forms:

\[
\begin{align*}
C_a^+ \text{ goto } \{ j_1, j_2 \}, & \quad C_a^- \text{ goto } \{ j_1, j_2 \}, & \quad \text{if } C_a = 0 \text{ goto } j_1 \text{ else goto } j_2, \\
\end{align*}
\]

where \( a \in \{ l, m, r \}, 0 \leq j_1, j_2 < n \). A configuration is a tuple \((i, j, k, t)\), where \( 0 \leq i < n \) is the next instruction to be executed, and \( j, k, t \in \mathbb{N} \) are the current values of the counters \( C_l, C_m, \) and \( C_r \). The execution of the instruction \( i \) \( C_a^+ \text{ goto } \{ j_1, j_2 \} \) \( (i; C_a^- \text{ goto } \{ j_1, j_2 \}, \text{resp.}) \) increments (decrements, resp.) the value of the counter \( C_a \) by 1. The next instruction is selected nondeterministically from the set \( \{ j_1, j_2 \} \). The instruction \( i \) \( C_a = 0 \text{ goto } j_1 \text{ else goto } j_2 \) checks whether the value of the counter \( C_a \) is currently 0 and proceeds to the next instruction accordingly. The consecutive relation \( \prec_c \) of configurations is defined as usual. The lossy consecutive relation \((i_1, i_2, i_3, i_4) \prec_c (j_1, j_2, j_3, j_4)\) of configurations holds if \( (i_1, i_2, i_3, i_4) \prec_c (j_1, j_2, j_3, j_4) \) holds for some \( i_2', i_3', i_4', j_2, j_3, j_4 \) with \( i_2, i_3, i_4 \geq i_2', i_3', i_4' \), \( j_2 \geq j_2, j_3 \geq j_3, \) and \( j_4 \geq j_4 \). A (lossy) computation is an infinite sequence of (lossy) consecutive configurations starting from the initial configuration \((0,0,0,0)\). A (lossy) computation is b-recurring if the instruction labelled \( b \) occurs infinitely often in it. Deciding whether a given non-deterministic 3-counter machine has a b-recurring (b-recurring lossy) computation for a given \( b \) is \( \Sigma^1_1 \)-complete (\( \Sigma^1_0 \)-complete, resp.) \([3, 44]\).

We reduce the existence of a b-recurring lossy computation of a given 3-counter machine \( M \) and an instruction label \( b \) to the model checking problem of TeamLTL(\( \subseteq, \varnothing \)). We also illustrate that with a single instance of A we can enforce non-lossy computation instead.

**Theorem 2.** Model checking for TeamLTL(\( \subseteq, \varnothing \)) is \( \Sigma^1_1 \)-hard.

**Proof.** Given a set \( I \) of instructions of a 3-counter machine \( M \), and an instruction label \( b \), we construct a TeamLTL(\( \subseteq, \varnothing \))-formula \( \varphi_{I,b} \) and a Kripke structure \( K_I \) such that

\[
(\text{Traces}(K_I),0) \models \varphi_{I,b} \iff M \text{ has a } b \text{-recurring computation. (1)}
\]

The \( \Sigma^1_1 \)-hardness then follows since our construction is clearly computable. The idea is the following: Put \( n := |I| \). A set \( T \) of traces using propositions \( \{ c_i, c_m, c_r, d, 0, \ldots, n-1 \} \) encodes the sequence \((c_j)_{j \in \mathbb{N}}\) of configurations, if for each \( j \in \mathbb{N} \) and \( c_j = (i, v_l, v_m, v_r) \)

\[
\begin{align*}
(1) & \quad t[j] \cap \{0, \ldots, n-1\} = \{i\}, \text{ for all } t \in T, \\
(2) & \quad |t[j, \infty] \cap c_s \in t[j], t \in T| = v_s, \text{ for each } s \in \{l, m, r\}.
\end{align*}
\]

Hence, we use \( T[j, \infty] \) to encode the configuration \( c_j \); the propositions \( 0, \ldots, n-1 \) are used to encode the next instruction, and \( c_l, c_m, c_r, d \) are used to encode the values of the counters. The proposition \( d \) is a dummy proposition used to separate traces with identical postfixes with respect to \( c_l, c_m, \) and \( c_r \). The Kripke structure \( K_I = (W, R, \eta, \nu_0) \) over the set of propositions \( \{ c_l, c_m, c_r, d, 0, \ldots, n-1 \} \) is defined such that every possible sequence of configurations of \( M \) starting from \((0,0,0,0)\) can be encoded by some team \((T,0)\), where \( T \subseteq \text{Traces}(K_I) \). A detailed construction of the formula \( \varphi_{I,b} \) and the Kripke structure \( K_I \) together with a detailed proof for the fact that (1) indeed holds can be found in Appendix B. ▶
We aim to identify fragments of the logics with similar expressivity to better understand the TeamLTL in the rest of the paper.

Σ by their indexed versions time step add an atomic proposition p formula also write φ is HyperQPTL HyperQPTL is uniform propositional quantifiers. We also study two syntactic fragments of ∀ p for traces and two for propositional quantification.

In this section, we define those quantification-based hyperlogics against which we compare TeamLTL(⊂, ∅) and TeamLTL(⊂, ∅, A) are \( \Sigma_1^0 \)-hard and \( \Sigma_1^1 \)-hard, resp.

4 Quantification-based Hyperlogics and Team Semantics

In this section, we define those quantification-based hyperlogics against which we compare TeamLTL in the rest of the paper. TeamLTL and HyperLTL are known to have orthogonal expressivity [34] but apart from that, nothing is known about the relationship between the different variants of TeamLTL and other temporal hyperlogics such as HyperQPTL [41, 9]. We aim to identify fragments of the logics with similar expressivity to better understand the relative expressivity of TeamLTL for the specification of hyperproperties.

HyperQPTL+ [16] is a temporal logic for hyperproperties. It subsumes HyperLTL and HyperQPTL, so we proceed to give a definition of HyperQPTL+ and define the latter logics as fragments. HyperQPTL+ extends LTL with explicit trace quantification and quantification of atomic propositions. As such, it also subsumes QPTL, which can express all \( \omega \)-regular properties. Fix an infinite set \( \mathcal{V} \) of trace variables. HyperQPTL+ has three types of quantifiers, one for traces and two for propositional quantification.

\[
\begin{align*}
\varphi &::= \forall \pi. \varphi \mid \exists \pi. \varphi \mid \mathbf{U} \varphi \mid \mathbf{F} \varphi \mid \forall p. \varphi \mid \exists p. \varphi \mid \psi \\
\psi &::= p_\pi \mid \neg p_\pi \mid \psi \lor \psi \mid \psi \land \psi \mid \mathbf{O} \psi \mid \mathbf{F} \psi \mid \mathbf{G} \psi
\end{align*}
\]

Here, \( p \in \text{AP} \), \( \pi \in \mathcal{V} \), and \( \forall \pi \) and \( \exists \pi \) stand for universal and existential trace quantifiers, \( \forall p \) and \( \exists p \) stand for (non-uniform) propositional quantifiers, and \( \mathbf{U} \varphi \) and \( \mathbf{F} \varphi \) stand for uniform propositional quantifiers. We also study two syntactic fragments of HyperQPTL+. HyperQPTL is HyperQPTL+ without non-uniform propositional quantifiers, and HyperLTL is HyperQPTL+ without any propositional quantifiers. In the context of HyperQPTL, we also write \( \forall p \) and \( \exists p \) instead of \( \mathbf{U} p \) and \( \mathbf{F} p \). For an LTL-formula \( \varphi \) and trace variable \( \pi \), we let \( \varphi_\pi \) denote the HyperLTL-formula obtained from \( \varphi \) by replacing all proposition symbols \( p \) by their indexed versions \( p_\pi \). We extend this convention to tuples of formulae as well.

The semantics of HyperQPTL+ is defined over a set \( T \) of traces. Intuitively, the atomic formula \( p_\pi \) asserts that \( p \) holds on trace \( \pi \). Uniform propositional quantifications \( \mathbf{U} p \) and \( \mathbf{F} p \) add an atomic proposition \( p \) such that all traces agree on the valuation of \( p \) on any given time step \( i \), whereas non-uniform propositional quantifications \( \forall p \) and \( \exists p \) colour the traces...
in T in an arbitrary manner. Non-uniform propositional quantification thus implements true second-order quantification, whereas uniform propositional quantification can be interpreted as a quantification of a set of points in time.

