Buy One, Get One Free !!![*]

Orna Bernholtz and Orna Grumberg
Department of Computer Science
The Technion
Haifa 32000, Israel
orna@cs.technion.ac.il

Abstract

The exponential gap between CTL and LTL model-checking complexity led to a development of model-checking tools for CTL, while model checkers for LTL have lagged behind. However, users of these tools have to struggle with the limited expressive power of CTL and are often compelled to give up checking many important behaviors. As a matter of course, finding specification languages which are strictly more expressive than CTL and yet maintain its attractive model-checking complexity is a challenging problem and has been an active area of research. In this paper we introduce such a language.

Our language, CTL^2, is an outcome of a new approach for defining sub-languages of $CTL^\star$. The approach allows a bounded number of linear-time operators within the path formulas of $CTL^\star$. We discuss the expressive power of CTL^2 and focus on the relation between CTL^2 and CTL. We show that beyond the increase in the expressive power, a substantial advantage of CTL^2 is the neat and intuitive presentation it provides for specifications whose CTL equivalences are complicated and very hard to understand. We introduce a model-checking procedure for CTL^2. Our model checker is of complexity linear in both the formula and the structure being checked, exactly as the one for CTL. In addition, we suggest an extension of it that, preserving its complexity, handles fairness.

1 Introduction

Propositional temporal logics, which are propositional modal logics that enable the description of occurrence of events in time, serve as a classical tool for verifying concurrent programs [Pnu81]. Two possible views regarding the nature of time induce two types of temporal logics. In linear temporal logics, time is treated as if each moment in time has a unique possible future, while in branching temporal logics, each moment in time may split into various possible futures. Lamport, in [Lam80], was the first to consider this issue. He examined two distinct interpretations of a temporal logic which consists of the temporal operators "always" and "sometimes". The first interpretation, in which the formulas are interpreted over paths, is associated with linear time, and the second, in which formulas are interpreted over states, is associated with branching time. The expressive power of these logics, as Lamport showed, is incomparable.

[*]A preliminary version appears in [BG94].

Clarke and Emerson took this one step forward. In [CE81], they defined the full branching temporal logic CTL* which includes both of Lamport's interpretations and allows the specification of both linear-time and branching-time properties. A comprehensive discussion of the expressive power of branching versus linear temporal logics can be found in [EH86]. Emerson and Halpern reexamined Lamport's approach, improved his incomparability result, and presented a hierarchy of the relative expressive power of some sub-languages of CTL*. They claimed that, on the one hand, linear temporal logics are suitable for reasoning about concurrent programs where interest is naturally restricted to properties that hold along all computation paths, whereas on the other hand, branching temporal logics have a relatively low model-checking complexity and enable specifying existential properties. They conclude by noting that "one should use the subset of CTL* most appropriate to the application", where appropriateness is measured by both expressiveness and complexity.

In temporal logic model checking, a given model (representing a program) is checked with respect to a propositional temporal logic formula (specifying a required behavior for the program). Model-checking procedures suggest a computerized mechanism for verifying finite state systems such as communication protocols and hardware designs. Recent methods and heuristics such as BDDs [Bry86, BCM$^+$92], modular model checking [GL91], partial order methods [WG93], and others, cope successfully with the known "state explosion" problem and give rise to model checking not only as a lovely theoretical issue, but also as a practical tool used for formal verification. As far as model-checking complexity is concerned, the following results are known: The model-checking problem for the branching temporal logic CTL is in deterministic linear time [EC82, CES86], and the model-checking problem for the linear temporal logic LTL, as well as the one for CTL*, is PSPACE-complete [SC85, CES86].

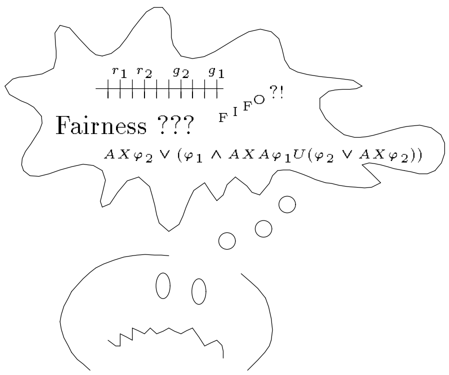
The exponential gap between CTL and LTL model-checking complexity led to a development of model checking tools for CTL [Bro86, McM93], while model checkers for LTL have lagged behind. However, users of these tools have to struggle with the limited expressive power of CTL. Unfortunately, the branching nature of CTL compels them to give up checking many important behaviors. As a matter of course, finding specification languages which are strictly more expressive than CTL and yet maintain its attractive model-checking complexity is a challenging problem and has been an active area of research [Eme90]. In this paper we introduce such a language.

Our language, CTL^2, is an outcome of a new approach for defining sub-languages of CTL*. The syntax of CTL* allows path quantifiers to be applied to path formulas composed of an arbitrary combination of linear-time operators. CTL syntax restricts path quantifiers to be applied to path formulas with a single linear-time operator. The idea behind our approach is to allow a bounded number of linear-time operators within the path formulas of CTL*. In CTL^2, we investigate the simplest version of this extension: path formulas may contain an assertion composed of two, possibly negated, linear-time operators, either nested or connected by a binary boolean operator. Thus, formulas like $AGF grant$, $E(X req) U grant$, $A(F \neg busy \vee G wait)$, and $E(\neg(busy\ U\ req)) U grant$, are all CTL^2 formulas. Indeed, CTL^2 is strong enough to neatly express both liveness and safety properties used for reasoning about concurrent programs, such as unconditional fairness, preservation of FIFO order, more accurate specification of accessibility, absence of unsolicited response, and others.

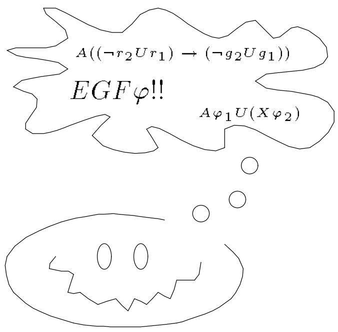
We compare the expressive power of CTL^2 with other branching temporal logics, linear temporal logics, and automata over infinite trees. We focus on the relation between CTL^2 and CTL. The CTL^2 formula $EGFp$, which is not expressible in CTL [EH86], testifies that CTL^2 is strictly more expressive than CTL. Yet, surprisingly, the increase in the expressive power is not as significant as we might expect. We characterize precisely CTL^2 formulas that have equivalences of CTL and show that beyond the increase in the expressive power, a substantial advantage of CTL^2 is the clear and intuitive presentation it provides for specifications which have clumsy and unreadable equivalences of CTL.

We introduce a model-checking procedure for CTL^2. Given a CTL^2 formula ψ and a structure K, the procedure determines for every state in K whether it satisfies ψ. As known model checkers for CTL, our procedure iteratively labels K with a set of formulas whose satisfaction has to do with the satisfaction of ψ. Unlike in the case of CTL, these formulas are not necessarily sub-formulas of ψ. However, their number is linear in $|\psi|$ and labelling K with each of them is of complexity linear in $|K|$. Thus, model checking of CTL^2 is of complexity linear in both the formula and the structure being checked, exactly as the one for CTL. We suggest an extension of our CTL^2 model checker that, preserving its complexity, handles fairness and, in particular, enables checking strong fairness.

The rest of this paper is organized as follows: In the figure below we give some background and motivation. In Section 2 we present basic definitions concerning temporal logics. In Section 3 we introduce the language CTL^2. In Section 4 we discuss its expressive power, and in Section 5 we present a linear model-checking procedure for it.



1. A designer that uses CTL.



2. A designer that uses CTL^2.

2 Temporal Logics

We describe the full branching temporal logic CTL* and its subsets CTL and BLTL.

The logic *CTL** combines both branching time and linear time operators. A path quantifier, either A ("for all computation paths") or E ("for some computation path") can prefix

an assertion composed of an arbitrary combination of the usual linear time operators X ("next time") and U ("until"), and boolean connectives. There are two types of formulas in CTL* : *state formulas* (whose satisfaction is related to a specific state) and *path formulas* (whose satisfaction is related to a specific path). Let AP be the set of atomic proposition names.
A state formula is either:

- *True*, *False* (represented in the sequel as t and f, respectively), or p, for all $p \in AP$.
- $\neg\varphi_1$ or $\varphi_1 \vee \varphi_2$, where φ_1 and φ_2 are CTL* state formulas.
- $E\varphi_1$ or $A\varphi_1$, where φ_1 is a CTL* path formula.

A path formula is either:

- A state formula.
- $\neg\varphi_1$, $\varphi_1 \vee \varphi_2$, $X\varphi_1$, or $\varphi_1 U \varphi_2$, where φ_1 and φ_2 are CTL* path formulas.

CTL* is the set of state formulas generated by the above rules. Actually, it is sufficient to allow only one of the path quantifiers A and E. The version we introduce here allows both, as it will be convenient in the sequel.

The logic *CTL* is a restricted subset of CTL* that permits only branching time operators. The operators X, U, and their negations must be immediately preceded by a path quantifier. Formally, it is the subset of CTL* obtained by restricting the path formulas to be $X\varphi_1$ or $\varphi_1 U \varphi_2$, where φ_1 and φ_2 are CTL state formulas.

