A Decision Procedure for CTL* Based on Tableaux and Automata

Oliver Friedmann[1], Markus Latte[1], and Martin Lange[2]

[1] Dept. of Computer Science, University of Munich, Germany
[2] Dept. of Electrical Engineering and Computer Science, University of Kassel, Germany

Abstract. We present a decision procedure for the full branching-time logic CTL* which is based on tableaux with global conditions on infinite branches. These conditions can be checked using automata-theoretic machinery. The decision procedure then consists of a doubly exponential reduction to the problem of solving a parity game. This has advantages over existing decision procedures for CTL*, in particular the automata-theoretic ones: the underlying tableaux only work on subformulas of the input formula. The relationship between the structure of such tableaux and the input formula is given by very intuitive tableau rules. Furthermore, runtime experiments with an implementation of this procedure in the MLSolver tool show the practicality of this approach within the limits of the problem's computational complexity of being 2EXPTIME-complete.

1 Introduction

The full branching-time temporal logic CTL* is an important tool for the specification and verification of reactive systems [9] and of agent-based systems [13], for program synthesis [16], etc. Emerson and Halpern have introduced CTL* [4] as a formalism which supersedes both the branching-time logic CTL and the linear-time logic LTL. As much as this has led to an easy unification of CTL and LTL, it has also proved to be quite a difficulty in obtaining decision procedures for this logic. The first was automata-theoretic [6], requiring the determinisation of ω-word automata resulting from linear-time formulas. A series of improvements in this part has eventually led to Emerson and Jutla's automata-theoretic decision procedure [5] whose asymptotic worst-case running time is optimal, namely doubly exponential [22]. Other procedures have been given some time later, namely Reynolds' proof system [17], Gabbay and Pnueli's proof system [9], and most recently Reynolds' tableaux [18].

In this paper we present a characterisation of CTL* satisfiability. It is formulated as a calculus of infinite tableaux with natural rules and with global conditions on their branches. The non-termination of the tableaux raises the question after an effective decision procedure based on this calculus, and it is only here that we use automata-theoretic machinery. Branches violating the global condition are recognisable by nondeterministic Büchi automata, and we can then use determinisation and complementation to reduce the question of existence of a tableau to the problem of solving a doubly exponentially large parity game.

This also yields an asymptotically optimal decision procedure which has two distinct advantages over some of the existing ones. First, the tableaux only use subformulas of the input formula, while automata are only used on top of the tableaux in order to check the global conditions. Second, the reduction is implemented in the modal fixpoint solver MLSolver [7] which uses the high-performance parity game solver PGSolver [8] as a backend. The work presented here is therefore—to the best of our knowledge—the first serious attempt at creating a practical decision procedure for CTL*.

The rest of the paper is organised as follows. Sect. 2 recalls CTL*. Sect. 3 introduces the tableau calculus. Soundness and completeness are technically non-trivial to prove but still proceed along standard lines. The detailed proofs are omitted for lack of space and contained in an appendix. Sect. 4 presents a decision procedure which uses automata-theory in order to reduce the satisfiability problem to the problem of solving a parity game. Sect. 5 highlights the advantages of this approach in comparison to existing others. Sect. 6 reports on experimental results.

2 CTL*

Let $\mathcal{P}$ be a countably infinite set of propositional constants. A transition system is a tuple $\mathcal{T} = (\mathcal{S}, s^*, \to, \lambda)$ with $(\mathcal{S}, \to)$ being a directed graph, $s^* \in \mathcal{S}$ being a designated starting state and $\lambda : \mathcal{S} \to 2^{\mathcal{P}}$ is a labeling function. We assume transition systems to be total, i.e. every state has at least one successor. A *path* π in $\mathcal{T}$ is an infinite sequence of states $s_0, s_1, \ldots$ s.t. $s_i \to s_{i+1}$ for all i. With π^k we denote the suffix of π starting with state s_k, and $\pi(k)$ denotes s_k in this case.

Branching-time temporal formulas are given by the following grammar.

$$\varphi \quad ::= \quad q \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathtt{X}\varphi \mid \varphi \mathtt{U} \varphi \mid \mathtt{E}\varphi$$

where $q \in \mathcal{P}$. Formulas of the form q or $\neg q$ are called *literals*. We use $\bar{\ell}$ to denote the complement of a literal ℓ, i.e. $\bar{\ell} = \neg q$ if $\ell = q$ and $\bar{\ell} = q$ if $\ell = \neg q$.

Other constructs like $\mathtt{tt}, \mathtt{ff}, \vee, \to$ are derived as usual, and so are the temporal ones $\varphi \mathtt{R} \psi := \neg(\neg\varphi \mathtt{U} \neg\psi)$, $\mathtt{G}\varphi = \mathtt{ff}\mathtt{R}\varphi$, $\mathtt{F}\varphi = \mathtt{tt}\mathtt{U}\varphi$, and $\mathtt{A}\varphi := \neg\mathtt{E}\neg\varphi$. A formula of this extended syntax is in *positive normal form* if $\neg$ only occurs in front of a propositional constant. The set of *subformulas* of φ is denoted by $Sub(\varphi)$ and defined as usual by setting $Sub(\varphi \circ \psi) := \{\varphi \circ \psi, \mathtt{X}(\varphi \circ \psi)\} \cup Sub(\varphi) \cup Sub(\psi)$ for $\circ$ being $\mathtt{U}$ or $\mathtt{R}$. The notation is extended to formula sets in the usual way. The size $|\varphi|$ of a formula φ is number of its subformulas. A *quantifier-bound formula block* is an $\mathtt{E}$- or $\mathtt{A}$-labeled set of formulas. We omit the braces for singleton sets. Formulas are interpreted over paths π of a transition systems $\mathcal{T} = (\mathcal{S}, s^*, \to, \lambda)$.

$$\begin{array}{lll}
\mathcal{T}, \pi \models q & \text{iff} & q \in \lambda(\pi(0)) \\
\mathcal{T}, \pi \models \neg\varphi & \text{iff} & \mathcal{T}, \pi \not\models \varphi \\
\mathcal{T}, \pi \models \varphi \wedge \psi & \text{iff} & \mathcal{T}, \pi \models \varphi \text{ and } \mathcal{T}, \pi \models \psi \\
\mathcal{T}, \pi \models \mathtt{X}\varphi & \text{iff} & \mathcal{T}, \pi^1 \models \varphi \\
\mathcal{T}, \pi \models \varphi \mathtt{U} \psi & \text{iff} & \exists k \in \mathbb{N}, \mathcal{T}, \pi^k \models \psi \text{ and } \forall j < k : \mathcal{T}, \pi^j \models \varphi \\
\mathcal{T}, \pi \models \mathtt{E}\varphi & \text{iff} & \exists \pi', \text{ s.t. } \pi'(0) = \pi(0) \text{ and } \mathcal{T}, \pi' \models \varphi
\end{array}$$

$$(\mathtt{A}\wedge)\ \frac{\mathtt{A}(\varphi,\Sigma),\mathtt{A}(\psi,\Sigma),\Phi}{\mathtt{A}(\varphi\wedge\psi,\Sigma),\Phi}\qquad (\mathtt{A}\vee)\ \frac{\mathtt{A}(\varphi,\psi,\Sigma),\Phi}{\mathtt{A}(\varphi\vee\psi,\Sigma),\Phi}\qquad (\mathtt{Al})\ \frac{\ell,\Phi\ \mid\ \mathtt{A}\Sigma,\Phi}{\mathtt{A}(\ell,\Sigma),\Phi}$$

$$(\mathtt{AU})\ \frac{\mathtt{A}(\psi,\varphi,\Sigma),\mathtt{A}(\psi,\mathtt{X}(\varphi\mathtt{U}\psi),\Sigma),\Phi}{\mathtt{A}(\varphi\mathtt{U}\psi,\Sigma),\Phi}\qquad (\mathtt{AR})\ \frac{\mathtt{A}(\psi,\Sigma),\mathtt{A}(\varphi,\mathtt{X}(\varphi\mathtt{R}\psi),\Sigma),\Phi}{\mathtt{A}(\varphi\mathtt{R}\psi,\Sigma),\Phi}$$

$$(\mathtt{AA})\ \frac{\mathtt{A}\varphi,\Phi\ \mid\ \mathtt{A}\Sigma,\Phi}{\mathtt{A}(\mathtt{A}\varphi,\Sigma),\Phi}\qquad (\mathtt{AE})\ \frac{\mathtt{E}\varphi,\Phi\ \mid\ \mathtt{A}\Sigma,\Phi}{\mathtt{A}(\mathtt{E}\varphi,\Sigma),\Phi}\qquad (\mathtt{Ett})\ \frac{\Phi}{\mathtt{E}\emptyset,\Phi}$$

$$(\mathtt{E}\vee)\ \frac{\mathtt{E}(\varphi,\Pi),\Phi\ \mid\ \mathtt{E}(\psi,\Pi),\Phi}{\mathtt{E}(\varphi\vee\psi,\Pi),\Phi}\qquad (\mathtt{E}\wedge)\ \frac{\mathtt{E}(\varphi,\psi,\Pi),\Phi}{\mathtt{E}(\varphi\wedge\psi,\Pi),\Phi}\qquad (\mathtt{El})\ \frac{\mathtt{E}\Pi,\ell,\Phi}{\mathtt{E}(\ell,\Pi),\Phi}$$

$$(\mathtt{EU})\ \frac{\mathtt{E}(\psi,\Pi),\Phi\ \mid\ \mathtt{E}(\varphi,\mathtt{X}(\varphi\mathtt{U}\psi),\Pi),\Phi}{\mathtt{E}(\varphi\mathtt{U}\psi,\Pi),\Phi}\qquad (\mathtt{EE})\ \frac{\mathtt{E}\varphi,\mathtt{E}\Pi,\Phi}{\mathtt{E}(\mathtt{E}\varphi,\Pi),\Phi}$$

$$(\mathtt{ER})\ \frac{\mathtt{E}(\psi,\varphi,\Pi),\Phi\ \mid\ \mathtt{E}(\psi,\mathtt{X}(\varphi\mathtt{R}\psi),\Pi),\Phi}{\mathtt{E}(\varphi\mathtt{R}\psi,\Pi),\Phi}\qquad (\mathtt{EA})\ \frac{\mathtt{A}\varphi,\mathtt{E}\Pi,\Phi}{\mathtt{E}(\mathtt{A}\varphi,\Pi),\Phi}$$

$$(\mathtt{X}_0)\ \frac{\mathtt{A}\Sigma_1,\ldots,\mathtt{A}\Sigma_m}{\mathtt{AX}\Sigma_1,\ldots,\mathtt{AX}\Sigma_m,\Lambda}\qquad (\mathtt{X}_1)\ \frac{\mathtt{E}\Pi_1,\mathtt{A}\Sigma_1,\ldots,\mathtt{A}\Sigma_m\quad\ldots\quad\mathtt{E}\Pi_n,\mathtt{A}\Sigma_1,\ldots,\mathtt{A}\Sigma_m}{\mathtt{EX}\Pi_1,\ldots,\mathtt{EX}\Pi_n,\mathtt{AX}\Sigma_1,\ldots,\mathtt{AX}\Sigma_m,\Lambda}$$

Fig. 1. The pre-tableau rules for CTL*.