A trace assignment is a function $\Pi : V \rightarrow T$ that maps each trace variable in V to some trace in T. A modified trace assignment $\Pi[\pi \mapsto t]$ is equal to $\Pi$ except that $\Pi[\pi \mapsto t](\pi) = t$. For any subset $A \subseteq AP$, we write $t \upharpoonright A$ for the projection of t on A (i.e., $(t \upharpoonright A)[i] := t[i] \cap A$ for all $i \in \mathbb{N}$). For any two trace assignments $\Pi$ and $\Pi'$, we write $\Pi =_A \Pi'$, if $(\Pi(\pi) \upharpoonright A) = (\Pi'(\pi) \upharpoonright A)$ for all $\pi \in V$. Similarly, $T =_A T'$ whenever $\{t \upharpoonright A \mid t \in T\} = \{t \upharpoonright A \mid t \in T\}$. For a sequence $s \in (2^{(p)})^\wedge$ over a single propositional variable $p$, we write $T[p \mapsto s]$ for the set of traces obtained from T by reinterpreting $p$ on all traces as in s while ensuring that $T[p \mapsto s] =_{AP\setminus(p)} T$. We use $\Pi[p \mapsto s]$ accordingly. The satisfaction relation $\Pi, i \models_T \varphi$ for HyperQPTL$^+$-formulae $\varphi$ is defined as follows:

\[
\begin{align*}
\Pi, i \models_T p\pi & \quad \text{if} \quad p \in \Pi(\pi)[i] \\
\Pi, i \models_T \varphi_1 \lor \varphi_2 & \quad \text{if} \quad \Pi, i \models_T \varphi_1 \quad \text{or} \quad \Pi, i \models_T \varphi_2 \\
\Pi, i \models_T \neg p\pi & \quad \text{if} \quad p \notin \Pi(\pi)[i] \\
\Pi, i \models_T \varphi_1 \land \varphi_2 & \quad \text{if} \quad \Pi, i \models_T \varphi_1 \quad \text{and} \quad \Pi, i \models_T \varphi_2 \\
\Pi, i \models_T p\pi & \quad \text{if} \quad p \in \Pi(\pi)[i] \\
\Pi, i \models_T \neg p\pi & \quad \text{if} \quad p \notin \Pi(\pi)[i] \\
\Pi, i \models_T \varphi_1 \lor \varphi_2 & \quad \text{if} \quad \Pi, i \models_T \varphi_1 \quad \text{or} \quad \Pi, i \models_T \varphi_2 \\
\Pi, i \models_T \varphi_1 \land \varphi_2 & \quad \text{if} \quad \Pi, i \models_T \varphi_1 \quad \text{and} \quad \Pi, i \models_T \varphi_2 \\
\Pi, i \models_T \exists \pi. \varphi & \quad \text{if} \quad \Pi[\pi \mapsto t], i \models_T \varphi \quad \text{for some} \quad t \in T \\
\Pi, i \models_T \forall \pi. \varphi & \quad \text{if} \quad \Pi[\pi \mapsto t], i \models_T \varphi \quad \text{for all} \quad t \in T \\
\Pi, i \models_T \exists p. \varphi & \quad \text{if} \quad \Pi[p \mapsto s], i \models_T[p \mapsto s] \varphi \quad \text{for some} \quad s \in (2^{(p)})^\wedge \\
\Pi, i \models_T \forall p. \varphi & \quad \text{if} \quad \Pi[p \mapsto s], i \models_T[p \mapsto s] \varphi \quad \text{for all} \quad s \in (2^{(p)})^\wedge \\
\Pi, i \models_T \exists \pi. \varphi & \quad \text{if} \quad \Pi'[\pi \mapsto t], i \models_T \varphi \quad \text{for some} \quad T' \subseteq (2^{AP})^\wedge \text{and} \quad \Pi' : V \rightarrow T' \quad \text{such that} \\
T =_{AP\setminus(p)} T' \quad \text{and} \quad \Pi =_{AP\setminus(p)} \Pi' \\
\Pi, i \models_T \forall p. \varphi & \quad \text{if} \quad \Pi'[\pi \mapsto t], i \models_T \varphi \quad \text{for all} \quad T' \subseteq (2^{AP})^\wedge \text{and} \quad \Pi' : V \rightarrow T' \quad \text{such that} \\
T =_{AP\setminus(p)} T' \quad \text{and} \quad \Pi =_{AP\setminus(p)} \Pi'
\end{align*}
\]

In the sequel, we describe fragments of HyperQPTL$^+$ by restricting the quantifier prefixes of formulae. We use $\exists_{\pi} / \forall_{\pi}$ to denote trace quantification, $\exists_{p} / \forall_{p}$ for uniform propositional quantification, and $\exists_{p} / \forall_{p}$ for non-uniform propositional quantification. We use $\exists (\forall, \text{resp.})$ if we do not need to distinguish between the different types of existential (universal, resp.) quantifiers. We write $Q$ to refer to both $\exists$ and $\forall$. For a logic $\mathcal{L}$ and a regular expression $e$, we write $e\mathcal{L}$ to denote the set of $\mathcal{L}$-formulae whose quantifier prefixes are generated by $e$. E.g., $\forall^\pi \exists^p$ HyperQPTL refers to HyperQPTL-formulae with quantifier prefix $\{\forall_p, \forall_{\pi}\}^* \{\exists_p, \exists_{\pi}\}^*$.

Next we relate the expressivity of extensions of TeamLTL to fragments of HyperQPTL$^+$. We show that TeamLTL$^+(\emptyset, \hat{A})$ and TeamLTL$^+(\sim, \hat{1}, A)$ can be translated to the prefix fragments $\exists_{p} Q^p_{\pi, \pi} \forall_{\pi}$ and $\exists_{p} Q^p_{p, \pi} \forall_{\pi}$ of HyperQPTL$^+$. The translations provide insight into the limits of the expressivity of different extensions of TeamLTL. In particular, they show that in order to simulate the generation of subteams with the $\forall$-operator in TeamLTL,
one existential second-order quantifier $\exists_p$ is sufficient. Meanwhile, the difference between downward closed team properties and general team properties manifests itself by a different need for trace quantifiers: for downward closed properties, a single $\forall_p$ quantifier is enough, whereas in the general case, a $\exists_p\forall_p$ quantifier alternation is needed.

As a prerequisite for the translation, we establish that evaluating TeamLTL$(\emptyset, \leadsto, \hat{A})$-formulae can only create countably many different teams. For a given team $(T,i)$ and TeamLTL$(\emptyset, \leadsto, \hat{A})$-formula $\varphi$, the verification of $(T,i) \models \varphi$ boils down to checking statements of the form $(S,j) \models \psi$, where $S \subseteq 2^T$ for some set $S_T$, $j \in \mathbb{N}$, and $\psi$ is an atomic formula, together with expressions of the form $S_1 = S_2 \cup S_3$, where $S_1, S_2, S_3 \in S_T$. The following lemma, proven in Appendix C, implies that the set $S_T$ can be fixed as a countable set that depends only on $T$.

- **Lemma 5.** For every set $T$ of traces over a countable AP, there exists a countable $S_T \subseteq 2^T$ such that, for every TeamLTL$(\emptyset, \leadsto, \hat{A})$-formula $\varphi$ and $i \in \mathbb{N}$, $(T,i) \models \varphi$ iff $(T,i) \models^* \varphi$, where the satisfaction relation $\models^*$ is defined the same way as $\models$ except that in the semantic clause for $\lor$ we require additionally that the two subteams $T_1, T_2 \in S_T$.

Using of the above lemma, we obtain translations from the most interesting extensions of TeamLTL to weak prefix fragments of HyperQPTL$^\dagger$: for details and proofs see Appendix C.

- **Theorem 6.** For every $\varphi \in$ TeamLTL$(\emptyset, \leadsto, \hat{A})$ there exists an equivalent HyperQPTL$^\dagger$-formula $\varphi^*$ in the $\exists_p\hat{Q}_p^\dagger\exists^*_p\forall_p$ fragment. If $\varphi \in$ TeamLTL$(\emptyset, \hat{A})$, $\varphi^*$ can be defined in the $\exists_p\hat{Q}_p^\dagger\forall_p$ fragment. The size of $\varphi^*$ is linear w.r.t. the size of $\varphi$.

## 5 Decidable fragments of TeamLTL

In this section, we further study the expressivity landscape between the frameworks of TeamLTL and HyperLTL. We utilise these connections to prove decidability of the model checking problem of certain variants of TeamLTL. We compare the expressivity of extensions of TeamLTL that satisfy certain semantic invariances to that of $\forall^\ast$HyperLTL and $\exists_p\forall_p$HyperQPTL. Thereby, we provide a partial answer to an open problem posed in [34] concerning the complexity of the model checking problem of TeamLTL and its extensions. The problem is known to be in PSPACE for the fragment of TeamLTL without $\lor$ [34]. However, for TeamLTL with $\lor$, no meaningful upper bounds for the problem was known before. The best previous upper bound could be obtained from TeamLTL$(\leadsto)$, for which the problem is highly undecidable [36]. The reason for this lack of results is that developing algorithms for team logics with $\lor$ turned out to be comparatively hard. The main source of difficulty is that the semantic definition of $\lor$ does not yield any reasonable compositional brute force algorithm: the verification of $(T,i) \models \varphi \lor \psi$ with $T$ generated by a finite Kripke structure proceeds by checking that $(T_1,i) \models \varphi$ and $(T_2,i) \models \psi$ for some $T_1 \cup T_2 = T$, but it can well be that $T_1$ and $T_2$ cannot be generated from any finite Kripke structure whatsoever.

The main results of this section are the decidability of the model checking problem of the $k$-coherent fragment of TeamLTL$(\leadsto)$ and the left-flat fragment of TeamLTL$(\emptyset, \hat{A})$. We obtain inclusions to EXPSPACE by translations to $\forall^\ast$HyperLTL and $\exists_p\forall_p$HyperQPTL.

### 5.1 The k-coherent fragment and $\forall^\ast$HyperLTL

The universal fragment of HyperLTL is one of the most studied fragments as it contains the set of safety hyperproperties expressible in HyperLTL [8]. In particular, formulae of the form...
∀π₁...∀πₖ.ψ state k-safety properties (if ψ is a safety LTL formula) [14], where non-satisfying trace sets contain bad prefixes of at most k traces. In general, ∀ₖHyperLTL formulae satisfy the following inherent invariance: ∅, i ⊨ₚ φ iff ∅, i ⊨ₚ φ, for all T ≤ T s.t. |T| ≤ k.
That is, a ∀ₖHyperLTL-formula φ is satisfied by a trace set T, iff it is satisfied by all subsets of T of size at most k. This property is called k-coherence in the team semantics literature [31].