The logic *LTL* is interpreted over sequences and provides no path quantifiers. An LTL formula is either:

- t, f, or p for all $p \in AP$.
- $\neg\varphi_1$, $\varphi_1 \vee \varphi_2$, $X\varphi_1$, or $\varphi_1 U \varphi_2$, where φ_1 and φ_2 are LTL formulas.

The logic *BLTL* consists of the set of state formulas of the form $A\varphi$ where φ is an LTL formula.

We use the following abbreviations in writing formulas:

- $\wedge$ and $\rightarrow$, interpreted in the usual way.
- $F\varphi = tU\varphi$ ("eventually").
- $G\varphi = \neg F \neg \varphi$ ("always").
- $\varphi_1 \bar{U} \varphi_2 = \neg(\varphi_1 U \varphi_2)$ ("not until").

We define the semantics of CTL*, CTL, and BLTL with respect to a *Kripke structure* $K = \langle W, R, L \rangle$, where W is a set of states, $R \subseteq W \times W$ is a transition relation that must be total, and $L : W \rightarrow 2^{AP}$ maps each state to a set of atomic propositions true in this state. A *path* in

K is a sequence of states, $\pi = w_0, w_1, \ldots$ such that for every $i \geq 0$, $\langle w_i, w_{i+1} \rangle \in R$. We use π^i to denote the suffix of π starting at w_i.

$K, w \models \varphi$ indicates that a state formula φ holds at state w of the Kripke structure K. Similarly, $K, \pi \models \varphi$ means that a path formula φ holds along the path π in K. The relation $\models$ is inductively defined as follows (assuming an agreed K).

- $w \models p$ for $p \in AP$ iff $p \in L(w)$.
- $w \models \neg\varphi_1$ iff $w \not\models \varphi_1$.
- $w \models \varphi_1 \vee \varphi_2$ iff $w \models \varphi_1$ or $w \models \varphi_2$.
- $w \models E\varphi_1$ iff there exists a path $\pi = w_0, w_1, \ldots$ such that $w = w_0$ and $\pi \models \varphi_1$.
- $w \models A\varphi_1$ iff for every path $\pi = w_0, w_1, \ldots$ with $w = w_0$, $\pi \models \varphi_1$.
- $\pi \models \varphi_1$ for a state formula φ_1, iff $w \models \varphi_1$ where w is the first state in π.
- $\pi \models \neg\varphi_1$ iff $\pi \not\models \varphi_1$.
- $\pi \models \varphi_1 \vee \varphi_2$ iff $\pi \models \varphi_1$ or $\pi \models \varphi_2$.
- $\pi \models X\varphi_1$ iff $\pi^1 \models \varphi_1$.
- $\pi \models \varphi_1 U \varphi_2$ iff there exists $i \geq 0$ such that $\pi^i \models \varphi_2$ and for every $0 \leq j < i$, $\pi^j \models \varphi_1$.

Given two formulas φ_1 and φ_2, we say that φ_1 and φ_2 are *equivalent* ($\varphi_1 \equiv \varphi_2$) if for every Kripke structure K, and for every state w of K, $K, w \models \varphi_1$ iff $K, w \models \varphi_2$. When comparing expressive power of two logics L_1 and L_2, we say that L_1 is *more expressive than* L_2 ($L_1 \geq L_2$), provided that for every formula φ_2 of L_2, there exists an equivalent formula φ_1 of L_1. Also, L_1 is *as expressive as* L_2 ($L_1 = L_2$), if both $L_1 \geq L_2$ and $L_2 \geq L_1$, L_1 is *strictly more expressive than* L_2 ($L_1 > L_2$), if both $L_1 \geq L_2$ and $L_2 \not\geq L_1$, and L_1 and L_2 are *incomparable* ($L_1 \neq L_2$) if both $L_1 \not\geq L_2$ and $L_2 \not\geq L_1$.

When comparing expressive power of logics and automata, we say that a formula φ is equivalent to an automaton A, iff A accepts exactly all the models of φ. As discussed in [ES84], the difference in the objects over which branching temporal logics and tree automata are interpreted (trees with varying and possibly infinite branching degrees, versus trees of fixed and finite branching degrees), and the disability of branching time logics to distinguish between different successors, impose technical problems in comparing expressibility of branching temporal logics and tree automata. Discussion is therefore restricted to symmetric tree automata interpreted over infinite trees of a fixed branching degree.

3 The Temporal Logic CTL^2

We introduce a new branching temporal logic, CTL^2. CTL^2 extends CTL by allowing two temporal operators within the path formulas. The temporal operators may be either nested or

connected by a binary boolean connective. Negating one or both of the temporal operators is also allowed. Formally, we distinguish between three types of formulas: state formulas, path formulas of degree 1, and path formulas of degree 2.
Given a set AP of atomic propositions, a CTL^2 state formula is either:

S1. t, f or p, for all $p \in AP$.

S2. $\neg\varphi_1$ or $\varphi_1 \vee \varphi_2$, where φ_1 and φ_2 are CTL^2 state formulas.

S3. $A\varphi_1$ or $E\varphi_1$, where φ_1 is a CTL^2 path formula of degree 1 or 2.

A CTL^2 path formula of degree 1 is either:

P1. $X\varphi_1$ or $\varphi_1 U \varphi_2$, where φ_1 and φ_2 are CTL^2 state formulas.

P2. $\neg\varphi_1$, where φ_1 is a CTL^2 path formula of degree 1.

A CTL^2 path formula of degree 2 is either:

P3. $X\varphi_1$, $\varphi_1 U \varphi_2$ or $\varphi_2 U \varphi_1$, where φ_1 is a CTL^2 path formula of degree 1 and φ_2 is a CTL^2 state formula.

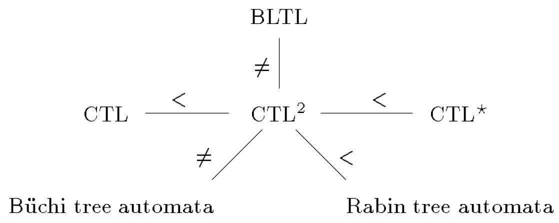
P4. $\varphi_1 \vee \varphi_2$, where φ_1 is either a CTL^2 state formula or a CTL^2 path formula of degree 1, and φ_2 is a CTL^2 path formula of degree 1.

P5. $\neg\varphi_1$, where φ_1 is a CTL^2 path formula of degree 2.

CTL^2 is the set of state formulas generated by the above rules. For example, $AGF\mathit{grant}$, $E(X\mathit{req})U\mathit{grant}$, $A(F\neg\mathit{busy} \vee G\mathit{wait})$, and $E(\neg(\mathit{busy}\ U\ \mathit{req}))U\mathit{grant}$, are all CTL^2 formulas. It is easy to see that as far as syntax is concerned, CTL $\subset$ CTL^2 $\subset$ CTL*, and CTL$^2 \neq$ BLTL.

4 Expressive Power of CTL^2

In this section we investigate the expressive power of CTL^2 with respect to other branching time logics and automata over infinite trees. We state and prove the following expressiveness relations:



We focus on the relation between CTL^2 and CTL. Definitely, an important advantage of CTL^2 over CTL is the increase in the expressive power. Consider the CTL^2 formula $A(\varphi_1 U(\varphi_2 \bar{U} \varphi_3))$

which is, as we shall later prove, not expressible in CTL. The formula is a base for many specifications used for reasoning about concurrent programs. Taking for instance $\varphi_1 = req_1$, $\varphi_2 = \neg req_2$, and $\varphi_3 = \neg CS_1$, it specifies the property "in all computation paths, process 1 keeps signalling req_1 until it eventually enters the critical section and stays there as long as process 2 does not signal req_2". However, not less important is the neat presentation that CTL^2 suggests for specifications whose CTL equivalences are very hard to understand. Indeed, the branching nature of CTL, not only makes it difficult to understand the specification, but also makes it difficult for the specifiers to express correctly their intuition. For example, the CTL^2 formula $A((\varphi_1 U \varphi_2) \rightarrow (\varphi_3 U \varphi_4))$ can express elegantly a FIFO policy. Taking $\varphi_1 = \neg req_1$, $\varphi_2 = req_2$, $\varphi_3 = \neg grant_1$, and $\varphi_4 = grant_2$, gives "in all computation paths, if req_1 does not precede req_2, then $grant_1$ does not precede $grant_2$". The CTL^2 formula $\neg E(\varphi_1 U(X\varphi_2))$ enables specifying more accurate properties than those enabled without nesting of the next operator. Taking $\varphi_1 = \neg invoke(p)$ and $\varphi_2 = execute(p)$, gives "in all computation paths, if p is executed, then p is invoked two steps earlier or before". Note that since we usually want to specify behaviors as invariants, we should precede those formulas with AG, resulting in formulas like $AGA\varphi_1 U \neg(\varphi_2 U \varphi_3)$ or $AG\neg E(\varphi_1 U(X\varphi_2))$, all syntactically correct in CTL^2. We start by comparing CTL^2 with CTL* and BTLT.

Lemma 4.1 *The formula $\psi = AF(p \wedge Xp)$ has no equivalent of CTL^2.*

The proof is a simple extension of Emerson and Halpern's proof from [EH86] for inexpressibility of ψ in CTL.

Claim 4.2 *(1) CTL^2 $\neq$ BLTL.*

(2) CTL^2 $<$ CTL*.

