SATISFIABILITY GAMES FOR BRANCHING-TIME LOGICS

OLIVER FRIEDMANN, MARKUS LATTE, AND MARTIN LANGE
e-mail address: {oliver.friedmann, markus.latte}@ifi.lmu.de

Department of Computer Science, Ludwig-Maximilians-University Munich, Germany

e-mail address: martin.lange@uni-kassel.de

School of Electrical Engineering and Computer Science, University of Kassel, Germany

Abstract. The satisfiability problem for branching-time temporal logics like CTL*, CTL and CTL$^+$ has important applications in program specification and verification. Their computational complexities are known: CTL* and CTL$^+$ are complete for doubly exponential time, CTL is complete for single exponential time. Some decision procedures for these logics are known; they use tree automata, tableaux or axiom systems.

In this paper we present a uniform game-theoretic framework for the satisfiability problem of these branching-time temporal logics. We define satisfiability games for the full branching-time temporal logic CTL* using a high-level definition of winning condition that captures the essence of well-foundedness of least fixpoint unfoldings. These winning conditions form formal languages of ω-words. We analyse which kinds of deterministic ω-automata are needed in which case in order to recognise these languages. We then obtain a reduction to the problem of solving parity or Büchi games. The worst-case complexity of the obtained algorithms matches the known lower bounds for these logics.

This approach provides a uniform, yet complexity-theoretically optimal treatment of satisfiability for branching-time temporal logics. It separates the use of temporal logic machinery from the use of automata thus preserving a syntactical relationship between the input formula and the object that represents satisfiability, i.e. a winning strategy in a parity or Büchi game. The games presented here work on a Fischer-Ladner closure of the input formula only. Last but not least, the games presented here come with an attempt at providing tool support for the satisfiability problem of complex branching-time logics like CTL* and CTL$^+$.

1998 ACM Subject Classification: F.3.1, F.4.1.

Key words and phrases: temporal logic, automata, parity games, decidability.

A preliminary version appeared as [FLL10].

Financial support was provided by the DFG Graduiertenkolleg 1480 (PUMA) and the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement no 259267.

LOGICAL METHODS IN COMPUTER SCIENCE DOI:10.2168/LMCS-???

1. Introduction

The full branching-time temporal logic CTL* is an important tool for the specification and verification of reactive [GP08] or agent-based systems [LSS$^+$05], and for program synthesis [PR88], etc. Emerson and Halpern have introduced CTL* [EH86] as a formalism which supersedes both the branching-time logic CTL [CE81] and the linear-time logic LTL [Pnu77].

Automata-theoretic approaches. As much as the introduction of CTL* 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 procedure by Emerson and Sistla was automata-theoretic [ES84] and roughly works as follows. A formula is translated into a doubly-exponentially large tree automaton whose states are Hintikka-like sets of sets of subformulas of the input formula. This tree automaton recognises a superset of the set of tree models of the input formula. It is lacking a mechanism that ensures that certain temporal operators are really interpreted as least fixpoints of certain monotone functions rather than arbitrary fixpoints. Such a mechanism is provided by intersecting this automaton with a tree automaton that recognises a language which is defined as the set of all trees such that all paths in such a tree belong to an ω-regular word language, recognised by some Büchi automaton for instance. In order to turn this into a tree automaton, it has to be determinised first. A series of improvements on Büchi automata determinisation for this particular word language has eventually led to Emerson and Jutla's automata-theoretic decision procedure [EJ00] whose asymptotic worst-case running time is optimal, namely doubly exponential [VS85]. This approach has a major drawback though, as noted by Emerson [Eme90]: "*. . . due to the delicate combinatorial constructions involved, there is usually no clear relationship between the structure of automaton and the candidate formula.*"

The constructions he refers to are determinisation and complementation of Büchi automata. Determinisation in particular is generally perceived as the bottleneck in applications that need deterministic automata for ω-regular languages. A lot of effort has been spent on attempts to avoid Büchi determinisation for temporal branching-time logics. Kupferman, Vardi and Wolper introduced alternating automata [MS87] for branching-time temporal logics [BVW94, KVW00]. The main focus of this approach was the model-checking problem for such logics, though. While satisfiability checking and model checking for linear-time temporal logics are virtually the same problem and therefore can be handled by the same machinery, i.e. class of automata and algorithms, the situation for branching-time temporal logics is different. In the automata-theoretic framework, satisfiability corresponds to the general emptiness problem whereas model-checking reduces to the simpler 1-letter emptiness problem. Still, alternating automata provide an alternative framework for the satisfiability checking problem for branching-time logics, and some effort has been paid in order to achieve emptiness checks, and therefore satisfiability checking procedures. Most notably, Kupferman and Vardi have suggested a way to test tree automata for emptiness which avoids Büchi determinisation [KV05]. However, it is based on a satisfiability-preserving reduction only rather than an equivalence-preserving one. Thus, it avoids the "*delicate combinatorial constructions*" which are responsible for the lack of a "*clear relationship between the structure of automaton and the candidate formula*" in Emerson's view [Eme90], but to avoid these constructions it gives up any will to preserve such a clear relationship.

Other approaches. Apart from these automata-theoretic approaches, a few different ones have been presented as well. For instance, there is Reynolds' proof system for validity [Rey01]. Its completeness proof is rather intricate and relies on the presence of a rule which violates the subformula property. In essence, this rule quantifies over an arbitrary set of atomic propositions. Thus, while it is possible to check a given tree for whether ot not it is a proof for a given CTL* formula, it is not clear how this system could be used in order to find proofs for given CTL* formulas.

