

Synchronous Team Semantics for Temporal Logics

ANDREAS KREBS, University of Tübingen, Tübingen, Germany

ARNE MEIER, Institut für Theoretische Informatik, Leibniz Universität Hannover, Hannover, Germany

JONNI VIRTEMA, University of Glasgow, Glasgow, UK

MARTIN ZIMMERMANN, Aalborg University, Aalborg, Denmark

We present team semantics for two of the most important linear and branching time specification languages, Linear Temporal Logic (LTL) and Computation Tree Logic (CTL).

With team semantics, LTL is able to express hyperproperties, which have in the last decade been identified as a key concept in the verification of information flow properties. We study basic properties of the logic and classify the computational complexity of its satisfiability, path and model-checking problems. Further, we examine how extensions of the basic logic react to adding additional atomic operators. Finally, we compare its expressivity to that one of HyperLTL, another recently introduced logic for hyperproperties. Our results show that LTL with team semantics is a viable alternative to HyperLTL, which complements the expressivity of HyperLTL and has partially better algorithmic properties.

For CTL with team semantics, we investigate the computational complexity of the satisfiability and model-checking problem. The satisfiability problem is shown to be EXPTIME-complete, while we show that model checking is PSPACE-complete.

CCS Concepts: • **Theory of computation** → **Modal and temporal logics; Problems, reductions and completeness**;

Additional Key Words and Phrases: Temporal Logics, Team Semantics, Model Checking, Satisfiability

ACM Reference format:

Andreas Krebs, Arne Meier, Jonni Virtema, and Martin Zimmermann. 2026. Synchronous Team Semantics for Temporal Logics. *ACM Trans. Comput. Logic* 27, 4, Article 26 (October 2026), 36 pages.

<https://doi.org/10.1145/3834561>

1 Introduction

Guaranteeing security and privacy of user information is a key requirement in software development. However, it is also one of the hardest goals to accomplish. One reason for this difficulty is that such requirements typically amount to reasoning about the flow of information and relating different execution traces of the system in question. Many of the requirements of interest are

This work was supported by the Deutsche Forschungsgemeinschaft projects ME4279/1-1 and ME4279/3-1, and DIREC—Digital Research Centre Denmark.

Authors' Contact Information: Andreas Krebs, University of Tübingen, Tübingen, Germany; e-mail: mail@krebs-net.de; Arne Meier, Institut für Theoretische Informatik, Leibniz Universität Hannover, Hannover, Germany; e-mail: meier@thi.uni-hannover.de; Jonni Virtema, University of Glasgow, Glasgow, UK; e-mail: jonni.virtema@glasgow.ac.uk; Martin Zimmermann (corresponding author), Aalborg University, Aalborg, Denmark; e-mail: mzi@cs.aau.dk.



This work is licensed under Creative Commons Attribution International 4.0.

© 2026 Copyright held by the owner/author(s).

ACM 1557-945X/2026/10-ART26

<https://doi.org/10.1145/3834561>

not trace properties, that is, properties whose satisfaction can be verified by considering each computation trace in isolation. Formally, a trace property φ is a set of traces and a system satisfies φ if each of its traces is in φ . An example of a trace property is the property ‘the system terminates eventually’, which is satisfied if every trace eventually reaches a terminating state. In contrast, the property ‘the system terminates within a bounded amount of time’ is no longer a trace property. Consider a system that has a trace t_n for every n , so that t_n only reaches a terminating state after n steps. This system does not satisfy the bounded termination property, but each individual trace t_n could also stem from a system that does satisfy it. Thus, satisfaction of the property cannot be verified by considering each trace in isolation. Properties with this characteristic were termed *hyperproperties* by Clarkson and Schneider [17]. Formally, a hyperproperty φ is a set of sets of traces, and a system satisfies φ if its set of traces is contained in φ . The conceptual difference to trace properties allows hyperproperties to specify a much richer landscape of properties, including information flow properties capturing security and privacy specifications. Furthermore, one can also express specifications for symmetric access to critical resources in distributed protocols and Hamming distances between code words in coding theory [60]. However, the increase in expressiveness requires novel approaches to specification and verification.

HyperLTL. Trace properties are typically specified in temporal logics, most prominently in **Linear Temporal Logic (LTL)** [58]. Verification of LTL specifications is routinely employed in industrial settings and marks one of the most successful applications of formal methods to real-life problems. Recently, this work has been extended to hyperproperties: HyperLTL, LTL equipped with trace quantifiers, has been introduced to specify hyperproperties [16]. A model of a HyperLTL formula is a set of traces, and the quantifiers range over these traces. This logic is able to express the majority of the information flow properties found in the literature (we refer to Section 3 of [16] for a comprehensive list). The satisfiability problem for HyperLTL is highly undecidable [27] while the model-checking problem is decidable, albeit of non-elementary complexity [16, 24]. In view of this, the full logic is too strong for practical applications. Fortunately, most information flow properties found in the literature can be expressed with at most one quantifier alternation and consequently belong to decidable (and tractable) fragments. Further works on HyperLTL have studied runtime verification [11, 22], connections to **First-Order (FO)** logic [25], provided tool support [21, 24] and presented applications to ‘software doping’ [18] and the verification of web-based workflows [23]. Recent works have also considered asynchronous extensions of HyperLTL [5, 8, 13, 32] and verification tools for full HyperLTL [10] (most of the previous tools were designed for fragments without quantifier alternation). In contrast, there are natural properties, e.g., bounded termination, which are not expressible in HyperLTL (which is an easy consequence of a much stronger non-expressibility result [12]).

Team Semantics. Intriguingly, there exists another modern family of logics which operates on sets of objects instead of objects alone. In 1997, Hodges introduced compositional semantics for Hintikka’s Independence-friendly logic [38] where the semantical object is a set of FO assignments. Hence formulae, in this setting, define sets of sets of FO assignments. We could call these definable sets *FO hyperproperties*, while Hodges himself called them *trumps*. A decade later, Väänänen [68] introduced *Dependence logic* that adopted Hodges’ semantics and reimagined Independence-friendly logic. Dependence logic extends FO logic by atoms expressing that ‘the value of a variable x functionally determines the value of a variable y ’. Obviously, such statements only make sense when being evaluated over a set of assignments. Therefore, they are, using the parlance introduced above, hyperproperties. In the language of dependence logic, such sets are called *teams*, and the semantics is termed *team semantics*.

After the introduction of dependence logic, a whole family of logics with different atomic statements have been introduced in this framework: *independence logic* [30] and *inclusion logic*

Logic	SAT	PC	MC
LTL	PSPACE [66]	P [46, 53]	PSPACE [66]
TeamLTL	PSPACE (Prop. 3.2)	PSPACE (Thm. 4.3)	PSPACE-hard (Thm. 4.3)
TeamLTL(dep)	PSPACE (Prop. 5.5)	PSPACE (Thm. 5.2)	NEXPTIME-hard (Thm. 5.7)
TeamLTL($\odot, \mathcal{D}$)	Σ_1^0 -hard [73]	PSPACE (Thm. 5.2)	Σ_1^0 -hard [73]
TeamLTL($\mathcal{D}, \sim$)	third-order arithmetic [51] ^a	PSPACE (Thm. 5.2)	third-order arithmetic [51] ^a
TeamLTL $- \vee$	?	?	$\in$ PSPACE (Thm. 4.4)
CTL	EXPTIME [26, 59]	—	P [15, 64]
TeamCTL	EXPTIME (Thm. 9.9)	—	PSPACE (Thm. 9.6)
TeamCTL($\mathcal{D}, \sim$)	?	—	PSPACE (Thm. 9.7)

Fig. 1. Overview of complexity results for TeamLTL and TeamCTL. ‘dep’ refers to dependence atoms, ‘ $\sim$ ’ refers to the contradictory negation, $\odot$ refers to the Boolean disjunction, $\mathcal{D}$ refers to any finite set of FO definable generalised atoms and ‘TeamLTL $- \vee$ ’ refers to disjunction-free TeamLTL. All results are completeness results unless otherwise stated. We write ‘?’ for open cases and ‘—’ if the problem is not meaningful for the logic. Results concerning $\sim$ remain true for extensions with connectives that are expressible with at most polynomial blow-up. In particular, $\varphi \odot \psi$ is equivalent with $\sim (\sim \varphi \wedge \sim \psi)$. ^aLück [51] proves this result without generalised atoms. However, since all generalised atoms are expressible in TeamLTL($\sim$) with double exponential blow-up in the formula size (see [72, Proposition 17] and [51, page 6]), the result carries over to the extension of TeamLTL($\sim$) with generalised atoms.

[29] being the most prominent. Interest in these logics is rapidly growing, and connections to a plethora of disciplines have been drawn, e.g., to database theory [35], real-valued computation [34], quantum foundations [1, 2], and to the study of argumentation [52] and causation [4].

Our Contribution. The conceptual similarities between hyperproperties and team properties raise the question *what is the natural team semantics for temporal logics*. In this article, we develop team semantics for LTL and **Computation Tree Logic (CTL)**, obtaining the logics TeamLTL and TeamCTL. We analyse their expressive power and the complexity of their satisfiability and model-checking problems and subsequently compare TeamLTL to HyperLTL. While team semantics has previously been defined for propositional and modal logic [69], we are the first to consider team semantics for temporal logics [44].

Our complexity results are summarised in Figure 1. We prove that the satisfiability problem (TSAT) for TeamLTL is PSPACE-complete by showing that the problem is equivalent to LTL satisfiability under classical semantics. We consider two variants of the model-checking problem. As there are uncountably many traces, we have to represent teams, i.e., sets of traces, in a finitary manner. The path-checking problem (TPC) asks to check whether a finite team of ultimately periodic traces satisfies a given formula. As our main technical result, we establish this problem to be PSPACE-complete. In the (general) model-checking problem (TMC), a team is represented by a finite transition system. Formally, given a transition system and a formula, the model-checking problem asks to determine whether the set of traces of the system satisfies the formula. We give a polynomial space algorithm for the model-checking problem for the disjunction-free fragment, while we leave open the complexity of the general problem. Disjunction plays a special role in team semantics, as it splits a team into two. As a result, this operator is commonly called *splitjunction*. Note that, even for a very simple transition system, the set of traces generated by the system may be uncountable. Hence, evaluating the splitjunction compositionally may require us to deal with infinitely many splits of an uncountable team, if a splitjunction is under the scope of a globally-operator (such as in the formula $G(\varphi \vee \psi)$). This raises interesting language-theoretic questions. Additionally, we study the effects on complexity that follow when our logics are extended by dependence atoms, the

contradictory negation $\sim$, the Boolean disjunction $\oplus$, and so-called *generalised atoms* [48]. Finally, we show that TeamLTL is able to specify properties which are not expressible in HyperLTL and *vice versa*.

Recall that satisfiability for HyperLTL is highly undecidable [27] and model checking is of non-elementary complexity [54, 60]. Our results show that similar problems for TeamLTL have a much simpler complexity, while some hyperproperties are still expressible (e.g., input determinism, see Section 5, or bounded termination). This demonstrates that TeamLTL is a viable alternative for the specification and verification of hyperproperties that complements HyperLTL.

In the second part of the article, we develop team semantics for CTL. We establish that the satisfiability problem for the resulting logic TeamCTL is EXPTIME-complete while the model-checking problem is PSPACE-complete. While CTL satisfiability is already EXPTIME-complete for classical semantics [26, 59], CTL model checking is P-complete [15, 64] for classical semantics, i.e., team semantics increases the complexity of the problem. For the team semantics setting, we extend our model checking result to cover also finite sets of FO-definable generalised atoms and the contradictory negation. Finally, we compare the expressiveness of TeamCTL and CTL with classical semantics.

Prior Work. Preliminary versions of this work have been published in the proceedings of the 43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018 [45], and in the proceedings of the 22nd International Symposium on Temporal Representation and Reasoning, TIME 2015 [44]. The following list summarises how this article extends the conference versions:

- The presentation of the article has been thoroughly revised. Moreover, we now include an overview of the subsequent works on temporal team semantics published after our conference articles.
- The proof of Lemma 4.1 has been considerably extended and now also contains the correctness part of the reduction.
- The proof of Lemma 4.2 was omitted in the conference version and is now included.
- The full proof of Theorem 4.4 is now included.
- The formulation of Theorem 5.2 is slightly strengthened, and a proof is now included.
- Theorem 8.3 is new.
- The construction and figures in the proof of Lemma 9.2 have been improved.
- The proof of Lemma 9.5 has been corrected.
- Theorem 9.7 is new.

Related Work on TeamLTL. Since the publication of the original conference articles [44, 45], several follow-up works have been published. Krebs et al. [45] also introduced an asynchronous variant of TeamLTL. This line of work has been continued by Kontinen et al. [40–42]. In particular, Kontinen and Sandström [40] considered the extension of TeamLTL with the contradictory negation and studied its complexity and translations to FO and second-order logic. Then, Kontinen et al. [42] fixed a mistake in the original definition of asynchronous TeamLTL [45], introduced a novel set-based semantics for TeamLTL and studied its complexity and expressivity. Hence, for the definition of asynchronous TeamLTL, please refer to the work of Kontinen et al. [42]. In their recent work on this topic, Kontinen et al. [41] revealed a tight connection between set-based asynchronous TeamLTL and the one-variable fragment of HyperLTL.

Lück [51] showed that the satisfiability and model-checking problems of synchronous TeamLTL with contradictory negation are complete for third-order arithmetic. Virtema et al. [73] studied the expressivity and complexity of various extensions of synchronous TeamLTL. In particular,

they identified undecidable cases and cases when team logics can be translated to HyperQPTL and HyperQPTL⁺. By doing so, they mapped the undecidability landscape of synchronous TeamLTL.

Gutsfeld et al. [31] introduced a flexible team-based formalism to logically specify asynchronous hyperproperties based on so-called **Time Evaluation Functions (tef)**, which subsumes synchronous TeamLTL and can be used to model diverse forms of asynchronicity.

Bellier et al. [9] introduced a team-based formalism for quantified propositional temporal logic that takes inspiration from FO team logics, which hence is orthogonal to TeamLTL.

Finally, while there exists a deep understanding of the complexity of the satisfiability and model-checking problems for classical LTL [6, 7] and CTL [43, 55, 56], these investigations do neither directly nor generally carry over to team semantics.

Finally, the complexity of the most important verification problems for HyperLTL and related hyperlogics has been settled [21, 24, 27, 28, 60–62].

2 Preliminaries

The non-negative integers are denoted by $\mathbb{N}$ and the power set of a set S is denoted by 2^S . Throughout the article, we consider a countably infinite set AP of atomic propositions.

Computational Complexity. We will make use of standard notions in complexity theory. In particular, we will use the complexity classes P, PSPACE, EXPTIME and NEXPTIME.

An **Alternating Turing machine (ATM)** is a non-deterministic Turing machine whose state space is partitioned into two types of states: existential and universal. Acceptance for ATMs is defined in an inductive way on any computation tree with respect to a given input as follows. A halting configuration is accepting if and only if it contains an accepting state. An ‘inner’ configuration (which is not a halting configuration) is accepting depending on its type of state: if it is existential then it is accepting if at least one of its children is accepting, and if it is universal then it is accepting if all of its children are accepting. An ATM accepts its input if and only if its initial configuration is accepting. Finally, the complexity class ATIME-ALT(x, y) is the set of problems solvable by ATMs in a runtime of x with y -many alternations.

Most reductions used in the article are $\leq_m^P$ -reductions, that is, polynomial time, many-to-one reductions.

Traces and Teams. A *trace* over AP is an infinite sequence from $(2^{\text{AP}})^\omega$; a *finite trace* is a finite sequence from $(2^{\text{AP}})^*$. The length of a finite trace t is denoted by $|t|$. The empty trace is denoted by ε and the concatenation of a finite trace t_0 and a finite or infinite trace t_1 by t_0t_1 . Unless stated otherwise, a trace is always assumed to be infinite.

A *team* is a (potentially infinite) set of traces. Given a trace $t = t(0)t(1)t(2) \cdots$ and $i \geq 0$, we define $t[i, \infty) := t(i)t(i+1)t(i+2) \cdots$, which we lift to teams $T \subseteq (2^{\text{AP}})^\omega$ by defining $T[i, \infty) := \{t[i, \infty) \mid t \in T\}$.

To serve as inputs for algorithms, we need to represent traces, infinite sequences of possibly infinite sets, finitely. A trace t is *ultimately periodic* if it is of the form $t = t_0 \cdot t_1^\omega = t_0t_1t_1t_1 \cdots$ for two finite traces t_0 and t_1 with $|t_1| > 0$. As a result, an ultimately periodic trace t is represented by the pair (t_0, t_1) ; we define $\llbracket (t_0, t_1) \rrbracket = t_0t_1^\omega$. Given a set $\mathcal{T}$ of such pairs, we define $\llbracket \mathcal{T} \rrbracket = \{ \llbracket (t_0, t_1) \rrbracket \mid (t_0, t_1) \in \mathcal{T} \}$, which is a team of ultimately periodic traces. We call $\mathcal{T}$ a *team encoding* of $\llbracket \mathcal{T} \rrbracket$. A team encoding is finite, if it contains finitely many elements (t_0, t_1) , where each letter of t_0 and each letter of t_1 is a finite subset of AP. This implies that a finite team encoding encodes a finite team, which can be represented finitely. The *representation size* of a finite team encoding $\mathcal{T}$, denoted by $||\mathcal{T}||$, is defined as

$$\sum_{(t_0, t_1) \in \mathcal{T}} |t_0| + |t_1| + 2,$$

where the ‘+2’ accounts for adequate separator symbols.

LTL. The formulae of LTL [58] are defined via the grammar

$$\varphi := p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid X\varphi \mid \varphi U \varphi \mid \varphi R \varphi,$$

where p ranges over the atomic propositions in AP. We define the following usual shorthands: $\top := p \vee \neg p$, $\perp := p \wedge \neg p$, $F\varphi := \top U \varphi$, $G\varphi := \perp R \varphi$. The length of a formula is defined to be the number of Boolean and temporal connectives occurring in it. Often, the length of an LTL formula is defined to be the number of syntactically different subformulae, which might be exponentially smaller. Here, we need to distinguish syntactically equal subformulae, which becomes clearer after defining the semantics. As we only consider formulae in **Negation Normal Form (NNF)**, we use the full set of temporal operators.

Next, we recall the classical semantics of LTL before we introduce team semantics. For traces $t \in (2^{\text{AP}})^\omega$ we define the following:

$$\begin{aligned} t \models p & \quad \text{if } p \in t(0), \\ t \models \neg p & \quad \text{if } p \notin t(0), \\ t \models \psi \wedge \varphi & \quad \text{if } t \models \psi \text{ and } t \models \varphi, \\ t \models \psi \vee \varphi & \quad \text{if } t \models \psi \text{ or } t \models \varphi, \\ t \models X\varphi & \quad \text{if } t[1, \infty) \models \varphi, \\ t \models \psi U \varphi & \quad \text{if } \exists k \geq 0 \text{ such that } t[k, \infty) \models \varphi \text{ and } \forall k' < k \text{ we have that } t[k', \infty) \models \psi \text{ and} \\ t \models \psi R \varphi & \quad \text{if } \forall k \geq 0 \text{ we have that } t[k, \infty) \models \varphi \text{ or } \exists k' < k \text{ such that } t[k', \infty) \models \psi. \end{aligned}$$

3 Team Semantics for LTL

Next, we introduce (synchronous) team semantics for LTL, obtaining the logic TeamLTL. The syntax of TeamLTL is the same as the one of LTL, but the semantics differ: For teams $T \subseteq (2^{\text{AP}})^\omega$ we define the following:

$$\begin{aligned} T \models p & \quad \text{if } \forall t \in T \text{ we have that } p \in t(0), \\ T \models \neg p & \quad \text{if } \forall t \in T \text{ we have that } p \notin t(0), \\ T \models \psi \wedge \varphi & \quad \text{if } T \models \psi \text{ and } T \models \varphi, \\ T \models \psi \vee \varphi & \quad \text{if } \exists T_1 \cup T_2 = T \text{ such that } T_1 \models \psi \text{ and } T_2 \models \varphi, \\ T \models X\varphi & \quad \text{if } T[1, \infty) \models \varphi, \\ T \models \psi U \varphi & \quad \text{if } \exists k \geq 0 \text{ such that } T[k, \infty) \models \varphi \text{ and } \forall k' < k \text{ we have that } T[k', \infty) \models \psi \text{ and} \\ T \models \psi R \varphi & \quad \text{if } \forall k \geq 0 \text{ we have that } T[k, \infty) \models \varphi \text{ or } \exists k' < k \text{ such that } T[k', \infty) \models \psi. \end{aligned}$$

We call expressions of the form $\psi \vee \varphi$ *splitjunctions* to emphasise that in team semantics disjunction splits a team into two parts. Similarly, the $\vee$ -operator is referred to as a *splitjunction*. Notice that the object (a trace or a team) left of $\models$ determines which of the above semantics is used (classical or team semantics). Some subsequent works consider team semantics over multisets of traces [31]. For synchronous TeamLTL (as introduced here), the generalisation to multisets would allow the implementation of new quantitative dependency statements as atomic formulae. From the above semantics, only disjunction would need to be reinterpreted via disjoint (multiset) unions. Since having multiset teams would not have any meaningful impact on TeamLTL, we adopt the slightly simpler set-based semantics.