Proof: Lemma 4.1 implies that BLTL $\not\leq$ CTL^2. The second direction follows from the known CTL $\not\leq$ BLTL result ([Lam80], reproved in [EH86]). CTL^2 $\leq$ CTL* holds by syntactic containment. Strictness follows from Lemma 4.1. □

Büchi tree automata are, as follows from [ES84], more expressive than CTL. We show that Büchi tree automata can not handle an additional temporal operator in the path formulas and thus, they are not more expressive than CTL^2. On the other hand, as Rabin tree automata are strictly more expressive than CTL* [ES84], they are also strictly more expressive than CTL^2.

Claim 4.3 *(1) CTL^2$\neq$ Büchi tree automata.*

(2) CTL^2$<$ Rabin tree automata.

Proof: Consider the CTL^2 formula $\psi = AFG\neg p$. ψ defines the set of trees $T_1 = \{V :$ in all paths of V, p is true only finitely often$\}$. In [Rab70], Rabin proves that there exists no Büchi tree automaton that recognizes T_1. Thus, Büchi tree automata $\not\geq$ CTL^2. For the second direction, we extend Wolper's result from [Wol83] to branching time logics and show that there exists no CTL^2 formula that defines the set of trees $T_2 = \{V$: in all paths of V, p is true at all even places$\}$. Yet, T_2 is recognizable by Büchi tree automata. It is interesting to note that extending CTL^2 by allowing quantification over atomic propositions [Wol82] results in a language which is strictly more expressive than Büchi tree automata. $\square$

Lemma 4.4 *The CTL^2 formula $\psi = EFGp$ has no equivalent of CTL.*

The Lemma is proved in [EH86] and the following claim is a straightforward corollary.

Claim 4.5 *CTL^2 $>$ CTL.*

In the rest of this section we show that the CTL^2 formula $EG(\varphi_2 U \varphi_3)$ embodies all the expressiveness superiority of CTL^2 over CTL. Namely, extending CTL by the binary operator EGU results in a logic which is as expressive as CTL. We first suggest a normal form for CTL^2 formulas.

Consider the equivalence $\neg X\varphi \equiv X\neg\varphi$. Since negation of state formulas is allowed, rules P1 and P2 that define a path formula of degree 1, can be replaced by a single rule, according to which, a path formula of degree 1 is either $X\varphi_1, \varphi_1 U \varphi_2$, or $\varphi_1 \bar{U} \varphi_2$, where φ_1 and φ_2 are CTL^2 state formulas. Consider now the equivalence $A\psi \equiv \neg E \neg \psi$. Again, as negation of state formulas is allowed and, moreover, as both A and E are allowed as path quantifiers, rules P3, P4, and P5 that define a path formula of degree 2, can be replaced by the following two rules, according to which, a path formula of degree 2 is either (where φ_1, φ_2, φ_3, and φ_4 are CTL^2 state formulas):

- $XX\varphi_1$, $X(\varphi_1 U \varphi_2)$, $X(\varphi_1 \bar{U} \varphi_2)$,
  $(X\varphi_1)U\varphi_2$, $(\varphi_1 U \varphi_2)U\varphi_3$, $(\varphi_1 \bar{U} \varphi_2)U\varphi_3$,
  $\varphi_1 U(X\varphi_2)$, $\varphi_1 U(\varphi_2 U \varphi_3)$, or $\varphi_1 U(\varphi_2 \bar{U} \varphi_3)$.
- $\varphi_1 \vee (X\varphi_2)$, $\varphi_1 \vee (\varphi_2 U \varphi_3)$, $\varphi_1 \vee (\varphi_2 \bar{U} \varphi_3)$,
  $(X\varphi_1) \vee (X\varphi_2)$, $(X\varphi_1) \vee (\varphi_2 U \varphi_3)$, $(X\varphi_1) \vee (\varphi_2 \bar{U} \varphi_3)$,
  $(\varphi_1 U \varphi_2) \vee (\varphi_3 U \varphi_4)$, $(\varphi_1 U \varphi_2) \vee (\varphi_3 \bar{U} \varphi_4)$, or $(\varphi_1 \bar{U} \varphi_2) \vee (\varphi_3 \bar{U} \varphi_4)$.

Hence, the following logic, called *normal-formed CTL^2*, is equivalent to CTL^2. Given a set AP of atomic propositions, a normal-formed CTL^2 formula is either (where Q stands for either A or E and φ_1, φ_2, φ_3, and φ_4 are normal-formed CTL^2 state formulas):

R1. t, f or p, for all $p \in AP$.

R2. $\neg\varphi_1$ or $\varphi_1 \vee \varphi_2$.

R3. $EX\varphi_1$, $E\varphi_1 U \varphi_2$, or $A\varphi_1 U \varphi_2$.

R4. $EXX\varphi_1$, $EX(\varphi_1 U \varphi_2)$, $EX(\varphi_1 \bar{U} \varphi_2)$,
$Q((X\varphi_1)U\varphi_2)$, $Q((\varphi_1 U \varphi_2)U\varphi_3)$, $Q((\varphi_1 \bar{U} \varphi_2)U\varphi_3)$,
$Q(\varphi_1 U(X\varphi_2))$, $Q(\varphi_1 U(\varphi_2 U \varphi_3))$, or $Q(\varphi_1 U(\varphi_2 \bar{U} \varphi_3))$.

R5. $Q(\varphi_1 \vee (X\varphi_2))$, $Q(\varphi_1 \vee (\varphi_2 U \varphi_3))$, $Q(\varphi_1 \vee (\varphi_2 \bar{U} \varphi_3))$,
$Q((X\varphi_1) \vee (X\varphi_2))$, $Q((X\varphi_1) \vee (\varphi_2 U \varphi_3))$, $Q((X\varphi_1) \vee (\varphi_2 \bar{U} \varphi_3))$,
$Q((\varphi_1 U \varphi_2) \vee (\varphi_3 U \varphi_4))$, $Q((\varphi_1 U \varphi_2) \vee (\varphi_3 \bar{U} \varphi_4))$, or $Q((\varphi_1 \bar{U} \varphi_2) \vee (\varphi_3 \bar{U} \varphi_4))$.

In the sequel, we assume that the CTL^2 formulas are given in that normal form. Note that this at most doubles the formula length.

Let us call CTL^2 formulas of the form $A\varphi_1 U(\varphi_2 \bar{U} \varphi_3)$ or its negation $\neg A\varphi_1 U(\varphi_2 \bar{U} \varphi_3)$, *sad formulas*. CTL^2 formulas of other forms are called *happy formulas*. Note that the normal-form formulas that correspond to the formulas $AFG\neg p$ and $EGFp$, used for proving Claims 4.3 and 4.5, are sad. In order to show that EGU embodies all the expressiveness superiority of CTL^2, we suggest CTL equivalences to all the happy formulas and, in addition, show that sad formulas can be expressed by CTL augmented with EGU.

Below we suggest CTL equivalences for all the happy formulas. We call them CTL equivalences, meaning that if φ_1, φ_2, φ_3, and φ_4 are CTL formulas, then the equivalent formula is of CTL [1]. Rule R5 contains formulas that have a boolean connective in their path formulas. Equivalences to formulas of that form are already known [Eme90] and we give them in Appendix A.1. Below we introduce and prove CTL equivalences for fourteen forms of formulas that cover all the happy formulas in rule R4. A summarizing list of those equivalences exists in Appendix A.2.

Claim 4.6 *(a)* $EXX\varphi_1 \equiv EXEX\varphi_1$.

(b) $EX\varphi_1 U \varphi_2 \equiv EXE\varphi_1 U \varphi_2$.

(c) $EX\varphi_1 \bar{U} \varphi_2 \equiv EXE\varphi_1 \bar{U} \varphi_2$.

Proof: We prove a stronger claim. Let ψ be a CTL* path formula, we prove $EX\psi \equiv EXE\psi$. $w \models EX\psi$ iff there exists a path $\pi = w_0, w_1, \ldots$, with $w_0 = w$, such that $\pi^1 \models \psi$. This holds iff there exists a path $\pi = w_0, w_1, \ldots$, with $w_0 = w$, such that $w_1 \models E\psi$, which holds iff $w \models EXE\psi$.

$\square$

Claim 4.7 *(a)* $A(X\varphi_2)U\varphi_3 \equiv A(AX\varphi_2)U\varphi_3$.

(b) $A(\varphi_1 U \varphi_2)U\varphi_3 \equiv A(A\varphi_1 U \varphi_2)U\varphi_3$.

(c) $A(\varphi_1 \bar{U} \varphi_2)U\varphi_3 \equiv A(A\varphi_1 \bar{U} \varphi_2)U\varphi_3$.

Proof: We prove a stronger claim. Let ψ be a CTL* path formula, we prove $A\psi U\varphi_3 \equiv A(A\psi)U\varphi_3$.

[1]We note here that the equivalences are valid also in the context of CTL*; i.e. taking φ_1, φ_2, and φ_3 as CTL* state formulas.

Assume first that $w \models A(A\psi)U\varphi_3$. Then, for every path $\pi = w_0, w_1, \ldots$, with $w_0 = w$, there exists $i \geq 0$ such that $w_i \models \varphi_3$, and for every $0 \leq j < i$, $w_j \models A\psi$. In particular, for every $0 \leq j < i$, $\pi^j \models \psi$. Thus $\pi \models \psi U \varphi_3$ and $w \models A\psi U\varphi_3$.

