Model Checking CTL is Almost Always Inherently Sequential[*]

Olaf Beyersdorff   Arne Meier   Michael Thomas   Heribert Vollmer
Theoretical Computer Science, University of Hannover, Germany
{*beyersdorff, meier, thomas, vollmer*}*@thi.uni-hannover.de*

Martin Mundhenk
Computer Science, University of Jena, Germany
mundhenk@cs.uni-jena.de

Thomas Schneider
Computer Science, University of Manchester, UK
schneider@cs.man.ac.uk

Abstract

The model checking problem for CTL *is known to be* P-*complete (Clarke, Emerson, and Sistla (1986), see Schnoebelen (2002)). We consider fragments of* CTL *obtained by restricting the use of temporal modalities or the use of negations—restrictions already studied for* LTL *by Sistla and Clarke (1985) and Markey (2004). For all these fragments, except for the trivial case without any temporal operator, we systematically prove model checking to be either inherently sequential (*P-*complete) or very efficiently parallelizable (*LOGCFL-*complete). For most fragments, however, model checking for* CTL *is already* P-*complete. Hence our results indicate that in most applications, approaching* CTL *model checking by parallelism will not result in the desired speed up.*

We also completely determine the complexity of the model checking problem for all fragments of the extensions ECTL, CTL$^+$, *and* ECTL$^+$.

1. Introduction

Temporal logic was introduced by Pnueli [12] as a formalism to specify and verify properties of concurrent programs. Computation Tree Logic (CTL), the logic of branching time, goes back to Emerson and Clarke [4] and contains temporal operators for expressing that an event occurs at some time in the future (F), always in the future (G), in the next point of time (X), always in the future until another event holds (U), or as long as it is not released by the occurrence of another event (R), as well as path quantifiers (E, A) for speaking about computation paths. The full language obtained by these operators and quantifiers is called CTL* [5]. In CTL, the interaction between the temporal operators and path quantifiers is restricted. The temporal operators in CTL are obtained by path quantifiers followed directly by any temporal operator, e.g., AF and AU are CTL-operators. Because they start with the universal path quantifier, they are called *universal* CTL-*operators*. Accordingly, EX and EG are examples for *existential* CTL-*operators*.

Since properties are largely verified automatically, the computational complexity of reasoning tasks is of great interest. Model checking (MC)—the problem of verifying whether a given formula holds in a state of a given model—is one of the most important reasoning tasks [15]. It is intractable for CTL* (PSPACE-complete [6, 15]), but tractable for CTL (solvable in, and even hard for, polynomial time [3, 15]).

Although model checking for CTL is tractable, its P-hardness means that it is presumably not efficiently parallelizable. We therefore search for fragments of CTL with a model checking problem of lower complexity. We will consider all subsets of CTL-operators, and examine the complexity of the model checking problems for all resulting fragments of CTL. Further, we consider three additional restrictions affecting the use of negation and study the extensions ECTL, CTL$^+$, and their combination ECTL$^+$.

The complexity of model checking for fragments of temporal logics has been examined in the literature: Markey [9] considered satisfiability and model checking for fragments of Linear Temporal Logic (LTL). Under systematic restrictions to the temporal operators, the use of negation, and the interaction of future and past operators, Markey classified the two

decision problems into NP-complete, coNP-complete, and PSPACE-complete. Further, [1] examined model checking for all fragments of LTL obtained by restricting the set of temporal and propositional operators. The resulting classification separated cases where model checking is tractable from those where it is intractable.

Concerning CTL and its extension ECTL, our results in this paper show that most restricted versions of the model checking problem exhibit the same hardness as the general problem. More precisely, we show that apart from the trivial case where CTL-operators are completely absent, the complexity of CTL model checking is a dichotomy: it is either P-complete or LOGCFL-complete. Unfortunately, the latter case only occurs for a few rather weak fragments and hence there is not much hope that in practice, model checking can be sped up by using parallelism—it is inherently sequential.

Put as a simple rule, model checking for CTL is P-complete for every fragment that allows to express a universal *and* an existential CTL-operator. Only for fragments involving the operators EX and EF (or alternatively AX and AG) model checking is LOGCFL-complete. This is visualized in Fig. 2 in Sect. 5. Recall that LOGCFL is defined as the class of problems logspace-reducible to context-free languages, and $NL \subseteq LOGCFL \subseteq NC^2 \subseteq P$. Hence, in contrast to inherently sequential P-hard tasks, problems in LOGCFL have very efficient parallel algorithms.

For the extensions CTL^+ and $ECTL^+$, the situation is more complex. In general, model checking CTL^+ and $ECTL^+$ is Δ_2^p-complete [8]. We show that for $T \subseteq \{A, E, X\}$, both model checking problems remain tractable, while for $T \not\subseteq \{A, E, X\}$, both problems become Δ_2^p-complete. Yet, for negation restricted fragments with only existential or only universal path quantifiers, we observe a complexity decrease to NP- resp. coNP-completeness.

This paper is organized as follows: Section 2 introduces CTL, its model checking problems, and the non-basics of complexity theory we use. Section 3 contains our main results, separated into upper and lower bounds. We also provide a refined analysis of the reductions between different model checking problems with restricted use of negation. The results are then generalized to extensions of CTL in Section 4. Finally, Section 5 concludes with a graphical overview of the results. For brevity, some proofs are omitted and will be included in the full version of this paper.

2. Preliminaries

2.1. Temporal Logic

We inductively define $CTL^\star$-formulae as follows. Let Φ be a finite set of atomic propositions. The symbols used are the atomic propositions in Φ, the constant symbols $\top$ and

$\bot$, the Boolean connectives $\neg$, $\wedge$, and $\vee$, and the temporal operator symbols A, E, X, F, G, U, and R.

A and E are called a *path quantifiers*, temporal operators aside from A and E are *pure temporal operators*. The atomic propositions and the constants $\top$ and $\bot$ are *atomic formulae*. There are two kinds of formulae, *state formulae* and *path formulae*. Each atomic formula is a state formula, and each state formula is a path formula. If φ, ψ are state formulae and χ, π are path formulae, then $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $A\chi$, $E\chi$ are state formulae, and $\neg\chi$, $(\chi \wedge \pi)$, $(\chi \vee \pi)$, $X\chi$, $F\chi$, $G\chi$, $[\chi U\pi]$, and $[\chi R\pi]$ are path formulae. The set of $CTL^\star$-formulae (or *formulae*) consists of all state formulae.