Example 3.1. If p is an atomic proposition encoding that a computation has ended, then Fp defines the hyperproperty *bounded termination*. In particular, a possibly infinite team T satisfies Fp if there is a natural number $n \in \mathbb{N}$ such that $p \in t(n)$, for every $t \in T$.

property	definition	
empty team property	$\emptyset \models \varphi$	✓
downward closure	$T \models \varphi$ implies $\forall T' \subseteq T: T' \models \varphi$	✓
union closure	$T \models \varphi$ and $T' \models \varphi$ implies $T \cup T' \models \varphi$	×
flatness	$T \models \varphi$ if and only if $\forall t \in T: \{t\} \models \varphi$	×
singleton equivalence	$\{t\} \models \varphi$ if and only if $t \models \varphi$	✓

Fig. 2. Structural properties overview for TeamLTL.

We consider several standard properties of team semantics (cf., e.g., [19]) and verify which of these hold for our semantics for LTL. These properties are later used to analyse the complexity of the satisfiability and model-checking problems. See Figure 2 for the definitions of the properties and a summary of which of the properties hold for our semantics. The positive results follow via simple inductive arguments. For the fact that team semantics is not union closed, consider teams $T = \{\{p\}\emptyset^\omega\}$ and $T' = \{\emptyset\{p\}\emptyset^\omega\}$. Then, we have $T \models Fp$ and $T' \models Fp$ but $T \cup T' \not\models Fp$. This also implies that flatness fails, for it implies union closure.

A Kripke structure $\mathcal{K} = (W, R, \eta, w_I)$ consists of a finite set W of worlds, a left-total transition relation $R \subseteq W \times W$, a labelling function $\eta: W \rightarrow 2^{AP}$ such that $\eta(w)$ is finite for each $w \in W$, and an initial world $w_I \in W$. A path π through $\mathcal{K}$ is an infinite sequence $\pi = \pi(0)\pi(1)\pi(2)\dots \in W^\omega$ such that $\pi(0) = w_I$ and $(\pi(i), \pi(i+1)) \in R$ for every $i \geq 0$. The trace of π is defined as $t(\pi) = \eta(\pi(0))\eta(\pi(1))\eta(\pi(2))\dots \in (2^{AP})^\omega$. A Kripke structure $\mathcal{K}$ induces the team $T(\mathcal{K}) = \{t(\pi) \mid \pi \text{ is a path through } \mathcal{K}\}$.

Next, we define the most important verification problems for TeamLTL, namely satisfiability and two variants of the model-checking problem. For classical LTL, one studies the path-checking problem and the model-checking problem. The difference between these two problems lies in the type of structures one considers. Recall that a model of an LTL formula is a single trace. In the path-checking problem, a trace t and a formula φ are given, and one has to decide whether $t \models \varphi$. This problem has applications to runtime verification and monitoring of reactive systems [46, 53]. In the model-checking problem, a Kripke structure $\mathcal{K}$ and a formula φ are given, and one has to decide whether every execution trace t of $\mathcal{K}$ satisfies φ .

The satisfiability problem of TeamLTL is defined as follows.

Problem: SAT(TeamLTL)—TeamLTL satisfiability.

Input: A TeamLTL formula φ .

Question: Is there a non-empty team T such that $T \models \varphi$?

The non-emptiness condition is necessary, as otherwise every formula is satisfiable due to the empty team property (see Figure 2). Also note that, due to downward closure (see Figure 2), a TeamLTL formula φ is satisfiable if and only if there is a singleton team that satisfies the formula. From this and singleton equivalence (see Figure 2), we obtain the following result from the identical result for LTL under classical semantics [66].

PROPOSITION 3.2. SAT(TeamLTL) is PSPACE-complete w.r.t. $\leq_m^P$ -reductions.

We consider the generalisation of the path-checking problem for LTL (denoted by PC(LTL)), which asks for a given ultimately periodic trace t and a given formula φ , whether $t \models \varphi$ holds. In the team semantics setting, the corresponding question is whether a given finite team comprised of ultimately periodic traces satisfies a given formula. Such a team is given by a team encoding $\mathcal{T}$. To simplify our notation, we will write $\mathcal{T} \models \varphi$ instead of $\llbracket \mathcal{T} \rrbracket \models \varphi$.

Problem: PC(TeamLTL)—TeamLTL Path Checking.
Input: A TeamLTL formula φ and a finite team encoding $\mathcal{T}$.
Question: Does $\mathcal{T} \models \varphi$?

Now, consider the generalised model-checking problem where one checks whether the team of traces of a Kripke structure satisfies a given formula. This is the natural generalisation of the model-checking problem for classical semantics, denoted by MC(LTL), which asks, for a given Kripke structure $\mathcal{K}$ and a given LTL formula φ , whether $t \models \varphi$ for every trace t of $\mathcal{K}$.

Problem: MC(TeamLTL)—TeamLTL Model Checking.
Input: A TeamLTL formula φ and a Kripke structure $\mathcal{K}$.
Question: Does $T(\mathcal{K}) \models \varphi$?

4 Complexity Results for TeamLTL

Next, we examine the computational complexity of path and model checking with respect to team semantics.

4.1 Path Checking

The following problem for **Quantified Boolean Formulae (QBF)** is well-known [49, 67] to be PSPACE-complete:

Problem: QBF-VAL—Validity problem for QBF.
Input: A QBF φ .
Question: Is φ valid?

LEMMA 4.1. PC(TeamLTL) is PSPACE-hard w.r.t. $\leq_m^P$ -reductions.

PROOF. To prove the claim of the lemma, we will show that QBF-VAL $\leq_m^P$ PC(TeamLTL). Given a QBF φ , we stipulate, w.l.o.g., that φ is of the form $\exists x_1 \forall x_2 \cdots Q x_n \chi$, where $\chi = \bigwedge_{j=1}^m \bigvee_{k=1}^3 \ell_{jk}$, $Q \in \{\exists, \forall\}$ and $x_1, \dots, x_n$ are exactly the free variables of χ and pairwise distinct. Notice that, depending on the choices made by the quantifiers, each of the m clauses of χ has to contain at least one true literal in order for φ to be valid. This will be mimicked in the reduction by particular propositions $c_1, \dots, c_m$.

In the following, we define a reduction which is composed of two functions f and g . Given a QBF φ , the function f will define an LTL formula and g will define a team such that φ is valid if and only if $g(\varphi) \models f(\varphi)$. Essentially, the team $g(\varphi)$ will contain three kinds of traces, see Figure 3, that will be used to encode variables $i \in \{1, \dots, n\}$, clauses $j \in \{1, \dots, m\}$ and literal positions $k \in \{1, 2, 3\}$:

- (1) traces which are used to mimic universal quantification ($U(i)$ and $E(i)$),
- (2) traces that are used to simulate existential quantification ($E(i)$) and
- (3) traces used to encode the matrix of φ ($L(j, k)$).

Moreover, the trace $T(i, 1)$ ($T(i, 0)$, resp.) is used inside the proof to encode an assignment that maps the variable x_i to true (false, resp.). Note that, $U(i)$, $T(i, 1)$, $T(i, 0)$, $L(j, k)$ are technically singleton sets of traces. For convenience, we identify them with the traces they contain.

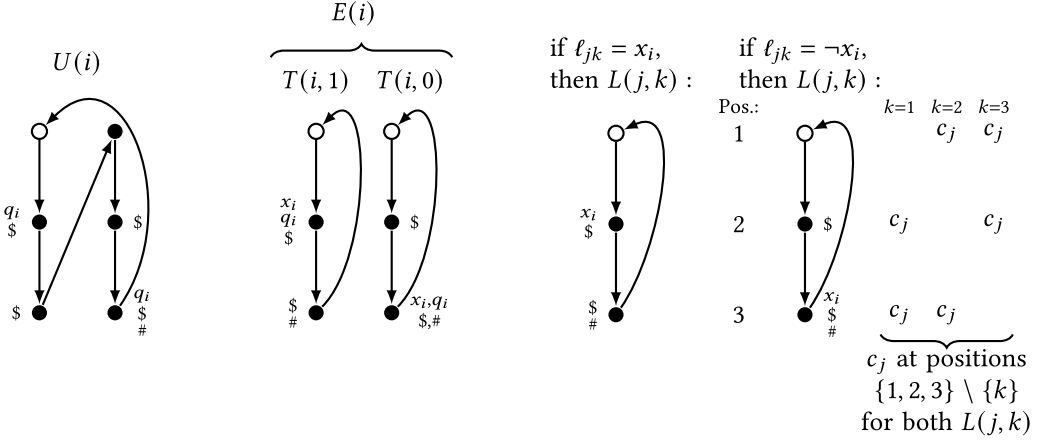


Fig. 3. Traces for the reduction presented in the proof of Lemma 4.1.

Next, we inductively define the reduction function f that maps QBFs to LTL formulae:

$$f(\chi) := \bigvee_{i=1}^n Fx_i \vee \bigvee_{i=1}^m Fc_i,$$

where χ is the 3CNF formula $\bigwedge_{j=1}^m \bigvee_{k=1}^3 \ell_{jk}$ with free variables $x_1, \dots, x_n$,

$$f(\exists x_i \psi) := (Fq_i) \vee f(\psi), \quad f(\forall x_i \psi) := (\$ \vee (\neg q_i \cup q_i) \vee F[\# \wedge \chi f(\psi)]) \cup \#.$$

Intuitively, the idea is that the left part of the formula $f(\chi)$ is used to satisfy one literal per clause and the right part is to take care of the remaining literals in a trivial way.

The reduction function g that maps QBFs to teams is defined as follows with respect to the traces in Figure 3:

$$g(\chi) := \bigcup_{j=1}^m \{L(j, 1) \cup L(j, 2) \cup L(j, 3)\},$$

$$g(\exists x_i \psi) := E(i) \cup g(\psi), \quad g(\forall x_i \psi) := U(i) \cup E(i) \cup g(\psi).$$

In Figure 3, the first position of each trace is marked with a white circle. For instance, the trace of $U(i)$ is then encoded via

$$(\varepsilon, \emptyset\{q_i, \$\}\{\$\}\emptyset\{\$\}\{q_i, \$, \#\}).$$

The reduction function showing $\text{QBF-VAL} \leq_m^P \text{PC(TeamLTL)}$ is then $\varphi \mapsto \langle g(\varphi), f(\varphi) \rangle$. Clearly $f(\varphi)$ and $g(\varphi)$ can be computed in linear time with respect to $|\varphi|$.

Intuitively, for the existential quantifier case, the formula $(Fq_i) \vee f(\psi)$ allows to continue in $f(\psi)$ with exactly one of $T(i, 1)$ or $T(i, 0)$. If $b \in \{0, 1\}$ is a truth value then selecting $T(i, b)$ in the team is the same as setting x_i to b . For the case of $f(\forall x_i \psi)$, the formula $(\neg q_i \cup q_i) \vee F[\# \wedge \chi f(\psi)]$ with respect to the team $(U(i) \cup E(i))[0, \infty)$ is similar to the existential case choosing x_i to be 1 whereas for $(U(i) \cup E(i))[3, \infty)$ one selects x_i to be 0. The use of the until operator in combination with $\$$ and $\#$ then forces both cases to happen.

Let $\varphi' = Q'x_{n'+1} \dots Qx_n \chi$, where $Q', Q \in \{\exists, \forall\}$ and let I be an assignment of the variables in $\{x_1, \dots, x_{n'}\}$ for $n' \leq n$. Then, let

$$g(I, \varphi') := g(\varphi') \cup \{T(i, I(x_i)) \mid x_i \in \text{Dom}(I)\}.$$

We claim $I \models \varphi'$ if and only if $g(I, \varphi') \models f(\varphi')$.

Note that when $\varphi' = \varphi$ it follows that $I = \emptyset$ and that $g(I, \varphi') = g(\varphi)$. Accordingly, the lemma follows from the correctness of the claim above. The claim is proven by induction on the number of quantifier alternations in φ' .

Induction Basis. $\varphi' = \chi$, this implies that φ' is quantifier-free and $\text{Dom}(I) = \{x_1, \dots, x_n\}$.

‘ $\Leftarrow$ ’: Let $g(I, \varphi') = T_1 \cup T_2$ s.t. $T_1 \models \bigvee_{i=1}^n Fx_i$ and $T_2 \models \bigvee_{i=1}^m Fc_i$. We assume w.l.o.g. T_1 and T_2 to be disjoint, which is possible due to downward closure. We then have that

$$T_2 \subseteq \{L(j, k) \mid 1 \leq j \leq m, 1 \leq k \leq 3\}$$

and

$$T_1 = (\{L(j, k) \mid 1 \leq j \leq m, 1 \leq k \leq 3\} \setminus T_2) \cup \{T(i, I(x_i)) \mid 1 \leq i \leq n\}.$$

Due to construction of the traces, $L(j, k) \in T_2$ can only satisfy the subformula $Fc_{j'}$ for $j' = j$. Moreover, note that there exists no $s \in \mathbb{N}$ such that $L(j, k)(s) \ni c_j$ for all $1 \leq k \leq 3$; hence $\{L(j, 1), L(j, 2), L(j, 3)\}$ falsifies Fc_j . These two combined imply that $T_2 \not\supseteq \{L(j, 1), L(j, 2), L(j, 3)\}$, for each $1 \leq j \leq m$. However, for each $1 \leq j \leq m$, any two of $L(j, k)$, $1 \leq k \leq 3$, can belong to T_2 and hence exactly one belongs to T_1 (due to downward closure). This is true because it is impossible to satisfy all three $L(j, k)$ per clause j simultaneously. One needs to decide for one literal per clause that is ‘the’ satisfied one and is contained in T_1 . The remaining literals are in T_2 (also if one or both are satisfied by the assignment).

Now, let $T_1 = T_1^1 \cup \dots \cup T_1^n$ such that $T_1^i \models Fx_i$. Note that Fx_i can be satisfied by $T(i', I(x_{i'}))$ only for $i' = i$. Since $T_1 \supseteq \{T(i, I(x_i)) \mid 1 \leq i \leq n\}$, it follows that $T(i, I(x_i)) \in T_1^i$, for each $1 \leq i \leq n$. Note also that, if $L(j, k) \in T_1$, it has to be in T_1^i , where x_i is the variable of ℓ_{jk} . By construction of the traces, if $T(i, 1) \in T_1^i$, we have $T_1^i(1) \models x_i$ and if $T(i, 0) \in T_1^i$ then $T_1^i(2) \models x_i$. Thus, by construction of the traces $L(j, k)$, if $L(j, k) \in T_1$, then $I \models \ell_{jk}$. Since, for each $1 \leq j \leq m$, there is a $1 \leq k \leq 3$ such that $L(j, k) \in T_1$ it follows that $I \models \varphi'$.

‘ $\Rightarrow$ ’: Now assume that $I \models \varphi'$. As a result, pick for each $1 \leq j \leq m$ a single $1 \leq k \leq 3$ such that $I \models \ell_{jk}$. Denote this sequence of choices by $k_1, \dots, k_m$. Choose $g(I, \varphi') = T_1 \cup T_2$ as follows:

$$T_1 := \{L(j, k_j) \mid 1 \leq j \leq m, k_j \in \{1, 2, 3\}\} \cup \{T(i, I(x_i)) \mid 1 \leq i \leq n\},$$

$$T_2 := \{L(j, 1), L(j, 2), L(j, 3) \mid 1 \leq j \leq m\} \setminus T_1.$$

Then $T_2 \models \bigvee_{j=1}^m Fc_j$, for exactly two traces per clause are in T_2 , and we can divide $T_2 = T_2^1 \cup \dots \cup T_2^m$, where

$$T_2^j := \{L(j, k), L(j, k') \mid k, k' \in \{1, 2, 3\} \setminus \{k_j\}\},$$

and, by construction of the traces, $T_2^j \models Fc_j$, for all $1 \leq j \leq m$. Furthermore, note that $T_1 = T_1^1 \cup \dots \cup T_1^n$, where

$$T_1^i := \{L(j, k_j) \mid 1 \leq j \leq m, I(x_i) \models \ell_{jk_j}\} \cup \{T(i, I(x_i))\}.$$

Note that the left set of the union is a singleton. Now, for all $L(j, k_j) \in T_1^i$, there are two possibilities:

– $I(x_i) = 1$: then $x_i \in (L(j, k_j)(1) \cap T(i, I(x_i))(1))$.

– $I(x_i) = 0$: then $x_i \in (L(j, k_j)(2) \cap T(i, I(x_i))(2))$.

In both cases, $T_1^i \models Fx_i$, and thus $T_1 \models \bigvee_{i=1}^n Fx_i$. Hence it follows that $g(I, \varphi') \models f(\varphi')$ and the induction basis is proven.

Induction Step. ‘Case $\varphi' = \exists x_i \psi$ ’. We show that $I \models \exists x_i \psi$ if and only if $g(I, \exists x_i \psi) \models f(\exists x_i \psi)$.

First note that $g(I, \exists x_i \psi) \models f(\exists x_i \psi)$ iff $E(i) \cup g(\psi) \cup \{T(i, I(x_i)) \mid x_i \in \text{Dom}(I)\} \models (Fq_i) \vee f(\psi)$, by the definitions of f and g . Clearly, $E(i) \not\models Fq_i$, but both $T(i, 1) \models Fq_i$ and $T(i, 0) \models Fq_i$. Observe that $E(i) = \{T(i, 1), T(i, 0)\}$ and q_i does not appear positively anywhere in $g(\psi) \cup \{T(i, I(x_i))\}$

$x_i \in \text{Dom}(I) \}$. Accordingly, and by downward closure, $E(i) \cup g(\psi) \cup \{T(i, I(x_i)) \mid x_i \in \text{Dom}(I) \} \models (Fq_i) \vee f(\psi)$ if and only if

$$\exists b \in \{0, 1\} : T(i, 1 - b) \models Fq_i \text{ and } (E(i) \cup g(\psi) \cup \{T(i, I(x_i)) \mid x_i \in \text{Dom}(I) \}) \setminus T(i, 1 - b) \models f(\psi). \quad (1)$$

Since $(E(i) \cup g(\psi) \cup \{T(i, I(x_i)) \mid x_i \in \text{Dom}(I) \}) \setminus T(i, 1 - b) = T(i, b) \cup g(\psi) \cup \{T(i, I(x_i)) \mid x_i \in \text{Dom}(I) \} = g(I[x_i \mapsto b], \psi)$, Equation (1) holds if and only if $g(I[x_i \mapsto b], \psi) \models f(\psi)$, for some bit $b \in \{0, 1\}$. By the induction hypothesis, the latter holds if and only if there exists a bit $b \in \{0, 1\}$ s.t. $I[x_i \mapsto b] \models \psi$. Finally by the semantics of $\exists$ this holds if and only if $I \models \exists x_i \psi$.

‘Case $\varphi' = \forall x_i \psi$ ’. We need to show that $I \models \forall x_i \psi$ if and only if $g(I, \forall x_i \psi) \models f(\forall x_i \psi)$.

First note that, by the definitions of f and g , we have

$$g(I, \forall x_i \psi) \models f(\forall x_i \psi)$$

if and only if

$$U(i) \cup E(i) \cup g(\psi) \cup \{T(i, I(x_i)) \mid x_i \in \text{Dom}(I) \} \models (\$ \vee (\neg q_i Uq_i) \vee F[\# \wedge Xf(\psi)])U\#. \quad (2)$$

In the following, we will show that (2) is true if and only if $T(i, b) \cup g(\psi) \cup \{T(i, I(x_i)) \mid x_i \in \text{Dom}(I) \} \models f(\psi)$ for all $b \in \{0, 1\}$. From this, the correctness follows analogously as in the case for the existential quantifier.