Two formulas φ and ψ are equivalent, written $\varphi \equiv \psi$, if for all paths π of all transition systems $\mathcal{T}$: $\mathcal{T},\pi \models \varphi$ iff $\mathcal{T},\pi \models \psi$. It is well-known and easy to see that every formula is equivalent to one in positive normal form. A formula φ is called a *state formula* if for all $\mathcal{T},\pi,\pi'$ with $\pi(0)=\pi'(0)$ we have $\mathcal{T},\pi \models \varphi$ iff $\mathcal{T},\pi' \models \varphi$. Hence, satisfaction of a state formula in a path only depends on the first state of the path. Note that φ is a state formula iff $\varphi \equiv \mathtt{E}\varphi$. For state formulas we also write $\mathcal{T},s \models \varphi$ for $s \in \mathcal{S}$. CTL* is the set of all branching-time formulas which are state formulas. A CTL* formula φ is *satisfiable* if there is a transition system $\mathcal{T}$ with an initial state s^* s.t. $\mathcal{T},s^* \models \varphi$.

3 Tableaux for CTL*

From now on, formulas are assumed to be in positive normal form. We will construct a tableau for a given state formula ϑ. The following notations are used: Σ and Π are finite (possibly empty) sets of formulas with Σ being interpreted as a disjunction of formulas and Π as a conjunction. We write Λ for a set of literals. For a set of formulas Γ let $\mathtt{X}\Gamma := \{\mathtt{X}\psi \mid \psi \in \Gamma\}$. A *goal (for ϑ)* is a non-empty set—the outermost braces are omitted—of the form $\mathtt{A}\Sigma_1,\ldots,\mathtt{A}\Sigma_n,\mathtt{E}\Pi_1,\ldots,\mathtt{E}\Pi_m,\Lambda$ where $n,m \geq 0$, and $\Sigma_1,\ldots,\Sigma_n,\Pi_1,\ldots,\Pi_m,\Lambda$ are subsets of $Sub(\vartheta)$. Such a goal stands for the state formula $\bigwedge_{i=1}^{n}\mathtt{A}\big(\bigvee_{\psi\in\Sigma_i}\psi\big)\wedge\bigwedge_{i=1}^{m}\mathtt{E}\big(\bigwedge_{\psi\in\Pi_i}\psi\big)\wedge\bigwedge_{\ell\in\Lambda}\ell$. Goals are denoted by $\mathcal{C}$. We write $Seq(\vartheta)$ for the set of all possible goals for ϑ. Note that this is a finite set of at most doubly exponential size in $|\vartheta|$. A goal $\mathcal{C}$ is *consistent* if there is no $q \in \mathcal{P}$ s.t. $q \in \mathcal{C}$ and $\neg q \in \mathcal{C}$.

Definition 1. *A* pre-tableau *for ϑ is a possibly infinite tree built according to the rules of Fig. 1 whose root is $\mathtt{E}\vartheta$, whose nodes are all consistent and do not contain $\mathtt{A}\emptyset$, and whose leaves consist of literals only.*

We write pre-tableaux as trees growing upwards. Consequently, a rule in Fig. 1 has a goal at the bottom and one or several subgoals at the top. Letter ℓ stands for arbitrary literals. Rule $(\mathtt{X_1})$ is the only rule with more than one subgoal, rules $(\mathtt{Al})$, $(\mathtt{AA})$, $(\mathtt{AE})$, $(\mathtt{E}\vee)$, $(\mathtt{EU})$, $(\mathtt{ER})$ each have a single subgoal which can be chosen nondeterministically to be of either of two forms.

An occurrence of a formula is called *principal* if it gets transformed by a rule. For example, the occurrence of $\varphi \wedge \psi$ is principal in $(\mathtt{E}\wedge)$. A principal formula has *descendants* in the subgoals. For example, both occurrences of φ and ψ are descendants of the principal $\varphi \wedge \psi$ in rule $(\mathtt{E}\wedge)$.

Note that, in the modal rules $(\mathtt{X_0})$ and $(\mathtt{X_1})$, every formula apart from those in the literal part is principal. Literals in the literal part can never be principal, but literals in an A- or E-block are principal in rules $(\mathtt{Al})$ and $(\mathtt{El})$. Finally, any non-principal occurrence of a formula in a goal may have a *copy* in one of the subgoals. The copy is the same formula since it has not been transformed. For instance, any formula in Σ in rule $(\mathtt{Al})$ has a copy in the subgoal if it is of the right form, but does not have a copy if it is of the left form.

A quantifier-bound block $\mathtt{A}\Sigma$ or $\mathtt{E}\Pi$ is called *principal* as well if it contains a principal formula, and possibly has descendants in the subgoal(s). For example, $\mathtt{A}(\varphi \wedge \psi, \Sigma)$ has two descendants $\mathtt{A}(\varphi, \Sigma)$ and $\mathtt{A}(\psi, \Sigma)$ in an application of $(\mathtt{A}\wedge)$.

Definition 2. *Let $\mathcal{C}$ be a goal to which a rule r is applicable and let $\mathcal{C}'$ be one of its subgoals. Furthermore, let $\mathtt{Q}_1\Delta_1$, resp. $\mathtt{Q}_2\Delta_2$ with $\mathtt{Q}_1, \mathtt{Q}_2 \in \{\mathtt{E}, \mathtt{A}\}$ and $\Delta_1, \Delta_2 \subseteq Sub(\vartheta)$ be quantifier-bound blocks occurring in the A- or E-part of $\mathcal{C}$, resp. $\mathcal{C}'$. We say that $\mathtt{Q}_1\Delta_1$ is* connected *to $\mathtt{Q}_2\Delta_2$ in $\mathcal{C}$ and $\mathcal{C}'$, if either*

- *$\mathtt{Q}_1\Delta_1$ is principal in r, and $\mathtt{Q}_2\Delta_2$ is one of its descendants in $\mathcal{C}'$; or*
- *$\mathtt{Q}_1\Delta_1$ is not principal in r and $\mathtt{Q}_2\Delta_2$ is a copy of $\mathtt{Q}_1\Delta_1$ in $\mathcal{C}'$.*

We write this as $(\mathcal{C}, \mathtt{Q}_1\Delta_1) \rightsquigarrow (\mathcal{C}', \mathtt{Q}_2\Delta_2)$. If the rule instance can be inferred from the context we may also simply write $\mathtt{Q}_1\Delta_1 \rightsquigarrow \mathtt{Q}_2\Delta_2$. Additionally, let ψ, resp. ψ' be a formula occurring in Δ_1, resp. Δ_2. We say that ψ is connected *to ψ' in $(\mathcal{C}, \mathtt{Q}_1\Delta_1)$ and $(\mathcal{C}', \mathtt{Q}_2\Delta_2)$, if either*

- *ψ is principal in r, and ψ' is one of its descendants in $\mathcal{C}'$; or*
- *ψ is not principal in r and ψ' is a copy of ψ in $\mathcal{C}'$.*

We write this as $(\mathcal{C}, \mathtt{Q}_1\Delta_1, \psi) \rightsquigarrow (\mathcal{C}', \mathtt{Q}_2\Delta_2, \psi')$. If the rule instance can be inferred from the context we may also simply write $(\mathtt{Q}_1\Delta_1, \psi) \rightsquigarrow (\mathtt{Q}_2\Delta_2, \psi')$. A block connection $(\mathcal{C}_1, \mathtt{Q}_1\Delta_1) \rightsquigarrow (\mathcal{C}_2, \mathtt{Q}_2\Delta_2)$ is called spawning *iff $\mathtt{Q}_2\psi \in \Delta_1$ is principal and $\Delta_2 = \{\psi\}$. The only rules that possibly induce a spawning block connection are $(\mathtt{EE})$, $(\mathtt{EA})$, $(\mathtt{AA})$ and $(\mathtt{AE})$.*

Definition 3. *Let $\mathcal{C}_0, \mathcal{C}_1, \ldots$ be an infinite branch of a pre-tableau t for some Φ. A* trace *Ξ in this branch is an infinite sequence $\mathtt{Q}_0\Delta_0, \mathtt{Q}_1\Delta_1, \ldots$ s.t. for all $i \in \mathbb{N}$:*

$(\mathcal{C}_i, \mathtt{Q}_i\Delta_i) \rightsquigarrow (\mathcal{C}_{i+1}, \mathtt{Q}_{i+1}\Delta_{i+1})$. *A* trace Ξ *is called an* E*-trace, resp.* A*-trace if there is an* $i \in \mathbb{N}$ *s.t.* $\mathtt{Q}_j = \mathtt{E}$*, resp.* $\mathtt{Q}_j = \mathtt{A}$ *for all* $j \geq i$*. We say that a trace is* finitely spawning *if it contains only finitely many spawning block connections.*

Lemma 4. *Every infinite branch of a pre-tableau contains infinitely many applications of rules* $(\mathtt{X}_0)$ *or* $(\mathtt{X}_1)$*.*

Proof. Assume by contradiction that there is an infinite branch $\mathcal{C}_0, \mathcal{C}_1, \ldots$ in a pre-tableau for some Φ that contains only finitely many applications of the modal rules. Note that there must be a trace $\mathtt{Q}_0\Delta_0, \mathtt{Q}_1\Delta_1, \ldots$ in the branch that is principal infinitely often. Now we consider a lexicographic measure on all the Δ_i that counts how many non-modal formulas of a certain depth are contained in the block. Note that every rule application except the modal rule decreases this measure. But this cannot be the case. □

Definition 5. *A* thread t *in a trace* $\Xi = \mathtt{Q}_0\Delta_0, \mathtt{Q}_1\Delta_1, \ldots$ *is an infinite sequence* $\psi_0, \psi_1, \ldots$ *s.t. for all* $i \in \mathbb{N}$*:* $(\mathcal{C}_i, \mathtt{Q}_i\Delta_i, \psi_i) \rightsquigarrow (\mathcal{C}_{i+1}, \mathtt{Q}_{i+1}\Delta_{i+1}, \psi_{i+1})$*. Such a thread* t *is called a* U*-thread, resp.* R*-thread if there is a formula* $\varphi\mathtt{U}\psi \in \mathit{Sub}(\Phi)$*, resp.* $\varphi\mathtt{R}\psi \in \mathit{Sub}(\Phi)$ *s.t.* $\psi_j = \varphi\mathtt{U}\psi$*, resp.* $\psi_j = \varphi\mathtt{R}\psi$ *for infinitely many* j*.*

An E*-trace is called* good *iff it has no* U*-thread; similarly, an* A*-trace is called* good *iff it has an* R*-thread.*

This immediately yields the definition of a *bad* trace: an E-trace is bad if it contains an U-thread, and an A-trace is bad if it contains no R-thread.

