Parallel Computational Tree Logic

Andreas Krebs
Wilhelm-Schickard-Institut für Informatik
Eberhard-Karls-Universität Tübingen
Sand 13, 72076 Tübingen, Germany
Email: mail@krebs-net.de

Arne Meier and Jonni Virtema
Institut für Theoretische Informatik
Leibniz Universität Hannover
Appelstr. 4, 30176 Hannover, Germany
Email: {meier,virtema}@thi.uni-hannover.de

Abstract—In the present paper we introduce a parallel version of the Computation Tree Logic. Here we distinguish between asynchronous and synchronous semantics. For both cases we investigate the computational complexity of the satisfiability as well as the model checking problem. Satisfiability is shown to be EXPTIME-complete whereas it does not matter which of the two semantics are considered. For model checking we prove a PSPACE-completeness result for the synchronous case, and show P-completeness for the asynchronous case. Further we exhibit several interesting properties of both semantics.

I. INTRODUCTION

Computational tree logic $\mathcal{CTL}$, introduced in the late 1950s by Prior [20], is a well-known and important logic in the area of computer science that has influenced the area of program verification significantly. Since the introduction of $\mathcal{CTL}$ a wide research field has emerged. Here the most seminal contributions have been made by Kripke [13], Pnueli [18], Emerson, Clarke, and Halpern [5, 3] to name a few.

In real life applications, especially in the field of program verification, computational complexity is of the greatest significance. In the framework of logic, the most significant related decision problems are the satisfiability problem and the model checking problem. From a software engineering view the satisfiability problem can be seen as the question of specification consistency: The specification of a program is expressed via a formula of some logic (e.g., $\mathcal{CTL}$). One then asks whether there exists a model that satisfies the given formula. For model checking an implementation of a system is depicted via a Kripke structure and a specification via a formula. One then wants to know whether the structure satisfies the formula (i.e., whether the system satisfies the specification). The satisfiability problem for $\mathcal{CTL}$ is known to be EXPTIME-complete by Fischer and Ladner, and Pratt [6, 19] whereas the model checking problem has been shown to be P-complete by Clarke et al., and Schnoebelen [2, 21].

The semantics for $\mathcal{CTL}$ is defined via pointed Kripke models, and is well suited for modelling single thread computation. However nowadays tasks are distributed and computed by different computers or threads in parallel. In this article we propose two variants of $\mathcal{CTL}$ that are designed to model parallel computation. We abandon the idea of defining semantics for $\mathcal{CTL}$ via pointed Kripke structures. Instead the semantics are defined via pairs $(K, T)$, where $K$ is an ordinary Kripke structure and $T$, called a team of $K$, is a subset of the domain of $K$. Intuitively this means that one considers multiple configurations at once. The state of our system is not described by a single node in the Kripke system but by a set of nodes, i.e., by a team. For team computation it is not important which thread accomplishes a certain job as the tasks cannot be distinguished in general. In brief, we give team semantics for $\mathcal{CTL}$.

Team semantics was introduced first to the framework of first-order logic by Hodges [9] and later in the framework of modal logic by Väänänen [24]. The fundamental idea behind team semantics is crisp. The idea is to shift from singletons to sets as satisfying elements of formulas. These sets of satisfying elements are called teams. In the team semantics of first-order logic formulas are evaluated with respect to first-order structures and sets of assignments. In the team semantics of modal logic formulas are evaluated with respect to Kripke structures and sets of worlds.

In recent years the research around first-order and modal team semantics has been vibrant. See, e.g., [4, 8, 12] for related research in the modal context. While team semantics has been considered in the context of regular modal logic, to the best knowledge of the authors, this is the first article to consider team like semantics for a more serious temporal logic.

In this article we consider two different models of parallel computation: synchronous model and an asynchronous one. In the synchronous model we stipulate that the evolution of time is synchronous among all threads, whereas in the asynchronous model we do not have this assumption. That is, the main difference between the two models is the existence of a global clock which keeps the threads synchronised (or not in the case of the absence of this clock). In the logic side the main difference of these two approaches can be seen in the definitions of the semantics for the modal operator Until (see Definition 3): Either the time is synchronous among all threads, and hence when we quantify over a time point in the future all team members will advance the same number of steps in the Kripke structure, or we consider an asynchronous model, where when we quantify over a future point each team member might advance a different number of steps.

For both models there are important examples with practical applications. Consider we have a synchronous timed system and we want to verify that it is possible to satisfy a property $p$ simultaneously in each thread. We need to verify that the formula $EF(p)$ holds. On the other hand many systems are
not timed and one cannot expect that different threads are executed at the same rate. Here one might want to verify that all tasks reach some terminating state after a finite amount of time, but we do not care if they reach it at the same speed. Hence in the asynchronous model a formula EFp should allow the individual team members to move a different number of steps.

**Related work.** Several models of parallel computation have been considered. Some of these approaches deal directly with computational devices as in circuit complexity. Here the complexity class NC consists of all problems for which there exist algorithms running in polylogarithmic time on a parallel computer with a polynomial number of processors. Formally, NC contains all problems solvable by polynomial-size polylogarithmic-depth logtime-uniform families of circuits with bounded fan-in AND, OR, NOT gates [26]. Another approach of this kind is the introduction of a parallel random access machine (PRAM) which has been done by Immerman [10]. Here the set of parallel computing processors are synchronised via a global clock. Each of them can read/write to a global memory depending on the policies of the machine. The connection between logic and distributed computing has been considered recently by Hella et al. [7]. They give a characterisation of constant time parallel computation in the spirit of descriptive complexity [11].

Moreover a classification of the computational complexity of fragments of the satisfiability as well as the model checking problem of CTL by means of allowed Boolean operators and/or combinations of allowed temporal operators has been obtained recently [15, 1].

A survey on Kripke semantics with connections to several areas of logic, e.g., temporal, dependence, and hybrid logic can be found in a work of Meier et al. [16].

