Model checking CTL^+ and FCTL is hard

F. Laroussinie, N. Markey, and Ph. Schnoebelen

Lab. Spécification & Vérification
ENS de Cachan & CNRS UMR 8643
61, av. Pdt. Wilson, 94235 Cachan Cedex France
email: {fl,markey,phs}@lsv.ens-cachan.fr

Abstract. Among the branching-time temporal logics used for the specification and verification of systems, CTL^+, FCTL and ECTL^+ are the most notable logics for which the precise computational complexity of model checking is not known. We answer this longstanding open problem and show that model checking these (and some related) logics is Δ_2^p-complete.

1 Introduction

Temporal Logic. Since [Pnu77], *temporal logic* is a widely used formalism for reasoning about reactive systems. Temporal logic allows *model checking*, i.e. the automatic verification that (a finite state model of) the system under study satisfies (the temporal formulae formalizing) its expected behavioral specifications. We refer to [Eme90,CGP99] for more motivations and background.

There exists a wide variety of different temporal logics, and it is still debated what should be the temporal logic of choice. However, it is fair to say that the three most popular temporal logics are PLTL, CTL and CTL^*. PLTL is the linear-time logic built on U ("until") and X ("next") while CTL is the branching-time logic built on these same modalities (hence the notations $\mathrm{PLTL} = \mathrm{L}(\mathsf{U}, \mathsf{X})$ and $\mathrm{CTL} = \mathrm{B}(\mathsf{U}, \mathsf{X})$ in [Eme90]). CTL^*, introduced in [EH86], was designed to be more expressive than both PLTL and CTL.

CTL *and fairness properties.* Several fragments of CTL^* are defined and studied in [EH85,EH86,ES89,Eme90] and other papers, where their expressive powers are compared. Clearly, what CTL really lacks in practice is the ability to express fairness properties, and this is what motivates the introduction in [EH86] of ECTL [1], or $\mathrm{B}(\mathsf{U}, \mathsf{X}, \overset{\infty}{\mathsf{F}})$, an extension of CTL with the $\mathsf{E}\overset{\infty}{\mathsf{F}}$ modality for stating fairness conditions. ECTL sits between CTL and CTL^* and, like CTL, it admits a polynomial-time model checking algorithm (while model checking CTL^* is PSPACE-complete).

[1] For "Extended CTL". There are two standard ways of denoting the logics we consider in this paper: [ES89] and [EH83] use the names ECTL, ECTL^+, etc., while [EH86] and [Eme90] use the notation $\mathrm{B}(\ldots)$. Here we use preferably the first series.

One thing ECTL lacks is the ability to *combine* fairness properties, and this is what motivated the introduction in [EH86] [2] of ECTL^+, where several temporal modalities can be combined in a boolean way (but not nested) under a path quantifier. Hence ECTL^+ allows stating $\mathsf{E}(\overset{\infty}{\mathsf{F}}A \wedge \overset{\infty}{\mathsf{F}}B)$, i.e. "there exists a path where A and B occur infinitely often", and $\mathsf{A}\big(\overset{\infty}{\mathsf{F}}A \;\Rightarrow\; B\mathsf{U}C\big)$, i.e. "all paths with infinitely many A satisfy $B\mathsf{U}C$". This makes ECTL^+ expressive enough in practical situations.

There exist other proposals aiming at extending CTL so that it can express fairness properties. These are FCTL, GFCTL (both from [EL87]), and CTL^F (from [CES86]), all of them logics where the fairness constraints are stated more or less outside of the temporal property itself (see section 2.4).

CTL *and* CTL^+. The idea of allowing boolean combinations of temporal modalities has also been applied to CTL (and other logics). In CTL^+ one can state $\mathsf{A}\big(\mathsf{G}C \wedge \mathsf{X}D \;\Rightarrow\; B\mathsf{U}C\big)$, i.e. "all paths with C everywhere and D in next state, satisfy $B\mathsf{U}C$".

A surprising result is that CTL^+ is not more expressive than CTL [EH85] while ECTL^+ is more expressive than ECTL [EH86] (see also [RS00]). However CTL^+ can be much more succinct than CTL, a fact that was conjectured since [EH85] but has only been proved recently [Wil99].

The complexity of model checking. That CTL^+ can be exponentially more succinct than CTL suggests that model checking can be harder for CTL^+ than for CTL. Indeed, while model checking CTL (or ECTL) is P-complete, model checking CTL^+ (or ECTL^+) is NP-hard and coNP-hard. This lower bound is a consequence of well-known results (from [ON80,SC85]) on the complexity of L(F). These same results entail that model checking CTL^+ can be done in P^NP (that is, in Δ_2^p, see section 3) as was observed in [CES86, Theo. 6.2]. Clearly, the same lower and upper bounds apply to FCTL and GFCTL.

Beyond these observations, nothing more is known about the complexity of model checking CTL^+, FCTL and ECTL^+, three notable branching-time logics for which the computational complexity of model checking has not been characterized precisely. Also note that [EL87, Coro. 4.8] incorrectly states that model-checking FCTL is NP-complete [3].

Our contribution. In this paper, we prove that model checking CTL^+, ECTL^+ and FCTL (and some related logics) is Δ_2^p-complete, thereby solving a long-standing open problem.

The result is surprising since Δ_2^p is a class for which very few complete problems are known. Indeed, in the polynomial-time hierarchy, the classes Σ_k^p or Π_k^p are more populated than the Δ_k^p. As far as we know, our result provides the first

[2] The logic called CTF in [EC80] is essentially ECTL^+.

[3] It seems that [EL87] implicitly assumes Turing or Cook reductions, instead of the usual many-one reductions. Turing reductions are too general for problems in NP and [EL87] does not prove membership in NP.

examples of Δ_2^p-complete problems from the field of temporal model checking and we believe it can be interesting outside of that field.

Plan of the paper. We first recall the necessary preliminary notions from temporal logic (section 2) and Δ_2^p-completeness (section 3). Sections 4 and 5 contain the main result, a reduction from SNSAT into model checking problems. Then section 6 shows that model checking for ECTL$^+$ is in Δ_2^p. Finally, a conclusion summarizes what has been proved.

