Computation Tree Logic for Synchronization Properties*

Krishnendu Chatterjee[1] and Laurent Doyen[2]

1 **IST Austria**
2 **LSV, ENS Cachan & CNRS, France**

Abstract

We present a logic that extends CTL (Computation Tree Logic) with operators that express synchronization properties. A property is synchronized in a system if it holds in all paths of a certain length. The new logic is obtained by using the same path quantifiers and temporal operators as in CTL, but allowing a different order of the quantifiers. This small syntactic variation induces a logic that can express non-regular properties for which known extensions of MSO with equality of path length are undecidable. We show that our variant of CTL is decidable and that the model-checking problem is in $\Delta_3^{\mathrm{P}} = \mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}}$, and is hard for the class of problems solvable in polynomial time using a parallel access to an NP oracle. We analogously consider quantifier exchange in extensions of CTL, and we present operators defined using basic operators of CTL* that express the occurrence of infinitely many synchronization points. We show that the model-checking problem remains in Δ_3^{P}. The distinguishing power of CTL and of our new logic coincide if the Next operator is allowed in the logics, thus the classical bisimulation quotient can be used for state-space reduction before model checking.

1998 ACM Subject Classification F.4.1 Temporal Logic

Keywords and phrases Computation Tree Logic, Synchronization, model-checking, complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.XXX

1 Introduction

In computer science, it is natural to view computations as a tree, where each branch represents an execution trace, and all possible execution traces are arranged in a tree. To reason about computations, the logical frameworks that express properties of trees have been widely studied [10, 20, 24], such as CTL, CTL*, μ-calculus, MSO, etc. These logics can express ω-regular properties about trees.

A key advantage of logics is to provide concise and formal semantics, and a rigorous language to express properties of a system. For example, the logic CTL is widely used in verification tools such as NuSMV [9], and hyperproperties, i.e. tree-based properties that cannot be defined over individual traces, are relevant in security [11, 12].

One key property that has been studied in different contexts is the property of synchronization, which intuitively requires that no matter how the system behaves it synchronizes to a common good point. Note that the synchronization property is inherently a tree-based property, and is not relevant for traces. Synchronization has been studied for automata [25, 8], probabilistic models such as Markov decision processes [15, 16], as well as partial-information,

* This research was partially supported by Austrian Science Fund (FWF) NFN Grant No S11407-N23 (RiSE/SHiNE), ERC Start grant (279307: Graph Games), Vienna Science and Technology Fund (WWTF) through project ICT15-003, and European project Cassting (FP7-601148). Full version [6].

weighted, and timed models [19, 17, 14], and has a rich collection of results as well as open problems, e.g., Černý's conjecture about the length of synchronizing words in automata is one of the long-standing and well-studied problems in automata theory [5, 25]. A natural question is how can synchronization be expressed in a logical framework.

First, we show that synchronization is a property that is not ω-regular. Hence it cannot be expressed in existing tree-based logics, such as MSO, CTL*, etc. A natural candidate to express synchronization in a logical framework is to consider MSO with quantification over path length. Unfortunately the quantification over path length in MSO leads to a logic for which the model-checking problem is undecidable [23, Theorem 11.6]. Thus an interesting question is how to express synchronization in a logical framework where the model-checking problem is decidable.

Contributions. In this work we introduce an elegant logic, obtained by a natural variation of CTL. The logic allows to exchange the temporal and path quantifiers in classical CTL formulas. For example, consider the CTL formula $\forall Fq$ expressing the property that in all paths there exists a position where q holds (quantification pattern $\forall\text{paths} \cdot \exists\text{position}$). In our logic, the formula $F\forall q$ with quantifiers exchanged expresses that there exists a position k such that for all paths, q holds at position k (quantification pattern $\exists\text{position} \cdot \forall\text{paths}$), see Figure 1a. Thus q eventually holds in all paths at the same position, expressing that the paths are eventually synchronized.

We show that the model-checking problem is decidable for our logic, which we show is in $\Delta_3^{\mathrm{P}} = \mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}}$ (in the third level of the polynomial hierarchy) and is hard for the class $\mathrm{P}_{\parallel}^{\mathrm{NP}}$ of problems solvable in polynomial time using a parallel access to an NP oracle (Theorem 1). The problems in $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}}$ can be solved by a polynomial-time algorithm that uses an oracle for a problem in $\mathrm{NP}^{\mathrm{NP}}$, and the problems in $\mathrm{NP}^{\mathrm{NP}}$ can be solved by a non-deterministic polynomial-time algorithm that uses an oracle for an NP-complete problem; the problems in $\mathrm{P}_{\parallel}^{\mathrm{NP}}$ can be solved by a polynomial-time algorithm that works in two phases, where in the first phase a list of queries is constructed, and in the second phase the queries are answered by an NP oracle (giving a list of yes/no answers) and the algorithm proceeds without further calling the oracle [26, 21].

We present an extension of our logic that can express the occurrence of infinitely many synchronization points (instead of one as in eventually sychronizing), and the absence of synchronization from some point on, with the same complexity status (Section 3). These properties are the analogue of the classical liveness and co-liveness properties in the setting of synchronization. We show that such properties cannot be expressed in the basic logic (Section 4). In Section 6, we consider the possibility to further extend our logic with synchronization to CTL*, and show that the exchange of quantifiers in CTL* formulas would lead to either a counter-intuitive semantics, or an artificial logic that would be inelegant.

We study the distinguishing power of the logics in Section 5, that is the ability of the logics, given two models, to provide a formula that holds in one model, and not in the other. The distinguishing power is different from the expressive power of a logic, as two logics with the same expressive power have the same distinguishing power but not vice versa. The distinguishing power can be used for state-space reduction before running a model-checking algorithm, in order to obtain a smaller equivalent model, that the logic cannot distinguish from the original model, and thus for which the answer of the model-checking algorithm is the same. We show that if the Next operator is allowed in the logic, then the distinguishing power coincides with that of CTL (two models are indistinguishable if and only if they are bisimilar), and if the Next operator is not allowed, then the distinguishing power lies between bisimulation and stuttering bisimulation, and is NP-hard to decide. In particular,

it follows that with or without the Next operator the state-space reduction with respect to bisimulation, which is computable in polynomial time, is sound for model-checking. Detailed proofs are available in [6].