**Results.** We introduce two new variants of CTL to model parallel computation: an asynchronous one and a synchronous one. We investigate the computational complexity of the satisfiability and the model checking problem of these variants. For model checking the complexity differs with respect to these variants. In the asynchronous case we show that the complexity is P-complete and hence the same as for CTL by exploiting structural properties of the satisfaction relation. For synchronised semantics surprisingly the complexity becomes PSPACE-complete. Hence having synchronised processors makes the model checking in this logic intractable under reasonable complexity separation assumptions. For the satisfiability problem we show that the complexity stays EXPTIME-complete similar as for CTL independently on which semantics is used.

**Structure of the paper.** In Section II we give syntax and semantics of two novel variants of computation tree logic CTL. In Section III we prove closure properties of the satisfaction relations of the two variants. Section IV deals with their expressive power. In Section V we completely classify the computational complexity of the satisfiability and the model checking problem with respect to both variants. Finally we present interesting further research directions and conclude.

### II. Preliminaries

We start this section with a brief summary of the relevant complexity classes for this paper. We then define the syntax and semantics of computational tree logic CTL. We deviate from the existing literature by using a convention that is customary related to logics with team semantics: We define the syntax of CTL in negation normal form, i.e., we require that negations may appear only in front of proposition symbols. We then introduce two variants of CTL that are designed to model parallel computation.

#### A. Complexity

The underlying computation model is Turing machines. We will make use of the complexity classes P, PSPACE, and EXPTIME. All reductions in this paper are logspace many-to-one reductions, i.e., computable by a deterministic Turing machine running in logarithmic space. For a deeper introduction into this topic we refer the reader to the good book of Pippenger [17].

#### B. Temporal Logic

Let PROP be a finite set of proposition symbols. The set of all CTL-formulas is defined inductively via the following grammar:

$$\varphi ::= p \mid \neg p \mid (\varphi \land \varphi) \mid (\varphi \lor \varphi) \mid PX\varphi \mid P[\varphi U \varphi] \mid P[\varphi W \varphi],$$

where $P \in \{A, E\}$ and $p \in$ PROP. We define the following usual shorthands: $\top ::= p \lor \neg p$, $\bot ::= p \land \neg p$, $F\varphi ::= [\top U \varphi]$, and $G\varphi ::= [\varphi U \bot]$. A Kripke structure $K$ is a tuple $(W, R, \eta)$ where $W$ is a finite, non-empty set of states, $R: W \times W$ is a total transition relation (i.e., for every $w \in W$ there is a $w' \in W$ such that $wRw'$), and $\eta: W \to 2^{PROP}$ is a labelling function. A path $\pi = \pi(1), \pi(2), \ldots$ is an infinite sequence of states $\pi(i) \in W$ such that $\pi(i)R\pi(i+1)$ holds. By $\Pi(w)$ we denote the (infinite) set of all paths $\pi$ for which $\pi(1) = w$.

**Definition 1** (Semantics of CTL). Let $K = (W, R, \eta)$ be a Kripke structure and $w \in W$ a state. The satisfaction relation $\models$ for CTL is defined as follows:

- $K, w \models p$ if $p \in \eta(w)$,
- $K, w \models \neg p$ if $p \notin \eta(w)$,
- $K, w \models \varphi \land \psi$ if $K, w \models \varphi$ and $K, w \models \psi$,
- $K, w \models \varphi \lor \psi$ if $K, w \models \varphi$ or $K, w \models \psi$,
- $K, w \models PX\varphi$ if $\exists \pi \in \Pi(w): K, \pi(2) \models \psi$,
- $K, w \models P[\varphi U \psi]$ if $\exists \pi \in \Pi(w): \forall k \in N : K, \pi(k) \models \psi$ and $\forall 1 \leq i < k : K, \pi(i) \models \varphi$, and
- $K, w \models P[\varphi W \psi]$ if $\exists \pi \in \Pi(w): \forall i : K, \pi(i) \models \varphi$ or $\exists k \in N : K, \pi(k) \models \psi$ and $\forall 1 \leq i < k : K, \pi(i) \models \varphi$.

The traditional semantics for CTL is defined via pointed Kripke models and is well suited for modelling single thread computation. Next we will introduce two alternative semantics for CTL that are designed for modelling parallel computation.
Semantics for these logics are defined via what is known as teams.

A multiset is a generalisation of the concept of a set that allows multiple instances of the same element in the multiset. We denote a multiset that has elements \( p, q, r, r \) by \( \{p, q, r, r\} \). When \( W \) is a set (or a multiset), we use \( T \subseteq W \) to denote that \( T \) is a multiset such that each element of \( T \) is also an element of \( W \).

**Definition 2 (Team).** Let \( K = (W, R, \eta) \) be a Kripke structure. Any multiset \( T \) such that \( T \subseteq W \) is called a team of \( K \).

Intuitively a team can be seen as a set of threads or processors which are processing in parallel. We now introduce two alternative approaches for defining the semantics of this approach. We first define the synchronous team-semantics for \( C^\infty \). In this approach we assume that the computation of the threads that compute in parallel has to be synchronous. We will then define the asynchronous team-semantics for \( C^\infty \). In this approach the computation of each thread of the threads that compute in parallel compute completely independently. In the side of semantics, the difference can be seen in the clauses for until and weak until. The difference of the two semantics is also depicted in Figure 1.

**Definition 3 (Synchronous and asynchronous team semantics).** Let \( K = (W, R, \eta) \) be a Kripke structure, \( T = \{t_1, \ldots, t_n\} \) be a team of \( K \), and \( \varphi \) and \( \psi \) be \( C^\infty \)-formulas. The synchronous satisfaction relation \( \models^s \) and the asynchronous satisfaction relation \( \models^a \) for \( C^\infty \) are defined as follows.

