This is an author produced version of *Model Checking CTL is Almost Always Inherently Sequential*.

White Rose Research Online URL for this paper:
http://eprints.whiterose.ac.uk/74808/

**Proceedings Paper:**

http://dx.doi.org/10.1109/TIME.2009.12
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

---

*Supported in part by grants DFG VO 630/6-1 and DAAD-ARC D/08/08881.
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*-formulae as follows. Let Φ be a finite set of atomic propositions. The symbols used are the atomic propositions in Φ, the constant symbols ⊤ and ⊥, the Boolean connectives ¬, ∧, and ∨, 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 ⊤ and ⊥ 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 ¬ϕ, (ϕ ∧ ψ), (ϕ ∨ ψ), Aχ, Eχ are state formulae, and ¬χ, (χ ∧ π), (χ ∨ π), Xχ, Fχ, Gχ, [χ U π], and [χ R π] are path formulae. The set of CTL*-formulae (or formulae) consists of all state formulae.

A Kripke structure is a triple K = (W, R, η), where W is a finite set of states, R ⊆ W × W a total relation (i.e., for each w ∈ W, there exists a w’ such that (w, w’) ∈ R), and η: W → Φ(Φ) is a labelling function. A path x is an infinite sequence x = (x1, x2, ...) ∈ Wω such that (xi, xi+1) ∈ R, for all i ≥ 1. For a path x = (x1, x2, ...) we denote by x1 the path (xi, xi+1, ...).

Let K = (W, R, η) be a Kripke structure, w ∈ W be a state, and x = (x1, x2, ...) ∈ Wω a path. Further, let ϕ, ψ be state formulae and χ, π be path formulae. The truth of a CTL*-formula w.r.t. K is inductively defined as:

K, w |= ⊤ always,
K, w |= ⊥ never,
K, w |= p if p ∈ Φ and p ∈ η(w),
K, w |= ¬ϕ if K, w ̸|= ϕ,
K, w |= (ϕ ∧ ψ) if K, w |= ϕ and K, w |= ψ,
K, w |= (ϕ ∨ ψ) if K, w |= ϕ or K, w |= ψ,
K, w |= Aχ if K, x |= χ for all paths x = (x1, x2, ...) with x1 = w,
K, x |= ϕ if K, x1 |= ϕ,
K, x |= ¬χ if K, x ̸|= χ,
K, x |= (χ ∧ π) if K, x |= χ and K, x |= π,
K, x |= (χ ∨ π) if K, x |= χ or K, x |= π,
K, x |= Xχ if K, x2 |= χ
K, x |= [χ U π] if there exists k ∈ N such that K, x+i |= χ for 1 ≤ i < k and K, x+k |= π.

The semantics of the remaining temporal operators is defined via the equivalences: Eχ ≡ ¬A¬χ, Fχ ≡ [T U χ], Gχ ≡ ¬F¬χ, and [χ R π] ≡ ¬[¬χ U ¬π]. A state formula ϕ is satisfied by a Kripke structure K if there exists w ∈ W such that K, w |= ϕ. We will also denoted this by K |= ϕ.

A CTL-formula is a CTL*-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*-formulae. For example, AG(Fp ∧ Fq) is a CTL-formula, but A(G(Fp ∧ 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-
operators, and EX, EF, EG, EU, and ER are existential CTL-operators. Let \( \text{ALL} \) denote the set of all CTL-operators. Note that \( [\psi \cup \chi] \equiv \text{AF} \chi \land \lnot \exists \psi \cup (\lnot \psi \land \lnot \chi) \), and thus \( E[\psi \Gamma \chi] \equiv \text{EG} \chi \lor E[\psi (\chi \lor \psi \land \chi)] \). Hence \{AX, AF, AR\} is a minimal set of CTL-operators for CTL (in presence of all Boolean connectives), whereas \{AX, AG, AU\} is not complete for CTL [7].

By CTL\((T)\) we denote the set of CTL-formulae using the connectives \{\land, \lor, \lnot\} 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 \( \text{MC} \) for CTL\(_{\text{a.n.}}\)-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”

![Figure 1. The expressive power of CTL\((T)\).](image-url)
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, \( \leq_{cd} \)-reductions are used). Hence, \( \text{LOGCFL} \subseteq \text{NC} \) (even \( \text{LOGCFL} \subseteq \text{NC}^2 \), the second level of the NC-hierarchy, where the depth of the occurring circuits is restricted to \( O(\log^2 n) \)). The class \( \text{LOGCFL} \) has a number of different maybe even somewhat surprising characterizations, e.g., languages in \( \text{LOGCFL} \) 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 \( \text{LOGCFL} \) as those problems computable by \( \text{SAC}^1 \) 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 \( \text{LOGCFL} \) 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:

- \( \text{NC}^1 \subseteq \text{L} \subseteq \text{NL} \subseteq \text{LOGCFL} = \text{SAC}^1 \subseteq \text{NC}^2 \);
- 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\(_{\text{pos}}\)

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

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, \( \text{CTL-MC}(T) \) is P-complete. If \( T = \emptyset \), then \( \text{CTL-MC}(T) \) is \( \text{NC}^1 \)-complete.