Reynolds has also presented a tableaux system for CTL* [Rey09, Rey11] which shares some commonalities with the automata-theoretic approach by Emerson and others as well as the game-based approach presented here. However, one of the main differences between tableaux on one side and automata and games on the other has a major effect in the case of such a complex branching-time logic: while automata- and game-based approaches typically separate the characterisation (e.g. tree automaton or parity game) from the algorithm (e.g. emptiness test or check for winning strategy), tableaux are often designed monolithically, i.e. with the characterisation and algorithm as one procedure. As a result, Reynolds' tableaux rely on some repetition test which, done in a naïve way, is hopelessly inefficient in practice. On the other hand, it is not immediately clear how a more clever and thus more efficient repetition check could be designed for these tableaux, and we predict that it would result in the introduction of Büchi determinisation.

A method that is traditionally used for predicate logics is resolution. It has also been used to devise decision procedures for temporal logics, starting with the linear-time temporal logic LTL [Fis91], followed by the simple branching-time temporal logic CTL [BF99, ZHD10]. Finally, there is also a resolution-based approach to CTL* which combines linear-time temporal logic resolution with additional techniques to handle path quantification [BDF99]. However, all resolution methods rely on the fact that the input formula is transformed into a specialised normal form. The known transformations are not trivial, and they only produce equi-satisfiable formulas. Thus, such methods also do not preserve a close connection between the models of the input formula and its subformulas.

The game-based framework. In this paper we present a game-based characterisation of CTL* satisfiability. In such games, two players play against each other with competing objectives: player 0 should show that the input formula is satisfiable whereas player 1 should show that it is not. Formally, the CTL* satisfiability game for some input formula is a graph of doubly exponential size on which the two players move a token along its edges. There is a winning condition in the form of a formal language of infinite plays which describes the plays that are won by player 0. This formal language turns out to be ω-regular, and it is known that arbitrary games with such a winning condition can be solved by a reduction to parity games. This yields an asymptotically optimal decision procedure. Still, the games only use subformulas of the input formula, and automata are only needed in the actual decision procedure but not in the definition of the satisfiability games as such. Thus, it moves the "*delicate combinatorial constructions*" to a place where they do not destroy a "*clear relationship between the [...] input formula*" and the parity game anymore. This is very useful in the setting of a user interacting with a satisfiability checker or theorem prover for CTL*, when they may want to be given a reason for why a formula is not satisfiable for instance.

The delicate combinatorial procedures, i.e. Büchi determinisation and complementation is kept at minimum by analysing carefully where it is needed. We decompose the winning

condition such that the transformation of a nondeterministic Büchi into a deterministic parity automaton [Saf88, Pit06, KW08, Sch09] is only needed for some part. The other is handled directly using manually defined deterministic automata.

We also consider two important fragments of CTL*, namely the well-known CTL and the lesser known CTL$^+$. The former has less expressive power and is computationally simpler: CTL satisfiability is complete for deterministic singly exponential time only [EH85]. The latter already carries the full complexity of CTL* despite sharing its expressive power with the weaker CTL [EH85]: CTL$^+$ satisfiability is also complete for doubly exponential time [JL03]. The simplicity of CTL when compared to CTL* also shows through in this game-based approach. The rules can be simplified a lot when only applied to CTL formulas, resulting in an exponential time procedure only. Even more so, the simplification gets rid of the need for automata determinisation procedures at all. Again, it is possible to construct a very small and deterministic Büchi automaton directly that can be used to check the winning conditions when simplified to CTL formulas.

The computational complexity of CTL$^+$ suggests that no major simplifications in comparison to CTL* are possible. Still, an analysis of the combinatorics imposed by CTL$^+$ formulas on the games shows that for such formulas it suffices to use determinisation for co-Büchi automata [MH84] instead of that for Büchi automata. This yields asymptotically smaller automata, is much easier to implement and also results in Büchi games rather than general parity games.

Advantages of the game-based approach. The game-theoretic framework achieves the following advantages.

- The framework uniformally treats the standard branching-time logics from the relatively simple CTL to the relatively complex CTL*.
- It yields *complexity-theoretic optimal* results, i.e. satisfiability checking using this framework is possible in exponential time for CTL and doubly exponential time for CTL* and CTL$^+$.
- Like the automata-theoretic approaches, it separates the characterisation of satisfiability through a syntactic object (a parity game) from the test for satisfiability (the problem of solving the game). Thus, advances in the area of parity game solving carry over to satisfiability checking.
- Like the tableaux-based approach, it keeps a very close relationship between the input formula and the structure of the parity game thus enabling feedback from a (counter-)model for applications in specification and verification.
- Satisfiability checking procedures based on this framework are implemented in the MLSolver platform [FL10] which uses the high-performance parity game solver PGSolver [FL09] as its algorithmic backbone — see the corresponding remark about the separation between characterisation and algorithm above.

Organisation. The rest of the paper is organised as follows. Section 2 recalls CTL*. Section 3 presents the satisfiability games. Section 4 gives the formal soundness and completeness proofs for the presented system. Section 5 describes the decision procedure, i.e. the reduction to parity games. Section 6 presents the simplifications one can employ in both the games and the reduction when dealing with formulas of CTL, respectively CTL$^+$. Section 7 compares the games presented here with other decision procedures for branching-time logics, in particular with respect to technical similarities, pragmatic aspects, results that follow

from them, etc. Section 8 concludes with some remarks on possible further work into this direction.