The following clauses are common to both semantics. In the clauses \( \models \) denotes either \( \models^s \) or \( \models^a \).

\[
\begin{align*}
K, T & \models^a \varphi \quad \text{iff} \quad \forall \pi \in \Pi(t) : \varphi(\pi) \\
K, T & \models^a \neg \varphi \quad \text{iff} \quad \forall \pi \in \Pi(t) : \neg \varphi(\pi) \\
K, T & \models^a (\varphi \land \psi) \quad \text{iff} \quad K, T \models^a \varphi, K, T \models^a \psi \\
K, T & \models^a (\varphi \lor \psi) \quad \text{iff} \quad \exists \pi_1 \in \Pi(t) : (\varphi(\pi_1) \lor \psi) \\
K, T & \models^a \Box \varphi \quad \text{iff} \quad \forall \pi_1 \in \Pi(t) : \forall \pi \in \Pi(t) : \varphi(\pi_1) \\
K, T & \models^a \Diamond \varphi \quad \text{iff} \quad \exists \pi_1 \in \Pi(t) : \exists \pi \in \Pi(t) : \varphi(\pi_1)
\end{align*}
\]

For the synchronous semantics we have the following clauses, where \( P \in \{A, E\}, \) and \( \varnothing = \forall \) if \( P = A \) and \( \varnothing = \exists \) if \( P = E \).

\[
\begin{align*}
K, T & \models^a P[\varphi U \psi] \quad \text{iff} \quad \\
\exists k \in N : K, \bigcup_{1 \leq j \leq n} \{ \pi_j(k) \} & \models^s \psi \\
\forall 1 \leq i < k : K, \bigcup_{1 \leq j \leq n} \{ \pi_j(i) \} & \models^s \varphi.
\end{align*}
\]

For the asynchronous semantics we have the following clauses, where \( P \in \{A, E\}, \) and \( \varnothing = \forall \) if \( P = A \) and \( \varnothing = \exists \) if \( P = E \).

\[
\begin{align*}
K, T & \models^a P[\varphi W \psi] \quad \text{iff} \quad \\
\exists k \in N : K, \bigcup_{1 \leq j \leq n} \{ \pi_j(k) \} & \models^a \psi \\
\forall 1 \leq i < k : K, \bigcup_{1 \leq j \leq n} \{ \pi_j(i) \} & \models^a \varphi.
\end{align*}
\]

III. PROPERTIES OF ASYNCHRONOUS AND SYNCHRONOUS SEMANTICS

In the following section we investigate several properties of the asynchronous and synchronous satisfaction relations \( \models^a, \models^s \). In particular, we will use them in the end to deduce a corollary for asynchronous semantics which shows the interplay with the usual \( C^\infty \) satisfaction relation.

Observe that \( K, T \models \perp \) holds if and only if \( T = \emptyset \). The proof of the following lemma then is very easy.

**Lemma 4 (Empty team property).** The following holds for every Kripke model \( K \) and \( \models \) is in \( \{ \models^a, \models^s \} \):

\[
K, \emptyset \models \varphi \quad \text{holds for every } C^\infty \text{-formula } \varphi.
\]

When restricted to singleton teams, the synchronous and asynchronous team-semantics coincide with the traditional semantics of \( C^\infty \) defined via pointed Kripke models.
Lemma 5 (Singleton equivalence). For every Kripke structure $K = (W, R, \eta)$ and every world $w \in W$ the equivalence holds:

$$K, \{w\} \models \varphi \iff K, w \models \varphi \iff K, w \models \varphi.$$  

Proof. It is straightforward to check that on singleton teams the synchronous semantics of until and weak until coincide with that of the asynchronized semantics. Since none of the clauses in the two semantics makes the size of teams grow, the equivalence (1) follows.

Now turn to (2). Let $K = (W, R, \eta)$ be an arbitrary Kripke structure. We first prove the claim via induction on $|\varphi|$:

Assume that $\varphi$ is a (negated) proposition symbol $p$. Now $K, w \models \varphi$ if $p$ is (not) in $\eta(w)$.

The case $\land$ trivial. For the $\lor$ case, assume that $\varphi = \psi \land \theta$. Now it holds that $K, w \models \psi \land \theta$ if $K, w \models \psi$ or $K, w \models \theta$.

Here the first equivalence holds by the semantics of disjunction, the second equivalence follow by the induction hypothesis, the third via the empty set property, the fourth via the empty set property in combination with the semantics of “or”, and the last by the team semantics of disjunction.

The cases for EX and AX, until and weak until are all similar and straightforward. We show here the case for EX. Assume $\varphi = \mathsf{EX}\psi$. Now $K, w \models \mathsf{EX}\psi$ iff there exists a point $\pi \in \Pi(w)$ such that $K, \pi(2) \models \psi$. Now since trivially $\bigwedge_{1 \leq j \leq 1} \{\pi_{i_1}(2)\} = \{\pi_{i_1}(2)\}$, and since by the induction hypothesis $K, \pi(2) \models \psi$ if $K, \{\pi(2)\} \models \psi$, the above is equivalent to $K, \{w\} \models \mathsf{EX}\psi$.

Let $\vdash$ denote a team satisfaction relation. We say that $\vdash$ is downward closed if the following holds for every Kripke structure $K$:

If $K, T \vdash \varphi$ and $T' \subseteq T$ then $K, T' \vdash \varphi$.

The proof of the following lemma is analogous with the corresponding proofs for modal and first-order dependence logic (see [23, 24]).

Lemma 6 (Downward closure). $\models$ and $\models$ are downward closed.

Proof. We proof the claim for $\models$ only. For $\models$ the argumentation is similar. The proof is by induction on $|\varphi|$.