If we consider only formulae from CTL\(_{\text{pos}}\), 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 \( \text{LOGCFL} \).

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

- \( \text{NC}^1 \)-complete if \( T = \emptyset \),
- \( \text{LOGCFL}\)-complete if \( \emptyset \neq T \subseteq \{\text{EX}, \text{EF}\} \) or \( \emptyset \neq T \subseteq \{\text{AX}, \text{AG}\} \), 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 \( \text{CTL}_{\text{pos}}\cdot \text{MC}(T) \) (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 \( T \subseteq \{\text{EX}, \text{EF}\} \) or \( T \subseteq \{\text{AX}, \text{AG}\} \). Then \( \text{CTL}_{\text{pos}}\cdot \text{MC}(T) \) is in \( \text{LOGCFL} \).

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

Algorithm 1 always terminates because each subformula of \( \varphi \) 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 \((\psi, w) \) 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 \( \text{CTL}_{\text{pos}}\cdot \text{MC}(T) \) is in \( \text{LOGCFL} \).

The case \( T \subseteq \{\text{AX}, \text{AG}\} \) is analogous and follows from closure of \( \text{LOGCFL} \) under complementation. \( \Box \)

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 \( \text{NC}^1 \) [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 \( K = (W, R, \eta) \), \( w_0 \in W \), \( \varphi \in \text{CTL}_{\text{pos}}(T) \)

1: \( \text{push}(S, (\varphi, w_0)) \)
2: while \( S \) is not empty do
3: \( (\varphi, w) \leftarrow \text{pop}(S) \)
4: if \( \varphi \) is a propositional formula then
5: if \( \varphi \) evaluates to false in \( w \) under \( \eta \) then
6: return false
7: end if
8: else if \( \varphi = \alpha \land \beta \) then
9: \( \text{push}(S, (\beta, w)) \)
10: \( \text{push}(S, (\alpha, w)) \)
11: end if
12: else if \( \varphi = \text{EX} \alpha \) then
13: nondet. \( \text{push}(S, (\alpha, w)) \) or \( \text{push}(S, (\beta, w)) \)
14: end if
15: else if \( \varphi = \text{EF} \alpha \) then
16: nondet. choose \( w' \in \{ w' \mid (w, w') \in R \} \)
17: \( \text{push}(S, (\alpha, w')) \)
18: end if
19: end while
20: 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 \( \text{CTL}_{\text{mon}}(\text{MC}(T)) \) is P-hard if \( T \) contains an existential and a universal CTL-operator.

**Proof.** First, assume that \( T = \{ \text{AX}, \text{EX} \} \). 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 \( c_1, \ldots, c_{p(n)} \) 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 \( W := \{ c_i \mid 1 \leq i \leq q(n) \} \) and the relation \( R \subseteq W \times W \) as

\[
R := \left\{ (c_i, c_{i+1}) \mid \begin{array}{l}
M \text{ reaches configuration } c_k \text{ from } c_i \text{ in one step, } 0 \leq j < p(n) \\
\text{or } (c_i, c_{i+1}) \text{ in one step, } 0 \leq j \leq p(n)
\end{array} \right\}
\]

\[
\cup \left\{ (c_i^{p(n)}, c_i^{p(n)}) \mid 1 \leq i \leq q(n) \right\}.
\]

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

\[
M \text{ accepts } x \iff K, c_0^0 \models \psi_1 \left( \psi_2 \left( \cdots \psi_{p(n)}(t) \right) \cdots \right),
\]

where \( \psi_i(x) := \text{AX}(x) \) if \( M \)'s configurations after the \( i \)th step are universal, and \( \psi_i(x) := \text{EX}(x) \) 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 \( \varphi \) is polynomial in the size of \( (M, x) \).

Moreover, \( K \) and \( \varphi \) can be constructed from \( M \) and \( x \) using \( AC^0 \)-circuits. Thus, \( A \leq_{\text{cd}} \text{CTL}_{\text{mon}}(\text{MC}(\{ \text{AX}, \text{EX} \})) \) for all \( A \in \text{ALOGSPACE} = P \).