Assume now that $w \models A\psi U\varphi_3$. Then, for every path $\pi = w_0, w_1, \ldots$, with $w_0 = w$, there exists $i \geq 0$ such that $w_i \models \varphi_3$, and for every $0 \leq j < i$, $\pi^j \models \psi$. Let i_π be the minimal index i for which $w_i \models \varphi_3$. We show that for every $0 \leq j < i_\pi$, $w_j \models A\psi$ and thus $\pi \models (A\psi)U\varphi_3$. Assume, by way of contradiction, that there exists some $0 \leq j < i$ for which $w_j \not\models A\psi$. That is, there exists a path $\pi' = w'_j, w'_{j+1}, \ldots$, with $w'_j = w_j$, such that $\pi' \not\models \psi$. Consider the path $\rho = w_0, w_1, \ldots, w_j, w'_{j+1}, w'_{j+2}, \ldots$. Since π and ρ coincide in their first j states, $i_\pi > j$ implies that $i_\rho > j$ and therefore, as $\rho \models \psi U\varphi_3$, all $0 \leq k \leq j$ have $\rho^k \models \psi$. In particular, $\pi' \models \psi$, and we reach a contradiction. $\square$

Claim 4.8 ***(a)*** *$E(X\varphi_2)U\varphi_3 \equiv \varphi_3 \vee EXE\varphi_2 U(\varphi_3 \wedge \varphi_2)$.*

(b) *$E(\varphi_1 U\varphi_2)U\varphi_3 \equiv \varphi_3 \vee E(\varphi_1 \vee \varphi_2)U((\varphi_2 \wedge EX\varphi_3) \vee (\varphi_3 \wedge E\varphi_1 U\varphi_2))$.*

(c) *$E(\varphi_1 \bar{U}\varphi_2)U\varphi_3 \equiv \varphi_3 \vee E(\neg\varphi_2)U(((\neg\varphi_1) \wedge (\neg\varphi_2) \wedge EX\varphi_3) \vee (\varphi_3 \wedge E\varphi_1 \bar{U}\varphi_2))$.*

Proof: All three formulas are of the form $E\psi U\varphi_3$, for a CTL* path formula ψ. Their CTL equivalences are based on the same idea: We isolate the case where φ_3, the eventuality of the top level until, holds in the present. If it does not, we modify the until-structure to guarantee both an occurrence of φ_3 eventually, and correctness of ψ along all the suffixes that start before the first occurrence of φ_3. We give the full details below.

(a) Assume first that $w \models \varphi_3 \vee EXE\varphi_2 U(\varphi_3 \wedge \varphi_2)$. Then, either $w \models \varphi_3$, in which case clearly $w \models E(X\varphi_2)U\varphi_3$, or $w \models EXE\varphi_2 U(\varphi_3 \wedge \varphi_2)$. Then, there exists a path $\pi = w_0, w_1, \ldots$, with $w_0 = w$, for which there exists $i \geq 1$ such that $w_i \models \varphi_3 \wedge \varphi_2$, and for every $1 \leq j < i$, $w_j \models \varphi_2$. This implies that $w_i \models \varphi_3$ and for every $0 \leq j < i$, $w_{j+1} \models \varphi_2$. Thus, $\pi \models (X\varphi_2)U\varphi_3$ and $w \models E(X\varphi_2)U\varphi_3$.

Assume now that $w \models E(X\varphi_2)U\varphi_3$. Then, there exists a path $\pi = w_0, w_1, \ldots$, with $w_0 = w$, for which there exists $i \geq 0$ such that $w_i \models \varphi_3$, and for every $0 \leq j < i$, $w_{j+1} \models \varphi_2$. If $i = 0$ then $w \models \varphi_3$. Otherwise, $i \geq 1$, w_i satisfies both φ_3 and φ_2, and for every $1 \leq j < i$, $w_j \models \varphi_2$. Thus $w \models \varphi_3 \vee EXE\varphi_1 U(\varphi_3 \wedge \varphi_2)$.

(b) Assume first that $w \models \varphi_3 \vee E(\varphi_1 \vee \varphi_2)U((\varphi_2 \wedge EX\varphi_3) \vee (\varphi_3 \wedge E\varphi_1 U\varphi_2))$. Then, either $w \models \varphi_3$, in which case clearly $w \models E(\varphi_1 U\varphi_1)U\varphi_3$, or $w \models E(\varphi_1 \vee \varphi_2)U((\varphi_2 \wedge EX\varphi_3) \vee (\varphi_3 \wedge E\varphi_1 U\varphi_2))$. Then, there exists a path $\pi = w_0, w_1, \ldots$, with $w_0 = w$, for which there exists $i \geq 0$ such that $w_i \models (\varphi_2 \wedge EX\varphi_3) \vee (\varphi_3 \wedge E\varphi_1 U\varphi_2))$, and for every $0 \leq j < i$, $w_j \models \varphi_1 \vee \varphi_2$. If $w_i \models \varphi_2 \wedge EX\varphi_3$, then there exists a successor w' of w_i such that $w' \models \varphi_3$. As $w_i \models \varphi_2$, it is guaranteed that every suffix π^j of π, eventually reaches a state (w_i or an earlier one) that satisfies φ_2. Also, as for every $0 \leq j < i$, $w_j \models \varphi_1 \vee \varphi_2$, then, as long as φ_2 is not reached, φ_1 is satisfied. Therefore, a path $\pi' = w'_0, w'_1, \ldots$, with $w'_k = w_k$ for every $0 \leq k \leq i$ and $w'_{i+1} = w'$, has $\pi' \models (\varphi_1 U\varphi_2)U\varphi_3$. Note that this takes care of the case where all the suffixes π^j satisfy the internal until $\varphi_1 U\varphi_2$, by having its eventuality, φ_2, satisfied before φ_3 occurs. We now consider the case where $w_i \models \varphi_3 \wedge E\varphi_1 U\varphi_2$, which corresponds to the case where the external until reaches its eventuality before all the internal ones do. If $w_i \models \varphi_3 \wedge E\varphi_1 U\varphi_2$, then there exists a path $\pi' = w'_i, w'_{i+1}, \ldots$, with $w'_i = w_i$ such that $\pi' \models \varphi_1 U\varphi_2$. Consider the path $\rho = w_0, w_1, \ldots, w_i, w'_{i+1}, w'_{i+2}, \ldots$. Similarly to the previous case $\rho \models (\varphi_1 U\varphi_2)U\varphi_3$, and thus $w \models E(\varphi_1 U\varphi_2)U\varphi_3$.

Assume now that $w \models E(\varphi_1 U\varphi_2)U\varphi_3$. Then, there exists a path $\pi = w_0, w_1, \ldots$, with $w_0 = w$, for which there exists $i \geq 0$ such that $w_i \models \varphi_3$, and for every $0 \leq j < i$, $\pi^j \models \varphi_1 U\varphi_2$. If $i = 0$ then $w \models \varphi_3$. If $i \geq 1$, let $i' = i-1$, if $w_{i-1} \models \varphi_2$, and $i' = i$, otherwise. Note that in the second case, since $\pi^{i-1} \models \varphi_1 U\varphi_2$ and $w_{i-1} \not\models \varphi_2$, then it is guaranteed that $\pi^i \models \varphi_1 U\varphi_2$ and thus $w_{i'} \models (\varphi_2 \wedge EX\varphi_3) \vee (\varphi_3 \wedge E\varphi_1 U\varphi_2)$. Also, since for every $0 \leq j < i$, $\pi^j \models \varphi_1 U\varphi_2$, then for every such j, $w_j \models \varphi_1 \vee \varphi_2$. Thus, $w \models \varphi_3 \vee E(\varphi_1 \vee \varphi_2)U((\varphi_2 \wedge EX\varphi_3) \vee (\varphi_3 \wedge E\varphi_1 U\varphi_2))$.

(c) Assume first that $w \models \varphi_3 \vee E(\neg\varphi_2)U(((\neg\varphi_1) \wedge (\neg\varphi_2) \wedge EX\varphi_3) \vee (\varphi_3 \wedge E\varphi_1 \bar{U}\varphi_2))$. Then, either $w \models \varphi_3$, in which case clearly $w \models E(\varphi_1 \bar{U}\varphi_2)U\varphi_3$, or $w \models E(\neg\varphi_2)U(((\neg\varphi_1) \wedge (\neg\varphi_2) \wedge EX\varphi_3) \vee (\varphi_3 \wedge E\varphi_1 \bar{U}\varphi_2))$. Then, there exists a path $\pi = w_0, w_1, \ldots$, with $w_0 = w$, for which there exists $i \geq 0$ such that $w_i \models ((\neg\varphi_1) \wedge (\neg\varphi_2) \wedge EX\varphi_3) \vee (\varphi_3 \wedge E\varphi_1 \bar{U}\varphi_2)$, and for every $0 \leq j < i$, $w_j \not\models \varphi_2$. If $w_i \models (\neg\varphi_1) \wedge (\neg\varphi_2) \wedge EX\varphi_3$, then there exists a successor w' of w_i such that $w' \models \varphi_3$. As $w_i \models (\neg\varphi_1) \wedge (\neg\varphi_2)$, it is guaranteed that every suffix π^j of π, eventually reaches a state (w_i or a preceding one) that does not satisfy φ_1. Also, as for every $0 \leq j \leq i$, $w_j \not\models \varphi_2$, it is guaranteed that φ_2 does not hold in this state and in all the states that precede it. Therefore, a path $\pi' = w'_0, w'_1, \ldots$, with $w'_k = w_k$ for every $0 \leq k \leq i$ and $w'_{i+1} = w'$, has $\pi' \models (\varphi_1 \bar{U}\varphi_2)U\varphi_3$. Note that this takes care of the case where all the suffixes π^j satisfy $\varphi_1 \bar{U}\varphi_2$, with having φ_1 interrupted (without φ_2 occurring) before φ_3, the eventuality of the until, occurs. The other disjunct corresponds to the case where φ_3 occurs before being φ_1 interrupted. If $w_i \models \varphi_3 \wedge E\varphi_1 \bar{U}\varphi_2$, then there exists a path $\pi' = w'_i, w'_{i+1}, \ldots$, with $w'_i = w_i$ such that $\pi' \models \varphi_1 \bar{U}\varphi_2$. Consider the path $\rho = w_0, w_1, \ldots, w_i, w'_{i+1}, w'_{i+2}, \ldots$. Similarly to the previous case $\rho \models (\varphi_1 \bar{U}\varphi_2)U\varphi_3$, and thus $w \models E(\varphi_1 \bar{U}\varphi_2)U\varphi_3$.