Let $K = (W, R, \eta)$ be an arbitrary Kripke structure and $T' \subseteq T$ be some teams of $K$. The cases for literals are trivial: Assume $K, T \vdash \varphi$. Then by definition $p \in \eta(w)$ for every $w \in T$. Now since $T' \subseteq T$, clearly $p \in \eta(w)$ for every $w \in T'$. Thus $K, T' \vdash \varphi$. The case for negated propositions is completely symmetric.

The case for $\land$ is clear. For the case for $\lor \psi$ assume that $K, T \vdash \varphi \land \psi$. Now by the definition of disjunction there exist $T_1 \cup T_2 = T$ such that $K, T_1 \models \varphi$ and $K, T_2 \models \psi$. By induction hypothesis it the follows that $K, T_1 \models \varphi$ and $K, T_2 \models \psi$. Now since clearly $T' = (T_1 \cap T') \cup (T_2 \cap T')$, it follows by the semantics of the disjunction that $K, T' \models \varphi \land \psi$.

Now consider $\mathsf{PX}\varphi$. Let $T = \{t_1, \ldots, t_n\}$, where $n \in \mathbb{N}$, and assume that $K, T \models \mathsf{PX}\varphi$. We have to show that $K, T' \models \mathsf{PX}\varphi$ for every $T' \subseteq T$. By the semantics of $\mathsf{PX}\varphi$ we have that

$$\forall t \in T : \text{if } K, t \models \varphi \land \psi \text{ holds. But this follows from (1) by the induction hypothesis. The cases for $U$ and $W$ are analogous.}$$

In this article, we consider multisets of points as teams. We do this, since we believe that multisets capture the idea of multithread computation better that the use of ordinary sets would. Observe that with respect to the satisfaction relation the use of multisets has no real consequence. The proof of the following corollary is self-evident. The proof uses the fact that both satisfaction relations are downward closed.

Corollary 7. Let $\varphi$ be a CTL-formula, $\vdash \in \{\models, \models\}$. $K$ be a Kripke structure, $T$ be a team of $K$, and $T'$ be the underlying set of the multiset $T$. Then $K, T \vdash \varphi$ iff $K, T' \vdash \varphi$.

A team satisfaction relation $\vdash$ is said to be union closed if for every Kripke structure $K$, formula $\varphi$, and teams $T$ and $T'$ of $K$, the following holds:

If $K, T \vdash \varphi$ and $K, T' \vdash \varphi$ then $K, T \cup T' \vdash \varphi$.

Lemma 8 (Union closure). $\models$ is union closed.

Proof. This property again can be shown via induction on $|\varphi|$. The interesting parts of the proof are the cases for the temporal operators $\mathsf{P}[\varphi \mathsf{U}\psi]$ and $\mathsf{P}[\varphi \mathsf{W}\psi]$. We will show the proof for $\mathsf{P}[\varphi \mathsf{U}\psi]$ only. The proof for $\mathsf{W}$ is completely analogous. Now let $K, T \models \mathsf{P}[\varphi \mathsf{U}\psi]$ and $K, T' \models \mathsf{P}[\varphi \mathsf{U}\psi]$. For simplicity we show the result only for $P = E$. Let $T = \{t_1, \ldots, t_n\}$ be a team. Then $K, T \models \mathsf{E}[\varphi \mathsf{U}\psi]$ implies that there are paths $\pi_1 \in \Pi(t_1), \ldots, \pi_n \in \Pi(t_n)$ and natural numbers $k_1, \ldots, k_n$ such that $K, \{\pi_j(k_j)\} \models \psi$ and for all $1 \leq i < j \leq n$ it holds that $K, \{\pi_j(i_j)\} \models \varphi$ for $1 \leq j \leq n$. Analogously let
$T' = \{ s_1, \ldots, s_m \}$ be a team. Then $K, T' \models E[\varphi \cup \psi]$ implies that there are paths $\pi'_1 \in \Pi(s_1), \ldots, \pi'_m \in \Pi(s_m)$ and natural numbers $k'_1, \ldots, k'_m$ such that $K, \{ \pi'_i(k'_j) \} \models \psi$ and for all $1 \leq j < k'_j$, it holds that $K, \{ \pi'_i(j) \} \models \varphi$ for $1 \leq j \leq m$. Thus clearly $K, T \cup T' \models E[\varphi \cup \psi]$ and the claim follows.

This leads to the following interesting corollary which allows one to consider only the elements of the team instead of the complete team together. This will later prove to be important in the classification of the complexity of the model checking problem for asynchronous semantics.

**Corollary 9.** For every Kripke structure $K = (W, R, \eta)$ and every team $T$ of $K$ the following equivalence holds:

$$K, T \models \varphi \iff \forall t \in T : K, t \models \varphi.$$  

IV. EXPRESSIVE POWER

In this section, we discuss in more details the relationship between the expressive powers of team $CTL$ with the synchronous semantics and team $CTL$ with the asynchronous semantics.

**Definition 10.** For each $CTL$-formula $\varphi$, define

$$\mathcal{F}_\varphi^{a,k} := \{(K, T) \mid K, T \models \varphi \}$$

and

$$\mathcal{F}_\varphi^{s,k} := \{(K, T) \mid K, T \models \varphi \}.$$

We say that $\varphi$ defines the class $\mathcal{F}_\varphi^{a,k}$ in asynchronous semantics (of $CTL$). Analogously, we say that $\varphi$ defines the class $\mathcal{F}_\varphi^{s,k}$ in synchronous semantics (of $CTL$). A class $\mathcal{F}$ of pairs of Kripke structures and teams is definable in asynchronous semantics (in synchronous semantics), if there exists some $\psi \in CTL$ such that $\mathcal{F} = \mathcal{F}_\varphi^{a,k}$ (resp., $\mathcal{F} = \mathcal{F}_\varphi^{s,k}$). Furthermore, for $k \in \mathbb{N}$, define

$$\mathcal{F}_\varphi^{a,k} := \{(K, T) \mid K, T \models \varphi \land |T| \leq k \},$$

and

$$\mathcal{F}_\varphi^{s,k} := \{(K, T) \mid K, T \models \varphi \land |T| \leq k \}.$$  

We say that $\varphi$ $k$-defines the class $\mathcal{F}_\varphi^{a,k}$ (resp., $\mathcal{F}_\varphi^{s,k}$) in asynchronous (resp., synchronous) semantics (of $CTL$). The definition of $k$-definability is analogous to that of definability.