Notice first that each trace in $U(i) \cup E(i) \cup g(\psi) \cup \{T(i, I(x_i)) \mid x_i \in \text{Dom}(I) \}$ is periodic with period length either 3 or 6, and exactly the last element of each period is marked by the symbol $\#$. Consequently, it is easy to see that (2) is true if and only if

$$(U(i) \cup E(i) \cup g(\psi) \cup \{T(i, I(x_i)) \mid x_i \in \text{Dom}(I) \})[j, \infty) \models \$ \vee (\neg q_i Uq_i) \vee F[\# \wedge Xf(\psi)], \quad (3)$$

for each $j \in \{0, 1, 2, 3, 4\}$. Note that

$$(U(i) \cup E(i) \cup g(\psi) \cup \{T(i, I(x_i)) \mid x_i \in \text{Dom}(I) \})[j, \infty) \models \$,$$

for each $j \in \{1, 2, 4\}$, whereas no non-empty subteam of $(U(i) \cup E(i) \cup g(\psi) \cup \{T(i, I(x_i)) \mid x_i \in \text{Dom}(I) \})[j, \infty)$, $j \in \{0, 3\}$, satisfies $\$$. Accordingly, (3) is true if and only if

$$(U(i) \cup E(i) \cup g(\psi) \cup \{T(i, I(x_i)) \mid x_i \in \text{Dom}(I) \})[j, \infty) \models (\neg q_i Uq_i) \vee F[\# \wedge Xf(\psi)], \quad (4)$$

for both $j \in \{0, 3\}$. Note that, by construction, q_i does not occur positively in $g(\psi)$. As a result, $X \cap g(\psi)[j, \infty) = \emptyset$, $j \in \{0, 3\}$, for all teams X s.t. $X \models \neg q_i Uq_i$. Also, none of the symbols $x_{i'}$, $c_{i'}$, $q_{i''}$, for $i', i'' \in \mathbb{N}$ with $i'' \neq i$, occurs positively in $U(i)$. On that account, $X \cap U(i)[j, \infty) = \emptyset$, $j \in \{0, 3\}$, for all X s.t. $X \models F[\# \wedge Xf(\psi)]$, for eventually each trace in X will end up in a team that satisfies one of the formulae of the form $Fx_{i'}$, $Fc_{i'}$ or $Fq_{i''}$ (see the inductive definition of f). Furthermore, notice that $(X \cap \{T(k, I(x_k)) \mid x_k \in \text{Dom}(I) \})[j, \infty) = \emptyset$ for $j \in \{0, 3\}$, for all teams X such that $X \models \neg q_i Uq_i$.

Moreover, it is easy to check that $(T(i, 1) \cup U(i))[0, \infty) \models \neg q_i Uq_i$, $(T(i, 0) \cup U(i))[0, \infty) \not\models \neg q_i Uq_i$, $(T(i, 0) \cup U(i))[3, \infty) \models \neg q_i Uq_i$ and $(T(i, 1) \cup U(i))[3, \infty) \not\models \neg q_i Uq_i$. From these, together with downward closure, it follows that (4) is true if and only if for $b_0 = 1$ and $b_3 = 0$

$$(U(i) \cup T(i, b_j))[j, \infty) \models \neg q_i Uq_i, \text{ for all } j \in \{0, 3\} \quad (5)$$

and

$$(T(i, 1 - b_j) \cup g(\psi) \cup \{T(i, I(x_i)) \mid x_i \in \text{Dom}(I) \})[j, \infty) \models F[\# \wedge Xf(\psi)], \quad (6)$$

Algorithm 1: Algorithm for PC(TeamLTL)

```

1 Procedure chk(Finite team encoding  $\mathcal{T}$ , formula  $\varphi$ );
2 if  $\varphi = p$  then return  $\bigwedge_{(t_0, t_1) \in \mathcal{T}} p \in t_0 t_1(0)$ ;
3 if  $\varphi = \neg p$  then return  $\bigwedge_{(t_0, t_1) \in \mathcal{T}} p \notin t_0 t_1(0)$ ;
4 if  $\varphi = \psi \wedge \psi'$  then return  $\text{chk}(\mathcal{T}, \psi) \wedge \text{chk}(\mathcal{T}, \psi')$ ;
5 if  $\varphi = \psi \vee \psi'$  then return  $\bigvee_{\mathcal{T}' \subseteq \mathcal{T}} \text{chk}(\mathcal{T}', \psi) \wedge \text{chk}(\mathcal{T} \setminus \mathcal{T}', \psi')$ ;
6 if  $\varphi = X\psi$  then return  $\text{chk}(\mathcal{T}[1, \infty), \psi)$ ;
7 if  $\varphi = \psi U \psi'$  then return  $(\bigvee_{k \leq \text{prfx}(\mathcal{T}) + \text{lcm}(\mathcal{T})} \text{chk}(\mathcal{T}[k, \infty), \psi') \wedge \bigwedge_{k' < k} \text{chk}(\mathcal{T}[k', \infty), \psi))$ ;
8 if  $\varphi = \psi R \psi'$  then return  $(\bigwedge_{k \leq \text{prfx}(\mathcal{T}) + \text{lcm}(\mathcal{T})} \text{chk}(\mathcal{T}[k, \infty), \psi') \vee \bigvee_{k' < k} \text{chk}(\mathcal{T}[k', \infty), \psi))$ ;

```

for both $j \in \{0, 3\}$. In fact, as (5) always is the case, (4) is equivalent to (6). By construction, (6) is true if and only if $(T(i, b) \cup g(\psi) \cup \{T(i, I(x_i)) \mid x_i \in \text{Dom}(I)\})[6, \infty) \models f(\psi)$, for both $b \in \{0, 1\}$. Now, since

$$(T(i, b) \cup g(\psi) \cup \{T(i, I(x_i)) \mid x_i \in \text{Dom}(I)\})[6, \infty) = T(i, b) \cup g(\psi) \cup \{T(i, I(x_i)) \mid x_i \in \text{Dom}(I)\}$$

the claim applies. $\square$

Now we turn our attention to proving a matching upper bound. To this end, we need to introduce some notation to manipulate team encodings. Given a pair (t_0, t_1) of finite traces $t_0 = t_0(0) \cdots t_0(n)$ and $t_1 = t_1(0) \cdots t_1(n')$, we define $(t_0, t_1)[1, \infty)$ to be $(t_0(1) \cdots t_0(n), t_1)$ if $t_0 \neq \varepsilon$, and to be $(\varepsilon, t_1(1) \cdots t_1(n')t_1(0))$ if $t_0 = \varepsilon$. Furthermore, we inductively define $(t_0, t_1)[i, \infty)$ to be (t_0, t_1) if $i = 0$, and to be $((t_0, t_1)[1, \infty))[i - 1, \infty)$ if $i > 0$. Then,

$$\llbracket (t_0, t_1)[i, \infty) \rrbracket = (\llbracket (t_0, t_1) \rrbracket)[i, \infty),$$

that is, we have implemented the prefix-removal operation on the finite representation. Furthermore, we lift this operation to team encodings $\mathcal{T}$ by defining $\mathcal{T}[i, \infty) = \{(t_0, t_1)[i, \infty) \mid (t_0, t_1) \in \mathcal{T}\}$. As a result, we have $\llbracket \mathcal{T}[i, \infty) \rrbracket = (\llbracket \mathcal{T} \rrbracket)[i, \infty)$.

Given a finite team encoding $\mathcal{T}$, let

$$\text{prfx}(\mathcal{T}) = \max\{|t_0| \mid (t_0, t_1) \in \mathcal{T}\}$$

and let $\text{lcm}(\mathcal{T})$ be the *least common multiple* of $\{|t_1| \mid (t_0, t_1) \in \mathcal{T}\}$. Then, $\mathcal{T}[i, \infty) = \mathcal{T}[i + \text{lcm}(\mathcal{T}), \infty)$ for every $i \geq \text{prfx}(\mathcal{T})$.

Furthermore, observe that if $\mathcal{T}$ is a finite team encoding and $i \geq \text{prfx}(\mathcal{T})$, then $\mathcal{T}[i, \infty)$ and $\mathcal{T}[i + \text{lcm}(\mathcal{T}), \infty)$ satisfy exactly the same TeamLTL formulae.

Remark 1. In particular, we obtain the following consequences for temporal operators (for finite $\mathcal{T}$):

$$\begin{aligned} \mathcal{T} \models \psi U \varphi &\text{ iff } \exists k \leq \text{prfx}(\mathcal{T}) + \text{lcm}(\mathcal{T}) \text{ such that } \mathcal{T}[k, \infty) \models \varphi \text{ and } \forall k' < k : \mathcal{T}[k', \infty) \models \psi. \\ \mathcal{T} \models \psi R \varphi &\text{ iff } \forall k \leq \text{prfx}(\mathcal{T}) + \text{lcm}(\mathcal{T}) \text{ we have that } \mathcal{T}[k, \infty) \models \varphi \text{ or } \exists k' < k : \mathcal{T}[k', \infty) \models \psi. \end{aligned}$$

Accordingly, we can restrict the range of the temporal operators when model checking a finite team encoding. This implies that a straightforward recursive algorithm implementing team semantics solves PC(TeamLTL).

LEMMA 4.2. PC(TeamLTL) is in PSPACE.

PROOF. Consider Algorithm 1 where $\vee$ and $\bigvee$ denote classical (meta-level) disjunctions, not splitjunctions, which are used to combine results from recursive calls.

The algorithm is an implementation of the semantics of TeamLTL with slight restrictions to obtain the desired complexity. In line 5, we only consider strict splits, i.e., the team is split into two disjoint parts. This is sufficient due to downward closure. Furthermore, the scope of the temporal operators in lines 7 and 8 is restricted to the interval $[0, \text{prfx}(\mathcal{T}) + \text{lcm}(\mathcal{T})]$. This is sufficient due to Remark 1.

It remains to analyse the algorithm's space complexity. Its recursion depth is bounded by the size of the formula. Further, in each recursive call, a team encoding has to be stored. Additionally, in lines 5, 7 and 8, a disjunction or conjunction of arity $2^{p(\|\mathcal{T}\|)}$ for some fixed polynomial p has to be evaluated. In each case, this only requires polynomial space in the input to make the recursive calls and to aggregate the return value. Thus, Algorithm 1 is implementable in polynomial space. $\square$

Combining Lemmas 4.1 and 4.2 settles the complexity of $\text{PC}(\text{TeamLTL})$.

THEOREM 4.3. $\text{PC}(\text{TeamLTL})$ is PSPACE-complete w.r.t. $\leq_m^P$ -reductions.

4.2 Model Checking

Here, we consider the model-checking problem $\text{MC}(\text{TeamLTL})$. We show that model checking for splitjunction-free formulae is in PSPACE (as is LTL model checking under standard semantics). Then, we discuss the challenges one has to overcome to generalise this result to formulae with splitjunctions, which we leave as an open problem.

THEOREM 4.4. $\text{MC}(\text{TeamLTL})$ restricted to splitjunction-free formulae is in PSPACE.

PROOF. Fix a Kripke structure $\mathcal{K} = (W, R, \eta, w_I)$ and a splitjunction-free formula φ . We define $S_0 = \{w_I\}$ and $S_{i+1} = \{w' \in W \mid (w, w') \in R \text{ for some } w \in S_i\}$ for all $i \geq 0$. By the pigeonhole principle, this sequence is ultimately periodic with a characteristic (s, p) satisfying $s + p \leq 2^{|W|}$. Next, we define a trace t over $\text{AP} \cup \{\bar{p} \mid p \in \text{AP}\}$ via

$$t(i) = \{p \in \text{AP} \mid p \in \eta(w) \text{ for all } w \in S_i\} \cup \{\bar{p} \mid p \notin \eta(w) \text{ for all } w \in S_i\}$$

that reflects the team semantics of (negated) atomic formulae, which have to hold in every element of the team.

An induction over the structure of φ shows that $T(\mathcal{K}) \models \varphi$ if and only if $t \models \bar{\varphi}$ (i.e., under classical LTL semantics), where $\bar{\varphi}$ is obtained from φ by replacing each negated atomic proposition $\neg p$ by $\bar{p}$. To conclude the proof, we show that $t \models \bar{\varphi}$ can be checked in non-deterministic polynomial space, exploiting the fact that t is ultimately periodic and of the same characteristic as $S_0 S_1 S_2 \dots$. However, as $s + p$ might be exponential, we cannot just construct a finite representation of t of characteristic (s, p) and then check satisfaction in polynomial space.

Instead, we present an on-the-fly approach which is inspired by similar algorithms in the literature. It is based on two properties:

- (1) Every S_i can be represented in polynomial space, and from S_i one can compute S_{i+1} in polynomial time.
- (2) For every LTL formula $\bar{\varphi}$, there is an equivalent non-deterministic Büchi automaton $\mathcal{A}_{\bar{\varphi}}$ of exponential size (see, e.g., [3] for a formal definition of Büchi automata and for the construction of $\mathcal{A}_{\bar{\varphi}}$). States of $\mathcal{A}_{\bar{\varphi}}$ can be represented in polynomial space, and given two states, one can check in polynomial time whether one is a successor of the other.

These properties allow us to construct both t and a run of $\mathcal{A}_{\bar{\varphi}}$ on t on the fly.

¹The characteristic of an encoding (t_0, t_1) of an ultimately periodic trace $t_0 t_1 t_1 t_1 \dots$ is the pair $(|t_0|, |t_1|)$. Slightly abusively, we say that $(|t_0|, |t_1|)$ is the characteristic of $t_0 t_1 t_1 t_1 \dots$, although this is not unique.

In detail, the algorithm works as follows. It guesses a set $S^* \subseteq W$ and a state q^* of $\mathcal{A}_{\bar{\varphi}}$ and checks whether there are $i < j$ satisfying the following properties:

- $S^* = S_i = S_j$,
- q^* is reachable from the initial state of $\mathcal{A}_{\bar{\varphi}}$ by some run on the prefix $t(0) \cdots t(i)$ and
- q^* is reachable from q^* by some run on the infix $t(i+1) \cdots t(j)$. This run has to visit at least one accepting state.

By an application of the pigeonhole principle, we can assume w.l.o.g. that j is at most exponential in $|W|$ and in $|\varphi|$.

Let us argue that these properties can be checked in non-deterministic polynomial space. Given some guessed S^* , we can check the existence of $i < j$ as required by computing the sequence $S_0 S_1 S_2 \cdots$ on-the-fly, i.e., by just keeping the current set in memory, comparing it to S^* , then computing its successor, and then discarding the current set. While checking these reachability properties, the algorithm also guesses corresponding runs as required in the second and third property. As argued above, both tasks can be implemented in non-deterministic space. To ensure termination, we stop this search when the exponential upper bound on j is reached. This is possible using a counter with polynomially many bits and does not compromise completeness, as argued above.

It remains to argue that the algorithm is correct. First, assume $t \models \bar{\varphi}$, which implies that $\mathcal{A}_{\bar{\varphi}}$ has an accepting run on t . Recall that t is ultimately periodic with characteristic (s, p) such that $s + p \leq 2^{|W|}$ and that $\mathcal{A}_{\bar{\varphi}}$ is of exponential size. Thus, an application of the pigeonhole principle yields $i < j$ with the desired properties.

Secondly, assume the algorithm finds $i < j$ with the desired properties. Then, the run to q and the one from q to q can be turned into an accepting run of $\mathcal{A}_{\bar{\varphi}}$ on t . That being so, $t \models \bar{\varphi}$. $\square$

Note that as long as we disallow splitjunctions, our algorithm is able to deal with contradictory negations and other extensions; we will return to this topic shortly in the next section.

The complexity of the general model-checking problem is left open. It is trivially PSPACE-hard, due to Theorem 4.3 and the fact that finite teams of ultimately periodic traces can be represented by Kripke structures. However, the problem is potentially much harder, as one has to deal with infinitely many splits of possibly uncountable teams with non-periodic traces, if a split occurs under the scope of a globally-operator.

5 Extensions of TeamLTL

Next, we take a brief look into extensions of our logic. Extensions present a flexible way to delineate the expressivity and complexity of team-based logics. The philosophy behind extensions is to consider what are the fundamental hyperproperties that we want a logic to be able to express and add those as new atomic expressions.

The most well-studied atomic expressions considered in team semantics are dependence and inclusion atoms. Intuitively, the dependence atom $\text{dep}(p_1, \dots, p_n; q_1, \dots, q_m)$ expresses that the truth values of the variables $q_1, \dots, q_m$ are functionally determined by the truth values of $p_1, \dots, p_n$. Formally, for Teams $T \subseteq (2^{\text{AP}})^\omega$, the satisfaction of a dependence atom $T \models \text{dep}(p_1, \dots, p_n; q_1, \dots, q_m)$ has the following meaning:

$$\forall t, t' \in T : (t(0) \stackrel{p_1}{\Leftrightarrow} t'(0) \wedge \cdots \wedge t(0) \stackrel{p_n}{\Leftrightarrow} t'(0)) \text{ implies } (t(0) \stackrel{q_1}{\Leftrightarrow} t'(0) \wedge \cdots \wedge t(0) \stackrel{q_m}{\Leftrightarrow} t'(0)),$$

where $t(0) \stackrel{p}{\Leftrightarrow} t'(0)$ means the sets $t(0)$ and $t'(0)$ agree on proposition p , i.e., both contain p or not. Observe that the formula $\text{dep}(\cdot; p)$ merely means that p has to be constant on the team. Often, due to convenience we will write $\text{dep}(p)$ instead of $\text{dep}(\cdot; p)$. Note that the hyperproperty ‘input

determinism', i.e., feeding the same input to a system multiple times yields identical behaviour, now can be very easily expressed via the formula $\text{dep}(i_1, \dots, i_n; o_1, \dots, o_m)$, where i_j are the (public) input variables and o_j are the (public) output variables.

Already in [20] it was realised that in the team semantics of modal logic, it makes sense to allow modal formulae as parameters of dependence atoms. In fact, in the modal logic setting, this is necessary in order to obtain a logic that is expressively complete for downward-closed team properties [37, Corollary 4.5]. Here, we take an analogous approach and allow parameters to dependence atoms to be LTL formulae. Intuitively, the dependence atom $\text{dep}(\varphi_1, \dots, \varphi_n; \psi_1, \dots, \psi_m)$ expresses that the truth values of the LTL formulae $\psi_1, \dots, \psi_m$ are functionally determined by the truth values of $\varphi_1, \dots, \varphi_n$. Formally, for Teams $T \subseteq (2^{\text{AP}})^\omega$, the satisfaction of a dependence atom $T \models \text{dep}(\varphi_1, \dots, \varphi_n; \psi_1, \dots, \psi_m)$ has the following meaning:

$$\forall t, t' \in T : (t \stackrel{\varphi_1}{\Leftrightarrow} t' \wedge \dots \wedge t \stackrel{\varphi_n}{\Leftrightarrow} t') \text{ implies } (t \stackrel{\psi_1}{\Leftrightarrow} t' \wedge \dots \wedge t \stackrel{\psi_m}{\Leftrightarrow} t'),$$

where $t \stackrel{\varphi}{\Leftrightarrow} t'$ means the traces t and t' agree on the truth value of φ with respect to the classical (trace-based) semantics of LTL.

Inclusion atoms $(\varphi_1, \dots, \varphi_n) \subseteq (\psi_1, \dots, \psi_n)$ on the other hand express the inclusion dependency that all the truth values occurring for $\varphi_1, \dots, \varphi_n$ must also occur as truth values for $\psi_1, \dots, \psi_n$. Hence, if $o_1, \dots, o_n$ denote public observable bits and c is a bit revealing confidential information, then the atom $(o_1, \dots, o_n, c) \subseteq (o_1, \dots, o_n, \neg c)$ expresses a form of non-inference by stating that an observer cannot infer the value of the confidential bit from the public observables.