2. The Full Branching Time Logic

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 in negation normal form[1] are given by the following grammar.

$$\varphi \quad ::= \quad \mathtt{tt} \mid \mathtt{ff} \mid p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathtt{X}\varphi \mid \varphi \mathtt{U} \varphi \mid \varphi \mathtt{R} \varphi \mid \mathtt{E}\varphi \mid \mathtt{A}\varphi$$

where $p \in \mathcal{P}$. Formulas of the form $\mathtt{tt}$, $\mathtt{ff}$, p or $\neg p$ are called *literals*.

Boolean constructs other than conjunction and disjunction, like $\to$ for instance, are derived as usual. Temporal operators other than the ones given here are also defined as usual: $\mathtt{F}\varphi := \mathtt{tt}\mathtt{U}\varphi$ and $\mathtt{G}\varphi := \mathtt{ff}\mathtt{R}\varphi$.

The set of *subformulas* of a formula φ, written as $Sub(\varphi)$, is defined as usual, in particular the set contains φ. In contrast, a formula ψ is a *proper subformula* of φ if both are different and ψ is a subformula of φ. The *Fischer-Ladner* closure of φ is the least set $FL(\varphi)$ that is closed under taking subformulas, and contains, for each $\psi_1 \mathtt{U} \psi_2$ or $\psi_1 \mathtt{R} \psi_2$, also the formulas $\mathtt{X}(\psi_1 \mathtt{U} \psi_2)$ respectively $\mathtt{X}(\psi_1 \mathtt{R} \psi_2)$. Note that $|FL(\varphi)|$ is at most twice the number of subformulas of φ. Let $FL_{\mathtt{R}}(\varphi)$ consist of all formulas in $FL(\varphi)$ that are of the form $\psi_1 \mathtt{R} \psi_2$ or $\mathtt{X}(\psi_1 \mathtt{R} \psi_2)$. The notation is extended to formula sets in the usual way. The size $|\varphi|$ of a formula φ is number of its subformulas. Formulas are interpreted over paths π of a transition systems $\mathcal{T} = (\mathcal{S}, s^*, \to, \lambda)$. We have $\mathcal{T}, \pi \models \mathtt{tt}$ but not $\mathcal{T}, \pi \models \mathtt{ff}$ for any $\mathcal{T}$ and π; and the semantics of the other constructs is given as follows.

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

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 $\mathcal{T}, \pi \models \psi$.

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

[1] Alternatively, we could have admitted negations everywhere—not only in front of a proposition. However, for any formula of one form there is an equivalent and linearly sized formula of the other form: just apply De Morgan's laws to the binary propositional connectors, e.g. $\neg(\varphi_1 \wedge \varphi_2) \equiv (\neg\varphi_1) \vee (\neg\varphi_2)$, fixpoint duality to fixpoints, e.g. $\neg(\varphi_1 \mathtt{U} \varphi_2) \equiv (\neg\varphi_1)\mathtt{R}(\neg\varphi_2)$ and the property $\neg\mathtt{X}\varphi \equiv \mathtt{X}\neg\varphi$.

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. $\mathcal{T}, s^* \models \varphi$.

Finally, we introduce the two most well-known fragments of CTL*, namely CTL and CTL$^+$. In CTL, no Boolean combinations or nestings of temporal operators are allowed; they have to be immediately preceded by a path quantifier. The syntax is given by the following grammar starting with φ.

$$\varphi \quad ::= \quad \mathtt{tt} \mid \mathtt{ff} \mid p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathtt{E}\psi \mid \mathtt{A}\psi \tag{2.1}$$

$$\psi \quad ::= \quad \mathtt{X}\varphi \mid \varphi \mathtt{U} \varphi \mid \varphi \mathtt{R} \varphi \tag{2.2}$$

Formulas generated by φ are state formulas.

The logic CTL$^+$ lifts the syntactic restriction slightly: it allows Boolean combinations of path operators inside a path quantifier, but no nestings thereof. It is defined by the following grammar starting with φ.

$$\varphi \quad ::= \quad \mathtt{tt} \mid \mathtt{ff} \mid p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathtt{E}\psi \mid \mathtt{A}\psi \tag{2.3}$$

$$\psi \quad ::= \quad \varphi \mid \psi \vee \psi \mid \psi \wedge \psi \mid \mathtt{X}\varphi \mid \varphi \mathtt{U} \varphi \mid \varphi \mathtt{R} \varphi \tag{2.4}$$

It should be clear that CTL is a fragment of CTL$^+$ which is, in turn, a fragment of CTL*. However, only the latter inclusion is proper w.r.t. expressivity as stated in the following.

Proposition 1 ([EH86])**.** CTL* is strictly more expressive than CTL$^+$, and CTL$^+$ is as expressive as CTL.

Nevertheless, there are families of properties which can be expressed in CTL$^+$ using a family of formulas that is linearly growing in size, whereas every family of CTL formulas expressing these properties must have exponential growth. This is called an exponential succinctness gap between the two logics.

Proposition 2 ([Wil99, AI01, Lan08])**.** There is an exponential succinctness gap between CTL$^+$ and CTL.

Such a succinctness gap can cause different complexities of decision procedures for the involved logics despite equal expressive power. This is true in this case.

Proposition 3 ([EH85, FL79])**.** Satisfiability checking for CTL is EXPTIME-complete.

The exponential succinctness gap causes on exponentially more difficult satisfiability problem which is shared with that of the more expressive CTL*.

Proposition 4 ([EJ00, VS85, JL03])**.** Satisfiability checking for CTL* and for CTL$^+$ are both 2EXPTIME-complete.