Next we will show that there exists a class $\mathcal{F}$ which is definable in asynchronous semantics, but is not definable in synchronous semantics.

**Theorem 11.** The class $\mathcal{F}_\varphi^{a,k}$ is not definable in synchronous semantics.

**Proof.** For the sake of a contradiction, assume that $\varphi$ is such that $\mathcal{F}_\varphi^{a,k} = \mathcal{F}_\varphi^{s,k}$. Consider the following Kripke model $K = (W, R, V)$, where $W = \{ 1, 2, 3 \}$, $R = \{ (2, 3) \}$, and $V(p) = \{ 1, 2 \}$. Clearly $K, \{ 1 \} \models EFp$ and $K, \{ 2 \} \models EFp$. Thus by our assumption, it follows that $K, \{ 1 \} \models \varphi$ and $K, \{ 2 \} \models \varphi$. From Corollary 9 it then follows that $K, \{ 1 \} \models \varphi$. But clearly $K, \{ 1, 2 \} \not\models EFp$.

**Corollary 12.** For $k > 1$, the class $\mathcal{F}_\varphi^{a,k}$ is not $k$-definable in synchronous semantics.

**Conjecture 13.** The class $\mathcal{F}_\varphi^{EFp}$ is not definable in asynchronous semantics.

**Theorem 14.** For every $k \in \mathbb{N}$ and $\varphi \in CTL$, the class $\mathcal{F}_\varphi^{a,k}$ is $k$-definable in asynchronous semantics.

**Proof.** Fix $k \in \mathbb{N}$ and $\varphi \in CTL$. Define

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

We will show that $\mathcal{F}_\varphi^{a,k} = \mathcal{F}_\varphi^{a,k}$. Let $K$ be an arbitrary Kripke structure and $T$ be a team of $K$ of size at most $k$. Then it holds

$$K, T \models \varphi \iff \forall w \in T : K, \{ w \} \models \varphi \iff \forall w \in T : K, \{ w \} \models \varphi \iff K, T \models \varphi'.$$

The first equivalence follows by Corollary 9, the second by Lemma 5, and the last by the semantics of disjunction and the downward closure property.

V. COMPLEXITY RESULTS

In this section we classify the problems with respect to the computational complexity. At first we start with the asynchronous semantics where each thread of the team may use differently deep paths with respect to the until operators while evaluating the formula. We will begin with model checking and will finish with satisfiability.

In the following we define the most important decision problems in these logics.

**Problem:** MC<sup>a</sup>

**Input:** A Kripke structure $K$, a team $T$ of $K$, a formula $\varphi \in CTL$.

**Question:** $K, T \models \varphi$?

**Problem:** SAT<sup>a</sup>

**Input:** A formula $\varphi \in CTL$.

**Question:** Does there exist a Kripke structure $K$ and a non-empty team $T$ of $K$ s.t. $K, T \models \varphi$?

Similarly we write MC<sup>s</sup>, resp., SAT<sup>s</sup> for the variants with synchronized semantics.

A. Model Checking

In this subsection we investigate the computational complexity of model checking. For usual $CTL$ model checking the following proposition summarizes what is known.

**Proposition (2, 21).** Model checking for $CTL$ formulas is P-complete.

At first we investigate the case for asynchronous semantics. Through combinations of the previous structural properties of $\models$ it is possible to show the same complexity degree.

**Theorem 16.** MC<sup>a</sup> is P-complete.
Proof. The lower bound is immediate from usual CTL model checking by Proposition 15. For the upper bound we apply Corollary 9 and separately use for each of the given team the usual CTL model checking algorithm.

Now we turn to the model checking problem for synchronous semantics. Here we show that the problem becomes intractable under reasonable complexity class separation assumptions, i.e., \( P \neq \text{PSPACE} \). The main idea is to exploit the synchronous semantics in a way to literally check in parallel all clauses for a given quantified Boolean formula for satisfiability for a set of relevant assignments.

Theorem 17. \( \text{MC}^* \) is \( \text{PSPACE} \)-hard.

Proof. From Stockmeyer [22] we know that validity of closed quantified Boolean formulas (QBF-VAL) of the form \( \exists x_1 \forall x_2 \cdots \exists x_n F \), where \( \exists \equiv \exists \) if \( n \) is odd, resp., \( \exists \equiv \forall \) if \( n \) is even, and \( F \) is in conjunctive normal form is \( \text{PSPACE} \)-complete.

Let \( \varphi := \exists x_1 \forall x_2 \cdots \exists x_n \wedge_{i=1}^m \forall_{j=1}^3 \ell_{i,j} \) be a closed quantified Boolean formula (QBF) and \( \exists \equiv \exists \) if \( n \) is odd, resp., \( \exists \equiv \forall \) if \( n \) is even. Now define a corresponding structure \((W, R, \eta)\) which is also partly shown in Figure 2 as follows:

\[
W := \bigcup_{i=1}^n \left\{ (w_{i}^1, w_{i,1}^2) \mid 1 \leq j \leq i \right\} \cup \left\{ (w_{i}^1, w_{i,j}^2) \mid 1 \leq j \leq m \right\} \cup \left\{ (w_{j}^1, w_{j,k}^2) \mid 1 \leq j \leq m, 1 \leq i \leq 3, 1 \leq k \leq 2 \right\}
\]

\[
R := \bigcup_{i=1}^n \left\{ (w_{i}^1, w_{i,j}^2) \mid 1 \leq j \leq i \right\}
\]

In Figure 3 an example for the reduction is shown for the instance \( \exists x_1 \forall x_2 \exists x_3 (x_1 \lor x_2 \lor \neg x_3) \land (x_1 \lor \neg x_2 \lor \neg x_3) \land (x_1 \lor \neg x_2 \lor x_3) \) which is a valid QBF and hence belongs to QBF-VAL.