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

\[
\eta(w) := \left\{ d_j \mid w = c_i^j, 1 \leq i \leq q(n) \right\} \cup \left\{ t \mid w \text{ is an accepting configuration} \right\},
\]

\[
\psi_i(x) := \begin{cases}
\text{AF}(d_{i+1} \land x), & \text{if } M \text{'s configurations in } (1) \\
\text{EF}(D_i \lor x), & \text{otherwise},
\end{cases}
\]

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

For the combinations of \( T \) being one of \{ \text{AF}, \text{EF} \}, \{ \text{AF}, \text{EX} \}, \{ \text{AG}, \text{EG} \}, \{ \text{AG}, \text{EX} \}, \{ \text{AX}, \text{EF} \}, \text{and } \{ \text{AX}, \text{EG} \} \), the P-hardness of \( \text{CTL}_{\text{mon}}(\text{MC}(T)) \) is obtained using analogous modifications to \( \eta \) and the \( \psi_i \)’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), \( \text{AF}(d_i \land x) \) and \( \text{EG}(D_i \lor x) \) are equivalent to \( \text{A}[d_{i-1} U x] \) and \( \text{E}[d_{i-1} U x] \), and \( R \) and \( U \) are duals. \( \square \)

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 \( \text{CTL}_{\text{MC}(T)} \) 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 \( \text{CTL}_{\text{mon}}(\text{MC}(T)) \) is P-hard if \( T \) contains at least one of the operators \( \text{EG}, \text{EU}, \text{AF}, \text{AU}, \text{or AR} \).

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 \( \text{AX} \) and \( \text{EX} \) can be simulated by using only \( \text{EG} \).

By Lemma 2.1, \( \text{CTL}_{\text{mon}}(\text{MC}(T)) \leq_{\text{cd}} \text{CTL}_{\text{pos}}(\text{MC}(T)) \) and hence the above results directly translate to model checking for \( \text{CTL}_{\text{pos}} \): for any set \( T \) of temporal operators, \( \text{CTL}_{\text{pos}}(\text{MC}(T)) \) is P-hard if \( T \not\subseteq \{ \text{EX}, \text{EF} \} \) or \( T \not\subseteq \{ \text{AX}, \text{AG} \} \). These results cannot be improved w.r.t. \( T \), as for \( T \subseteq \{ \text{EX}, \text{EF} \} \) and \( T \subseteq \{ \text{AX}, \text{AG} \} \) 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 $\text{CTL}$-operators, the model checking problem $\text{CTL}_{\text{mon}}\text{-MC}(T)$ 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 $\lor$-gates with unbounded fan-in and $\land$-gates with fan-in 2. For every single $\text{CTL}$-operator $O$, we will show that $\text{CTL}_{\text{mon}}\text{-MC}(T)$ is LOGCFL-hard for all $T \supseteq \{O\}$ by giving a generic $\leq_{cd}$-reduction $f$ from the word problem for SAC$^1$ circuits to $\text{CTL}_{\text{mon}}\text{-MC}(T)$.

First, consider $EX \in T$. Let $C$ be a logtime-uniform SAC$^1$ circuit of depth $\ell$ 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 $\land$-gates and $\lor$-gates and that the output gate of $C$ is an $\lor$-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 $\ell$. Denote the graph of $C$ by $G := (V,E)$, where $V := V_{in} \cup V_{\land} \cup V_o$ is partitioned into the sets corresponding to the (possibly negated) input-gates, the $\land$-gates, and the $\lor$-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 $\land$-gate from each other. This will be required to model proof trees using $\text{CTL}_{\text{mon}}\{\{\lor\}\}$-formulae.

For $i \in \{1, 2\}$, let $V^i_{in} := \{v \mid v \in V_{in}\}$, $V^i_{\land} := \{v \mid v \in V_{\land}\}$ and define $V^i_{in,\lor} := V^i_{in} \cup V^i_{\lor}$. Further define

$$E' := \{(v, u') \in V_{\land} \times V^i_{in,\lor} \mid (u, v) \in E \text{ and } u \text{ is the } i\text{th predecessor of } v\} \cup \{(v, v) \mid v \in V^i_{in} \cup V^i_{\land}\} \cup \bigcup_{i=1,2} \{(v^i, u) \in V^i_{in,\lor} \times V_{\land} \mid (u, v) \in E\},$$