The main result of this section is that all k-coherent properties expressible in TeamLTL(~) are expressible in ∀ₖHyperLTL. This implies that, with respect to trace properties, all logics between TeamLTL(总队) and TeamLTL(~) are equi-expressive to ∀ₖHyperLTL, i.e., LTL.

Definition 7. Let A be any collection of atoms and connectives introduced so far. A formula φ in TeamLTL(A) is said to be k-coherent (k ∈ N) if for every team (T, i),

(T, i) ⊨ φ iff (S, i) |= φ for every S ⊆ T with |S| ≤ k.

We will next show that, with respect to k-coherent properties, TeamLTL(~) is at most as expressive as ∀ₖHyperLTL. We define a translation from TeamLTL(~) to ∀ₖHyperLTL that preserves the satisfaction relation with respect to teams of bounded size. Given a finite set Φ of trace variables, the translation is defined as follows:

\[ p^\Phi := \bigwedge_{\pi \in \Phi} p_\pi \]
\[ (\neg p)^\Phi := \bigwedge_{\pi \in \Phi} \neg p_\pi \]
\[ (\sim \varphi)^\Phi := \neg \varphi^\Phi \]
\[ (O \varphi)^\Phi := O \varphi^\Phi \]
\[ (\varphi \land \psi)^\Phi := \varphi^\Phi \land \psi^\Phi \]
\[ (\varphi \lor \psi)^\Phi := \bigvee_{\Phi(\psi_\pi) = \Phi} \varphi^\Phi \land \psi^\Phi \]
\[ (\varphi U \psi)^\Phi := \varphi^\Phi U \psi^\Phi \]
\[ (\varphi W \psi)^\Phi := \varphi^\Phi W \psi^\Phi \]

where \( \neg \varphi^\Phi \) stands for the negation of \( \varphi^\Phi \) in negation normal form. The following lemma, from which the subsequent theorem follows, is proved by induction. See Appendix D for detailed proofs.

Lemma 8. Let φ be a formula of TeamLTL(~) and Φ = {π₁,...,πₖ} a finite set of trace variables. For any team (T, i) with |T| ≤ k, any set S ⊆ T of traces, and any assignment Π with Π[Φ] = T, we have that (T, i) ⊨ φ Π iff i ⊨ₚ Φ. Furthermore, if φ is downward closed and T ≠ ∅, then (T, i) ⊨ φ iff ∀ i ⊨ₚ ∀π₁...∀πₖ.φ.

Theorem 9. Every k-coherent property that is definable in TeamLTL(~) is also definable in ∀ₖHyperLTL.

Since model checking for ∀ₖHyperLTL is PSPACE-complete and its data complexity (model checking with a fixed formula) is NL-complete [19], and since the above translation from TeamLTL(~) to ∀ₖHyperLTL is exponential for any k, we get the following corollary:

Corollary 10. For any fixed k ∈ N, the model checking problem for TeamLTL(~), restricted to k-coherent properties, is in EXPSPACE, and in NL for data complexity.

Clearly (T, i) ⊨_A φ iff ∅, i ⊨ₚ ∀π.φ for any φ ∈ LTL, and hence we obtain the following:

Corollary 11. The restriction of TeamLTL(总队) to formulae of the form Aφ is expressively equivalent to ∀ₖHyperLTL.

While model checking for k-coherent properties is decidable, checking whether a given formula defines a k-coherent property is not, in general, decidable.

Theorem 12. Checking whether a TeamLTL(总队; φ)-formula is 1-coherent is undecidable.
Proof. The idea of the undecidability proof is as follows: Given any TeamLTL(⊆, ⋀)-formula ϕ, we can use a simple rewriting rule to obtain an LTL-formula ϕ* such that ϕ is not satisfiable (in the sense of TeamLTL) if and only if ϕ is 1-coherent and ϕ* is not satisfiable (in the LTL-sense). Now, since checking LTL-satisfiability can be done in PSPACE [45] and non-satisfiability for TeamLTL(⊆, ⋀) is Π10-hard by Corollary 4, it follows that checking 1-coherence is Π10-hard as well. For a detailed proof, see Appendix D.

The same holds for any extension of TeamLTL with an undecidable satisfiability or validity problem and whose formulae can be computably translated to TeamLTL while preserving satisfaction over singleton teams.

5.2 The left-flat fragment and HyperQPTL+

In this subsection, we show that formulae ϕ from the left-flat fragment of TeamLTL(∅, urrency operators) (defined below) can be translated to HyperQPTL formulae that are linear in the size of ϕ. The known model checking algorithm of HyperQPTL [41] then immediately yields a model checking algorithm for the left-flat fragment of TeamLTL(∅, urrency operators).

Definition 13 (The left-flat fragment). Let A be a collection of atoms and connectives. A TeamLTL(A)-formula belongs to the left-flat fragment if in each of its subformulae of the form ψUφ or ψWφ, ψ is a flat formula (as defined in Section 2).

Such defined fragment allows for arbitrary use of the ♦ operator, and therefore remains incomparable to HyperLTL [34]. For instance, ♦dep(a, b)∨♦dep(c, d) is a nontrivial formula in this fragment. It states that the set of traces can be partitioned into two parts, one where eventually a determines the value of b, and another where eventually c determines the value of d. The property is not expressible in HyperLTL, because HyperLTL cannot state the property “there is a point in time at which p holds on all (or infinitely many) traces” [5].

It follows from Theorem 12 that checking whether a TeamLTL(⊆, ⋀)-formula belongs to the left-flat fragment is undecidable (as flatness equals 1-coherency). Nevertheless, a decidable syntax for left-flat formulae can be obtained by using the operator ♦. Formulae ♦ψ are always flat and equivalent to ψ if ψ is flat. Therefore, in Definition 13, instead of imposing the semantic condition of ψ being flat in subformulae ψUφ and ψWφ, we could require that the subformulae must be of the form (♦ψ)Uφ or (♦ψ)Wφ.

We now describe a translation from the left-flat fragment of TeamLTL(∅, urrency operators) to the 2★Σ★ϕ+ fragment of HyperQPTL. In this translation, we make use of the fact that satisfaction of flat formulae ϕ can be determined with the usual (single-traced) LTL semantics. In the evaluation of ϕ, it is thus sufficient to consider only finitely many subteams, whose temporal behaviour can be reflected by existentially quantified q-sequences. The quantified sequences refer to points in time, at which subformulae have to hold for a trace to belong to a team.

A left-flat TeamLTL(∅, urrency operators)-formula ϕ will be translated into a formula with existential propositional quantifiers followed by a single trace quantifier. The existential propositional quantifiers either indicate a point in time at which a subformula of ϕ_i is evaluated or resolve the decision for ⋀-choices. For subformulae, we use propositions rψ, and if the same subformula occurs multiple times, it is associated with different rψ. For the resolution of ⋀-choices we use propositions dψ. Additionally, r is a free proposition for the point in time at which ϕ is to be evaluated. The universal quantifier ∀π sorts each trace into one of the finitely many teams.
Let $\forall \pi. \hat{\psi}$ be the HyperLTL formula given by Theorem 9 for any flat formula $\psi$ (since a flat formula is $1$-coherent). We translate $\varphi$ inductively with respect to $r$:

\[
\begin{align*}
[p, r] &:= \Box (r_\pi \rightarrow p_\pi) \\
[\neg p, r] &:= \Box (r_\pi \rightarrow \neg p_\pi) \\
[\bigcirc \varphi, r] &:= \Box (r_\pi \leftrightarrow \bigcirc r_\pi^\psi) \land [\varphi, r^\varphi] \\
[\hat{A} \varphi, r] &:= \Box (r_\pi \rightarrow \hat{\varphi}) \\
[\varphi \land \psi, r] &:= [\varphi, r] \land [\psi, r] \\
[\varphi \lor \psi, r] &:= [\varphi, r] \lor [\psi, r] \\
[\varphi \lozenge \psi, r] &:= (d_\pi^\lozenge \varphi \rightarrow [\varphi, r]) \land (\neg d_\pi^\lozenge \varphi \rightarrow [\psi, r]) \\
[\varphi \mathcal{W} \psi, r] &:= \Box (r_\pi \rightarrow r_\pi^\psi \mathcal{W} (r_\pi^\psi \land \bigcirc \neg r_\pi^\psi))
\end{align*}
\]

Now, let $r^1, \ldots, r^n$ be the free propositions occurring in $[\varphi, r]$ and $\pi$ the free trace variable. Define the following $\exists \pi^\varphi$ HyperQPTL formula: $\exists \pi^1 \ldots \exists \pi^n. \forall \pi. r_\pi \land \bigcirc \neg r_\pi \land [\varphi, r]$. Correctness of the translation can be argued intuitively as follows. The left-flat formula $\varphi$ can be evaluated independently from the other traces in a team. Therefore, the operators $\mathcal{U}$ and $\mathcal{W}$, whose right-hand sides argue only about a single point in time, can only generate finitely many teams. Thus, there are only finitely many points of synchronization, all of which are quantified existentially. Every trace fits into one of the teams described by the quantified propositional variables. We verify that the translation is indeed correct in Appendix D. As the construction in Theorem 9 yields a formula $\hat{\varphi}$ whose size is linear in the original formula $\varphi$, the translation is obviously linear. We therefore state the following theorem.