A *Kripke structure* is a triple $K = (W, R, \eta)$, where W is a finite set of states, $R \subseteq W \times W$ a total relation (i. e., for each $w \in W$, there exists a w' such that $(w, w') \in R$), and $\eta \colon W \to \mathfrak{P}(\Phi)$ is a labelling function. A *path* x is an infinite sequence $x = (x_1, x_2, \ldots) \in W^\omega$ such that $(x_i, x_{i+1}) \in R$, for all $i \geq 1$. For a path $x = (x_1, x_2, \ldots)$ we denote by x^i the path $(x_i, x_{i+1}, \ldots)$.

Let $K = (W, R, \eta)$ be a Kripke structure, $w \in W$ be a state, and $x = (x_1, x_2, \ldots) \in W^\omega$ be a path. Further, let φ, ψ be state formulae and χ, π be path formulae. The truth of a $CTL^\star$-formula w. r. t. K is inductively defined as:

$$K, w \models \top \qquad \text{always,}$$
$$K, w \models \bot \qquad \text{never,}$$
$$K, w \models p \qquad \text{iff } p \in \Phi \text{ and } p \in \eta(w),$$
$$K, w \models \neg\varphi \qquad \text{iff } K, w \not\models \varphi,$$
$$K, w \models (\varphi \wedge \psi) \text{ iff } K, w \models \varphi \text{ and } K, w \models \psi,$$
$$K, w \models (\varphi \vee \psi) \text{ iff } K, w \models \varphi \text{ or } K, w \models \psi,$$
$$K, w \models A\chi \qquad \text{iff } K, x \models \chi \text{ for all paths}$$
$$\qquad\qquad x = (x_1, x_2, \ldots) \text{ with } x_1 = w,$$
$$K, x \models \varphi \qquad \text{iff } K, x_1 \models \varphi,$$
$$K, x \models \neg\chi \qquad \text{iff } K, x \not\models \chi,$$
$$K, x \models (\chi \wedge \pi) \text{ iff } K, x \models \chi \text{ and } K, x \models \pi,$$
$$K, x \models (\chi \vee \pi) \text{ iff } K, x \models \chi \text{ or } K, x \models \pi,$$
$$K, x \models X\chi \qquad \text{iff } K, x^2 \models \chi$$
$$K, x \models [\chi U\pi] \quad \text{iff there exists } k \in \mathbb{N} \text{ such that}$$
$$\qquad\qquad K, x^i \models \chi \text{ for } 1 \leq i < k \text{ and}$$
$$\qquad\qquad K, x^k \models \pi.$$

The semantics of the remaining temporal operators is defined via the equivalences: $E\chi \equiv \neg A\neg\chi$, $F\chi \equiv [\top U\chi]$, $G\chi \equiv \neg F\neg\chi$, and $[\chi R\pi] \equiv \neg[\neg\chi U\neg\pi]$. A state formula φ is *satisfied by a Kripke structure* K if there exists $w \in W$ such that $K, w \models \varphi$. We will also denoted this by $K \models \varphi$.

A CTL-*formula* is a $CTL^\star$-formula in which each path quantifier is followed by exactly one pure temporal operator and each pure temporal operator is preceded by exactly one path quantifier. The set of CTL-formulae forms a strict subset of the set of all $CTL^\star$-formulae. For example, AGEFp is a CTL-formula, but $A(GFp \wedge Fq)$ is not. Pairs of path quantifiers and pure temporal operators are called CTL-*operators*. The operators AX, AF, AG, AU, and AR are *universal* CTL-
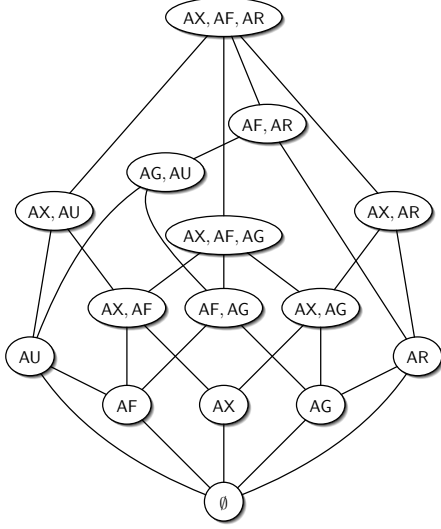
Figure 1. The expressive power of CTL(T).

operators, and EX, EF, EG, EU, and ER are *existential* CTL-operators. Let ALL denote the set of all CTL-operators. Note that $\mathsf{A}[\psi \mathsf{U} \chi] \equiv \mathsf{AF}\chi \wedge \neg\mathsf{E}[\neg\chi\mathsf{U}(\neg\psi \wedge \neg\chi)]$, and thus $\mathsf{E}[\psi\mathsf{R}\chi] \equiv \mathsf{EG}\chi \vee \mathsf{E}[\chi\mathsf{U}(\psi \wedge \chi)]$. Hence $\{\mathsf{AX}, \mathsf{AF}, \mathsf{AR}\}$ is a minimal set of CTL-operators for CTL (in presence of all Boolean connectives), whereas $\{\mathsf{AX}, \mathsf{AG}, \mathsf{AU}\}$ is not complete for CTL [7].

By CTL(T) we denote the set of CTL-formulae using the connectives $\{\wedge, \vee, \neg\}$ and the CTL-operators in T only. Figure 1 shows the structure of sets of CTL-operators with respect to their expressive power. Moreover, we define the following fragments of CTL(T):

- CTL$_{\text{pos}}(T)$ (positive): CTL-operators may not occur in the scope of a negation,
- CTL$_{\text{a.n.}}(T)$ (atomic negation): negation signs appear only directly in front of atomic propositions,
- CTL$_{\text{mon}}(T)$ (monotone): no negation signs allowed.