The left three branching systems choose the value of the \( x_i \)'s. Deciding for the left/right path means setting \( x_i = 1/0 \).

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

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

Now we will prove that \( K, T \models \varphi \). Observe that \( T = \{ w_1^1, \ldots, w_n^1, w_1^2 \} \). For \( w_1^1 \) there is no choice in the next \( n \) steps defined by the prefix of \( \varphi \). For \( w_1^1, \ldots, w_n^1 \) we decide as follows depending on the assignment. Denote by \( f(\varphi) = (W, R, \eta, T, \varphi) \) the value of the reduction function and denote with \( K \) the structure \((W, R, \eta)\).

Now that during the evaluation of \( \varphi \) w.r.t. \( T \) and \( K \) in the first \( n \) CTL operators of \( \varphi \) the AX operators are treated in the proof now as EX. This is because here we decide for the relevant assignments according to \( S \). Hence if \( s(x_i) = 1 \) then choose in step \( i \) of this prefix from \( w_i^1 \) the successor world \( w_i^1,1 \). If \( s(x_i) = 0 \) then choose \( w_i^1,2 \) instead.

Now after \( n \) steps the current team \( T \) then is \( \{ w_{n+1}^1 \} \cup \{ w_{n+1,1}^1 \mid s(x_i) = 1, 1 \leq i \leq n \} \cup \{ w_{n+2,1}^1 \mid s(x_i) = 0, 1 \leq i \leq n \} \), and \( s \) will agree with the assignment \( s \). In the next step the team branches now on all clauses for \( F \) and becomes \( \{ w_1^1 \mid 1 \leq j \leq m \} \cup \{ w_{n+1,1}^1 \mid s(x_i) = 1, 1 \leq i \leq n \} \cup \{ w_{n+1,2}^1 \mid s(x_i) = 0, 1 \leq i \leq n \} \). Now continuing with an EX in \( \varphi \) the team members of the “formula” (we here refer to the elements \( \{ w_1^1 \mid 1 \leq j \leq m \} \) of the team) have to decide for a literal which satisfies the respective clause. As \( s \models F \) this must be possible. W.l.o.g. assume that in clause \( C_j \) the literal \( l_j \) satisfies \( C_j \) by \( s(\ell_j) = 1 \) for \( 1 \leq j \leq m \) (denote with \( s(\ell) \) the value \( 1 - s(x) \) if \( x \) is the corresponding variable to literal \( \ell_j \)). Let \( \text{ind}(\ell_j) \in \{ 1, 2, 3 \} \) denote the “index” of \( \ell_j \) in \( C_j \), i.e., the value \( i \in \{ 1, 2, 3 \} \) such that \( \ell_j = \ell_{i,j} \) in \( F \). Then we choose the world \( w_{j,1,\text{index}(\ell_j),1}^1 \) as a successor from \( w_{i,1}^1 \) for \( 1 \leq j \leq m \).

For the (“variable” team members) \( w_{n+1,2}^1 \) with \( k \in \{ 1, 2 \} \) we have no choice and proceed to \( w_{n+3,k}^1 \). Now we have to satisfy the remainder of \( \varphi \) which is \( \bigwedge_{i=1}^n \text{EX} x_i \). Observe that for variable team members \( w_{n+3,1}^1 \) only has \( x_i \) labeled in the current world and not in the successor world \( w_{n+4,1}^1 \), i.e., \( x_i \notin \eta(w_{n+4,1}^1) \).

Symmetrically this is true for the \( w_{n+3,2}^1 \) worlds but \( x_i \notin \eta(w_{n+3,2}^1) \) and \( x_i \notin \eta(w_{n+4,2}^1) \).

Hence “staying” in the world (hence immediately satisfying the EX \( x_i \)) means setting \( x_i \) to true by \( s \) whereas making a further step means setting \( x_i \) to false by \( s \).

Further observe for the formula team members we have depending on the value of \( s(\ell_j) \) that \( x_i \in \eta(w_{n+4,1}^1) \).
and \( x \notin \eta(w_{n+3,\text{index}(\ell_j),2}) \) if \( s(\ell_j) = 1 \), and \( x \notin \eta(w_{n+3,\text{index}(\ell_j),1}) \) and \( x \in \eta(w_{n+3,\text{index}(\ell_j),2}) \) if \( s(\ell_j) = 0 \).

Thus according to synchronous semantics the step depth w.r.t. \( \eta \) to the usual until cases and just use non-determinism to agree on the quantifier prefix of \( \varphi \). Hence \( K, T \models \varphi \).

For the direction “\( \Leftarrow \)” observe that with similar arguments we can deduce from the “final” team in the end what has to be a satisfying assignment depending on the choices of \( w_{n+3,k}^x \) and \( k \in \{1,2\} \). Hence by construction any of these assignments satisfies \( F \). Let again denote by \( S \) a set of teams which satisfy \( \text{AXEX} \wedge \bigwedge_{i=1}^n x_i \) according to the prefix of \( n \text{CTL} \) operators. Then define a set \( S' \) of assignments from \( S \) by getting the assignment \( s \) from the team \( t \in S \) by setting \( s(x_i) = 1 \) if there is a world \( w_{n+1,1}^x \) in \( t \) and otherwise \( s(x_i) = 0 \). Then it analogously follows that \( s \models F \). \( S' \) also agrees on the quantifier prefix of \( \varphi \). Hence \( \varphi \in \text{QBF-VAL} \).