2 Branching-time temporal logic

2.1 Syntax

We write $\mathbb{N}$ for the set of natural numbers, and $AP = \{P_1, P_2, \ldots\}$ for a countable set of *atomic propositions.*

The formulae of ECTL$^+$ are given by the following grammar:

$$\begin{aligned} \varphi, \psi \; &::= \; \mathsf{E}\varphi_p \mid \neg\varphi \mid \varphi \wedge \psi \mid P_1 \mid P_2 \mid \ldots \\ \text{and } \; \varphi_p, \psi_p \; &::= \; \varphi \mathsf{U} \psi \mid \mathsf{X}\varphi \mid \overset{\infty}{\mathsf{F}} \varphi \mid \neg\varphi_p \mid \varphi_p \wedge \psi_p \mid P_1 \mid P_2 \mid \ldots \end{aligned}$$

where only *state formulae* (ranged over by $\varphi, \psi, \ldots$) are considered as *bona fide* ECTL$^+$ formulae, while *path formulae* (ranged over by $\varphi_p, \psi_p, \ldots$) only occur as subformulae.

We use the standard abbreviations $\top$, $\bot$, $\varphi \vee \psi$, $\varphi \Rightarrow \psi$, as well as $\mathsf{A}\varphi_p$ (for $\neg\mathsf{E}\neg\varphi_p$), $\mathsf{F}\varphi$ (for $\top\mathsf{U}\varphi$), $\mathsf{G}\varphi$ (for $\neg\mathsf{F}\neg\varphi$) and $\overset{\infty}{\mathsf{G}} \varphi$ (for $\neg \overset{\infty}{\mathsf{F}} \neg\varphi$).

Remark 2.1. Classical definitions of CTL$^+$ and ECTL$^+$ do not allow atomic propositions P_1,... as path formulae. We use such path formulae for clarity but will avoid them in the proof of our main hardness result. Hence all our results also hold with the restricted definition.

2.2 Semantics

ECTL$^+$ formulae are interpreted over states (also called nodes) in Kripke structures. Formally

Definition 2.2. *A Kripke structure (a "KS") is a tuple $S = \langle Q_S, q_0, R_S, l_S\rangle$ where $Q_S = \{q, \ldots\}$ is a non-empty set of* nodes*, $R_S \subseteq Q_S \times Q_S$ is a total* transition relation*, and $l_S : Q_S \to 2^{AP}$ labels every node with the propositions it satisfies.*

We only consider finite KSs, i.e. KSs where Q_S and all $l_S(q)$ are finite. The size of a finite KS, written $|S|$, is defined as $|Q_S| + |R_S|$, i.e. the size of the underlying directed graph.

Below, we drop the "S" subscript in our notations whenever no ambiguity will arise. A *computation* (or a *path*) in a KS is an infinite sequence π of the form $q_0 q_1 \ldots$ s.t. $(q_i, q_{i+1}) \in R$ for all $i \in \mathbb{N}$. For $i \in \mathbb{N}$, $\pi(i)$ denotes q_i, the i-th node of π. We write $\Pi(q)$ for the set of all computations starting from q. $\Pi(q)$ is never empty since R is total.

Fig. 1 defines when a node q (a path π) in some KS S, satisfies an ECTL$^+$ formula φ (resp. path formula φ_p), written $q \models_S \varphi$ (resp. $\pi \models_S \varphi_p$), by induction over the structure of the formulae. As usual, we write $S \models \varphi$ when $q_0 \models_S \varphi$.

$q \models \mathsf{E}\varphi_p$	iff there exists $\pi \in \Pi(q)$ s.t. $\pi \models \varphi_p$,
$q \models \neg\varphi$	iff $q \not\models \varphi$,
$q \models \varphi \wedge \psi$	iff $q \models \varphi$ and $q \models \psi$,
$q \models P_i$	iff $P_i \in l(q)$,
$\pi \models \varphi \mathsf{U} \psi$	iff there exists $i \geq 0$ s.t. $\pi(i) \models \psi$ and $\pi(j) \models \varphi$ for all $0 \leq j < i$,
$\pi \models \mathsf{X}\varphi$	iff $\pi(1) \models \varphi$,
$\pi \models \overset{\infty}{\mathsf{F}}\, \varphi$	iff for all $i \geq 0$ there is a $j > i$ s.t. $\pi(j) \models \varphi$,
$\pi \models \neg\varphi_p$	iff $\pi \not\models \varphi_p$,
$\pi \models \varphi_p \wedge \psi_p$	iff $\pi \models \varphi_p$ and $\pi \models \psi_p$,
$\pi \models P_i$	iff $P_i \in l(\pi(0))$.

Fig. 1. Semantics of ECTL$^+$

2.3 Fragments of ECTL$^+$

Several branching-time logics can be seen as fragments of ECTL$^+$:

- ECTL, denoted $\mathrm{B}(\mathsf{U}, \mathsf{X}, \overset{\infty}{\mathsf{F}})$ in [Eme90], is the fragment of ECTL$^+$ where the path quantifiers E or A are immediately over a temporal modality U, X or $\overset{\infty}{\mathsf{F}}$ (no boolean combinator is allowed in between).
- CTL [CE81], or $\mathrm{B}(\mathsf{U}, \mathsf{X})$, is the fragment of ECTL where $\overset{\infty}{\mathsf{F}}$ is not allowed.
- UB [BPM83], or $\mathrm{B}(\mathsf{X}, \mathsf{F})$, is the fragment of CTL where U is only allowed in the weaker form of F.
- BTL [Lam80], or $\mathrm{B}(\mathsf{F})$, is the fragment of UB where X is not allowed.

