Temporal Team Semantics Revisited

Jens Oliver Gutsfeld
Christoph Ohrem
jens.gutsfeld@uni-muenster.de
christoph.ohrem@uni-muenster.de
Institut für Informatik, Westfälische Wilhelms-Universität Münster
Münster, Germany

Arne Meier
meier@thi.uni-hannover.de
Institut für Theoretische Informatik, Leibniz Universität Hannover
Hannover, Germany

Jonni Virtema
j.t.virtema@sheffield.ac.uk
Department of Computer Science, University of Sheffield
Sheffield, United Kingdom

ABSTRACT

In this paper, we study a novel approach to asynchronous hyperproperties by reconsidering the foundations of temporal team semantics. We consider three logics: TeamLTL, TeamCTL and TeamCTL*, which are obtained by adding quantification over so-called *time evaluation functions* controlling the asynchronous progress of traces. We then relate synchronous TeamLTL to our new logics and show how it can be embedded into them. We show that the model checking problem for $\exists$TeamCTL with Boolean disjunctions is highly undecidable by encoding recurrent computations of non-deterministic 2-counter machines. Finally, we present a translation from TeamCTL* to Alternating Asynchronous Büchi Automata and obtain decidability results for the path checking problem as well as restricted variants of the model checking and satisfiability problems.

CCS CONCEPTS

• **Theory of computation → Modal and temporal logics**; **Problems, reductions and completeness**; Logic and verification.

KEYWORDS

Team Semantics, Temporal Logic, Hyperproperties, Automata Theory, Model Checking, Asynchronicity

ACM Reference Format:
Jens Oliver Gutsfeld, Christoph Ohrem, Arne Meier, and Jonni Virtema. 2022. Temporal Team Semantics Revisited. In *37th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) (LICS '22), August 2–5, 2022, Be'er Sheva, Israel.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3531130.3533360

1 INTRODUCTION

Since the 1980s, model checking has become a staple in verification. For Linear Temporal Logic (LTL) and its progeny, the model checking problem asks whether every trace of a given system fulfils a given temporal specification such as a liveness or fairness property. Notably, this specification considers the traces of the input in isolation and cannot relate different traces to each other. However, it is not hard to come up with natural properties that require viewing different traces in tandem. For example, asking whether the value of a variable x on average exceeds some constant c amounts to summing up the value of x for *all* traces and then averaging the result. In this context, the information given by a single trace viewed in isolation is of little avail. Likewise, it does not suffice to consider properties of isolated traces, when we consider executions of parallel programs in which individual threads are represented by single traces. The same is true for information-flow properties of systems like observational determinism or generalised non-interference. This need to be able to specify properties of collections of traces has lead to the introduction of the notion of a *hyperproperty* [10]. Technically, a trace property is just a set of traces, and vice versa. Hyperproperties on the other hand describe properties of sets of traces, and thus correspond to sets of sets of traces. Since established temporal logics like LTL can express only trace properties, but not genuine hyperproperties, new logics were developed for hyperproperties. Generally, the approach has been to pick a temporal logic defined on traces and lift it to sets of traces by adding quantification over named paths, For example, LTL becomes HyperLTL [9], QPTL becomes HyperQPTL [31], PDL-Δ becomes HyperPDL-Δ [19] and so on.

A promising alternative approach for lifting temporal logics to hyperproperties is to shift to the so-called *team semantics.* In the past decade, team logics have established themselves as a vibrant area of research [1, 17]. The term *team semantics* was coined by Väänänen [32], inspired by the earlier work of Hodges [23]. The idea behind all logics utilising team semantics is to evaluate formulae, not over single states of affairs such as assignments or traces, but over sets of such states of affairs (i.e., over *teams*). Soon after its origin, team semantics was already applied to first-order, propositional, and modal settings. At the heart of these logics lies the ability to enrich the logical language with novel atomic formulae for stating properties of teams. The most prominent of these atoms is the *dependence atom* $\mathrm{dep}(\bar{x}, \bar{y})$ stating that the variables $\bar{x}$ functionally determine the values of $\bar{y}$ with respect to some given team (a set of assignments). Another important atom is the *inclusion atom* $\bar{x} \subseteq \bar{y}$ expressing the inclusion dependency that all the values that occur for $\bar{x}$ in a given team, also occur as a value for $\bar{y}$. Team Logics implement concepts and formalisms from a wealth of different disciplines such as statistics and database theory [17]. While the bulk of the research has concerned itself on logics expressing qualitative properties of data, recent discoveries in multiset [16] and probabilistic [21, 22] variants of team semantics have shifted the focus to the quantitative setting.

Recently, Krebs et al. [26] made an important advancement to the field by introducing the first team based temporal logics for

LICS '22, August 2–5, 2022, Be'er Sheva, Israel
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9351-5/22/08.
https://doi.org/10.1145/3531130.3533360

hyperproperties. The logic TeamLTL does not add quantifiers or names to LTL, but instead achieves the lifting to hyperproperties by adapting team semantics, i.e., by evaluating formulae directly over sets of traces and adding new atomic statements that can be used to express hyperproperties such as *non-inference* directly. Like LTL, but unlike HyperLTL and related logics, TeamLTL retains the property of being a purely combinatory, quantifier-free logic. It can also express specifications for which no analogue in HyperLTL is available. Most works on the named quantifier approach and the team semantics approach concentrate on synchronous interactions between traces. In 2018, Krebs et al. [26] introduced two different semantics for TeamLTL: a synchronous one and an asynchronous one. They can be seen as polar opposites: in the first, computations progress in lock step, and in the second, there is no way of relating the passage of time between distinct traces. This asynchronous semantics is rather weak and cannot deal with the plethora of ways asynchronicity occurs in real-world systems. For example, in order to capture multithreaded environments in which processes are not scheduled lockstepwise but still have some rules governing the computation, a setting that can model different modes of aynchronicity is required. The ubiquity of asynchronicity thus calls for the development of new hyperlogics that can express asynchronous specifications. In 2021, Gutsfeld et al. [20] conducted the first systematic study of asynchronous hyperproperties and introduced both a temporal fix-point calculus, H_μ, and an automata-theoretic framework, Alternating Asynchronous Parity Automata (AAPA), to tackle this class of properties.

Our contribution. In this paper, we present a new approach to TeamLTL by using explicit quantification over *time evaluation functions* (tefs for short) that describe the asynchronous interleavings of traces. This allows for the fine-grained use of asynchronicity in TeamLTL specifications. Using the new approach, we reconstruct the semantics of TeamLTL from scratch, thereby also defining novel team semantics variants of CTL and CTL* (i.e., TeamCTL and TeamCTL*). As an example (see Sections 3.2 and 3.3 for the precise semantics), let $o_1, \dots, o_n$ be observable outputs, c_1, c_2 be confidential outputs, and s be a secret. The formula $\varphi := G_\forall(o_1, \dots, o_n, s) \subseteq (o_1, \dots, o_n, \neg s)$ expresses a form of *non-inference* by stating that independent of the asynchronous behaviour of the system, the observer cannot infer the current value of the secret from the outputs. The formula $\psi := G_\exists \mathrm{dep}(c_1, c_2, s)$ expresses that for some asynchronous behaviour of the system, the confidential outputs functionally determine the secret. Finally, the formula $\psi \lor \varphi$ states that the executions of the system can be decomposed into two parts; in the first part, the aforementioned dependence holds, while in the second part, the non-inference property holds.

We wish to emphasise that we are not redefining asynchronous TeamLTL (or synchronous TeamLTL for that matter). Our goal is to define semantics for TeamLTL that is versatile enough to deal with the plethora of different modes of asynchronicity that occur in real-world applications. We propose a formalism that can express both synchronous and asynchronous behaviour. Indeed, we show that synchronous TeamLTL can be embedded into our new logics. Asynchronous TeamLTL (as defined by Krebs et al. [26]) cannot be directly embedded into our new setting, for each time evaluation function describes a dependence between the global clock and the local clocks. In asynchronous TeamLTL, no such dependence exists and the setting resembles somewhat our TeamCTL, albeit with modified semantics.

Besides quantified tefs and LTL constructs, we also study several extensions of TeamLTL with different atoms from the team semantics literature, e.g. the dependence and inclusion atoms mentioned above. We establish that our logics provide a unifying framework in which previous temporal logics with team semantics can be embedded in an intuitive and efficient manner. We show that the model checking problem is highly undecidable already for the extension of ∃TeamCTL with the Boolean disjunction. However, we also present a translation from TeamCTL* to Alternating Asynchronous Büchi Automata (AABA), a subset of AAPA with a Büchi condition, over finite teams of fixed size. This translation allows us to transfer *restricted interleaving semantics* for AAPA, such as the k-*synchronous* and k-*context-bounded* semantics of [20], to TeamCTL* and employ decidability results for these restricted semantics for the path checking problem and finite variants of the satisfiability and model checking problems. This translation is of independent interest because it constitutes the first application of automata-theoretic methods in the context of team semantics and dependence logic, and can therefore serve as a cornerstone for further development in this area. Our complexity results for the model checking problem are depicted on page 12 in Table 2.

Related work. Hyperlogics that add quantifiers for named paths have been studied before [4, 6, 9, 14, 15, 19]. These logics are orthogonal to ours as they do not involve team semantics. Hyperlogics with team semantics have been studied in the past as well [24–27, 33]. These logics either have purely synchronous semantics or they do not allow for fine-grained control of the asynchronicity as do our time evaluation functions and fragments with restricted asynchronous semantics.

In recent years, several research groups have embarked on a systematic study on asynchronous hyperproperties [3, 5, 7, 20]. Gutsfeld et al. [20] study asynchronous hyperproperties using the fix-point calculus H_μ and Alternating Asynchronous Parity Automata. Other asynchronous variants of HyperLTL have been introduced in [3, 5, 7]. While Bozelli et al. [7] focus on both HyperLTL variants with special modalities referring to stuttering on paths and contexts describing asynchronous behaviour, Bonakdarpour et al. [5] and Baumeister et al. [3] use quantifications over so-called *trajectories* which determine the asynchronous interleaving of traces. These trajectories are similar to our tefs, but they are not studied in the context of team semantics or AABA and no specific analysis of the properties for them is presented. As we show, variants of our logics can express properties such as synchronicity or fairness for tefs and there is no way in sight to do this in the logic of Baumeister et al. [3]. Coenen et al. [11] compare the expressive power of different hyperlogics systematically. However, none of these logics utilise team semantics.

2 PRELIMINARIES

We assume familiarity with complexity theory [28] and make use of the classes PSPACE, EXPSPACE, and P. Also we deal with different degrees of undecidability, e.g., Σ^0_1 and Σ^1_1. A thorough introduction in this regard can be found in the textbook of Pippenger [29].

General Notation. If $\vec{a} = (a_0, \ldots, a_{n-1})$ is an n-tuple of elements and $i < n$ a natural number, we set $\vec{a}[i] \coloneqq a_i$. For n-tuples $\vec{a}, \vec{b} \in \mathbb{N}^n$ ($n \in \mathbb{N} \cup \{\omega\}$), write $\vec{a} \leq \vec{b}$ whenever $\vec{a}[i] \leq \vec{b}[i]$, for each $i < n$; write $\vec{a} < \vec{b}$, if additionally $\vec{a} \neq \vec{b}$.

Multisets. Intuitively, a multiset is a generalisation of a set that records the multiplicities of its elements. The collections $\{a, a, b\}$ and $\{a, b\}$ are different multisets, while they are identical when interpreted as sets. Here, we encode multisets as sets by appending unique indices to the elements of the multisets.

Let I be some infinite set of indices such as $\mathbb{N} \cup \mathbb{N}^\omega$. A *multiset* is a set A of pairs (i, v), where $i \in I$ is an index value and v is a set element, such that $a[0] \neq b[0]$ for all distinct $a, b \in A$. Multisets A and B are the *same multisets* (written $A = B$), if there exists a bijection $f : I \to I$ such that $B = \{ (f(a[0]), a[1]) \mid a \in A \}$. Using this notation, the collection $\{a, a, b\}$ can be written, e.g., as $\{(1, a), (2, a), (42, b)\}$. When denoting elements (i, v) of multisets, we often drop the indices and write simply the set element v instead of the pair (i, v). The *disjoint union* $A \uplus B$ of multisets A and B is defined as the set union $A' \cup B'$, where $A' = \{ ((i, 0), v) \mid (i, v) \in A \}$ and $B' = \{ ((i, 1), v) \mid (i, v) \in B \}$.