3. Satisfiability Games for CTL*

Here we are concerned with special 2-player zero-sum games of perfect information. They can be seen as a finite, directed graph whose node set is partitioned into sets belonging to each player. Formally, a game is a tuple $\mathcal{G} = (V, V_0, E, v_0, L)$ where (V, E) is a directed graph. We restrict our attention to total graphs, i.e. every node is assumed to have at least one successor. The set $V_0 \subseteq V$ consists of all nodes owned by player 0. This naturally induces the set $V_1 := V \setminus V_0$ of all nodes owned by player 1. The node v_0 is the designated initial node.

Any play starts from this initial node by placing a token there. Whenever the token is on a node that belongs to player i, it is his/her turn to push it along an edge to a successor node. In the infinite, this results in a play, and the winning condition $L \subseteq V^\omega$ prescribes which of these plays are won by player 0.

A *strategy* for player i is a function $\sigma : V^*V_i \to V$ which tells him/her how to move in any given situation in a play. Formally, a play $v_0, v_1, \ldots$ conforms to strategy σ for player i, if for every j with $v_j \in V_i$ we have $v_{j+1} = \sigma(v_0 \ldots v_j)$. A *winning strategy* for player i is a strategy such that he/she wins any play regardless of the opponent's choices. Formally, σ is a winning strategy if for all plays $\pi = v_0, v_1, \ldots$ that conform to σ we have $\pi \in L$.

It is easy to relax the requirements of totality. In that case we attach two additional designated nodes win_0 and win_1 such that every node originally without successors gets an edge to either of them, each of these only has one edges to itself, and the winning condition L includes all words of the form $V^*\mathsf{win}_0^\omega$ and excludes all of the form $V^*\mathsf{win}_1^\omega$. In the following we will therefore allow ourselves to have plays ending in states without successors which can be turned into total games using this simple transformation. In other words every dead end is lost by the player who owns the node.

3.1. **The Game Rules.** We present satisfiability games for branching-time state formulas in negation normal form. Let ϑ be a CTL*-formula fixed for the remainder of this section. For convenience, the games will be presented w.r.t. to this particular formula ϑ.

We need the following notions: Σ and Π are finite (possibly empty) sets of formulas with Σ being interpreted as a disjunction of formulas and Π as a conjunction. A *quantifier-bound formula block* is an $\mathtt{E}$- or $\mathtt{A}$-labelled set of formulas, i.e. either a $\mathtt{E}\Pi$ or a $\mathtt{A}\Sigma$. Any set under an $\mathtt{E}$-bound resp. $\mathtt{A}$-bound block is assumed to be read as a conjunction resp. disjunction of the formulas. We identify an empty Σ with $\mathtt{ff}$ and an empty Π with $\mathtt{tt}$. We write Λ for a set of literals. For a set of formulas Γ let $\mathtt{X}\Gamma := \{\mathtt{X}\psi \mid \psi \in \Gamma\}$.

In order to ease readability we will omit as many curly brackets as possible and often use round brackets to group formulas into a set. For instance $\mathtt{E}(\varphi \wedge \psi, \Pi)$ denotes a block that is prefixed by $\mathtt{E}$ and which consists of the union of Π and $\{\varphi \wedge \psi\}$, implicitly assuming that this does not occur in Π already.

A *configuration (for ϑ)* is a non-empty set 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 $FL(\vartheta)$. The meaning of such a configuration is given by the state formula

$$\bigwedge_{i=1}^{n} \mathtt{A}\Big(\bigvee_{\psi \in \Sigma_i} \psi\Big) \wedge \bigwedge_{i=1}^{m} \mathtt{E}\Big(\bigwedge_{\psi \in \Pi_i} \psi\Big) \wedge \bigwedge_{\ell \in \Lambda} \ell \ .$$

Note that a configuration only contains existentially quantified conjunctions and universally quantified disjunctions as blocks. There are no blocks of the form $\mathtt{E}\Sigma$ or $\mathtt{A}\Pi$ simply because an existential path quantifier commutes with a disjunction, and so does a universal path quantifier with a conjunction. Thus, $\mathtt{A}\Sigma$ would be equivalent to $\bigwedge\{\mathtt{A}\varphi \mid \varphi \in \Sigma\}$ for instance.

A configuration $\mathcal{C}$ is *consistent* if it does not contain $\mathtt{ff}$ and there is no $p \in \mathcal{P}$ s.t. $p \in \mathcal{C}$ and $\neg p \in \mathcal{C}$. Note that the meaning of an inconsistent configuration is unsatisfiable, but the converse does not hold because consistency is only concerned with the occurrence of literals. Unsatisfiability can also be given be conflicting temporal operators, e.g. $\mathtt{E}(\mathtt{X}p, \mathtt{X}\neg p)$.

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

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

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

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

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

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

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

Figure 1: The game rules for CTL*.

We write $\mathit{Conf}(\vartheta)$ for the set of all consistent configurations for ϑ. Note that this is a finite set of at most doubly exponential size in $|\vartheta|$.

Definition 5. The satisfiability game $\mathcal{G}_\vartheta$ for the formula ϑ is a game $(\mathit{Conf}(\vartheta), V_0, E, v_0, L)$ whose nodes are all possible configurations and whose edge relation is given by the game rules in Figure 1. They also determine which configurations belong to player 0, i.e. to V_0, namely all but those to which rule $(\mathtt{X}_1)$ is applied.

Note that the rules are written such that a configuration at the bottom of the rule has, as its successors, all configurations at the top of the rule. It is only rules $(\mathtt{Al})$, $(\mathtt{AA})$, $(\mathtt{AE})$, $(\mathtt{E}\vee)$, $(\mathtt{EU})$, and $(\mathtt{ER})$ which always produce two successors, rule $(\mathtt{X}_1)$ can have an arbitrary number of successors that is at least one. It is understood that the formulas which are stated explicitly under the line do not occur in the sets Λ or Φ. The symbol ℓ stands for an arbitrary literal.