One can also take a more abstract view and consider atoms whose semantics can be written in FO logic over some trace properties; this includes both dependence and inclusion atoms as special cases. The notion of generalised atoms in the setting of FO team semantics was introduced by Kuusisto [48]. An n -ary generalised atom is an expression of the form $\#(\varphi_1, \dots, \varphi_n)$ that takes n LTL formulae as parameters. We consider FO formulae over the signature $(A_{x_i})_{1 \leq i \leq n}$, where each A_{x_i} is a unary predicate, as defining formulae for n -ary generalised atoms. We interpret a team T as a relational structure $\mathfrak{U}(T)$ over the same signature with universe T . When we evaluate $\#(\varphi_1, \dots, \varphi_n)$, the interpretations of A_{x_i} in $\mathfrak{U}(T)$ are determined by the interpretation of the parameters of $\#$. That is, $t \in T$ is in $A_{x_i}^{\mathfrak{U}}$ if and only if $t \models \varphi_i$ in the classical semantics of LTL.

Definition 5.1. An FO formula ψ defines the n -ary generalised atom $\#$ if $T \models \#(\varphi_1, \dots, \varphi_n) \Leftrightarrow \mathfrak{U}(T) \models \psi$. In this case, $\#$ is also called an *FO-definable generalised atom*.

In the view of FO-definable atoms, the dependence atom $\text{dep}(p; q)$ is FO-definable by

$$\forall t \forall t' ((A_{x_1}(t) \leftrightarrow A_{x_1}(t')) \rightarrow (A_{x_2}(t) \leftrightarrow A_{x_2}(t'))).$$

We call a TeamLTL formula extended by a generalised atom $\#$ a TeamLTL($\#$) formula. Similarly, we lift this notion and the corresponding decision problems to *sets of generalised atoms* $\mathcal{D}$, i.e., $\text{PC}(\text{TeamLTL}(\mathcal{D}))$ is the path-checking problem for TeamLTL formulae which may use the generalised atoms in $\mathcal{D}$.

Another way to extend TeamLTL is to introduce additional connectives. Here, the usual connectives to consider are the Boolean disjunction $\oplus$ and the contradictory negation $\sim$. A team T satisfies $\varphi \oplus \psi$ if it satisfies φ or ψ (or both). Contradictory negation combined with team semantics allows for powerful constructions. For instance, the complexity of model checking for propositional logic jumps from NC^1 to PSPACE [57], whereas the complexity of validity and satisfiability jumps all the way to alternating exponential time with polynomially many alternations ($\text{ATIME-ALT}(\exp, \text{pol})$) [36]. Formally, we define that $T \models \sim\varphi$ if $T \not\models \varphi$. Note that the contradictory

negation $\sim$ is not equivalent to the negation $\neg$ of atomic propositions defined earlier, i.e., $\sim p$ and $\neg p$ are not equivalent.

It turns out that the algorithm for PC(TeamLTL) (Algorithm 1) is very robust to strengthenings of the logic via the aforementioned constructs. The result of Theorem 4.3 can be extended to facilitate also the Boolean disjunction, contradictory negation and FO definable generalised atoms.

THEOREM 5.2. *Let $\mathcal{D}$ be a finite set of FO definable generalised atoms. PC(TeamLTL($\mathbb{Q}$, $\sim$, D)) is PSPACE-complete w.r.t. $\leq_m^P$ -reductions.*

PROOF. The lower bound applies from Theorem 4.3. For the upper bound, we extend the algorithm stated in the proof of Lemma 4.2 for the cases of the Boolean disjunction, contradictory negation and FO-definable atoms. The case for the Boolean disjunction is obtained by adding the following line to the recursive algorithm of the proof of Lemma 4.2 (where the latter $\mathbb{Q}$ denotes the classical meta-level disjunction):

if $\varphi = \psi \mathbb{Q} \psi'$ then return $\text{chk}(\mathcal{T}, \psi) \mathbb{Q} \text{chk}(\mathcal{T}, \psi')$.

The case for the contradictory negation is obtained by adding the following line to the recursive algorithm (where $\neg$ denotes the classical meta-level negation):

if $\varphi = \sim \varphi'$ then return $\neg \text{chk}(\mathcal{T}, \varphi')$.

Note that, since the extension is no longer a downward-closed logic, we need to modify the case for the splitjunction to reflect this. Hence we use the case:

if $\varphi = \psi \vee \psi'$ then return $\bigvee_{\mathcal{T}_1 \cup \mathcal{T}_2 = \mathcal{T}} \text{chk}(\mathcal{T}_1, \psi) \wedge \text{chk}(\mathcal{T}_2, \psi')$.

Finally, whenever a FO definable atom $\#$ appears in the computation of the algorithm, we need to solve an FO model-checking problem and classical LTL path-checking problems. As FO model checking is solvable in logarithmic space for any fixed formula [39] and LTL path checking can be done in NC (combined complexity) [46], the theorem follows. $\square$

The next proposition translates a result from Hannula et al. [36] to our setting. They show completeness for ATIME-ALT(exp, pol) for the satisfiability problem of propositional team logic with contradictory negation. This logic coincides with TeamLTL formulae without temporal operators.

PROPOSITION 5.3 ([36]). *SAT(TeamLTL($\sim$)) for formulae without temporal operators is complete for the class ATIME-ALT(exp, pol) w.r.t. $\leq_m^P$ -reductions.*

The result from the previous proposition will be utilised in the proof of the next theorem. It shows that already a very simple fragment of TeamLTL($\sim$) has an ATIME-ALT(exp, pol)-hard model-checking problem. Lück has established that the general problem is complete for third-order arithmetic [51].

THEOREM 5.4. *MC(TeamLTL($\sim$)) is ATIME-ALT(exp, pol)-hard w.r.t. $\leq_m^P$ -reductions.*

PROOF. We will state a reduction from the satisfiability problem of propositional team logic with the contradictory negation $\sim$ (short PL($\sim$)). The stated hardness then follows from Proposition 5.3.

For $P = \{p_1, \dots, p_n\}$, consider the traces starting from the root r of the Kripke structure $\mathcal{K}_P$ depicted in Figure 4 using proposition symbols $p_1, \dots, p_n, \bar{p}_1, \dots, \bar{p}_n$. Each trace in the model corresponds to a propositional assignment on P . For $\varphi \in \text{PL}(\sim)$, let φ^* denote the TeamLTL($\sim$) formula

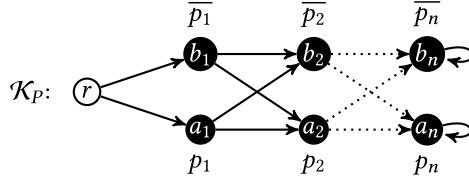


Fig. 4. Kripke structure for the proof of Theorem 5.4.

obtained by simultaneously replacing each (non-negated) variable p_i by Fp_i and each negated variable $\neg p_i$ by $F\bar{p}_i$. Let P denote the set of variables that occur in φ . Recall that we defined $\top := (p \vee \neg p)$ and $\perp := p \wedge \neg p$, then $T(\mathcal{K}_P) \models (\top \vee ((\sim \perp) \wedge \varphi^*))$ if and only if $T' \models \varphi^*$ for some non-empty $T' \subseteq T(\mathcal{K}_P)$. It is easy to check that $T' \models \varphi^*$ if and only if the propositional team corresponding to T' satisfies φ and thus the above holds if and only if φ is satisfiable. $\square$

In the following, problems of the form $\text{SAT}(\text{TeamLTL}(\text{dep}))$, and so on, refer to the corresponding problem for the extension of TeamLTL with dependence atoms. The following proposition follows from the corresponding result for classical LTL using downward closure and the fact that on singleton teams dependence atoms are trivially fulfilled.

PROPOSITION 5.5. $\text{SAT}(\text{TeamLTL}(\text{dep}))$ is PSPACE-complete w.r.t. $\leq_m^P$ -reductions.

The following result from Virtema talks about the validity problem of propositional team logic.

PROPOSITION 5.6 ([71]). *Validity of propositional logic with dependence atoms is NEXPTIME-complete w.r.t. $\leq_m^P$ -reductions.*

We use this result to obtain a lower bound on the complexity of $\text{MC}(\text{TeamLTL}(\text{dep}))$.

THEOREM 5.7. $\text{MC}(\text{TeamLTL}(\text{dep}))$ is NEXPTIME-hard w.r.t. $\leq_m^P$ -reductions.

PROOF. The proof of this result uses the same construction idea as in the proof of Theorem 5.4, but this time from a different problem, namely, validity of propositional logic with dependence atoms, which settles the lower bound by Proposition 5.6.

Let φ be a formula of propositional logic with dependence atoms and P be the set of proposition symbols that occur in φ . Let $\mathcal{K}_P$ be the Kripke structure depicted in Figure 4, which can be clearly constructed from φ in polynomial time. The idea here is that $T(\mathcal{K}_P)$ encodes the set all propositional assignments for P . Due to downward closure, the validity of propositional formulae with dependence atoms boils down to model checking the maximal team in the propositional (and not in the trace) setting, which is represented by $T(\mathcal{K}_P)$. Similar to the proof of Theorem 5.4, let φ^* denote the TeamLTL(dep) formula obtained from φ by simultaneously replacing each (non-negated) variable p_i by Fp_i and each negated variable $\neg p_i$ by $F\bar{p}_i$. Now, $T(\mathcal{K}_P) \models \varphi^*$, if and only if, the propositional team consisting of all assignments for P satisfies φ , if and only if, φ is valid (in the sense of Proposition 5.6). $\square$

6 TeamLTL vs. HyperLTL

TeamLTL expresses hyperproperties [17], that is, sets of teams, or equivalently, sets of sets of traces. HyperLTL [16], which extends LTL by trace quantification, is another logic expressing hyperproperties. For example, input determinism can be expressed as follows: every pair of traces that coincides on their input variables also coincides on their output variables (this can be expressed in TeamLTL by a dependence atom dep as sketched in Section 5). This results in a powerful formalism

that has vastly different properties than LTL [25]. After introducing the syntax and semantics of HyperLTL, we compare the expressive power of TeamLTL and HyperLTL.

The formulae of HyperLTL are given by the grammar

$$\varphi ::= \exists \pi. \varphi \mid \forall \pi. \varphi \mid \psi, \quad \psi ::= p_\pi \mid \neg \psi \mid \psi \vee \psi \mid X\psi \mid \psi U \psi,$$

where p ranges over atomic propositions in AP and where π ranges over a given countable set $\mathcal{V}$ of *trace variables*. The other Boolean connectives and the temporal operators release R, eventually F and always G are derived as usual, due to closure under negation. A sentence is a closed formula, i.e., one without free trace variables.

The semantics of HyperLTL is defined with respect to trace assignments, which are partial mappings $\Pi: \mathcal{V} \rightarrow (2^{\text{AP}})^\omega$. The assignment with empty domain is denoted by $\Pi_\emptyset$. Given a trace assignment Π , a trace variable π and a trace t , denote by $\Pi[\pi \rightarrow t]$ the assignment that coincides with Π everywhere but at π , which is mapped to t . Furthermore, $\Pi[i, \infty)$ denotes the assignment mapping every π in Π 's domain to $\Pi(\pi)[i, \infty)$. For teams T and trace-assignments Π we define the following²:

$$\begin{aligned} (T, \Pi) \models p_\pi & \quad \text{if } p \in \Pi(\pi)(0), \\ (T, \Pi) \models \neg \psi & \quad \text{if } (T, \Pi) \not\models \psi, \\ (T, \Pi) \models \psi_1 \vee \psi_2 & \quad \text{if } (T, \Pi) \models \psi_1 \text{ or } (T, \Pi) \models \psi_2, \\ (T, \Pi) \models X\psi & \quad \text{if } (T, \Pi[1, \infty)) \models \psi, \\ (T, \Pi) \models \psi_1 U \psi_2 & \quad \text{if } \exists k \geq 0 \text{ such that } (T, \Pi[k, \infty)) \models \psi_2 \text{ and} \\ & \quad \forall 0 \leq k' < k \text{ we have that } (T, \Pi[k', \infty)) \models \psi_1, \\ (T, \Pi) \models \exists \pi. \psi & \quad \text{if } \exists t \in T \text{ such that } (T, \Pi[\pi \rightarrow t]) \models \psi \text{ and} \\ (T, \Pi) \models \forall \pi. \psi & \quad \text{if } \forall t \in T \text{ we have that } (T, \Pi[\pi \rightarrow t]) \models \psi. \end{aligned}$$

We say that T satisfies a sentence φ if $(T, \Pi_\emptyset) \models \varphi$, and write $T \models \varphi$.

The semantics of HyperLTL is synchronous; this can be seen from how the until and next operators are defined. Hence, one could expect that HyperLTL is closely related to TeamLTL as defined here, which is synchronous as well. In the following, we refute this intuition. In fact, HyperLTL is closely related to the asynchronous variant of TeamLTL [41].

Formally, a HyperLTL sentence φ and a TeamLTL formula φ' are equivalent if for all teams T we have that $T \models \varphi$ if and only if $T \models \varphi'$.

THEOREM 6.1.

- (1) No TeamLTL formula is equivalent to $\exists \pi. p_\pi$.
- (2) No HyperLTL sentence is equivalent to the TeamLTL formula Fp .

PROOF. (1) Consider $T = \{\emptyset^\omega, \{p\}\emptyset^\omega\}$. We have $T \models \exists \pi. p_\pi$. Assume there is an equivalent TeamLTL formula, call it φ . Then, $T \models \varphi$ and thus $\{\emptyset^\omega\} \models \varphi$ by downward closure. Hence, by equivalence, $\{\emptyset^\omega\} \models \exists \pi. p_\pi$, yielding a contradiction.

(2) Bozzelli et al. proved that the property encoded by the TeamLTL formula Fp cannot be expressed in HyperLTL [12]. $\square$

Note that these separations are obtained by very simple formulae. Furthermore, the same separation hold for TeamLTL(dep), using the same arguments.

COROLLARY 6.2. HyperLTL and TeamLTL are of incomparable expressiveness.

²Note that we use the same $\models$ symbol for HyperLTL that we used for TeamLTL. It is clear from the context which semantics is used.

7 Team Semantics for CTL

After having presented team semantics for LTL, arguably the most important specification language for linear time properties, we now turn our attention to branching time properties. Here, we present team semantics for CTL.

Recall that in the LTL setting teams are sets of traces, as LTL is evaluated on a single trace. CTL is evaluated on a vertex of a Kripke structure. Consequently, teams in the CTL setting are (multi)sets of vertices. For the complexity results of this article, it does not really matter whether team semantics for CTL is defined with respect to multisets or plain sets. However, we have chosen multisets since the subsequent logic will be conceptually simpler than a variant based on sets. See the work of Kontinen et al. [42] for a development of asynchronous TeamLTL under set-based semantics. The challenges there to define the right set-based semantics for asynchronous TeamLTL are analogous to what would arise from the use of path quantifiers in CTL.

We will reuse some notions from LTL, e.g., the countably infinite set of propositions is AP. The set of all CTL formulae is defined inductively via the following grammar:

$$\varphi := p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \text{PX}\varphi \mid \text{P}[\varphi \text{U}\varphi] \mid \text{P}[\varphi \text{R}\varphi],$$

where $\text{P} \in \{\text{A}, \text{E}\}$ and $p \in \text{AP}$. We define the following usual shorthands: $\top := p \vee \neg p$, $\perp := p \wedge \neg p$, $\text{F}\varphi := [\top \text{U}\varphi]$ and $\text{G}\varphi := [\perp \text{R}\varphi]$. Note that the formulae are in NNF, which is the convention in team semantics. In the classical setting, this is not a severe restriction as transforming a given formula into its NNF requires linear time in the input length. In team semantics, the issue is more involved as here $\neg\varphi$ does not have an agreed compositional semantics and the contradictory negation often increases the complexity and expressiveness of a logic considerably, as was seen in Section 5.

We will again use Kripke structures as previously defined for LTL. When the initial world is not important, we will omit it from the tuple definition. By $\Pi(w)$, for a world w in a Kripke structure $\mathcal{K}$, we denote the (possibly infinite) set of all paths π for which $\pi(0) = w$. For a set $V \subseteq W$, we define $\Pi(V) := \bigcup_{w \in V} \Pi(w)$.

Definition 7.1. Let $\mathcal{K} = (W, R, \eta)$ be a Kripke structure and let $w \in W$ a world. The satisfaction relation $\models$ for CTL is defined as follows:

$$\begin{aligned} \mathcal{K}, w &\models p && \text{if } p \in \eta(w), \\ \mathcal{K}, w &\models \neg p && \text{if } p \notin \eta(w), \\ \mathcal{K}, w &\models \varphi \wedge \psi && \text{if } \mathcal{K}, w \models \varphi \text{ and } \mathcal{K}, w \models \psi, \\ \mathcal{K}, w &\models \varphi \vee \psi && \text{if } \mathcal{K}, w \models \varphi \text{ or } \mathcal{K}, w \models \psi, \\ \mathcal{K}, w &\models \text{EX}\varphi && \text{if } \exists \pi \in \Pi(w) \text{ such that } \mathcal{K}, \pi(1) \models \varphi, \\ \mathcal{K}, w &\models \text{E}[\varphi \text{U}\psi] && \text{if } \exists \pi \in \Pi(w) \exists k \in \mathbb{N} \text{ s.t. } \mathcal{K}, \pi(k) \models \psi \text{ and } \forall k' < k \text{ we have that } \mathcal{K}, \pi(k') \models \varphi, \\ \mathcal{K}, w &\models \text{E}[\varphi \text{R}\psi] && \text{if } \exists \pi \in \Pi(w) \forall k \in \mathbb{N} \text{ we have that } \mathcal{K}, \pi(k) \models \psi \text{ or } \exists k' < k \text{ s.t. } \mathcal{K}, \pi(k') \models \varphi, \\ \mathcal{K}, w &\models \text{AX}\varphi && \text{if } \forall \pi \in \Pi(w) \text{ we have that } \mathcal{K}, \pi(1) \models \varphi, \\ \mathcal{K}, w &\models \text{A}[\varphi \text{U}\psi] && \text{if } \forall \pi \in \Pi(w) \exists k \in \mathbb{N} \text{ s.t. } \mathcal{K}, \pi(k) \models \psi \text{ and } \forall k' < k \text{ we have that } \mathcal{K}, \pi(k') \models \varphi, \\ \mathcal{K}, w &\models \text{A}[\varphi \text{R}\psi] && \text{if } \forall \pi \in \Pi(w) \forall k \in \mathbb{N} \text{ we have that } \mathcal{K}, \pi(k) \models \psi \text{ or } \exists k' < k \text{ s.t. } \mathcal{K}, \pi(k') \models \varphi. \end{aligned}$$

Next, we will introduce team semantics for CTL based on multisets, obtaining the logic TeamCTL. Notice that, for TeamLTL, a team is a set of traces while here it is a multiset of worlds mimicking the behaviour of satisfaction in vanilla CTL. A *multiset* is a generalisation of the concept of a set that allows for having multiple instances of the same element. Technically, we define that a multiset is a functional set of pairs (i, x) , where i is an element of some sufficiently large set $\mathbb{I}$ of indices, and x is an element whose multiplicities we encode in the multiset. The *support* of a multiset M is the image set $M[\mathbb{I}] := \{x \mid (i, x) \in M, i \in \mathbb{I}\}$, while the *multiplicity* of an element x in a multiset M

is the cardinality of the pre-image $M^{-1}[\{x\}] := \{i \in \mathbb{I} \mid (i, x) \in M\}$. We often omit the index, and write simply x instead of (i, x) to simplify the notation. We also write $\{\{\dots\}\}$ to denote a multiset with its elements inside the curly brackets, omitting the indices. For multisets A and B , we write $A \uplus B := \{((i, 0), x) \mid (i, x) \in A\} \cup \{((i, 1), x) \mid (i, x) \in B\}$ to denote the disjoint union of A and B . Note that $A \cup B$ denotes the standard union of the sets A and B , which is not always a (functional) multiset. In the following, we introduce some notation for multisets.

Definition 7.2. Let A, B be two multisets over some index set $\mathbb{I}$. We write $A \overset{\#}{=} B$ if there exists a permutation $\pi: \mathbb{I} \rightarrow \mathbb{I}$ such that $A = \{(\pi(i), x) \mid (i, x) \in B\}$. Likewise, we write $A \subseteq B$ if there exists a permutation $\pi: \mathbb{I} \rightarrow \mathbb{I}$ such that $A \subseteq \{(\pi(i), x) \mid (i, x) \in B\}$.

Now, we turn to the definition of teams in the setting of CTL. Note that this notion is different from the notion of a team in the context of LTL, where a team is a possibly infinite set of traces.

Definition 7.3. Let $\mathcal{K} = (W, R, \eta)$ be a Kripke structure. Any multiset T , where $T[\mathbb{I}] \subseteq W$ is called a (multi)team of $\mathcal{K}$.