> **Theorem 14.** For every formula $\varphi$ from the left-flat fragment of TeamLTL$(\otimes, \hat{A})$, we can compute an equivalent $\exists \pi^\varphi$ HyperQPTL formula of size linear in the size of $\varphi$.

Recall that the model checking problem of HyperLTL formulae with one quantifier alternation is EXPSPACE-complete [19] in the size of the formula, and PSPACE-complete in the size of the Kripke structure [19]. These results directly transfer to HyperQPTL [41] (in which HyperQPTL was called HyperLTL with extended quantification instead): for model checking a HyperQPTL formula, the Kripke structure can be extended by two states generating all possible $q$-sequences. Since the translation from TeamLTL$(\otimes, \hat{A})$ to HyperQPTL yields a formula in the $\exists \pi^\psi$ fragment with a single quantifier alternation and preserves the size of the formula, we obtain the following theorem.

> **Theorem 15.** The model checking problem for left-flat TeamLTL$(\otimes, \hat{A})$-formulae is in EXPSPACE, and in PSPACE for data complexity.

## Conclusion

We studied TeamLTL under the synchronous semantics. TeamLTL is a powerful but not yet well-studied logic that can express hyperproperties without explicit quantification over traces or propositions. As such, properties which need various different quantifiers in traditional (quantification-based) hyperlogics become expressible in a concise fashion. One of the main advantages of TeamLTL is the ability to equip it with a range of atomic statements and connectives to obtain logics of varying expressivity and complexity.

We systematically studied TeamLTL with respect to two of the main questions related to logics: the decision boundary of its model checking problem and its expressivity compared to
other logics for hyperproperties. We related the expressivity of TeamLTL to the hyperlogics HyperLTL, HyperQPTL, and HyperQPTL+1, which are obtained by extending the traditional temporal logics LTL and QPTL with trace quantifiers. We discovered that the logics TeamLTL(∅, A) and TeamLTL(∅, ∼⊥, A) are expressively complete with respect to all downward closed, and all atomic notions of dependence, respectively. We were able to show that TeamLTL(∅, ∼⊥, A) can be expressed in a fragment of HyperQPTL+1. Furthermore, for k-coherent properties, TeamLTL(·) is subsumed by ∀∗HyperLTL. Finally, the left-flat fragment of TeamLTL(∅, A) can be translated to HyperQPTL. The last two results induce efficient model checking algorithms for the respective logics. In addition, we showed that model checking of TeamLTL(⊆, ∅) is already undecidable, and that the additional use of the A quantifier makes the problem highly undecidable.

We conclude with some open problems and directions for future work: What is the complexity of model checking for TeamLTL (with the disjunction ∨ but without additional atoms and connectives)? Is it decidable, and is there a translation to HyperQPTL? An interesting avenue for future work is also to explore team semantics of more expressive logics than LTL such as linear time µ-calculus, or branching time logics such as CTL and the full modal µ-calculus.

References


9. Norine Coenen, Bernd Finkbeiner, Christopher Hahn, and Jana Hofmann. The hierarchy of hyperlogics. In 34th Annual ACM/IEEE Symposium on Logic in Computer Science,


LTL with Team Semantics: Expressivity and Complexity


Section 3 established that already very weak fragments of TeamLTL(∼) have highly undecidable model checking. One way to obtain expressive, but computationally well behaved, team-logics for hyperproperties is to carefully introduce novel atoms stating those atomic properties that are of the most interest, and then seeing whether decidable logics, or fragments, with those atoms can be uncovered. In Section 2 we already saw a few examples of important atomic statements; namely dependence and inclusion atoms. More generally, arbitrary properties of teams induce generalised atoms. These atoms were first introduced,

**Definition 16 (Generalised atoms for LTL).** An n-ary generalised atom is an n-ary operator \( \#_G(\varphi_1, \ldots, \varphi_n) \), with an associated nonempty set \( G \) of n-ary relations over the Boolean domain \( \{0, 1\} \), that applies only to LTL-formulae \( \varphi_1, \ldots, \varphi_n \). Its team semantics is defined as:

\[
(T, i) \models \#_G(\varphi_1, \ldots, \varphi_n) \iff \{([\varphi_1]_{(t,i)}, \ldots, [\varphi_n]_{(t,i)}) \mid t \in T\} \in G.
\]

E.g., dependence atoms of type \( \text{dep}(\varphi, \psi) \) can be expressed as generalised atoms via the second-order truth table \( G := \{A \subseteq \{0, 1\}^2 \mid \text{if } (a, b_1), (a, b_2) \in A \text{ then } b_1 = b_2\} \). Thus, when considered as a generalised atom, the dependence atom is, in fact, a collection of generalised atoms; one for each arity. Note that \( \hat{A} \) can be interpret as a downward closed generalised atom where \( G := \{\emptyset, \{1\}\} \), when applied to TeamLTL-formulae. From now on, we treat \( \hat{A} \) as a generalised atom. Often this distinction has no effect, for example, TeamLTL(\( \emptyset, \hat{A} \))-formula \( \hat{A}\varphi \) is equivalent with the TeamLTL(\( \hat{A} \))-formula \( \hat{A}\varphi' \), where \( \varphi' \) is obtained from \( \varphi \) by replacing all \( \emptyset \) by \( \lor \), and simply eliminating the symbols \( \hat{A} \) from the formula. We denote by \( A_{\text{all}} \) (\( A_{\text{dc}} \), resp.) the collection of all (all downward closed, resp.) generalised atoms.

It is straightforward to verify (by induction) that for any collection \( A \) of downward closed atoms and connectives (a connective is downward closed, if it preserves downward closure), the logic TeamLTL(\( A \)) is downward closed as well. For instance, TeamLTL(\( \text{dep}, \emptyset, \hat{A} \)) is downward closed.

Propositional and modal logic with \( \emptyset \) are known to be expressively complete with respect to nonempty downward closed properties (that are closed under so-called team bisimulations)[27, 48]. The next proposition establishes that, in the LTL setting, TeamLTL(\( \emptyset, \hat{A} \)) is very expressive among the downward closed logics; all downward closed atoms can be expressed in the logic. The translation given in the proposition is inspired by an analogous one given in [48] for propositional team logics.

**Proposition 17.** For any n-ary generalised atom \( \#_G \) and LTL-formulae \( \varphi_1, \ldots, \varphi_n \), we have that

\[
\#_G(\varphi_1, \ldots, \varphi_n) \equiv \bigwedge_{R \in G} (\hat{A}(\varphi_1^{b_1} \land \cdots \land \varphi_n^{b_n}) \land \neg \bot)
\]

where \( \varphi_i^0 := \varphi_i \) and \( \varphi_i^1 := \neg \varphi_i \) in negation normal form. If \( \#_G \) is downward closed, the above translation can be simplified to

\[
\#_G(\varphi_1, \ldots, \varphi_n) \equiv \bigwedge_{R \in G} \hat{A}(\varphi_1^{b_1} \land \cdots \land \varphi_n^{b_n}).
\]

Consequently, TeamLTL(\( A_{\text{dc}}, \emptyset \)) \( \equiv \) TeamLTL(\( \emptyset, \hat{A} \)) and TeamLTL(\( A_{\text{all}}, \emptyset \)) \( \equiv \) TeamLTL(\( \emptyset, \hat{A}, \neg \bot \)) \( \leq \) TeamLTL(\( \neg \)), where \( \neg \bot \) is read as a generalised atom stating that the team is non-empty. The elimination of each atom yields a doubly exponential disjunction over linear sized formulae.

**Proof.** For any \( \vec{b} = (b_1, \ldots, b_n) \in R \in G \), put

\[
\vec{\varphi}^{\vec{b}} = \varphi_1^{b_1} \land \cdots \land \varphi_n^{b_n}.
\]
Put also
\[ [\varphi]_{(t,i)} = ([\varphi_1]_{(t,i)}, \ldots, [\varphi_n]_{(t,i)}) \]
for any trace \( t \). Now, by the empty team property, we have
\[ (S,i) \models \hat{A}\varphi^b \iff S = \emptyset \text{ or } \forall t \in S : [\varphi]_{(t,i)} = \hat{b} \]
\[ \iff \{ [\varphi]_{(t,i)} \mid t \in S \} \subseteq \{ \hat{b} \} . \]

Thus,
\[ (T,i) \models \bigvee_{\hat{b} \in R} \hat{A}\varphi^b \]
\[ \iff \forall \hat{b} \in R, \exists T_{\hat{b}} \text{ s.t. } T = \bigcup_{\hat{b} \in R} T_{\hat{b}} \text{ and } \{ [\varphi]_{(t,i)} \mid t \in T_{\hat{b}} \} \subseteq \{ \hat{b} \} \]
\[ \iff \{ [\varphi]_{(t,i)} \mid t \in T \} \subseteq R . \]

Since \((S,i) \models \sim \bot \iff S \neq \emptyset\), we have similarly that
\[ (S,i) \models \hat{A}\varphi^b \land \sim \bot \iff \{ [\varphi]_{(t,i)} \mid t \in S \} = \{ \hat{b} \} , \]
and thus
\[ (T,i) \models \bigvee_{\hat{b} \in R} (\hat{A}\varphi^b \land \sim \bot) \iff \{ [\varphi]_{(t,i)} \mid t \in T \} = R . \]

Finally, if \( \#_G \) is downward closed, then
\[ (T,i) \models \bigvee_{R \in G, \hat{b} \in R} (\hat{A}\varphi^b) \]
\[ \iff (T,i) \models \bigvee_{\hat{b} \in R} (\hat{A}\varphi^b) \text{ for some } R \in G \]
\[ \iff \{ [\varphi]_{(t,i)} \mid t \in T \} \subseteq R \text{ for some } R \in G \]
\[ \iff \{ [\varphi]_{(t,i)} \mid t \in T \} \in G \quad (\because \#_G \text{ is downward closed}) \]
\[ \iff (T,i) \models \#_G(\varphi_1, \ldots, \varphi_n) . \]

Similarly, if no additional condition is assumed for \( \#_G \), then \((T,i) \models \bigvee_{R \in G, \hat{b} \in R} (\hat{A}\varphi^b \land \sim \bot) \iff (T,i) \models \#_G(\varphi_1, \ldots, \varphi_n) . \)

The operator \( \hat{A} \) used in the above translation is essential for the result. We now illustrate with an example that the logic TeamLTL(\( \otimes \)) without the operator \( \hat{A} \) is strictly less expressive than TeamLTL(\( \otimes, \hat{A} \)) or TeamLTL(A_{dc}, \( \otimes \)).