Temporal Logics. Let us start by recalling the syntax of CTL*, CTL, and LTL from the literature [8]. We adopt, as is common in studies on team logics, the convention that formulae are given in negation normal form. Fix a set AP of *atomic propositions*. The set of formulae of CTL* (over AP) is generated by the following grammar:

$$\varphi \Coloneqq p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathsf{X}\varphi \mid \varphi \mathsf{U} \varphi \mid \varphi \mathsf{W} \varphi \mid \exists \varphi \mid \forall \varphi,$$

where $p \in$ AP is a proposition symbol, X, U and W are temporal operators, and $\exists$ and $\forall$ are path quantifiers. CTL is the syntactic fragment of CTL*, where each temporal operator directly follows a path quantifier (and vice versa). In order to simplify the notation, we write $\mathsf{X}_\forall \varphi$, $\psi \mathsf{U}_\exists \varphi$, etc., instead of $\forall \mathsf{X} \varphi$ and $\exists \psi \mathsf{U} \varphi$. That is, the CTL syntax (over AP) is given by the grammar:

$$\varphi \Coloneqq p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathsf{X}_\exists \varphi \mid \mathsf{X}_\forall \varphi \mid \varphi \mathsf{U}_\exists \varphi \mid \varphi \mathsf{U}_\forall \varphi \mid \varphi \mathsf{W}_\exists \varphi \mid \varphi \mathsf{W}_\forall \varphi,$$

where $p \in$ AP. Finally, LTL is the syntactic fragment of CTL* without any path quantifiers. The Kripke semantics for CTL* is defined in the usual manner with respect to Kripke structures and traces generated from them [30]. For an LTL-formula φ, a trace t, and $i \in \mathbb{N}$, we write $[\![\varphi]\!]_{(t,i)}$ for the truth value of $t[i, \infty] \Vdash \varphi$ using standard LTL Kripke semantics. Here $t[i, \infty]$ is the postfix of t starting from its ith element. The logical constants $\top, \bot$ and connectives $\to, \leftrightarrow$ are defined as usual (e.g., $\bot \coloneqq p \wedge \neg p$), and $\mathsf{F}\varphi \coloneqq \top \mathsf{U} \varphi$ and $\mathsf{G}\varphi \coloneqq \varphi \mathsf{W} \bot$.

Kripke Structures. A *rooted Kripke structure* is a 4-tuple $\mathfrak{K} = (W, R, \eta, r)$, where W is a finite non-empty set of states, $R \subseteq W^2$ a left-total relation, $\eta \colon W \to 2^{\mathrm{AP}}$ a labelling function, and $r \in W$ an initial state of W. A *path* σ through a Kripke structure $\mathfrak{K} = (W, R, \eta, r)$ is an infinite sequence $\sigma \in W^\omega$ such that $\sigma[0] = r$ and $(\sigma[i], \sigma[i+1]) \in R$ for every $i \geq 0$. The *trace of* σ is defined as $t(\sigma) \coloneqq \eta(\sigma[0])\eta(\sigma[1]) \cdots \in (2^{\mathrm{AP}})^\omega$. A Kripke structure $\mathfrak{K}$ induces a multiset of traces, defined as $\mathrm{Traces}(\mathfrak{K}) = \{ (\sigma, t(\sigma)) \mid \sigma \text{ is a path through } \mathfrak{K} \}$.

Property of tef	Definition
Monotonicity	$\forall i \in \mathbb{N} : \tau(i) \leq \tau(i+1)$
Strict Mon.	$\forall i \in \mathbb{N} : \tau(i) < \tau(i+1)$
Stepwiseness	$\forall i \in \mathbb{N} : \tau(i) \leq \tau(i+1) \leq \tau(i) + \vec{1}$
*Fairness	$\forall i \in \mathbb{N}\, \forall t \in T\, \exists j \in \mathbb{N} : \tau(j, t) \geq i$
*Non-Parallelism	$\forall i \in \mathbb{N} : i = \sum_{t \in T} \tau(i, t)$
*Synchronicity	$\forall i \in \mathbb{N}\, \forall t, t' \in T : \tau(i, t) = \tau(i, t')$

Table 1: Some properties of tefs. * marks optional properties. Here, we write $\tau(i)$ to denote the tuple $\big(\tau(i, t)\big)_{t \in T}$.

3 REVISITING TEMPORAL TEAM SEMANTICS

In this section, we return to the drawing board and reconstruct the semantics of TeamLTL from scratch. By doing so, we end up also defining team semantics variants of CTL and CTL* (i.e., TeamCTL and TeamCTL*). Our starting goal is to consider hyperproperties in a setting where synchronicity of the passage of time between distinct computation traces is not presupposed. Instead, we stipulate a global lapse of time (global clock) and relate the lapse of time on computation traces (local clocks) to the lapse of time on the global clock using a concept we call *time evaluation functions*. Our approach here is similar to the one of Baumeister et al. [3] and Bonakdarpour et al. [5], where our *time evaluation functions* are called *trajectories*.

3.1 Time evaluation functions and temporal teams

Given a (possibly infinite) multiset of traces T, a *time evaluation function* (*tef* for short) for T is a function $\tau \colon \mathbb{N} \times T \to \mathbb{N}$ that, given a trace $t \in T$ and a value of the global clock $i \in \mathbb{N}$, outputs the value $\tau(i, t)$ of the local clock of trace t at global time i. Needless to say, not all functions $\tau \colon \mathbb{N} \times T \to \mathbb{N}$ satisfy properties that a function should *a priori* satisfy in order to be called a time evaluation function. We refer the reader to Table 1 for a list of tef properties considered in this paper. Intuitively, a tef is *monotonic* if the values of the local clocks only increase; *strict monotonicity* requires that at least one local clock advance in every step; *stepwiseness* refers to local clocks advancing at most one step each time; *fairness* implies that no local clock gets stuck infinitely long; *non-parallelism* forces exactly one clock to advance each time step; *synchronicity* means that all clocks advance in lockstep.

In the current paper, we are designing logics for hyperproperties of discrete linear time execution traces. It is thus clear that all tefs should at least satisfy *monotonicity.* One design principle of our setting is to use a global reference clock in addition to the local clocks of the computations. It is natural to assume that in order for a local clock to advance, the global clock has to advance as well. Thus, we stipulate that every tef must satisfy *stepwiseness.*

Finally, a crucial property that should hold for all logics with team semantics is that TeamLTL should be a conservative extension of LTL. That is, on singleton teams, TeamLTL semantics should coincide with the semantics of standard non-team-based LTL. In order, for example, for the next operator X to enjoy this invariance between LTL and TeamLTL, we stipulate *strict monotonicity* instead of simple *monotonicity* as a property for all tefs. We arrive at the following formal definitions.

Definition 3.1. A function of the type $\mathbb{N} \times T \rightarrow \mathbb{N}$ is a *stuttering tef for T* if it satisfies monotonicity, a *tef for T* if it satisfies strict monotonicity and stepwiseness, and a *synchronous tef for T* if it satisfies strict monotonicity, stepwiseness, and synchronicity.

We write $\tau(i)$ to denote the tuple $\big(\tau(i,t)\big)_{t\in T}$. A tef is *initial*, if $\tau(0,t) = 0$ for each $t \in T$. If τ is a tef and $k \in \mathbb{N}$ is a natural number, then $\tau[k,\infty]$ is the *k-shifted tef* defined by putting $\tau[k,\infty](i,t) := \tau(i+k,t)$, for every $t \in T$ and $i \in \mathbb{N}$.

Definition 3.2. A *temporal team* is a pair (T,τ), where T is a multiset of traces and τ is a tef for T. A pair (T,τ) is called a *stuttering temporal team* if τ is a stuttering tef for T.

3.2 TeamLTL, TeamCTL, and TeamCTL*

Let (T,τ) be a stuttering temporal team. Team semantics for LTL (i.e., TeamLTL) is defined recursively as follows.

$$\begin{array}{lll}
(T,\tau) \models p & \text{iff} & \forall t \in T : p \in t[\tau(0,t)] \\
(T,\tau) \models \neg p & \text{iff} & \forall t \in T : p \notin t[\tau(0,t)] \\
(T,\tau) \models (\varphi \wedge \psi) & \text{iff} & (T,\tau) \models \varphi \text{ and } (T,\tau) \models \psi \\
(T,\tau) \models (\varphi \vee \psi) & \text{iff} & \exists T_1 \uplus T_2 = T : (T_1,\tau) \models \varphi \text{ and } (T_2,\tau) \models \psi \\
(T,\tau) \models \mathsf{X}\varphi & \text{iff} & (T,\tau[1,\infty]) \models \varphi \\
(T,\tau) \models [\varphi \mathsf{U} \psi] & \text{iff} & \exists k \in \mathbb{N} \text{ such that } (T,\tau[k,\infty]) \models \psi \text{ and} \\
 & & \quad \forall m : 0 \le m < k \Rightarrow (T,\tau[m,\infty]) \models \varphi \\
(T,\tau) \models [\varphi \mathsf{W} \psi] & \text{iff} & \forall k \in \mathbb{N} : (T,\tau[k,\infty]) \models \varphi \text{ or} \\
 & & \quad \exists m \text{ s.t. } m \le k \text{ and } (T,\tau[m,\infty]) \models \psi
\end{array}$$

Note that $(T,\tau) \models \bot$ iff $T = \emptyset$. In the literature, there exist two variants of team semantics for the split operator $\vee$: strict and lax semantics. Strict semantics enforces the split to be a partition whereas lax does not. The first is the more natural version in the multiset setting which we follow here. In multiset semantics the multiplicities of traces are recorded. Having lax disjunction would blur these multiplicities. Problems of strict disjunction arise in set semantics, where multiplicities are not recorded, and thus in set semantics the lax disjunction is more natural.

If τ is an initial synchronous tef, we obtain the synchronous team semantics of LTL as defined by Krebs et al. [26].

While any given multiset of traces T induces a unique initial synchronous tef, the same does not hold for tefs in general. Consequently, two different modes of TeamLTL satisfaction naturally emerge: *existential* (a formula is satisfied by some initial tef) and *universal* (a formula is satisfied by all initial tefs) satisfaction. In the special case where a unique initial tef exists, these two modes naturally coincide.

Given a multiset of traces T and a formula $\varphi \in$ TeamLTL, we write $T \models_\exists \varphi$ if $(T,\tau) \models \varphi$ for some initial tef of T. Likewise, we write $T \models_\forall \varphi$ if $(T,\tau) \models \varphi$ for all initial tefs of T. Finally, we write $T \models_s \varphi$ if $(T,\tau) \models \varphi$ for the unique initial synchronous tef of T.

We sometimes refer to the universal and existential interpretations of satisfaction by using ∀TeamLTL and ∃TeamLTL, respectively. For referring to the synchronous interpretation, we write *synchronous* TeamLTL.

TeamCTL and TeamCTL* loan their syntax from CTL and CTL*, respectively. However, while the quantifiers ∃ and ∀ refer to path quantification in CTL and CTL*, in the team semantics setting the quantifiers range over tefs. The formal semantics of the quantifiers are as one would assume:

$$(T,\tau) \models \exists\varphi \text{ iff } (T,\tau') \models \varphi \text{ for some tef } \tau' \text{ of } T \text{ s.t. } \tau'(0) = \tau(0),$$
$$(T,\tau) \models \forall\varphi \text{ iff } (T,\tau') \models \varphi \text{ for all tefs } \tau' \text{ of } T \text{ s.t. } \tau'(0) = \tau(0).$$

We write ∃TeamCTL and ∀TeamCTL to denote the fragments of TeamCTL without the modalities $\{\mathsf{U}_\forall, \mathsf{W}_\forall, \mathsf{X}_\forall\}$ and $\{\mathsf{U}_\exists, \mathsf{W}_\exists, \mathsf{X}_\exists\}$, respectively. Likewise, we write ∃TeamCTL* and ∀TeamCTL* to denote the fragments of TeamCTL* without the quantifier ∀ and ∃, respectively. We extend the notation $\models_\exists$ and $\models_\forall$ to TeamCTL*-formulae as well.

In this paper, we consider the following decision problems for different combinations of logics $L \in$ {TeamLTL, TeamCTL, TeamCTL*} and modes of satisfaction $\models_* \in \{\models_\exists, \models_\forall, \models_s\}$.

Satisfiability: Given an $(L, \models_*)$-formula φ, is there a multiset of traces T such that $T \models_* \varphi$?

Model Checking: Given an $(L, \models_*)$-formula φ and a Kripke structure $\mathfrak{K}$, does $\text{Traces}(\mathfrak{K}) \models_* \varphi$ hold?

Path Checking: Given an $(L, \models_*)$-formula φ and a finite multiset of ultimately periodic traces T, does $T \models_* \varphi$?