All these logics can be extended so that boolean combinations of path formulae are allowed. [Eme90] denotes them by $\mathrm{B}(\ldots, \wedge, \neg)$, so that ECTL$^+$ really is $\mathrm{B}(\mathsf{U}, \mathsf{X}, \overset{\infty}{\mathsf{F}}, \wedge, \neg)$. We let CTL$^+$, UB$^+$, BTL$^+$ denote the logics obtained by extending CTL, UB and BTL in the corresponding way. It is well known [EH85,EH86] that we have the following hierarchy:

$$\mathrm{BTL} \begin{array}{ccc} < & \mathrm{UB} & < \\ < & \mathrm{BTL}^+ & < \end{array} \mathrm{UB}^+ < \mathrm{CTL} = \mathrm{CTL}^+ < \mathrm{ECTL} < \mathrm{ECTL}^+$$

where $L < L'$ means that L' is strictly more expressive than L, and $L = L'$ means that L and L' have the same expressive power.

2.4 CTL with fairness

ECTL^+ is not the only logic where one can mix CTL formulae with fairness constraints, but other proposals can all be seen as fragments of ECTL^+:

- GFCTL [EL87] is CTL where every path quantifier is indexed with a fairness constraint. One write $\mathsf{E}_\Phi \varphi_p$ to state that there exists a fair path satisfying φ_p. The fairness constraint Φ can be any boolean combination of $\overset{\infty}{\mathsf{F}}\ \varphi_i$'s where the φ_i are state formulae. E.g. $\mathsf{E}_{(\overset{\infty}{\mathsf{F}} A \wedge \overset{\infty}{\mathsf{G}} \mathsf{EX} B)} C \mathsf{U} D$ is a GFCTL formula.
- FCTL [EL87] is GFCTL where the fairness constraint Φ is restricted to boolean combinations of $\overset{\infty}{\mathsf{F}}\ \pm A_i$ for atomic propositions A_is, and where Φ is the same for all occurrences of a path quantifier. Then it is more customary to see a FCTL formula as a pair (φ_s, Φ) of a CTL path formula and a global fairness constraint.
- $\mathrm{CTL}^{\mathrm{F}}$ [CES86] is FCTL where the fairness constraint Φ is further restricted to a conjunctive $\bigwedge_i (\overset{\infty}{\mathsf{F}} \bigvee_j \pm A_{i,j})$.

2.5 Complexity of model checking

The *model checking problem* for a temporal logic L is to decide, given a KS S with distinguished node q_0, and a (state) formula $\varphi \in L$, whether $q_0 \models_S \varphi$. Model checking temporal logics has many practical applications [Eme90,McM93,CGP99] and it is important to be able to classify the most common temporal logics according to the computational complexity of their model checking problems.

Model checking for CTL and CTL^* is known to be P-complete and PSPACE-complete respectively. Model checking for ECTL and $\mathrm{CTL}^{\mathrm{F}}$ is P-complete too. For logics like CTL^+, FCTL, and ECTL^+, the exact complexity is not known. It has been observed [CES86, Theo. 6.2] that for CTL^+ the problem is NP-hard and coNP-hard and is in Δ_2^p (and thus believed to be easier than PSPACE-complete problems). The same applies to FCTL and GFCTL despite the wrong claim that model checking is NP-complete for FCTL [EL87, Coro. 4.8].

3 SNSAT and Δ_2^p-complete problems

Δ_2^p is the class P^{NP}, i.e. the class of problems solvable by a deterministic polynomial-time Turing machine querying an NP set oracle [Sto76]. This class is above NP and coNP in the polynomial-time hierarchy.

The class of problems complete for Δ_2^p does not contain many natural examples [Pap84,Kre88,Wag87]. In fact, in the polynomial-time hierarchy, it is easier to come up with problems complete for the Σ_k^p or Π_k^p levels than for the Δ_k^p levels.

In this paper we introduce SNSAT (for *sequentially nested* satisfiability), a logical problem with nested satisfiability questions, that is a convenient basis for our reducibility proof.

Definition 3.1. *An instance $\mathcal{I}$ of SNSAT is given by a set $X = \{x_1, \ldots, x_n\}$ of boolean variables together with a list $\mathcal{L}$ of equivalences*

$$\begin{aligned} x_1 &:\Leftrightarrow \exists Z_1\ F_1(Z_1), \\ x_2 &:\Leftrightarrow \exists Z_2\ F_2(x_1, Z_2), \\ &\vdots \\ x_n &:\Leftrightarrow \exists Z_n\ F_n(x_1, \ldots, x_{n-1}, Z_n), \end{aligned}$$

where, for $i = 1, \ldots, n$, Z_i is a set $\{z_i^1, \ldots, z_i^{p_i}\}$ of boolean variables, and F_i is a boolean formula with variables among $Z_i \cup \{x_1, \ldots, x_{i-1}\}$.

Note that in $\mathcal{I}$ the sets X, Z_1, ..., and Z_n are pairwise disjoint. We write $Z = \{z_1, \ldots, z_p\}$ for $Z_1 \cup \cdots \cup Z_n$, and $\mathit{Var} = \{u, \ldots\}$ for $X \cup Z$.

The equivalences $\mathcal{L}$ in $\mathcal{I}$ define a unique valuation $v_\mathcal{I}$ of the variables in X:

$$v_\mathcal{I}(x_i) = \top \overset{\text{def}}{\Leftrightarrow} F_i(v_\mathcal{I}(x_1), \ldots, v_\mathcal{I}(x_{i-1}), Z_i) \text{ is satisfiable.} \tag{1}$$

Observe that there exists a simple algorithm in Δ_2^p that computes $v_\mathcal{I}$ one value at a time. When $v_\mathcal{I}$ is known over $\{x_1, \ldots, x_{i-1}\}$, the value of $v_\mathcal{I}(x_i)$ is computed by solving a boolean satisfiability problem, "*is F_i satisfiable with the given values of $x_1, \ldots, x_{i-1}$?*", for which a SAT oracle is sufficient.

The computational problem called SNSAT is, given an instance $\mathcal{I}$ as above, to decide whether $v_\mathcal{I}(x_n) = \top$ (in which case we say $\mathcal{I}$ is a positive instance).