2 CTL + Synchronization

We introduce the logic CTL+Sync after presenting basic definitions related to Kripke structures. A *Kripke structure* is a tuple $K = \langle T, \Pi, \pi, R\rangle$ where T is a finite set of states, Π is a finite set of atomic propositions, $\pi : T \to 2^{\Pi}$ is a labeling function that maps each state t to the set $\pi(t)$ of propositions that are true at t, and $R \subseteq T \times T$ is a transition relation. We denote by $R(t) = \{t' \mid (t, t') \in R\}$ the set of successors of a state t according to R, and given a set $s \subseteq T$ of states, let $R(s) = \bigcup_{t\in s} R(t)$. A Kripke structure is *deterministic* if $R(t)$ is a singleton for all states $t \in T$. A *path* in K is an infinite sequence $\rho = t_0 t_1 \dots$ such that $(t_i, t_{i+1}) \in R$ for all $i \geq 0$. For $n \in \mathbb{N}$, we denote by $\rho + n$ the suffix $t_n t_{n+1} \dots$.

2.1 Syntax and semantics

In the CTL operators, a path quantifier always precedes the temporal quantifiers (e.g., $\exists\mathcal{U}$ or $\forall\mathcal{U}$). We obtain the logic CTL+Sync from traditional CTL by allowing to switch the order of the temporal and path quantifiers. For example, the CTL formula $p\,\forall\mathcal{U}\,q$ holds in a state t if for all paths ($\forall$) from t, there is a position where q holds, and such that p holds in all positions before ($\mathcal{U}$). In the CTL+Sync formula $p\mathcal{U}\forall\, q$, the quantifiers are exchanged, and the formula holds in t if there exists a position k, such that for all positions $j < k$ before ($\mathcal{U}$), in all paths ($\forall$) from t, we have that q holds at position k and p holds at position j (see Figure 1d). Thus the formula $p\mathcal{U}\forall\, q$ requires that q holds synchronously after the same number of steps in all paths, while the formula $p\,\forall\mathcal{U}\,q$ does not require such synchronicity across several paths.

The syntax of the formulas in CTL+Sync is as follows:

$$\varphi ::= p \mid \neg\varphi_1 \mid \varphi_1 \vee \varphi_2 \mid Q\mathcal{X}\,\varphi_1 \mid \varphi_1\, Q\mathcal{U}\,\varphi_2 \mid \varphi_1\,\mathcal{U}Q\,\varphi_2$$

where $p \in \Pi$ and $Q \in \{\exists, \forall\}$. We define true and additional Boolean connectives as usual, and let

- $\exists F\varphi \equiv \mathsf{true}\,\exists\mathcal{U}\,\varphi$, and $F\exists\varphi \equiv \mathsf{true}\mathcal{U}\exists\,\varphi$, etc.
- $\exists G\varphi \equiv \neg\forall F\neg\varphi$, etc.

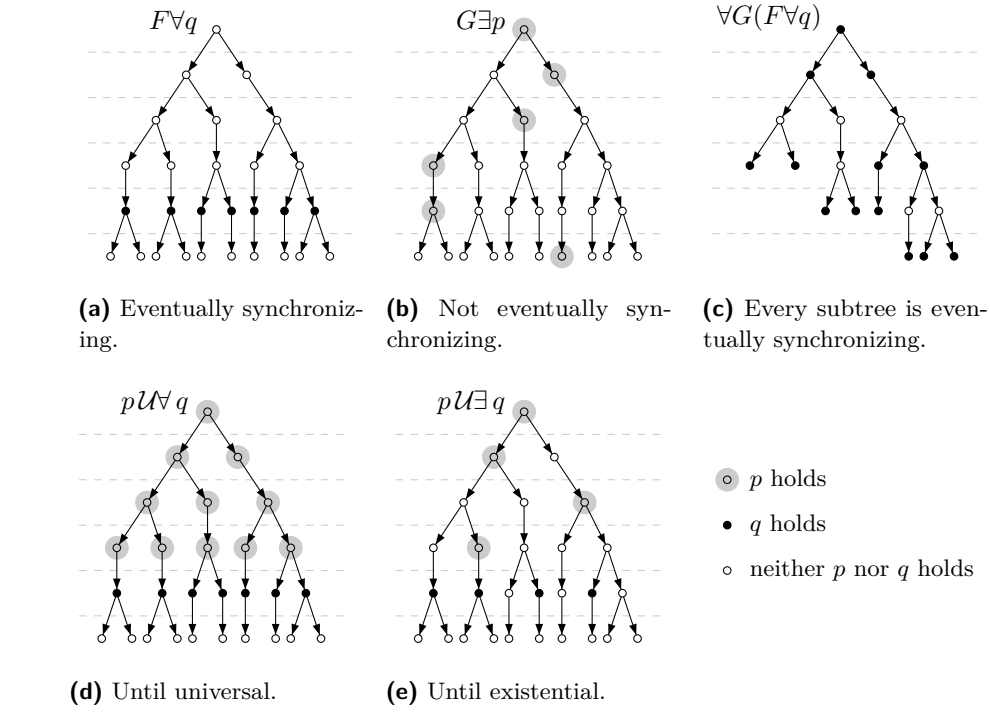
Note that the Next operators $Q\mathcal{X}$ has only one quantifier, and thus there is no point in switching quantifiers or defining an operator $\mathcal{X}Q$.

Given a Kripke structure $K = \langle T, \Pi, \pi, R\rangle$, and a state $t \in T$, we define the satisfaction relation $\models$ as follows. The first cases are standard and exist already in CTL:

- $K, t \models p$ if $p \in \pi(t)$.
- $K, t \models \neg\varphi_1$ if $K, t \not\models \varphi_1$.
- $K, t \models \varphi_1 \vee \varphi_2$ if $K, t \models \varphi_1$ or $K, t \models \varphi_2$.
- $K, t \models \exists\mathcal{X}\,\varphi_1$ if $K, t' \models \varphi_1$ for some $t' \in R(t)$.
- $K, t \models \forall\mathcal{X}\,\varphi_1$ if $K, t' \models \varphi_1$ for all $t' \in R(t)$.

The interesting new cases are built using the until operator of CTL:

- $K, t \models \varphi_1\,\exists\mathcal{U}\,\varphi_2$ if there exists a path $t_0 t_1 \dots$ in K with $t_0 = t$ and there exists $k \geq 0$ such that: $K, t_k \models \varphi_2$, and $K, t_j \models \varphi_1$ for all $0 \leq j < k$.