This restricted use of negation was introduced and studied in the context of linear temporal logic, LTL, by Sistla and Clarke [17] and Markey [9]. Their original notation was $\widetilde{\mathrm{L}}(T)$ for CTL$_{\text{a.n.}}(T)$ and $\mathrm{L}^+(T)$ for CTL$_{\text{pos}}(T)$.

2.2. Model Checking

Now we define the *model checking problems* for the above mentioned fragments of CTL. Let $\mathcal{L}$ be CTL, CTL$_{\text{mon}}$, CTL$_{\text{a.n.}}$, or CTL$_{\text{pos}}$.

Problem: $\mathcal{L}$-MC(T)
Input: A Kripke structure $K = (W, R, \eta)$, a state $w \in W$, and an $\mathcal{L}(T)$-formula φ.
Question: Does $K, w \models \varphi$ hold?

2.3. Complexity Theory

We assume familiarity with standard notions of complexity theory (cf. [11]). Next we will introduce the notions from circuit complexity that we use for our results. All reductions in this paper are $\leq_{\text{cd}}$-reductions defined as follows: A language A is *constant-depth reducible* to B, $A \leq_{\text{cd}} B$, if there is a logtime-uniform AC^0-circuit family with oracle gates for B that decides membership in A. That is, there is a circuit family $\mathcal{C} = (C_1, C_2, C_3, \dots)$ such that

- for every n, C_n computes the characteristic function of A for inputs of length n,
- there is a polynomial p and a constant d such that for all input lengths n, the size of C_n is bounded by $p(n)$ and the depth of C_n is bounded by d,
- each circuit C_n consists of unbounded fan-in AND and OR gates, negation gates, and gates that compute the characteristic function of B (the *oracle gates*),
- there is a linear-time Turing machine M that can check the structure of the circuit family, i.e., given a tuple $\langle n, g, t, h \rangle$ where n, g, h are binary numbers and $t \in \{\text{AND}, \text{OR}, \text{NOT}, \text{ORACLE}\}$, M accepts if C_n contains a gate g of type t with predecessor h.

Circuit families $\mathcal{C}$ with this last property are called *logtime-uniform* (the name stems from the fact that the time needed by M is linear in the length of its input tuple, hence logarithmic in n). For background information we refer to [13, 18].

We easily obtain the following relations between model checking for fragments of CTL with restricted negation:

Lemma 2.1. *For every set T of* CTL-*operators, we have* CTL$_{\text{mon}}$-MC$(T) \leq_{\text{cd}}$ CTL$_{\text{a.n.}}$-MC$(T) \leq_{\text{cd}}$ CTL$_{\text{pos}}$-MC(T). *Further, for model checking, atomic negation can be eluded, i. e.,* CTL$_{\text{a.n.}}$-MC$(T) \leq_{\text{cd}}$ CTL$_{\text{mon}}$-MC(T).

In Sect. 3.3 we complete the picture by showing that also CTL$_{\text{pos}}$-MC$(T) \leq_{\text{cd}}$ CTL$_{\text{mon}}$-MC(T).

The class P consists of all languages that have a polynomial-time decision algorithm. A problem is P-complete if it is in P and every other problem in P reduces to it. P-complete problems are sometimes referred to as *inherently sequential*, because P-complete problems most likely (formally: if P $\neq$ NC) do not possess NC-algorithms, that is, 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.

There is an NC-algorithm for parsing context-free languages, that is, CFL $\subseteq$ NC. Therefore, complexity theorists have studied the class LOGCFL of all problems reducible to context-free languages (the name "LOGCFL"

refers to the original definition of the class in terms of logspace-reductions, however it is known that the class does not change if instead, as everywhere else in this paper, $\leq_{\mathrm{cd}}$-reductions are used). Hence, LOGCFL $\subseteq$ NC (even LOGCFL $\subseteq$ NC^2, the second level of the NC-hierarchy, where the depth of the occurring circuits is restricted to $O(\log^2 n)$). The class LOGCFL has a number of different maybe even somewhat surprising characterizations, e. g., languages in LOGCFL are those that can be decided by non-deterministic Turing machines operating in polynomial time that have a worktape of logarithmic size and additionally a stack whose size is not bounded.

More important for this paper is the characterization of LOGCFL as those problems computable by SAC^1 circuit families, that is families of circuits that

- have polynomial size and logarithmic depth,
- consist of unbounded fan-in OR gates and bounded fan-in AND gates and negation gates, but the latter are only allowed at the input-level,
- are logtime-uniform (as defined above).

Since the class LOGCFL is known to be closed under complementation, the second condition can equivalently be replaced to allow unbounded fan-in AND gates and restrict the fan-in of OR gates to be bounded.

To summarize:

$$\mathrm{NC}^1 \subseteq \mathrm{L} \subseteq \mathrm{NL} \subseteq \mathrm{LOGCFL} = \mathrm{SAC}^1 \subseteq \mathrm{NC}^2;$$

and problems in these classes possess very efficient parallel algorithms: they can be solved in time $O(\log^2 n)$ on a parallel machine with a tractable number of processors. For more background on these and related complexity classes, we refer the reader to [18].

3. Model Checking CTL and CTL$_{\mathrm{pos}}$

This section contains our main results on the complexity of model checking for CTL and CTL$_{\mathrm{pos}}$. We defer the analysis of the fragments CTL$_{\mathrm{a.n.}}$ and CTL$_{\mathrm{mon}}$ to Sect. 3.3 where we will see that their model-checking problems are computationally equivalent to model checking for CTL$_{\mathrm{pos}}$.

While model checking for CTL in general is known to be polynomial time solvable and in fact P-complete [3, 15], we improve the lower bound by showing that only one temporal operator is sufficient to obtain hardness for P.

Theorem 3.1. *For each nonempty set T of* CTL-*operators,* CTL-MC(T) *is* P-*complete. If $T = \emptyset$, then* CTL-MC(T) *is* NC^1-*complete.*

If we consider only formulae from CTL$_{\mathrm{pos}}$, where no CTL-operators are allowed inside the scope of a negation, the situation changes and the complexity of model checking exhibits a dichotomous behavior. As long as EG or AF are

expressible the model checking problem remains P-complete. Otherwise, its complexity drops to LOGCFL.