**Theorem 18.** \( \text{MC}^s \) is in PSPACE.

**Proof.** The following PSPACE-algorithm solves \( \text{MC}^s \). The weak until cases are omitted as they can be defined analogously to the usual until cases and just use non-determinism to operate on the disjunction.

The procedure \( \text{s-check} \) (see Algorithm 1) computes for a given Kripke structure \( K \), a team \( T \) and a formula \( \varphi \) if

**Figure 2. General view on the created Kripke structure in the proof of Theorem 17.**

**Figure 3. Example structure built in proof of Lemma 17.**

\( K, T \models \varphi \). The correctness of the algorithm can be verified by induction over the formula \( \varphi \) as the different cases in the procedure \( \text{s-check} \) merely restate the semantical definition of our team logic.

For the case \( \varphi = \bigvee \{ \pi \in \Pi \} ^ t \) by definition we need to check if there exists paths \( \pi_1 \in \Pi(t_1), \ldots, \pi_n \in \Pi(t_n) \) and a \( k \in \mathbb{N} \) such that

\[
\forall 1 \leq i \leq k : K, \bigvee_{1 \leq j \leq n} \{ \pi \in \Pi(t_j) \} \models \alpha.
\]

The algorithm checks exactly the same conditions, but guesses the number \( k \) only up to \( |W|^{|T|} \). We show this is sufficient as the size \( |T| \) of the team does not change. Suppose such a
Algorithm 1: Model checking algorithm for $\mathcal{MC}^a$

1. **Procedure** $\text{succ}(\text{Structure } K=(W,R,\eta), \text{team } T)$:
2. guess a multiset $T'$ with $|T'| = |T'|$ s.t. i.a. $t \in T$ there exists a $u \in T'$ with $tRu$ and vice versa;
3. return $T'$;

4. **Procedure** $s$-check (Kripke structure $K=(W,R,\eta)$, team $T$, formula $\phi$):
5. if $\phi = T$ then return 1;
6. if $\phi = \bot$ then return $T = \emptyset$;
7. if $\phi = p$ then return $\forall w \in T : p \in \eta(w)$;
8. if $\phi = \neg p$ then return $\forall w \in T : p \notin \eta(w)$;
9. if $\phi = \alpha \land \beta$ then
10. return $s$-check($K,T,\alpha$)$\land$s-check($K,T,\beta$);
11. if $\phi = \alpha \lor \beta$ then
12. guess $T_1 \cup T_2 = T$;
13. return $s$-check($K,T_1,\alpha$)$\land$s-check($K,T_2,\beta$);
14. if $\phi = \text{EX} \alpha$ then
15. $T' \leftarrow \text{succ}(K,T)$;
16. return $s$-check($K,T',\alpha$);
17. if $\phi = \text{AX} \alpha$ then
18. bool $v \leftarrow 1$;
19. for every possible guess $T' \leftarrow \text{succ}(K,T)$ do
20. $v \leftarrow v \land s$-check($K,T',\alpha$);
21. return $v$;
22. if $\phi = \text{E}[\alpha \cup \beta]$ then
23. guess a binary number $k \in [0,|W|^{|T|}]$, $v \leftarrow 1$, and $T_{last} \leftarrow T$;
24. for $1 \leq i < k$ do
25. $T' \leftarrow \text{succ}(K,T_{last})$;
26. $v \leftarrow v \land s$-check($K,T',\alpha$);
27. $T_{last} \leftarrow T'$ and $i \leftarrow i+1$;
28. if $i = k$ then $T_{last} \leftarrow \text{succ}(K,T_{last})$;
29. return $v \land s$-check($K,T_{last},\beta$);
30. if $\phi = \text{A}[\alpha \cup \beta]$ then
31. let $T_{last} \leftarrow T$, and bool $v \leftarrow 1$;
32. for $1 \leq i \leq |W|^{|T|}$ do
33. for every possible guess $T' \leftarrow \text{succ}(K,T)$ do
34. $T_{last} \leftarrow T'$, and guess $A \in \{0,1\}$;
35. if $A$ then $v \leftarrow v \land s$-check($K,T',\alpha$);
36. else return $v \land s$-check($K,T',\beta$);
37. $i \leftarrow i+1$;
38. return $v \land s$-check($K,T',\beta$);

$k$ exists but $k > |W|^{|T|}$, then there are $i_1 < i_2$ such that all paths have a loop from $i_1$ to $i_2$, i.e.,

$$\forall 1 \leq j \leq n : \pi_{i_1}(i_1) = \pi_{i_2}(i_2).$$

We can generate a new set of paths $\pi'_{i_1} \in \Pi(t_1), \ldots, \pi'_{i_n} \in \Pi(t_n)$ by removing the loop from $i_1$ to $i_2$ and let $k' = k - i_2 + i_1$. Then these paths and the new constant $k'$ also satisfy the conditions above. We can repeat this process until we gained a constant less then $|W|^{|T|}$. Hence if there is such a $k$ we can find a $k \leq |W|^{|T|}$.

Similar it suffices in the case $\phi = A[\alpha \cup \beta]$ to verify that $\beta$ is satisfied after at most $|W|^{|T|}$ steps.

Also our algorithm runs in alternating polynomial time; the nondeterministic choices occur in the Until-case and in the procedure $\text{succ}$, where they correspond to existential and universal quantifications. Hence the algorithm runs in $\mathcal{PSPACE}$.

**Corollary 19.** $\mathcal{MC}^a$ is $\mathcal{PSPACE}$-complete.

**B. Satisfiability**

The following proposition summarises what is known about usual $\mathcal{CTL}$ satisfiability.

**Proposition 20** ([6, 19]). Satisfiability for $\mathcal{CTL}$ formulas is $\mathcal{EXPTIME}$-complete.

For the parallel computation tree logic the computational complexity of the satisfiability problem is proven to be the same as for $\mathcal{CTL}$.