(a) Eventually synchronizing. **(b)** Not eventually synchronizing. **(c)** Every subtree is eventually synchronizing.

(d) Until universal. **(e)** Until existential.

Figure 1 Formulas of CTL+Sync.

- $K, t \models \varphi_1 \,\mathcal{U}\exists\, \varphi_2$ if there exists $k \geq 0$ such that for all $0 \leq j < k$, there exists a path $t_0 t_1 \ldots$ in K with $t_0 = t$ such that $K, t_j \models \varphi_1$ and $K, t_k \models \varphi_2$.
- $K, t \models \varphi_1 \,\forall\mathcal{U}\, \varphi_2$ if for all paths $t_0 t_1 \ldots$ in K with $t_0 = t$, there exists $k \geq 0$ such that: $K, t_k \models \varphi_2$, and $K, t_j \models \varphi_1$ for all $0 \leq j < k$.
- $K, t \models \varphi_1 \,\mathcal{U}\forall\, \varphi_2$ if there exists $k \geq 0$ such that for all $0 \leq j < k$ and for all paths $t_0 t_1 \ldots$ in K with $t_0 = t$, we have $K, t_j \models \varphi_1$ and $K, t_k \models \varphi_2$.

We often write $t \models \varphi$ when the Kripke structure K is clear from the context. Examples of formulas are given in Figure 1. The examples show the first steps of the unravelling of Kripke structures defined over atomic propositions $\{p, q\}$. The formula $F\forall q$ expresses that q eventually holds synchronously on all paths, after the same number of steps (Figure 1a). This is different from the CTL formula $\forall Fq$, which expresses that all paths eventually visit a state where q holds, but not necessarily after the same number of steps in all paths. The dual formula $G\exists p$ requires that at every depth (i.e., for all positions k), there exists a path where p holds at depth k (Figure 1b). On the other hand note that $F\exists q \equiv \exists Fq$ and dually $G\forall p \equiv \forall Gp$. Another example is the formula $\forall G(F\forall q)$ expressing that every subtree is eventually synchronizing (Figure 1c). The until universal formula $p\,\mathcal{U}\forall\, q$ holds if q holds at a certain position in every path (like for the formula $F\forall q$), and p holds in all positions before (Figure 1d). The until existential formula $p\,\mathcal{U}\exists\, q$ says that it is possible to find path(s) where q holds at the same position, and such that for all smaller positions there is one of those paths where p holds at that position (Figure 1e).

▶ Remark. The definition of CTL+Sync, although very similar to the definition of CTL, interestingly allows to define non-regular properties, thus not expressible in CTL (or even

in MSO over trees). It is easy to show using a pumping argument that the property $F\forall q$ of eventually sychronizing is not regular (Figure 1a). This property of eventually sychronizing can be expressed in MSO extended with a length predicate, by a formula such as $\exists \rho \in T^* \cdot \forall \rho' \in T^* : |\rho| = |\rho'| \implies q(\rho')$ where $T = \{0,1\}$ and $q(\cdot)$ is a monadic predicate for the proposition q over the binary tree T^*, where $q(\rho)$ means that q holds in the last state of ρ. However, model-checking for the logic MSO extended with the "equal-length" predicate p defined by $p(\rho,\rho') \equiv |\rho| = |\rho'|$ is undecidable [23, Theorem 11.6]. In contrast, we show in Theorem 1 that the logic CTL+Sync is decidable.

2.2 Model-checking

Given a CTL+Sync formula φ, a Kripke structure K, and a state t, the *model-checking problem for CTL+Sync* is to decide whether $K, t \models \varphi$ holds.

Model-checking of CTL+Sync can be decided by considering a powerset construction for the Kripke structure, and evaluating a CTL formula on it. For example, to evaluate a formula $\varphi_1 \mathcal{U} \forall \varphi_2$ from state t_I in a Kripke structure K, it suffices to consider the sequence $s_1 s_2 \dots$ defined by $s_1 = \{t_I\}$ and $s_{i+1} = R(s_i)$ for all $i \geq 1$, where a set s is labeled by φ_1 if $K, t \models \varphi_1$ for all $t \in s$ (and analogously for φ_2). The formula $\varphi_1 \mathcal{U} \forall \varphi_2$ holds in t_I if and only if the formula $\varphi_1 \mathcal{U} \varphi_2$ holds in the sequence $s_1 s_2 \dots$ (note that on a single sequence the operators $\forall\mathcal{U}$ and $\exists\mathcal{U}$ are equivalent, thus we simply write $\mathcal{U}$).

For the formula $\varphi_1 \mathcal{U} \exists \varphi_2$, intuitively it holds in t_I if there exists a set P of finite paths $\rho_1, \rho_2, \dots, \rho_n$ from t_I in K, all of the same length k, such that φ_2 holds in the last state of ρ_i for all $1 \leq i \leq n$, and for every $1 \leq j < k$ there is a path ρ_{i_j} such that φ_1 holds in the jth state of ρ_{i_j}. To evaluate $\varphi_1 \mathcal{U} \exists \varphi_2$ from t_I, we construct the Kripke structure $2^K = \langle 2^T, \{\varphi_1, \varphi_2\}, \pi, \hat{R}\rangle$ where $(s, s') \in \hat{R}$ if for all $t \in s$ there exists $t' \in s'$ such that $(t, t') \in R$, thus we have to choose (nondeterministically) at least one successor from each state in s, that is for every set P of paths $\rho_1, \rho_2, \dots, \rho_n$ as above, there is a path $s_1, s_2, \dots, s_k$ (with $s_1 = \{t_I\}$) in 2^K where the sets s_i are obtained by following simultaneously the finite paths $\rho_1, \dots, \rho_n$, thus such that s_i is the set of states at position i of the paths in P. The path $s_1, s_2, \dots, s_k$ in 2^K corresponds to a set P of finite paths in K that show that $\varphi_1 \mathcal{U} \exists \varphi_2$ holds if (1) φ_2 holds in all states of s_k, and (2) φ_1 holds in some state of s_i $(i = 1, \dots, k-1)$. Hence we define the labeling function π in 2^K as follows: for all $s \in 2^T$ let $\varphi_2 \in \pi(s)$ if $K, t \models \varphi_2$ for all $t \in s$, and let $\varphi_1 \in \pi(s)$ if $K, t \models \varphi_1$ for some $t \in s$. Finally it suffices to check whether the CTL formula $\varphi_1 \exists \mathcal{U} \varphi_2$ holds in 2^K from $\{t_I\}$.