3.3 Extensions of TeamLTL, TeamCTL, and TeamCTL*

Team logics can easily be extended by atoms describing properties of teams. These extensions are a well defined way to delineate the expressivity and complexity of the logics we consider. The most studied of these atoms are *dependence atoms* $\text{dep}(\varphi_1,\ldots,\varphi_n,\psi)$ and *inclusion atoms* $\varphi_1,\ldots,\varphi_n \subseteq \psi_1,\ldots,\psi_n$, where $\varphi_1,\ldots,\varphi_n,\psi,\psi_1,\ldots,\psi_n$ are propositional formulae.[1] Dependence atoms state that the truth value of ψ is functionally determined by the truth values of all $\varphi_1,\ldots,\varphi_n$. Inclusion atoms state that each value combination of $\varphi_1,\ldots,\varphi_n$ must also occur as a value combination of $\psi_1,\ldots,\psi_n$. Their formal semantics is defined as follows:

$$\begin{aligned}
(T,\tau) \models \text{dep}(\varphi_1,\ldots,\varphi_n,\psi) &\text{ iff } \forall t,t' \in T : \\
\bigwedge_{1\le j\le n} [\![\varphi_j]\!]_{(t,\tau(0,t))} &= [\![\varphi_j]\!]_{(t',\tau(0,t'))} \\
&\text{implies } [\![\psi]\!]_{(t,\tau(0,t))} = [\![\psi]\!]_{(t',\tau(0,t'))}, \\
(T,\tau) \models \varphi_1,\ldots,\varphi_n \subseteq \psi_1,\ldots,\psi_n &\text{ iff } \forall t \in T\, \exists t' \in T : \\
\bigwedge_{1\le j\le n} [\![\varphi_j]\!]_{(t,\tau(0,t))} &= [\![\psi_j]\!]_{(t',\tau(0,t'))}.
\end{aligned}$$

[1] In the team semantics literature atoms whose parameters are propositional variables are often called (proper) atoms, while extended atoms allow arbitrary formulae without atoms in their place. In [26] arbitrary LTL-formulae were allowed as parameters. Here we take a middle ground and restrict parameters of atoms to propositional formulae. One reason for this restriction is that the combination of extended atoms and time evaluation functions can have unwanted consequences in the TeamLTL setting.

We also consider other connectives known in the team semantics literature: *Boolean disjunction* $\varovee$, the *non-emptiness atom* NE, and the *universal subteam quantifier* $\overset{1}{\mathsf{A}}$, with their semantics defined as:

$$\begin{array}{lll}(T,\tau)\models\varphi\varovee\psi & \text{iff} & (T,\tau)\models\varphi \text{ or } (T,\tau)\models\psi\\ (T,\tau)\models \mathrm{NE} & \text{iff} & T\neq\emptyset\\ (T,\tau)\models\overset{1}{\mathsf{A}}\,\varphi & \text{iff} & \forall t\in T:(\{t\},\tau)\models\varphi\end{array}$$

If C is a collection of atoms and connectives, we denote by TeamLTL(C) the extension of TeamLTL with the atoms and connectives in C. For any atom or connective $\circ$, we write TeamLTL$(C,\circ)$ instead of TeamLTL$(C\cup\{\circ\})$.

It is known that, in the setting of synchronous TeamLTL, all (all downward closed, resp.) *Boolean properties* of teams are expressible in TeamLTL$(\varovee,\mathrm{NE},\overset{1}{\mathsf{A}})$ (in TeamLTL$(\varovee,\overset{1}{\mathsf{A}})$, resp.) [33]. Let B be a set of n-ary Boolean relations and $\varphi_1,\dots,\varphi_n$ propositional formulae. We define the semantics of an expression $[\varphi_1,\dots,\varphi_n]_B$ as follows:

$$(T,\tau)\models[\varphi_1,\dots,\varphi_n]_B \quad\text{iff}\quad \{\,(\llbracket\varphi_1\rrbracket_{(t,\tau(t,0))},\dots,\llbracket\varphi_n\rrbracket_{(t,\tau(t,0))})\mid t\in T\,\}\in B.$$

Expressions of the form $[\varphi_1,\dots,\varphi_n]_B$ are called *generalised atoms*. If B is downward closed (i.e. $S\in B$ whenever $S\subseteq R\in B$), it is *a downward closed generalised atom*. Dependence and inclusion atoms can also be defined as generalised atoms.

The following was proved in the setting of synchronous TeamLTL. It is, however, easy to check that the same proof works also in our more general setting.

Theorem 3.3 ([33]). *Any generalised atom is expressible in* TeamLTL$(\varovee,\mathrm{NE},\overset{1}{\mathsf{A}})$ *and all downward closed generalised atoms can be expressed in* TeamLTL$(\varovee,\overset{1}{\mathsf{A}})$.

3.4 Expressing properties of tefs

Here, we show that some of the properties labelled optional in Table 1, namely fairness and synchronicity, are indeed optional properties in some extensions of TeamLTL in the sense that these properties become definable in the extensions. Since these properties make assumptions about the progress of a tef on each trace, we need a way to track this progress. For this purpose, we introduce a fresh atomic proposition o that is set on exactly every other position on every trace. The parity induced by o then ensures that we have progressed by exactly one time step whenever the valuation flips from o to $\neg o$ or vice versa.

Expressing the alternation on o. In the context of the model checking problem, ensuring the alternation on o is straightforward. Given a Kripke structure, create two copies of every state of the structure, one labelled with o and another not labelled with o. Then, transitions from each o labelled state take to the copy of the target state not labelled o, and transitions from each state not labelled o take to the copy of the target state labelled o. In this new structure, every trace has the property that o holds on exactly every other index. Moreover, after dropping o, the two structures are indistinguishable with respect to their traces. For the satisfiability problem, the valuation of o is not restricted by any structure. Thus, we construct a formula that it is unsatisfiable by a set of traces that does not correctly alternate on o.

Expressing synchronicity. In order to express synchronicity of a tef, we encode in a formula that o alternates on all traces simultaneously. That is, for every step, either o holds on all traces and $\neg o$ holds in the next step, or vice versa: $\varphi_{\mathsf{synch}} \coloneqq \mathsf{G}((o\wedge\mathsf{X}\neg o)\varovee(\neg o\wedge\mathsf{X}o))$. Note that this formula is unsatisfiable by a set of traces violating the alternation property, since all subformulae refer to the whole set of traces. This formula shows how to encode synchronicity using the Boolean disjunction $\varovee$. We can alter the formula a little to also show that synchronicity can be expressed without this extension: $\varphi'_{\mathsf{synch}}\coloneqq o\wedge\mathsf{G}((o\wedge\mathsf{X}\neg o)\vee(\neg o\wedge\mathsf{X}o))$. In this formula, we make use of a split $\vee$ instead of the Boolean disjunction $\varovee$. However, since we demand that o hold in the first step, the split can only ever be made true by splitting a set of traces T into T and $\emptyset$. Thus, $\vee$ behaves like $\varovee$ in this formula and can replace the undesired connective. In Section 4, we make use of this formula to show how synchronous TeamLTL can be embedded into different fragments of TeamCTL*.

Expressing fairness. Fairness can be expressed using the universal subteam quantifier $\overset{1}{\mathsf{A}}$. Our formula states that for every trace, the valuation of o flips infinitely often and thus the current tef never stops making progress on this trace. This is equivalent to the definition of fairness that requires every index on every trace to be reached. The formula is: $\varphi_{\mathsf{fair}}\coloneqq\overset{1}{\mathsf{A}}\,\mathsf{G}((o\wedge\mathsf{F}\neg o)\vee(\neg o\wedge\mathsf{F}o))$. Note that for this property, we do not need to enforce strict alternation on o. We only require that the valuation of o alternates after some finite amount of steps.

Quantifying tefs with expressible properties. The formulae expressing tef properties can be used to quantify over tefs with these properties. For example, quantification over fair tefs could be implemented in the following way: if T is a multiset of traces satisfying alternation for o, we have that $(T,\tau)\models\exists(\varphi_{\mathsf{fair}}\wedge\varphi)$ iff there exists a fair tef τ' such that $\tau'(0)=\tau(0)$ and $(T,\tau')\models\varphi$.

3.5 Basic properties of the logics

TeamLTL is a conservative extension of LTL. The next proposition follows by a straightforward inductive argument that is almost identical to the corresponding proof for synchronous TeamLTL [26]:

Proposition 3.4. *For any* TeamLTL*-formula φ, trace t, and initial tef τ for $\{t\}$, the following holds: $(\{t\},\tau)\models\varphi$ iff $t\Vdash\varphi$, where $\Vdash$ denotes the standard satisfaction relation of* LTL.

Note that in the setting of the above proposition TeamCTL and TeamCTL* both collapse to LTL as well.

Let L be a logic and $\models_*\in\{\models_\exists,\models_\forall,\models_s\}$. We say that $(\mathrm{L},\models_*)$ is *downward closed* if, for every $\varphi\in\mathrm{L}$, $T\models_*\varphi$ implies $S\models_*\varphi$ whenever $S\subseteq T$. Likewise, we say that $(\mathrm{L},\models_*)$ is *union closed* if, for every $\varphi\in\mathrm{L}$, $T\uplus S\models_*\varphi$ holds whenever $T\models_*\varphi$ and $S\models_*\varphi$ hold.

It is known [26] that (TeamLTL, $\models_s$) is not union closed, but satisfies the downward closure property. We establish in Section 4 that synchronous TeamLTL can be simulated in $\forall$TeamLTL and $\exists$TeamLTL. From these results it follows that neither (TeamLTL, $\models_\exists$) nor (TeamLTL, $\models_\forall$) is union closed. However, unlike synchronous TeamLTL, the following example shows that (TeamLTL, $\models_\exists$) is not downward closed.

Example 3.5. Let $t = \{p\}\{p\}\emptyset^\omega$ be a given trace, and define two teams $T = \{(1,t),(2,t)\}$ and $S = \{(1,t)\}$ that are multisets of traces. It is easy to check that $T \models_\exists \mathsf{XX}p$, but $S \not\models_\exists \mathsf{XX}p$. The reason behind this is that in tefs at least one of its traces advances each step of the global clock (see Table 1). In team S this would yield leaving the p-labelled prefix while for T in the first step one trace can advance and in the next the other one can.

The example above also illustrates that the use of multisets of traces is essential in our logics; otherwise we would violate *locality*. The locality principle dictates that the satisfaction of a formula with respect to a team should not depend on the truth values of proposition symbols that do not occur in the formula. For this, consider a variant of this example with $t_1 = \{p,q\}\,\{p\}\,\emptyset^\omega$, $t_2 = \{p\}\,\{p\}\,\emptyset^\omega$ and $T = \{t_1,t_2\}$. Then, removing q from the traces would yield the set $S = \{t_2\}$ under non-multiset semantics and thus change the truth value of $\mathsf{XX}p$ as seen in Example 5. This shows that the use of multiset semantics is vital. The fact that multiset semantics can be used to retain locality was observed in [13]. The proof of the following proposition can be found in the extended version of the paper [18].

Proposition 3.6. (TeamLTL, $\models_\forall$) *is downward closed.*

4 FRAGMENTS OF TeamCTL* AND SYNCHRONOUS TeamLTL

In this section, we examine connections between our new temporal team logics and the older synchronous TeamLTL.

4.1 Satisfiability of ∃TeamLTL and validity of ∀TeamLTL

It is straightforward to check that the satisfiability problem of ∃TeamLTL and the validity problem of ∀TeamLTL are, in fact, equivalent to the corresponding problems of synchronous TeamLTL. To see this, first note that the synchronous tef for a multiset of traces is a tef itself. Conversely, for every tef τ for a multiset of traces T there exists another multiset of traces T_τ whose synchronous tef is indistinguishable from τ from the point of view of TeamLTL formulae.

Theorem 4.1. *Any given* TeamLTL*-formula is satisfiable in* ∃TeamLTL *if and only if it is satisfiable in synchronous* TeamLTL. *Likewise, a given* TeamLTL*-formula is valid in* ∀TeamLTL *if and only if it is valid in synchronous* TeamLTL.

In the following section, we establish that the connection between synchronous TeamLTL and ∃TeamLTL/∀TeamLTL is more profound than just a connection between the problems of satisfiability and validity. We show how model checking of extensions of synchronous TeamLTL can be efficiently embedded into ∃TeamLTL and ∀TeamLTL. These results imply that the model checking problem of extensions of ∃TeamLTL and ∀TeamLTL are at least as hard as the corresponding problem for synchronous TeamLTL. The same holds for the validity problem of ∃TeamLTL-extensions and for the satisfiability problem of ∀TeamLTL-extensions. However, we conjecture that the latter problems for ∃TeamLTL and ∀TeamLTL are harder than for synchronous TeamLTL, due to the alternation of quantification (between multisets of traces and tefs) that is taking place.