Theorem 3.2. *Let T be any set of* CTL-*operators. Then* CTL$_{\mathrm{pos}}$-MC(T) *is*

- NC^1-*complete if $T = \emptyset$,*
- LOGCFL-*complete if $\emptyset \neq T \subseteq \{\mathsf{EX}, \mathsf{EF}\}$ or $\emptyset \neq T \subseteq \{\mathsf{AX}, \mathsf{AG}\}$, and*
- P-*complete otherwise.*

We split the proofs of Theorems 3.1 and 3.2 into the upper and lower bounds in the following two subsections.

3.1. Upper Bounds

In general, model checking for CTL is known to be solvable in P [3]. While this upper bound also applies to CTL$_{\mathrm{pos}}$-MC(T) (for every T), we improve it for positive CTL-formulae using only EX and EF, or only AX and AG.

Proposition 3.3. *Let T be a set of* CTL-*operators such that $T \subseteq \{\mathsf{EX}, \mathsf{EF}\}$ or $T \subseteq \{\mathsf{AX}, \mathsf{AG}\}$. Then* CTL$_{\mathrm{pos}}$-MC$(T)$ *is in* LOGCFL.

Proof. First consider the case $T \subseteq \{\mathsf{EX}, \mathsf{EF}\}$. We claim that Algorithm 1 recursively decides whether the Kripke structure $K = (W, R, \eta)$ satisfies the CTL$_{\mathrm{pos}}(T)$-formula φ in state $w_0 \in W$. There, S is a stack that stores pairs $(\varphi, w) \in \mathrm{CTL}_{\mathrm{pos}}(T) \times W$ and $R^\star$ denotes the transitive closure of R.

Algorithm 1 always terminates because each subformula of φ is pushed to the stack S at most once. For correctness, an induction on the structure of formulae shows that Algorithm 1 returns **false** if and only if for the most recently popped pair (ψ, w) from S, we have $K, w \not\models \psi$. Thence, in particular, Algorithm 1 returns **true** iff $K, w \models \varphi$.

Algorithm 1 can be implemented on a nondeterministic polynomial-time Turing machine that besides its (unbounded) stack uses only logarithmic memory for the local variables. Thus CTL$_{\mathrm{pos}}$-MC(T) is in LOGCFL.

The case $T \subseteq \{\mathsf{AX}, \mathsf{AG}\}$ is analogous and follows from closure of LOGCFL under complementation. $\square$

Finally, for the trivial case where no CTL-operators are present, model checking CTL$(\emptyset)$-formulae is equivalent to the problem of evaluating a propositional formula. This problem is known to be solvable in NC^1 [2].

3.2. Lower Bounds

The P-hardness of model checking for CTL is folklore in the model checking community (cf. [15]), but we could not find a formal proof.[1] We improve this lower bound and

---

[1] In [15], an informal proof sketch is given.

Algorithm 1 Determine whether $K, w_0 \models \varphi$.

Require: a Kripke structure $K = (W, R, \eta)$, $w_0 \in W$,
   $\varphi \in \mathrm{CTL_{pos}}(T)$
 1: $\mathrm{push}(S, (\varphi, w_0))$
 2: **while** S is not empty **do**
 3:   $(\varphi, w) \leftarrow \mathrm{pop}(S)$
 4:   **if** φ is a propositional formula **then**
 5:     **if** φ evaluates to false in w under η **then**
 6:       **return false**
 7:     **end if**
 8:   **else if** $\varphi = \alpha \wedge \beta$ **then**
 9:     $\mathrm{push}(S, (\beta, w))$
10:     $\mathrm{push}(S, (\alpha, w))$
11:   **else if** $\varphi = \alpha \vee \beta$ **then**
12:     nondet. $\mathrm{push}(S, (\alpha, w))$ or $\mathrm{push}(S, (\beta, w))$
13:   **else if** $\varphi = \mathsf{EX}\alpha$ **then**
14:     nondet. choose $w' \in \{w' \mid (w, w') \in R\}$
15:     $\mathrm{push}(S, (\alpha, w'))$
16:   **else if** $\varphi = \mathsf{EF}\alpha$ **then**
17:     nondet. choose $w' \in \{w' \mid (w, w') \in R^\star\}$
18:     $\mathrm{push}(S, (\alpha, w'))$
19:   **end if**
20: **end while**
21: **return true**

concentrate on the smallest fragments of monotone CTL—w. r. t. CTL-operators—with P-hard model checking.

Proposition 3.4. *Let T denote a set of* CTL-*operators. Then* $\mathrm{CTL_{mon}}$-$\mathrm{MC}(T)$ *is* P-*hard if T contains an existential and a universal* CTL-*operator.*

Proof. First, assume that $T = \{\mathsf{AX}, \mathsf{EX}\}$. We give a generic reduction from alternating Turing machines working in logarithmic space. Let M be such a machine and let x be an input to M. We may assume w. l. o. g. that each transition of M leads from an existential to a universal configuration and vice versa. Further we may assume that each computation of M ends after the same number $p(n)$ of steps, where p is a polynomial and n is the length of M's input.

Let $c_1, \ldots, c_{q(n)}$ be an enumeration of all possible configurations of M on input x, starting with the initial configuration c_1 and polynomial q. We construct a Kripke structure $K := (W, R, \eta)$ by defining the set $W := \{c_i^j \mid 1 \le i \le q(n), 0 \le j \le p(n)\}$ and the relation $R \subseteq W \times W$ as

$$R := \left\{ (c_i^j, c_k^{j+1}) \,\middle|\, \begin{array}{l} M \text{ reaches configuration } c_k \text{ from} \\ c_i \text{ in one step}, 0 \le j < p(n) \end{array} \right\}$$
$$\cup \, \{(c_i^{p(n)}, c_i^{p(n)}) \mid 1 \le i \le q(n)\}.$$

The labelling function η is defined as $\eta(w) := \{t\}$ iff w is an accepting configuration, and $\eta(w) = \emptyset$ otherwise. Then it holds that

$$M \text{ accepts } x \iff K, c_1^0 \models \psi_1\Big(\psi_2\big(\cdots \psi_{p(n)}(t)\cdots\big)\Big),$$

where $\psi_i(x) := \mathsf{AX}(x)$ if M's configurations after the ith step are universal, and $\psi_i(x) := \mathsf{EX}(x)$ otherwise. Notice that the constructed CTL-formula does not contain any propositional operator. Since $p(n)$ and $q(n)$ are polynomials, the size of K and φ is polynomial in the size of (M, x). Moreover, K and φ can be constructed from M and x using AC^0-circuits. Thus, $A \le_{\mathrm{cd}} \mathrm{CTL_{mon}}$-$\mathrm{MC}(\{\mathsf{AX}, \mathsf{EX}\})$ for all $A \in \mathrm{ALOGSPACE} = \mathrm{P}$.