This approach gives an exponential algorithm, and even a PSPACE algorithm by exploring the powerset construction on the fly. However, we show that the complexity of the model-checking problem is much below PSPACE. For example our model-checking algorithm for the formula $F\forall q$ relies on guessing a position $k \in \mathbb{N}$ (in binary) and checking that q holds on all paths at position k. To compute the states reachable after exactly k steps, we compute the kth power of the transition matrix $M \in \{0,1\}^{T \times T}$ where $M(t,t') = 1$ if there is a transition from state t to state t'. The power M^k can be computed in polynomial time by successive squaring of M. For this formula, we obtain an NP algorithm. For the whole logic, combining the guessing and squaring technique with a dynamic programming algorithm that evaluates all subformlas, we obtain an algorithm in $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}}$ for the model-checking problem [6]. We present a hardness result for the class $\mathrm{P}_{\|}^{\mathrm{NP}}$ of problems solvable in polynomial time using a parallel access to an NP oracle [26, 21].

▶ **Theorem 1.** *The model-checking problem for CTL+Sync lies in $P^{NP^{NP}}$ and is $P_{\|}^{NP}$-hard.*

The complexity lower bounds for the model-checking problem in Theorem 1 are based on Lemma 2 where we establish complexity bounds for fixed formulas.

▶ **Lemma 2.** *Let $p, q \in \Pi$ be two atomic propositions. The model-checking problem is:*
- *NP-complete for the formulas $p\mathcal{U}\forall q$ and $F\forall q$,*
- *DP-hard for the formula $p\mathcal{U}\exists q$, and*
- *coNP-complete for the formula $G\exists q$.*

Proof. We prove the hardness results (complexity lower bounds), since the complexity upper bounds follow from the proof of Theorem 1.

The proof technique is analogous to the NP-hardness proof of [22, Theorem 6.1], and based on the following. Given a Boolean propositional formula ψ over variables $x_1, \ldots, x_n$, consider the first n prime numbers $p_1, \ldots, p_n$. For a number $z \in \mathbb{N}$, if $z \mod p_i \in \{0, 1\}$ for all $1 \leq i \leq n$, then the binary vector $(z \mod p_1, \ldots, z \mod p_n)$ defines an assignment to the variables of the formula. Note that conversely, every such binary vector can be defined by some number $z \in \mathbb{N}$ (by the Chinese remainder theorem).

NP-hardness of $F\forall q$ (and thus of $p\mathcal{U}\forall q$). The proof is by a reduction from the Boolean satisfiability problem 3SAT which is NP-complete [13]. Given a Boolean propositional formula ψ in CNF, with set C of (disjunctive) clauses over variables $x_1, \ldots, x_n$ (where each clause contains three variables), we construct a Kripke structure K_ψ as follows: for each clause $c \in C$, we construct a cycle $t_0, t_1, \ldots, t_{r-1}$ of length $r = p_u \cdot p_v \cdot p_w$ where the three variables in the clause are x_u, x_v, and x_w. We call t_0 the origin of the cycle, and we assign to every state t_i the label q if the number i defines an assignment that satisfies the clause c. The Kripke structure K_ψ is the disjoint union of the cycles corresponding to each clause, and an initial state t_I with transitions from t_I to the origin of each cycle. Note that the Kripke structure K_ψ can be constructed in polynomial time, as the sum of the first n prime numbers is bounded by a polynomial in n: $\sum_{i=1}^{n} p_i \in O(n^2 \log n)$ [2].

It follows that a number z defines an assignment that satisfies the formula ψ (i.e., satisfies all clauses of ψ) if and only if every path of length $z + 1$ from t_I reaches a state labelled by q. Therefore the formula ψ is satisfiable if and only if $K_\psi, t_I \models F\forall q$, and it follows that the model-checking problem is NP-hard for the formulas $F\forall q$ and for $p\mathcal{U}\forall q$ (let p hold in every state of K_ψ).

NP-hardness of $p\mathcal{U}\exists q$. The proof is by a reduction from 3SAT [13]. The reduction is illustrated in Figure 2. Given a Boolean propositional formula ψ in CNF, with set C of (disjunctive) clauses over variables $x_1, \ldots, x_n$ (where each clause contains three variables), we construct a Kripke structure K as follows: let $m = |C|$ be the number of clauses in ψ, and construct m disjoint simple paths π_i from t_I of length $m + 1$ (of the form $t_I, t_1, \ldots, t_m$), where the last state of each path π_i has a transition to the origin of a cycle corresponding to the ith clause (the cycles and their labeling are as defined in the NP-hardness proof of $F\forall q$). The state t_I and all states of the cycles are also labelled by p, and in the ith path from t_I, the ith state after t_I is labelled by p. The construction can be obtained in polynomial time.

We show that ψ is satisfiable if and only if the formula $p\mathcal{U}\exists q$ holds from t_I in K. Recall that $p\mathcal{U}\exists q$ holds if there exists $k \geq 0$ such that for all $0 \leq j < k$, there exists a path $t_0 t_1 \ldots$ in K with $t_0 = t_I$ and $K, t_j \models p$ and $K, t_k \models q$.

For the first direction of the proof, if ψ is satisfiable, then let $z \in \mathbb{N}$ define a satisfying assignment, and let $k = m + 2 + z$. Then all paths of length k from t_I in K end up in a state labelled by q. Now we consider an arbitrary $j < k$ and show that there exists a path of length k from t_I that ends up in a state labelled by q, and with the jth state labelled by p. For $j = 0$ and for $j > m$, the conditions are satisfied by all paths, and for $j \leq m$, the conditions are satisfied by the jth path from t_I.