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 $V' := V^1_{in,\lor} \cup V^2_{in,\lor} \cup V_{\land}$, relation $E'$, and labelling function $\eta : V' \rightarrow \mathfrak{P}\{\{1, 2\}, t\}$.

$$\eta(v) := \begin{cases} \{i, t\}, & \text{if } v = v_{inj} \in V^i_{in} \text{ and } x_j = 1, \\ \{i\}, & \text{if } v = v_{inj} \in V^i_{in} \text{ and } x_j = 0, \\ \{j\}, & \text{if } v = v_{inj} \in V^i_{in} \text{ and } x_j = 0, \\ \{i\}, & \text{if } v = v_{inj} \in V^i_{in} \text{ and } x_j = 1, \\ \{i\}, & \text{if } v = v_{inj} \in V^2_{in} \text{ and } x_j = 1, \\ \emptyset, & \text{otherwise}, \end{cases}$$

where $i = 1, 2$, $j = 1, \ldots, n$ and $v_{in1}, \ldots, v_{inn}$, $v_{in1}, \ldots, v_{inn}$ enumerate the input gates and their negations. The formula $\varphi$ that is to be evaluated on $K$ will consist of atomic propositions $1, 2$ and $t$, Boolean connectives $\land$ and $\lor$, and the $\text{CTL}$-operator $EX$. To construct $\varphi$ we recursively define formulae $(\varphi_i)_{0 \leq i \leq \ell}$ by

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

We define the reduction function $f$ as the mapping $(C, x) \mapsto \langle K, v_0, \varphi \rangle$, where $v_0$ is the node corresponding to the output gate of $C$ and $\varphi := \varphi_0$. We stress that the size of $\varphi$ 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 $K'$ such that $K', v_0 \models \varphi$ where

1. $K'$ includes $v_0$,
2. $K'$ includes one successor for every node corresponding to an $\lor$-gate, and
3. $K'$ includes the two successors of every node corresponding to an $\land$-gate.

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

Second, consider $EF \in T$. 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 $\eta$ 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 $(\varphi')_{0 \leq i \leq \ell}$ as

$$\varphi'_i := \begin{cases} d_i \land t, & \text{if } i = \ell, \\ EF(d_{i+1} \land \varphi'_{i+1}), & \text{if } i \text{ is even}, \\ \land_{i=1,2} EF(d_{i+1} \land i \land \varphi'_{i+1}), & \text{if } i \text{ is odd}. \end{cases}$$

Redefining the reduction function $f$ as $(C, x) \mapsto \langle K', v_0, \varphi' \rangle$ yields the LOGCFL-hardness of $\text{CTL}_{\text{mon}}\text{-MC}(T)$ for $EF \in T$.

Third, let $AX \in T$. Consider the reduction in case 1 for $\text{CTL}_{\text{mon}}\{\{\lor\}\}$-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 $\varphi'$ be the formula obtained from $\neg \varphi$ by multiplying the negation into the formula. Then $\varphi'$ is a $\text{CTL}_{\text{mon}}\{\{\land\}\}$-formula. Since LOGCFL is closed under complement, it follows that $\text{CTL}_{\text{mon}}\text{-MC}(\{\land\})$ is LOGCFL-hard. Using Lemma 2.1, we obtain that $\text{CTL}_{\text{mon}}\text{-MC}(\{\land\})$ is LOGCFL-hard, too.

An analogous argument can be used for the case $AG \in T$. The remaining fragments are even P-complete by Proposition 3.6.
Using Lemma 2.1 we obtain LOGCFL-hardness of CTL\textsubscript{pos}-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\textsuperscript{1} [2, 16].

3.3. The Power of Negation

We will now show that model checking for the fragments CTL\textsubscript{a,n} and CTL\textsubscript{pos} is computationally equivalent to model checking for CTL\textsubscript{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 CTL\textsubscript{mon}-MC(T) also hold for CTL\textsubscript{a,n}-MC(T) and CTL\textsubscript{pos}-MC(T). Moreover, the algorithms for CTL\textsubscript{pos}-MC(T) also work for CTL\textsubscript{mon}-MC(T) and CTL\textsubscript{a,n}-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 CTL\textsubscript{mon}-MC(T), CTL\textsubscript{a,n}-MC(T), and CTL\textsubscript{pos}-MC(T) are

- NC\textsuperscript{1}-complete if T is empty,
- LOGCFL-complete if \( \emptyset \neq T \subseteq \{EX, EF\} \) or \( \emptyset \neq T \subseteq \{AX, AG\} \),
- 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 CTL\textsubscript{mon}-MC(T), CTL\textsubscript{a,n}-MC(T), and CTL\textsubscript{pos}-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 \( \land \)- and \( \lor \)-connectives. This counting cannot be achieved by an AC\textsuperscript{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 \( \bar{F} \)-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 \( \chi \)

\[
K, x \models \bar{F}\chi \text{ iff } K, x^i \models F\chi \text{ for all } i \in \mathbb{N}.
\]

The dual operator \( \bar{G} \) 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 ECTL-MC(T) \( \equiv_{cd} \) CTL-MC(T) and ECTL\textsubscript{pos}-MC(T) \( \equiv_{cd} \) CTL\textsubscript{pos}-MC(T'), where T' is obtained from T by substituting \( \bar{F}, \bar{G} \) with F, G.