The initial configuration is $v_0 = \mathtt{E}\vartheta$. The winning condition L will be described in Definition 15 in the next subsection.

As for the representation, examples in this paper will use tailored rules for the abbreviations $\mathtt{F}$ and $\mathtt{G}$ instead of the rules $(\mathtt{AU})$, $(\mathtt{EU})$, $(\mathtt{AR})$ and $(\mathtt{ER})$. Take for instance a construct of the form $\mathtt{AF}\psi$. A rule for this can easily be derived by applying the rules for the unabbreviated version of this.

$$\begin{array}{l}(\mathtt{Al})\\(\mathtt{AU})\end{array}\ \frac{\mathtt{A}(\psi,\mathtt{XF}\psi,\Sigma),\Phi,\mathtt{tt}}{\mathtt{A}(\psi,\mathtt{tt},\Sigma),\mathtt{A}(\psi,\mathtt{XF}\psi,\Sigma),\Phi}$$
$$\frac{}{\mathtt{A}(\underbrace{\mathtt{tt}\mathtt{U}\psi}_{=\mathtt{F}\psi},\Sigma),\Phi}$$

The additional $\mathtt{tt}$ in the literal part does not affect consistency of a configuration, nor the applicability of any other rule. Hence, it can be dropped. Therefore, we can use the following

rule for this construct.

$$(\mathtt{AF})\ \frac{\mathtt{A}(\psi, \mathtt{XF}\psi, \Sigma), \Phi}{\mathtt{A}(\mathtt{F}\psi, \Sigma), \Phi}$$

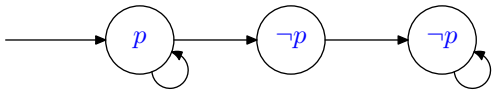
The other abbreviated rules follow the same lines.

$$(\mathtt{AG})\ \frac{\mathtt{A}(\psi, \Sigma), \mathtt{A}(\mathtt{XG}\psi, \Sigma), \Phi}{\mathtt{A}(\mathtt{G}\psi, \Sigma), \Phi} \qquad (\mathtt{EF})\ \frac{\mathtt{E}(\psi, \Pi), \Phi \ \mid\ \mathtt{E}(\mathtt{XF}\psi, \Pi), \Phi}{\mathtt{E}(\mathtt{F}\psi, \Pi), \Phi}$$

$$(\mathtt{EG})\ \frac{\mathtt{E}(\psi, \mathtt{XG}\psi, \Pi), \Phi}{\mathtt{E}(\mathtt{G}\psi, \Pi), \Phi}$$

Note that for the (EG) rule — which is based on the (ER) rule — it is never the wrong choice to select the right alternative instead of the left one. Choosing the left one would leave us with a configuration denoting $\mathtt{E}(\psi \wedge \mathtt{ff} \wedge \bigwedge \Pi) \wedge \bigwedge \Phi$ which can never be satisfied because of the constant ff.

Example 6. A strategy for player 0 in the game on $\mathtt{AFG}p \ \wedge\ \mathtt{EGEF}\neg p$ is represented in Figure 2. Note that such strategies can be seen as infinite trees. The bold arrows in Figure 2 point towards repeating configurations in this strategy. This is meant to represent the infinite tree that is obtained by repeatedly continuing as it is done in the two finite branches. Also note that in general, strategies may not be representable in a finite way like this. The twin lines indicate hidden configurations whenever unary rules can be applied in parallel. For instance, the double line at the bottom represents the application of the rules (AF) and (EG). The thin arrows will only be used in the next subsection in order to explain the winning conditions in the satisfiability games. A strategy for player 0 induces canonically a tree model by collapsing successive configurations that are not separated by applications of the rules ($\mathtt{X}_0$) and ($\mathtt{X}_1$). Doing this to the strategy in Figure 2 results in the following transition system. Note that the tableau of Figure 2 gives no specification on whether p should be included in the right-most node. It is natural to only include those propositions that are required to be true.



Note that it does not satisfy the formula $\mathtt{AFG}p \wedge \mathtt{EGEF}\neg p$. The overall goal is to characterise satisfiability in CTL* through these games. Hence, it is important to define the winning conditions such that this strategy is not a winning strategy.

3.2. **The Winning Conditions.** 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 successor configurations. 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 inside of an A- or E-block are principal in rules (Al) and (El). Finally, any non-principal occurrence of a formula in a configuration may have a *copy* in one of the successor configurations. The copy is the same formula since it has not been transformed. For instance, any formula in Σ in rule (Al) has a copy in the successor written on the right but does not have a copy in the successor on the left.