Theorem 3.2. *SNSAT is Δ_2^p-complete.*

Proof. Membership in Δ_2^p has been explained above. Δ_2^p-hardness of SNSAT is shown incidentally in [Got95, proof of Theorem 3.4] where SNSAT is not identified as an interesting subproblem. (Alternatively, there are simple direct reductions from our SNSAT to the DSAT problem of [Pap84] and vice versa, but explaining DSAT requires a lot of notations.) □

The equivalences in $\mathcal{I}$ can be seen as a large satisfiability problem where we have to find correct values for the boolean variables in Z, aiming at satisfying the F_i's as much as possible, while respecting the values of the x_i's across equivalences. With this in mind, we say a valuation w of Var is:
safe: if, for all $i = 1, \ldots, n$, $w(x_i)$ implies $F_i(v_\mathcal{I}(x_1), \ldots, v_\mathcal{I}(x_{i-1}), w(Z_i))$,
correct: if, for all $i = 1, \ldots, n$, $w(x_i) = F_i(v_\mathcal{I}(x_1), \ldots, v_\mathcal{I}(x_{i-1}), w(Z_i))$,
admissible: if w is correct and coincide with $v_\mathcal{I}$ over X.
Thus a safe valuation only assigns positive values to some x_i if this is consistent with the values given to $x_1, \ldots, x_{i-1}$ and the variables in Z_i. A correct valuation is safe and is also consistent for negative values assigned to some x_i. Still, there is no guarantee that the values of variables in Z are best possible. An arbitrary valuation over Z extends into a correct valuation in a unique way, and checking that a given w is correct can be done in polynomial-time.

An admissible valuation is just a valuation for Z that yields $v_\mathcal{I}$ for X. Hence it is optimal over Z. Clearly, admissible valuations exist for any SNSAT instance, positive or negative, but checking that a given w is admissible is Δ_2^p-complete.

4 Hardness of model checking CTL$^+$

In this section we show that there exists a logspace transformation from SNSAT into model checking for BTL$^+$. Aiming at improved clarity, we proceed in two steps: first we give a reduction of SNSAT to a model checking problem for CTL$^+$, then we adapt the construction and obtain a model checking problem for BTL$^+$.

From now on we assume that we are given an instance $\mathcal{I}$ of SNSAT with the notations of Def. 3.1. W.l.o.g. we assume that every F_i is a CNF, i.e. a conjunction of disjunctions of literals, and write F_i under the form $\bigwedge_l \bigvee_m \alpha_{i,l,m}$ where $\alpha_{i,l,m}$ is a literal $\pm u$ built with a variable u from $Z_i \cup \{x_1, \ldots, x_{i-1}\}$.

With $\mathcal{I}$ we associate a Kripke structure $S_{\mathcal{I}}$ and a CTL$^+$ formula $\Phi_{\mathcal{I}}$ s.t. $v_{\mathcal{I}}(x_n) = \top$ iff $S_{\mathcal{I}} \models \Phi_{\mathcal{I}}$ (see Coro 4.2). Figure 2 depicts $S_{\mathcal{I}}$. As shown in Fig. 2,

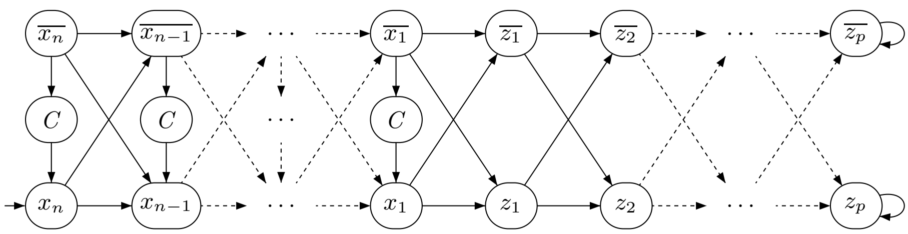

Fig. 2. Kripke structure $S_{\mathcal{I}}$ associated with SNSAT instance $\mathcal{I}$

the nodes of $S_{\mathcal{I}}$ are of two kinds: (1) one node per literal u and $\overline{u}$ with $u \in \mathit{Var}$, and (2) one C-node between a $\overline{x_i}$-node and the corresponding x_i-node.

The nodes are labeled with propositions taken from $\{C\} \cup \{P_\alpha \mid \alpha \text{ a literal}\}$. The labeling is given by Fig. 2 where we shortly wrote α for P_α. Below we sometimes call α the literal-node labeled by P_α.

The transitions of $S_{\mathcal{I}}$ are of two kinds: (1) transitions from a literal $\pm u$ to a literal $\pm u'$ if u' immediately follows u in the left-to-rigth sequence x_n, x_{n-1}, $\ldots$, x_2, x_1, z_1, z_2, $\ldots$, z_p, (2) transitions from $\overline{x_i}$ to the ith C-node, and from there to x_i. Additionally, two self-loops on the $\pm z_p$-nodes ensure that the transition relation is total.

The structure of $S_{\mathcal{I}}$ is such that a path π from $\pm x_n$ that never visits a C-node visits exactly one literal for every $u \in \mathit{Var}$ so that there is a valuation w_π associated with π in the obvious way. Reciprocally, we can associate a path π_w with any valuation w in such a way that π_w starts from x_n or $\overline{x_n}$ (depending on $w(x_n)$) and never visits a C-node.

Furthermore, some properties of w can be stated as temporal properties of π_w: if $\pi \models \mathsf{G}\neg c$ then w_π is defined, and then w_π is safe iff $\pi \models \bigwedge_{i=1}^{n}\Big[\big(\mathsf{F}\,P_{x_i}\big) \Rightarrow \bigwedge_l \bigvee_m \mathsf{F}\,P_{\alpha_{i,l,m}}\Big]$.