4.2 Simulating synchronous TeamLTL with fragments of TeamCTL*

We show how to embed synchronous TeamLTL into extensions of different subfragments of TeamCTL*: ∃TeamLTL, ∀TeamLTL, ∃TeamCTL and ∀TeamCTL. This is done by using and expanding on the idea from Subsection 3.4 to use a proposition o with alternating truth values on all traces of a team to track progress. We define translations $\varphi \mapsto \varphi^+$ from synchronous TeamLTL to fragments of TeamCTL* that are used in the embeddings. The translations are designed such that $T \models_s \varphi$ if and only if $T_o \models_* \varphi^+$ (for fitting $* \in \{\forall,\exists\}$), where T_o is obtained from T by introducing a fresh alternating proposition o. Some of the translations additionally preserve satisfiability, i.e., φ is satisfiable if and only if φ^+ is.

Let us now formalise our results a bit more. Given a set of traces T over AP, let T_o for $o \notin \mathsf{AP}$ be the set of traces $\{\, t \mid t\upharpoonright_{\mathsf{AP}} \in T \text{ and } o \in t[i] \text{ iff } i \mod 2 = 0 \,\}$ over $\mathsf{AP} \uplus \{o\}$. Here, we use $t\upharpoonright_{\mathsf{AP}}$ to denote the restriction of t to AP.

Theorem 4.2. *Given a synchronous* TeamLTL *formula* φ, *one can construct in time linear in* $|\varphi|$ *a formula* φ^+ *in* ∃TeamLTL *(resp.,* ∃TeamCTL($\varovee$)*) and* φ^- *in* ∀TeamLTL($\varovee$, NE) *(resp.,* ∀TeamCTL($\subseteq$)*) such that for all multisets of traces* T:

$$T \models_s \varphi \textit{ iff } T_o \models_\exists \varphi^+ \quad \textit{and} \quad T \models_s \varphi \textit{ iff } T_o \models_\forall \varphi^-.$$

Proof. First, the embedding into ∃TeamLTL and ∀TeamLTL. For ∃TeamLTL, we can use one of the formulae φ_{synch} or $\varphi'_{\mathsf{synch}}$ from Subsec. 3.4. They ensure that the existentially quantified tef is synchronous and therefore progresses on a set of traces in the same way a set of traces would make progress in the synchronous setting. We translate a synchronous TeamLTL formula φ into ∃TeamLTL in the following way: $\varphi \mapsto \varphi \wedge \vartheta$, where ϑ can be either φ_{synch} or $\varphi'_{\mathsf{synch}}$.

For ∀TeamLTL, we use a dual approach. Rather than identifying a synchronous tef, we instead eliminate all non-synchronous ones. We make use of the formula $\varphi_{\mathsf{off}} := \big((\mathsf{NE} \wedge o \wedge \mathsf{X}o) \vee \top\big) \varovee \big((\mathsf{NE} \wedge \neg o \wedge \mathsf{X}\neg o) \vee \top\big)$. The formula $\mathsf{F}\varphi_{\mathsf{off}}$ expresses that in the current tef, there is a *defect* where some of the traces in the set of traces do not move for one step. Using this, we can rule out all non-synchronous tefs from the universal quantifier. Our translation is: $\varphi \mapsto \varphi \vee \mathsf{F}\varphi_{\mathsf{off}}$. Note that this formula does not express that the alternation on o is correct in the set of traces. This indeed proves to be difficult in this setting since the formula has to work for all tefs and thus, to establish the alternation on o, no assumption about a tef's progress can be made. Thus, the formula presented here only works in the setting of model checking.

Now we consider the embedding into ∃TeamCTL and ∀TeamCTL. For an embedding into TeamCTL, we have to expand on the ideas used for ∃TeamLTL further. Since we make use of the quantified versions of the modalities here, we have to find a new formula that expresses the alternation on o. Also, since we do not have a global tef that can be checked for synchronicity anymore, we need to find a translation that ensures synchronicity for each modality. Consider the following formulae:

$$\psi_{\mathsf{synch}} := (o \wedge \mathsf{X}_\exists \neg o) \varovee (\neg o \wedge \mathsf{X}_\exists o),$$
$$\psi'_{\mathsf{synch}} := (o \wedge \mathsf{X}_\exists \neg o) \vee (\neg o \wedge \mathsf{X}_\exists o),$$

$$\psi''_{\mathsf{synch}} \coloneqq \big(o \wedge \mathsf{X}_\forall(\neg o \vee o \subseteq \neg o)\big) \vee \big(\neg o \wedge \mathsf{X}_\forall(o \vee o \subseteq \neg o)\big).$$

By using these formulae, we can express the alternation on o using the TeamCTL modalities: $\mathsf{G}_\exists\psi_{\mathsf{synch}}$ expresses this property directly. The formulae $o \wedge \mathsf{G}_\forall\psi'_{\mathsf{synch}}$ and $o \wedge \mathsf{G}_\forall\psi''_{\mathsf{synch}}$ are variants of this formula, that do not use $\circledvee$. Finally, if we impose fairness for time evaluation functions, then the formula $o \wedge \mathsf{G}_\exists\psi'_{\mathsf{synch}}$ yields the same effect as well. Thus, we have a formula that expresses the alternation of o in $\forall$TeamCTL($\subseteq$) and $\exists$TeamCTL($\circledvee$), and, if we impose fairness, also in $\exists$TeamCTL. Note that these formulae are only needed as conjuncts to make a formula unsatisfiable by teams violating the property. For model checking, we can directly encode the property into the structure as sketched earlier.

We start with the embedding into $\exists$TeamCTL. Compared to the embedding into TeamLTL where the time evaluation function is constant throughout the formula and thus can be checked for synchronicity via φ_{synch}, we have to deal with newly quantified time evaluation functions for each operator in the embedding into TeamCTL. This is done by enforcing synchronicity in the translation of every operator. We use a function $(\_)^*$ that replaces all modalities in a formula with a synchronous variant and leaves atomic propositions and Boolean connectives unchanged. For the non-trivial cases, the translation is defined as follows:

$$\begin{aligned}
(\mathsf{F}\varphi)^* &\coloneqq [\mathsf{dep}(o)\mathsf{U}_\exists(\varphi)^* \wedge \mathsf{dep}(o)],\\
(\mathsf{G}\varphi)^* &\coloneqq [(\varphi)^* \wedge \mathsf{dep}(o)\mathsf{W}_\exists\bot],\\
(\varphi\mathsf{U}\psi)^* &\coloneqq [(\varphi)^* \wedge \mathsf{dep}(o)\mathsf{U}_\exists(\psi)^* \wedge \mathsf{dep}(o)],\\
(\mathsf{X}\varphi)^* &\coloneqq \mathsf{X}_\exists\big(\mathsf{dep}(o) \wedge (\varphi)^*\big),\\
(\varphi\mathsf{W}\psi)^* &\coloneqq [(\varphi)^* \wedge \mathsf{dep}(o)\mathsf{W}_\exists(\psi)^* \wedge \mathsf{dep}(o)].
\end{aligned}$$

The translation for a synchronous TeamLTL formula φ into $\exists$TeamCTL($\circledvee$) (note that dependence atoms can be defined using $\circledvee$) is then $\varphi \mapsto (\varphi)^* \wedge \theta$, where θ is one of the formulae $\mathsf{G}_\exists\psi_{\mathsf{synch}}$ or $o \wedge \mathsf{G}_\exists\psi'_{\mathsf{synch}}$ (if fairness for time evaluation is presupposed).

For the embedding into $\forall$TeamCTL($\subseteq$), we use the same ideas as for the embedding into $\exists$TeamCTL($\circledvee$). The only difference here is that we have to construct a different version of the function $(\_)^*$ that makes use of the universally quantified instead of the existentially quantified modalities. It is given as follows:

$$\begin{aligned}
(\mathsf{F}\varphi)^* &\coloneqq [\top\mathsf{U}_\forall(\varphi)^* \vee o \subseteq \neg o]\\
(\mathsf{G}\varphi)^* &\coloneqq [(\varphi)^*\mathsf{W}_\forall o \subseteq \neg o]\\
(\mathsf{X}\varphi)^* &\coloneqq \mathsf{X}_\forall\big(o \subseteq \neg o \vee (\varphi)^*\big)\\
(\varphi\mathsf{U}\psi)^* &\coloneqq [(\varphi)^*\mathsf{U}_\forall(\psi)^* \vee o \subseteq \neg o]\\
(\varphi\mathsf{W}\psi)^* &\coloneqq [(\varphi)^*\mathsf{W}_\forall(\psi)^* \vee o \subseteq \neg o]
\end{aligned}$$

The translation then is $\varphi \mapsto (\varphi)^* \wedge o \wedge \mathsf{G}_\forall\psi''_{\mathsf{synch}}$. □

Apart from the previous translation and the corresponding theorem, we also make use of the synchronous $\exists$TeamCTL modalities in the proof of Theorem 5.2. There, we use $\varphi\mathsf{U}_\sigma\psi$ for $\varphi \wedge \mathsf{dep}(o)\mathsf{U}_\exists\psi \wedge \mathsf{dep}(o)$, $\mathsf{X}_\sigma\varphi$ for $\mathsf{X}_\exists\varphi \wedge \mathsf{dep}(o)$ etc.

In conjunction with the considerations about establishing the alternation on o, we obtain the following two corollaries:

Corollary 4.3. *Model checking for synchronous* TeamLTL *can be reduced in linear time in the given formula length and model to model checking for* $\exists$TeamLTL, $\forall$TeamLTL($\circledvee$,NE)*,* $\exists$TeamCTL($\circledvee$) *and* $\forall$TeamCTL($\subseteq$).

Corollary 4.4. *Satisfiability for synchronous* TeamLTL *can be reduced in linear time in the given formula length to satisfiability for* $\exists$TeamLTL, $\exists$TeamCTL($\circledvee$) *and* $\forall$TeamCTL. *Assuming fairness of tefs, this also holds for* $\exists$TeamCTL($\subseteq$).

5 TEAMCTL($\circledvee$) IS HIGHLY UNDECIDABLE

We show how to obtain high undecidability by encoding recurrent computations of *non-deterministic 2-counter machines (N2C)*. A non-deterministic 2-counter machine M consists of a list I of n instructions that manipulate two counters (ℓeft and $r$ight) C_ℓ and C_r. All instructions are in one of the following three forms:

$$\begin{gathered}
C_a^+ \textbf{ goto } \{j, j'\},\ \text{or } C_a^- \textbf{ goto } \{j, j'\}, \textit{or}\\
\textbf{if } C_a = 0 \textbf{ goto } j \textbf{ else goto } j',
\end{gathered}$$

where $a \in \{\ell, r\}$, $0 \leq j, j' < n$. A *configuration* is a tuple (i, j, k), where $0 \leq i < n$ is the next instruction to be executed, and $j, k \in \mathbb{N}$ are the current values of the counters C_ℓ and C_r. The execution of the instruction $i\colon C_a^+$ **goto** $\{j, j'\}$ ($i\colon C_a^-$ **goto** $\{j, j'\}$, resp.) increments (decrements, resp.) the value of the counter C_a by 1. The next instruction is selected nondeterministically from the set $\{j, j'\}$. The instruction $i\colon$ **if** $C_a = 0$ **goto** j, **else goto** j' checks whether the value of the counter C_a is currently 0 and proceeds to the next instruction accordingly. The *consecution relation* of configurations is defined as usual. A *computation* is an infinite sequence of consecutive configurations starting from the initial configuration $(0, 0, 0)$. A computation is *b-recurring* if the instruction labelled b occurs infinitely often in it.

Theorem 5.1 ([2]). *Deciding whether a given non-deterministic 2-counter machine has a b-recurring computation for a given label b is* Σ^1_1*-complete.*

We reduce the existence of a b-recurring computation of a given N2C machine M and an instruction label b to the model checking problem of $\exists$TeamCTL($\circledvee$).