\section*{Proposition 18.} The formula \( \hat{A} \otimes p \) is not expressible in TeamLTL(\( \otimes \)). Thus TeamLTL(\( \otimes \)) < TeamLTL(\( \otimes, \hat{A} \)).

\textbf{Proof.} Define trace sets
\[ T_{n^+} := \{ \{ \}^n \{ p \} \}^\omega \mid n \in \mathbb{N} \}, \quad T_{\emptyset} := \{ \{ \}^\omega \} . \]
and \( T_{\mathbb{N}} := T_{\mathbb{N}^*} \cup T_0 \). Clearly \((T_{\mathbb{N}^*}, 0) \models \hat{A} \Diamond p\) and \((T_{\mathbb{N}}, 0) \not\models \hat{A} \Diamond p\). We now show, by induction, that for every Team\(\text{LTL}(\varnothing)\)-formula \(\varphi, i \in \mathbb{N}\), and infinite subset \(T \subseteq T_{\mathbb{N}^+}\),
\[
(T, i) \models \varphi \iff (T \cup T_0, i) \models \varphi.
\]

Since Team\(\text{LTL}(\varnothing)\) is downward closed, it suffices to show the "\(\Rightarrow\)" direction. The base case that \(\varphi = p\) or \(\neg p\) follows from the observation that \((S, i) \models \neg p\) for any infinite \(S \subseteq T_{\mathbb{N}}\). We now only give details for the most interesting inductive cases.

If \((T, i) \models \varphi \lor \psi\), then there exist \(T_1, T_2 \subseteq T\) s.t. \(T_1 \cup T_2 = T\), \((T_1, i) \models \varphi\) and \((T_2, i) \models \psi\), where either \(T_1\) or \(T_2\) is infinite, as \(T\) is infinite. W.l.o.g. we may assume that \(T_1\) is infinite. Now by induction hypothesis, we have that
\[
(T_1 \cup T_0) \cup T_2 = T \cup T_0, \quad (T_1 \cup T_0, i) \models \varphi, \quad \text{and} \quad (T_2, i) \models \psi.
\]
Therefore, \((T \cup T_0, i) \models \varphi \lor \psi\).

If \((T, i) \models \varphi \land \psi\), then there exists \(k \geq i\) s.t. \((T, k) \models \psi\) and \((T, j) \models \varphi\) whenever \(i \leq j < k\). By induction hypothesis, we have that \((T \cup T_0, k) \models \psi\) and \((T \cup T_0, j) \models \varphi\) whenever \(i \leq j < k\). Hence, we conclude that \((T \cup T_0, i) \models \varphi \land \psi\).

It is a fascinating open problem whether variants of the infamous Kamp’s theorem can be developed for Team\(\text{LTL}(\varnothing, \hat{A})\) and Team\(\text{LTL}(\sim)\). In the modal team semantics setting variants of the famous van Benthem-Rosen characterisation theorem have been shown for the modal logic analogues ML(\(\varnothing\)) [27] and ML(\(\sim\)) [32] of Team\(\text{LTL}(\varnothing, \hat{A})\) and Team\(\text{LTL}(\sim)\), respectively.

### B Missing Proofs of Section 3

\(\triangleright\) Claim 19. The claim (1) on page 7 holds.

**Proof.** We start by defining the formula \(\varphi_{I, b}\) that enforces that the configurations encoded by \(T[i, \infty]\), \(i \in \mathbb{N}\), encode an accepting computation of the counter machine.

Define \(\varphi_{I, b} := (\theta_{\text{comp}} \land \theta_{b-\text{rec}}) \forall L \in T\), where \(\forall L\) is a shorthand for the following condition:

\[
(T, i) \models \varphi \land \forall L \psi \iff \exists T_1, T_2 \text{ s.t. } T_1 \neq \emptyset, T_1 \cup T_2 = T, (T_1, i) \models \varphi \text{ and } (T_2, i) \models \psi.
\]

The disjunction \(\forall L\) can be defined using \(\sqsubseteq, \lor\), and a built-in trace \(\{p\}^\omega\), where \(p\) is a fresh proposition (see e.g., [26, Lemma 3.4]). The formula \(\theta_{b-\text{rec}} := \Box \Diamond b\) describes the \(b\)-recurrence condition of the computation. The other formula \(\theta_{\text{comp}}\), which we define below in steps, states that the encoded computation is a legal one.

First, define
\[
\text{singleton} := \Box \bigwedge_{a \in \text{AP}} (a \otimes \neg a), \quad c_s\text{-decrease} := c_s \lor (\neg c_s \land \Box \neg c_s), \quad \text{for } s \in \{l, m, r\}.
\]

The intuitive idea behind the above formulae are as follows: A team satisfying the formula singleton contains at most a single trace with respect to the propositions in \(\text{AP}\). If a team \((T, i)\) satisfies \(c_s\text{-decrease}\), then the number of traces in \(T[i + 1, \infty]\) satisfying \(c_s\) is less or equal to the number of traces in \(T[i, \infty]\) satisfying \(c_s\). In our encoding of counters this would mean that the value of the counter \(c\) in the configuration \(c_{i+1}\) is less or equal to its value in the configuration \(c_i\). Thus \(c_s\text{-decrease}\) will be handy below for encoding lossy computation.

Next, for each instruction label \(i\), we define a formula \(\theta_i\) describing the result of the execution of the instruction:
For the instruction $i: C^+_i \text{ goto } \{j, j'\}$, define
\[ \theta_i := \bigcirc(j \sqcup j') \land ((\text{singleton } \land \neg C_1 \land \bigcirc C_1) \lor C_1\text{-decrease}) \land C_r\text{-decrease} \land C_m\text{-decrease}. \]

For the instruction $i: C^-_i \text{ goto } \{j, j'\}$, define
\[ \theta_i := \bigcirc(j \sqcup j') \land ((C_1 \land \bigcirc \neg C_1) \lor \land C_1\text{-decrease}) \land C_r\text{-decrease} \land C_m\text{-decrease}. \]

For the instructions $i: C^+_s \text{ goto } \{j, j'\}$ and $i: C^-_s \text{ goto } \{j, j'\}$ with $s \in \{m, r\}$, the formulae $\theta_i$ are defined analogously with the indices $l, m$, and $r$ permuted.

For the instruction $i$: if $C_s = 0 \text{ goto } j$, else $\text{ goto } j'$, define
\[ \theta_i := (\bigcirc (\neg C_s \land j) \lor (T \subseteq C_s \land \bigcirc j')) \land C_r\text{-decrease} \land C_m\text{-decrease} \land C_c\text{-decrease}. \]

Finally, define $\theta_{\text{comp}} := \square \bigcirc_{i < n} (i \land \theta_i)$. We next describe the intuition of the above formulae. The left-most conjunct of $\theta_i$ for $i: C^+_i \text{ goto } \{j, j'\}$ expresses that after executing the instruction $i$, the label of the next instruction is either $j$ or $j'$. The third and the fourth conjunct express that the values of counters $C_r$ and $C_m$ will not increase, but might decrease. The second conjunct expresses that the value of the counter $C_r$ might increase by one, stay the same, or decrease. The meaning of $\theta_i$ for $i: C^-_i \text{ goto } \{j, j'\}$ is similar. Finally, the formula $\theta_i$ for $i$: if $C_s = 0 \text{ goto } j$, else $\text{ goto } j'$ expresses that, in the lossy execution of $i$, a) the values of the counters $C_l, C_m,$ and $C_r$ might decrease, but cannot increase, b) the next instruction is either $j$ or $j'$, c) if the next instruction is $j$ then the value of the counter $C_s$, after the lossy execution, is 0, and d) if the next instruction is $j'$ then the value of the counter $C_s$, before the lossy execution, was not 0.

Next, we define the Kripke structure $K_T = (W, R, \eta, w_0)$ over the set of propositions \{ $c_i, c_m, c_r, d, 0, \ldots, n - 1$ \}. The structure is defined such that every possible sequence of $M$ starting from $(0, 0, 0, 0)$ can be encoded by some team $(T, 0)$, where $T \subseteq \text{Traces}(K_T)$. Define $W := \{(i, j, k, t, l) \mid 0 \leq i < n \text{ and } j, k, t, l \in \{0, 1\}\}$, $w_0 := (0, 0, 0, 0, 0)$, $R := W \times W$, and $\eta$ as the valuation such that $\eta((i, j, k, t, l)) \cap \{0, \ldots, n - 1\} = i$.