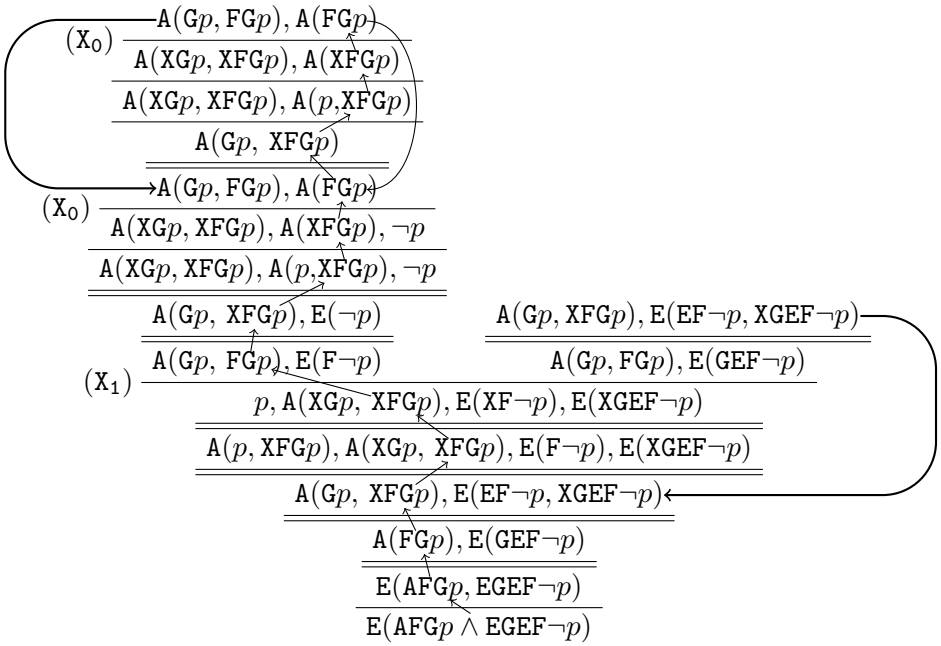


Figure 2: A strategy for player 0 in the satisfiability game for $\mathtt{AFG}p \wedge \mathtt{EGEF}\neg p$.

The gap between the existence of strategies for player 0 and satisfiability is caused by unfulfilled eventualities: an eventuality is a formula of the form $\mathtt{U}$ or its abbreviation $\mathtt{F}$. Note how the rules handle these by unfolding using the CTL* equivalence $Q(\psi_1 \mathtt{U} \psi_2) \equiv Q(\psi_2 \vee (\psi_1 \wedge \mathtt{X}(\psi_1 \mathtt{U} \psi_2)))$ for any $Q \in \{\mathtt{E}, \mathtt{A}\}$. The rules for the Boolean operators and for the $\mathtt{X}$-modalities can lead to a configuration in which $\psi_1 \mathtt{U} \psi_2$ occurs again inside of a Q-block. Note that inside an $\mathtt{E}$-block this is only possible if player 0 decides not to choose the successor containing ψ_2. Inside of an $\mathtt{A}$-block the situation is slightly different; player 0 has no choices there. Still, it is important to note that a $\mathtt{U}$-formula should not be unfolded infinitely often because $\psi_1 \mathtt{U} \psi_2$ asserts that eventually ψ_2 will be true, and unfolding postpones this by one state in a possible model. Thus, the winning conditions have to ensure that player 0 cannot let an eventuality formula get unfolded infinitely many times without its right argument being satisfied infinitely many times as well.

In order to track the infinite behaviour of eventualities, one needs to follow single formulas through the branches that get transformed by a rule from time to time. Note that a formula can occur inside of several blocks. Thus it is important to keep track of the block structure as well.

In the following we develop the technical definitions that are necessary in order to capture such unfulfilled eventualities and present some of their properties.

Definition 7. A quantifier-bound block $\mathtt{A}\Sigma$ or $\mathtt{E}\Pi$ is called *principal* as well if it contains a principal formula. A quantifier-bound block might have *descendants* in the successor(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 8. Let $\mathcal{C}_1$ be a configuration to which a rule r is applicable and let $\mathcal{C}_2$ be one of its successors. Furthermore, let $Q_1\Delta_1$, resp. $Q_2\Delta_2$ with $Q_1, Q_2 \in \{\mathtt{E}, \mathtt{A}\}$ and $\Delta_1, \Delta_2 \subseteq FL(\vartheta)$

be quantifier-bound blocks occurring in the A- or E-part of $\mathcal{C}_1$, resp. $\mathcal{C}_2$. We say that $Q_1\Delta_1$ is *connected* to $Q_2\Delta_2$ in $\mathcal{C}_1$ and $\mathcal{C}_2$, if either

- $Q_1\Delta_1$ is principal in r, and $Q_2\Delta_2$ is one of its descendants in $\mathcal{C}_2$; or
- $Q_1\Delta_1$ is not principal in r and $Q_2\Delta_2$ is a copy of $Q_1\Delta_1$ in $\mathcal{C}_2$.

We write this as $(\mathcal{C}_1, Q_1\Delta_1) \rightsquigarrow (\mathcal{C}_2, Q_2\Delta_2)$. If the rule instance can be inferred from the context we may also simply write $Q_1\Delta_1 \rightsquigarrow Q_2\Delta_2$. Additionally, let ψ_1, resp. ψ_2 be a formula occurring in Δ_1, resp. Δ_2. We say that ψ_1 is *connected* to ψ_2 in $(\mathcal{C}_1, Q_1\Delta_1)$ and $(\mathcal{C}_2, Q_2\Delta_2)$, if either

- ψ_1 is principal in r, and ψ_2 is one of its descendants in $\mathcal{C}_2$; or
- ψ_1 is not principal in r and ψ_2 is a copy of ψ_1 in $\mathcal{C}_2$.

We write this as $(\mathcal{C}_1, Q_1\Delta_1, \psi_1) \rightsquigarrow (\mathcal{C}_2, Q_2\Delta_2, \psi_2)$. If the rule instance can be inferred from the context we may also simply write $(Q_1\Delta_1, \psi_1) \rightsquigarrow (Q_2\Delta_2, \psi_2)$. A block connection $(\mathcal{C}_1, Q_1\Delta_1) \rightsquigarrow (\mathcal{C}_2, Q_2\Delta_2)$ is called *spawning* if there is a formula ψ s.t. $Q_2\psi \in \Delta_1$ is principal and $\Delta_2 = \{\psi\}$. The only rules that possibly induce a spawning block connection are (EE), (EA), (AA) and (AE). For example $(C_1, \mathtt{A}\{q, \mathtt{E}p\}) \rightsquigarrow (C_2, \mathtt{E}\{p\})$ is spawning while $(C_1, \mathtt{A}\{q, \mathtt{E}p\}) \rightsquigarrow (C_2, \mathtt{A}\{q\})$ is not.