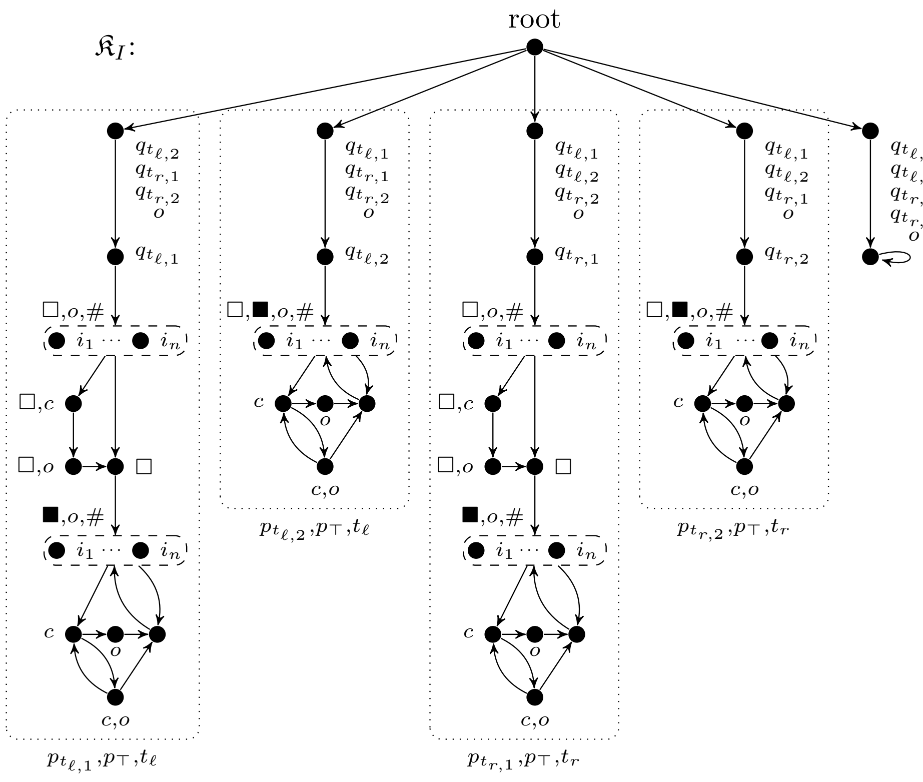
Theorem 5.2. *Model checking for* $\exists$TeamCTL($\circledvee$) *is* Σ^1_1*-hard.*

Proof. Let I be a given set of instructions of a 2-counter machine M with the set of labels $\mathbb{I} \coloneqq \{i_1, \ldots, i_n\}$ for $n \in \mathbb{N}$, and an instruction label $b \in \mathbb{I}$. We construct a TeamCTL($\circledvee$)-formula $\varphi_{I,b}$ and a Kripke structure $\mathfrak{K}_I$ such that

$$\mathrm{Traces}(\mathfrak{K}_I) \models_\exists \varphi_{I,b} \text{ iff } M \text{ has a } b\text{-recurring computation.} \qquad (1)$$

From $\mathfrak{K}_I$ one can obtain all sequences of configurations (even those which are not consecutive computations) for the machine M. The formula $\varphi_{I,b}$ then allow us to pick some particular traces generated from the structure. This essentially corresponds to existential quantification of the computation. The Kripke structure $\mathfrak{K}_I$ is depicted in Fig. 1.

Intuitively, the structure is partitioned into five parts. The two left-most parts (see Fig. 1) encode values of the left counter, the two following parts encode values of the right counter, and the



$\mathfrak{K}_I$:
root
$q_{t_{\ell,2}}$ $q_{t_{r,1}}$ $q_{t_{r,2}}$ o
$q_{t_{\ell,1}}$
$\square$,o,#
i_1 ⋯ i_n
$\square$,c
$\square$,o
$\square$
$\blacksquare$,o,#
c
o
c,o
$p_{t_{\ell,1}}$,$p_\top$,t_ℓ
$q_{t_{\ell,1}}$ $q_{t_{r,1}}$ $q_{t_{r,2}}$ o
$q_{t_{\ell,2}}$
$\square$,$\blacksquare$,o,#
$p_{t_{\ell,2}}$,$p_\top$,t_ℓ
$q_{t_{\ell,1}}$ $q_{t_{\ell,2}}$ $q_{t_{r,2}}$ o
$q_{t_{r,1}}$
$p_{t_{r,1}}$,$p_\top$,t_r
$q_{t_{\ell,1}}$ $q_{t_{\ell,2}}$ $q_{t_{r,1}}$ o
$q_{t_{r,2}}$
$p_{t_{r,2}}$,$p_\top$,t_r
$q_{t_{\ell,1}}$ $q_{t_{\ell,2}}$ $q_{t_{r,1}}$ $q_{t_{r,2}}$ o

Figure 1: Kripke structure $\mathfrak{K}_I$ used in proof of Theorem 5.2. Dotted boxes mean that the propositions below it are labelled in every state within the box. Dashed boxes with states having labels i_j for $1 \le j \le n$ simplify the presentation as follows: the incoming/outgoing edges to these boxes are connected with every vertex in the dashed box; the propositions ($\square$, o, #, $\blacksquare$) above dashed boxes are labelled at every state in the box. We call the trace generated via the rightmost part π_d.

right-most part is a "dummy trace" π_d (we come to an explanation for π_d a bit later). Every trace that is different from π_d has a proposition $p_\top$ labeled in every state after the root. We are using four *types* of traces: two trace (types) $t_{\ell,1}, t_{\ell,2}$ (the first and second from left in Fig. 1) for counter C_ℓ and two trace (types) $t_{r,1}, t_{r,2}$ (the third and fourth from left in Fig. 1) for counter C_r. Intuitively, $t_{s,2}$ encodes the current value for the counter C_s, for $s \in \{r, \ell\}$. Our construction ensures that trace $t_{s,1}$ is always one step ahead of $t_{s,2}$. We enforce that they have the same c labelling and therefore the counter value is carried from one # position to the next (subject to an increment/decrement operation). Such trace pairs are also called ℓ-traces (or r-traces, respectively) and globally have a proposition t_ℓ (resp., t_r) labelled in their non-root states. Each such trace type $t \in \mathbb{T} := \{t_{\ell,1}, t_{\ell,2}, t_{r,1}, t_{r,2}\}$ also has a proposition p_t that is globally true everywhere (except in the root) and another proposition q_t that is true in the second state while the other three $q_{t'}$ for $t' \in \mathbb{T} \setminus \{t\}$ are true in the first state (this is used for identification purposes). These trace-pairs are used to simulate incrementing, resp., decrementing the value of the respective counter. The value $m \in \mathbb{N}$ of a counter is simulated via a #-symbol divided sequence of states in a trace where m states contain a proposition c. This is depicted in Fig. 2.

The alternating o-labels on $\mathfrak{K}_I$-states in combination with strict-monotonicity of tefs, allow for the definition of variants of U- F-, G- and X-operators that are evaluated in our setting in a synchronous way:

$$\begin{aligned}[][\varphi \mathsf{U}_\sigma \psi] &:= [\mathsf{dep}(o) \land \varphi \mathsf{U}_\exists \mathsf{dep}(o) \land \psi],\\ \mathsf{F}_\sigma \varphi &:= [\top \mathsf{U}_\sigma \varphi],\\ \mathsf{G}_\sigma \varphi &:= \mathsf{G}_\exists(\mathsf{dep}(o) \land \varphi),\\ \mathsf{X}_\sigma \varphi &:= \mathsf{X}_\exists(\mathsf{dep}(o) \land \varphi).\end{aligned}$$

Initially, Traces($\mathfrak{K}_I$) contains all possible combinations and sequences of counter modifications. Intuitively, we sort these traces into five groups of $\mathbb{T}$ plus the dummy trace. In the first step, we need to cut this tremendous number of traces down to four traces (plus the dummy trace). This is realised by the formula $\varphi_{I,b}$ that is a split into two subformulae (after we synchronously make one step):

$$\varphi_{I,b} := \mathsf{X}_\sigma(p_\top \lor (\varphi_{\text{struc}} \land \mathsf{X}_\sigma \mathsf{X}_\sigma \varphi_{\text{comp}})).$$

Note that π_d has to be always split to the right side (as it has not $p_\top$ labelled) and thereby prevents an empty split on that side. The vast majority of unconsidered traces are assigned to the left side of the split.

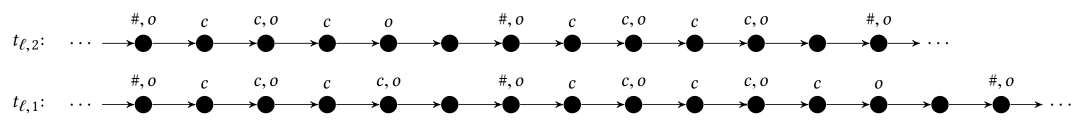


Figure 2: Simulation of a N2C via traces as used in the proof of Theorem 5.2. For presentation reasons, some of the labelled propositions have been omitted. The excerpt of the traces shows that the counter C_ℓ is incremented first from 3 to 4 and then to 5. Note that $t_{\ell,2}$ encodes the current value for the counter C_ℓ.

We need to define an auxiliary formula

$$\varphi_{\text{single}}(S) := \mathsf{G}_\sigma \bigwedge_{p\in S} \text{dep}(p)$$

expressing, that a particular trace team agrees on each time step with respect to the set of propositions in S. Now, we turn to the right formula that is a conjunction of two formulae:

$$\varphi_{\text{struc}} := \bigwedge_{t\in\mathbb{T}} \mathsf{X}_\exists q_t \wedge \left[(t_\ell \wedge \varphi_{\text{single}}(S)) \otimes (t_r \wedge \varphi_{\text{single}}(S)) \vee \neg p_\top\right],$$

where $S := \{\#, c\} \cup \mathbb{I}$. The first conjunct of this formula ensures that we are dealing with at least one trace of each type of $t \in \mathbb{T}$ (the trace π_d and every different type $t' \neq t$ is 'paused'). The second conjunct splits away π_d and groups the traces according to the left and right counter. There, it forces that $t_{s,1}$ has the same labelling (with respect to #, c and instruction labels from $\mathbb{I}$) as $t_{s,2}$ for $s \in \{r, \ell\}$. This together with the construction of $\mathfrak{K}_I$ ensures that we are dealing with exactly four traces.

The next formula, φ_{comp}, is used to desynchronise trace $t_{s,1}$ from $t_{s,2}$ (for $s \in \{\ell, r\}$) by exactly one #-interval to enable enforcing the de/increment-operation later. Again, we first split π_d away and then say that the label b is occurring infinitely often via $\theta_{\text{brec}} := \mathsf{G}_\exists \mathsf{F}_\exists b$, then desynchronise with the Until operator and say that the computation is valid. Note that it is crucial to use $\mathsf{U}_\exists$ for desynchronising the traces (compare Fig. 1: $t_{s,2}$ has to stay at the first #, while $t_{s,1}$ advances to the second #; so, this is handled via the □ together with the ■). The i_1 at the end ensures that the first instruction of I is executed first:

$$\varphi_{\text{comp}} := \neg p_\top \vee ([\square \mathsf{U}_\exists \blacksquare \wedge \theta_{\text{valid}}] \wedge \theta_{\text{brec}} \wedge i_1).$$

In the following, we define the formulae required for implementing the instructions of the N2C machine; in this context, we write $s = \ell$ and $\bar{s} = r$ or vice versa. When we increment/decrement a counter, we need a formula that makes sure that on the traces for the other counter the counter value stays the same (hence $t_{\bar{s},1}$ and $t_{\bar{s},2}$ have the same next #-interval with respect to c):

$$\text{halt} := \mathsf{X}_\sigma[\text{dep}(c) \wedge \neg\# \mathsf{U}_\sigma \#].$$

Now, we can define the formula that states incrementation of the counter C_s. Notice that the c in the left part of the synchronous Until makes sure that the counter value differs by exactly one. The synchronous Until operator matches the number of c-labelled states on both s-traces and then verifies that the count for the first trace contains one more c:

$$C_s\text{-inc} := \big(\mathsf{X}_\sigma[\, c\, \mathsf{U}_\sigma(p_{t_{s,2}} \wedge \neg c) \vee (p_{t_{s,1}} \wedge c \wedge \mathsf{X}_\sigma \neg c)]\big) \vee (\text{halt} \wedge t_{\bar{s}}).$$

Symmetrically, we can define a decrement operation:

$$C_s\text{-dec} := \big(\mathsf{X}_\sigma[\, c\, \mathsf{U}_\sigma(p_{t_{s,2}} \wedge c \wedge \mathsf{X}_\sigma \neg c) \vee (p_{t_{s,1}} \wedge \neg c)]\big) \vee (\text{halt} \wedge t_{\bar{s}}).$$

Now, we define the formulae for the possible instructions.

i: C_s^+ **goto** $\{j, j'\}$**:** The following formula ensures, that we increase the counter C_s and then reach the next #, where either j or j' is uniformly true.

$$\theta_i := C_s\text{-inc} \wedge \mathsf{X}_\sigma([\neg\# \mathsf{U}_\exists \# \wedge (j \otimes j')])$$

i: C_s^- **goto** $\{j, j'\}$**:** As before, jump to the next # and have j or j' uniformly after decreasing the counter C_s.

$$\theta_i := C_s\text{-dec} \wedge \mathsf{X}_\sigma([\neg\# \mathsf{U}_\exists \# \wedge (j \otimes j')])$$

i: **if** $C_s = 0$ **goto** j**, else goto** j'**:** Here, we use the Boolean disjunction for distinguishing the part of the if-then-else construct. Recall that $t_{s,2}$ encodes the current counter value of C_s. The halt-formula ensures that the counter values stays the same:

$$\theta_i := \mathsf{X}_\sigma\Big(\big(((q_{t_{s,2}} \wedge \neg c) \vee (\neg q_{t_{s,2}})) \wedge [\neg\# \mathsf{U}_\exists \# \wedge j]\big) \otimes \big(((q_{t_{s,2}} \wedge c) \vee (\neg q_{t_{s,2}})) \wedge [\neg\# \mathsf{U}_\exists \# \wedge j']\big)\Big) \wedge \text{halt}.$$

Next, we need a formula stating that only the label $j \in \mathbb{I}$ is true:

$$\text{only}(j) := j \wedge \bigwedge_{i\neq j} \neg i.$$

Now, let θ_{valid} state that each #-labelled position encodes the correct instruction of the N2C machine. Note that due to dep(#) the encoding of the traces remains in synch:

$$\theta_{\text{valid}} := \mathsf{G}_\exists\left(\text{dep}(\#) \wedge \left(\left(\# \wedge \bigvee_{i<n} (\text{only}(i) \wedge \theta_i)\right) \vee \neg\#\right)\right).$$

The length of the formula $\varphi_{I,b}$ is linear in $|\mathbb{I}|$.