We are now ready for the main technical difficulties: we define a sequence $\varphi_0, \varphi_1, \varphi_2, \ldots$ of CTL$^+$ formulae by $\varphi_0 \stackrel{\text{def}}{=} \top$ and, for $k > 0$,

$$\varphi_k \stackrel{\text{def}}{=} \mathsf{E}\left[\begin{array}{c} \mathsf{G}\Big[\big(P_{\overline{x_1}} \vee \cdots \vee P_{\overline{x_n}}\big) \Rightarrow \mathsf{EX}\big(C \wedge \mathsf{EX}(\neg\varphi_{k-1})\big)\Big] \\ \wedge\ \mathsf{G}\neg C \ \wedge\ \bigwedge_{i=1}^{n}\Big[\big(\mathsf{F}\,P_{x_i}\big) \Rightarrow \bigwedge_l \bigvee_m \mathsf{F}\,P_{\alpha_{i,l,m}}\Big] \end{array}\right].$$

Thus φ_k has the form $\mathsf{E}[\psi_{k-1} \wedge \mathsf{G}\neg C \wedge \rho]$ where ψ_{k-1} and ρ are complex path formulae, and where $\mathsf{G}\neg C \wedge \rho$ was used above to state that w_π is safe.

The next lemma states how φ_k is satisfied in nodes x_i and $\overline{x_i}$ of $S_{\mathcal{I}}$, justifying the whole construction:

Lemma 4.1 (Correctness of the reduction). *For $k \in \mathbb{N}$ and $r = 1, \ldots, n$:*
(a) if $k \geq 2r-1$ then $\big(v_{\mathcal{I}}(x_r) = \top$ iff $x_r \models \varphi_k\big)$,
(b) if $k \geq 2r$ then $\big(v_{\mathcal{I}}(x_r) = \bot$ iff $\overline{x_r} \models \varphi_k\big)$.

Proof. By induction on k. The case $k = 0$ holds vacuously. We now assume that $k > 0$ and that Lemma 4.1 holds for $k-1$.
i. We prove the "$\Rightarrow$" direction of both "iff"s:

Let w be an admissible valuation and π be the suffix of π_w that starts from x_r (or $\overline{x_r}$, depending on the value of $w(x_r)$). We claim that if $k \geq 2r-1$ (resp. $k \geq 2r$) then π is a witness for $x_r \models \varphi_k$ (resp. for $\overline{x_r} \models \varphi_k$). Clearly $\pi \models \mathsf{G}\neg C$ and $\pi \models \rho$ (because w is admissible) so that we only have to show $\pi \models \psi_{k-1}$, for which the $\overline{x_i}$ nodes must be checked. Now, whenever π visits a $\overline{x_i}$ for some $1 \leq i \leq r$, we have $v_{\mathcal{I}}(x_i) = \bot$ because w is admissible. We know $k \geq 2i$: if $i = r$ then we are proving the (b) part and $k \geq 2r$, and otherwise $i < r$. Hence $k-1 \geq 2i-1$ and the ind. hyp. entails $x_i \not\models \varphi_{k-1}$ so that $\overline{x_i} \models \mathsf{EX}(C \wedge \mathsf{EX}(\neg\varphi_{k-1}))$.

ii. We now prove the "$\Leftarrow$" direction of both "iff"s:

Assume $k \geq 2r-1$ and $x_r \models \varphi_k$ (or $k \geq 2r$ and $\overline{x_r} \models \varphi_k$). Thus there is a path π from x_r (resp. from $\overline{x_r}$) s.t. $\pi \models \psi_{k-1} \wedge \mathsf{G}\neg C \wedge \rho$. We claim that the valuation w_π induced by π is such that $w_\pi(x_i) = v_{\mathcal{I}}(x_i)$ for $i = 1, \ldots, r$, and prove this by induction on i. There are two cases:
(1) if $w_\pi(x_i) = \top$ then $\bigwedge_l \bigvee_m w(\alpha_{i,l,m}) = \top$ since $\pi \models \rho$, i.e. w is safe. Thus, by ind. hyp., $F_i(v_{\mathcal{I}}(x_1), \ldots, v_{\mathcal{I}}(x_{i-1}), w_\pi(Z_i)) = \top$ so that $v_{\mathcal{I}}(x_i) = \top$.
(2) if $w_\pi(x_i) = \bot$ then $\overline{x_i} \models \mathsf{EX}(C \wedge \mathsf{EX}(\neg\varphi_{k-1}))$ since $\pi \models \psi_{k-1}$ and thus $x_i \not\models \varphi_{k-1}$. Now if $i < r$, we have $k-1 \geq 2i-1$ and, by ind. hyp., $v_{\mathcal{I}}(x_i) = \bot$. If $i = r$ we must be in the case where $k \geq 2r$ and $\overline{x_r} \models \varphi_k$, so that $k-1 \geq 2i-1$ and again $v_{\mathcal{I}}(x_i) = \bot$ by ind. hyp. $\square$

With Lemma 4.1, we get:

Corollary 4.2. *For any instance $\mathcal{I}$ of SNSAT, $v_{\mathcal{I}}(x_n) = \top$ iff $x_n \models_{S_{\mathcal{I}}} \varphi_{2n-1}$.*

The size of φ_{2n-1} is in $O(n \times |\mathcal{I}|)$. Since $S_{\mathcal{I}}$ and φ_{2n-1} can be built in logspace from $\mathcal{I}$, Coro. 4.2 effectively provides a transformation from SNSAT into model checking for CTL$^+$ (in fact, for UB$^+$), proving model checking for CTL$^+$ is Δ_2^p-hard.

The definition of the φ_k's uses EX and AX (in the ψ_{k-1} part) with the consequence that φ_k is a UB$^+$ and not a BTL$^+$ formula. However, a similar albeit clumsier construction can be given, proving the following

Theorem 4.3. *Model checking for* BTL$^+$ *is Δ_2^p-hard.*

Proof (Idea). We define a structure $S'_{\mathcal{I}}$ by modifying $S_{\mathcal{I}}$: Fig. 3 shows how so-called *stop nodes*, labeled with s, are inserted in $S_{\mathcal{I}}$, and how self-loops are added on the x_i-nodes.