Assume now that $w \models E(\varphi_1 \bar{U}\varphi_2)U\varphi_3$. Then, there exists a path $\pi = w_0, w_1, \ldots$, with $w_0 = w$, for which there exists $i \geq 0$ such that $w_i \models \varphi_3$, and for every $0 \leq j < i$, $\pi^j \models \varphi_1 \bar{U}\varphi_2$. If $i = 0$ then $w \models \varphi_3$. Otherwise, let $i' = i - 1$, if $w_{i-1} \not\models \varphi_1$, and let $i' = i$, otherwise. Note that since $\pi^{i-1} \models \varphi_1 \bar{U}\varphi_2$, then it is guaranteed that $w_{i-1} \not\models \varphi_2$. Moreover, if $w_{i-1} \models \varphi_1$, then it is guaranteed that $\pi^i \models \varphi_1 \bar{U}\varphi_2$. Thus, $w_{i'} \models ((\neg\varphi_1) \wedge (\neg\varphi_2) \wedge EX\varphi_3) \vee (\varphi_3 \wedge E\varphi_1 \bar{U}\varphi_2)$. Also, since for every $0 \leq j < i$, $\pi^j \models \varphi_1 \bar{U}\varphi_2$, then for every such j, $w_j \not\models \varphi_2$. Hence, $w \models \varphi_3 \vee E(\neg\varphi_2)U(((\neg\varphi_1) \wedge (\neg\varphi_2) \wedge EX\varphi_3) \vee (\varphi_3 \wedge E\varphi_1 \bar{U}\varphi_2))$.

□

Claim 4.9 ***(a)*** *$A\varphi_1 U(X\varphi_2) \equiv AX\varphi_2 \vee (\varphi_1 \wedge AXA\varphi_1 U(\varphi_2 \vee AX\varphi_2))$.*

(b) *$A\varphi_1 U(\varphi_2 U\varphi_3) \equiv A\varphi_1 U(A\varphi_2 U\varphi_3)$.*

Proof: Both formulas are of the form $A\varphi_1 U\psi$ for a CTL* path formula ψ. Again, the idea is to distinguish between the case where ψ, the eventuality of the until, holds in all paths in the present, and the case it does not. In the latter, modification of the until-structure is required. In (b), as $(A\psi) \vee A\varphi_1 U(A\psi) \equiv A\varphi_1 U(A\psi)$, both cases coincide. Note that the sad formula

$A\varphi_1 U(A\varphi_2 \bar{U} \varphi_3)$, that has no CTL equivalent, has also the form $A\varphi_1 U\psi$, for a CTL* path formula ψ. We give the full details below.

(a) Assume first that $w \models AX\varphi_2 \vee (\varphi_1 \wedge AXA\varphi_1 U(\varphi_2 \vee AX\varphi_2))$. Then, either $w \models AX\varphi_2$, in which case clearly $w \models A\varphi_1 U(X\varphi_2)$, or $w \models \varphi_1 \wedge AXA\varphi_1 U(\varphi_2 \vee AX\varphi_2)$. Then, for every path $\pi = w_0, w_1, \ldots$, with $w_0 = w$, there exists $i \geq 1$ such that $w_i \models \varphi_2 \vee AX\varphi_2$ and for every $0 \leq j < i$, $w_j \models \varphi_1$. This implies that for every such path, there exists $i' \geq 0$, which equals either i, if $w_i \models AX\varphi_2$, or $i - 1$, if $w_i \models \varphi_2$, such that $w_{i'+1} \models \varphi_2$, and for every $0 \leq j < i'$, $w_j \models \varphi_1$. Thus, $\pi \models \varphi_1 U(X\varphi_2)$ and $w \models A\varphi_1 U(X\varphi_2)$.

Assume now that $w \models A\varphi_1 U(X\varphi_2)$. Then, for every path $\pi = w_0, w_1, \ldots$, with $w_0 = w$, there exists $i \geq 0$, such that $w_{i+1} \models \varphi_2$, and for every $0 \leq j < i$, $w_j \models \varphi_1$. Let i_π be the minimal index i for which $w_{i+1} \models \varphi_2$. If $w_{i_\pi} \models \varphi_1$, then $\pi \models \varphi_1 \wedge X\varphi_1 U\varphi_2$. We show that if $w_{i_\pi} \not\models \varphi_1$, then $w_{i_\pi} \models AX\varphi_2$, and thus $\pi \models \varphi_1 UAX\varphi_2$. Assume that $w_{i_\pi} \not\models \varphi_1$, and assume, by way of contradiction, that $w_{i_\pi} \not\models AX\varphi_2$. That is, there exists a successor w' of w_{i_π} such that $w' \not\models \varphi_2$. Consider a path $\pi' = w'_0, w'_1, \ldots$, with $w'_k = w_k$ for every $0 \leq k \leq i_\pi$ and $w'_{i_\pi+1} = w'$. Since π and π' coincide in their first i_π states, $i_{\pi'} \geq i_\pi$. Moreover, as $w'_{i_\pi+1} \not\models \varphi_2$, then $i_{\pi'} > i_\pi$. $\pi' \models \varphi_1 U(X\varphi_2)$ and therefore, for every $0 \leq k \leq i_{\pi'}$, $w'_k \models \varphi_1$. In particular, $w_{i_\pi} \models \varphi_1$, and we reach a contradiction. Hence, for every path $\pi = w_0, w_1, \ldots$, with $w_0 = w$, either $\pi \models \varphi_1 \wedge X\varphi_1 U\varphi_2$ or $\pi \models \varphi_1 U(AX\varphi_2)$. This implies that either $w \models AX\varphi_2$, or that for every such π, there exists $i \geq 1$ such that $w_i \models \varphi_2 \vee AX\varphi_2$ and for every $0 \leq j < i$, $w_j \models \varphi_1$. Thus, $w \models AX\varphi_2 \vee (\varphi_1 \wedge AXA\varphi_1 U(\varphi_2 \vee AX\varphi_2))$.

(b) Assume first that $w \models A\varphi_1 U(A\varphi_2 U\varphi_3)$. Then, for every path $\pi = w_0, w_1, \ldots$, with $w_0 = w$, there exists $i \geq 0$ such that $w_i \models A\varphi_2 U\varphi_3$ and for every $0 \leq j < i$, $w_j \models \varphi_1$. In particular, $\pi^i \models \varphi_2 U\varphi_3$ and thus $w \models A\varphi_1 U(\varphi_2 U\varphi_3)$.

Assume now that $w \models A\varphi_1 U(\varphi_2 U\varphi_3)$. Then, for every path $\pi = w_0, w_1, \ldots$, with $w_0 = w$, there exists $i \geq 0$ such that $\pi^i \models \varphi_2 U\varphi_3$ and for every $0 \leq j < i$, $w_j \models \varphi_1$. That is, there exists $k \geq i$ such that $w_k \models \varphi_3$ and for every $i \leq j < k$, $w_j \models \varphi_2$. Thus, π has some state, w_k, that satisfies φ_3, and the prefix $w_0, \ldots, w_k$, of π, has some $0 \leq i \leq k$ such that for every $0 \leq j < i$, $w_j \models \varphi_1$ and for every $i \leq j < k$, $w_j \models \varphi_2$. Let k_π be the minimal index k for which $w_k \models \varphi_3$, and let i_π be the maximal index i that satisfies the above conditions; That is, either $i_\pi = k_\pi$, or $w_{i_\pi} \not\models \varphi_1$. We show that $w_{i_\pi} \models A\varphi_2 U\varphi_3$, and thus $\pi \models \varphi_1 U(A\varphi_2 U\varphi_3)$. Assume, by way of contradiction, that $w_{i_\pi} \not\models A\varphi_2 U\varphi_3$. That is, there exists a path $\pi' = w'_{i_\pi}, w'_{i_\pi+1}, \ldots$, with $w'_{i_\pi} = w_{i_\pi}$ such that $\pi' \not\models \varphi_2 U\varphi_3$. Consider the path $\rho = s_0, s_1, \ldots$, where $s_j = w_j$ for $0 \leq j \leq i_\pi$, and $s_j = w'_j$ for $j > i_\pi$. $\rho \models \varphi_1 U(\varphi_2 U\varphi_3)$ and therefore, ρ has some state, s_{k_ρ}, such that $s_{k_\rho} \models \varphi_3$ and for every $0 \leq j < k_\rho$, $s_j \not\models \varphi_3$ (i.e., k_ρ is minimal). Since ρ and π coincide in their first i_π states and $k_\pi \geq i_\pi$, then $k_\rho \geq i_\pi$. Also, there exists some state s_{i_ρ}, such that $0 \leq i_\rho \leq k_\rho$, for every $0 \leq j < i_\rho$, $s_j \models \varphi_1$, for every $i_\rho \leq j < k_\rho$, $s_j \models \varphi_2$, and either $i_\rho = k_\rho$ or $s_{i_\rho} \not\models \varphi_1$ (i.e., i_ρ is the maximal). If $i_\rho > i_\pi$, then $w_{i_\pi} \models \varphi_1$ and as $w_{i_\pi} \not\models \varphi_3$, this contradicts the maximality of i_π. Therefore, $i_\rho \leq i_\pi$. Thus, π' has $k_\rho \geq i_\pi$ such that $w_{k_\rho} \models \varphi_3$, and for every $i_\pi \leq j < k_\rho$, $w_j \models \varphi_2$. Thus $\pi' \models \varphi_2 U\varphi_3$ and we reach a contradiction. Hence, $w \models A\varphi_1 U(\varphi_2 U\varphi_3)$.