For $T = \{\mathsf{AF}, \mathsf{EG}\}$ we take new atomic propositions $d_0, \ldots, d_{p(n)}$ and modify the above reduction by defining the formulas η and ψ_i as follows:

$$\eta(w) := \{d_j \mid w = c_i^j, 1 \le i \le q(n)\} \cup$$
$$\{t \mid w \text{ is an accepting configuration}\}$$

$$\psi_i(x) := \begin{cases} \mathsf{AF}(d_{i+1} \wedge x), & \text{if } M\text{'s configurations in} \\ & \text{step } i \text{ are universal,} \\ \mathsf{EG}(D_{i+1} \vee x), & \text{otherwise,} \end{cases} \quad (1)$$

where $D_i = \bigvee_{i \ne j \in \{0, \ldots, p(n)\}} d_j$.

For the combinations of T being one of $\{\mathsf{AF}, \mathsf{EF}\}$, $\{\mathsf{AF}, \mathsf{EX}\}$, $\{\mathsf{AG}, \mathsf{EG}\}$, $\{\mathsf{AG}, \mathsf{EX}\}$, $\{\mathsf{AX}, \mathsf{EF}\}$, and $\{\mathsf{AX}, \mathsf{EG}\}$, the P-hardness of $\mathrm{CTL_{mon}}$-$\mathrm{MC}(T)$ is obtained using analogous modifications to η and the ψ_i's.

For the remaining combinations involving the until or the release operator, observe that w. r. t. the Kripke structure K as defined in (1), $\mathsf{AF}(d_i \wedge x)$ and $\mathsf{EG}(D_i \vee x)$ are equivalent to $\mathsf{A}[d_{i-1}\mathsf{U}x]$ and $\mathsf{E}[d_{i-1}\mathsf{U}x]$, and R and U are duals. $\qquad\square$

In the presence of arbitrary negation, universal operators are definable by existential operators and vice versa. Hence, from Proposition 3.4 we obtain the following corollary.

Corollary 3.5. *The model checking problem* CTL-$\mathrm{MC}(T)$ *is* P-*hard for each nonempty set T of* CTL-*operators.*

Returning to monotone CTL, in most cases even one operator suffices to make model checking P-hard:

Proposition 3.6. *Let T denote a set of* CTL-*operators. Then* $\mathrm{CTL_{mon}}$-$\mathrm{MC}(T)$ *is* P-*hard if T contains at least one of the operators* EG, EU, ER, AF, AU, *or* AR.

The proof of this proposition proceeds similarly as the proof of Proposition 3.4, but is technically more involved. In essence, it shows that both AX and EX can be simulated by using only EG.

By Lemma 2.1, $\mathrm{CTL_{mon}}$-$\mathrm{MC}(T) \le_{\mathrm{cd}} \mathrm{CTL_{pos}}$-$\mathrm{MC}(T)$ and hence the above results directly translate to model checking for $\mathrm{CTL_{pos}}$: for any set T of temporal operators, $\mathrm{CTL_{pos}}$-$\mathrm{MC}(T)$ is P-hard if $T \not\subseteq \{\mathsf{EX}, \mathsf{EF}\}$ or if $T \not\subseteq \{\mathsf{AX}, \mathsf{AG}\}$. These results cannot be improved w. r. t. T, as for $T \subseteq \{\mathsf{EX}, \mathsf{EF}\}$ and $T \subseteq \{\mathsf{AX}, \mathsf{AG}\}$ we obtain a LOGCFL upper bound for model checking from Proposition 3.3. In the following proposition we prove the matching LOGCFL lower bound.

Proposition 3.7. *For every nonempty set T of* CTL-*operators, the model checking problem* $\mathrm{CTL_{mon}}$-$\mathrm{MC}(T)$ *is* LOGCFL-*hard.*

Proof. As explained in Sect. 2.3, LOGCFL can be characterized as the set of languages recognizable by logtime-uniform SAC^1 circuits, i.e., circuits of logarithmic depth and polynomial size consisting of $\vee$-gates with unbounded fan-in and $\wedge$-gates with fan-in 2. For every single CTL-operator O, we will show that $\mathrm{CTL_{mon}}$-$\mathrm{MC}(T)$ is LOGCFL-hard for all $T \supseteq \{O\}$ by giving a generic $\leq_{\mathrm{cd}}$-reduction f from the word problem for SAC^1 circuits to $\mathrm{CTL_{mon}}$-$\mathrm{MC}(T)$.

First, consider $\mathsf{EX} \in T$. Let C be a logtime-uniform SAC^1 circuit of depth ℓ with n inputs and let $x = x_1 \ldots x_n \in \{0,1\}^n$. Assume w.l.o.g. that C is layered into alternating layers of $\wedge$-gates and $\vee$-gates and that the output gate of C is an $\vee$-gate. We number the layers bottom-up, that is, the layer containing (only) the output gate has level 0, whereas the input-gates and negations of the input-gates are situated in layer ℓ. Denote the graph of C by $G := (V, E)$, where $V := V_{\mathrm{in}} \uplus V_\wedge \uplus V_\vee$ is partitioned into the sets corresponding to the (possibly negated) input-gates, the $\wedge$-gates, and the $\vee$-gates, respectively. G is acyclic and directed with paths leading from the input to the output gates. From (V, E), we construct a Kripke structure that allows to distinguish the two predecessors of an $\wedge$-gate from each other. This will be required to model proof trees using $\mathrm{CTL_{mon}}(\{\mathsf{EX}\})$-formulae.