The direction "$\Leftarrow$" of Eq. (1) follows by construction of $\varphi_{I,b}$ and $\mathfrak{K}_I$. For the other direction, note that if $\varphi_{I,b}$ is satisfied then this means the label b needs to appear infinitely often and the encoded computation of the N2C machine is valid. □

6 TRANSLATING FROM TeamCTL* TO AABA

In the previous section, we showed that model checking is highly undecidable even for a very restricted fragment of TeamCTL*. This motivates the search for decidable restrictions. For this purpose, we first introduce Alternating Asynchronous Büchi Automata (AABA).

Let $M = \{1, 2, \dots, n\}$ be a set of directions and Σ an input alphabet. An *Alternating Asynchronous Büchi Automaton (AABA)* [20] is a tuple $\mathcal{A} = (Q, \rho^0, \rho, F)$ where Q is a finite set of states, $\rho^0 \in \mathcal{B}^+(Q)$ is a positive Boolean combination of initial states, $\rho\colon Q \times \Sigma \times M \to \mathcal{B}^+(Q)$ maps triples of control locations, input symbols and directions to positive Boolean combinations of control locations, and $F \subseteq Q$ is a set of final states. A *tree* $\mathfrak{T}$ is a subset of $\mathbb{N}^*$ such that for every node $\mathrm{t} \in \mathbb{N}^*$ and every positive integer $n \in \mathbb{N}$: $\mathrm{t} \cdot n \in \mathfrak{T}$ implies (i) $\mathrm{t} \in \mathfrak{T}$ (we then call $\mathrm{t} \cdot n$ a *child* of t), and (ii) for every $0 < m < n$, $\mathrm{t} \cdot m \in \mathfrak{T}$. We assume every node has at least one child. A *path* in a tree $\mathfrak{T}$ is a sequence of nodes $\mathrm{t}_0\mathrm{t}_1\ldots$ such that $\mathrm{t}_0 = \varepsilon$ and t_{i+1} is a child of t_i for all $i \in \mathbb{N}$. A *run* of an AABA $\mathcal{A}$ over n infinite words $w_1, \ldots, w_n \in \Sigma^\omega$ is defined as a Q-labeled tree $(\mathfrak{T}, r)$ where $r : \mathfrak{T} \to Q$ is a labelling function. Additionally, for each $\mathrm{t} \in \mathfrak{T}$, we have n offset counters $c_1^{\mathrm{t}}, \ldots, c_n^{\mathrm{t}}$ starting at $c_i^{\mathrm{t}} = 0$ for all i and t with $|\mathrm{t}| \leq 1$. Together, the labelling function and offset counters satisfy the following conditions:

(i) we have $\{ r(\mathrm{t}) \mid \mathrm{t} \in \mathfrak{T}, |\mathrm{t}| = 1 \} \models \rho^0$, and
(ii) when node $\mathrm{t} \in \mathfrak{T} \setminus \{\varepsilon\}$ has children $\mathrm{t}_1, \ldots, \mathrm{t}_k$, then there is a $d \in M$ such that
  (a) $c_d^{\mathrm{t}_i} = c_d^{\mathrm{t}} + 1$ and $c_{d'}^{\mathrm{t}_i} = c_{d'}^{\mathrm{t}}$ for all i and $d' \neq d$,
  (b) we have $1 \leq k \leq |Q|$, and
  (c) the valuation assigning true to $r(\mathrm{t}_1), \ldots, r(\mathrm{t}_k)$ and false to all other states satisfies $\rho(r(\mathrm{t}), w_d(c_d^{\mathrm{t}}), d)$.

A run $(\mathfrak{T}, r)$ is an *accepting run* iff for every path $\mathrm{t}_0\mathrm{t}_1\ldots$ in $(\mathfrak{T}, r)$, a control location $q \in F$ occurs infinitely often. We say that $(w_1, \ldots, w_n)$ is *accepted* by $\mathcal{A}$ iff there is an accepting run of $\mathcal{A}$ on $w_1, \ldots, w_n$. The *set of tuples of infinite words* accepted by $\mathcal{A}$ is denoted by $\mathcal{L}(\mathcal{A})$.

We also use AABA with a generalised Büchi acceptance condition which we call *Generalised Alternating Asynchronous Büchi Automata (GAABA)*. Formally, a GAABA is an AABA in which the set $F = \bigcup F_i$ consists of several acceptance sets F_i and the Büchi acceptance condition is refined such that a run $(\mathfrak{T}, r)$ is accepting iff every path visits a state in every set F_i infinitely often. A GAABA can be translated to an AABA with quadratic blowup similar to the standard translation from Generalised Büchi Automata to Büchi Automata [12]. For this purpose, we first annotate states with the index i of the next set F_i that requires a visit of an accepting state. We then consecutively move to the states annotated with the next index if such an accepting state is seen. Only those states are declared accepting that indicate that an accepting state in every F_i has been visited.

Let $\mathcal{S} := \{\circledvee, \mathrm{NE}, \overset{1}{\mathrm{A}}, \mathrm{dep}, \subseteq\}$. We now present a translation of TeamCTL$^*(\mathcal{S})$ to AABA over a fixed number $n \in \mathbb{N}$ of traces. Note that Theorem 3.3 would allow us to drop 'dep' and '$\subseteq$' from the translation, but the cost would be an exponential blowup, that does not occur in our direct translation. We first describe the translation for $\exists$TeamCTL$^*(\mathcal{S})$ formulae with $\models_\exists$ mode of satisfaction and then explain how this translation can be lifted to the full logic TeamCTL$^*(\mathcal{S})$. Our translation is obtained by first constructing a suitable Generalized Alternating Asynchronous Büchi Automaton (GAABA) and then translating that GAABA to an AABA as described before. For this section, it is convenient to use a different but equivalent version of the semantics of TeamCTL*. In this version, we add the global time step to the left side of the satisfaction relation $\models$ and write $(T, \tau, i) \models \varphi$ instead of $(T, \tau) \models \varphi$. When progressing with modalities, tefs are not *cut off* as in our current definition; instead they are evaluated at a later global time step, just as in the semantics definition of synchronous TeamLTL [33]. The definition of the satisfaction relation for quantifiers is then adjusted accordingly: $(T, \tau, i) \models \exists\varphi$ iff there is a tef τ' with $\tau'(j) = \tau(j)$ for all $j \leq i$ such that $(T, \tau', i) \models \varphi$. This definition allows us to define corresponding restrictions for the semantics of AABA and our logics.

Let $(\mathfrak{T}, r)$ be a run of an AABA over $n \in \mathbb{N}$ words. For every path $\pi \in (\mathfrak{T}, r)$, we denote by τ_π the time evaluation function obtained by setting $\tau_\pi(i) := (c_1^{\pi(i)}, \ldots, c_n^{\pi(i)})$. The set $I_{(\mathfrak{T},r)} := \{ \tau_\pi \mid \pi$ is a path in $(\mathfrak{T}, r) \}$ is the set of *induced tefs* for a run $(\mathfrak{T}, r)$. For an AABA $\mathcal{A}$ and a set of tefs TE, the *language of $\mathcal{A}$ restricted to TE* is given by $\mathcal{L}_{TE}(\mathcal{A}) := \{ w = (w_1, \ldots, w_n) \in (\Sigma^\omega)^n \mid \exists(\mathfrak{T}, r) : (\mathfrak{T}, r)$ is an accepting run of $\mathcal{A}$ on w and $I_{(\mathfrak{T},r)} \subseteq TE \}$. The *TE emptiness problem for an AABA $\mathcal{A}$* is to check whether $\mathcal{L}_{TE}(\mathcal{A})$ is empty.

Given a team (T, τ) and $k \in \mathbb{N}$, the tef τ is *k-synchronous* iff for all $i \in \mathbb{N}$ and $t, t' \in T$, $|\tau(i, t) - \tau(i, t')| \leq k$. The term k-synchronicity indicates that traces are not allowed to diverge by more than k steps during the execution. Let $\mathrm{switch}_\tau(i) := |\{ t \in T \mid \tau(i-1, t) = \tau(i, t), \tau(i, t) \neq \tau(i+1, t)\}|$ for $i \geq 1$ and $\mathrm{switch}_\tau(0) := 0$ be the number of context switches performed between indices i and $i + 1$. A tef τ is *k-context-bounded* iff $\mathrm{switch}_\tau(i) \leq 1$ for all $i \in \mathbb{N}$ and $\sum_{i \in \mathbb{N}} \mathrm{switch}_\tau(i) \leq k$. The term k-context-boundedness states that only a single trace is allowed to progress on each global time step and we can only switch between different traces at most k times. For AABA, the following holds.

Theorem 6.1 ([20, Cors. 3.6&3.13, Thms. 3.12&3.17]).

(1) The emptiness problem for AABA is undecidable.
(2) The k-synchronous emptiness problem for AABA with n traces is EXPSPACE*-complete and is* PSPACE*-complete for fixed n.*
*(3) The k-context-bounded emptiness problem for AABA is $(k-2)$-*EXPSPACE*-complete.*

Note that the hardness results in [20] were formulated for AAPA, not AABA. However, the proofs rely only on reachability and not on a parity acceptance condition. Thus, they carry over to AABA as well.

We now define a restricted semantics for our logics similar to the restricted semantics for AABA. For a team (T, τ), a TeamCTL* formula φ and a set of tefs TE with $\tau \in TE$, we write $(T, \tau) \models_{TE} \varphi$ to denote that (T, τ) satisfies φ under a semantics in which quantifiers $\exists$ and $\forall$ range only over tefs in TE. This is straightforwardly extended to modes of satisfaction $\models_{*, TE}$ for $* \in \{\exists, \forall\}$. The *fixed size TE satisfiability problem* fs-SAT(TE) is then to decide for a given natural number n and a (TeamCTL*, $\models_*$) formula φ whether there exists a multiset of traces T with $|T| = n$ such that $T \models_{*, TE} \varphi$. The *fixed size TE model checking problem* fs-MC(TE) is to decide for a Kripke model, a (TeamCTL*, $\models_*$) formula φ and natural number n whether all multisets of traces $T \subseteq \mathrm{Traces}(\mathfrak{K})$ of size n satisfy φ under TE semantics. The *TE path checking problem* PC(TE) is defined analogously. We set $\mathcal{L}_{TE}^n(\varphi) := \{ T \mid |T| = n \wedge T \models_{*, TE} \varphi \}$ to be the set of finite multisets of traces of size n satisfying φ under the TE semantics.

We now describe the translation from $\exists$TeamCTL$^*(\mathcal{S})$ formulae to GAABA over teams of fixed size n. We also only consider the $\models_\exists$ mode of satisfaction. After that we show how to generalise the construction to the full logic TeamCTL$^*(\mathcal{S})$. This translation is based on the classical Fischer-Ladner construction translating LTL formulae to non-deterministic Büchi automata [12]. We construct a GAABA $\mathcal{A}_\psi$ inductively over the quantification depth such that

$\mathcal{A}_\psi$ has an accepting run over the input multiset of traces under *TE* semantics if and only if there is a tef such that ψ is fulfilled by the input multiset of traces under *TE* semantics. This allows us to apply decidability results for AABA under restricted semantics, especially those from [20], to decision problems for TeamCTL$^*(\mathcal{S})$.