By abuse of notation, if the indices are irrelevant, we will write $w \in T$ instead of $(i, w) \in T$.

Let $\mathcal{K} = (W, R, \eta)$ be a Kripke structure and T be a team of $\mathcal{K}$. If $f: T \rightarrow \Pi(W)$ is a function that maps elements of the team to paths in W such that $f((i, x)) \in \Pi(x)$ for all $(i, x) \in T$ then we call f *T-compatible*, and define $T[f]$ to be the following multiset of traces

$$T[f] := \{(i, f((i, x))) \mid (i, x) \in T\}.$$

Let $n \in \mathbb{N}$. We define $T[f, n]$ to be the team of $\mathcal{K}$

$$T[f, n] := \{(i, t(n)) \mid (i, t) \in T[f]\},$$

that is, the multiset of worlds reached by synchronously proceeding to the n th element of each trace.

The syntax of TeamCTL is the same as for CTL above. Next, we define (synchronous) team semantics for CTL.

Definition 7.4. Let $\mathcal{K} = (W, R, \eta)$ be a Kripke structure, T be a team of $\mathcal{K}$ and φ, ψ be CTL formulae. The satisfaction relation $\models$ for TeamCTL is defined as follows.

$$\begin{aligned} \mathcal{K}, T \models p & \quad \text{if } \forall w \in T: p \in \eta(w). \\ \mathcal{K}, T \models \neg p & \quad \text{if } \forall w \in T: p \notin \eta(w). \\ \mathcal{K}, T \models \varphi \wedge \psi & \quad \text{if } \mathcal{K}, T \models \varphi \text{ and } \mathcal{K}, T \models \psi. \\ \mathcal{K}, T \models \varphi \vee \psi & \quad \text{if } \exists T_1 \cup T_2 = T \text{ s.t. } T_1 \cap T_2 = \emptyset, \mathcal{K}, T_1 \models \varphi \text{ and } \mathcal{K}, T_2 \models \psi. \\ \mathcal{K}, T \models \text{EX}\varphi & \quad \text{if } \exists T\text{-compatible } f \text{ s.t. } \mathcal{K}, T[f, 1] \models \varphi. \\ \mathcal{K}, T \models \text{E}[\varphi \cup \psi] & \quad \text{if } \exists T\text{-compatible } f \exists k \in \mathbb{N} \text{ s.t. } \mathcal{K}, T[f, k] \models \psi \text{ and } \forall k' < k: \mathcal{K}, T[f, k'] \models \varphi \\ \mathcal{K}, T \models \text{E}[\varphi \text{R} \psi] & \quad \text{if } \exists T\text{-compatible } f \forall k \in \mathbb{N}: \mathcal{K}, T[f, k] \models \psi \text{ or } \exists k' < k: \mathcal{K}, T[f, k'] \models \varphi \\ \mathcal{K}, T \models \text{AX}\varphi & \quad \text{if } \forall T\text{-compatible } f: \mathcal{K}, T[f, 1] \models \varphi. \\ \mathcal{K}, T \models \text{A}[\varphi \cup \psi] & \quad \text{if } \forall T\text{-compatible } f \exists k \in \mathbb{N} \text{ s.t. } \mathcal{K}, T[f, k] \models \psi \text{ and } \forall k' < k: \mathcal{K}, T[f, k'] \models \varphi. \\ \mathcal{K}, T \models \text{A}[\varphi \text{R} \psi] & \quad \text{if } \forall T\text{-compatible } f \forall k \in \mathbb{N}: \mathcal{K}, T[f, k] \models \psi \text{ or } \exists k' < k \text{ s.t. } \mathcal{K}, T[f, k'] \models \varphi. \end{aligned}$$

Basic Properties. In the following, we investigate several fundamental properties of the satisfaction relation. It is immediate that for every CTL formula φ and multiteams T and S such that $T \overset{\#}{=} S$, it holds that $\mathcal{K}, T \models \varphi$ if and only if $\mathcal{K}, S \models \varphi$. Notice that the following properties also hold for TeamLTL as defined in Section 3. Observe that $\mathcal{K}, T \models \perp$ if and only if $T = \emptyset$. The proof of the following proposition is trivial.

PROPOSITION 7.5 (EMPTY TEAM PROPERTY). *For every Kripke structure $\mathcal{K}$ we have that $\mathcal{K}, \emptyset \models \varphi$ holds for every CTL formula φ .*

When restricted to singleton teams, team semantics coincides with the traditional semantics of CTL defined via pointed Kripke structures.

PROPOSITION 7.6 (SINGLETON EQUIVALENCE). *For every Kripke structure $\mathcal{K} = (W, R, \eta)$, every world $w \in W$ and every CTL formula φ the following equivalence holds:*

$$\mathcal{K}, \{\{w\}\} \models \varphi \Leftrightarrow \mathcal{K}, w \models \varphi.$$

PROOF. Let $\mathcal{K} = (W, R, \eta)$ be an arbitrary Kripke structure. We prove the claim via induction on the structure of φ :

Assume that φ is a (negated) proposition symbol p . Now

$$\begin{aligned} \mathcal{K}, w &\models \varphi \\ \text{iff } & p \text{ is (not) in } \eta(w) \\ \text{iff } & \text{for all } w' \in \{\{w\}\} \text{ it holds that } p \text{ is (not) in } \eta(w') \\ \text{iff } & \mathcal{K}, \{\{w\}\} \models \varphi. \end{aligned}$$

The case $\wedge$ is trivial. For the $\vee$ case, assume that $\varphi = \psi \vee \theta$. Now it holds that

$$\begin{aligned} \mathcal{K}, w &\models \psi \vee \theta \\ \text{iff } & \mathcal{K}, w \models \psi \text{ or } \mathcal{K}, w \models \theta \\ \text{iff } & \mathcal{K}, \{\{w\}\} \models \psi \text{ or } \mathcal{K}, \{\{w\}\} \models \theta \\ \text{iff } & (\mathcal{K}, \{\{w\}\} \models \psi \text{ and } \mathcal{K}, \emptyset \models \theta) \text{ or } (\mathcal{K}, \emptyset \models \psi \text{ and } \mathcal{K}, \{\{w\}\} \models \theta) \\ \text{iff } & \exists T_1 \cup T_2 = \{\{w\}\} \text{ s.t. } T_1 \cap T_2 = \emptyset, \mathcal{K}, T_1 \models \psi \text{ and } \mathcal{K}, T_2 \models \theta \\ \text{iff } & \mathcal{K}, \{\{w\}\} \models \psi \vee \theta. \end{aligned}$$

Here, the first equivalence holds by the semantics of disjunction, the second equivalence follows by the induction hypothesis, the third via the empty team property, the fourth equivalence follows since there are only two ways to partition a singleton into an ordered pair of sets and the last by the team semantics of disjunction.

The cases for EX and AX, until and release are all similar and straightforward. We show here the case for EX. Assume $\varphi = \text{EX}\psi$. Then, $\mathcal{K}, w \models \text{EX}\psi$ iff there exists a trace $\pi \in \Pi(w)$ such that $\mathcal{K}, \pi(1) \models \psi$. Thus, by the induction hypothesis $\mathcal{K}, \pi(1) \models \psi$ iff $\mathcal{K}, \{\{\pi(1)\}\} \models \psi$. Hence, equivalently, there exists a $\{\{w\}\}$ -compatible function f such that $\mathcal{K}, \{\{w\}\}[f, 1] \models \psi$ which is equivalent to $\mathcal{K}, \{\{w\}\} \models \text{EX}\psi$. $\square$

TeamCTL is *downward closed* if the following holds for every Kripke structure $\mathcal{K}$, for every CTL formula φ and for all teams T and T' of $\mathcal{K}$:

$$\text{If } \mathcal{K}, T \models \varphi \text{ and } T' \sqsubseteq T \text{ then } \mathcal{K}, T' \models \varphi.$$

The proof of the following lemma is analogous to the corresponding proofs for modal and FO dependence logic (see [69, 70]).

PROPOSITION 7.7 (DOWNWARD CLOSURE). *TeamCTL is downward closed.*

PROOF. The proof is by induction on the structure of φ . Let $\mathcal{K} = (W, R, \eta)$ be an arbitrary Kripke structure and $T' \sqsubseteq T$ be some teams of $\mathcal{K}$. Hence, there is a bijection π witnessing $T' \sqsubseteq T$, i.e., it satisfies $T' \subseteq \{(\pi(i), x) \mid (i, x) \in T\}$.

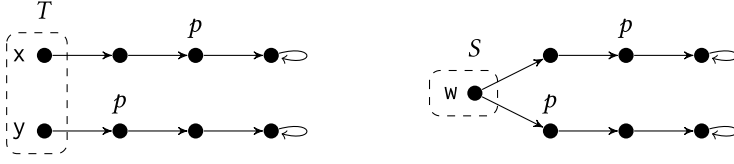


Fig. 5. (Left) A team T does not satisfy the TeamCTL formula EFp while all singleton teams $\{\{v\}\} \subseteq T$ individually do satisfy the formula. (Right) The multiplicity of elements in a team matters: If the multiplicity of the world w in S is 1 then S satisfies AFp , while for larger multiplicities S falsifies AFp .

The cases for literals are trivial: Assume $\mathcal{K}, T \models p$. Then by definition $p \in \eta(w)$ for every $w \in T$. Now since $T' \sqsubseteq T$, clearly $p \in \eta(w)$ for every $w \in T'$. Thus, we have that $\mathcal{K}, T' \models p$. The case for negated proposition symbols is identical.

The case for $\wedge$ is clear: For the case for $\varphi \vee \psi$ assume that $\mathcal{K}, T \models \varphi \vee \psi$. Now, by the definition of disjunction there exist $T_1 \cup T_2 = T$ such that $T_1 \cap T_2 = \emptyset$, $\mathcal{K}, T_1 \models \varphi$ and $\mathcal{K}, T_2 \models \psi$. Define $T'_1 = \{(\pi(i), x) \in T' \mid (i, x) \in T_1\}$ and T'_2 analogously. Then, we have $T'_1 \sqsubseteq T_1$ and $T'_2 \sqsubseteq T_2$ (the inclusions are both witnessed by π from above). Hence, by induction hypothesis it then follows that $\mathcal{K}, T'_1 \models \varphi$ and $\mathcal{K}, T'_2 \models \psi$. Finally, since $T' = T'_1 \cup T'_2$ and $T'_1 \cap T'_2 = \emptyset$, it follows by the semantics of the disjunction that $\mathcal{K}, T' \models \varphi \vee \psi$.

Now consider $EX\varphi$ and assume that $\mathcal{K}, T \models EX\varphi$. We have to show that $\mathcal{K}, T' \models EX\varphi$. Notice that by the semantics of EX there exists a T -compatible function f such that $\mathcal{K}, T[f, 1] \models \varphi$. Now, since $T' \sqsubseteq T$, the function $f \upharpoonright T'$ (i.e., the restriction of f to the domain of T') is T' -compatible. Consequently, by induction hypothesis, $\mathcal{K}, T'[f \upharpoonright T', 1] \models \varphi$ and thus $\mathcal{K}, T' \models EX\varphi$.

The proofs for the cases for all remaining operators are analogous. $\square$

Similar to TeamLTL, TeamCTL violates flatness and union closure. In this setting, a logic is *union closed* if it satisfies

$$\mathcal{K}, T \models \varphi \text{ and } \mathcal{K}, T' \models \varphi \text{ implies } \mathcal{K}, T \uplus T' \models \varphi,$$

and has the *flatness* property if

$$\mathcal{K}, T \models \varphi \text{ if and only if for all } w \in T \text{ it holds that } \mathcal{K}, \{\{w\}\} \models \varphi.$$

PROPOSITION 7.8. *CTL is neither union closed nor flat.*

PROOF. The counterexample for both is depicted in Figure 5 (left). $\square$

In addition, as exemplified in the right-hand side of Figure 5, TeamCTL is sensitive to multiplicities.

PROPOSITION 7.9. *Let T and T' be teams with the same support. In general, it need not hold that $\mathcal{K}, T \models \varphi$ if and only if $\mathcal{K}, T' \models \varphi$.*

Comparison to TeamCTL with tef. After the publication of the conference version of this work [44], Gutsfeld et al. [31] revisited the foundations of temporal team semantics and introduced several logics based on a concept they aptly named *tef*. Note first that, using a slightly different notation than in this article, the satisfying element in (synchronous) TeamLTL is a pair (T, i) , where T is a set of traces and $i \in \mathbb{N}$ is the current timestep. In this article, we write $T[i, \infty]$ instead of (T, i) . Hence, in synchronous TeamLTL, the traces share access to a common clock. Simply put, time evaluation functions $\tau: \mathbb{N} \times T \rightarrow \mathbb{N}$ model asynchronous evaluation of time on traces which are equipped with local clocks. The idea is that $\tau(i, t)$ denotes the value of the local clock of trace t at

global time i . Hence the satisfying element in this setting is a triplet (T, τ, n) , where T is a set of traces, τ is a time evaluation function and n is the value of the global clock. TeamLTL with tef's is then defined similarly to our TeamLTL, where logical operators modify T and n . In particular, if τ is the synchronous tef (i.e., $\tau(i, t) = i$ for all i and t), then exactly synchronous TeamLTL, as defined in this article, is obtained. Finally, they consider a logic they also named TeamCTL, which takes the syntax of CTL and reinterprets path quantifiers as quantifiers ranging over tef's. For further details, we refer to the work of Gutsfeld et al. [31].

The team semantics of Definition 7.4 for CTL can be modified to a team semantics for the full branching time logic CTL* by working with multisets of traces with time steps $(T[f], n)$ instead of teams $T[f, n]$. This approach is essentially equivalent to synchronous time evaluation functions as in the work of Gutsfeld et al. [31], when interpreted over computation trees.

8 Expressive Power of TeamCTL

Next, we study the expressiveness of TeamCTL and exemplify that some simple properties are not expressible in CTL with classical semantics. In order to relate the team semantics to the classical semantics, we first present a definition that lifts the classical semantics to multisets of worlds.

Definition 8.1. For every Kripke structure $\mathcal{K} = (W, R, \eta)$, every CTL formula φ and every team T of $\mathcal{K}$, we let

$$\mathcal{K}, T \models^c \varphi \text{ if and only if } \forall w \in T : \mathcal{K}, w \models \varphi.$$

We refer to this notion as *classical (multiset) semantics*.

Observe that classical multiset semantics is flat by definition (i.e., $\mathcal{K}, T \models^c \varphi$ if and only if for all $w \in T$ we have that $\mathcal{K}, \{\{w\}\} \models^c \varphi$), and hence union and downward closed. In the following, we introduce the notions of definability and k -definability.

Definition 8.2. For each CTL formula φ , define

$$\begin{aligned} \mathfrak{F}_\varphi^c &:= \{(\mathcal{K}, T) \mid \mathcal{K}, T \models^c \varphi \text{ under classical multiset semantics}\} \text{ and} \\ \mathfrak{F}_\varphi^t &:= \{(\mathcal{K}, T) \mid \mathcal{K}, T \models \varphi \text{ under team semantics}\}. \end{aligned}$$

We say that φ *defines the class* $\mathfrak{F}_\varphi^c$ *in classical multiset semantics*. Analogously, we say that φ *defines the class* $\mathfrak{F}_\varphi^t$ *in team semantics*.

A class $\mathfrak{F}$ of pairs of Kripke structures and teams is *definable in classical multiset semantics* (in team semantics) if there exists some TeamCTL formula ψ such that $\mathfrak{F} = \mathfrak{F}_\psi^c$ ($\mathfrak{F} = \mathfrak{F}_\psi^t$). Furthermore, for $k \in \mathbb{N}$, define

$$\mathfrak{F}_\varphi^{c,k} := \{(\mathcal{K}, T) \mid \mathcal{K}, T \models^c \varphi \text{ and } |T| \leq k\} \text{ and } \mathfrak{F}_\varphi^{t,k} := \{(\mathcal{K}, T) \mid \mathcal{K}, T \models \varphi \text{ and } |T| \leq k\}.$$

We say that φ *k-defines the class* $\mathfrak{F}_\varphi^{c,k}$ ($\mathfrak{F}_\varphi^{t,k}$, *resp.*) *in classical multiset (team, resp.) semantics*. The definition of *k-definability* is analogous to that of definability.

Next we will show that there exists a class $\mathfrak{F}$ which is definable in team semantics, but is not definable in classical multiset semantics.

THEOREM 8.3. *The class $\mathfrak{F}_{\text{EFp}}^t$ is not definable in classical multiset semantics.*

PROOF. For the sake of a contradiction, assume that there is a CTL formula φ such that $\mathfrak{F}_\varphi^c = \mathfrak{F}_{\text{EFp}}^t$. Consider the Kripke structure from Figure 5 (left). Clearly $\mathcal{K}, \{\{x\}\} \models \text{EFp}$ and $\mathcal{K}, \{\{y\}\} \models \text{EFp}$. Thus, by our assumption, it follows that $\mathcal{K}, \{\{x\}\} \models^c \varphi$ and $\mathcal{K}, \{\{y\}\} \models^c \varphi$. From Definition 8.1, it then follows that $\mathcal{K}, \{\{x, y\}\} \models^c \varphi$. But clearly $\mathcal{K}, \{\{x, y\}\} \not\models \text{EFp}$. $\square$

COROLLARY 8.4. *For $k > 1$, the class $\mathfrak{F}_{\text{EFp}}^{t,k}$ is not k -definable in classical multiset semantics.*

We conjecture that the class $\mathfrak{F}_{\text{EFp}}^c$ is not definable in team semantics. However, we will see that the class $\mathfrak{F}_{\text{EFp}}^{c,k}$ is k -definable in team semantics.

THEOREM 8.5. *For every $k \in \mathbb{N}$ and every CTL formula φ , the class $\mathfrak{F}_{\varphi}^{c,k}$ is k -definable in team semantics.*

PROOF. Fix $k \in \mathbb{N}$ and a CTL formula φ . We define

$$\varphi' := \bigvee_{1 \leq i \leq k} \varphi.$$

We will show that $\mathfrak{F}_{\varphi}^{c,k} = \mathfrak{F}_{\varphi'}^{t,k}$. Let $\mathcal{K}$ be an arbitrary Kripke structure and T be a team of $\mathcal{K}$ of size at most k . Then the following is true

$$\begin{aligned} \mathcal{K}, T \models^c \varphi &\Leftrightarrow \forall w \in T : \mathcal{K}, w \models \varphi \\ &\Leftrightarrow \forall w \in T : \mathcal{K}, \{\{w\}\} \models \varphi \\ &\Leftrightarrow \mathcal{K}, T \models \varphi'. \end{aligned}$$

The first equivalence follows by Definition 8.1, the second is due to Proposition 7.6 and the last by the semantics of disjunction, empty team property and downward closure. $\square$

9 Complexity Results for TeamCTL

Next, we define the most important decision problems for TeamCTL and classify their computational complexity. Notice that, in comparison to TeamLTL, the problems are defined in a way that a team is explicitly given as part of the input, while for TeamLTL, the team is given implicitly by a Kripke structure.

Problem:	MC(TeamCTL)—TeamCTL Model Checking.
Input:	A Kripke structure $\mathcal{K}$, a finite team T of $\mathcal{K}$, and a TeamCTL formula φ .
Question:	$\mathcal{K}, T \models \varphi$?

Problem:	SAT(TeamCTL)—TeamCTL Satisfiability.
Input:	A TeamCTL formula φ .
Question:	Does there exist a Kripke structure $\mathcal{K}$ and a non-empty team T of $\mathcal{K}$ such that $\mathcal{K}, T \models \varphi$?

9.1 Model Checking

We first investigate the computational complexity of model checking. Our benchmark here is the complexity of model checking for classical CTL.

PROPOSITION 9.1 ([15, 64]). *Model checking for CTL formulae under classical semantics is P-complete.*

Now we turn to the model-checking problem for TeamCTL. Here we show that the problem becomes intractable under reasonable complexity class separation assumptions, i.e., $P \neq PSPACE$. The main idea is to exploit the synchronicity of the team semantics in a way to check in parallel all clauses of a given QBF for satisfiability by a set of relevant assignments.