Lemma 6. *Every trace in an infinite branch of a pre-tableau is either an* A*-trace or an* E*-trace and only finitely spawning.*

Proof. Let $\Xi = \mathtt{Q}_0\Delta_0, \mathtt{Q}_1\Delta_1, \ldots$ be a trace and assume by contradiction that $\{i \mid \mathtt{Q}_i\Delta_i \rightsquigarrow \mathtt{Q}_{i+1}\Delta_{i+1} \text{ is spawning}\}$ is infinite. Let $i_0 < i_1 < \ldots$ be the ascending sequence of numbers in this infinite set and let ϕ_{i_j} denote the formula in the singleton set Δ_{i_j+1}. Note that for all j it is the case that $\phi_{i_{j+1}} \in \mathit{Sub}(\phi_{i_j})$ and $\phi_{i_j} \neq \phi_{i_{j+1}}$, hence the set cannot be infinite. Now note that every finitely spawning trace eventually must be either an A- or an E-trace. □

Lemma 7. *Every thread in a trace of an infinite branch of a pre-tableau is either an* U*- or an* R*-thread.*

Proof. Let $t = \psi_0, \psi_1, \ldots$ be a thread. Assume that t is neither an U- nor an R-thread, hence there is a position i^* s.t. ψ_i is neither of the form $\psi'\mathtt{U}\psi''$ nor of the form $\psi'\mathtt{R}\psi''$ for all $i \geq i^*$, hence $\psi_{i+1} \in \mathit{Sub}(\psi_i)$ for all $i \geq i^*$. By Lemma 4 it follows that $\psi_{i+1} \neq \psi_i$ for infinitely many i which cannot be the case, hence t has to be a U- or an R-thread. Finally, assume that t is both an U- and an R-thread, i.e. there are positions $i_0 < i_1 < i_2$ s.t. $\psi_{i_0} = \psi_{i_2} = \psi'\mathtt{R}\psi''$ and $\psi_{i_1} = \varphi'\mathtt{U}\varphi''$. Hence $\psi_{i_1} \in \mathit{Sub}(\psi_{i_0}) \setminus \{\psi_{i_0}\}$ and $\psi_{i_2} \in \mathit{Sub}(\psi_{i_1}) \setminus \{\psi_{i_1}\}$, thus $\psi'\mathtt{R}\psi'' \in \mathit{Sub}(\psi'\mathtt{R}\psi'') \setminus \{\psi'\mathtt{R}\psi''\}$ which cannot be the case. □

Lemma 8. *For every* U*- and every* R*-thread (in a trace of an infinite branch of a pre-tableau)* $\psi_0, \psi_1, \ldots$ *there is an* $i \in \mathbb{N}$ *such that* ψ_i *is an* U*-, or an* R*-formula resp., and* $\psi_j = \psi_i$ *or* $\psi_j = \mathtt{X}\psi_i$ *for all* $j \geq i$*.*

Proof. For all $i \in \mathbb{N}$, it holds that $\psi_{i+1} \in Sub(\psi_i)$, or $\psi_{i+1} = \mathtt{X}\psi_i$ provided that ψ_i is an `U`- or an `R`-formula. The map removing from a formula its frontal `X` converts the thread into a chain which is weakly decreasing with respect to the subformula order. Because this order is well-founded the claim follows. □

Definition 9. *A* tableau *for ϑ is a pre-tableau for ϑ that does not contain a branch which contains a bad trace.*

In other words, all traces in a tableau must be good. Such tableaux exactly characterise satisfiability of CTL* in the sense of the following theorem.

Theorem 10. *For all $\vartheta \in$ CTL*: ϑ is satisfiable iff there is a tableau for ϑ.*

The completeness proof is technically tedious but does not use any heavy machinery once the right invariants etc. are being found. Given a model for ϑ we use this to construct a pre-tableau in a certain way. Then assume that the result is not a tableau and derive a contradiction from it. Soundness can be shown by collapsing a tableau into a tree-like transition system and verifying that it is indeed a model of ϑ. The proofs for completeness and soundness are presented in the appendix.

4 A Decision Procedure for CTL*

4.1 Using Automata to Recognise Tableau Branches

The main difficulty in deciding the existence of a tableau for a formula φ is the global condition on infinite branches being required to be good. We propose to use automata-theory for this. Pre-tableau branches can be represented as infinite words over a certain alphabet, and we will show that the language of good branches is recognisable by a nondeterministic Büchi-automaton (NBA). This is not trivial since a nondeterministic machine cannot easily check for absence of `U`-threads in `E`-traces for instance. However, we can use the nondeterminism in order to check for violations, i.e. the presence of `U`-threads in an `E`-trace for instance. This then has to be complemented and combined with something checking for the presence of an `R`-thread in an `A`-trace in order to have a device recognising exactly the set of good paths of a tableau.

The goal is then to replace the global condition on branches of having good traces by an annotation of the tableau nodes with automaton states and a global condition on these states. For instance, if the resulting automaton was of Büchi type, then a tableau can be seen as a pre-tableau with nodes annotated by the automaton s.t. on every infinite path, infinitely many final states occur.

Now note that the automaton recognising good paths needs to be deterministic: suppose there are two branches uv and uw with a common prefix u s.t. both branches are recognised by $\mathcal{A}$. If $\mathcal{A}$ is nondeterministic then it may have two different accepting runs on uv and uw that differ on the common prefix u already. Remember that an annotation of tableau nodes with a single automaton state is required. However, this is possible if $\mathcal{A}$ is deterministic.

The problem of deciding existence of a tableau then reduces to the problem of solving a game. Its nodes are pre-tableau nodes annotated with states of the deterministic automaton. Nondeterministic choices in the tableau rules translate into choices of the existential player in the game; the branching rules ($\mathtt{X_0}$) and ($\mathtt{X_1}$) translate into choices of the universal player. The type of the game is the same as the type of $\mathcal{A}$. For instance, if $\mathcal{A}$ is a deterministic parity automaton then the game is a parity game.

Here we are particularly interested in Büchi and parity automata [11]. An NBA is a tuple $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ with Q being a finite set of *states*, Σ a finite *alphabet*, $q_0 \in Q$ an *initial state*, $\delta \subseteq Q \times \Sigma \times Q$ the *transition relation* and $F \subseteq Q$ a set of *final states*. A *run* of $\mathcal{A}$ on a $a_0 a_1 \ldots \in \Sigma^\omega$ is an infinite sequence $q_0, q_1, \ldots$ s.t. $(q_i, a_i, q_{i+1}) \in \delta$ for all $i \in \mathbb{N}$. It is accepting if $q_i \in F$ for infinitely many i. The *language* of the NBA $\mathcal{A}$ is $L(\mathcal{A}) = \{w \mid \text{there is an accepting run of } \mathcal{A} \text{ on } w\}$. A *co-Büchi automaton* (NcoBA) is syntactically the same as a NBA. However, a run $q_0, q_1, \ldots$ of an NcoBA is accepting if it only contains finitely many non-final states. A *parity automaton* (NPA) is a tuple $\mathcal{A} = (Q, \Sigma, q_0, \delta, \Omega)$ with Q, Σ, q_0, δ as above and $\Omega : Q \to \mathbb{N}$ assigns to each state a *priority*. A run $q_0, q_1, \ldots$ is accepting if $\max\{\Omega(q) \mid q = q_i \text{ for infinitely many } i \in \mathbb{N}\}$ is even. The *index* of an NPA $\mathcal{A}$ is the number of different priorities occurring, i.e. $|\Omega[Q]|$.

An NBA / NcoBA / NPA with transition relation δ is *deterministic* (DBA / DcoBA / DPA) if $|\{q' \mid (q, a, q') \in \delta\}| = 1$ for all $q \in Q$ and $a \in \Sigma$. Determinism and the duality between Büchi and co-Büchi condition as well as the self-duality of the parity acceptance condition makes it easy to complement a DcoBA to a DBA as well as a DPA to a DPA again. The following is a standard and straight-forward result [11, Sec. 1.2] in the theory of ω-word automata.

Lemma 11. *For every DcoBA, resp. DPA, $\mathcal{A}$ there is a DBA, resp. DPA, $\overline{\mathcal{A}}$ with $L(\overline{\mathcal{A}}) = \overline{L(\mathcal{A})}$ and $|\overline{\mathcal{A}}| = |\mathcal{A}|$.*

4.2 Automata for Tableau Branches

We regard rule applications—more precisely: pairs of a goal and one of its subgoals in one of the tableau rules—in a pre-tableau for a formula φ as symbols of a finite alphabet. Naïvely, this would yield an alphabet of doubly exponential size since there are doubly exponentially many different goals. However, note that such a pair is entirely determined by the principal block and the principal formula of the goal and a number specifying the subgoal. This enables a smaller symbolic encoding. For instance, the transition from the goal $\mathtt{A}(\mathtt{E}\varphi, \Sigma), \Phi$ to the subgoal $\mathtt{A}\Sigma, \Phi$ in rule ($\mathtt{AE}$) would be represented by the quadruple $(\mathtt{A}, \{\mathtt{E}\varphi\} \cup \Sigma, \mathtt{E}\varphi, 1)$. The other possible premiss would have index 0 instead. There are three exceptions to this: applications of rules ($\mathtt{Ett}$) and ($\mathtt{X_0}$) can be represented using a constant name, and the premiss in rule ($\mathtt{X_1}$) is entirely determined by one of the $\mathtt{E}$-blocks in the subgoal. Hence, let

$$\Sigma_\varphi^{\mathsf{br}} \;:=\; (\{\mathtt{A}, \mathtt{E}\} \times 2^{Sub(\varphi)} \times Sub(\varphi) \times \{0,1\}) \;\cup\; \{0,1\} \;\cup\; 2^{Sub(\varphi)}$$

Note that $|\Sigma_\varphi^{\mathsf{br}}| = 2^{\mathcal{O}(|\varphi|)}$.

An infinite branch $\pi = \mathcal{C}_0, \mathcal{C}_1, \ldots$ in a pre-tableau for φ then induces a word $\pi' = r_0, r_1, \ldots \in (\Sigma_\varphi^{\mathsf{br}})^\omega$ in a straight-forward way: r_i is the symbolic representation of the goal/subgoal pair (C_i, C_{i+1}). We will not distinguish formally between an infinite branch π and its induced ω-word π' over $\Sigma_\varphi^{\mathsf{br}}$.