$\square$

Claim 4.10 *(a)* $E\varphi_1 U(X\varphi_2) \equiv E\varphi_1 U(EX\varphi_2)$.

(b) $E\varphi_1 U(\varphi_2 U\varphi_3) \equiv E\varphi_1 U(E\varphi_2 U\varphi_3)$.

(c) $E\varphi_1 U(\varphi_2 \bar{U}\varphi_3) \equiv E\varphi_1 U(E\varphi_2 \bar{U}\varphi_3)$.

Proof: We prove a stronger claim. Let ψ be a CTL* path formula, we prove $E\varphi_1 U\psi \equiv E\varphi_1 UE\psi$. $w \models E\varphi_1 U\psi$ iff there exists a path $\pi = w_0, w_1, \ldots$, with $w_0 = w$, for which there exists $i \geq 0$ such that $\pi^i \models \psi$ and for every $0 \leq j < i$, $w_j \models \varphi_1$. This holds iff there exists a path $\pi' = w_0, w_1, \ldots, w_i, w'_{i+1}, \ldots$, such that $w_i \models E\psi$ and for every $0 \leq j < i$, $w_j \models \varphi_1$. That is, iff $w \models E\varphi_1 UE\psi$. $\square$

We now show that a formula of the form $A\varphi_1 U(\varphi_2 \bar{U}\varphi_3)$ can be expressed by CTL augmented with $EG(\varphi_2 U\varphi_3)$. For simplicity, we consider its negation $E\varphi_1 \bar{U}(\varphi_2 \bar{U}\varphi_3)$. As CTL^2 allows negation of state formulas, the result for $A\varphi_1 U(\varphi_2 \bar{U}\varphi_3)$ follows.

Claim 4.11 $E\varphi_1 \bar{U}(\varphi_2 \bar{U}\varphi_3) \equiv E(\varphi_2 \vee \varphi_3)U((\neg\varphi_1) \wedge E(\varphi_2 U\varphi_3)) \vee EG(\varphi_2 U\varphi_3)$.

Proof: Consider the equivalence

$$\varphi_1 \bar{U}\psi \equiv ((\neg\psi)U((\neg\varphi_1) \wedge (\neg\psi))) \vee G\neg\psi.$$

Taking $\psi = \varphi_2 \bar{U}\varphi_3$, we get $E\varphi_1 \bar{U}(\varphi_2 \bar{U}\varphi_3) \equiv E(((\varphi_2 U\varphi_3)U((\neg\varphi_1) \wedge (\varphi_2 U\varphi_3))) \vee G(\varphi_2 U\varphi_3))$. By distributing the E quantifier over the disjunction, we get

$$E\varphi_1 \bar{U}(\varphi_2 \bar{U}\varphi_3) \equiv E(\varphi_2 U\varphi_3)U((\neg\varphi_1) \wedge (\varphi_2 U\varphi_3)) \vee EG(\varphi_2 U\varphi_3)).$$

Hence, it is left to show that $E(\varphi_2 U\varphi_3)U((\neg\varphi_1) \wedge (\varphi_2 U\varphi_3)) \equiv E(\varphi_2 \vee \varphi_3)U((\neg\varphi_1) \wedge E(\varphi_2 U\varphi_3))$.

It is easy to see that a state that satisfies the left-hand side of the equivalence, satisfies also its right-hand side. We consider the second direction. Assume that $w \models E(\varphi_2 \vee \varphi_3)U((\neg\varphi_1) \wedge E\varphi_2 U\varphi_3)$. Then, there exists a path $\pi = w_0, w_1, \ldots$, with $w_0 = w$, for which there exists $i \geq 0$ such that $w_i \models (\neg\varphi_1) \wedge E\varphi_2 U\varphi_3$, and for all $0 \leq j < i$, $w_j \models \varphi_2 \vee \varphi_3$. If $w_i \models \varphi_3$, then it is guaranteed that every suffix π^j of π eventually reaches a state (w_i or an earlier one) that satisfies φ_3. Also, as for all $0 \leq j < i$, $w_j \models \varphi_2 \vee \varphi_3$, then, as long as φ_3 is not reached, φ_2 is satisfied. Therefore, in this case, $\pi \models (\varphi_2 U\varphi_3)U((\neg\varphi_1) \wedge (\varphi_2 U\varphi_3))$. If $w_i \not\models \varphi_3$, then there must exist a path $\pi' = w'_i, w'_{i+1}, \ldots$, with $w'_i = w_i$ such that $\pi' \models \varphi_2 U\varphi_3$. Consider the path $\rho = w_0, w_1, \ldots, w_i, w'_{i+1}, w'_{i+2}, \ldots$. Similarly to the previous case, $\rho \models (\varphi_2 U\varphi_3)U((\neg\varphi_1) \wedge (\varphi_2 U\varphi_3))$ and thus $w \models E(\varphi_2 U\varphi_3)U((\neg\varphi_1) \wedge (\varphi_2 U\varphi_3))$. $\square$

In the next section we show that the additional expressive power of CTL^2 is given for free. Namely, the model-checking problem for the logic CTL^2 is of the same complexity as the one for CTL.

5 A Linear Model-Checking Procedure for CTL^2

In this section we introduce an efficient algorithm for the model-checking problem for CTL^2. Given a Kripke structure $K = \langle W, R, L\rangle$ and a CTL^2 formula ψ, our algorithm labels K such that for every state $w \in W$, w is labeled with ψ iff $w \models \psi$. The algorithm is of complexity linear in both the size of K and the length of ψ. Thus, the increase in the expressive power does not effect model-checking complexity which remains linear, exactly as the one for CTL. In Section 5.1 we extend our algorithm, preserving its complexity, to handle fairness.

The algorithm extends the efficient model checker for CTL introduced in [CES86]. There, a formula is handled by successively applying a state labelling algorithm to its sub-formulas. In more details, given a CTL formula ψ and a Kripke structure K, the algorithm takes the sub-formulas of ψ, starting with the innermost ones, and, iteratively, labels with each sub-formula exactly those states of K that satisfy it. Each iteration handles a single sub-formula which may have one of seven forms that together cover CTL modalities. Since ψ has at most $|\psi|$ sub-formulas, the check terminates after at most $|\psi|$ iterations. Each iteration requires $O(|K|)$ steps and hence, the entire check is accomplished after $O(|\psi| * |K|)$ steps.

In CTL^2, there are more than seven forms. All the possible combinations of one or two temporal operators should be considered. However, as we showed in the previous section, all those forms, except $A\varphi_1 U(\varphi_2 \bar{U} \varphi_3)$, have CTL equivalences. Thus, labelling with happy sub-formulas can be done by labelling their CTL equivalences. In addition, we introduce a procedure that handles sad sub-formulas.

Below we define the *closure of a formula* for every CTL^2 formula. The closure of φ, $cl(\varphi)$, extends the notion of sub-formulas used in [CES86]. Intuitively, It is the set of CTL^2 formulas that contains φ and all the formulas that the labelling of φ depends on. Given a CTL^2 formula φ, $cl(\varphi)$ is inductively defined as follows (φ_1, φ_2, and φ_3 are CTL^2 formulas):

- If φ is either t or f, then $cl(\varphi) = \{\varphi\}$.
- If φ is a proposition, then $cl(\varphi) = \{\varphi, t, f\}$.
- If φ is $\neg\varphi_1$ or $EX\varphi_1$, then $cl(\varphi) = \{\varphi\} \cup cl(\varphi_1)$.
- If φ is $\varphi_1 \vee \varphi_2$, $E\varphi_1 U \varphi_2$, or $A\varphi_1 U \varphi_2$, then $cl(\varphi) = \{\varphi\} \cup cl(\varphi_1) \cup cl(\varphi_2)$.
- If φ is a happy formula, then $cl(\varphi) = cl(\varphi')$, where φ' is the CTL equivalent of φ.
- If φ is $A\varphi_1 U(\varphi_2 \bar{U} \varphi_3)$, then $cl(\varphi) = \{\varphi, EG(\varphi_2 U \varphi_3)\} \cup cl(E(\varphi_2 \vee \varphi_3) U((\neg\varphi_1) \wedge E(\varphi_2 U \varphi_3)))$.