Definition 9. Let $\mathcal{C}_0, \mathcal{C}_1, \ldots$ be an infinite play of a satisfiability game for some formula ϑ. A *trace* Ξ in this play is an infinite sequence $Q_0\Delta_0, Q_1\Delta_1, \ldots$ s.t. for all $i \in \mathbb{N}$: $(\mathcal{C}_i, Q_i\Delta_i) \rightsquigarrow (\mathcal{C}_{i+1}, Q_{i+1}\Delta_{i+1})$. A trace Ξ is called an E-*trace*, resp. A-*trace* if there is an $i \in \mathbb{N}$ s.t. $Q_j = \mathtt{E}$, resp. $Q_j = \mathtt{A}$ for all $j \geq i$. We say that a trace is *finitely spawning* if it contains only finitely many spawning block connections.

Lemma 10. Every infinite play contains infinitely many applications of rules ($\mathtt{X_0}$) or ($\mathtt{X_1}$).

Proof. First, we define the *duration of a formula* ψ as the syntactic height when X-subformulas are treated as atoms. More formally:

$$\mathrm{dur} : \psi \mapsto \begin{cases} 1 & \text{if } \psi \equiv \mathtt{tt}, \mathtt{ff}, p, \neg p, \mathtt{X}\psi' \\ 1 + \mathrm{dur}(\psi') & \text{if } \psi \equiv \mathtt{E}\psi', \mathtt{A}\psi' \\ 1 + \max(\mathrm{dur}(\psi_1), \mathrm{dur}(\psi_2)) & \text{if } \psi \equiv \psi_1 \vee \psi_2, \psi_1 \wedge \psi_2, \psi_1 \mathtt{U} \psi_2, \psi_1 \mathtt{R} \psi_2 \end{cases}$$

A well-ordering $<$ on the duration of formulas is induced by the well-ordering on natural numbers. Let F be $\{\mathrm{dur}(\varphi) \mid \varphi \in FL(\vartheta)\}$, the range of these durations, and let B be the range of all block sizes, that is $\{0, \ldots, |FL(\vartheta)|\}$. Both sets are finite.

Second, we define the *duration of a block* $Q\Delta$ as a map $\mathrm{dur}(Q\Delta) : F \to B$ that returns the number of subformulas of a certain duration. More formally:

$$\mathrm{dur}(Q\Delta) : n \mapsto |\Delta \cap \mathrm{dur}^{-1}(n)|$$

A well-ordering $\prec$ on the duration of blocks is given as follows (as the domain of the duration is finite and its range is well-founded).

$$f \prec g \quad :\Longleftrightarrow \quad \exists n \in F : f(n) < g(n) \ \wedge \ \forall m > n : f(m) = g(m)$$

Third, we define the *duration of a configuration* $\mathcal{C}$ as a map $\mathrm{dur}(\mathcal{C}) : B^F \to \mathbb{N}$ that returns the number of blocks of a certain duration. More formally:

$$\mathrm{dur}(\mathcal{C}) : f \mapsto |\mathcal{C} \cap \mathrm{dur}^{-1}(f)|$$

A well-ordering $\lhd$ on the duration of configurations is given as follows.

$$C \lhd D \quad :\Longleftrightarrow \quad \exists f \in B^F : C(f) < D(f) \ \wedge \ \forall g \succ f : C(g) = D(g)$$

Indeed, $\lhd$ is well-founded as the domain of durations, B^F, is finite.

The claim now follows from the fact that every rule application except for $(\mathtt{X_0})$ and $(\mathtt{X_1})$ strictly decreases the duration of the configuration. □

Definition 11. Let $\mathcal{C}_0, \mathcal{C}_1, \ldots$ be an infinite play. A *thread* t in a trace $\Xi = Q_0\Delta_0, Q_1\Delta_1, \ldots$ within $\mathcal{C}_0, \mathcal{C}_1, \ldots$ is an infinite sequence $\psi_0, \psi_1, \ldots$ s.t. for all $i \in \mathbb{N}$: $(\mathcal{C}_i, Q_i\Delta_i, \psi_i) \rightsquigarrow (\mathcal{C}_{i+1}, Q_{i+1}\Delta_{i+1}, \psi_{i+1})$. Such a thread t is called a $\mathtt{U}$-thread, resp. $\mathtt{R}$-thread if there is a formula $\varphi\mathtt{U}\psi \in FL(\vartheta)$, resp. $\varphi\mathtt{R}\psi \in FL(\vartheta)$ s.t. $\psi_j = \varphi\mathtt{U}\psi$, resp. $\psi_j = \varphi\mathtt{R}\psi$ for infinitely many j.

An $\mathtt{E}$-trace is called *good* iff it has no $\mathtt{U}$-thread; similarly, an $\mathtt{A}$-trace is called *good* iff it has an $\mathtt{R}$-thread. In other words, an $\mathtt{E}$-trace is called *bad* if it contains an $\mathtt{U}$-thread, and an $\mathtt{A}$-trace is called *bad* if it contains no $\mathtt{R}$-thread.

Lemma 12. Every trace in an infinite play is either an $\mathtt{A}$-trace or an $\mathtt{E}$-trace, and is only finitely spawning.