For $i \in \{1, 2\}$, let $V_{\mathrm{in}}^i := \{v^i \mid v \in V_{\mathrm{in}}\}$, $V_\vee^i := \{v^i \mid v \in V_\vee\}$ and define $V_{\mathrm{in},\vee}^i := V_{\mathrm{in}}^i \cup V_\vee^i$. Further define

$$
\begin{aligned}
E' :=& \big\{(v, u^i) \in V_\wedge \times V_{\mathrm{in},\vee}^i \mid (u,v) \in E \text{ and } u \text{ is the } i\text{th} \\
& \text{predecessor of } v\big\} \cup \big\{(v,v) \mid v \in V_{\mathrm{in}}^1 \cup V_{\mathrm{in}}^2\big\} \cup \\
& \bigcup_{i \in \{1,2\}} \big\{(v^i, u) \in V_{\mathrm{in},\vee}^i \times V_\wedge \mid (u,v) \in E\big\},
\end{aligned}
$$

where the ordering of the predecessors is implicitly given in the encoding of C. We now define a Kripke structure $K := (V', E', \eta)$ with states $V' := V_{\mathrm{in},\vee}^1 \cup V_{\mathrm{in},\vee}^2 \cup V_\wedge$, relation E', and labelling function $\eta : V' \to \mathfrak{P}(\{1, 2, t\})$,

$$
\eta(v) := \begin{cases}
\{i, t\}, & \text{if } v = v_{\mathrm{in}_j} \in V_{\mathrm{in}}^i \text{ and } x_j = 1, \\
\{i, t\}, & \text{if } v = \overline{v}_{\mathrm{in}_j} \in V_{\mathrm{in}}^i \text{ and } x_j = 0, \\
\{i\}, & \text{if } v = v_{\mathrm{in}_j} \in V_{\mathrm{in}}^i \text{ and } x_j = 0, \\
\{i\}, & \text{if } v = \overline{v}_{\mathrm{in}_j} \in V_{\mathrm{in}}^i \text{ and } x_j = 1, \\
\{i\}, & \text{if } v \in V_\vee^i, \\
\emptyset, & \text{otherwise,}
\end{cases}
$$

where $i = 1, 2$, $j = 1, \ldots, n$ and $v_{\mathrm{in}_1}, \ldots, v_{\mathrm{in}_n}$, $\overline{v}_{\mathrm{in}_1}, \ldots, \overline{v}_{\mathrm{in}_n}$ enumerate the input gates and their negations. The formula φ that is to be evaluated on K will consist of atomic propositions 1, 2 and t, Boolean connectives $\wedge$ and

$\vee$, and the CTL-operator EX. To construct φ we recursively define formulae $(\varphi_i)_{0 \leq i \leq \ell}$ by

$$
\varphi_i := \begin{cases}
t, & \text{if } i = \ell, \\
\mathsf{EX}\varphi_{i+1}, & \text{if } i \text{ is even } (\vee\text{-layers}), \\
\bigwedge_{i=1,2} \mathsf{EX}(i \wedge \varphi_{i+1}), & \text{if } i \text{ is odd } (\wedge\text{-layers}).
\end{cases}
$$

We define the reduction function f as the mapping $(C, x) \mapsto (K, v_0, \varphi)$, where v_0 is the node corresponding to the output gate of C and $\varphi := \varphi_0$. We stress that the size of φ is polynomial, for the depth of C is logarithmic only. Clearly, each minimal accepting subtree (cf. [14] or [18, Definition 4.15]) of C on input x translates into a sub-structure K' of K such that $K', v_0 \models \varphi$ where

1. K' includes v_0,

2. K' includes one successor for every node corresponding to an $\vee$-gate, and

3. K' includes the two successors of every node corresponding to an $\wedge$-gate.

As $C(x) = 1$ iff there exists a minimal accepting subtree of C on x, the LOGCFL-hardness of $\mathrm{CTL_{mon}}$-$\mathrm{MC}(T)$ for $\mathsf{EX} \in T$ follows.

Second, consider $\mathsf{EF} \in T$. We have to extend our Kripke structure to contain information about the depth of the corresponding gate. We may assume w.l.o.g. that C is encoded such that each gate contains an additional counter holding the distance to the output gate (which is equal to the number of the layer it is contained in, cf. [18]). We extend η to include this distance i, $1 \leq i \leq \ell$ into "depth-propositions" d_i as in the proof of Proposition 3.4. Denote this modified Kripke structure by K'. Further, we define $(\varphi_i')_{0 \leq i \leq \ell}$ as

$$
\varphi_i' := \begin{cases}
d_\ell \wedge t, & \text{if } i = \ell, \\
\mathsf{EF}(d_{i+1} \wedge \varphi_{i+1}'), & \text{if } i \text{ is even}, \\
\bigwedge_{i=1,2} \mathsf{EF}(d_{i+1} \wedge i \wedge \varphi_{i+1}'), & \text{if } i \text{ is odd}.
\end{cases}
$$

Redefining the reduction f as $(C, x) \mapsto (K', v_0, \varphi_0')$ yields the LOGCFL-hardness of $\mathrm{CTL_{mon}}$-$\mathrm{MC}(T)$ for $\mathsf{EF} \in T$.

Third, let $\mathsf{AX} \in T$. Consider the reduction in case 1 for $\mathrm{CTL_{mon}}(\{\mathsf{EX}\})$-formulae, and let $f(C, x) = (K, v_0, \varphi)$ be the output of the reduction function. It holds that $C(x) = 1$ iff $K, v_0 \models \varphi$, and equivalently $C(x) = 0$ iff $K, v_0 \models \neg\varphi$. Let φ' be the formula obtained from $\neg\varphi$ by multiplying the negation into the formula. Then φ' is a $\mathrm{CTL_{a.n.}}(\{\mathsf{AX}\})$-formula. Since LOGCFL is closed under complement, it follows that $\mathrm{CTL_{a.n.}}$-$\mathrm{MC}(\{\mathsf{AX}\})$ is LOGCFL-hard. Using Lemma 2.1, we obtain that $\mathrm{CTL_{mon}}$-$\mathrm{MC}(\{\mathsf{AX}\})$ is LOGCFL-hard, too.