Lemma 5.1 *For every CTL^2 formula ψ,*

(a) *All the formulas in $cl(\psi)$ are either t, f, propositions, or have the form $\neg\varphi_1$, $\varphi_1 \vee \varphi_2$, $EX\varphi_1$, $E\varphi_1 U \varphi_2$, $A\varphi_1 U \varphi_2$, $A\varphi_1 U(\varphi_2 \bar{U} \varphi_3)$, or $EG(\varphi_2 U \varphi_3)$.*

(b) *The size of $cl(\psi)$ is linear in $|\psi|$.*

procedure model_check (ψ, K);
begin
  for $i := 1$ **to** $|cl(\psi)|$ **do**
    for every $\varphi \in cl(\psi)$ with $|cl(\varphi)| = i$ **do**
      case structure of φ is of the form
        An atomic proposition : **noop**;
        $\varphi_1 \vee \varphi_2$: **for** every $w \in W$ **do**
            if $\varphi_1 \in L(w)$ or $\varphi_2 \in L(w)$ **then** add $\varphi_1 \vee \varphi_2$ to $L(w)$;
        $\neg\varphi_1$: **for** every $w \in W$ **do**
            if $\varphi_1 \notin L(w)$ **then** add $\neg\varphi_1$ to $L(w)$;
        $EX\varphi_1$: **for** every $w \in W$ **do**
            if $\varphi_1 \in L(w^{'})$ for some successor $w^{'}$ of w **then** add $EX\varphi_1$ to $L(w)$;
        $E\varphi_1 U \varphi_2$: *EU_check* $(\varphi_1, \varphi_2, \varphi)$;
        $A\varphi_1 U \varphi_2$: *AU_check* $(\varphi_1, \varphi_2, \varphi)$;
        $A\varphi_1 U(\varphi_2 \bar{U} \varphi_3)$: **for** every $w \in W$ **do**
            if $E(\varphi_2 \vee \varphi_3)U((\neg\varphi_1) \wedge E(\varphi_2 U \varphi_3)) \notin L(w)$ and $EG(\varphi_2 U \varphi_3) \notin L(w)$ **then**
              add $A\varphi_1 U(\varphi_2 \bar{U} \varphi_3)$ to $L(w)$;
        $EG(\varphi_2 U \varphi_3)$: *EGU_check* $(\varphi_2, \varphi_3, \varphi)$
      end;
end;

Figure 1: The model-checking procedure for CTL^2.

Lemma 5.1 is the key for the model-checking procedure, presented in Figure 1.

Given a CTL^2 formula ψ, the procedure is called with the CTL equivalent of ψ (if exists). As in [CES86], it iteratively labels formulas from $cl(\varphi)$. It is guaranteed that when φ is labeled, then all the formulas in $cl(\varphi) \setminus \{\varphi\}$ are already labeled. Formulas of the form $\neg\varphi_1$, $\varphi_1 \vee \varphi_2$, $EX\varphi_1$, $E\varphi_1 U \varphi_2$, or $A\varphi_1 U \varphi_2$ are labeled using the corresponding procedure described in [CES86]. Formulas of the form $A\varphi_1 U(\varphi_2 \bar{U} \varphi_3)$, are labeled, according to Claim 4.11, using the procedure *EGU_Check*$(\varphi_2, \varphi_3, \varphi)$, presented in Figure 2 and explained below.

EGU_check$(\varphi_2, \varphi_3, \varphi)$ labels states w of K for which $w \models EG\varphi_2 U \varphi_3$. Note that $w \models EG\varphi_2 U \varphi_3$ iff there exists a path π starting at w such that all the states in π satisfy either φ_2 or φ_3, and infinitely many states in π satisfy φ_3. Accordingly, *EGU_check* first labels states that belong to a maximal strongly connected component (MSCC) whose all states are labeled by either φ_2 or φ_3, and there exists a state in it that satisfies φ_3. Those states are inserted into the set S^0, meaning that there is a path of length 0 from them to such a MSCC. Then, the procedure proceeds backwards the transition relation, inserting to S^j states for which there exists a path (whose all states satisfy either φ_2 or φ_3) of length j from them to such a MSCC. Clearly, there is no need to proceed with j's greater than $|W|$.

EGU_check uses the procedure *find_MSCCs*(ψ), which, given a formula ψ, finds the MSCCs in the sub-structure of K that consists of all the states in K that are labeled with ψ (we refer to non-trivial MSCC. I.e., a single state without a self loop is not a MSCC). The output of

```
procedure EGU_check (φ2, φ3, φ);
begin
  for j ∈ [0 ... |W|] do S^j := φ end;
  for every C ∈ find_MSCCs(φ2 ∨ φ3) do
      if there exists w ∈ C such that φ3 ∈ L(w) then
        for every w ∈ C do add φ to L(w); add w to S^0 end
  end;
  j := 0;
  while j < |W| do
    S := S^j;
    while S ≠ φ do
      remove some w from S;
      for every predecessor w' of w do
          if φ ∉ L(w') and (φ2 ∈ L(w') or φ3 ∈ L(w')) then
            add φ to L(w') ; add w' to S^{j+1}
      end;
    end;
    j := j + 1;
  end;
end;
```

Figure 2: The procedure *EGU_check*.

find_MSCCs(ψ) is a set of (disjoint) subsets of W such that every subset is a MSCC.

We now consider the complexity of the procedure. As *find_MSCCs*(ψ) is of complexity linear in $|K|$ [Tar72], initialization is performed in linear time. Since all the sets S^j are pairwise disjoint, the body of the for-loop is executed at most once for each transition in R. Hence, the whole procedure terminates after at most $O(|K|)$ steps.

5.1 A Fair Model Checker for CTL^2

In [CES86], fairness is introduced into CTL. A new logic, CTL^F, interpreted over Kripke structures with fairness constraint, is suggested. Formally, a Kripke structure with fairness constraint (a fair structure) is a four-tuple $\langle W, R, L, \mathcal{F}\rangle$, where W, R, and L are as for usual structures, and $\mathcal{F} \subseteq 2^W$ is a set of subsets of W. Given a path π in K, let $inf(\pi)$ be the set of states that appear in π infinitely often. A path π is fair iff for each set $F \in \mathcal{F}$, $inf(\pi) \cap F \neq \phi$. CTL^F has exactly the same syntax as CTL. Its semantics is also identical to that of CTL, except that all path quantifiers range over fair paths. This enables, while verifying concurrent programs, to refer only to fair execution sequences. In this section we show how the fair model checker, introduced in [CES86], can be used to handle CTL^2.

As in CTL, let *fair-CTL*^{2} (CTL^2_F) be the logic CTL^2 when interpreted over fair structures. Given a fair structure, a state w in it, and a CTL^2_F formula ψ, $w \models_F \psi$ indicates that ψ holds in w, with path quantifiers ranging over fair paths only. Given two formulas ψ_1 and ψ_2, we say that ψ_1 and ψ_2 are *fair-equivalent* ($\psi_1 \equiv_F \psi_2$) if for every fair structure and for every state w in it, $w \models_F \psi_1$ iff $w \models_F \psi_2$. We first show that translating a happy CTL^2 formula into its CTL equivalent, using the equivalences suggested in Section 4, preserves equivalence also with respect to CTL^2_F and CTL$_F$.

Claim 5.2 *Given a happy CTL^2 formula ψ and its CTL equivalent ψ', $\psi \equiv_F \psi'$.*

Proof: We have to show that the fourteen equivalences given in Section 4, remain valid when path quantifiers range over fair paths only. Fortunately, the proofs given there remain valid too. That is, replacing "path" by "fair path" preserves the validity of the proof. $\square$

It is left to extend our *EGU_check* procedure to handle fairness. This can easily be done by requiring each MSCC in *find_MSCCs*$(\varphi_2 \vee \varphi_3)$ to contain at least one state from each $F \in \mathcal{F}$.

As in CTL, fairness constraints can be used to increase the expressive power of the logic. Instead a set of subsets of W, one can specify the fairness constraint $\mathcal{F}$ as a set of CTL^2 formulas. A path π is fair iff for each formula $\xi \in \mathcal{F}$, there are infinitely many states in π that satisfy ξ. Consider the CTL* formula $\psi = A(GFp \rightarrow \varphi)$, where p is atomic. Checking the formula $A\varphi$ with respect to a fair structure that has $\mathcal{F} = \{\{p\}\}$ as its fairness constraint, is equivalent to checking ψ in a structure that has no fairness constraint. Using the extended model checker for CTL^2, we can thus check formulas of the form $A(GFp \rightarrow \varphi)$ for which $A\varphi$ is a CTL^2 formula. In particular, we can check the formula $A(GFp \rightarrow GFq)$ that specifies strong fairness.

6 Discussion

We introduced the branching temporal logic CTL^2. CTL^2 overpasses CTL in both expressiveness and ease of use. Still, model-checking for CTL^2 is linear.