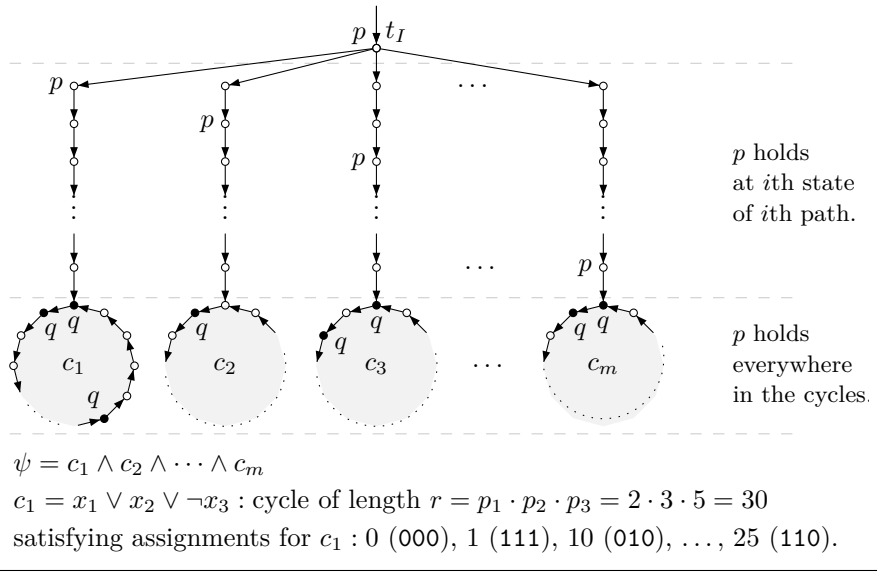


Figure 2 Reduction to show NP-hardness of $p\mathcal{U}\exists\, q$ in Lemma 2.

For the second direction of the proof, let k be a position such that for all $0 \leq j < k$, there exists a path $t_0 t_1 \dots$ in K with $t_0 = t_I$ and $K, t_j \models p$ and $K, t_k \models q$. Then $k \geq m + 2$ since only the states in the cycles are labelled by q. Consider the set P containing, for each $j = 1, 2, \dots, m$, a path $t_I t_1 \dots$ in K with $K, t_j \models p$ and $K, t_k \models q$. It is easy to see by the construction of K that P contains all the paths of length k in K. Therefore, all paths of length $z = k - (m+2)$ from the origin of each cycle end up in a state labelled by q. It follows that z defines an assignment that satisfies all clauses in ψ, thus ψ is satisfiable.

DP-hardness of $p\mathcal{U}\exists\, q$. The DP-hardness proof of $p\mathcal{U}\exists\, q$ uses a reduction of the same flavor as in the NP-hardness of $F\forall q$ [6].

coNP-hardness of $G\exists q$. The result follows from the NP-hardness of $F\forall q$ since $G\exists q$ is equivalent to $\neg F\forall \neg q$. ◀

The complexity result of Theorem 1 is not tight, with a $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}}$ upper bound and hardness for $\mathrm{P}_{\|}^{\mathrm{NP}}$. Even for the fixed formula $p\mathcal{U}\exists\, q$, the gap between the $\mathrm{NP}^{\mathrm{NP}}$ upper bound and the DP-hardness result provides an interesting open question for future work.

3 Extension of CTL+Sync with Always and Eventually

We consider an extension of CTL+Sync with formulas of the form $\mathcal{T}Q\varphi$ where $\mathcal{T} \in \{F, G\}^+$ is a sequence of unary temporal operators Eventually (F) and Always (G). For example, the formula $FG\forall p$ expresses strong synchronization, namely that from some point on, all positions on every path satisfy p; the formula $GF\forall p$ expresses weak synchronization, namely that there are infinitely many positions such that, on every path at those positions p holds. In fact only the combination of operators FG and GF need to be considered, as the other combinations of operators reduce to either FG or GF using the LTL identities $FGF\varphi \equiv GF\varphi$ and $GFG\varphi \equiv FG\varphi$. Formally, define:

- $K, t \models GF\forall\varphi_1$ if for all $k \geq 0$, there exists $j \geq k$ such that for all paths $t_0 t_1 \dots$ in K with $t_0 = t$, we have $K, t_j \models \varphi_1$.

- $K, t \models GF\exists\varphi_1$ if for all $k \geq 0$, there exists $j \geq k$ and there exists a path $t_0 t_1 \dots$ in K with $t_0 = t$ such that $K, t_j \models \varphi_1$.
- $K, t \models FG\forall\varphi_1$ if $K, t \not\models GF\exists\neg\varphi_1$.
- $K, t \models FG\exists\varphi_1$ if $K, t \not\models GF\forall\neg\varphi_1$.

The model-checking problem for the formula $GF\forall\varphi_1$ is NP-complete: guess positions $n, k \leq 2^{|T|}$ (represented in binary) and check in polynomial time that the states reachable by all paths of length n satisfy φ_1, and that set of the states reachable after $n + k$ steps is the same as the set of states reachable after n steps, where $k > 0$. This corresponds to finding a lasso in the subset construction for the Kripke structure K. A matching NP lower bound follows from the reduction in the NP-hardness proof of $F\forall q$ (Lemma 2).

The model-checking problem for the formula $GF\exists\varphi_1$ can be solved in polynomial time, as this formula is equivalent to saying that there exists a state labeled by φ_1 that is reachable from a reachable non-trivial strongly connected component (SCC) — an SCC is trivial if it consists of a single state without self-loop. To prove this, note that if a state t^* labeled by φ_1 is reachable from a reachable non-trivial SCC, then t^* can be reached by an arbitrarily long path, thus the formula $GF\exists\varphi_1$ holds. For the other direction, if no state labeled by φ_1 is reachable from a reachable non-trivial SCC, then every path to a state labeled by φ_1 is acyclic (otherwise, the path would contain a cycle, belonging to an SCC). Since acyclic paths have length at most $|T|$, it follows that the formula $GF\exists\varphi_1$ does not hold, which concludes the proof.

From the above arguments, it follows that the complexity status of the model-checking problem for this extension of CTL+Sync is the same as the complexity of CTL+Sync model-checking in Theorem 1.

▶ **Theorem 3.** *The model-checking problem for CTL+Sync extended with sequences of unary temporal operators lies in $P^{NP^{NP}}$ and is $P_{\|}^{NP}$-hard.*

4 Expressive Power

The expressive power of CTL+Sync (even extended with Always and Eventually) is incomparable with the expressive power of MSO. By the remark at the end of Section 2.1, CTL+Sync can express non-regular properties, and thus is not subsumed by MSO, and standard argument based on counting properties [27] showing that CTL is less expressive than MSO apply straightforwardly to show that formulas of MSO are not expressible in CTL+Sync [10].

We show that the formulas $GF\forall p$ and $FG\forall p$ for weak and strong synchronization cannot be expressed in the logic CTL+Sync, thus CTL+Sync extended with sequences of unary temporal operators is strictly more expressive than CTL+Sync. The result holds if the Next operator is not allowed, and also if the Next operator is allowed.