An analogous argument can be used for the case $\mathsf{AG} \in T$. The remaining fragments are even P-complete by Proposition 3.6. $\qquad\square$

Using Lemma 2.1 we obtain LOGCFL-hardness of $\text{CTL}_{\text{pos}}\text{-MC}(T)$ for all nonempty sets T of CTL-operators.

In the absence of CTL-operators, the lower bound for the model checking problem again follows from the lower bound for evaluating monotone propositional formulae. This problem is known to be hard for NC^1 [2, 16].

3.3. The Power of Negation

We will now show that model checking for the fragments $\text{CTL}_{\text{a.n.}}$ and CTL_{pos} is computationally equivalent to model checking for CTL_{mon}, for any set T of CTL-operators. Since we consider cd-reductions, this is not immediate.

From Lemma 2.1 it follows that the hardness results for $\text{CTL}_{\text{mon}}\text{-MC}(T)$ also hold for $\text{CTL}_{\text{a.n.}}\text{-MC}(T)$ and $\text{CTL}_{\text{pos}}\text{-MC}(T)$. Moreover, the algorithms for $\text{CTL}_{\text{pos}}\text{-MC}(T)$ also work for $\text{CTL}_{\text{mon}}\text{-MC}(T)$ and $\text{CTL}_{\text{a.n.}}\text{-MC}(T)$ without using more computation resources. Both observations together yield the same completeness results for all CTL-fragments with restricted negations.

Theorem 3.8. *Let T be any set of CTL-operators. Then $\text{CTL}_{\text{mon}}\text{-MC}(T)$, $\text{CTL}_{\text{a.n.}}\text{-MC}(T)$, and $\text{CTL}_{\text{pos}}\text{-MC}(T)$ are*

- NC^1-*complete if T is empty,*
- LOGCFL-*complete if $\emptyset \neq T \subseteq \{\text{EX}, \text{EF}\}$ or $\emptyset \neq T \subseteq \{\text{AX}, \text{AG}\}$,*
- P-*complete otherwise.*

As all complete sets for a class are equivalent, we obtain reductions between the model checking problems for all negation-restricted fragments of CTL. This extends Lemma 2.1.

Corollary 3.9. *For every set T of CTL-operators, the problems $\text{CTL}_{\text{mon}}\text{-MC}(T)$, $\text{CTL}_{\text{a.n.}}\text{-MC}(T)$, and $\text{CTL}_{\text{pos}}\text{-MC}(T)$ are equivalent w.r.t. cd-reducibility.*

We remark that this equivalence is not straightforward. Simply applying de Morgan's laws to transform one problem into another requires counting the number of negations on top of $\wedge$- and $\vee$-connectives. This counting cannot be achieved by an AC^0-circuit and does not lead to the aspired reduction. Here we obtain equivalence of the problems as a consequence of our generic hardness proofs in Sect. 3.2.

4. Model Checking Extensions of CTL

What CTL lacks in practice is the ability to express fairness properties. To address this shortcoming, Emerson and Halpern introduced ECTL in [5]. ECTL extends CTL with the $\overset{\infty}{\text{F}}$-operator, which states that for every moment in the future, the enclosed formula will eventually be satisfied again:

for a Kripke structure K, a path $x = (x_1, x_2, \ldots)$, and a path formula χ

$$K, x \models \overset{\infty}{\text{F}}\chi \quad \text{iff} \quad K, x^i \models \text{F}\chi \text{ for all } i \in \mathbb{N}.$$

The dual operator $\overset{\infty}{\text{G}}$ is defined analogously. As for CTL, model checking for ECTL is known to be tractable. Moreover, our next result shows that even for all fragments, model checking for ECTL is not harder than for CTL.

Theorem 4.1. *Let T be a set of temporal operators. Then $\text{ECTL-MC}(T) \equiv_{\text{cd}} \text{CTL-MC}(T')$ and $\text{ECTL}_{\text{pos}}\text{-MC}(T) \equiv_{\text{cd}} \text{CTL}_{\text{pos}}\text{-MC}(T')$, where T' is obtained from T by substituting $\overset{\infty}{\text{F}}, \overset{\infty}{\text{G}}$ with F, G.*

Another extension of CTL is CTL^+ where Boolean combinations of pure temporal operators are allowed in the scope of path quantifiers. In contrast to CTL, model checking for CTL^+ is not tractable, but Δ_2^{p}-complete [8]. Below we classify the complexity of model checking for both full and positive fragments of CTL^+.

Theorem 4.2. *Let T be a set of temporal operators. Then $\text{CTL}^+\text{-MC}(T)$ is*

- NC^1-*complete if $T \subseteq \{\text{A}, \text{E}\}$ or $T = \{\text{X}\}$,*
- P-*complete if $\{\text{X}\} \subsetneq T \subseteq \{\text{A}, \text{E}, \text{X}\}$, and*
- Δ_2^{p}-*complete otherwise.*

Theorem 4.3. *Let T be a set of temporal operators. Then $\text{CTL}_{\text{pos}}^+\text{-MC}(T)$ is*

- NC^1-*complete if $T \subseteq \{\text{A}, \text{E}\}$ or $T = \{\text{X}\}$,*
- LOGCFL-*complete if $T = \{\text{A}, \text{X}\}$ or $T = \{\text{E}, \text{X}\}$,*
- P-*complete if $T = \{\text{A}, \text{E}, \text{X}\}$,*
- NP-*complete if $\text{E} \in T$, $\text{A} \notin T$, and T contains a pure temporal operator aside from X,*
- coNP-*complete if $\text{A} \in T$, $\text{E} \notin T$, and T contains a pure temporal operator aside from X, and*
- Δ_2^{p}-*complete otherwise.*

Finally, ECTL^+ is the combination of ECTL and CTL^+. For its model checking problem we obtain:

Corollary 4.4. *Let T be a set of temporal operators. Then $\text{ECTL}^+\text{-MC}(T) \equiv_{\text{cd}} \text{CTL}^+\text{-MC}(T')$ and $\text{ECTL}_{\text{pos}}^+\text{-MC}(T) \equiv_{\text{cd}} \text{CTL}_{\text{pos}}^+\text{-MC}(T)$, where T' is obtained from T by substituting $\overset{\infty}{\text{F}}, \overset{\infty}{\text{G}}$ with F, G.*