The fact that we were able to provide a linear-time model-checking procedure for CTL^2 is not surprising. Reducing branching-time model checking to linear-time model checking [EL85] can be used to show that bounding the length of the path formulas in CTL* reduces model-checking complexity to polynomial. The advantage of our procedure is that once having a CTL model-checking package, extending it to a one that handles CTL^2 is very easy. Indeed, for the happy subset of CTL^2, it involves only a translation. For the rest of CTL^2, techniques like BDDs [BCM+92] that are helpful in CTL model checking, are adaptable for CTL^2[Gei94]. In addition, the extension preserves the ability to handle fairness.

Hence, CTL^2 constitutes an additional step in the process of making model checking a practical and convenient tool for formal verification.

Acknowledgement We thank Danny Geist for suggesting the representation of sad formulas using the *EGU* operator.

References

[BCM+92] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, June 1992.

[BG94] O. Bernholtz and O. Grumberg. Buy one, get one free !!! In *Proceedings of the First International Conference on Temporal Logic*, Bonn, July 1994.

[Bro86] M.C. Browne. An improved algorithm for the automatic verification of finite state systems using temporal logic. In *Proceedings of the First Symposium on Logic in Computer Science*, pages 260–266, Cambridge, June 1986.

[Bry86] R.E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8), 1986.

[CE81] E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proc. Workshop on Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer-Verlag, 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, January 1986.

[EC82] E.A. Emerson and E.M. Clarke. Using branching time logic to synthesize synchronization skeletons. *Science of Computer Programming*, 2:241–266, 1982.

[EH86] E.A. Emerson and J.Y. Halpern. Sometimes and not never revisited: On branching versus linear time. *Journal of the ACM*, 33(1):151–178, 1986.

[EL85] E.A. Emerson and C.-L. Lei. Modalities for model checking: Branching time logic strikes back. In *Proceedings of the Twelfth ACM Symposium on Principles of Programming Languages*, pages 84–96, New Orleans, January 1985.

[Eme90] E.A. Emerson. Temporal and modal logic. *Handbook of theoretical computer science*, pages 997–1072, 1990.

[ES84] A.E. Emerson and A.P. Sistla. Deciding full branching time logics. *Information and Control*, 61(3):175–201, 1984.

[Gei94] D. Geist. Private communication. 1994.

[GL91] O. Grumberg and D. Long. Model checking and modular verification. In *Proc. 2nd Conference on Concurrency Theory*, volume 527 of *Lecture Notes in Computer Science*, 1991.

[Lam80] L. Lamport. Sometimes is sometimes “not never” - on the temporal logic of programs. In *Proceedings of the 7th ACM Symposium on Principles of Programming Languages*, pages 174–185, January 1980.

[McM93] K.L. McMillan. *Symbolic model checking*. Kluwer Academic Publishers, 1993.

[Pnu81] A. Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13:45–60, 1981.

[Rab70] M.O. Rabin. Weakly definable relations and special automata. In *Proc. Symp. Math. Logic and Foundations of Set Theory*, pages 1–23. North Holland, 1970.

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

[Tar72] R.E. Tarjan. Depth first search and linear graph algorithms. *SIAM Journal of Computing*, 1(2):146–160, 1972.

[WG93] P. Wolper and P. Godefroid. Partial-order methods for temporal verification. In *Proc. 4th Conferance on Concurrency Theory*, volume 715 of *Lecture Notes in Computer Science*, pages 233–246, Hildesheim, August 1993. Springer-Verlag.

[Wol82] P. Wolper. Specification and synthesis of communicating processes using an extended temporal logic. In *Proc. 9th Symposium on Principles of Programming Languageseedings*, pages 20–33, Albuquerque, January 1982.

[Wol83] P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1–2):72–99, 1983.

A Summery of the equivalences

A.1 Equivalences for Rule R5.

The following equivalences show that the formulas in Rule R5 do not add to the expressive power of CTL. Their redundancy is already known, yet the equivalences we suggest have $cl(\psi')$ of size linear in $|\psi|$, where ψ' is the CTL equivalent of ψ. We use the notation $\varphi_1 \tilde{U} \varphi_2 = (\neg\varphi_1)\bar{U}(\neg\varphi_2)$.

1. $Q(\varphi_1 \vee (X\varphi_2)) \equiv \varphi_1 \vee QX\varphi_2$
$Q(\varphi_1 \vee (\varphi_2 U \varphi_3)) \equiv \varphi_1 \vee Q\varphi_2 U \varphi_3$

2. $Q((X\varphi_1) \vee (X\varphi_2)) \equiv QX(\varphi_1 \vee \varphi_2)$
$Q((X\varphi_1) \vee (\varphi_2 U \varphi_3)) \equiv \varphi_3 \vee QX\varphi_1 \vee (\varphi_2 \wedge QX(\varphi_1 \vee Q\varphi_2 U \varphi_3))$
$Q((X\varphi_1) \vee (\varphi_2 \bar{U} \varphi_3)) \equiv ((\neg\varphi_3) \wedge (\neg\varphi_2)) \vee QX\varphi_1 \vee ((\neg\varphi_3) \wedge QX(\varphi_1 \vee Q\varphi_2 \bar{U} \varphi_3))$

3. $E((\varphi_1 U \varphi_2) \vee (\varphi_3 U \varphi_4)) \equiv (E\varphi_1 U \varphi_2) \vee (E\varphi_3 U \varphi_4)$
$E((\varphi_1 U \varphi_2) \vee (\varphi_3 \bar{U} \varphi_4)) \equiv (E\varphi_1 U \varphi_2) \vee (E\varphi_3 \bar{U} \varphi_4)$
$E((\varphi_1 \bar{U} \varphi_2) \vee (\varphi_3 \bar{U} \varphi_4)) \equiv (E\varphi_1 \bar{U} \varphi_2) \vee (E\varphi_3 \bar{U} \varphi_4)$

4. $A((\varphi_1 U \varphi_2) \vee (\varphi_3 U \varphi_4)) \equiv$
$A((\varphi_2 \vee \varphi_4)\tilde{U}(\varphi_3 \vee A\varphi_1 U \varphi_2)) \wedge A((\varphi_2 \vee \varphi_4)\tilde{U}(\varphi_1 \vee A\varphi_3 U \varphi_4)) \wedge AF(\varphi_2 \vee \varphi_4)$
$A((\varphi_1 U \varphi_2) \vee (\varphi_3 \tilde{U} \varphi_4)) \equiv$
$A((\varphi_2 \vee \varphi_3)\tilde{U}(\varphi_4 \vee A\varphi_1 U \varphi_2)) \wedge A((\varphi_2 \vee \varphi_3)\tilde{U}(\varphi_2 \vee A\varphi_3 U \varphi_4))$
$A((\varphi_1 \tilde{U} \varphi_2) \vee (\varphi_3 \tilde{U} \varphi_4)) \equiv$
$A((\varphi_1 \vee \varphi_3)\tilde{U}(\varphi_2 \vee A\varphi_3 \tilde{U} \varphi_4)) \wedge A((\varphi_1 \vee \varphi_3)\tilde{U}(\varphi_4 \vee A\varphi_1 \tilde{U} \varphi_2))$

A.2 Equivalences for Rule R4.

Below we list the equivalences for the formulas in rule R4.

1. $EXX\varphi_1 \equiv EXEX\varphi_1$
$EX\varphi_1 U \varphi_2 \equiv EXE\varphi_1 U \varphi_2$
$EX\varphi_1 \bar{U} \varphi_2 \equiv EXE\varphi_1 \bar{U} \varphi_2$

2. $A(X\varphi_1)U\varphi_2 \equiv A(AX\varphi_1)U\varphi_2$
$A(\varphi_1 U \varphi_2)U\varphi_3 \equiv A(A\varphi_1 U \varphi_2)U\varphi_3$
$A(\varphi_1 \bar{U} \varphi_2)U\varphi_3 \equiv A(A\varphi_1 \bar{U} \varphi_2)U\varphi_3$

3. $E(X\varphi_1)U\varphi_2 \equiv \varphi_2 \vee EXE\varphi_1 U(\varphi_2 \wedge \varphi_1)$
$E(\varphi_1 U \varphi_2)U\varphi_3 \equiv \varphi_3 \vee E(\varphi_1 \vee \varphi_2)U((\varphi_2 \wedge EX\varphi_3) \vee (\varphi_3 \wedge E\varphi_1 U \varphi_2))$
$E(\varphi_1 \bar{U} \varphi_2)U\varphi_3 \equiv \varphi_3 \vee E(\neg\varphi_2)U(((\neg\varphi_1) \wedge (\neg\varphi_2) \wedge EX\varphi_3) \vee (\varphi_3 \wedge E\varphi_1 \bar{U} \varphi_2))$

4. $A\varphi_1 U(X\varphi_2) \equiv AX\varphi_2 \vee (\varphi_1 \wedge AXA\varphi_1 U(\varphi_2 \vee AX\varphi_2))$
$A\varphi_1 U(\varphi_2 U \varphi_3) \equiv A\varphi_1 U(A\varphi_2 U \varphi_3)$

5. $E\varphi_1 U(X\varphi_2) \equiv E\varphi_1 U(EX\varphi_2)$
$E\varphi_1 U(\varphi_2 U \varphi_3) \equiv E\varphi_1 U(E\varphi_2 U \varphi_3)$
$E\varphi_1 U(\varphi_2 \bar{U} \varphi_3) \equiv E\varphi_1 U(E\varphi_2 \bar{U} \varphi_3)$