Remember that we want to define an NBA which accepts exactly those branches which are not good, i.e. which either contain an E-trace with an U-thread or an A-trace with no R-thread. Nondeterminism can be used in order to guess the trace in the branch, and it can also be used in order to guess an U-thread in an E-trace. However, it is not necessarily useful for showing that no R-thread exists in an A-trace. We therefore use complementation for this subproblem again.

An A-trace-marked branch is a pre-tableau branch in which a single A-trace is marked. It can be represented as an infinite word over the alphabet $\Sigma_\varphi^{\mathsf{tmb}} = \Sigma_\varphi^{\mathsf{br}} \times 2^{Sub(\varphi)}$. The second component of the alphabet simply names the set of subformulas which form the current A-block on the marked trace. Then we define a co-Büchi automaton $\mathcal{C}_\varphi$ which recognises exactly those A-trace-marked branches which contain an R-thread in the marked trace. It is $\mathcal{C}_\varphi = (\{\mathtt{W}, \mathtt{F}\} \cup Sub(\varphi), \Sigma_\varphi^{\mathsf{tmb}}, \mathtt{W}, \delta, F)$ with $F = Sub(\varphi)$. We define the transition relation δ by intuitively describing its behaviour. Starting in the waiting state W it guesses a formula of the form $\psi_1 \mathtt{R} \psi_2$ which occurs in the marked A-trace. It then tracks this formula in its state for as long as it is unfolded with rule (AR) and remains in the marked trace. If it leaves the marked trace then $\mathcal{C}_\varphi$ moves into the failure state F. The following proposition is easily seen to be true.

Lemma 12. *Let $w \in (\Sigma_\varphi^{\mathsf{tmb}})^\omega$ be an A-trace-marked branch of a pre-tableau for φ. Then $w \in \mathcal{L}(\mathcal{C}_\varphi)$ iff the marked trace of w contains a R-thread.*

Remember that we are interested in branches whose A-traces do not contain R-threads. Hence, we need complementation. Luckily, an NcoBA can be determinised into a DcoBA using the Miyano-Hayashi construction [14] which can easily be complemented into a DBA according to Lemma 11.

Theorem 13 ([14]). *For every NcoBA $\mathcal{A}$ with n states there is a DBA $\overline{\mathcal{A}}$ with at most 3^n states s.t. $L(\overline{\mathcal{A}}) = \overline{L(\mathcal{A})}$.*

Equally, we can define an E-trace-marked branch as a word over $\Sigma_\varphi^{\mathsf{tmb}}$ and an NcoBA $\mathcal{B}_\varphi$ which accepts exactly those which contain an U-thread in the marked E-trace. It is $\mathcal{B}_\varphi := (\{\mathtt{W}, \mathtt{F}\} \cup Sub(\varphi), \Sigma_\varphi^{\mathsf{tmb}}, \mathtt{W}, \delta, F)$ with $F = Sub(\varphi)$. Its behaviour is almost the same as that of $\mathcal{C}_\varphi$ with the difference that it tracks an U-formula in its state component rather than an R-formula.

Lemma 14. *Let $w \in (\Sigma_\varphi^{\mathsf{tmb}})^\omega$ be an E-trace-marked branch of a pre-tableau for φ. Then $w \in \mathcal{L}(\mathcal{B}_\varphi)$ iff the marked trace of w contains an U-thread.*

Then we can define an NBA $\mathcal{A}_\varphi$ that accepts exactly those branches which contain a bad trace. Let $\overline{\mathcal{C}_\varphi} = (Q^{\mathcal{C}}, \Sigma_\varphi^{\mathsf{tmb}}, q_0^{\mathcal{C}}, \delta^{\mathcal{C}}, F^{\mathcal{C}})$ be the DBA obtained from $\mathcal{C}_\varphi$ using the complementation construction of Thm. 13, and $\mathcal{B}_\varphi = (Q^{\mathcal{B}}, \Sigma_\varphi^{\mathsf{tmb}}, q_0^{\mathcal{B}}, \delta^{\mathcal{B}}, F^{\mathcal{B}})$. Then define $\mathcal{A}_\varphi := (Q, \Sigma_\varphi^{\mathsf{br}}, \mathtt{W}, \delta, F)$ where $Q = \{\mathtt{W}, \mathtt{F}\} \cup 2^{Sub(\varphi)} \times \big(Q^{\mathcal{C}} \dot\cup$

$Q^{\mathcal{B}}$). Again, we describe its behaviour informally. It starts in the waiting state `W`. At some point it guesses a block that is contained in the given alphabet symbol and tracks this block in the first component of its state space in order to check that it is a non-spawing trace. Depending on whether or not it is an `A`- or `E`-block it simulates in its second component the automaton $\overline{\mathcal{C}_\varphi}$, resp. $\mathcal{B}_\varphi$ on the letters which are composed of the input letter and the first component. Thus, it effectively guesses a trace and simulates one of the two automata on the branch in which this trace is marked. If the trace disappears using rule (`Ett`) for instance, it moves to the failure state `F`. Its accepting states F are $2^{Sub(\varphi)} \times (F^{\mathcal{C}} \dot\cup F^{\mathcal{B}})$. The following is not too difficult to see using Lem. 12 and 14 as well as Thm. 13.

Lemma 15. *Let $w \in (\Sigma_\varphi^{\mathsf{br}})^\omega$ be a branch of a pre-tableau for φ. Then $w \in \mathcal{L}(\mathcal{A}_\varphi)$ iff w contains a trace which is not good.*

Furthermore, a close inspection of the constructions together which Thm. 13 yields the following estimation on the size of $\mathcal{A}_\varphi$. Note that the initial waiting states of $\mathcal{C}_\varphi$ and $\mathcal{B}_\varphi$ are redundant since waiting is also done in the initial state of $\mathcal{A}_\varphi$.

Proposition 16. *The number of states of $\mathcal{A}_\varphi$ is bounded by $2 + 2^{|\varphi|} \cdot (3^{|\varphi|+2} \cdot (|\varphi| + 2)) \leq 2^{\mathcal{O}(|\varphi|)}$.*

Finally, remember that we need a deterministic automaton recognising the complement of the language recognised by $\mathcal{A}_\varphi$. Luckily, there are determinisation constructions for Büchi automata. We are particularly interested in those that yield parity automata [15, 12, 20].

Theorem 17 ([15]). *For every NBA with n states there is an equivalent DPA with at most n^{2n+2} states and index at most $2n - 1$.*

Together with Lemma 11 we obtain a DPA $\overline{\mathcal{A}}_\varphi$ which accepts exactly those branches containing good traces only, and has size $2^{2^{\mathcal{O}(|\varphi|)}}$ and index $2^{\mathcal{O}(|\varphi|)}$.

4.3 The Reduction to Parity Games

The problem of deciding the existence of a tableau can easily be phrased as a game: starting with the initial goal $\mathtt{E}\varphi$, the *proponent* chooses a rule instance that can be applied to the current goal, and the *opponent* chooses a subgoal whenever the instance is a branching rule. Note this is only the case for the modal rules. This yields a pre-tableau branch in the limit. The proponent wins iff all traces on this branch are good, otherwise the opponent wins. Clearly, there is a tableau for φ iff the proponent has a winning strategy in this game. We will now use the automata-theoretic machinery of the previous subsection in order to formalise this game and present a reduction of the satisfiability problem for CTL* to the problem of solving a parity game.

A *parity game* is a $\mathcal{G} = (V, V_0, V_1, v_0, E, \Omega)$ s.t. (V, E) is a finite, directed graph with total edge relation E, V_0, V_1 is a partition of the node set V into

nodes owned by player 0 and 1, resp., $v_0 \in V$ is a designated starting node, and $\Omega : V \to \mathbb{N}$ assigns priorities to the nodes. A *play* is an infinite sequence $v_0, v_1, \ldots$ starting in v_0 s.t. $(v_i, v_{i+1}) \in E$ for all $i \in \mathbb{N}$. It is won by player 0 if $\max\{\Omega(v) \mid v = v_i \text{ for infinitely many } i\}$ is even. A *(non-positional) strategy* for player i is a function $\sigma : V^*V_i \to V$, s.t. for all sequences $v_0 \ldots v_n$ with $(v_j, v_{j+1}) \in E$ for all $j = 0, \ldots, n-1$, and all $v_n \in V_i$ we have: $(v_n, \sigma(v_0 \ldots v_n)) \in E$. A play $v_0 v_1 \ldots$ *conforms* to a strategy σ for player i if for all $j \in \mathbb{N}$ we have: if $v_j \in V_i$ then $v_{j+1} = \sigma(v_0 \ldots v_j)$. A strategy σ for player i is a *winning strategy* in node v if player i wins every play that begins in v and conforms to σ. A *(positional) strategy* for player i is a strategy σ for player i s.t. for all $v_0 \ldots v_n \in V^*V_i$ and all $w_0 \ldots w_m \in V^*V_i$ we have: if $v_n = w_m$ then $\sigma(v_0 \ldots v_n) = \sigma(w_0 \ldots w_m)$. Hence, we can identify positional strategies with $\sigma : V_i \to V$. It is a well-known fact that for every node $v \in V$, there is a winning strategy for either player 0 or player 1 for node v. In fact, parity games enjoy positional determinancy meaning that there is even a positional winning strategy for node v for one of the two player [2]. The problem of *solving* a parity game is to determine which player has a winning strategy for v_0. It is solvable [19] in time polynomial in $|V|$ and exponential in $|\Omega[V]|$.

Definition 18. *Let φ be a state formula and $\overline{\mathcal{A}}_\varphi = (Q, \Sigma_\varphi^{\mathsf{br}}, q_0, \delta, \Omega)$ be the DPA according to the previous subsection which recognises good branches in pre-tableaux for φ. The* satisfiability game *for φ is a parity game $\mathcal{G}_\varphi = (V, V_0, V_1, v_0, E, \Omega')$ defined as follows.*

- $V := Seq(\varphi) \times Q$
- $V_1 := \{(C, q) \in V \mid$ *rule* $(\mathtt{X_0})$ *or* $(\mathtt{X_1})$ *applies to* $C\}$
- $V_0 := V \setminus V_1$
- $v_0 := (\mathtt{E}\varphi, q_0)$
- $((C, q), (C', q')) \in E$ *iff* (C, C') *is an instance of a rule application which is symbolically represented by* $r \in \Sigma_\varphi^{\mathsf{br}}$ *and* $q' = \delta(q, r)$*, or no rule is applicable to* C *and* $C = C'$ *and* $q = q'$*,*
- $\Omega'(C, q) := \begin{cases} 0 & \text{if } C \text{ is a consistent set of literals} \\ \Omega(q) & \text{if there is a rule applicable to } C \\ 1 & \text{otherwise} \end{cases}$

The following theorem states correctness of this construction. It is not difficult to prove. In fact, a winning strategy for player 0 is basically a finite representation of an infinite tableau.