Let us explain this construction in turn. In general, we perform the standard Fischer-Ladner construction for LTL, i.e., we annotate states with the formulae that should hold whenever an accepting run visits the respective states. The transition function is then defined such that the requirements for the successor induced by these formulae are met. Finally, we pick accepting sets such that in an accepting run, $\psi_1 \mathsf{U} \psi_2$-formulae are either not required at a certain time step or the run must eventually visit a state containing ψ_2. The indices associated with formulae indicate the traces of the team which we consider for the respective formula. There are three additional problems that are not captured by the standard construction: asynchronicity, quantifiers and non-standard logical operators that are not present in LTL, i.e. the split operator and additional team semantics atoms.

In a GAABA, time evaluation functions with asynchronous progress on different traces can be modelled straightforwardly by asynchronous moves. In every step, we non-deterministically pick the traces to be advanced, ensure that the atomic propositions in the next state of $\mathcal{A}_\varphi$ match those given by the next index of that trace and maintain the atomic propositions for all other traces. W.l.o.g., we assume that only one trace is advanced in each time step by a given tef and it will be clear how to extend the construction to general tefs. In order to handle quantifiers, we use conjunctive alternation. For every existential subformula $\exists\psi$ contained in a state q, we guess an initial state of $\mathcal{A}_\psi$ which agrees with q on the atomic propositions and then conjunctively proceed from that state according to the transition function of $\mathcal{A}_\psi$. Let us finally discuss the split operator and team semantics atoms. For the former, we use indexed subformulae to track which traces we analyse with the regard to the respective subformula. In the initial states, we work with the full index set $[n]$ for the formula φ and whenever we encounter a split $\psi_1 \vee \psi_2$, we non-deterministically partition the given index set into two sets and check each ψ_i only with regard to its respective index set. Using this information, we can directly infer whether a dependence atom $\mathrm{dep}(\psi_1, \ldots, \psi_n, \varphi)$ holds in that state by checking whether all traces agreeing on the formulae $\psi_1, \ldots, \psi_n$ also agree on φ. Likewise, the atoms NE, $\overset{1}{\mathsf{A}}$ and $\subseteq$ are handled by explicitly checking the semantics for each state. Notice again that we only allow propositional formulae in dependence atoms, so the semantics of these atoms depend only on the current state.

For the full logic and the $\models_\forall$ mode of satisfaction, we handle universal quantifiers by complementing automata for the existential quantifiers and introducing Boolean negation.

Formally, for a natural number n (this will correspond to the number of traces later), we set $[n] := \{1, \ldots, n\}$ and $I := 2^{[n]}$. For an index set $M \in I$, let $\mathrm{SP}(M) = \{ (M_1, M_2) \mid M = M_1 \uplus M_2 \}$ be the set of possible splits. Notice that M_1 or M_2 can be the empty set here. We denote by $\mathrm{Sub}(\varphi)$ the set of subformulae, closed under negation, of an $\exists$TeamCTL* formula φ that is defined in the obvious way.

Definition 6.2. Let $n \in \mathbb{N}$. For an $\exists$TeamCTL$^*(\mathcal{S})$ formula φ, the *indexed Fischer-Ladner-closure* over n traces $\mathrm{cl}(\varphi) \subseteq 2^{(\mathrm{Sub}(\varphi)) \times I}$ is given by the set of all pairs (ψ, M) of subformulae ψ of φ and index sets $M \in I$ with the following properties:

- If $\psi \in \mathrm{Sub}(\varphi)$ and $M \in I$, then $(\psi, M) \in \mathrm{cl}(\varphi)$.
- If $(p, M) \in \mathrm{cl}(\varphi)$ for $p \in \mathrm{AP}$, then $(\neg p, M) \in \mathrm{cl}(\varphi)$.
- If $(\mathrm{dep}(\varphi_1, \ldots, \varphi_k, \psi), M) \in \mathrm{cl}(\varphi)$, then $(\neg\varphi_i, M) \in \mathrm{cl}(\varphi)$ for $1 \le i \le k$ and $(\neg\psi, M) \in \mathrm{cl}(\varphi)$.
- If $(\varphi_1 \cdots \varphi_k \subseteq \psi_1 \cdots \psi_k, M) \in \mathrm{cl}(\varphi)$, then $(\neg\varphi_i, M)$, $(\neg\psi_i, M) \in \mathrm{cl}(\varphi)$ for $1 \le i \le k$.

Definition 6.3. Let $n \in \mathbb{N}$ and φ be an $\exists$TeamCTL$^*(\mathcal{S})$ formula, and $\mathrm{cl}(\varphi)$ be the indexed Fischer-Ladner-closure over n traces. A set $S \in 2^{\mathrm{cl}(\varphi)}$ is *consistent* iff the following holds:

(1) Let ψ be a propositional formula such that $\psi, \neg\psi$ occur in $\mathrm{cl}(\varphi)$. Then we have that
   - for all $i \in [n]$: $(\psi, \{i\}) \in S$ iff $(\neg\psi, \{i\}) \notin S$.
   - for all $M \in I$: $(\psi, M) \in S$ iff $\forall j \in M : (\psi, \{j\}) \in S$.
   - for all $M \in I$: $(\neg\psi, M) \in S$ iff $\forall j \in M : (\neg\psi, \{j\}) \in S$.

(2) If $(\psi_1 \wedge \psi_2, M) \in \mathrm{cl}(\varphi)$, then $(\psi_1 \wedge \psi_2, M) \in S$ iff $(\psi_1, M) \in S$ and $(\psi_2, M) \in S$.

(3) If $(\psi_1 \vee \psi_2, M) \in \mathrm{cl}(\varphi)$, then $(\psi_1 \vee \psi_2, M) \in S$ iff $(\psi_1, M_1) \in S$ and $(\psi_2, M_2) \in S$ for some $(M_1, M_2) \in \mathrm{SP}(M)$.

(4) If $(\psi_1 \circledvee \psi_2, M) \in \mathrm{cl}(\varphi))$, then $(\psi_1 \circledvee \psi_2, M) \in S$ iff $(\psi_1, M) \in S$ or $(\psi_2, M) \in S$.

(5) If $(\psi_1 \mathrm{op} \psi_2, M) \in \mathrm{cl}(\varphi)$, then $(\psi_1 \mathrm{op} \psi_2, M) \in S$ implies $(\psi_1, M) \in S$ or $(\psi_2, M) \in S$ for $\mathrm{op} \in \{\mathsf{U}, \mathsf{W}\}$.

(6) If $(\mathrm{dep}(\varphi_1, \ldots, \varphi_k, \psi), M) \in \mathrm{cl}(\varphi)$, then $(\mathrm{dep}(\varphi_1, \ldots, \varphi_k, \psi), M) \in S$ iff $\forall i, j \in M : (\forall 1 \le \ell \le k : (\varphi_\ell, \{i\}) \in S \iff (\varphi_\ell, \{j\}) \in S)$ implies $((\psi, \{i\}) \in S \iff (\psi, \{j\}) \in S)$.

(7) If $(\mathrm{NE}, M) \in \mathrm{cl}(\varphi)$, then $(\mathrm{NE}, M) \in S$ iff $M \neq \emptyset$.

(8) If $(\overset{1}{\mathsf{A}}\,\psi, M) \in \mathrm{cl}(\varphi)$, then $(\overset{1}{\mathsf{A}}\,\psi, M) \in S$ iff $\forall i \in M : (\psi, \{i\}) \in S$.

(9) If $(\varphi_1 \cdots \varphi_k \subseteq \psi_1 \cdots \psi_k, M) \in \mathrm{cl}(\varphi)$, then $(\varphi_1 \cdots \varphi_k \subseteq \psi_1 \cdots \psi_k, M) \in S$ iff for all $i \in M$ there exists $j \in M$: If $(\varphi_\ell, \{i\}) \in S$ then $(\psi_\ell, \{j\}) \in S$, and if $(\neg\varphi_\ell, \{i\}) \in S$ then $(\neg\psi_\ell, \{j\}) \in S$, for $1 \le \ell \le k$.

We denote the set of all consistent sets by $\mathrm{Con}(\varphi)$.

Definition 6.4. Let $\Sigma := 2^{\mathrm{AP}}$. The *local transition relation* $\rightarrow$ is a maximal subset of $\mathrm{Con}(\varphi) \times [n] \times \Sigma \times \mathrm{Con}(\varphi)$ such that for $S \xrightarrow{i,\sigma} S'$ (meaning $(S, i, \sigma, S') \in \rightarrow$) with $S, S' \in \mathrm{Con}(\varphi)$, $\sigma \in \Sigma$ and $i \in [n]$, the following holds:

- For $j \neq i$ with $(\psi, \{j\}) \in S$: $(\psi, \{j\}) \in S'$, where $\psi \in \{ p, \neg p \mid p \in \mathrm{AP} \}$.
- For all $p \in \mathrm{AP} : (p, \{i\}) \in S' \iff p \in \sigma$.
- If $(\mathsf{X}\psi, M) \in S$, then $(\psi, M) \in S'$.
- If $(\psi_1 \mathrm{op} \psi_2, M) \in S$ and $(\psi_2, M) \notin S$, then $(\psi_1 \mathrm{op} \psi_2, M) \in S'$ for $\mathrm{op} \in \{\mathsf{U}, \mathsf{W}\}$.

This relation roughly denotes whether a state would be a suitable successor of another state according to the transition relation in the standard Fischer-Ladner construction when an input symbol is read on the i-th component. Notice that this relation and the transition function we are about to define progress on a single trace in one step, while our tefs can progress on multiple steps at once. This is without loss of generality as simultaneous progress on multiple traces can easily be simulated using consecutive transitions over intermediate states.

Using the definitions just described, we can construct a suitable AABA for a $\mathrm{TeamCTL}^*(\mathcal{S})$ formula φ. The details of the translation including Boolean negation can be found in the extended arXiv version of the paper [18]. We then obtain the following Theorem.

Theorem 6.5. *Let* $\mathcal{S} = \{\varovee, \mathrm{NE}, \overset{1}{\mathrm{A}}, \mathrm{dep}, \subseteq\}$.

(1) For every $(\mathrm{TeamCTL}^*(\mathcal{S}), \models_*)$ *formula* φ *and natural number* n*, there is a GAABA* $\mathcal{A}_\varphi$ *such that* $\mathcal{L}_{TE}(\mathcal{A}_\varphi)$ *is equivalent*[2] *to* $\mathcal{L}^n_{TE}(\varphi)$ *for all sets of tefs TE.*

(2) For every $(\mathrm{TeamCTL}^*(\mathcal{S}), \models_*)$ *formula* φ *and natural number* n*, there is an AABA* $\mathcal{A}_\varphi$ *such that* $\mathcal{L}_{TE}(\mathcal{A}_\varphi)$ *is equivalent*[2] *to* $\mathcal{L}^n_{TE}(\varphi)$ *for all sets of tefs TE.*

(3) For all sets of tefs TE, if it is decidable whether $\mathcal{L}_{TE}(\mathcal{A})$ *is empty for every AABA* $\mathcal{A}$*, then the finite TE satisfiability and model checking problems and the TE path checking problem for* $\mathrm{TeamCTL}^*(\mathcal{S})$ *are decidable.*

We note that the undecidability proof for $\mathrm{TeamCTL}(\varovee)$ model checking of Theorem 5.2 relies on the use of infinite teams. This directs us to consider teams of fixed size in order to recover decidability. Finally, applying Theorem 6.1 to Theorem 6.5, and utilising Theorem 3.3, we obtain the following. Here ALL denotes the set of all generalised atoms.

Corollary 6.6. *Let TE be the set of* k*-synchronous or* k*-context-bounded tefs and* $\mathcal{S} = \{\varovee, \mathrm{NE}, \overset{1}{\mathrm{A}}, \mathrm{dep}, \subseteq\}$.

(1) The problems fs-SAT(*TE*), fs-MC(*TE*)*, and* PC(*TE*) *for the logics* $\mathrm{TeamCTL}^*(\mathcal{S}, \mathrm{ALL})$ *are decidable.*

(2) For a fixed formula φ *and fixed* $n, k \in \mathbb{N}$*, the fixed size* k*-synchronous or* k*-context-bounded model checking over* n *traces for* $\mathrm{TeamCTL}^*(\mathcal{S})$ *is decidable in polynomial time.*

The last item follows from the fact that for fixed parameters, the automata $\mathcal{A}_\varphi$ are of constant size. Translations described in [20, Theorem 3.11, Corollary 3.16] can be used to yield equivalent Büchi automata of constant size. The emptiness test on the product with the automaton accepting the input traces is then possible in polynomial time.