We also modify the definition of φ_k by replacing $\mathsf{EX}(C \wedge \mathsf{EX}(\neg\varphi_{k-1}))$ in the ψ_{k-1} part with

$$\mathsf{E}\Big[\neg \mathsf{F} s \;\wedge\; \mathsf{F}\Big((P_{x_1} \vee \ldots \vee P_{x_n}) \;\wedge\; \neg\varphi_{k-1}\Big)\Big].$$

This gives BTL$^+$ formulae for which we can prove Lemma 4.1 adapted to $S'_{\mathcal{I}}$. □

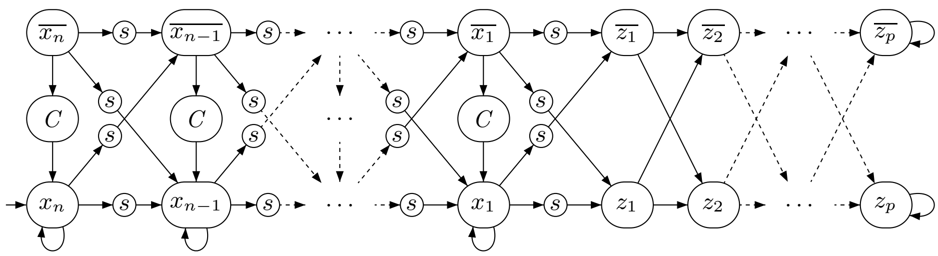

Fig. 3. $S'_{\mathcal{I}}$, a variant of $S_{\mathcal{I}}$ with *stop nodes*

5 Hardness of model checking FCTL

The ideas underlying the construction of $S_{\mathcal{I}}$ can be adapted in order to show Δ_2^p-hardness of model-checking for FCTL. Figure 4 describes $S''_{\mathcal{I}}$. One sees that, because of the outermost loop, an infinite path π in $S''_{\mathcal{I}}$ may visit both u and $\overline{u}$ for any variable u (even if it never visits a C-node), so that there is no direct corresponding valuation w_π: we need more assumptions over paths.

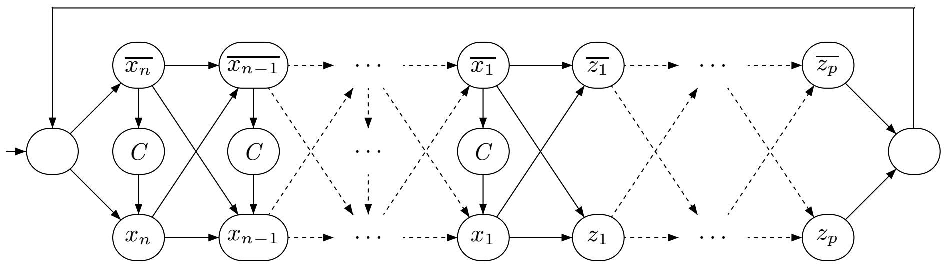

Fig. 4. $S''_{\mathcal{I}}$, a variant of $S_{\mathcal{I}}$ with outermost loop

Consider the following fairness constraint:

$$\Phi \stackrel{\text{def}}{=} \bigwedge_{u \in Var} \Big(\overset{\infty}{\mathsf{G}} \neg P_u \vee \overset{\infty}{\mathsf{G}} \neg P_{\overline{u}} \Big).$$

Now an infinite path π that verifies Φ defines a natural valuation: there exists a suffix π' of π s.t. for any $u \in Var$, π' never visits u (and then π' visits only $\overline{u}$) or π' never visits $\overline{u}$ (and then π' visits only u). Note that $\mathsf{G}\neg C$ holds for π'. Reciprocally, with any valuation w, we can associate a path π_w satisfying Φ in such a way that π_w visits infinitely often u or $\overline{u}$ depending on $w(u)$.

We now define FCTL formulae inspired by the φ_ks from the previous section. First we define ECTL$^+$ formulae ξ_k by $\xi_0 \stackrel{\text{def}}{=} \top$ and, for $k > 0$,

$$\xi_k \stackrel{\text{def}}{=} \mathsf{E} \left[\begin{array}{l} \mathsf{G}\Big[\big(P_{\overline{x_1}} \vee \cdots \vee P_{\overline{x_n}}\big) \Rightarrow \mathsf{EX}\big(C \wedge \mathsf{EX}(\neg \xi_{k-1})\big)\Big] \\ \wedge \bigwedge_{i=1}^{n} \Big[\big(\overset{\infty}{\mathsf{F}}\ P_{x_i}\big) \Rightarrow \bigwedge_l \bigvee_m \overset{\infty}{\mathsf{F}}\ P_{\alpha_{i,l,m}}\Big] \wedge \bigwedge_{u \in Var} \Big[\overset{\infty}{\mathsf{G}} \neg P_u \vee \overset{\infty}{\mathsf{G}} \neg P_{\overline{u}}\Big] \end{array} \right].$$

Note that ξ_k has the form $\mathsf{E}[\chi_{k-1} \wedge \rho' \wedge \Phi]$ where ρ' and Φ are fairness constraints which are used in any ξ_k. Clearly, in $S''_{\mathcal{I}}$ one can prove a variant of Lemma 4.1 for the ξ_ks, but the ξ_ks are not FCTL formulae.

Now observe that in $S''_{\mathcal{I}}$ the subformula $\mathsf{EX}(C \wedge \mathsf{EX}(\neg \xi_{k-1}))$ is equivalent to the following formula

$$\mathsf{E}\Big[\Phi \wedge \rho' \wedge \mathsf{X}\Big(C \wedge \mathsf{E}(\Phi \wedge \rho' \wedge \mathsf{X}(\neg \xi_{k-1}))\Big)\Big]$$

where we inserted the fairness constraint $\Phi \wedge \rho'$ under the two path quantifiers. The equivalence holds because, from any node in $S''_{\mathcal{I}}$, there exists an infinite fair path (it is sufficient to visit only $\overline{u}$ nodes).