Theorem 19. *Player 0 wins $\mathcal{G}_\varphi$ iff there is a tableau for φ.*

Proof. Assume that player 0 wins $\mathcal{G}_\varphi$ with a positional winning strategy σ. Unfolding the game $\mathcal{G}_\varphi$ starting with v^* and conforming to σ results in a possibly infinite tree that can be easily transformed into a pre-tableau P for φ by removing all annotations of the branch-checking automaton and by replacing all consistent-set-loops with consistent-set-leafs. Note that it is impossible that a finite branch does not end in a consistent set with player 0 winning from v^*. Given an arbitrary

infinite branch π in P, it holds that $\pi \in \mathcal{L}(\overline{\mathcal{A}}_\varphi)$, hence by Lemma 15 it follows that π contains no bad trace. Consequently, P is a tableau.

For the other direction, let P be a tableau for φ. Starting with v^*, every goal in P can be labeled with the corresponding game state; then, every player 0 position of $\mathcal{G}_\varphi$ corresponding to a node in the labeled version of P can be used as a non-positional strategy decision for player 0. The player 0 strategy obtained in that manner is indeed a winning strategy for player 0 starting in v^*: let π be an arbitrary play conforming to the strategy; if π is finite, it must correspond to a branch in P that ends in a consistent set of literals, hence it is won by player 0, otherwise the branch corresponding to π contains only good traces, and hence by Lemma 15 it follows that π is won by player 0. □

The proofs of the following corollaries are given in the appendix.

Corollary 20. *Deciding existence of a tableau for some φ is in 2EXPTIME.*

Corollary 21. *Any satisfiable* CTL* *formula φ has a model of size at most $2^{2^{\mathcal{O}(|\varphi|)}}$ and branching-width at most $2^{|\varphi|}$.*

5 Comparison with Existing Methods

We briefly compare the tableau/automata-based reduction to parity games with existing decision procedures for CTL*, namely Emerson/Jutla's tree automata [5], Reynolds' proof system [17], Gabbay/Pnueli's proof system [9], and Reynolds' tableaux [18].

Emerson/Jutla's procedure transforms a CTL* φ formula in some normal form into a tree-automaton recognising exactly the tree-unfoldings of fixed branching-width of all models of φ. This uses a translation of linear-time formulas into Büchi automata and then into deterministic (Rabin) automata for the same reasons as outlined above. This has a drawback, as Emerson [3, Sec. 6.5] notes himself: "*... due to the delicate combinatorial constructions involved, there is usually no clear relationship between the structure of the automaton and the candidate formula.*"

Note that our approach does not use tree-automata as such—even though one may argue that the constructed parity games represent tree automata. However, the crucial difference is the separation between the use of tableau-machinery for the characterisation of satisfiability in CTL* and the use of automata-machinery only in order to obtain a decision procedure. In particular, we do not need translations of linear-time temporal formula into ω-word automata. The relationship between input formula and resulting structure (here: game) is given by the tableau rules. Furthermore, this separation makes a huge difference in practice, as we believe, because it allows the branching-width of models of φ to be flexible. Note that this is given by the number of premisses of rule ($\mathtt{X}_1$), whereas in Emerson/Jutla's approach it is fixed a priori to a number which is linear in the size of the input formula. While this does not increase the asymptotic worst-case complexity, it does have an effect on the efficiency in practice. Not

surprisingly, we do not know of any attempt to implement the tree-automata approach.

Reynolds' proof system is an approach at giving a sound and complete finite axiomatisation for CTL*. Its proof of correctness is rather intricate and the system itself is useless for practical purposes since it uses a second-order rule and it is therefore not even clear how a decision procedure, i.e. proof search could be done. In comparison, our calculus has the subformula property and comes with an implementable decision procedure. The only price to pay for this is the characterisation of satisfiability through infinite objects instead.

Gabbay/Pnueli's proof system is a unifying approach to compositional model checking and validity checking. Their work focuses on obtaining a sound and complete system. It is not clear whether this could be used in practice and we also do not know of any implementation based on that calculus. Also, the system is only complete for a special model of reactive systems in which path quantifiers are implicitly relativised.

Reynold's recent tableau system shares some similarities with our tableau system. He also uses sets of sets of formulas as well as traces (which he calls threads), etc. Even though his tableaux are finite, the difference in this respect is marginal. Finiteness is obtained through looping back, i.e. those branches might be called infinite as well. One of the real differences between the two systems lies in the way that the semantics of the CTL* operators shows up. In Reynolds' system it translates into technical requirements on nodes in the tableaux, whereas our system comes with relatively straight-forward tableau rules. The other main difference is the loop-check. Reynolds says that "*... we are only able to give some preliminary results on mechanisms for tackling repetition.* [...] *The task of making a quick and more generally usable repetition checker will be left to be advanced and presented at a later date.*" Our method comes with a non-trivial repetition checker: it is given by the annotated automata. Finally, Reynolds reports of a prototype implementation of his tableau decision procedure [18]. This implementation is, however, not publicly available, and tests are only performed on single short formulas such that no asymptotic behaviour can be inferred from those results. We strongly believe that this implementation is greatly outperformed by ours. For example, the formula $\mathtt{AG}(\mathtt{EX}p \wedge \mathtt{EX}\neg p) \wedge \mathtt{AG}(\mathtt{G}p \vee (\neg r)\mathtt{U}(r \wedge \neg p))$ apparently cannot be checked for satisfiability by Reynolds' implementation anymore whereas ours takes 0.04s for this task.

6 An Implementation

We report on practical aspects of the decision procedure described above. As said in the introduction, it is implemented in the MLSolver tool, a framework for solving satisfiability and validity problems of modal fixpoint logics. It reduces such problems to parity games and then uses PGSolver, a high-performance solver for parity games. Both tools are publicly available[1].

[1] `http://www.tcs.ifi.lmu.de/{mlsolver,pgsolver}`

Optimisations. (1) It is possible to partially *determinise the proponent's strategy* without compromising on soundness or completeness: except for the modal rules ($\mathtt{X_0}$) and ($\mathtt{X_1}$), it is not important which rule is to be applied next. Instead of allowing the proponent to choose the rule we use a function which determines for each goal the rule that has to be applied to it next. This leaves the proponent with the choices of the disjuncts to be preserved in the current goal in rules ($\mathtt{A1}$), ($\mathtt{AA}$), ($\mathtt{AE}$), ($\mathtt{E}\vee$), ($\mathtt{EU}$), and ($\mathtt{ER}$) and reduces the out-degree of the resulting parity games.

(2) MLSolver allows parity games to be generated in *compact mode*. This means that not every pair of pre-tableau goal and automaton state is present in the game. Instead, the game only contains those pairs in which rule ($\mathtt{X_0}$) or ($\mathtt{X_1}$) applies to the pre-tableau goal. This is possible because CTL* formulas are guarded in the sense that every infinite pre-tableau path must contain infinitely many applications of one of these rules. Compact mode does not only create much smaller games, they are also often generated faster because loop-checks do not need to be performed for every newly generated pair.

(3) MLSolver is able to perform *literal propagation* in each step of the creation of a pre-tableau. This means that whenever a literal becomes top-level in a goal, its other occurrences which are not under the scope of a temporal operator are replaced by $\mathtt{tt}$. Equally, all such occurrences of the complement literal are replaced by $\mathtt{ff}$. The resulting goal can be simplified according to the usual rules for boolean operations and then provide less disjunctive choices or allow to detect inconsistencies earlier.

(4) MLSolver prefers large formulas as principals. This scheduling reduces the branching width to linear—in contrast to the general case, cf. Cor. 21. A proof is given in the appendix.

(5) Finally, PGSolver contains implementations of basically all known algorithms for solving parity games. While some of them are consistently bad in practice, there are some which perform quite well even though none of them is always best. These are furthermore aided by modules performing simplifications on the parity games which speed up the solving. The running times reported below are obtained using the solving algorithm which is best on the respective instance—usually the one by Stevens and Stirling [21].

Benchmarks. (1) We consider two simple families of formulas that feature deep nestings of modal operators. Let $\alpha_0 := q$, $\alpha_{n+1} := \mathtt{AFG}\alpha_n$, $\beta_0 := q$, $\beta_{n+1} := \mathtt{AFAG}\beta_n$, $\psi_n := \alpha_n \to \beta_n$, and $\varphi_n := \beta_n \to \alpha_n$ for $n \geq 0$. Both families are checked for validity, but note that ψ_n is falsifiable whereas φ_n is valid. Thus, there is a tableau for $\neg\psi_n$ but none for $\neg\varphi_n$.

(2) We consider $n+1$ programs $0, \ldots, n$. A proposition p_i states whether or not the program i is running. A scheduler is assumed which guarantees that at any time at least one program is running, and that each program runs infinitely often. Then for any execution sequence and at any time, if program 0 is running then the programs 1 to n will run in this order but possibly interrupted by others. The hole setting is given by the formula $(\mathtt{AG}(\bigvee_i p_i) \wedge \bigwedge_i \mathtt{AGF}p_i) \;\to\; \mathtt{AG}(p_0 \to \tau_1)$ where $\tau_i = \mathtt{F}(p_i \wedge \tau_{i+1})$ and $\tau_{n+1} = \mathtt{tt}$.

Family		n	Explored Game Size	Time
Nested Modal Operators	ψ_n	14	102,200	727.55s
		15	123,774	1,315.55s
		16	148,213	1,663.43s
		17	175,697	3,173.65s
		18	?	†
	φ_n	2	400	0.25s
		3	5,581	10.18s
		4	?	†

Family		n	Explored Game Size	Time
Scheduler		1	81	0.08s
		2	852	1.21s
		3	12,320	30.82s
		4	?	†
Limit Closure	α_n	1	49	0.12s
		2	7,213	328.15s
		3	?	†
	β_n	1	49	0.12s
		2	?	†

Fig. 2. Runtime results.