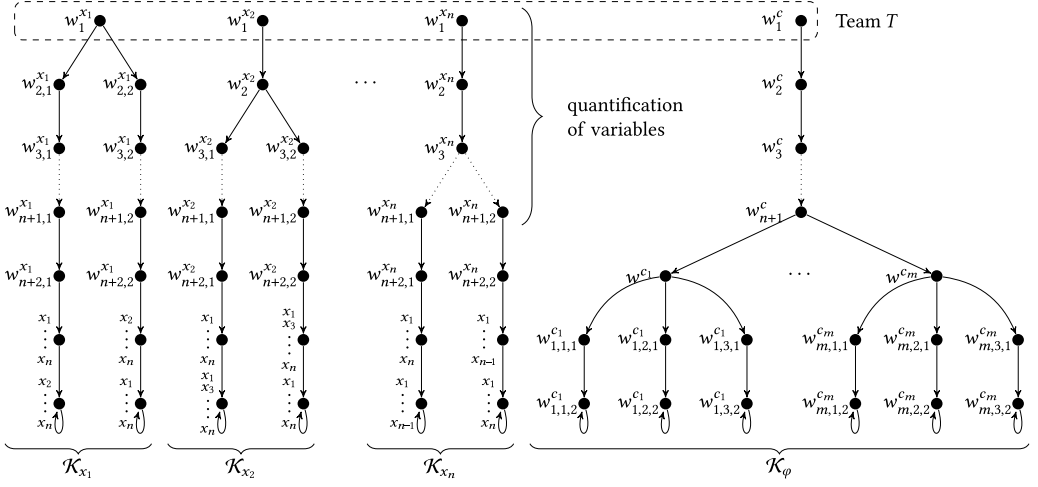


Fig. 6. General view of the created Kripke structure in the proof of Lemma 9.2. For $\mathcal{K}_{x_i}$ choosing the left path of the structure corresponds to setting x_i to 1, choosing the right path to setting x_i to 0. Synchronicity of the variable Kripke structures $\mathcal{K}_{x_i}$ and the structure $\mathcal{K}_\varphi$ ensures choosing consistent values for the x_i 's while satisfying all clauses.

LEMMA 9.2. $\text{MC}(\text{TeamCTL})$ is PSPACE-hard w.r.t. $\leq_m^P$ -reductions.

PROOF. We will reduce from QBF-VAL. Let $\varphi := \exists x_1 \forall x_2 \dots Q x_n \wedge \bigvee_{i=1}^m \ell_{j,i}$ be a closed quantified Boolean formula (QBF) and $Q = \exists$ if n is odd, resp., $Q = \forall$ if n is even.

Now define the corresponding structure $\mathcal{K} := (W, R, \eta)$ over the set $\text{AP} = \{x_1, \dots, x_n\}$ of atomic propositions as follows (the structure is given in Figure 6). The structure $\mathcal{K}$ is constructed out of smaller structures in a modular way. Formally, if $\mathcal{K}_1 = (W_1, R_1, \eta_1)$ and $\mathcal{K}_2 = (W_2, R_2, \eta_2)$ are two Kripke structures, then let $\mathcal{K}_1 \cup \mathcal{K}_2$ be defined as the Kripke structure $(W_1 \cup W_2, R_1 \cup R_2, \eta_1 \cup \eta_2)$. Without loss of generality, we will always assume that $W_1 \cap W_2 = \emptyset$, $R_1 \cap R_2 = \emptyset$ and $\eta_1 \cap \eta_2 = \emptyset$. For each x_i define a Kripke structure $\mathcal{K}_{x_i} := (W_{x_i}, R_{x_i}, \eta_{x_i})$, where

$$\begin{aligned} W_{x_i} &:= \{w_1^{x_i}, \dots, w_i^{x_i}\} \cup \{w_{j,1}^{x_i}, w_{j,2}^{x_i} \mid i < j \leq n+4\}, \\ R_{x_i} &:= \{(w_j^{x_i}, w_{j+1}^{x_i}) \mid 1 \leq j < i\} \cup \{(w_i^{x_i}, w_{i+1,a}^{x_i}) \mid a \in \{1, 2\}\} \cup \\ &\quad \cup \{(w_{j,a}^{x_i}, w_{j+1,a}^{x_i}) \mid a \in \{1, 2\}, i < j < n+4\} \cup \{(w_{n+4,a}^{x_i}, w_{n+4,a}^{x_i}) \mid a \in \{1, 2\}\}, \\ \eta_{x_i} &:= \{(w, \{x_1, \dots, x_n\}) \mid w \in \{w_{n+3,1}^{x_i}, w_{n+4,2}^{x_i}\}\} \cup \{(w, \{x_1, \dots, x_n\} \setminus \{x_i\}) \mid \\ &\quad w \in \{w_{n+4,1}^{x_i}, w_{n+3,2}^{x_i}\}\}, \end{aligned}$$

and for all other not yet defined $\eta_{x_i}(w)$ with $w \in W_{x_i}$, we let $\eta_{x_i}(w) = \emptyset$.

If ℓ is a literal, then we will write $\mathcal{V}(\ell)$ to denote the corresponding variable of ℓ . Furthermore, we define a Kripke structure $\mathcal{K}_\varphi := (W_\varphi, R_\varphi, \eta_\varphi)$, where

$$\begin{aligned} W_\varphi &:= \{w_i^c \mid 1 \leq i \leq n+1\} \cup \{w_j^c \mid 1 \leq j \leq m\} \cup \{w_{j,i,k}^{c_j} \mid 1 \leq j \leq m, 1 \leq i \leq 3, 1 \leq k \leq 2\}, \\ R_\varphi &:= \{(w_i^c, w_{i+1}^c) \mid 1 \leq i \leq n\} \cup \{(w_{n+1}^c, w_j^c) \mid 1 \leq j \leq m\} \cup \\ &\quad \cup \{(w_j^c, w_{j,i,1}^{c_j}), (w_{j,i,1}^{c_j}, w_{j,i,2}^{c_j}) \mid 1 \leq i \leq 3, 1 \leq j \leq m\} \cup \\ &\quad \cup \{(w_{j,i,2}^{c_j}, w_{j,i,2}^{c_j}) \mid 1 \leq i \leq 3, 1 \leq j \leq m\}, \text{ and} \\ \eta_\varphi &:= \{(w_{j,i,1}^{c_j}, \{x_k \mid \ell_{j,i} = x_k\}) \cup \{x_k \mid x_k \neq \mathcal{V}(\ell_{j,i})\} \mid 1 \leq j \leq m, 1 \leq i \leq 3\} \cup \end{aligned}$$

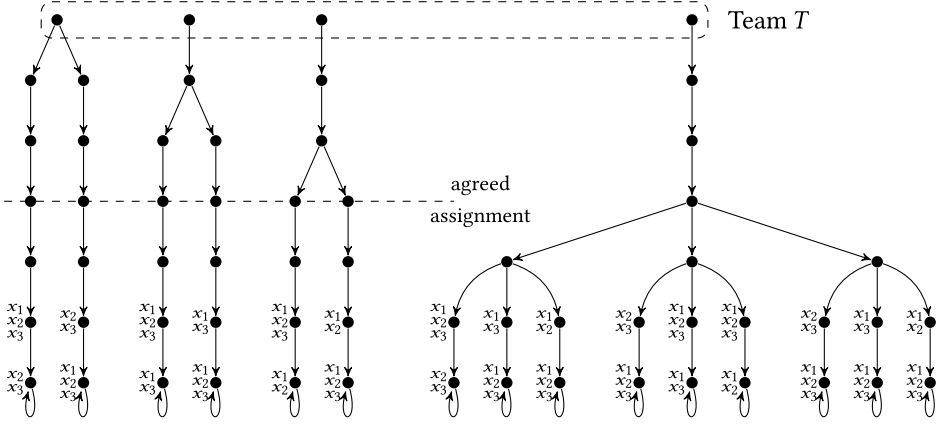


Fig. 7. Example structure built in proof of Lemma 9.2 for the QBF $\exists x_1 \forall x_2 \exists x_3 (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$.

$$\cup \{ \{w_{j,i,2}^{c_j}, \{x_k \mid \ell_{j,i} = \neg x_k\} \cup \{x_k \mid x_k \neq \mathcal{V}(\ell_{j,i})\} \} \mid 1 \leq j \leq m, 1 \leq i \leq 3 \},$$

and for all other not yet defined $\eta_\varphi(w)$ for $w \in W_\varphi$, we let $\eta_\varphi(w) = \emptyset$.

Finally, let $\mathcal{K} = (W, R, \eta)$ be the Kripke structure defined as

$$\bigcup_{1 \leq i \leq n} \mathcal{K}_{x_i} \cup \mathcal{K}_\varphi.$$

Furthermore, set

$$T := \{ \{w_1^{x_1}, \dots, w_1^{x_n}, w_1^c\} \} \text{ and } \psi := \underbrace{\text{EXAX} \dots \text{PX}}_n \text{AXEX} \bigwedge_{i=1}^n \text{EF}x_i,$$

where $P = E$ if n is odd and $P = A$ if n is even. We define the reduction as $f: \langle \varphi \rangle \mapsto \langle \mathcal{K}, T, \psi \rangle$.

Figure 7 shows an example of the reduction for the instance

$$\exists x_1 \forall x_2 \exists x_3 (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3).$$

Note that this formula is a valid QBF and hence belongs to QBF-VAL. The left three branching systems choose the values of the x_i 's. A decision for the left (right, resp.) path should be understood as setting the variable x_i to the truth value 1 (0, resp.).

For the correctness of the reduction, we need to show that $\varphi \in \text{QBF-VAL}$ iff $f(\varphi) \in \text{MC}(\text{TeamCTL})$.

' $\Rightarrow$ ': Let $\varphi \in \text{QBF-VAL}$, $\varphi = \exists x_1 \forall x_2 \dots Qx_n \chi$, where $\chi = \bigwedge_{j=1}^m \bigvee_{i=1}^3 \ell_{j,i}$. Thus, there exists a subtree S of the full binary assignment tree on the variable set, obtained by encoding the existential choices of the quantifier prefix $\exists x_1 \forall x_2 \dots Qx_n$, such that $s \models \chi$ for every leaf s in the subtree S . The tree S specifies the choices for each existential variable depending on the preceding universal choices.

Now we will prove that $\mathcal{K}, T \models \psi$. For w_1^c there is no choice in the next n steps defined by the prefix of φ . For $w_1^{x_1}, \dots, w_1^{x_n}$, after n steps for each i one successor of $w_{n+1,1}^{x_i}$ or $w_{n+1,2}^{x_i}$ is determined depending on the subtree S and the to x_i related path to a leaf of the assignment subtree in S , i.e., the corresponding assignment s tells which value is assigned to x_i : if x_i is mapped to 1 the choice is the successor world $w_{i+1,1}^{x_i}$; if x_i is mapped to 0 the choice is $w_{i+1,2}^{x_i}$ instead.

Now fix an arbitrary path in the subtree S which specifies a particular leaf s . Let us write $s(x)$ for the value of x in the leaf s . Furthermore, recall that $s \models \chi$, so all clauses in χ are satisfied

by s . Let us start with the evaluation of $\mathcal{K}, T \models \psi$. After n steps, the current team T' then is $\{\{w_{n+1}^c\}\} \cup \{\{w_{n+1,1}^{x_i} \mid s(x_i) = 1, 1 \leq i \leq n\}\} \cup \{\{w_{n+1,2}^{x_i} \mid s(x_i) = 0, 1 \leq i \leq n\}\}$ (note that now the team agrees with the assignment in s). In the next step, an AX in ψ has to be evaluated. Hence, from any T' -compatible function f the successor team $T'[f, 1]$ has to be considered. Choose an arbitrary such f which maps w_{n+1}^c to some $w_{n+2}^{c_j}$ for $1 \leq j \leq m$ (meaning that any clause in χ has to be satisfied) while for the $\mathcal{K}_{x_i}$ -vertices one proceeds one step to $w_{n+2,k}^{x_i}$ depending on the respective $k \in \{1, 2\}$. As a result, $T'[f, 1]$ is $\{\{w_{n+2}^{c_j}\}\} \cup \{\{w_{n+2,1}^{x_i} \mid s(x_i) = 1, 1 \leq i \leq n\}\} \cup \{\{w_{n+2,2}^{x_i} \mid s(x_i) = 0, 1 \leq i \leq n\}\}$. Now continuing with an EX in ψ , the team member $w_{n+2}^{c_j}$ has to select a compatible function f which chooses some successor world. This choice is made according to the assignment s under consideration. From $s \models \chi$, we know that for the j th clause C_j , we have that $s \models C_j$, hence, at least one literal is satisfied in that clause. W.l.o.g. assume that in clause C_j the literal $\ell_{j,r}$ (for some $r \in \{1, 2, 3\}$) is satisfied by s , i.e., $s \models \ell_{j,r}$. Then, the function f maps to the world $w_{n+3,1}^{c_j}$ as a successor from $w_{n+2}^{c_j}$. For the $\mathcal{K}_{x_i}$ vertices $w_{n+2,k}^{x_i}$ with $k \in \{1, 2\}$, we again have no choice and proceed to $w_{n+3,k}^{x_i}$.

Now, the remainder of ψ which is $\bigwedge_{i=1}^n \text{EF}x_i$ has to be satisfied. For the purpose of the proceeding proof, fix an $1 \leq i \leq n$ and consider each case $\text{EF}x_i$ separately. Observe that for the variable team members $w_{n+3,1}^{x_i}$ (i.e., those corresponding to variables that are set to 1 by s) all variables are in the label of the current world, but in the successor world $w_{n+4,1}^{x_i}$ the variable x_i is not in the label, i.e., $x_i \notin \eta(w_{n+4,1}^{x_i})$. The opposite is true for the $w_{n+3,2}^{x_i}$ worlds (i.e., those corresponding to variables that are set to 0 by s), where $x_i \notin \eta(w_{n+3,2}^{x_i})$ but all variables are in the label of the successor world, i.e., $x_i \in \eta(w_{n+4,2}^{x_i})$. Depending on what kind of literal $\ell_{j,r}$ is, the corresponding variable $x_{i'}$ is either in the label of the world $w_{n+3,1}^{c_j}$ or in the one of the world $w_{n+3,2}^{c_j}$. First, assume that $i = i'$. If $\ell_{j,r} = x_i$, then $x_i \in \eta(w_{n+3,1}^{c_j})$ and if $\ell_{j,r} = \neg x_i$, then $x_i \in \eta(w_{n+3,2}^{c_j})$. As $s \models \ell_{j,r}$, it follows that if $\ell_{j,r} = x_i$, then $s(x_i) = 1$ and hence $x_i \in \eta(w_{n+3,1}^{c_j})$ and $x_i \in \eta(w_{n+3,1}^{x_i})$ (similarly, if $\ell_{j,r} = \neg x_i$, then $s(x_i) = 0$ and hence $x_i \in \eta(w_{n+3,2}^{c_j})$ and $x_i \in \eta(w_{n+3,2}^{x_i})$). Thus, in the first case, $\text{EF}x_i$ is satisfied by choosing the k in the EF to be 0 and thereby proceeding in $w_{n+3,1}^{c_j}$, resp., in $w_{n+3,1}^{x_i}$, i.e., $T[f, 0]$ is chosen to satisfy x_i . This is correct, as in this case x_i is in the label of these worlds. For the second case, the choice is $T[f, 1]$, hence having $k = 1$ in the EF-evaluation, and this is also correct, as in this case x_i is in the label of the respective successor worlds $w_{n+4,2}^{x_i}$ and $w_{n+3,2}^{c_j}$. In the other case, $i \neq i'$, it is irrelevant for the satisfaction of $\text{EF}x_i$ as $x_{i'}$ is in the labels of both worlds.

As we considered an arbitrary path in S , and arbitrary functions f for the branching choices from w_{n+1}^c , it follows that $\mathcal{K}, T \models \psi$.

‘ $\Leftarrow$ ’: Let $\mathcal{K}, T \models \psi$. Then, by the semantics of ψ , there is a sequence of n choices for the EX and AX operators in the prefix of ψ such that for any choice of the AX operators, the formula $\text{AXEX} \bigwedge_{i=1}^n \text{EF}x_i$ is satisfied in the end. Let us denote by a binary vector $s \in \{0, 1\}^n$ a possible sequence of choices for the EX/AX operators in the prefix of ψ . To be more precise, if the i th choice is to proceed to $w_{i+1,1}^{x_i}$, then set the i th bit of s to $s_i = 1$ and otherwise (where one proceeds to $w_{i+1,2}^{x_i}$) to $s_i = 0$. Irrespectively how the choices for the AX operators are made, in the end $\text{AXEX} \bigwedge_{i=1}^n \text{EF}x_i$ has to be satisfied by assumption.

The next AX-choice considers all $w_{n+2}^{c_j}$ for $1 \leq j \leq m$. Then, the next EX-choice considers a $w_{n+3,1}^{c_j}$ for $1 \leq r \leq 3$. Finally, the formula $\bigwedge_{i=1}^n \text{EF}x_i$ has to be satisfied. These n EF-choices have to be made consistently with respect to the states $w_{n+3,1}^{x_i}$ and $w_{n+3,2}^{x_i}$ for $1 \leq i \leq n$ (which have been decided before already by the choices encoded in the vector s). Here, consistent means that choices that lead to $w_{n+3,1}^{x_i}$ have to choose $k = 0$ in the EF operator and those that lead to $w_{n+3,2}^{x_i}$ have to choose $k = 1$ in the EF operator. This is due to the labelling of these worlds. From s we can construct

Algorithm 2: Algorithm for EG-PathSearch Procedure

```

1 Procedure EG-PathSearch(Kripke structure  $\mathcal{K}$ , team  $T_1$ , team  $T_2$  with  $|T_1| = |T_2|$ , formula  $\varphi$ , integer  $c$ );
2 if  $c = 0$  then return  $(T_1 \stackrel{\#}{=} T_2) \wedge \text{MC}(\mathcal{K}, T_1, \varphi)$ ;
3 if  $c = 1$  then return  $(T_2 \text{ is a successor team of } T_1) \wedge \text{MC}(\mathcal{K}, T_1, \varphi) \wedge \text{MC}(\mathcal{K}, T_2, \varphi)$ ;
4 existentially branch on all teams  $T_{\text{mid}}$  of  $\mathcal{K}$  with  $|T_{\text{mid}}| = |T_1| = |T_2|$  and
   return  $\text{EG-PathSearch}(\mathcal{K}, T_1, T_{\text{mid}}, \varphi, \lfloor c/2 \rfloor) \wedge \text{EG-PathSearch}(\mathcal{K}, T_{\text{mid}}, T_2, \varphi, \lceil c/2 \rceil)$ 

```

an assignment s' in the obvious way: $s'(x_i) = s_i$ for all $1 \leq i \leq n$. This assignment s' is indeed well-defined. Moreover, depending on the $w_{j,r,1}^{c_j}$ (for $1 \leq r \leq 3$) that has been decided, we can deduce that the corresponding literal $\ell_{j,r}$ is satisfied by s' . As $\bigwedge_{i=1}^n \text{EF}x_i$ is satisfied, irrespective on the choice of w^{c_j} , we can deduce that $s' \models \bigvee_{j=1}^3 \ell_{i,j}$ for all $1 \leq j \leq m$. From this we can conclude that $s' \models \chi$, and as s was determined according to the EX/AX-prefix of length n of ψ , finally, $\varphi \in \text{QBF-VAL}$. $\square$

Now, we will turn towards proving the PSPACE upper bound for $\text{MC}(\text{TeamCTL})$. Before, we will need a definition and two auxiliary lemmas. Given two teams T_1, T_2 of a Kripke structure $\mathcal{K}$, we say T_2 is a *successor team* of T_1 if there exists a T_1 -compatible function f such that $T_2 \stackrel{\#}{=} T_1[f, 1]$.

LEMMA 9.3. *Let $\mathcal{K}$ be a Kripke structure and T be a team of $\mathcal{K}$. Furthermore, let $T_1, T_2, \dots, T_m$ be a sequence of teams such that $T_1 = T$ and such that T_{i+1} is a successor team of T_i for all i . Then the largest m such that there exists no $1 \leq i \neq j \leq m$ with $T_i \stackrel{\#}{=} T_j$ is bounded from above by $|W|^{|T|}$. In particular, if there is such a sequence of length $|W|^{|T|} + 1$, then there is an infinite sequence of this kind.*

PROOF. The number of teams of $\mathcal{K}$ with cardinality $|T|$ is bounded by the number of functions of the kind $f: T \rightarrow W$. As $|W|^{|T|}$ is the number of all functions from T to W , the claim follows. $\square$

The following lemma shows that the successor relation over teams can be decided in polynomial time. Here, we measure the size of a team by the cardinality of the multiset and the size of the Kripke structure by the number of its worlds.