5. Conclusion

We have shown (Theorem 3.2) that model checking for $\text{CTL}_{\text{pos}}(T)$ is already P-complete for most fragments of

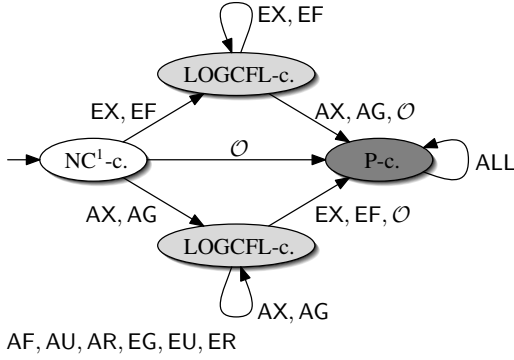**Figure 2. The complexity of** $\text{CTL}_{\text{pos}}\text{-MC}(T)$ **for all sets** T **of** CTL**-operators (depicted as a "finite automaton" where states indicate completeness results and arrows indicate an increase of the set of** CTL**-operators).**

CTL. Only for some weak fragments, model checking becomes easier: if $T \subseteq \{\text{EX}, \text{EF}\}$ or $T \subseteq \{\text{AX}, \text{AG}\}$, then $\text{CTL}_{\text{pos}}\text{-MC}(T)$ is LOGCFL-complete. In the case that no CTL-operators are used, NC^1-completeness of evaluating propositional formulae applies. As a direct consequence (Theorem 3.1), model checking for $\text{CTL}(T)$ is P-complete for every nonempty T. This shows that for the majority of interesting fragments, model checking $\text{CTL}(T)$ is inherently sequential and cannot be sped up by using parallelism.

While all the results above can be transferred to ECTL (Theorem 4.1), CTL^+ and ECTL^+ exhibit different properties. For both logics, the general model checking problem was shown to be complete for Δ_2^{p} in [8]. Here we proved that model checking fragments of $\text{CTL}^+(T)$ and $\text{ECTL}^+(T)$ for $T \subseteq \{\text{A}, \text{E}, \text{X}\}$ remains tractable, while the existential and the universal fragments of $\text{CTL}_{\text{pos}}^+(T)$ and $\text{ECTL}_{\text{pos}}^+(T)$ containing temporal operators other than X are complete for NP and coNP, respectively.

Instead of restricting only the use of negation as done in this paper, one might go one step further and restrict the allowed Boolean connectives in an arbitrary way. One might, e. g., allow the exclusive-OR as the only propositional connective. This has been done for the case of linear temporal logic LTL in [1], where the complexity of $\text{LTL-MC}(T, B)$ for an arbitrary set T of temporal operators and B of propositional connectives was studied. We think that a corresponding study for CTL (or $\text{CTL}^\star$) is an interesting topic for further research. For example, restricting the Boolean connectives to only one of the functions AND or OR leads to many NL-complete fragments, but a full classification is still open. The computational complexity of the corresponding satisfiability problems $\text{CTL-SAT}(T, B)$ and $\text{CTL}^\star\text{-SAT}(T, B)$ has been completely determined in [10].

References

[1] M. Bauland, M. Mundhenk, T. Schneider, H. Schnoor, I. Schnoor, and H. Vollmer. The tractability of model checking for LTL: the good, the bad, and the ugly fragments. In *Proc. Methods for Modalities 5*, pages 125–140. ENS Cachan, 2007. Also at CoRR http://arxiv.org/abs/0805.0498.

[2] S. R. Buss. The Boolean formula value problem is in ALOGTIME. In *Proc. 19th Symposium on Theory of Computing*, pages 123–131. ACM Press, 1987.

[3] E. Clarke, E. A. Emerson, and A. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[4] E. A. Emerson and E. M. Clarke. Using branching time temporal logic to synthesize synchronization skeletons. *Science of Computer Programming*, 2(3):241–266, 1982.

[5] E. A. Emerson and J. Y. Halpern. "Sometimes" and "not never" revisited: on branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, 1986.

[6] E. A. Emerson and C.-L. Lei. Modalities for model checking: Branching time logic strikes back. *Science of Computer Programming*, 8(3):275–306, 1987.

[7] F. Laroussinie. About the expressive power of CTL combinators. *Information Processing Letters*, 54(6):343–345, 1995.

[8] F. Laroussinie, N. Markey, and P. Schnoebelen. Model Checking CTL+ and FCTL is Hard. In *Proc. Foundations of Software Science and Computation Structure (FOSSACS'2001)*, pages 318–331. Springer, 2001.

[9] N. Markey. Past is for free: on the complexity of verifying linear temporal properties with past. *Acta Informatica*, 40(6-7):431–458, 2004.

[10] A. Meier, M. Mundhenk, M. Thomas, and H. Vollmer. The complexity of satisfiability for fragments of CTL and $\text{CTL}^\star$. In *Proc. 2nd Workshop on Reachability Problems in Computational Models (RP 2008)*, pages 201–213, 2008.

[11] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, MA, 1994.

[12] A. Pnueli. The temporal logic of programs. In *Proc. 18th Symposium on Foundations of Computer Science*, pages 46–57. IEEE Computer Society Press, 1977.

[13] K. Regan and H. Vollmer. Gap-languages and log-time complexity classes. *Theoretical Computer Science*, 188:101–116, 1997.

[14] W. L. Ruzzo. Tree-size bounded alternation. *Journal of Computer and System Sciences*, 21:218–235, 1980.

[15] P. Schnoebelen. The complexity of temporal logic model checking. In *Advances in Modal Logic*, pages 393–436, 2002.

[16] H. Schnoor. The complexity of the Boolean formula value problem. Technical report, Theoretical Computer Science, University of Hannover, 2005.

[17] A. Sistla and E. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985.

[18] H. Vollmer. *Introduction to Circuit Complexity – A Uniform Approach*. Texts in Theoretical Computer Science. Springer Verlag, Berlin Heidelberg, 1999.