(3) The formula $\lambda(\varphi, \psi, \varphi', \psi') := (\varphi \wedge \mathtt{AG}(\varphi \rightarrow \mathtt{EX}(\psi \mathtt{U} \varphi))) \rightarrow \mathtt{EG}(\psi' \mathtt{U} \varphi')$ is a variation of the limit closure property, and is a tautology if φ' is a consequence of φ and ψ' of ψ. Its iterations serve as benchmarks, that are

$$\alpha_0 := q \rightarrow q \text{ and } \alpha_{n+1} := \lambda(p, \varphi, p, \varphi') \text{ where } \varphi \rightarrow \varphi' = \alpha_n, \text{ and}$$
$$\beta_0 := p \rightarrow p \text{ and } \beta_{n+1} := \lambda(\varphi, q, \varphi', q) \text{ where } \varphi \rightarrow \varphi' = \beta_n.$$

Experimental Results. All tests have been carried out on a 64-bit machine with four quad-core Opteron™ CPUs and 128 GB RAM space. The implementation does not (yet) support parallel computations, hence, each test runs on one core only and needed less than 4 GB RAM. We only present instances of non-negligible running times. On the other hand, the solving of larger instances not presented in Fig. 2 anymore has experienced time-outs after one hour, marked †.

7 Further Work

The results of the previous section show that the tableau/automata approach to deciding CTL* is reasonably viable in practice. Note that the implementation so far only features optimisations on one of three fronts: it uses the latest and optimised technology for solving the resulting games. However, there are two more fronts for optimisations which have not been exploited so far. The main advantage of this approach is—as we believe—the combination of tableau-, automata- and game-machinery and therefore the possible benefit from optimisation techniques in any of these areas. It remains to be seen for instance whether the automaton determinisation procedure can be improved or replaced by a better one. Also, the tableau community has been extremely successful in speeding up tableau-based procedures using various optimisations. It also remains to be seen how those can be incorporated in the combined method.

Furthermore, it remains to expand this work to extensions of CTL*, for example CTL* with past operators, multi-agent logics based on CTL*, etc.

References

1. M. Dam. CTL* and ECTL* as fragments of the modal μ-calculus. *TCS*, 126(1):77–96, 1994.
2. E. Emerson and C. Jutla. Tree automata, μ-calculus and determinacy. In *Proc. 32nd Symp. on Foundations of Computer Science*, pages 368–377, San Juan, 1991. IEEE.
3. E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, chapter 16, pages 996–1072. Elsevier and MIT Press, New York, USA, 1990.
4. E. A. Emerson and J. Y. Halpern. "Sometimes" and "not never" revisited: On branching versus linear time temporal logic. *J. of the ACM*, 33(1):151–178, 1986.
5. E. A. Emerson and C. S. Jutla. The complexity of tree automata and logics of programs. *SIAM Journal on Computing*, 29(1):132–158, 2000.
6. E. A. Emerson and A. P. Sistla. Deciding full branching time logic. *Information and Control*, 61(3):175–201, 1984.
7. O. Friedmann and M. Lange. A solver for modal fixpoint logics. In *Proc. 6th Workshop on Methods for Modalities, M4M-6*, 2009.
8. O. Friedmann and M. Lange. Solving parity games in practice. In *Proc. 7th Int. Symp. on Automated Technology for Verification and Analysis, ATVA'09*, volume 5799 of *LNCS*, pages 182–196, 2009.
9. D. M. Gabbay and A. Pnueli. A sound and complete deductive system for CTL* verification. *Logic Journal of the IGPL*, 16(6):499–536, 2008.
10. E. Grädel. Guarded fixed point logics and the monadic theory of countable trees. *Theor. Comput. Sci*, 288(1):129–152, 2002.
11. E. Grädel, W. Thomas, and Th. Wilke, editors. *Automata, Logics, and Infinite Games*, LNCS. Springer, 2002.
12. D. Kähler and Th. Wilke. Complementation, disambiguation, and determinization of Büchi automata unified. In *Proc. 35th Int. Coll. on Automata, Languages and Programming, ICALP'08*, volume 5125 of *LNCS*, pages 724–735. Springer, 2008.
13. X. Luo, K. Su, A. Sattar, Q. Chen, and G. Lv. Bounded model checking knowledge and branching time in synchronous multi-agent systems. In *Proc. 4th Int. Conf. on Auton. Agents and Multiagent Syst., AAMAS'05*, pages 1129–1130. ACM, 2005.
14. S. Miyano and T. Hayashi. Alternating finite automata on omega-words. *TCS*, 32(3):321–330, 1984.
15. N. Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. In *Proc. 21st Symp. on Logic in Computer Science, LICS'06*, pages 255–264. IEEE Computer Society, 2006.
16. A. Pnueli and R. Rosner. A framework for the synthesis of reactive modules. In *Proc. Int. Conf. on Concurrency*, volume 335 of *LNCS*, pages 4–17. Springer, 1988.
17. M. Reynolds. An axiomatization of full computation tree logic. *Journal of Symbolic Logic*, 66(3):1011–1057, 2001.
18. M. Reynolds. A tableau for CTL*. In *Proc. 16th. Int. Symp. on Formal Methods, FM'09*, volume 5850 of *LNCS*, pages 403–418. Springer, 2009. Long version availabe as technical report of the University of Western Australia.
19. S. Schewe. Solving parity games in big steps. In *Proc. 27th Int. Conf. on Foundations of Software Technology and Theoretical Computer Science, FSTTCS'07*, volume 4855 of *LNCS*, pages 449–460. Springer, 2007.
20. S. Schewe. Tighter bounds for the determinisation of Büchi automata. In *Proc. 12th Int. Conf. on Foundations of Software Science and Computation Structures, FOSSACS'09*, volume 5504 of *LNCS*, pages 167–181. Springer, 2009.

21. P. Stevens and C. Stirling. Practical model-checking using games. In *Proc. 4th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'98*, volume 1384 of *LNCS*, pages 85–101. Springer, 1998.
22. M. Y. Vardi and L. Stockmeyer. Improved upper and lower bounds for modal logics of programs. In *Proc. 17th Symp. on Theory of Computing, STOC'85*, pages 240–251, Baltimore, USA, 1985. ACM.

A Soundness

Theorem 22 (Soundness). *Let P a tableau for $\vartheta \in \mathrm{CTL}^*$. Then ϑ is satisfiable.*

Proof. Let $\mathcal{V}$ be the nodes in P, and $\mathcal{S}$ those nodes which are leaves or which are conclusions of the rules $(\mathtt{X_0})$ or $(\mathtt{X_1})$. Let $\hat{s} : \mathcal{V} \to \mathcal{S}$ be the function which points a node v to its oldest descendants—including v—in $\mathcal{S}$. Since all rules are unary besides $(\mathtt{X_1})$, Lemma 4 ensures that the function $\hat{s}$ is well-defined. For r being the root of P, we set $s^\star := \hat{s}(r)$. Moreover, the infix relation $\to \in \mathcal{S} \times \mathcal{S}$ is defined as $\{(s, \hat{s}(t)) \mid t \text{ is a child of } s \text{ in } P\} \cup \{(s, s) \mid s \text{ is a leaf in } P\}$.

The tableau induces the transition system $\mathcal{T}_\vartheta = (\mathcal{S}, s^\star, \to, \ell)$ such that $\ell(s) = \mathcal{C} \cap \mathcal{P}$ for any $s \in S$ labeled with a goal $\mathcal{C}$. Note that $\mathcal{T}_\vartheta$ is total. In the following, we omit the transition system $\mathcal{T}_\vartheta$ in the context of "$\models$". Moreover, we identify a node with its annotated goal.

We claim that $\mathcal{T}_\vartheta, s^\star \models \vartheta$. For the sake of a contradiction, assume that this is not the case. We will show that the tableau has a branch which contains a bad trace. For this purpose, we construct a trace $\Xi = (Q_i \Gamma_i)_{i \in \mathbb{N}}$ in a branch $(\mathcal{C}_i)_{i \in \mathbb{N}}$ of P and a partial sequence π_i of paths in $\mathcal{T}_\vartheta$ such that the following properties hold for all $i \in \mathbb{N}$.

(1) $\mathcal{C}_0$ is the root of P, and $\mathcal{C}_{i+1}$ is a child of $\mathcal{C}_i$.
(2) $Q_i \Gamma_i \in \mathcal{C}_i$.
(3) $(\mathcal{C}_i, Q_i \Gamma_i) \rightsquigarrow (\mathcal{C}_{i+1}, Q_{i+1} \Gamma_{i+1})$.
(4) If $Q_i = \mathtt{E}$ then $\hat{s}(\mathcal{C}_i) \not\models \mathtt{E}(\bigwedge_{\gamma \in \Gamma_i} \gamma)$.
(5) If $Q_i = \mathtt{A}$ then π_i is defined, $\hat{s}(\mathcal{C}_i) = \pi_i(0)$, and $\pi_i \not\models \bigvee_{\gamma \in \Gamma_i} \gamma$.
(6) If $Q_i = Q_{i+1} = \mathtt{A}$, and $\mathcal{C}$ is an instance of $(\mathtt{X_0})$ or $(\mathtt{X_1})$ then $\pi_{i+1} = \pi_i^1$.
(7) If $Q_i = Q_{i+1} = \mathtt{A}$, and $\mathcal{C}$ is an instance of neither $(\mathtt{X_0})$ nor $(\mathtt{X_1})$ then $\pi_{i+1} = \pi_i$.
(8) If $Q_i \Gamma_i = \mathtt{A}(\varphi \mathtt{R} \psi, \Sigma)$, if $\mathcal{C}_i$ is the conclusion of $(\mathtt{AR})$ such that $\varphi \mathtt{R} \psi$ and $Q_i \Gamma_i$ are principal, and if $Q_{i+1} \Gamma_{i+1} = \mathtt{A}(\varphi, \mathtt{X}(\varphi \mathtt{R} \psi), \Sigma)$ then $\pi_i \models \psi$.

The construction of such a trace is straight forward. We detail the proof for some cases, and thereto use formulas and notations as shown in Fig. **??**. As for the rule $(\mathtt{EA})$, if $\hat{s}(\mathcal{C}_i) \not\models \mathtt{E}(\mathtt{A}\varphi \wedge \bigwedge_{\gamma \in \Pi} \gamma)$ then $\hat{s}(\mathcal{C}_i) \not\models \mathtt{A}\varphi$ or $\hat{s}(\mathcal{C}_i) \not\models \mathtt{E}(\bigwedge_{\gamma \in \Pi} \gamma)$. In the first case the trace is continued with $\mathtt{E}\Pi$. Otherwise, $Q_{i+1} \Gamma_{i+1} = \mathtt{A}\varphi$ and π_{i+1} is an arbitrary path in $\mathcal{T}_\vartheta$ which starts at $\hat{s}(\mathcal{C}_i) = \hat{s}(\mathcal{C}_{i+1})$ and which fulfills $\pi_{i+1} \not\models \varphi$. As for the rule $(\mathtt{AR})$, we have $\pi_i \not\models \varphi \mathtt{R} \psi \vee \bigvee_{\gamma \in \Sigma} \gamma$. Using that $\pi_i = \pi_{i+1}$ and an unrolling of the $\mathtt{R}$-operator, $\pi_{i+1} \not\models \psi$ or $\pi_{i+1} \not\models \varphi \vee \mathtt{X}(\varphi \mathtt{R} \psi)$. In the first case the trace is continued with $\mathtt{A}(\psi, \Sigma)$, and with $\mathtt{A}(\varphi, \mathtt{X}(\varphi \mathtt{R} \psi), \Sigma)$ otherwise. As for case of $(\mathtt{X_0})$ and $(\mathtt{X_1})$, the constraints determine the successor uniquely.