We now have a variant of the ξ_ks where the same simple fairness constraint is used everywhere, that is, we have a FCTL formula! Formally, we define φ'_k by $\varphi'_0 \stackrel{\text{def}}{=} \top$ and $\varphi'_k \stackrel{\text{def}}{=} \mathsf{EG}\Big[\big(P_{\overline{x_1}} \vee \cdots \vee P_{\overline{x_n}}\big) \Rightarrow \mathsf{EX}\big(C \wedge \mathsf{EX}(\neg \varphi'_{k-1})\big)\Big]$ and we couple this CTL formula with the fairness constraint $(\Phi \wedge \rho')$. Following the notations of section 2.4 we have:

$$\eta_k \stackrel{\text{def}}{=} \begin{cases} \varphi_s : \mathsf{EG}\Big[\big(P_{\overline{x_1}} \vee \cdots \vee P_{\overline{x_n}}\big) \Rightarrow \mathsf{EX}\big(C \wedge \mathsf{EX}(\neg\varphi'_{k-1})\big)\Big], \\ \Phi : \bigwedge_{u \in Var} \Big[\overset{\infty}{\mathsf{G}} \neg P_u \vee \overset{\infty}{\mathsf{G}} \neg P_{\overline{u}}\Big] \wedge \bigwedge_{i=1}^{n} \Big[\big(\overset{\infty}{\mathsf{F}}\ P_{x_i}\big) \Rightarrow \bigwedge_{l} \bigvee_{m} \overset{\infty}{\mathsf{F}}\ P_{\alpha_{i,l,m}}\Big]. \end{cases}$$

Now, Lemma 4.1 and corollary 4.2 can be reformulated for $S''_{\mathcal{I}}$ and η_k, proving the Δ_2^p-harness of FCTL model checking:

Theorem 5.1. *Model checking for* FCTL *is Δ_2^p-hard.*

6 Upper bounds

In this section we show that model checking for ECTL^+ is in Δ_2^p. This is a slight extension of the corresponding result for CTL^+ (a result not widely known).

A path $\pi = q_0 q_1 \ldots$ (in some KS S) is *ultimately periodic* if there exist $m, k \in \mathbb{N}$ ($k > 0$) s.t. $q_{i+k} = q_i$ for all $i \geq m$. Then π is written under the form $q_0 \ldots q_{m-1}(q_m \ldots q_{m+k-1})^\omega$ and we say π has size $m + k$.

A path π' is *extracted from* π if it has the form $\pi' = q_{i_0} \ldots q_{i_{p-1}}(q_{i_p} \ldots q_{i_s})^\omega$ where the sequence $i_0, i_1, \ldots$ is such that

$$0 \leq i_0 < i_1 < \ldots < i_{p-1} \leq m-1 < i_p < i_{p+1} < \ldots < i_s \leq m+k-1.$$

Let φ be an ECTL^+ formula of the form $\mathsf{E}\varphi_p$ where φ_p is *flat*, i.e. does not contain any path quantifier. The *principal subformulae* of φ_p are all subformulae of the form $\psi_1 \mathsf{U} \psi_2$ or $\overset{\infty}{\mathsf{F}}\ \psi$ or $\mathsf{X}\psi$, i.e. subformulae that have a modality at their root.

With $\pi = q_0 \ldots q_{m-1}(q_m \ldots q_{m+k-1})^\omega$ and φ_p we associate a set $w(\pi, \varphi_p) \subseteq \{0, 1, \ldots, m+k-1\}$ of *witness positions* along π: $w(\pi, \varphi_p)$ has one (or sometimes zero) position for every principal subformula of φ_p. Specifically:
- if $\mathsf{X}\psi$ is a principal subformula, then the witness position is 1,
- if $\overset{\infty}{\mathsf{F}}\ \psi$ is a principal subformula, then there is a witness position only if $\pi \models \overset{\infty}{\mathsf{F}}\ \psi$ and it is the first $i \geq m$ s.t. that $q_i \models \psi$,
- if $\psi_1 \mathsf{U} \psi_2$ is a principal subformula, then there are three cases: if $\pi \models \psi_1 \mathsf{U} \psi_2$, then the witness position is the first $i \geq 0$ s.t. $q_i \models \psi_2$, if $\pi \not\models \psi_1 \mathsf{U} \psi_2$ and $\pi \models \mathsf{F}\psi_2$, then it is the first $i \geq 0$ s.t. $q_i \models \neg(\psi_1 \wedge \psi_2)$, if $\pi \not\models \mathsf{F}\psi_2$, then there is no witness position for this subformula.

Lemma 6.1. *Assume $\pi' = q_{i_0} q_{i_1} \ldots$ is an ultimately periodic path extracted from π, with $i_0 = 0$ and such that $w(\pi, \varphi_p) \subseteq \{i_0, i_1, \ldots, i_s\}$. Then $\pi' \models \varphi_p$ iff $\pi \models \varphi_p$.*

Proof. By construction π' agrees with π on all principal subformulae, then on all subformulae, of φ_p. □

Lemma 6.2 (Small witnesses for ECTL$^+$). *Let S be a Kripke structure with n nodes, and $\mathsf{E}\varphi_p$ be a* ECTL$^+$ *formula where φ_p is flat. Then if $S \models \mathsf{E}\varphi_p$, there is a path $\pi \in \Pi(q_0)$ satisfying φ_p that is ultimately periodic and has size in $O(n \times |\varphi_p|)$.*

Proof. Assume $S \models \mathsf{E}\varphi_p$. Since φ_p is a PLTL formula, it is known (e.g. [SC85]) that there exists an ultimately periodic $\pi \in \Pi(q_0)$ s.t. $\pi \models \varphi_p$. Now we extract from π an ultimately periodic π' by keeping only positions in $w(\pi, \varphi_p)$ and the smallest number of intermediary positions that are required to ensure connectivity between the positions from $w(\pi, \varphi_p)$ (i.e. we want π' to be a path in S). Since $w(\pi, \varphi_p)$ has $O(|\varphi_p|)$ positions and since at most $n-1$ intermediary positions are required between any two positions in $w(\pi, \varphi_p)$, the size of π' is in $O(n \times |\varphi_p|)$. Finally, $\pi' \models \varphi_p$ by Lemma 6.1. □

The corollary is that there is an NP-algorithm for model checking ECTL$^+$ formulae of the form $\mathsf{E}\varphi_p$ with flat φ_p: one non-deterministically guesses an ultimately periodic π path of size $O(n \times |\varphi_p|)$ and then checks $\pi \models \varphi_p$ in time $O(n \times |\varphi_p|)$, e.g. seeing π as a deterministic Kripke structure on which φ_p can be read as a CTL formula.