Assume first that $M$ has a $b$-recurrence. Let $(\vec{c}_j)_{j \in \mathbb{N}}$ be the related sequence of configurations of $M$. Let $T \subseteq \text{Traces}(K_T)$ be a set of traces that encodes $(\vec{c}_j)_{j \in \mathbb{N}}$. Define $C_l, c_m, c_r, d, 0, \ldots, n - 1$ in the way described above, and such that, for every $j \in \mathbb{N}$,

- $c_i \in \eta((i, j, k, t, l))$ if $j = 1$,
- $c_m \in \eta((i, j, k, t, l))$ if $k = 1$,
- $c_r \in \eta((i, j, k, t, l))$ if $t = 1$,
- $d \in \eta((i, j, k, t, l))$ if $l = 1$.

The aforementioned condition makes sure that the traces in $T$ that encode the incrementation of counter values do not change erratically. Clearly, such a $T$ always exists, given that $(\vec{c}_j)_{j \in \mathbb{N}}$ encodes a lossy computation. Furthermore, since $T$ encodes $(\vec{c}_j)_{j \in \mathbb{N}}$ and the related $b$-recurrent lossy computation follows the instructions in $I$, we have that $(T, 0) \models \theta_{\text{comp}} \land \theta_{b\text{-rec}}$. Finally, as $\emptyset \neq T \subseteq \text{Traces}(K_T)$, $(\text{Traces}(K_T), 0) \models (\theta_{\text{comp}} \land \theta_{b\text{-rec}}) \lor \land T$ follows.

Assume then that $(\text{Traces}(K_T), 0) \models \varphi_{I, b}$. Hence there exists some nonempty subset $T$ of Traces$(_T)$ such that $(T, 0) \models \theta_{\text{comp}} \land \theta_{b\text{-rec}}$. It is now easy to construct a sequence $(\vec{c}_j)_{j \in \mathbb{N}}$ of configurations that encode a $b$-recurrent lossy computation for $M$; for each $j \in \mathbb{N}$, define $c_j = (i, v_t, v_m, v_r)$ such that $\bigcup T[j] \cap \{0, \ldots, n - 1\} = \{i\}$, and $|T[j, \infty]| \cap c_s \in t[j], t \in T| = v_s$, for each $s \in \{l, m, r\}$.

\textbf{Theorem 3.} Model checking for TeamLTL($\subseteq, \forall, A$) is \textbf{\textit{Sigma}}$^3$-hard. This holds already for the fragment with a single occurrence of $A$. ◀
Proof. The proof is analogous to that of Theorem 2. The following modifications are required to shift from lossy computation to non-lossy one. Firstly, we define a formula $\theta_{\text{diff}}$ that expresses that if two traces differ (i.e., $t \neq t'$) then all of their postfixes differ as well (i.e., $t[j, \infty] \neq t'[j, \infty]$, for each $j \in \mathbb{N}$):

$$
\theta_{\text{diff}} := A \Box ((c_l \ominus c_l) \land (c_m \ominus c_m) \land (c_r \ominus c_r) \land (d \ominus d)) \\
\otimes \Box \Box \Box ((\top \subseteq c_l \land \bot \subseteq c_l) \otimes (\top \subseteq c_m \land \bot \subseteq c_m) \\
\otimes (\top \subseteq c_r \land \bot \subseteq c_r) \otimes (\top \subseteq d \land \bot \subseteq d)).
$$

For $s \in \{l, m, r\}$, instead of using the formula $c_s$-decrease in Theorem 2, we make use of $c_s$-preserve defined as:

$$
c_s \text{-preserve} := (c_s \land \Box c_s) \lor (\neg c_s \land \Diamond c_s).
$$

Finally, we define

$$
\varphi_{I,L} := (\theta_{\text{diff}} \land \theta'_{\text{comp}} \land \theta_{b-\text{rec}}) \lor _L \top,
$$

where $\theta'_{\text{comp}} := \Box \square_{i<s}(i \land \theta'_i)$ and $\theta'_i$ for each instruction $i$ is defined as follows:

- For the instruction $i: C_i^+ \text{ goto } \{j, j'\}$, define
  $$
  \theta'_i := \Box (j \land j') \land ((\text{singleton} \land \neg c_l \land \Box c_l) \lor_L \Box c_l \text{-preserve}) \land \land c_i \text{-preserve} \land c_m \text{-preserve}.
  $$

- For the instruction $i: C_i^- \text{ goto } \{j, j'\}$, define
  $$
  \theta'_i := \Box (j \land j') \land ((\text{singleton} \land \neg c_l \land \Box c_l) \lor_L \Box c_l \text{-preserve}) \land \land c_i \text{-preserve} \land c_m \text{-preserve}.
  $$

- For $i: C_i^+ \text{ goto } \{j, j'\}$ and $i: C_i^- \text{ goto } \{j, j'\}$ with $s \in \{m, r\}$, the formulae $\theta'_i$ are defined analogously with the indices $l, m, r$ permuted.

- For the instruction $i: \text{ if } C_s = 0 \text{ goto } j, \text{ else goto } j'$, define
  $$
  \theta'_i := ((\neg c_s \land \Box j) \lor (\top \subseteq c_s \land \Box j')) \land \land c_i \text{-preserve} \land c_m \text{-preserve} \land c_r \text{-preserve}.
  $$

\[\blacktriangleleft\]

**Corollary 4.** The satisfiability problems for TeamLTL$(\subseteq, \Box)$ and TeamLTL$(\subseteq, \Box, A)$ are $\Sigma^0_1$-hard and $\Sigma^1_1$-hard, resp.

Proof. Given a finite Kripke structure $K = (W, R, \eta, w_0)$ over a finite set of propositions $\Delta P$, we introduce a fresh propositional variable $p_w$, for each $w \in W$, and let $K'$ be the unique extension of $K$ in which $p_w$ holds only in $w$. We then construct a formula $\theta_{K'}$ whose intention is to characterise the team $(\text{Traces}(K'), 0)$. Define

$$
\theta_{K'} := p_{w_0} \land \Box \bigvee_{w \in W} \left( p_w \land (\bigwedge_{w \neq v \in W} \neg p_v) \land (\bigwedge_{(w, v) \in R} \top \subseteq \Box p_v) \land (\bigvee_{(w, v) \in R} p_v) \right) \\
\land \bigwedge_{p \in \eta(w)} p \land \bigwedge_{p \in \Delta P \setminus \eta(w)} \neg p.
$$

It is not hard to verify that the only sets of traces over $\Delta P \cup \{p_w \mid w \in W\}$ for which $(T, 0) \models \theta_{K'}$ holds are the empty set and $\text{Traces}(K')$. Now, given a finite Kripke structure $K$ and a formula $\varphi$ of TeamLTL$(\subseteq, \Box)$ $(\text{TeamLTL}(\subseteq, \Box, A)$, resp.) it holds that $(\text{Traces}(K), 0) \models \varphi$ iff $\theta_{K'} \land \varphi$ is satisfiable.

\[\blacktriangleleft\]
Lemma 5. For every set $T$ of traces over a countable $AP$, there exists a countable set $S_T \subseteq 2^T$ such that, for every TeamLTL($\emptyset, \sim \bot, A$)-formula $\varphi$ and $i \in \mathbb{N},$

$$(T, i) \models \varphi \iff (T, i) \models^* \varphi,$$

where the satisfaction relation $\models^*$ is defined the same way as $\models$ except that in the semantic clause for $\lor$ we require additionally that the two subteams $T_1, T_2 \subseteq S_T$.

Proof. We first define inductively and nondeterministically a function $\text{Sub} : (2^T \times \mathbb{N}) \times \text{TeamLTL}(\emptyset, A) \to 2^T$ as follows:

- If $\varphi$ is an atomic formula or a generalised atom, define $\text{Sub}((S, j), \varphi) := \{S\}$
- $\text{Sub}((S, j), \bigcirc \varphi) := \text{Sub}((S, j), \varphi)$
- $\text{Sub}((S, j), \varphi \land \psi) := \text{Sub}((S, j), \varphi) \cup \text{Sub}((S, j), \psi)$
- $\text{Sub}((S, j), \varphi \lor \psi) := \text{Sub}((S, j), \varphi) \cup \text{Sub}((S, j), \psi)$
- $\text{Sub}((S, j), \varphi \rightarrow \psi) := \text{Sub}((S, j), \varphi) \cup \text{Sub}((S, j), \psi)$
- $\text{Sub}((S, j), \exists \psi) := \{S\} \cup \text{Sub}((S, j), \psi)$

Clearly $\text{Sub}((S, j), \psi) \subseteq S_T$ is a countable set. Now, define

$S_T := \bigcup_{j \in \mathbb{N}} \text{Sub}((T, j), \psi).$

The set $S_T$ is a countable union of countable sets, and thus itself countable. For each team $(S, j)$ and formula $\psi$, assuming $S_T \supseteq \text{Sub}((S, j), \psi)$, it is not hard to show by induction that (2) holds.

Theorem 6. For every $\varphi \in \text{TeamLTL}(\emptyset, \sim \bot, A)$ there exists an equivalent HyperQPTL$^+$-formula $\varphi^*$ in the $\exists_p Qp_\forall^\pi \exists^\pi$ fragment. If $\varphi \in \text{TeamLTL}(\emptyset, A)$, $\varphi^*$ can be defined in the $\exists_p Qp_\forall^\pi$ fragment. The size of $\varphi^*$ is linear w.r.t. the size of $\varphi$.