Back to the main proof, Ξ is either an $\mathtt{E}$- or an $\mathtt{A}$-trace, by Lemma 6.

Case: Ξ is an $\mathtt{E}$-trace. We show that Ξ is a bad trace by revealing an $\mathtt{U}$-thread in Ξ. Let $i^\star \in \mathbb{N}$ such that $Q_i = \mathtt{E}$, for all $i \geq i^\star$. By π we denote the subsequence of the branch $(\mathcal{C}_i)_{i \geq i^\star}$ which consists of nodes in $\mathcal{S}$ only. For a

node $\mathcal{C}$ in the branch, we write $\pi^{\mathcal{C}}$ to denote the subsequence of π starting at $\hat{s}(\mathcal{C})$. A simple case distinction on the used rule yields the following property.

$$\left.\begin{array}{l}\text{Let } i \geq i^\star, \text{ and let } \gamma \in \Gamma_i \text{ with } \pi^{\mathcal{C}_i} \not\models \gamma. \text{ Then there exists a } \gamma' \in \Gamma_{i+1} \\ \text{such that } (\mathcal{C}_i, Q_i\Gamma_i, \gamma) \rightsquigarrow (\mathcal{C}_{i+1}, Q_{i+1}\Gamma_{i+1}, \gamma') \text{ and } \pi^{\mathcal{C}_{i+1}} \not\models \gamma'.\end{array}\right\} \quad (\dagger)$$

By the property (4), there is a $\gamma \in \Gamma_{i^\star}$ such that $\pi^{\hat{s}(\mathcal{C}_{i^\star})} \not\models \gamma$. Using previous property (†), any connected finite sequence of formulas starting with γ can be expanded ad infinitum. However, there is a certain degree of freedom. Recall that any such connected infinite sequence of formulas is a thread. Fix such a thread $\xi = (\xi_i)_{i \geq i^\star}$ starting at γ with the following property: For any $i \geq i^\star$, if $\mathcal{C}_i$ is the conclusion of the rule ($\mathtt{ER}$) such that $Q_i\Gamma_i$ and $\xi_i = \varphi \mathtt{R} \psi$ are principal, and if $\pi^{\mathcal{C}_i} \not\models \psi$ then $\xi_{i+1} = \psi$.

By Lemma 7, ξ is either an $\mathtt{U}$- or an $\mathtt{R}$-thread. In the first case, we are done as ξ witnesses that the considered branch contains Ξ as bad trace. So, suppose for the sake of a contradiction that ξ is an $\mathtt{R}$-thread.

By Lemma 8, there are $i^{\star\star} \geq i^\star$ and $\varphi, \psi \in \mathit{Sub}(\vartheta)$ such that $\xi_i = \varphi \mathtt{R} \psi$ or $\xi_i = \mathtt{X}(\varphi \mathtt{R} \psi)$ for all $i \geq i^{\star\star}$. Along the branch $(\mathcal{C}_i)_{i \geq i^{\star\star}}$, between any two consecutive applications of the rules ($\mathtt{X_0}$) or ($\mathtt{X_1}$), the rule ($\mathtt{ER}$) must have been applied such that $\xi_i = \varphi \mathtt{R} \psi$ and $Q_i\Gamma_i$ are principal for some $i \geq i^{\star\star}$. By the definition of a thread and choice of $i^{\star\star}$, the following element, $Q_{i+1}\Gamma_{i+1}$, of the trace is $\mathtt{E}(\psi, \mathtt{X}(\varphi \mathtt{R} \psi), \Pi)$ for some $\Pi \subseteq \mathit{Sub}(\vartheta)$. By choice of ξ, we have $\pi^{\mathcal{C}_i} \models \psi$ and $\xi_{i+1} = \mathtt{X}(\varphi \mathtt{R} \psi)$. Since this is true for any such two consecutive applications, $\pi^{\mathcal{C}_i} \models \psi$ for all $i \geq i^{\star\star}$. Therefore, $\pi^{\mathcal{C}_{i^{\star\star}}}$ also models $\varphi \mathtt{R} \psi$ and $\mathtt{X}(\varphi \mathtt{R} \psi)$, in particular. But, this is a contradiction to the choice of $i^{\star\star}$ and the construction of ξ.

Case: Ξ is an $\mathtt{A}$-trace. We show that Ξ is a bad trace. Suppose for the sake of a contradiction that Ξ contains an $\mathtt{R}$-thread $(\xi_i)_{i \in \mathbb{N}}$. Let $i^\star \in \mathbb{N}$ and $\varphi, \psi \in \mathit{Sub}(\vartheta)$ such that $Q_i = \mathtt{A}$, and $\xi_i = \varphi \mathtt{R} \psi$ or $\xi_i = \mathtt{X}(\varphi \mathtt{R} \psi)$ for all $i \geq i^\star$, cf. Lemma 8.

Along the branch $(\mathcal{C}_i)_{i \geq i^\star}$, between any two consecutive applications of the rules ($\mathtt{X_0}$) or ($\mathtt{X_1}$), the rule ($\mathtt{AR}$) must have been applied such that $\xi_i = \varphi \mathtt{R} \psi$ and $Q_i\Gamma_i$ are principal for some $i \in \mathbb{N}$. By the definition of a thread and choice of $i^\star$, the following element, $Q_{i+1}\Gamma_{i+1}$, of the trace is $\mathtt{A}(\varphi, \mathtt{X}(\varphi \mathtt{R} \psi), \Sigma)$ for some $\Sigma \subseteq \mathit{Sub}(\vartheta)$. Hence, thanks to (8) we have $\pi_i \models \psi$. Because the block quantifier remains $\mathtt{A}$, the properties (6) and (7) yield that $\pi_{i^\star}^j \models \psi$ for all $j \in \mathbb{N}$. Therefore, $\pi_{i^\star} \models \varphi \mathtt{R} \psi$ and $\pi_{i^\star} \models \mathtt{X}(\varphi \mathtt{R} \psi)$ hold. But this is a contradiction to (5). Thus, the considered branch contains Ξ as a bad trace. □

B Completeness

In order to show completeness, we need to assume a fixed well-ordering on the states of a model of some φ. Such an ordering trivially exists for finite models, and it is well-known that CTL* has the finite model property. However, using this result is cheating because the finite model property usually *follows* from

completeness and soundness of a system. However, it should be clear that such an ordering always exists for a model of countable size. Furthermore, CTL* can be embedded into least fixpoint logic LFP via a translation to the modal μ-calculus [1], and LFP possesses the Löwenheim-Skolem property, i.e. every satisfiable LFP formula has a countable model [10].

For any transition system $\mathcal{T} = (\mathcal{S}, s^*, \to, \lambda)$ let $\prec_\mathcal{T}$ denote an arbitrary but fixed well-ordering on $\mathcal{S}$. We extend $\prec_\mathcal{T}$ to a well-ordering $\lhd_\mathcal{T}$ on paths over $\mathcal{S}$ as follows: $\pi \lhd_\mathcal{T} \pi'$ holds iff there is an i s.t. $\pi(i) \prec_\mathcal{T} \pi'(i)$ and for each $j < i$ it holds that $\pi(j) = \pi'(j)$. It is well-known that the lexicographical extension of well-orderings to infinite sequences is again a well-ordering.

Lemma 23. *Given a transition system $\mathcal{T}$, the relation $\lhd_\mathcal{T}$ is a well-ordering on paths over $\mathcal{S}$.*

Let $\mathcal{T} = (\mathcal{S}, s^*, \to, \lambda)$ be a transition system, $s \in \mathcal{S}$ be a state and ψ be a formula s.t. $s \models \mathtt{E}\psi$. The *minimal s-rooted path that satisfies ψ* is denoted by $\xi_\mathcal{T}(s, \psi)$ and fulfills the following properties: $\xi_\mathcal{T}(s, \psi)(0) = s$, $\xi_\mathcal{T}(s, \psi) \models \psi$ and there is no path π with $\pi \lhd_\mathcal{T} \xi_\mathcal{T}(s, \psi)$ and $\pi(0) = s$ s.t. $\pi \models \psi$.

A *T-labeled (pre-)tableau* is a (pre-)tableau with every goal being labeled with a state s.t. the root is labeled with s^*, and for every s-labeled goal and every s'-labeled immediate subgoal it holds that $s \to s'$ if the corresponding rule application is $(\mathtt{X_1})$ or $(\mathtt{X_0})$ and $s = s'$ otherwise.

Theorem 24 (Completeness). *Let $\vartheta \in$ CTL* be satisfiable. Then there is a tableau for ϑ.*

Proof. Let ϑ be a formula and $\mathcal{T} = (\mathcal{S}, s^*, \to, \lambda)$ be a transition system and $s^* \in \mathcal{S}$ be a state s.t. $s^* \models \mathtt{E}\vartheta$.

We inductively construct a $\mathcal{T}$-labeled pre-tableau as follows. Starting with the labeled sequence $s^* : \mathtt{E}\vartheta$, we apply the rules in an arbitrary but eligible ordering systematically backwards by preserving $s \models \Phi$ for every state-labeled goal $s : \Phi$ as well as the following additional properties.

1. If the rule application to follow Φ is $(\mathtt{Al})$, $(\mathtt{AE})$ or $(\mathtt{AA})$, with $\mathtt{A}(\psi, \Gamma)$ being the principal block in Φ and ψ being the principal (state) formula, and $s \models \psi$, then the immediate subgoal of Φ follows ψ and discards the original $\mathtt{A}$-block.
2. If the rule application to follow Φ is $(\mathtt{EU})$, with $\mathtt{E}(\varphi\mathtt{U}\psi, \Sigma)$ being the principal block in Φ and $\varphi\mathtt{U}\psi$ being the principal formula, and $\xi_\mathcal{T}(s, (\varphi\mathtt{U}\psi, \Sigma)) \models \psi$, then the immediate subgoal of Φ follows ψ (instead of φ and $\mathtt{X}(\varphi\mathtt{U}\psi)$).
3. If the rule application to follow Φ is $(\mathtt{ER})$, with $\mathtt{E}(\varphi\mathtt{R}\psi, \Sigma)$ being the principal block in Φ and $\varphi\mathtt{R}\psi$ being the principal formula, and $\xi_\mathcal{T}(s, (\varphi\mathtt{R}\psi, \Sigma)) \models \varphi$, then the immediate subgoal of Φ follows ψ, φ (instead of $\psi, \mathtt{X}(\varphi\mathtt{R}\psi)$).
4. If the rule application to follow Φ is $(\mathtt{E}\vee)$, with $\mathtt{E}(\psi_1 \vee \psi_2, \Sigma)$ being the principal block in Φ and $\psi_1 \vee \psi_2$ being the principal formula, and $\xi_\mathcal{T}(s, (\psi_1 \vee \psi_2, \Sigma)) \models \psi_i$ for some $i \in \{1, 2\}$, then the immediate subgoal of Φ follows ψ_i.
5. If the rule application to follow Φ is $(\mathtt{X_1})$, with subgoals $s_1 : \mathtt{E}\Sigma_1, \Phi_1, \ldots$ $s_k : \mathtt{E}\Sigma_k, \Phi_k$, then $\xi_\mathcal{T}(s, \mathtt{X}\Sigma_i)(1) = s_i$ for every $1 \leq i \leq k$.