LEMMA 9.4. *The question whether a given team T_2 is a successor team of a given team T_1 (w.r.t. a given Kripke structure $\mathcal{K}$) can be decided in polynomial time in $|T_1| + |T_2| + |\mathcal{K}|$.*

PROOF. Let $T_1 = \{t_1, \dots, t_n\}$ and $T_2 = \{t'_1, \dots, t'_n\}$. Then, T_2 is a successor team of T_1 if and only if the following is true

- (1) $|T_1| = |T_2|$,
- (2) for all $w \in T_1$ there exists a $w' \in T_2$ such that wRw' and
- (3) for all $w \in T_2$ there exists a $w' \in T_1$ such that $w'Rw$.

The first two items alone do not suffice because some z in T_2 could be reached from two different y 's in T_1 . The last item ensures that there is no x in T_2 that has no R -predecessor in T_1 .

The first item can be checked in time $O(|T_1|)$. The second and third item can both be checked in polynomial time in $|T_1| \cdot |T_2| \cdot |\mathcal{K}|$. Together this is polynomial time in the input length. $\square$

LEMMA 9.5. $\text{MC}(\text{TeamCTL})$ is in PSPACE.

PROOF. We construct an algorithm that runs in alternating polynomial time. As $\text{APTIME} = \text{PSPACE}$ [14], this proves the desired upper bound.

Algorithm 3: Algorithm for AF-PathSearch Procedure

```

1 Procedure AF-PathSearch(Kripke structure  $\mathcal{K}$ , team  $T_1$ , team  $T_2$  with  $|T_1| = |T_2|$ , formula  $\varphi$ , integer  $c$ );
2 if  $c = 0$  then return  $(T_1 \not\equiv T_2) \otimes \text{MC}(\mathcal{K}, T_1, \varphi)$ ;
3 if  $c = 1$  then return  $(T_2 \text{ is not a successor team of } T_1) \otimes \text{MC}(\mathcal{K}, T_1, \varphi) \otimes \text{MC}(\mathcal{K}, T_2, \varphi)$ ;
4 universally branch on all teams  $T_{\text{mid}}$  of  $\mathcal{K}$  with  $|T_{\text{mid}}| = |T_1| = |T_2|$  and
   return  $\text{AF-PathSearch}(\mathcal{K}, T_1, T_{\text{mid}}, \varphi, \lfloor c/2 \rfloor) \otimes \text{AF-PathSearch}(\mathcal{K}, T_{\text{mid}}, T_2, \varphi, \lceil c/2 \rceil)$ 

```

Before we come to the model-checking algorithm, we want to shortly discuss two subroutines that are used in it. The first subroutine (depicted in Algorithm 2) is used in the cases for the existential binary temporal operators, and also makes use of the main algorithm MC (depicted in Algorithm 4): The procedure EG-PathSearch will recursively determine whether there exists a path of length c between two given teams such that a given formula φ is satisfied at each team of the path. Intuitively, the procedure works similarly to the path search method in the proof of Savitch's Theorem [63] (also, see Sipser's textbook [65, Section 8.1]). It will be called for path lengths c that are bounded from above by $|W|^{|T|} + 1$ (Lemma 9.3). As every recursion halves the path length, the recursion depth is bounded by $O(\log(|W|^{|T|})) = O(|T| \log |W|)$ and hence is polynomial in the input length.

The second subroutine (depicted in Algorithm 3) is used for the universal binary temporal connectives, and also makes use of the main algorithm MC. Intuitively, it is the dual of Algorithm 2, i.e., it will recursively determine whether all paths of length c between two given teams are such that a given formula φ is satisfied at at least one team of the path. Thus, the subroutine will reject if and only if there is a path of length c between the two teams such that at all teams of this path φ is *falsified*. Again, it will be called for path lengths c that are bounded from above by $|W|^{|T|} + 1$, which implies that the recursion depth is polynomial in the input length.

Next, we define the main algorithm. Let φ be a TeamCTL formula, $\mathcal{K}$ be a Kripke structure and T be a team of $\mathcal{K}$. Given these, the algorithmic call $\text{MC}(\mathcal{K}, T, \varphi)$ returns true if and only if $\mathcal{K}, T \models \varphi$. The algorithm is depicted in Algorithm 4.

Intuitively, the algorithm is a recursive alternating tableaux algorithm that branches on the operators of the subformulae of φ . For the existential temporal operators, we are searching (using nondeterminism) for paths witnessing the satisfaction, while for the universal temporal operators, we are searching (using universal branching) for the absence of paths witnessing violation of the formula. Note that we do not allow classical negation in our logic, hence, we use duality of ATMs to handle negation.

Observe that the recursion depth of the main algorithm is bounded by the number of subformulae; therefore, it is linear in the input length. The recursion depth of the calls to the subroutines is also bounded polynomially, as argued above. Also note that the size of each guessed value is polynomial in the input size; for the value of c , notice that it is encoded in binary.

As for the correctness of the algorithm, each step directly implements the corresponding semantics. Several times we rely on the fact that the successor team check is in P for the guessed teams (see Lemma 9.4), i.e., in lines 9, 12, 18 and 42. All cases but the four for U and R are straightforward applications of the semantics, so let us focus on the latter.

We begin our explanations with the existential variants. Here, both the U and R cases make use of Lemma 9.3 and thereby restrict the path length to an exponential value: if there is a path witnessing satisfaction of a subformula, then there is also one of at most exponential length. To find such paths, we rely on Algorithm 2.

For the U case, the algorithm checks whether β holds directly at T (line 16) or if there are teams T_{end} and $T_{\text{end}-1}$ such that T_{end} is a successor of $T_{\text{end}-1}$, β holds in T_{end} , and there is a path from T to

Algorithm 4: Algorithm for MC(TeamCTL)

```

1 Procedure MC(Kripke structure  $\mathcal{K}$ , team  $T$ , formula  $\varphi$ );
2 if  $\varphi = p$  (resp.,  $\varphi = \neg p$ ) and  $p \in \text{AP}$  then
3   if for all  $w \in T$  we have that  $p \in \eta(w)$  (resp.,  $p \notin \eta(w)$ ) then return true else return false;
4 if  $\varphi = \psi_1 \wedge \psi_2$  then return MC( $\mathcal{K}, T, \psi_1$ )  $\wedge$  MC( $\mathcal{K}, T, \psi_2$ ) ;
5 if  $\varphi = \psi_1 \vee \psi_2$  then
6   existentially branch on all  $T_1 \cup T_2 = T$  s.t.  $T_1 \cap T_2 = \emptyset$ ;
7   return MC( $\mathcal{K}, T_1, \psi_1$ )  $\wedge$  MC( $\mathcal{K}, T_2, \psi_2$ )
8 if  $\varphi = \text{EX}\psi$  then
9   existentially branch on all successors  $T'$  of  $T$ ;    // successor check is in P by Lem. 9.4
10  return MC( $\mathcal{K}, T', \psi$ )
11 if  $\varphi = \text{AX}\psi$  then
12  universally branch on all successors  $T'$  of  $T$ ;    // successor check is in P by Lem. 9.4
13  return MC( $\mathcal{K}, T', \psi$ )
14 if  $\varphi = \text{E}[\alpha \cup \beta]$  then
15  existentially branch on path length  $c \in [0, |W|^{|T|}]$  ;    // Lem. 9.3
16  if  $c = 0$  then return MC( $\mathcal{K}, T, \beta$ );
17  else
18    existentially branch on  $T_{\text{end}-1}, T_{\text{end}}$  s.t.  $T_{\text{end}}$  is a successor of  $T_{\text{end}-1}$  and
19     $|T| = |T_{\text{end}}| = |T_{\text{end}-1}|$ ;
20    return EG-PathSearch( $\mathcal{K}, T, T_{\text{end}-1}, \alpha, c - 1$ )  $\wedge$  MC( $\mathcal{K}, T_{\text{end}}, \beta$ )
21 if  $\varphi = \text{E}[\alpha \text{R} \beta]$  then
22  existentially branch on  $\{\text{no-}\alpha, \text{some-}\alpha\}$ ;
23  if no- $\alpha$  then
24    existentially branch on  $T_{\text{end}}$  with  $|T| = |T_{\text{end}}|$ ;
25    return EG-PathSearch( $\mathcal{K}, T, T_{\text{end}}, \beta, |W|^{|T|} + 1$ )    // Lem. 9.3, infinite sequence
26  else if some- $\alpha$  then
27    existentially branch on path length  $c \in [0, |W|^{|T|}]$  ;    // Lem. 9.3
28    existentially branch on  $T_{\text{end}}$  with  $|T| = |T_{\text{end}}|$ ;
29    return MC( $\mathcal{K}, T_{\text{end}}, \alpha$ )  $\wedge$  EG-PathSearch( $\mathcal{K}, T, T_{\text{end}}, \beta, c$ )
30 if  $\varphi = \text{A}[\alpha \cup \beta]$  then
31  universally branch on  $\{\text{reach-}\beta, \text{all-}\alpha\text{-before-first-}\beta\}$ ;
32  if reach- $\beta$  then
33    universally branch on  $T_{\text{end}}$  with  $|T| = |T_{\text{end}}|$ ;
34    return AF-PathSearch( $\mathcal{K}, T, T_{\text{end}}, \beta, |W|^{|T|} + 1$ ) // Lem. 9.3 implies bounded sequence
35  else if all- $\alpha$ -before-first- $\beta$  then
36    universally branch on path length  $c \in [0, |W|^{|T|}]$  ;    // Lem. 9.3
37    universally branch on  $T_{\text{end}}$  with  $|T| = |T_{\text{end}}|$ ;
38    return MC( $\mathcal{K}, T_{\text{end}}, \alpha$ )  $\odot$  MC( $\mathcal{K}, T_{\text{end}}, \beta$ )  $\odot$  AF-PathSearch( $\mathcal{K}, T, T_{\text{end}}, \beta, c$ )
39 if  $\varphi = \text{A}[\alpha \text{R} \beta]$  then
40  universally branch on path length  $c \in [0, |W|^{|T|}]$  ;    // Lem. 9.3 implies longer sequences
41  if  $c = 0$  then return MC( $\mathcal{K}, T, \beta$ );
42  else
43    universally branch on  $T_{\text{end}-1}, T_{\text{end}}$  s.t.  $T_{\text{end}}$  is a successor of  $T_{\text{end}-1}$  and  $|T| = |T_{\text{end}}| = |T_{\text{end}-1}|$ ;
44    return AF-PathSearch( $\mathcal{K}, T, T_{\text{end}-1}, \alpha, c - 1$ )  $\odot$  MC( $\mathcal{K}, T_{\text{end}}, \beta$ )

```

$T_{\text{end}-1}$, where α holds at every position (lines 18 and 19). Such a path can be assumed to have at most exponential length due to Lemma 9.3. This captures exactly the semantics of the existential until.

For the R case, the algorithm checks one of two cases. Either (lines 23 and 24), there is an exponentially long path on which β holds everywhere (i.e., α must not hold), which implies, due to Lemma 9.3, that there is also an infinite such path. Or (lines 26–28), there is a team T_{end} such that α holds at T_{end} and there is a path from T to T_{end} , where β holds everywhere, including at T_{end} . Such a path can again be assumed to have at most exponential length due to Lemma 9.3. The disjunction of these two cases captures exactly the semantics of the existential release.

Let us finish by considering the universal variants of the temporal operators. Here, we again rely on Lemma 9.3 (this time to bound the length of counterexamples, as we consider universal path quantifiers) and Algorithm 3.

Let us begin with the R case, as it is the simpler one. Here, the algorithm checks whether T satisfies β (line 40) and whether for every pair $(T_{\text{end}}, T_{\text{end}-1})$ of teams such that T_{end} is a successor of $T_{\text{end}-1}$, T_{end} satisfies β or whether every path from T to $T_{\text{end}-1}$ satisfies α somewhere (lines 42 and 43), i.e., if β does not hold at T_{end} , then it must have been released by α on every path leading to T_{end} . Again, due to Lemma 9.3, we can restrict ourselves here to paths of exponential length, since this property always has a counterexample of exponential length. This captures exactly the semantics of the universal release.

Finally, for the U case, the algorithm checks two properties. The first one (lines 32 and 33) is that all paths of exponential length contain at least one team where β holds. Lemma 9.3 implies that this is equivalent to stating that in all infinite paths β eventually holds. The second one (lines 35–37) is that for every team T_{end} at least one of the following cases holds: T_{end} satisfies α , or T_{end} satisfies β , or all paths leading from T to T_{end} contain a team satisfying β . This second constraint boils down to the property that in every path α holds forever or until β is fulfilled. Again, by Lemma 9.3, it suffices to consider only paths of exponential length for the second property. Thus, both properties holding capture exactly the semantics of the universal until.

This overall guarantees polynomial runtime as well as the correctness of the alternating algorithm. As $\text{APTIME} = \text{PSPACE}$ [14], this proves the desired upper bound. $\square$

THEOREM 9.6. *MC(TeamCTL) is PSPACE-complete w.r.t. $\leq_m^P$ -reductions.*

PROOF. By Lemma 9.5 we know that MC(TeamCTL) is in PSPACE. By Lemma 9.2 we know that MC(TeamCTL) is PSPACE-hard. Thus, MC(TeamCTL) is PSPACE-complete. $\square$

Similar to LTL, we can consider FO-definable atoms (see Definition 5.1) in the CTL setting. Here, we restrict the parameters to generalised atoms to be formulae in propositional logic. Analogous to the proof of Theorem 5.2, we can extend the model-checking algorithm of Lemma 9.5 to deal with FO-definable generalised atoms and the contradictory negation.

THEOREM 9.7. *Let $\mathcal{D}$ be a finite set of FO definable generalised atoms. MC(TeamCTL($\mathcal{D}$, $\sim$)) is PSPACE-complete w.r.t. $\leq_m^P$ -reductions.*

9.2 Satisfiability

Again our benchmark here is CTL satisfiability with classical semantics.

PROPOSITION 9.8 ([26, 59]). *Satisfiability for CTL formulae under classical semantics is EXPTIME-complete w.r.t. $\leq_m^P$ -reductions.*

The same complexity result is easily transferred to TeamCTL.

THEOREM 9.9. *SAT(TeamCTL) is EXPTIME-complete w.r.t. $\leq_m^P$ -reductions.*

PROOF. The problem merely asks whether there exists a Kripke structure $\mathcal{K}$ and a non-empty team T of $\mathcal{K}$ such that $\mathcal{K}, T \models \varphi$ for given CTL formula φ . By downward closure (Proposition 7.7), it suffices to check whether φ is satisfied by a singleton team. By singleton equivalence (Proposition 7.6), we then immediately obtain the same complexity bounds as for classical CTL satisfiability. Hence, the claim follows from Proposition 9.8. $\square$

10 Conclusions

We introduced and studied team semantics for the temporal logics LTL and CTL. We concluded that TeamLTL (with and without generalised atoms) is a valuable logic which allows expressing relevant hyperproperties and complements the expressiveness of HyperLTL while allowing for computationally simpler decision problems.

For TeamCTL, the complexity of the model-checking problem increases from P-complete for usual CTL to PSPACE-complete for TeamCTL. This fact stems from the expressive notion of synchronicity between team members and is in line with the results of Kupferman et al. [47].

We conclude with some directions of future work and open problems.

10.1 Future Work for TeamLTL

We showed that some important properties that cannot be expressed in HyperLTL (such as uniform termination) can be expressed in TeamLTL. Moreover, input determinism can be expressed in TeamLTL (dep). Can we identify tractable TeamLTL variants (i.e., syntactic fragments with particular dependency atoms) that can express a rich family of hyperproperties?

We showed that with respect to expressive power, HyperLTL and TeamLTL are incomparable. However, the expressive power of HyperLTL and the different extensions of TeamLTL introduced here is left open. For example, the HyperLTL formula $\exists \pi. p_\pi$ is expressible in TeamLTL($\sim$).

Another interesting question is whether we can characterise the expressive power of relevant extensions of TeamLTL as has been done in FO and modal contexts. Recent works have shown limits of expressivity of TeamLTL variants via translations to extensions and fragments of HyperLTL [41, 73] and FO [40]. It is also open whether the one-way translations in those papers can be strengthened to precise characterisations of expressivity as was done in the work of Kontinen et al. [41] for asynchronous set-based TeamLTL.

We studied the complexity of the path-checking, model-checking and satisfiability problems of TeamLTL and its extensions. However, many problems are still open: Can we identify matching upper and lower bounds for the missing cases and partial results of Figure 1? In particular, what is the complexity of model-checking TeamLTL when splitjunctions are allowed?

Finally, the complexity of the validity and implication problems is open for almost all cases.

10.2 Future Work for TeamCTL

We only scratched the surface of the complexity of the satisfiability problem of TeamCTL. There are two obvious directions for future work here. Complexity of TeamCTL extended with atoms and connectives that preserve downward closure, and complexity of TeamCTL extended with non-downward-closed atoms and connectives. The complexity of the former is expected to stay relatively low and comparable to vanilla TeamCTL, since it still suffices to consider only singleton teams. The complexity of the latter is expected to be much higher and perhaps even undecidable. Note that TeamLTL with the contradictory negation is highly undecidable [51].

The tautology or validity problem for this new logic is quite interesting and seems to have a higher complexity than the related satisfiability problem. This is due to alternation of set quantification: the validity problem quantifies over teams universally while the splitjunction implements an existential set quantification. We leave the considerations related to the validity problem as future work. Formally the corresponding problem is defined as follows:

Problem:	VAL(TeamCTL)—TeamCTL Validity Problem.
Input:	A TeamCTL formula φ .
Question:	Does $\mathcal{K}, T \models \varphi$ hold for every Kripke structure $\mathcal{K}$ and every team T of $\mathcal{K}$?

In the context of team-based modal logics, the computational complexity of the validity problem has been studied by Virtema [71], Hannula [33] and Lück [50]. Virtema and Hannula showed that the problem for modal dependence logic is NEXPTIME-complete whereas Lück established that the problem for modal logic extended with the contradictory negation is complete for the complexity class TOWER(poly), the class of problems that can be solved in time that is bounded by some n -fold exponential, where n itself is bounded polynomially in the input length.

It is well-known that there are several ways to measure the complexity of a model-checking problem. In general, a model and a formula are given, and then one needs to decide whether the model satisfies the formula. *System complexity* considers the computational complexity for the case of a fixed formula, whereas *specification complexity* fixes the underlying Kripke structure. We considered in this article the *combined complexity* where both a formula and a model belong to the given input. Yet the other two approaches might give more specific insights into the intractability of the model-checking case we investigated. In particular, the study of the *specification complexity*, where the team or the team size is assumed to be fixed, might as well be of independent interest.

Finally, this leads to the consideration of different kinds of restrictions on the problems. In particular, for the quite strong PSPACE-completeness result for model checking in team semantics, it is of interest how this intractability can be explained. Hence, the investigation of fragments by means of allowed temporal operators and/or Boolean operators will lead to a better understanding of this presumably untameably high complexity.

Acknowledgement

We thank the anonymous reviewers for their valuable feedback.