7 CONCLUSION

In this paper, we revisited temporal team semantics and introduced quantification over tefs in order to obtain fine-grained control of asynchronous progress over different traces. We discussed required properties for tefs in depth and showed that, unlike previous asynchronous hyperlogics, variants of our logic are able to express some of these properties (like synchronicity and fairness) themselves. Table 2 summarises the complexity results for the model checking problem. We showed that the model checking problem is highly undecidable already for $\exists$TeamCTL with Boolean disjunctions. However, we also showed that $\mathrm{TeamCTL}^*(\varovee, \mathrm{NE}, \overset{1}{\mathrm{A}}, \mathrm{dep}, \subseteq)$ can be translated to AABA over teams of fixed size, thus yielding a general approach to define restricted asynchronous semantics and obtain decidability for the path checking problem as well as the finite model checking and satisfiability problem.

[2]Here, we treat sets of tuples as multisets in the obvious way.

Model Checking Problem for	Complexity & Reference
$\exists\mathrm{TeamLTL}(\varovee, \subseteq)$	Σ^0_1-hard (Cor. 4.3 & [33, Thm. 2])
$\forall\mathrm{TeamLTL}(\varovee, \subseteq, \mathrm{NE})$	Σ^0_1-hard (Cor. 4.3 & [33, Thm. 2])
$\exists\mathrm{TeamCTL}(\varovee, \subseteq)$	Σ^0_1-hard (Cor. 4.3 & [33, Thm. 2])
$\forall\mathrm{TeamCTL}(\varovee, \subseteq)$	Σ^0_1-hard (Cor. 4.3 & [33, Thm. 2])
$\exists\mathrm{TeamCTL}(\varovee)$	Σ^1_1-hard (Thm. 5.2)
$\mathrm{TeamCTL}^*(\mathcal{S}, \mathrm{ALL})$ for k-synchronous or k-context-bounded tefs	decidable (Thm. 3.3, Cor. 6.6)
$\mathrm{TeamCTL}^*(\mathcal{S})$ for k-synchronous or k-context-bounded tefs, where k and the number of traces is fixed	polynomial time (Cor. 6.6)

Table 2: Complexity results overview. The Σ^0_1-hardness results follow via embeddings of synchronous TeamLTL**, whereas the Σ^1_1-hardness truly relies on asynchronity.** ALL **is the set of all generalised atoms and** $\mathcal{S} = \{\varovee, \mathrm{NE}, \overset{1}{\mathrm{A}}, \mathrm{dep}, \subseteq\}$**.**

A possible direction for future work would be to study further restricted classes of tefs beyond k-synchronicity and k-context-boundedness for which the corresponding emptiness problem for AABA is decidable, since this would not only lead to new decidable semantics for our logic, but also for other logics such as H_μ [20].

ACKNOWLEDGMENTS

The first two authors were partially funded by the DFG grant MU 1508/3-1. The third author was partially funded by the DFG grant ME 4279/1-2. The fourth author was partially funded by the DFG grant VI 1045/1-1. We thank the anonymous reviewers for their helpful comments.

REFERENCES

[1] Samson Abramsky, Juha Kontinen, Jouko Väänänen, and Heribert Vollmer (Eds.). 2016. *Dependence Logic, Theory and Applications.* Springer. https://doi.org/10.1007/978-3-319-31803-5

[2] Rajeev Alur and Thomas A. Henzinger. 1994. A Really Temporal Logic. *J. ACM* 41, 1 (1994), 181–204. https://doi.org/10.1145/174644.174651

[3] Jan Baumeister, Norine Coenen, Borzoo Bonakdarpour, Bernd Finkbeiner, and César Sánchez. 2021. A Temporal Logic for Asynchronous Hyperproperties. In *CAV (1) (Lecture Notes in Computer Science, Vol. 12759).* Springer, 694–717.

[4] Raven Beutner and Bernd Finkbeiner. 2021. A Temporal Logic for Strategic Hyperproperties. In *32nd International Conference on Concurrency Theory, CONCUR 2021, August 24-27, 2021, Virtual Conference (LIPIcs, Vol. 203)*, Serge Haddad and Daniele Varacca (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 24:1–24:19. https://doi.org/10.4230/LIPIcs.CONCUR.2021.24

[5] Borzoo Bonakdarpour, Pavithra Prabhakar, and César Sánchez. 2020. Model Checking Timed Hyperproperties in Discrete-Time Systems. In *NASA Formal Methods - 12th International Symposium, NFM 2020, Moffett Field, CA, USA, May 11-15, 2020, Proceedings (Lecture Notes in Computer Science, Vol. 12229)*, Ritchie Lee, Susmit Jha, and Anastasia Mavridou (Eds.). Springer, 311–328. https://doi.org/10.1007/978-3-030-55754-6_18

[6] Laura Bozzelli, Bastien Maubert, and Sophie Pinchinat. 2015. Unifying Hyper and Epistemic Temporal Logics. In *Foundations of Software Science and Computation Structures - 18th International Conference, FoSSaCS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings (Lecture Notes in Computer Science, Vol. 9034)*, Andrew M. Pitts (Ed.). Springer, 167–182. https://doi.org/10.1007/978-3-662-46678-0_11

[7] Laura Bozzelli, Adriano Peron, and César Sánchez. 2021. Asynchronous Extensions of HyperLTL. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021.* IEEE, 1–13. https://doi.org/10.1109/LICS52264.2021.9470583

[8] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. 2001. *Model checking.* MIT Press.

[9] Michael R. Clarkson, Bernd Finkbeiner, Masoud Koleini, Kristopher K. Micinski, Markus N. Rabe, and César Sánchez. 2014. Temporal Logics for Hyperproperties. In *Principles of Security and Trust - Third International Conference, POST 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014, Proceedings (Lecture Notes in Computer Science, Vol. 8414)*, Martín Abadi and Steve Kremer (Eds.). Springer, 265–284. https://doi.org/10.1007/978-3-642-54792-8_15

[10] Michael R. Clarkson and Fred B. Schneider. 2010. Hyperproperties. *Journal of Computer Security* 18, 6 (2010), 1157–1210. https://doi.org/10.3233/JCS-2009-0393

[11] Norine Coenen, Bernd Finkbeiner, Christopher Hahn, and Jana Hofmann. 2019. The Hierarchy of Hyperlogics. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019.* IEEE, 1–13. https://doi.org/10.1109/LICS.2019.8785713

[12] Stéphane Demri, Valentin Goranko, and Martin Lange. 2016. *Temporal Logics in Computer Science: Finite-State Systems.* Cambridge University Press. https://doi.org/10.1017/CBO9781139236119

[13] Arnaud Durand, Miika Hannula, Juha Kontinen, Arne Meier, and Jonni Virtema. 2018. Approximation and dependence via multiteam semantics. *Ann. Math. Artif. Intell.* 83, 3-4 (2018), 297–320. https://doi.org/10.1007/s10472-017-9568-4

[14] Bernd Finkbeiner, Markus N. Rabe, and César Sánchez. 2015. Algorithms for Model Checking HyperLTL and HyperCTL*. In *Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 9206)*, Daniel Kroening and Corina S. Pasareanu (Eds.). Springer, 30–48. https://doi.org/10.1007/978-3-319-21690-4_3

[15] Bernd Finkbeiner and Martin Zimmermann. 2017. The First-Order Logic of Hyperproperties. In *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany (LIPIcs, Vol. 66)*, Heribert Vollmer and Brigitte Vallée (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 30:1–30:14. https://doi.org/10.4230/LIPIcs.STACS.2017.30

[16] Erich Grädel and Richard Wilke. 2022. Logics with Multiteam Semantics. *ACM Trans. Comput. Log.* 23, 2 (2022), 13:1–13:30. https://doi.org/10.1145/3487579

[17] Erich Grädel, Phokion G. Kolaitis, Juha Kontinen, and Heribert Vollmer. 2019. Logics for Dependence and Independence (Dagstuhl Seminar 19031). *Dagstuhl Reports* 9, 1 (2019), 28–46. https://doi.org/10.4230/DagRep.9.1.28

[18] Jens Oliver Gutsfeld, Arne Meier, Christoph Ohrem, and Jonni Virtema. 2021. Temporal Team Semantics Revisited. *CoRR* abs/2110.12699 (2021). arXiv:2110.12699 https://arxiv.org/abs/2110.12699

[19] Jens Oliver Gutsfeld, Markus Müller-Olm, and Christoph Ohrem. 2020. Propositional Dynamic Logic for Hyperproperties. In *31st International Conference on Concurrency Theory, CONCUR 2020, September 1-4, 2020, Vienna, Austria (Virtual Conference) (LIPIcs, Vol. 171)*, Igor Konnov and Laura Kovács (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 50:1–50:22. https://doi.org/10.4230/LIPIcs.CONCUR.2020.50

[20] Jens Oliver Gutsfeld, Markus Müller-Olm, and Christoph Ohrem. 2021. Automata and fixpoints for asynchronous hyperproperties. *Proc. ACM Program. Lang.* 5, POPL (2021), 1–29. https://doi.org/10.1145/3434319

[21] Miika Hannula, Juha Kontinen, Jan Van den Bussche, and Jonni Virtema. 2020. Descriptive complexity of real computation and probabilistic independence logic. In *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*, Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller (Eds.). ACM, 550–563. https://doi.org/10.1145/3373718.3394773

[22] Miika Hannula and Jonni Virtema. 2021. Tractability Frontiers in Probabilistic Team Semantics and Existential Second-Order Logic over the Reals. In *Logics in Artificial Intelligence - 17th European Conference, JELIA 2021, Virtual Event, May 17-20, 2021, Proceedings (Lecture Notes in Computer Science, Vol. 12678)*, Wolfgang Faber, Gerhard Friedrich, Martin Gebser, and Michael Morak (Eds.). Springer, 262–278. https://doi.org/10.1007/978-3-030-75775-5_18

[23] Wilfrid Hodges. 1997. Some strange quantifiers. In *Structures in Logic and Computer Science*, J. Mycielski, G. Rozenberg, and A. Salomaa (Eds.). Lecture Notes in Computer Science, Vol. 1261. Springer Berlin / Heidelberg, 51–65.

[24] Juha Kontinen and Max Sandström. 2021. On the Expressive Power of TeamLTL and First-Order Team Logic over Hyperproperties. In *Logic, Language, Information, and Computation - 27th International Workshop, WoLLIC 2021, Virtual Event, October 5-8, 2021, Proceedings (Lecture Notes in Computer Science, Vol. 13038)*, Alexandra Silva, Renata Wassermann, and Ruy J. G. B. de Queiroz (Eds.). Springer, 302–318. https://doi.org/10.1007/978-3-030-88853-4_19

[25] Andreas Krebs, Arne Meier, and Jonni Virtema. 2015. A Team Based Variant of CTL. In *TIME*. IEEE Computer Society, 140–149.

[26] Andreas Krebs, Arne Meier, Jonni Virtema, and Martin Zimmermann. 2018. Team Semantics for the Specification and Verification of Hyperproperties. In *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 117)*, Igor Potapov, Paul Spirakis, and James Worrell (Eds.). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 10:1–10:16. https://doi.org/10.4230/LIPIcs.MFCS.2018.10

[27] Martin Lück. 2020. On the complexity of linear temporal logic with team semantics. *Theor. Comput. Sci.* 837 (2020), 1–25.

[28] Christos H. Papadimitriou. 2007. *Computational complexity.* Academic Internet Publ.

[29] Nicholas Pippenger. 1997. *Theories of computability.* Cambridge University Press.

[30] Nir Piterman and Amir Pnueli. 2018. Temporal Logic and Fair Discrete Systems. In *Handbook of Model Checking*, Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem (Eds.). Springer, 27–73. https://doi.org/10.1007/978-3-319-10575-8_2

[31] Markus N. Rabe. 2016. *A temporal logic approach to Information-flow control.* Ph. D. Dissertation. Saarland University. http://scidok.sulb.uni-saarland.de/volltexte/2016/6387/

[32] Jouko A. Väänänen. 2007. *Dependence Logic - A New Approach to Independence Friendly Logic.* London Mathematical Society student texts, Vol. 70. Cambridge University Press. http://www.cambridge.org/de/knowledge/isbn/item1164246/?site_locale=de_DE

[33] Jonni Virtema, Jana Hofmann, Bernd Finkbeiner, Juha Kontinen, and Fan Yang. 2021. Linear-Time Temporal Logic with Team Semantics: Expressivity and Complexity. In *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2021, December 15-17, 2021, Virtual Conference (LIPIcs, Vol. 213)*, Mikolaj Bojanczyk and Chandra Chekuri (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 52:1–52:17. https://doi.org/10.4230/LIPIcs.FSTTCS.2021.52