**Theorem 21.** $\mathcal{SAT}^a$ and $\mathcal{SAT}^a$ are $\mathcal{EXPTIME}$-complete.

**Proof.** In both cases the problem merely asks whether there exists a Kripke structure $K$ and a non-empty team $T$ of $K$ such that $K,T \models \phi$, resp., $K,T \models \phi$ for given formula $\phi \in \mathcal{CTL}$. By Lemma 6 we can just quantify for a singleton sized team, i.e., $|T| = 1$. By Lemma 5 we immediately obtain the same complexity bounds from usual satisfiability for $\mathcal{CTL}$. Hence Proposition 20 applies and proves the theorem. 

**VI. FUTURE WORK**

The tautology or validity problem of given formulas in this new logic is quite interesting and seems to have a larger complexity than satisfiability however we have not been able to prove a result yet. Formally the corresponding problems are defined as follows:

<table>
<thead>
<tr>
<th>Problem:</th>
<th>$\mathcal{VAL}^a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input:</td>
<td>A formula $\varphi \in \mathcal{TCTL}$.</td>
</tr>
<tr>
<td>Question:</td>
<td>Does for all Kripke structures $K$ and teams $T$ of $K$ s.t. $K,T \models \varphi$?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem:</th>
<th>$\mathcal{VAL}^a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input:</td>
<td>A formula $\varphi \in \mathcal{TCTL}$.</td>
</tr>
<tr>
<td>Question:</td>
<td>Does for all Kripke structures $K$ and teams $T$ of $K$ s.t. $K,T \models \varphi$?</td>
</tr>
</tbody>
</table>

In the context of propositional and modal logic the computational complexity of the validity problem has been determined.
by Virtema [25]. Virtema shows that the problem for propositional dependence logic is \textsc{NEXPTIME}-complete whereas for (extended) modal dependence logic it is \textsc{NEXPTIME}-hard and in \textsc{NEXPTIME}^\text{NP}.

As further research questions one might consider to answer 13 which we currently were not able to prove. Intuitively here the weak until operator makes the argument quite difficult to prove due to the possibility of infinite computation paths (informally hence the G operator).

Recently there is another quite prospering area in logic where team semantics have been extensively used: Dependence Logic. This logic was introduced by Väänänen [23] in 2007 to express dependencies between variables in systems. There are several important applications of this logic and semantics, e.g., computational biology, data base systems, social choice theory, and cryptography. Also there exists a modal logic variant which also refers back to the work of Väänänen [24]—this time from 2008. In Dependence Logic a novel atom has been introduced (and also gave the logic its name) which is known as a dependence atom \text{dep}(p_1, \ldots, p_n) stating that the values of \( p_1, \ldots, p_{n-1} \) determine the value of \( p_n \), where \( p_1, \ldots, p_n \) are propositions. This operator might be a further step to construct a more flexible and elegant Parallel Computation Tree Logic which can express several interesting dependencies relevant to practice. Formally this kind of operator is defined in our terms as follows if \( K = (W, R, \eta) \) is a Kripke structure, \( T = \{t_1, \ldots, t_n\} \) is a team, and \( \varphi_1, \ldots, \varphi_n \) are \textsc{CTL} formulas, then \( K, T \models \text{dep}(\varphi_1, \ldots, \varphi_n) \) holds if and only if

\[
\forall t_1, t_2 \in T : \bigwedge_{i=1}^{n-1} (K, \{t_i\} \models \varphi_i \iff K, \{t_2\} \models \varphi_i) \implies (K, \{t_1\} \models \varphi_n \iff K, \{t_2\} \models \varphi_n).
\]

This definition strictly follows the notion of what is known as the Extended Modal Dependence Logic \( \mathcal{EMDL} \) introduced by Ebbing et al. [4].

It is well-known that there are different possibility to consider the model checking complexity of a logic. System complexity just considers the computational complexity for the case of a fixed formula whereas specification complexity fixes the underlying Kripke structure. We considered in this paper the combined complexity where both parts belong to the given input. Yet the other two approaches might give more specific insights into the intractability of the synchronous model checking case we investigated. In particular the study of so-to-speak team complexity, where the team or the team size is assumed to be fixed, might as well be of independent interest.

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

\section{Conclusion}

In the present paper we made a first step of defining a Parallel Computation Tree Logic which illustrates the reasoning of the well-known and established Computation Tree Logic CTL for multiple processor instances (or in parallel computing threads of a program). Further we examined two possibilities to define the semantics of the logics: asynchronous and synchronous. With respect to the latter semantics the computation of the processors is synchronised via centralised clock which is important with respect to the eventuality operator until. From satisfiability perspective the complexity of the new logic behaves similar as \textsc{CTL}. However one might consider a different kind of satisfiability question: given a formula \( \varphi \) and a team size asking if there exists a Kripke structure \( K \) and a team \( T \) of size \( k \) in \( K \) such that \( K, T \models \varphi \), resp., \( K, T \not\models \varphi ? \) Yet the use of the multi-set notion easily tames this approach and then lets us conclude with the same result as in Theorem 21.

Surprisingly the complexity behavior of the Parallel Computation Tree Logic with respect to model checking is somewhat different to usual \textsc{CTL}. For the synchronous case the problem becomes quite difficult namely \textsc{PSPACE}-complete. We proved the lower bound by a reduction from \textsc{QBF} validity and constructed a Ladner [14] styled algorithm to prove the upper bound. One might guess that the complexity of the asynchronous case of model checking is also different to the quite efficient \textsc{CTL} case (which is \textsc{P}-complete). However using closure properties of the relation \( \models \) allows us to separately check for each team member if it is satisfied in the given structure. This just leads to a multiple (polynomially) application of the usual \textsc{CTL} model checking algorithm and thereby proves the same upper bound.

\section*{Acknowledgements}

The work is partly supported by Jenny and Antti Wihuri Foundation.

\section*{References}