Another extension of CTL is CTL\textsuperscript{+} where Boolean combinations of pure temporal operators are allowed in the scope of path quantifiers. In contrast to CTL, model checking for CTL\textsuperscript{+} is not tractable, but \( \Delta^p_2 \)-complete [8]. Below we classify the complexity of model checking for both full and positive fragments of CTL\textsuperscript{+}.

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

- NC\textsuperscript{1}-complete if \( T \subseteq \{A, E\} \) or \( T = \{X\} \),
- P-complete if \( \{X\} \subseteq T \subseteq \{A, E, X\} \), and
- \( \Delta^p_2 \)-complete otherwise.

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

- NC\textsuperscript{1}-complete if \( T \subseteq \{A, E\} \) or \( T = \{X\} \),
- LOGCFL-complete if \( T = \{A, X\} \) or \( T = \{E, X\} \),
- P-complete if \( T = \{A, E, X\} \),
- NP-complete if \( E \in T \), \( A \notin T \), and T contains a pure temporal operator aside from X,
- coNP-complete if \( A \in T \), \( E \notin T \), and T contains a pure temporal operator aside from X, and
- \( \Delta^p_2 \)-complete otherwise.

Finally, ECTL\textsuperscript{+} is the combination of ECTL and CTL\textsuperscript{+}. For its model checking problem we obtain:

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

5. Conclusion

We have shown (Theorem 3.2) that model checking for CTL\textsubscript{pos}(T) is already P-complete for most fragments of
was shown to be complete for $\Delta$ (Theorem 3.1), model checking for CTL (Theorem 4.1), CTL propositional formulae applies. As a direct consequence of temporal logic $LTL$ in $[\text{tional connective. This has been done for the case of lin-}]

might, e. g., allow the exclusive-OR as the only proposi-
tional connectives was studied. W e think

the allowed Boolean connectives in an arbitrary way . One

one might go one step further and restrict

[10].

model checking fragments of $CTL$ and $CTL^+$ 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 proper-
ties. For both logics, the general model checking problem

was shown to be complete for $\Delta^P$ in [8]. Here we proved

that model checking fragments of $CTL^+(T)$ and $ECTL^+(T)$
for $T \subseteq \{A, E, X\}$ remains tractable, while the existential

and the universal fragments of $CTL^+_\text{pos}(T)$ and $ECTL^+_\text{pos}(T)$
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 proposi-
tional connective. This has been done for the case of lin-
ear temporal logic $LTL$ in [1], where the complexity of

$LTL$-$MC(T, B)$ for an arbitrary set $T$ of temporal operators

and $B$ of propositional connectives was studied. We think

that a corresponding study for $CTL$ (or $CTL^+$) is an inter-

esting 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 clas-

sification is still open. The computational complexity of

the corresponding satisfiability problems $CTL$-$SAT(T, B)$ and

$CTL^+$-$SAT(T, B)$ 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 check-
ing for LTL: the good, the bad, and the ugly fragments. In
Proc. Methods for modalities 5, pages 125–140. ENS Cachan,

[2] S. R. Buss. The Boolean formula value problem is in ALOG-
TIME. In Proc. 19th Symposium on Theory of Computing,

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

poral logic to synthesize synchronization skeletons. Science

never” revisited: on branching versus linear time temporal

Branching time logic strikes back. Science of Computer

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

[8] F. Laroussinie, N. Markey, and P. Schnoebelen. Model Check-
ing $CTL^+$ and $FCTL$ is Hard. In Proc. Foundations of Soft-
ware Science and Computation Structure (FOSSACS’2001),

linear temporal properties with past. Acta Informatica,

complexity of satisfiability for fragments of CTL and $CTL^+$.
In Proc. 2nd Workshop on Reachability Problems in Compu-


Symposium on Foundations of Computer Science, pages 46–

plexity classes. Theoretical Computer Science, 188:101–116,
1997.


[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