Proof. Let $q^{S_T}, q,$ and $r$ be distinct propositional variables. We define a compositional translation $\text{TR}_{(q, r)}$ such that for every team $(T, i)$ and TeamLTL($\emptyset, \sim \bot, A$)-formula $\varphi$,

$$(T, i) \models \varphi \iff (T, i) \models_{T, i} \exists q^{S_T} \exists q^{S_T} \forall q^{S_T} \forall q^{S_T} (\text{TR}_{(q, r)}(\varphi) \land \varphi_{\text{aux}}),$$

where $\varphi_{\text{aux}} := \forall \pi. q^{S_T} \land q^{S_T} \land r^{S_T}.$ (3)

We first fix some conventions. All quantified variables in the translation below are assumed to be fresh and distinct. We also assume that the uniformly quantified propositional variables $q, r, \ldots$ are true in exactly one level, that is, they satisfy the formula $\forall \pi. q^{S_T} \land q^{S_T} \land r^{S_T}.$ We will omit the presentation of the translation for simplicity.

The idea behind the translation is the following. Let $(T', i)$ denote the team obtained from $(T, i)$ by evaluating the quantifier $\exists q^{S_T}.$

- The variable $q^{S_T}$ is used to encode the countable set $S_T$ of sets of traces given by Lemma 5. To be precise, for each $i \in \mathbb{N}, q^{S_T}$ encodes the set $\{t \in T' \mid q^{S_T} \in t[i]\}$.
- The uniformly quantified variable $q$ in $\text{TR}_{(q, r)}$ is used to encode an element of $S_T$ using $q^{S_T}$: If $q \in t[i],$ then $q$ encodes the set $\{t \in T' \mid q^{S_T} \in t[i]\}$.
- The uniformly quantified variable $r$ in $\text{TR}_{(q, r)}$ is used to encode the time step $i$ of a team $(T, i)$: If $r \in t[i],$ then $r$ encodes the time step $i.$
After fixing a suitable interpretation for $q^{S_T}$, teams $(S,i)$ can be encoded with pairs of uniformly quantified variables $(q,r)$, whenever $S \in S_T$. The formula $\varphi_{aux}$ expresses that the pair $(q,r)$ encodes the team $(T,i)$ in question.

The translation $\text{TR}_{(q,r)}(\ell)$ is defined inductively as follows:

$$\text{TR}_{(q,r)}(\ell) := \forall \pi. (\Diamond(q_{\pi} \land q_{F_{\pi}^r}) \rightarrow \Diamond(r_{\pi} \land \ell_{\pi})),$$

where $\ell \in \{\varnothing, \neg \varnothing\}$.

$$\text{TR}_{(q,r)}(\varphi \land \psi) := \text{TR}_{(q,r)}(\varphi) \land \text{TR}_{(q,r)}(\psi).$$

$$\text{TR}_{(q,r)}(\varphi) := \begin{cases} \exists q'. (\Box(q_{\pi} \leftrightarrow q'_{\pi}) \land \text{TR}_{(q',r')}(\varphi)) & \text{if } \varphi \in S, \Box; \\
\text{TR}_{(q,r)}(\varphi) & \text{if } \varphi \notin S, \Box. \end{cases}$$

where $\varphi_{aux}(q,q',q^*_{\pi}) := \Diamond(q_{\pi} \land q^*_{F_{\pi}^r} \land q^*_{\pi}) \rightarrow \Diamond((q'_{\pi} \lor q^*_{\pi}) \land q^*_{\pi})$.

Next we describe the procedure needed for the case $\hat{A}$ of the translation of Theorem 6. It suffices to establish a linear translation $^*: \text{TeamLTL}(\varnothing, \sim \bot) \rightarrow \text{TeamLTL}$ such that $\hat{A}\varphi \equiv \hat{A}\varphi^*$, for every $\varphi \in \text{TeamLTL}(\varnothing, \sim \bot)$. The translation is written bottom-up, and the symbols $\varnothing$ and $\sim \bot$ do not occur in the formulae $\psi$ and $\theta$ below. (We wish to remind the reader that all TeamLTL formulae are satisfied by the empty team.) For formulae $\varphi$ in which $\sim \bot$ does not occur, the translation only replaces the symbols $\varnothing$ by $\lor$. In other cases the translation depends on the position of $\sim \bot$ in the subformulae. Below $\psi$ and $\theta$ are allowed to be empty (i.e., $\sim \bot \land \psi$ covers the case $\sim \bot$).

$$\begin{align*}
(\sim \bot \land \psi) \lor (\sim \bot \land \theta) & \quad \rightarrow \quad \sim \bot \land (\psi \lor \theta) \\
(\sim \bot \land \psi) \lor \theta & \quad \rightarrow \quad \sim \bot \land \psi \\
(\sim \bot \land \psi) \land (\sim \bot \land \theta) & \quad \rightarrow \quad \sim \bot \land (\psi \land \theta) \\
(\sim \bot \land \psi) \land \theta & \quad \rightarrow \quad \sim \bot \land (\psi \land \theta) \\
\Box(\sim \bot \land \psi) & \quad \rightarrow \quad \sim \bot \land \Box \psi \\
(\sim \bot \land \psi) \mathcal{U}(\sim \bot \land \theta) & \quad \rightarrow \quad \sim \bot \land (\psi \mathcal{U} \theta) \\
\psi \mathcal{U}(\sim \bot \land \theta) & \quad \rightarrow \quad \sim \bot \land (\psi \mathcal{U} \theta) \\
(\sim \bot \land \psi) \mathcal{U} \theta & \quad \rightarrow \quad \psi \mathcal{U} \theta \\
(\sim \bot \land \psi) \mathcal{W}(\sim \bot \land \theta) & \quad \rightarrow \quad \sim \bot \land (\psi \mathcal{W} \theta) \\
\psi \mathcal{W}(\sim \bot \land \theta) & \quad \rightarrow \quad \psi \mathcal{W} \theta \\
(\sim \bot \land \psi) \mathcal{W} \theta & \quad \rightarrow \quad \psi \mathcal{W} \theta
\end{align*}$$
It is straightforward to check that, for any TeamLTL-formulae \( \psi \) and \( \theta \), the above translation preserves the truth value of formulae over teams of cardinality at most 1. Hence, when using the translation to \( \varphi \), we obtain a formula \( \varphi^+ \) of the form \( \sim \land \psi \) or \( \psi \), where \( \psi \in \text{TeamLTL} \), such that \( \varphi \) and \( \varphi^+ \) are equivalent. Finally note that, by the semantics of \( \hat{A} \), the formulae \( \hat{A}(\sim \land \psi) \) and \( \hat{A}\psi \) are equivalent. By defining \( \varphi^* := \psi \) we obtain the linear size formula needed in Theorem 6.

### D Missing Proofs of Section 5

**Lemma 8.** Let \( \varphi \) be a formula of TeamLTL(\( \sim \)) and \( \Phi = \{\pi_1, \ldots, \pi_k\} \) a finite set of trace variables. For any team \( (T, i) \) with \( |T| \leq k \), any set \( S \supseteq T \) of traces, and any assignment \( \Pi \) with \( \Pi[\Phi] = T \), we have that \( (T, i) \models \varphi \iff \Pi, i \models_S \varphi^\Phi \). Furthermore, if \( \varphi \) is downward closed and \( T \neq \emptyset \), then \( (T, i) \models \varphi \iff \emptyset, i \models_T \forall \pi_1 \ldots \forall \pi_k. \varphi^\Phi \).

**Proof.** The second claim of the lemma follows easily from the first claim. We prove the first claim by induction on \( \varphi \). If \( \varphi = p \), then

\[
(T, i) \models p \iff \forall t \in T : p \in t[i] \\
\iff \forall \pi \in \Phi : p \in \Pi(\pi)[i] \\
\iff \Pi, i \models_S \bigwedge_{\pi \in \Phi} p_\pi \\
\iff \Pi, i \models_S p^\Phi
\]

Note that if \( T = \emptyset \), then \( \Phi = \emptyset \) and by definition \( p^\emptyset = \top \). In this special case the above proof still goes through, and in particular \( (\emptyset, i) \models p \) and \( \Pi, i \models_S \top \).

The case \( \varphi = \sim \psi \) is proved analogously. If \( \varphi = \sim \psi \), then

\[
(T, i) \models \sim \psi \iff (T, i) \not\models \psi \\
\iff (\Pi, i) \not\models_S \psi^\Phi \\
\iff (\Pi, i) \models_S \sim \psi^\Phi.
\]

If \( \varphi = \psi \lor \chi \), then

\[
(T, i) \models \psi \lor \chi \\
\iff (\Pi[\Phi_0], i) \models \psi \text{ and } (\Pi[\Phi_1], i) \models \chi \text{ for some } \Phi_0, \Phi_1 \\
\text{with } \Phi_0 \cup \Phi_1 = \Phi \text{ (since } T = \Pi[\Phi] \text{)} \\
\iff \Pi, i \models_S \psi^{\Phi_0} \text{ and } \Pi, i \models_S \chi^{\Phi_1} \text{ for some } \Phi_0, \Phi_1 \text{ with } \\
\Phi_0 \cup \Phi_1 = \Phi \text{ (by IH, } S \supseteq T = \Pi[\Phi] \supseteq \Pi[\Phi_0], \Pi[\Phi_1] \text{)} \\
\iff \Pi, i \models_S \bigvee_{\Phi_0 \cup \Phi_1 = \Phi} (\psi^{\Phi_0} \land \chi^{\Phi_1}) \\
\iff \Pi, i \models_S (\psi \lor \chi)^\Phi.
\]

If \( \varphi = \chi U \psi \), then

\[
(T, i) \models \chi U \psi \\
\iff \exists n \geq i : (T, n) \models \psi, \text{ and } \forall m : i \leq m < n \Rightarrow (T, m) \models \chi \\
\iff \exists n \geq i : \Pi, n \models_S \psi^\Phi, \text{ and } \forall m : i \leq m < n \Rightarrow \Pi, m \models_S \chi^\Phi \\
\iff \Pi, i \models_S \chi^\Phi U \psi^\Phi
\]
\[\therefore \Pi, i \models s \ (\chi U \psi)^s.\]

The other inductive cases are proved by using a routine argument. \hfill \blacktriangleleft

\textbf{Theorem 9.} Every k-coherent property that is definable in TeamLTL(\textasciitilde) is also definable in \(\forall^k\text{HyperLTL}\).