Consider that this construction indeed yields a pre-tableau with each state-labeled sequence $s : \Phi$ satisfying all side conditions. Moreover note that every finite branch ends in a node labeled with consistent literals only.

By contradiction assume that the pre-tableau is not a tableau, hence there is a labeled branch $s_0 : \Phi_0, s_1 : \Phi_1, \ldots$ (with $\Phi_0 = \mathtt{E}\vartheta$) containing a bad trace $B_0, B_1, \ldots$. We define a *lift operation* $\hat{i}$ that selects the next modal rule application as follows.

$$\hat{i} := \min\{j \geq i \mid \Phi_j \text{ is the goal of an application of } (\mathtt{X_1}) \text{ or } (\mathtt{X_0})\}$$

Due to Lemma 4, $\hat{i}$ is well-defined for every i.

Additionally, we define the *modal distance* $\delta(i,j) := |\{i \leq k \leq j \mid k = \hat{k}\}|$ as well that counts the number of modal rule application between i and j. Every i induces a generic path π_i by $\pi_i : j \mapsto s_{\min\{k \mid \delta(i,k)=j\}}$ and note π_i indeed is well-defined for every i.

By Lemma 6, the bad trace is either an $\mathtt{A}$- or an $\mathtt{E}$-trace that is eventually not spawning, i.e. there is a position i^* s.t. $B_j \equiv \mathtt{E}\Pi_j$ or $B_j \equiv \mathtt{A}\Sigma_j$ for all $j \geq i^*$ with (B_j, B_{j+1}) being not spawning. Let i^* be the least of such kind.

Next, we use the bad trace to find an $\mathtt{U}$-thread in it that is satisfied by the transition system.

For the purpose of finding the thread, we construct an infinite sequence of formulas $\phi_{i^*}, \phi_{i^*+1}, \ldots$ s.t. the following holds for all $i \geq i^*$:

a. $\phi_i \in B_i$ and $\pi_i \models \phi_i$,
b. $(B_i, \phi_i) \rightsquigarrow (B_{i+1}, \phi_{i+1})$,
c. $\phi_i = \varphi_1 \mathtt{U} \varphi_2$ for some φ_1, φ_2 and $\phi_i \neq \phi_{i+1}$ s.t. $\pi_i \models \varphi_2$ implies that $\phi_{i+1} = \varphi_2$.

Additionally, $\phi_i = \varphi_1 \mathtt{U} \varphi_2$ for infinitely many $i \geq i^*$ and some φ_1, φ_2.

Case: $B_0, B_1, \ldots$ **is an** $\mathtt{E}$**-trace.** If the trace is a bad $\mathtt{E}$-trace, it follows by definition of good traces that there is a $\mathtt{U}$-thread continuing with $\phi_{i^*}, \phi_{i^*+1}, \ldots$ in the trace fulfilling all these properties by construction.

Case: $B_0, B_1, \ldots$ **is an** $\mathtt{A}$**-trace.** Since i^* is the least s.t. $B_j = \mathtt{A}\Sigma_j$ for all $j \geq i^*$, Σ_{i^*} has to be a singleton, hence define ϕ_{i^*} to be the single formula in Σ_{i^*}.

For $i \rightsquigarrow i+1$, we apply a case distinction on whether $\hat{i} = i$.

If $\hat{i} = i$, i.e. the modal rule is to be applied next, $\phi_i = \mathtt{X}(\phi_i')$; simply set $\phi_{i+1} := \phi_i'$. Otherwise, assume that $\hat{i} \neq i$. If B_i is not principal in the rule instance set $\phi_{i+1} := \phi_i$. Otherwise, we apply a case distinction on whether ϕ_i is principal. If this is not the case, we simply set $\phi_{i+1} := \phi_i$; by construction, ϕ_{i+1} B_i is not spawning. Otherwise, if ϕ_i is the principal formula, we apply a case distinction on the rule instance. Note that ϕ_i is not of the form $\mathtt{E}\psi'$, $\mathtt{A}\psi'$ or ℓ by construction.

If $\phi_i = \psi_1 \mathtt{R} \psi_2$ let ϕ_{i+1} be one of the successors ψ' of ϕ_i contained in B_{i+1} with $\pi_i \models \psi'$ and note that there is at least one. If $\phi_i = \psi_1 \mathtt{U} \psi_2$ set $\phi_{i+1} := \psi_2$ iff $\pi_i \models \psi_2$ and $\phi_{i+1} := \psi'$ to the other successor ψ' of ϕ_i in B_{i+1} otherwise. Otherwise, $\phi_i = \psi_1 \wedge \psi_2$ or $\phi_i = \psi_1 \vee \psi_2$. Set $\phi_{i+1} := \psi_k$ s.t. ψ_k is connected to ϕ_i in B_{i+1} and $\pi_i \models \psi_k$.

By definition of bad traces, $\phi_{i^*}, \phi_{i^*+1}, \ldots$ has to be the continuation of an $\mathtt{U}$-thread.

Finally, we apply the definition of approximants on the obtained thread in order to show that it is impossible that the transition system satisfies the thread the whole time.

Let $A = \{i \geq i^* \mid \phi_i = \mathtt{X}(\varphi_1 \mathtt{U} \varphi_2) \text{ and } \phi_{i+1} = \varphi_1 \mathtt{U} \varphi_2\}$ and note that A is infinite due to Lemma 4 and the fact that $\phi_i = \varphi_1 \mathtt{U} \varphi_2$ for infinitely many $i \geq i^*$. Let $i_0 < i_1 < \ldots$ denote the ascending sequence of all $i \in A$. Additionally note that between two $i_j < i_{j+1}$ there is exactly one application of the $(\mathtt{X}_1)$ or $(\mathtt{X}_0)$ rule due to Lemma 8.

Since $\pi_{i_0} \models \mathtt{X}(\varphi_1 \mathtt{U} \varphi_2)$ it follows that is a k s.t. $\pi_{i_0} \models^k \varphi_2$, hence particularly $\pi_{i_{k+1}} \models \varphi_2$. By construction, $\phi_{i_{k+1}+1} = \varphi_2$, but by definition of A, it follows that $\phi_{i_{k+1}+1} = \varphi_1 \mathtt{U} \varphi_2$ which cannot be the case. □

C Corollaries

Proof (of Corollary 20). The number of states and edges in $\mathcal{G}_\varphi$ is $2^{2^{\mathcal{O}(|\varphi|)}}$, and its index is $2^{\mathcal{O}(|\varphi|)}$. It is known that parity games of size m and index k can be solved in time $m^{\mathcal{O}(k)}$ [19] from which the claim follows immediately. □

Proof (of Corollary 21). Suppose φ is satisfiable. According to Thm. 10 it has a tableau, and according to Thm. 19 player 0 has a winning strategy for the game $\mathcal{G}_\varphi$. It is well-known that he then also has a positional winning strategy, and another model of φ can be obtained from the subgraph induced by this strategy. Clearly, this subgraph and therefore the model can be at most as large as the parity game itself. The upper-bound on the branching-width is given by the fact that rule $(\mathtt{X}_1)$ can have at most $2^{|\varphi|}$ many premisses. □

D Linearly bounded branching width

Cor. 21 yield to an exponential branching-width in general. However, this degree can be reduced to a linear one by restricting the rule applications—as announced in Sect. 6. The following argumentation implicitly excludes the rules $(\mathtt{X}_0)$ and $(\mathtt{X}_1)$. Therefore, any considered rule application has exactly one principal formula.

We limit the application of every rule besides $(\mathtt{X}_0)$ and $(\mathtt{X}_1)$ to those applications where the principal formula is a largest formula among those formulas in the goal which have not $\mathtt{X}$ as their outermost connectives. The restriction is sound because any rule decomposes the principal formula, say φ, into smaller ones—excluding $\mathtt{X}\varphi$—, and because any two immediately rule application besides of $(\mathtt{X}_0)$ and of $(\mathtt{X}_1)$ can be swapped—including their principal formulas. Thereby the formerly first rule might be duplicated or be erased.

As a measure of a goal we take the number of its $\mathtt{E}$-blocks plus the number of formulas having the form $\mathtt{E}\varphi$ such that this formula is a subformula, but not under the scope of an $\mathtt{X}$-connective, of some formula in the goal and such that

$\mathtt{E}\{\varphi\}$ is not a block in this goal. This measure is bounded by $|\vartheta| + 1$ at the initial goal $\mathtt{E}\{\vartheta\}$ and at any premise of the rules $(\mathtt{X}_0)$ and $(\mathtt{X}_1)$.

The size restriction ensures that any rule instance besides $(\mathtt{X}_0)$ and $(\mathtt{X}_1)$ weakly decreases the measure. First, we consider the contribution of formulas to the measure. An inspection of the rules yields that any subformula $\mathtt{E}\varphi$ which contributes to the measure of the premise also occurs in the conclusion as a subformula. For the sake of contradiction, assume that $\mathtt{E}\varphi$ does not contribute to the measure of the conclusion. Hence, the principal block is preventing $\mathtt{E}\varphi$ from being counted and, hence, it has the shape $\mathtt{E}\{\varphi\}$. Therefore, the formula which hosts $\mathtt{E}\varphi$ is larger than the principal. But this situation contradicts the size restriction.

Since the rules besides $(\mathtt{EE})$ and $(\mathtt{AE})$ do not produce any new $\mathtt{E}$-block we are done with these rules. For the two remaining cases the names as shown in Fig. **??** are used. If $\mathtt{E}\varphi$ is excluded from the measure of the conclusion then and only then $\mathtt{E}\{\varphi\}$ is a block in Φ. Therefore, in the positive case this block is not new for the premise. And in the negative case the new block in the premise is paid by the formula in the conclusion and prevents other instances of this formula in the premise from being counted.