▶ **Theorem 4.** *The logic CTL+Sync (even without the Next operator) extended with sequences of unary temporal operators is strictly more expressive than CTL+Sync (even using the Next operator).*

Proof. We show that the formula $GF\forall p$ cannot be expressed in CTL+Sync, even using the Next operator. To prove this, given an arbitrary CTL+Sync formula φ, we construct two Kripke structures such that φ holds in both Kripke structures, but the formula $GF\forall p$ holds in one and not in the other. It follows that φ is not equivalent to $GF\forall p$.

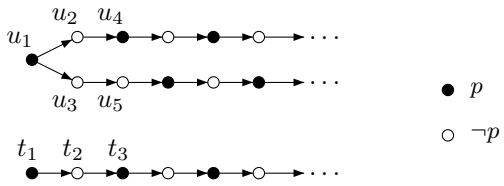


Figure 3 States t_1 and u_1 are indistinguishable by formulas of CTL+Sync.

Given the formula φ, we construct the two Kripke structures as follows. Consider two Kripke structures whose unravelling is shown in Figure 3 where the states reachable from t_1 are satisfying alternately p and $\neg p$, and the states reachable from u_2 and u_5 are satisfying alternately $\neg p$ and p. Call black states the states where p holds, and white states the states where $\neg p$ holds. If n is the maximum number of nested Next operators in φ, then we construct the n-stuttering of the two Kripke structures in Figure 3, where the n-stuttering of a Kripke structure $K = \langle T, \Pi, \pi, R\rangle$ is the Kripke structure $K^n = \langle T \times \{1, \ldots, n\}, \Pi, \pi^n, R^n\rangle$ where $\pi^n(t, i) = \pi^n(t)$ for all $1 \leq i \leq n$, and the transition relation R^n contains all pairs $((t, i), (t, i + 1))$ for all $t \in T$ and $1 \leq i < n$, and all pairs $((t, n), (t', 1))$ for all $(t, t') \in R$.

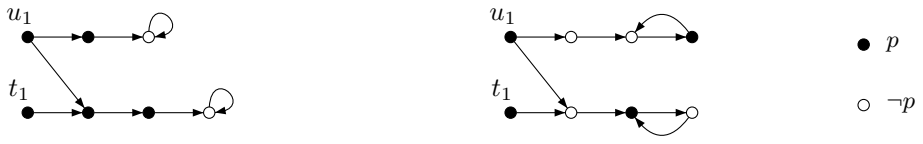
We claim that the formula φ holds either in both $(t_1, 1)$ and $(u_1, 1)$, or in none of $(t_1, 1)$ and $(u_1, 1)$, while the formula $GF\forall p$ holds in $(t_1, 1)$ and not in $(u_1, 1)$. We show by induction on the nesting depth of CTL+Sync formulas φ (that have at most n nested Next operators) that (t_1, i) and (u_1, i) are equivalent for φ (for all $1 \leq i \leq n$), and that for all black states t, u, the copies $(t, 1)$ and $(u, 1)$ are equivalent for φ, and analogously for all pairs of white states.

The result holds trivially for formulas of nesting depth 0, that is atomic propositions. For the induction step, assume the claim holds for formulas of nesting depth k, and consider a formula φ of nesting depth $k + 1$. If the outermost operator of φ is a Boolean operator, or a CTL operator ($Q\mathcal{X}$ or $Q\mathcal{U}$), then the result follows from the induction hypothesis and the result of [18, Theorem 2] showing two paths that differ only in the number of consecutive repetitions of a state, as long as the number of repetitions is at least n, are equivalent for the formulas with at most n nested Next operators. If the outermost operator of φ is either $\mathcal{U}\exists$ or $\mathcal{U}\forall$, that is $\varphi \equiv \varphi_1 \,\mathcal{U}\exists\, \varphi_2$ or $\varphi \equiv \varphi_1 \,\mathcal{U}\forall\, \varphi_2$, then consider a state where φ holds: either φ_2 holds in that state, and by the induction hypothesis, φ_2 also holds in the corresponding state (that we claimed to be equivalent), or φ_2 holds in the states of the other color than the current state, and φ_1 holds on the path(s) at all positions before. By the induction hypothesis, at the same distance from the claimed equivalent states, we can find a state where φ_2 holds in all paths, and φ_1 holds on all positions before, which concludes the proof for the induction step. ◀

5 Distinguishing Power

Two states of a Kripke structure can be distinguished by a logic if there exists a formula in the logic that holds in one state but not in the other. Each logic induces an indistinguishability relation (which is an equivalence) on Kripke structures that characterizes the distinguishing power of the logic. Two states t and t' of a Kripke structure K are indistinguishable by a logic $\mathcal{L}$ if they satisfy the same formulas of $\mathcal{L}$, that is $\{\varphi \in \mathcal{L} \mid K, t \models \varphi\} = \{\varphi \in \mathcal{L} \mid K, t' \models \varphi\}$.

For CTL (with the Next operator), the distinguishing power is standard bisimulation, and



(a) The states t_1 and u_1 are stuttering bisimilar (they satisfy the same CTL formulas without the Next operator), but they can be distinguished by the CTL+Sync formula $p\mathcal{U}\forall\neg p$ which holds in t_1 but not in u_1.

(b) The states t_1 and u_1 are indistinguishable by CTL+Sync formulas, but they are not bisimilar, i.e. they can be distinguished by CTL formulas with the Next operator, for example $\forall\mathcal{X}\,\forall\mathcal{X}\,p$ which holds in t_1 but not in u_1.

Figure 4 The distinguishing power of CTL+Sync lies strictly between bisimulation and stuttering bisimulation.

for CTL without the Next operator, the distinguishing power is stuttering bisimulation [4]. Stuttering bisimulation is a variant of bisimulation where intuitively several transitions can be used to simulate a single transition, as long as the intermediate states of the transitions are all equivalent (for stuttering bisimulation). We omit the definition of bisimulation and stuttering bisimulation [4], and in this paper we consider that they are defined as the distinguishing power of respectively CTL and CTL without the Next operator.

It is easy to show by induction on the nesting depth of formulas that the distinguishing power of CTL+Sync is the same as for CTL, since (*i*) CTL+Sync contains CTL, and (*ii*) if two states t and t' are bisimilar, there is a correspondence between the paths starting from t and the paths starting from t' (for every path from t, there is a path from t' such that their states at position i are bisimilar, for all $i \in \mathbb{N}$, and analogously for every path of t' [4, Lemma 3.1]), which implies the satisfaction of the same formulas in CTL+Sync. The same argument holds for CTL+Sync extended with unary temporal operators (Section 3).