\textbf{Proof.} Let \(\varphi\) be a k-coherent TeamLTL(\textasciitilde)-formula. If \(\emptyset, i \not\models \varphi\) for some \(i \in \mathbb{N}\), then \((T, j) \not\models \varphi\) for every team \((T, j)\). We then translate \(\varphi\) to \(\forall\pi \perp\). Assume now \(\emptyset, i \models \varphi\). For any nonempty team \((T, i)\), we have that

\[\begin{align*}
&\text{Assume that} \\
&\text{where the right-most satisfaction relation is the standard one for TeamLTL.(\textasciitilde)}
\end{align*}\]

\textbf{Theorem 12.} Checking whether a TeamLTL(\(\subseteq, \otimes\))-formula is 1-coherent is undecidable.

\textbf{Proof.} For a given TeamLTL(\(\subseteq, \otimes\))-formula \(\varphi\), let \(\varphi^*\) denote the TeamLTL-formula obtained from \(\varphi\) by first replacing each disjunctions \(\otimes\) by \(\lor\), and then replacing each inclusion atom of the form \(\varphi_1, \ldots, \varphi_n \subseteq \psi_1, \ldots, \psi_n\) by the conjunction

\[\bigwedge_{i \leq n} \varphi_i \leftrightarrow \psi_i.\]

It is easy to verify, by induction, that over singleton teams \(\varphi\) and \(\varphi^*\) are equivalent. Moreover, since TeamLTL satisfies the singleton equivalence property, we obtain the following equivalences, for every trace \(t\):

\[\begin{align*}
&\text{((\{t\}, i) \models \varphi) \iff ((\{t\}, i) \models \varphi^* \iff (t, i) \models \varphi^*},
\end{align*}\]

where the right-most satisfaction relation is the standard one for LTL.

Next, we show that for any formula \(\varphi \in \text{TeamLTL}(\subseteq, \otimes)\)

\[\varphi\text{ is not satisfiable} \iff \varphi\text{ is 1-coherent and }\varphi^*\text{ is not satisfiable in the sense of LTL.}\]

Assume that \(\varphi\) is not satisfiable. Since no team satisfies \(\varphi\), it is trivially 1-coherent. In particular, for any trace \(t\), we have \((\{t\}, i) \not\models \varphi\), which then implies that \((t, i) \not\models \varphi^*\). Thus \(\varphi^*\) is not satisfiable in the sense of LTL. Conversely, suppose \(\varphi\) is 1-coherent and \(\varphi^*\) is not satisfiable. Then for any trace \(t\), we have \((t, i) \not\models \varphi^*\), which implies that \((\{t\}, i) \not\models \varphi\). It then follows, by 1-coherence, that \((T, i) \not\models \varphi\) for every team \(T\). Thus \(\varphi\) is not satisfiable.

Now, since checking LTL-satisfiability can be done in \(\text{PSPACE}[45]\) and non-satisfiability for TeamLTL(\(\subseteq, \otimes\)) is \(\Pi^0_2\)-hard by Corollary 4, we conclude that checking 1-coherence is undecidable. \hfill \blacktriangleleft

\textbf{Theorem 14.} For every formula \(\varphi\) from the left-flat fragment of TeamLTL(\(\otimes, \hat{A}\)), we can compute an equivalent \(\forall^p \chi \varphi\) HyperQPTL formula of size linear in the size of \(\varphi\).

The theorem follows from the following lemma.
Lemma 20. Let $\varphi$ be a TeamLTL($\Box, \land$)-formula, and $r_1, \ldots, r_n$ free variables occurring in $[\varphi, r]$ but not in $\varphi$. Let $i \in \mathbb{N}$, and $s \in (2^r)^\omega$ a sequence that has $r$ set exactly at position $i$. For every team $(T, i)$,

$$(T, i) \models \varphi \iff \emptyset, 0 \models_{T[r \mapsto s]} \exists r_1 \ldots r_n \forall \pi. [\varphi, r].$$

Proof. We proceed by induction on $\varphi$.

Case for $\Box p$:

$$(T, i) \models \Box p \iff \forall t \in T : p \in t[i]$$

$$(T, i) \models \emptyset, 0 \models_{T[r \mapsto s]} \forall \pi. \Box(r_{\pi} \rightarrow p_{\pi})$$

Case for $\neg p$: Similar to the above.

Case for $\land$:

$$(T, i) \models \land \varphi \iff \forall t \in T : (t, i) \models \varphi$$

$$(T, i) \models \emptyset, 0 \models_{T} \forall \pi. \Box(r_{\pi} \rightarrow \varphi)$$

(by Theorem 9, as $\land \varphi$ is 1-coherent)

Case for $\lor$:

$$(T, i) \models \lor \varphi \iff \exists t_1, t_2 \text{ s.t. } T = T_1 \cup T_2$$

$$(T_1, i) \models \varphi \text{ and } (T_2, i) \models \varphi$$

$$(T_1, i) \models \emptyset, 0 \models_{T_1[r \mapsto s]} \exists r_1 \ldots r_n \forall \pi. [\varphi, r]$$

$$(T_2, i) \models \emptyset, 0 \models_{T_2[r \mapsto s]} \exists r_2 \ldots r_n \forall \pi. [\varphi, r]$$

$$(T, i) \models \emptyset, 0 \models_{T[r \mapsto s]} \exists r_1 \ldots r_n \forall \pi. [\varphi, r] \lor [\psi, r]$$

(since $T_1[r \mapsto s] \cup T_2[r \mapsto s] = T[r \mapsto s]$)

Case for $\land$:

$$(T, i) \models \land \varphi \iff \forall t \in T : p \in t[i]$$

$$(T, i) \models \emptyset, 0 \models_{T[r \mapsto s]} \land \varphi$$

$(T, i) \models \emptyset, 0 \models_{T[r \mapsto s]} \exists r_1 \ldots r_n \forall \pi. [\varphi, r]$ or

$$(T, i) \models \emptyset, 0 \models_{T[r \mapsto s]} \exists r_2 \ldots r_n \forall \pi. [\varphi, r]$$

$$(T, i) \models \emptyset, 0 \models_{T[r \mapsto s]} \exists r_1 \ldots r_n \forall \pi. [\varphi, r] \land [\psi, r]$$

$$(T, i) \models \emptyset, 0 \models_{T[r \mapsto s]} \exists r_1 \ldots r_n \forall \pi. (d_{\pi}^{\varphi} \rightarrow [\varphi, r]) \land (d_{\pi}^{\psi} \rightarrow [\psi, r])$$

Case for $U$:

$$(T, i) \models U \varphi \iff \exists i' \geq i, (T, i') \models \varphi$$

$$(T, i) \models \emptyset, 0 \models_{T[r \mapsto s]} \forall i \leq i' < i'. (T, i') \models \varphi$$

$$(T, i) \models \emptyset, 0 \models_{T[r \mapsto s]} \forall i \leq i' < i'. \exists i'' \geq i, (T, i'') \models \varphi$$

(by Theorem 9, since $\varphi$ is 1-coherent)
\[ \forall i' \geq i, 0, 0 \models_{T[r^\psi ightarrow s^\psi]} \exists r_1 \ldots r_n. \forall \pi. [\psi, r^\psi] \text{ and } \forall i \leq i'' < i', 0, i'' \models_T \forall \pi. \phi \]

(where \( r^\psi \) is set in \( s^\psi \) exactly at position \( i' \))

\[ \leftrightarrow \exists i'. 0, 0 \models_{T[r^\psi ightarrow s^\psi, r^\phi \rightarrow s^\phi]} \exists r_1 \ldots r_n. \forall \pi. [\phi^r \rightarrow \hat{\phi}] \wedge [\psi, r^\psi] \]

(where \( r^\psi \) is set in \( s^\psi \) exactly at position \( i' \) and \( r^\phi \) is set in \( s^\phi \) exactly at all positions between \( i \) and \( i' \))

\[ \leftrightarrow 0, 0 \models_{T[r^\psi]} = \exists r^\psi. \exists r^\phi. \exists r_1 \ldots r_n. \forall \pi. [\phi^r \rightarrow r^\phi \bigwedge \square \phi^r \rightarrow \hat{\phi}] \wedge \square \phi^r \rightarrow \hat{\phi}] \wedge [\psi, r^\psi] \]

Case for \( \phi \models \psi \). Similar to \( \phi \models U \psi \).