Now, for model checking non-flat ECTL$^+$ formulae, we can use the general algorithm given in [EL87, Section 6] for branching-time logics of the form B(L(...)), i.e., logics obtained by adding path quantifiers to linear-time logics L(...). This algorithm is a simple polynomial-time procedure calling an oracle for model checking L(...). In the case of ECTL$^+$, we end with a P$^{\text{NP}}$ algorithm, hence

Theorem 6.3. *Model checking for* ECTL$^+$ *is in Δ_2^p.*

7 Conclusions

Combining Theorems 4.3, 5.1 and 6.3, we obtain

Theorem 7.1. *The model checking problems for* BTL$^+$*,* UB$^+$*,* CTL$^+$*,* FCTL*,* GFCTL *and* ECTL$^+$ *are all Δ_2^p-complete.*

We also deduce

Theorem 7.2. *The model checking problem for* BT* *is Δ_2^p-complete.*

where BT* is the fragment of CTL* where F is the only allowed temporal modality (U and X are forbidden, G and $\overset{\infty}{\mathsf{F}}$ are allowed since they can be written with F).

Proof (of Theo. 7.2). Since BT* contains BTL$^+$, model checking BT* is Δ_2^p-hard. Since model checking flat $\mathsf{E}\varphi_p$ formulae is in NP when φ_p is in L(F) [SC85,DS98], a reasoning similar to the proof of Theo. 6.3 shows membership in Δ_2^p (already indicated in [CES86]). □

8 Acknowledgments

We thank A. Rabinovich for pointing to us that the complexity of model checking CTL^+ was not precisely characterized, and the anonymous referee who suggested that we look at FCTL.

References

[BPM83] M. Ben-Ari, A. Pnueli, and Z. Manna. The temporal logic of branching time. *Acta Informatica*, 20:207–226, 1983.

[CE81] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proc. Logics of Programs Workshop, Yorktown Heights, New York, May 1981*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer, 1981.

[CES86] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[CGP99] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 1999.

[DS98] S. Demri and Ph. Schnoebelen. The complexity of propositional linear temporal logics in simple cases (extended abstract). In *Proc. 15th Ann. Symp. Theoretical Aspects of Computer Science (STACS '98), Paris, France, Feb. 1998*, volume 1373 of *Lecture Notes in Computer Science*, pages 61–72. Springer, 1998. Journal version as [DS02].

[DS02] S. Demri and Ph. Schnoebelen. The complexity of propositional linear temporal logics in simple cases. *Information and Computation*, 174(1):84–103, 2002.

[EC80] E. A. Emerson and E. M. Clarke. Characterizing correctness properties of parallel programs using fixpoints. In *Proc. 7th Coll. Automata, Languages and Programming (ICALP '80), Noordwijkerhout, NL, Jul. 1980*, volume 85 of *Lecture Notes in Computer Science*, pages 169–181. Springer, 1980.

[EH83] E. A. Emerson and J. Y. Halpern. “Sometimes” and “Not Never” revisited: On branching versus linear time. In *Proc. 10th ACM Symp. Principles of Programming Languages (POPL '83), Austin, TX, USA, Jan. 1983*, pages 127–140, 1983.

[EH85] E. A. Emerson and J. Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *Journal of Computer and System Sciences*, 30(1):1–24, 1985.

[EH86] 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.

[EL87] E. A. Emerson and Chin-Laung Lei. Modalities for model checking: Branching time logic strikes back. *Science of Computer Programming*, 8(3):275–306, 1987.

[Eme90] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 16, pages 995–1072. Elsevier Science, 1990.

[ES89] E. A. Emerson and J. Srinivasan. Branching time temporal logic. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, Proc. REX School/Workshop, Noordwijkerhout, NL, May-June 1988*, volume 354 of *Lecture Notes in Computer Science*, pages 123–172. Springer, 1989.

[Got95] G. Gottlob. NP trees and Carnap's modal logic. *Journal of the ACM*, 42(2):421–457, 1995.

[Kre88] M. W. Krentel. The complexity of optimization problems. *Journal of Computer and System Sciences*, 36(3):490–509, 1988.

[Lam80] L. Lamport. "Sometimes" is sometimes "Not Never". In *Proc. 7th ACM Symp. Principles of Programming Languages (POPL '80), Las Vegas, NV, USA, Jan. 1980*, pages 174–185, 1980.

[McM93] K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic, 1993.

[ON80] H. Ono and A. Nakamura. On the size of refutation Kripke models for some linear modal and tense logics. *Studia Logica*, 39(4):325–333, 1980.

[Pap84] C. H. Papadimitriou. On the complexity of unique solutions. *Journal of the ACM*, 31(2):392–400, 1984.

[Pnu77] A. Pnueli. The temporal logic of programs. In *Proc. 18th IEEE Symp. Foundations of Computer Science (FOCS '77), Providence, RI, USA, Oct.-Nov. 1977*, pages 46–57, 1977.

[RS00] A. Rabinovich and Ph. Schnoebelen. BTL_2 and expressive completeness for $ECTL^+$. Research Report LSV-00-8, Lab. Specification and Verification, ENS de Cachan, Cachan, France, October 2000. 21 pages.

[SC85] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985.

[Sto76] L. J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976.

[Wag87] K. W. Wagner. More complicated questions about maxima and minima, and some closures of NP. *Theoretical Computer Science*, 51(1–2):53–80, 1987.

[Wil99] T. Wilke. CTL^+ is exponentially more succint than CTL. In *Proc. 19th Conf. Found. of Software Technology and Theor. Comp. Sci. (FST&TCS '99), Chennai, India, Dec. 1999*, volume 1738 of *Lecture Notes in Computer Science*, pages 110–121. Springer, 1999.