▶ **Theorem 5.** *Two states t and t' of a Kripke structure K are indistinguishable by CTL+Sync formulas (even extended with unary temporal operators) if and only if t and t' are bisimilar.*

Without the Next operator, the logic CTL+Sync has a distinguishing power that lies strictly between bisimulation and stuttering bisimulation, as shown by the examples in Figure 4a and Figure 4b. Indistinguishability by CTL+Sync formulas without the Next operator implies indistinguishability by standard CTL without the Next operator, and thus stuttering bisimilarity. We obtain the following result.

▶ **Theorem 6.** *The following implications hold for all states t, t' of a Kripke structure K:*

- *if t and t' are bisimilar, then t and t' are indistinguishable by CTL+Sync formulas without the Next operator (even extended with unary temporal operators);*
- *if t and t' are indistinguishable by CTL+Sync formulas without the Next operator (even extended with unary temporal operators), then t and t' are stuttering bisimilar.*

It follows from the first part of Theorem 6 that the state-space reduction techniques based on computing a bisimulation quotient before evaluating a CTL formula will work for CTL+Sync. Although the exact indistinguishability relation for CTL+Sync is coarser than bisimulation, we show that deciding this relation is NP-hard, and thus it may not be relevant to compute it for quotienting before model-checking, but rather use the polynomial-time computable bisimulation.

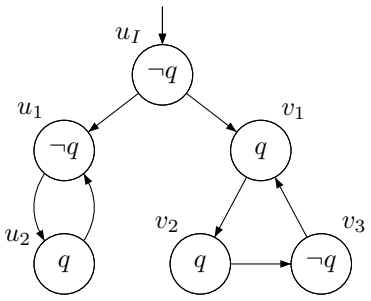


Figure 5 The Kripke structure K in the proof of Theorem 7.

▶ **Theorem 7.** *Deciding whether two states of a Kripke structure are indistinguishable by CTL+Sync formulas without the Next operator is NP-hard.*

Proof. The proof is by a reduction from the Boolean satisfiability problem 3SAT which is NP-complete [13]. Given a Boolean propositional formula ψ in CNF, we construct two Kripke structures K and K_ψ that are indistinguishable (from their initial state) if and only if ψ is satisfiable, where:

- K is the Kripke structure shown in Figure 5, and
- K_ψ is the Kripke structure constructed in the NP-hardness proof of $F\forall q$ (Lemma 2).

We know from the proof of Lemma 2 that $K_\psi, t_I \models F\forall q$ if and only if ψ is satisfiable. Hence, it suffices to show that K and K_ψ are indistinguishable if and only if the formula $F\forall q$ holds in t_I. Since the formula $F\forall q$ holds in u_I, we only need to show that if $F\forall q$ holds in t_I, then K and K_ψ are indistinguishable. To do this, we assume that $F\forall q$ holds in t_I, and we show that for all CTL+Sync formulas φ without the Next operator, $t_I \models \varphi$ if and only if $u_I \models \varphi$. The proof proceeds by induction on the nesting depth of φ and simple combinatorial arguments [6]. ◀

6 CTL* + Synchronization

CTL* is a branching-time extension of LTL (and of CTL) where several nested temporal operators and Boolean connectives can be used under the scope of a single path quantifier. For example the CTL* formula $\exists(G\varphi \to G\psi)$ says that there exists a path in which either φ does not hold in every position, or ψ holds at every position. Note that φ and ψ may also contain path quantifiers.

Extending CTL+Sync with formula quantification analogous to CTL* presents some difficulties. Even considering only Boolean connectives and $\{F, G\}$ operators leads to a logic that is hard to define. For example, one may consider a formula like $(Fp \vee Fq)\forall$ which could be naturally interpreted as there exist two positions $m, n \geq 0$ such that on all paths ρ, either p holds at position m in ρ, or q holds at position n in ρ. In this definition the $\vee$ operator would not be idempotent, that is $\psi_1 = (Fp \vee Fp)\forall$ is not equivalent to $\psi_2 = (Fp)\forall$, where ψ_1 means that the set of all paths can be partitioned into two sets of paths where p holds synchronously at some position, but not necessarily the same position in both sets, while ψ_2 expresses the property that p holds synchronously at some position in all paths.

Another difficulty with binary operators is the semantics induced by the order of the operands. For instance, the formula $(Fp \vee Gq)\forall$ can be interpreted as (i) there exists a position $m \geq 0$ such that for all positions $n \geq 0$, on all paths ρ, either $\rho + m \models p$ or $\rho + n \models q$;

or it can be interpreted as (ii) for all $n \geq 0$, there exists $m \geq 0$ such that on all paths ρ, either $\rho + m \models p$ or $\rho + n \models q$. These two interpretations differ on the Kripke structure that produces exactly two paths ρ_1 and ρ_2 such that p and q hold at the following positions (p holds nowhere except at position 1 in ρ_1 and position 3 in ρ_2, and q holds everywhere except position 2 in ρ_1 and position 4 in ρ_2):

in ρ_1:	$\{\bar{p}, q\}$	$\{p, q\}$	$\{\bar{p}, \bar{q}\}$	$\{\bar{p}, q\}$	$\{\bar{p}, q\}$	$\{\bar{p}, q\}$	$\ldots$
in ρ_2:	$\{\bar{p}, q\}$	$\{\bar{p}, q\}$	$\{\bar{p}, q\}$	$\{p, q\}$	$\{\bar{p}, \bar{q}\}$	$\{\bar{p}, q\}$	$\ldots$
	0	1	2	3	4	5	

Note that the two paths agree on their initial position, and we can construct a Kripke structure that produces exactly those two paths. It is easy to see that the formula $(Fp \vee Gq)\forall$ does not hold according to the first interpretation (indeed, for $m = 1$ we can take $n = 4$ and consider the path ρ_2 where p does not hold at position 1 and q does not hold at position 4, and for all other values of m, take $n = 2$ and consider the path ρ_1 where p does not hold at position m and q does not hold at position 2), but it does hold according to the second interpretation (for $n = 2$ take $m = 1$, for $n = 4$ take $m = 3$, and for all other values of n take arbitrary value of m, for example $m = n$). The trouble is that the order of the existential quantifier (associated to the left operand Fp) and the universal quantifier (associated to the right operand Gq) actually matters in the semantics of the formula, leading to an annoying situation that $(Fp \vee Gq)\forall$ is not equivalent to $(Gq \vee Fp)\forall$ in any of the interpretations. One way could be to use the branching Henkin quantifiers, like $\binom{\exists m}{\forall n}$ where the existential choice of m does not depend on the universal choice of m. This interpretation suffers from lack of symmetry, as the negation of such a branching Henkin quantifier is in general not expressible as a branching Henkin quantifier [3].