References

- [1] Samson Abramsky, Joni Puljujärvi, and Jouko Väänänen. 2026. Team semantics and independence notions in quantum physics. *The Bulletin of Symbolic Logic* 32, 1 (2026), 82–135. DOI : <https://doi.org/10.1017/bsl.2025.10089>
- [2] Rafael Albert and Erich Grädel. 2022. Unifying hidden-variable problems from quantum mechanics by logics of dependence and independence. *Annals of Pure and Applied Logic* 173, 10 (2022), 103088. DOI : <https://doi.org/10.1016/J.APAL.2022.103088>
- [3] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of Model Checking*. The MIT Press, Cambridge, MA.
- [4] Fausto Barbero and Jonni Virtema. 2024. Expressivity landscape for logics with probabilistic interventionist counterfactuals. In *CSL. Aniello Murano and Alexandra Silva (Eds.), Leibniz International Proceedings in Informatics, Vol. 288. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Article 15, 1–19*. DOI : <https://doi.org/10.4230/LIPICS.CSL.2024.15>
- [5] Ezio Bartocci, Thomas A. Henzinger, Dejan Nickovic, and Ana Oliveira da Costa. 2023. Hypernode automata. In *CONCUR. Guillermo A. Pérez and Jean-François Raskin (Eds.), Leibniz International Proceedings in Informatics, Vol. 279. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 21:1–21:16*. DOI : <https://doi.org/10.4230/LIPICS.CONCUR.2023.21>
- [6] Michael Bauland, Martin Mundhenk, Thomas Schneider, Henning Schnoor, Ilka Schnoor, and Heribert Vollmer. 2011. The tractability of model checking for LTL: The good, the bad, and the ugly fragments. *ACM Transactions Computational Logic* 12, 2, Article 13 (2011), 1–28. DOI : <https://doi.org/10.1145/1877714.1877719>

- [7] Michael Bauland, Thomas Schneider, Henning Schnoor, Ilka Schnoor, and Heribert Vollmer. 2009. The complexity of generalized satisfiability for linear temporal logic. *Logical Methods Computer Science* 5, 1 (2009), 1–21. DOI : [https://doi.org/10.2168/LMCS-5\(1:1\)2009](https://doi.org/10.2168/LMCS-5(1:1)2009)
- [8] Jan Baumeister, Norine Coenen, Borzoo Bonakdarpour, Bernd Finkbeiner, and César Sánchez. 2021. A temporal logic for asynchronous hyperproperties. In *CAV*. Alexandra Silva and K. Rustan M. Leino (Eds.), Lecture Notes in Computer Science, Vol. 12759, Springer, 694–717. DOI : https://doi.org/10.1007/978-3-030-81685-8_33
- [9] Dylan Bellier, Massimo Benerecetti, Dario Della Monica, and Fabio Mogavero. 2023. Good-for-game QPTL: An alternating hedges semantics. *ACM Transactions Computational Logic* 24, 1, Article 4 (2023), 1–57. DOI : <https://doi.org/10.1145/3565365>
- [10] Raven Beutner and Bernd Finkbeiner. 2023. AutoHyper: Explicit-state model checking for HyperLTL. In *TACAS*. Sriram Sankaranarayanan and Natasha Sharygina (Eds.), Lecture Notes in Computer Science, Vol. 13993, Springer, 145–163. DOI : https://doi.org/10.1007/978-3-031-30823-9_8
- [11] Borzoo Bonakdarpour and Bernd Finkbeiner. 2016. Runtime verification for HyperLTL. In *RV*. Yliès Falcone and César Sánchez (Eds.), Lecture Notes in Computer Science, Vol. 10012, Springer, 41–45. DOI : https://doi.org/10.1007/978-3-319-46982-9_4
- [12] Laura Bozzelli, Bastien Maubert Sophie Pinchinat. 2015. Unifying hyper and epistemic temporal logics. In *FoSSaCS*. Andrew M. Pitts (Ed.), Lecture Notes in Computer Science, Vol. 9034, Springer, 167–182. DOI : https://doi.org/10.1007/978-3-662-46678-0_11
- [13] Laura Bozzelli, Adriano Peron, and César Sánchez. 2021. Asynchronous extensions of HyperLTL. In *LICS*. IEEE, 1–13. DOI : <https://doi.org/10.1109/LICS52264.2021.9470583>
- [14] Ashok K. Chandra, Dexter Kozen, and Larry J. Stockmeyer. 1981. Alternation. *Journal of the ACM* 28, 1 (1981), 114–133. DOI : <https://doi.org/10.1145/322234.322243>
- [15] Edmund M. Clarke, E. Allen Emerson, and A. Prasad Sistla. 1986. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages Systems* 8, 2 (1986), 244–263. DOI : <https://doi.org/10.1145/5397.5399>
- [16] Michael R. Clarkson, Bernd Finkbeiner, Masoud Kolehini, Kristopher K. Micinski, Markus N. Rabe, and César Sánchez. 2014. Temporal logics for hyperproperties. In *POST*. M. Abadi and S. Kremer (Eds.), Lecture Notes in Computer Science, Vol. 8414, Springer, 265–284. DOI : https://doi.org/10.1007/978-3-642-54792-8_15
- [17] Michael R. Clarkson and Fred B. Schneider. 2010. Hyperproperties. *Journal of Computer Security* 18, 6 (2010), 1157–1210. DOI : <https://doi.org/10.3233/JCS-2009-0393>
- [18] Pedro R. D’Argenio, Gilles Barthe, Sebastian Biewer, Bernd Finkbeiner, and Holger Hermanns. 2017. Is your software on dope? – Formal analysis of surreptitiously “enhanced” programs. In *ESOP*. H. Yang (Ed.), Lecture Notes in Computer Science, Vol. 10201, Springer, 83–110. DOI : https://doi.org/10.1007/978-3-662-54434-1_4
- [19] Arnaud Durand, Juha Kontinen, and Heribert Vollmer. 2016. Expressivity and complexity of dependence logic. In *Dependence Logic: Theory and Applications*. S. Abramsky, J. Kontinen, J. Väänänen, and H. Vollmer (Eds.), Springer, 5–32. DOI : https://doi.org/10.1007/978-3-319-31803-5_2
- [20] Johannes Ebbing, Lauri Hella, Arne Meier, Julian-Steffen Müller, Jonni Virtama, and Heribert Vollmer. 2013. Extended modal dependence logic. In *WoLLIC*. Leonid Libkin, Ulrich Kohlenbach, and Ruy J. G. B. de Queiroz (Eds.), Lecture Notes in Computer Science, Vol. 8071, Springer, 126–137. DOI : https://doi.org/10.1007/978-3-642-39992-3_13
- [21] Bernd Finkbeiner and Christopher Hahn. 2016. Deciding hyperproperties. In *CONCUR*. José Desharnais and Radha Jagadeesan (Eds.), Leibniz International Proceedings in Informatics, Vol. 59, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Article 13, 1–14. DOI : <https://doi.org/10.4230/LIPIcs.CONCUR.2016.13>
- [22] Bernd Finkbeiner, Christopher Hahn, Marvin Stenger, and Leander Tentrup. 2017. Monitoring hyperproperties. In *RV*. S. Lahiri and G. Reger (Eds.), Lecture Notes in Computer Science, Vol. 10548, Springer, 190–207. DOI : https://doi.org/10.1007/978-3-319-67531-2_12
- [23] Bernd Finkbeiner, Christian Müller, Helmut Seidl, and Eugen Zalinescu. 2017. Verifying security policies in multi-agent workflows with loops. In *CCS*. ACM, 633–645. DOI : <https://doi.org/10.1145/3133956.3134080>
- [24] Bernd Finkbeiner, Markus N. Rabe, and César Sánchez. 2015. Algorithms for model checking HyperLTL and HyperCTL*. In *CAV*. D. Kroening and C. Păsăreanu (Eds.), Lecture Notes in Computer Science, Vol. 9206, Springer, 30–48. DOI : https://doi.org/10.1007/978-3-319-21690-4_3
- [25] Bernd Finkbeiner and Martin Zimmermann. 2017. The first-order logic of hyperproperties. In *STACS*. Heribert Vollmer and Brigitte Vallée (Eds.), Leibniz International Proceedings in Informatics, Vol. 66, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Article 30, 1–14. DOI : <https://doi.org/10.4230/LIPIcs.STACS.2017.30>
- [26] Michael J. Fischer and Richard E. Ladner. 1979. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences* 18, 2 (1979), 194–211. DOI : [https://doi.org/10.1016/0022-0000\(79\)90046-1](https://doi.org/10.1016/0022-0000(79)90046-1)
- [27] Marie Fortin, Louwe B. Kuijter, Patrick Totzke, and Martin Zimmermann. 2025. HyperLTL satisfiability is highly undecidable, HyperCTL* is even harder. *Logical Methods in Computer Science* 21, 1 (2025), 3. DOI : [https://doi.org/10.46298/LMCS-21\(1:3\)2025](https://doi.org/10.46298/LMCS-21(1:3)2025)

- [28] Hadar Frenkel and Martin Zimmermann. 2025. The complexity of second-order HyperLTL. In *CSL*. Jörg Endrullis and Sylvain Schmitz (Eds.), Leibniz International Proceedings in Informatics, Vol. 326, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Article 10, 1–23. DOI: <https://doi.org/10.4230/LIPICS.CSL.2025.10>
- [29] Pietro Galliani. 2012. Inclusion and exclusion dependencies in team semantics - On some logics of imperfect information. *Annals of Pure and Applied Logic* 163, 1 (2012), 68–84. DOI: <https://doi.org/10.1016/j.apal.2011.08.005>
- [30] Erich Grädel and Jouko Väänänen. 2013. Dependence and independence. *Studia Logica* 101, 2 (2013), 399–410. DOI: <https://doi.org/10.1007/s11225-013-9479-2>
- [31] Jens Oliver Gutsfeld, Arne Meier, Christoph Ohrem, and Jonni Virtema. 2022. Temporal team semantics revisited. In *LICS*. ACM, Article 44, 1–13. DOI: <https://doi.org/10.1145/3531130.3533360>
- [32] Jens Oliver Gutsfeld, Markus Müller-Olm, and Christoph Ohrem. 2021. Automata and fixpoints for asynchronous hyperproperties. *Proceedings of the ACM on Programming Languages* 5, POPL (2021), 1–29. DOI: <https://doi.org/10.1145/3434319>
- [33] Miika Hannula. 2019. Validity and entailment in modal and propositional dependence logics. *Logical Methods in Computer Science* 15, 2 (2019), 1–24. DOI: [https://doi.org/10.23638/LMCS-15\(2:4\)2019](https://doi.org/10.23638/LMCS-15(2:4)2019)
- [34] Miika Hannula, Juha Kontinen, Jan Van den Bussche, and Jonni Virtema. 2020. Descriptive complexity of real computation and probabilistic independence logic. In *LICS*. ACM, 550–563. DOI: <https://doi.org/10.1145/3373718.3394773>
- [35] Miika Hannula, Juha Kontinen, and Jonni Virtema. 2020. Polyteam semantics. *Journal of Logic and Computation* 30, 8 (2020), 1541–1566. DOI: <https://doi.org/10.1093/logcom/exaa048>
- [36] Miika Hannula, Juha Kontinen, Jonni Virtema, and Heribert Vollmer. 2018. Complexity of propositional logics in team semantic. *ACM Transactions on Computational Logic* 19, 1, Article 2 (Jan. 2018), 1–14. DOI: <https://doi.org/10.1145/3157054>
- [37] Lauri Hella, Kerkko Luosto, Katsuhiko Sano, and Jonni Virtema. 2014. The expressive power of modal dependence logic. In *AiML*. Rajeev Goré, Barteld P. Kooi, and Agi Kurucz (Eds.), College Publications, 294–312. Retrieved from <http://www.aiml.net/volumes/volume10/Hella-Luosto-Sano-Virtema.pdf>
- [38] Wilfrid Hodges. 1997. Compositional semantics for a language of imperfect information. *Logic Journal of IGPL* 5, 4 (1997), 539–563. DOI: <https://doi.org/10.1093/jigpal/5.4.539>
- [39] Neil Immerman. 1999. *Descriptive Complexity*. Springer, New York, NY. DOI: <https://doi.org/10.1007/978-1-4612-0539-5>
- [40] Juha Kontinen and Max Sandström. 2021. On the expressive power of TeamLTL and first-order team logic over hyperproperties. In *WoLLIC*. Alexandra Silva, Renata Wassermann, and Ruy de Queiroz (Eds.), Lecture Notes in Computer Science, Vol. 13038. Springer, 302–318. DOI: https://doi.org/10.1007/978-3-030-88853-4_19
- [41] Juha Kontinen, Max Sandström, and Jonni Virtema. 2024. A remark on the expressivity of asynchronous TeamLTL and HyperLTL. In *FoIKS*. A. Meier and M. Ortiz (Eds.), Lecture Notes in Computer Science, Vol. 14589. Springer, 275–286. DOI: https://doi.org/10.1007/978-3-031-56940-1_15
- [42] Juha Kontinen, Max Sandström, and Jonni Virtema. 2025. Set semantics for asynchronous TeamLTL: Expressivity and complexity. *Information and Computation* 304 (2025), 1–18. DOI: <https://doi.org/10.1016/j.IJC.2025.105299>
- [43] Andreas Krebs, Arne Meier, and Martin Mundhenk. 2019. The model checking fingerprints of CTL operators. *Acta Informatica* 56, 6 (2019), 487–519. DOI: <https://doi.org/10.1007/S00236-018-0326-9>
- [44] Andreas Krebs, Arne Meier, and Jonni Virtema. 2015. A team based variant of CTL. In *TIME*. Fabio Grandi, Martin Lange, and Alessio Lomuscio (Eds.), IEEE Computer Society, 140–149. DOI: <https://doi.org/10.1109/TIME.2015.11>
- [45] Andreas Krebs, Arne Meier, Jonni Virtema, and Martin Zimmermann. 2018. Team semantics for the specification and verification of hyperproperties. In *MFCS*. Igor Potapov, Paul Spirakis, and James Worrell (Eds.), Leibniz International Proceedings in Informatics, Vol. 117. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Article 10, 1–16. DOI: <https://doi.org/10.4230/LIPICS.MFCS.2018.10>
- [46] Lars Kuhlitz and Bernd Finkbeiner. 2009. LTL path checking is efficiently parallelizable. In *ICALP*. S. Albers, A. Marchetti-Spaccamela, Y. Matias, S. Nikolettseas, and W. Thomas (Eds.), Lecture Notes in Computer Science, Vol. 5556. Springer, 235–246. DOI: https://doi.org/10.1007/978-3-642-02930-1_20
- [47] Orna Kupferman, Moshe Y. Vardi, and Pierre Wolper. 2000. An automata-theoretic approach to branching-time model checking. *Journal of the ACM* 47, 2 (2000), 312–360. DOI: <https://doi.org/10.1145/333979.333987>
- [48] Antti Kuusisto. 2015. A double team semantics for generalized quantifiers. *Journal of Logic, Language, and Information* 24, 2 (2015), 149–191. DOI: <https://doi.org/10.1007/s10849-015-9217-4>
- [49] Richard Ladner. 1977. The computational complexity of provability in systems of modal propositional logic. *SIAM Journal on Computing* 6, 3 (1977), 467–480. DOI: <https://doi.org/10.1137/0206033>
- [50] Martin Lück. 2018. Canonical models and the complexity of modal team logic. In *CSL*. Dan Ghica and Achim Jung (Eds.), Leibniz International Proceedings in Informatics, Vol. 119. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Article 30, 1–23. DOI: <https://doi.org/10.4230/LIPICS.CSL.2018.30>

- [51] Martin Lück. 2020. On the complexity of linear temporal logic with team semantics. *Theoretical Computer Science* 837 (2020), 1–25. DOI : <https://doi.org/10.1016/J.TCS.2020.04.019>
- [52] Yasir Mahmood, Jonni Virtema, Timon Barlag, and Axel-Cyrille Ngonga Ngomo. 2024. Computing repairs under functional and inclusion dependencies via argumentation. In *FoKS*. A. Meier and M. Ortiz (Eds.), Lecture Notes in Computer Science, Vol. 14589. Springer, 23–42. DOI : https://doi.org/10.1007/978-3-031-56940-1_2
- [53] Nicolas Markey and Philippe Schnoebelen. 2003. Model checking a path. In *CONCUR*. Roberto M. Amadio and Denis Lugiez (Eds.), Lecture Notes in Computer Science, Vol. 2761, Springer, Marseille, France, 248–262. DOI : https://doi.org/10.1007/978-3-540-45187-7_17
- [54] Corto Mascle and Martin Zimmermann. 2020. The keys to decidable HyperLTL satisfiability: Small models or very simple formulas. In *CSL*. Maribel Fernández and Anca Muscholl (Eds.), Leibniz International Proceedings in Informatics, Vol. 152. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Article 29, 1–16. DOI : <https://doi.org/10.4230/LIPICS.CSL.2020.29>
- [55] Arne Meier, Julian-Steffen Müller, Martin Mundhenk, and Heribert Vollmer. 2012. Complexity of model checking for logics over kripke models. *Bulletin of the EATCS* 108 (2012), 49–89. Retrieved from <http://eatcs.org/beatcs/index.php/beatcs/article/view/50>
- [56] Arne Meier, Michael Thomas, Heribert Vollmer, and Martin Mundhenk. 2009. The complexity of satisfiability for fragments of CTL and CTL*. *International Journal of Foundations of Computer Science* 20, 5 (2009), 901–918. DOI : <https://doi.org/10.1142/S0129054109006954>
- [57] Julian-Steffen Müller. 2014. *Satisfiability and Model Checking in Team Based Logics*. Ph.D. Dissertation. Leibniz University of Hannover. Retrieved from <https://d-nb.info/1054741921>
- [58] Amir Pnueli. 1977. The temporal logic of programs. In *FOCS*. IEEE Computer Society, 46–57. DOI : <https://doi.org/10.1109/SFCS.1977.32>
- [59] Vaughan R. Pratt. 1980. A near-optimal method for reasoning about action. *Journal of Computer and System Sciences* 20, 2 (1980), 231–254. DOI : [https://doi.org/10.1016/0022-0000\(80\)90061-6](https://doi.org/10.1016/0022-0000(80)90061-6)
- [60] Markus N. Rabe. 2016. *A Temporal Logic Approach to Information-Flow Control*. Ph.D. Dissertation. Saarland University. Retrieved from <http://scidok.sulb.uni-saarland.de/volltexte/2016/6387/>
- [61] Gaëtan Regaud and Martin Zimmermann. 2024. The complexity of HyperQPTL. arXiv:2412.07341. Retrieved from <https://arxiv.org/abs/2412.07341>
- [62] Gaëtan Regaud and Martin Zimmermann. 2025. The complexity of fragments of second-order HyperLTL. arXiv:2501.19046. Retrieved from <https://arxiv.org/abs/2501.19046>
- [63] Walter J. Savitch. 1969. Deterministic simulation of Non-Deterministic turing machines (detailed abstract). In *STOC*. ACM, 247–248. DOI : <https://doi.org/10.1145/800169.805439>
- [64] Philippe Schnoebelen. 2002. The complexity of temporal logic model checking. In *AiML*. King’s College Publications, 393–436. Retrieved from <http://www.aiml.net/volumes/volume4/Schnoebelen.ps>
- [65] Michael Sipser. 2013. *Introduction to the Theory of Computation* (3rd ed.). Course Technology, Boston, MA.
- [66] A. P. Sistla and E. M. Clarke. 1985. The complexity of propositional linear temporal logics. *Journal of the ACM* 32, 3 (1985), 733–749. DOI : <https://doi.org/10.1145/3828.3837>
- [67] Larry J. Stockmeyer. 1976. The polynomial-time hierarchy. *Theoretical Computer Science*. 3, 1 (1976), 1–22. DOI : [https://doi.org/10.1016/0304-3975\(76\)90061-X](https://doi.org/10.1016/0304-3975(76)90061-X)
- [68] Jouko Väänänen. 2007. *Dependence Logic*. Cambridge University Press, Cambridge, UK.
- [69] Jouko Väänänen. 2008. Modal dependence logic. In *New Perspectives on Games and Interaction*. Krzysztof R. Apt and Robert van Rooij (Eds.), Amsterdam University Press, Amsterdam, 237–254.
- [70] Jouko A. Väänänen. 2011. *Models and Games*. Cambridge Studies in Advanced Mathematics, Vol. 132. Cambridge University Press, Cambridge, UK. Retrieved from http://www.cambridge.org/de/knowledge/isbn/item6173714/?site_locale=de_DE
- [71] Jonni Virtema. 2017. Complexity of validity for propositional dependence logics. *Information and Computation* 253 (2017), 224–236. DOI : <https://doi.org/10.1016/j.ic.2016.07.008>
- [72] Jonni Virtema, Jana Hofmann, Bernd Finkbeiner, Juha Kontinen, and Fan Yang. 2020. Linear-time temporal logic with team semantics: Expressivity and complexity. arXiv:2010.03311. Retrieved from <https://arxiv.org/abs/2010.03311>
- [73] Jonni Virtema, Jana Hofmann, Bernd Finkbeiner, Juha Kontinen, and Fan Yang. 2021. Linear-time temporal logic with team semantics: Expressivity and complexity. In *FSTTCS*. Mikołaj Bojańczyk and Chandra Chekuri (Eds.), Leibniz International Proceedings in Informatics, Vol. 213. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Article 52, 1–17. DOI : <https://doi.org/10.4230/LIPICS.FSTTCS.2021.52>

Received 27 September 2024; revised 9 March 2026; accepted 29 June 2026