7 Conclusion

The logic CTL+Sync and its extensions presented in this paper provide an elegant framework to express non-regular properties of synchronization. It is intriguing that the exact optimal complexity of the model-checking problem remains open, specially even for the fixed formula $p\mathcal{U}\exists\, q$ (which we show is in $\mathrm{NP}^{\mathrm{NP}}$, and DP-hard). Extending CTL+Sync to an elegant logic *à la* CTL* seems challenging. One may want to express natural properties with the flavor of synchronization, such as the existence of a fixed number of synchronization points, or the property that all paths synchronize in either of a finite set of positions, etc. (see also Section 6). Another direction is to consider alternating-time temporal logics (ATL [1]) with synchronization. ATL is a game-based extension of CTL for which the model-checking problem remains in polynomial time. For instance, ATL can express the existence of a winning strategy in a two-player reachability game. For the synchronized version of reachability games (where the objective for a player is to reach a target state after a number of steps that can be fixed by this player, independently of the strategy of the other player), it is known that deciding the winner is PSPACE-complete [15]. Studying general game-based logics such as ATL or strategy logic [7] combined with quantifier exchange is an interesting direction for future work.

Acknowledgment. We thank Stefan Göller and anonymous reviewers for their insightful comments and suggestions.

References

1 R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49:672–713, 2002.

2 E. Bach and J. Shallit. *Algorithmic Number Theory, Vol. 1: Efficient Algorithms*. MIT Press, 1996.

3 A. Blass and Y. Gurevich. Henkin quantifiers and complete problems. *Ann. Pure Appl. Logic*, 32:1–16, 1986.

4 M. C. Browne, E. M. Clarke, and O. Grumberg. Characterizing finite Kripke structures in propositional temporal logic. *Theor. Comput. Sci.*, 59:115–131, 1988.

5 J. Černý. Poznámka k. homogénnym experimentom s konečnými automatmi. In *Matematicko-fyzikálny Časopis*, volume 14(3), pages 208–216, 1964.

6 K. Chatterjee and L. Doyen. Computation tree logic for synchronization properties. *CoRR*, arXiv:1604.06384, 2016.

7 K. Chatterjee, T. A. Henzinger, and N. Piterman. Strategy logic. *Inf. Comput.*, 208(6):677–693, 2010.

8 D. Chistikov, P. Martyugin, and M. Shirmohammadi. Synchronizing automata over nested words. In *Proc. of FOSSACS: Foundations of Software Science and Computation Structures*, LNCS 9634, pages 252–268. Springer, 2016.

9 A. Cimatti, E. M. Clarke, F. Giunchiglia, and M. Roveri. NUSMV: A new symbolic model checker. *STTT*, 2(4):410–425, 2000.

10 E. M. Clarke, O. Grumberg, and D. Peled. *Model checking*. MIT Press, 2001.

11 M. R. Clarkson, B. Finkbeiner, M. Koleini, K. K. Micinski, M. N. Rabe, and C. Sánchez. Temporal logics for hyperproperties. In *Proceedings of POST: Principles of Security and Trust*, LNCS 8414, pages 265–284. Springer, 2014.

12 M. R. Clarkson and F. B. Schneider. Hyperproperties. *Journal of Computer Security*, 18(6):1157–1210, 2010.

13 S. A. Cook. The complexity of theorem proving procedures. In *Proc. of STOC: Symposium on the Theory of Computing*, pages 151–158. ACM Press, 1971.

14 L. Doyen, L. Juhl, K. G. Larsen, N. Markey, and M. Shirmohammadi. Synchronizing words for weighted and timed automata. In *Proc. of FSTTCS: Foundations of Software Technology and Theoretical Computer Science*, LIPIcs, pages 121–132. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014.

15 L. Doyen, T. Massart, and M. Shirmohammadi. Limit synchronization in Markov decision processes. In *Proc. of FoSSaCS: Foundations of Software Science and Computation Structures*, LNCS 8412, pages 58–72. Springer-Verlag, 2014.

16 L. Doyen, T. Massart, and M. Shirmohammadi. Robust synchronization in Markov decision processes. In *Proc. of CONCUR: Concurrency Theory*, volume LNCS 8704, pages 234–248. Springer, 2014.

17 J. Kretínský, K. G. Larsen, S. Laursen, and J. Srba. Polynomial time decidability of weighted synchronization under partial observability. In *Proc. of CONCUR: Concurrency Theory*, volume 42 of *LIPIcs*, pages 142–154. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

18 A. Kučera and Jan Strejček. The stuttering principle revisited. *Acta Inf.*, 41(7-8):415–434, 2005.

19 K. G. Larsen, S. Laursen, and J. Srba. Synchronizing strategies under partial observability. In *Proc. of CONCUR: Concurrency Theory*, LNCS 8704, pages 188–202. Springer, 2014.

20 G. Lenzi. Recent results on modal mu-calculus: a survey. *Rend. Istit. Mat. Univ. Trieste*, 42(2):235–255, 2010.

21 H. Spakowski. *Completeness for Parallel Access to NP and Counting Class Separations*. PhD thesis, Heinrich-Heine-Universität Düsseldorf, 2005.

22 L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time: Preliminary report. In *Proc. of STOC: Symposium on Theory of Computing*, pages 1–9. ACM, 1973.

23 W. Thomas. Automata on infinite objects. In *Handbook of Theoretical Computer Science, Vol. B: Formal Models and Sematics*, pages 133–192. MIT Press, 1990.

24 W. Thomas. Languages, automata, and logic. In *Handbook of Formal Languages, Vol. 3: Beyond Words*, pages 389–455. Springer, 1997.

25 M. V. Volkov. Synchronizing automata and the Černý conjecture. In *Proc. of LATA: Language and Automata Theory and Applications*, LNCS 5196, pages 11–27. Springer, 2008.

26 K. W. Wagner. More complicated questions about maxima and minima, and some closures of NP. *Theor. Comput. Sci.*, 51:53–80, 1987.

27 P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1/2):72–99, 1983.