Proof. Let $Q_0\Delta_0, Q_1\Delta_1, \ldots$ be a trace and assume that $\{i \mid Q_i\Delta_i \rightsquigarrow Q_{i+1}\Delta_{i+1} \text{ is spawning}\}$ 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 $\phi_{i_{j+1}}$ is a proper subformula of ϕ_{i_j}. Hence the set cannot be infinite. Now note that every finitely spawning trace eventually must be either an $\mathtt{A}$- or an $\mathtt{E}$-trace because the change of the quantifier on the current block in a trace is only possible in a moment that the trace is spawning. □

Lemma 13. Every thread in a trace of an infinite play is either a $\mathtt{U}$- or an $\mathtt{R}$-thread.

Proof. Let $t = \psi_0, \psi_1, \ldots$ be a thread. Assume that t is neither a $\mathtt{U}$- nor an $\mathtt{R}$-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 ψ_{i+1} is a subformula of ψ_i for all $i \geq i^*$. By Lemma 10 it follows that $\psi_{i+1} \neq \psi_i$ for infinitely many i which cannot be the case, hence t has to be a $\mathtt{U}$- or an $\mathtt{R}$-thread. Finally, assume that t is both a $\mathtt{U}$- and an $\mathtt{R}$-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 ψ_{i_1} is a proper subformula of ψ_{i_0} and ψ_{i_2} is a proper subformula of ψ_{i_1}, thus ψ_{i_0} would be a proper subformula of itself. □

Lemma 14. For every $\mathtt{U}$- and every $\mathtt{R}$-thread $\psi_0, \psi_1, \ldots$ in a trace of an infinite play there is an $i \in \mathbb{N}$ such that ψ_i is a $\mathtt{U}$-, or an $\mathtt{R}$-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 ψ_{i+1} is a subformula ψ_i, or $\psi_{i+1} = \mathtt{X}\psi_i$ provided that ψ_i is a $\mathtt{U}$- or an $\mathtt{R}$-formula. The map which removes the frontal $\mathtt{X}$ from a formula converts the thread into a chain which is weakly decreasing with respect to the subformula order. Because this order is well-founded, the chain is eventually constant, say from n onwards. By Lemma 10, either $(\mathtt{X_0})$ or $(\mathtt{X_1})$ has been applied at a position $i-1$ for some $i > n$. Hence, ψ_i is either a $\mathtt{U}$- or an $\mathtt{R}$-formula, and i meets the claimed property. □

We now have obtained all the necessary technical material that is needed to define the winning conditions in the satisfiability game $\mathcal{G}_\vartheta$.

Definition 15. The winning condition L of $\mathcal{G}_\vartheta = (Conf(\vartheta), V_0, E, v_0, L)$ consists of every finite play which ends in a consistent set of literals, and of every infinite play which does not contain a bad trace.

In other words, player 0's objective is to create a play in which every $\mathtt{U}$-formula inside of an $\mathtt{E}$-trace gets fulfilled eventually. She can control this using rule $(\mathtt{EU})$. Inside of an $\mathtt{A}$-trace

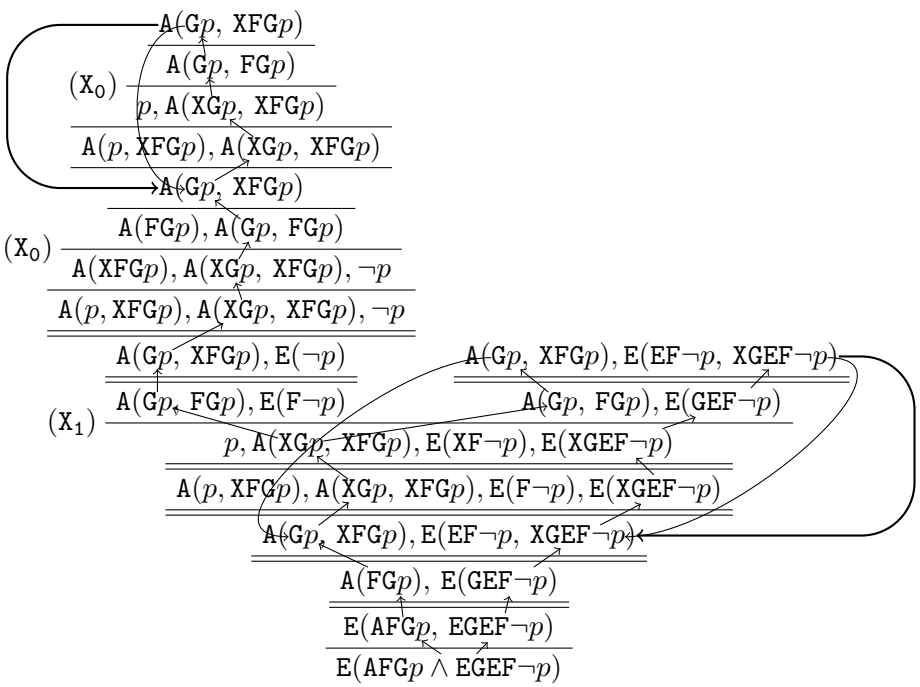
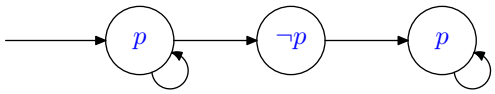


Figure 3: A winning strategy for player 0 in the satisfiability game on $\mathtt{AFG}p \wedge \mathtt{EGEF}\neg p$.

She must hope that not every formula that gets unfolded infinitely often is of the U-type. Note that sets inside of an E-block are conjunctions, hence, one unfulfilled formula makes the entire block false. Inside of an A-block the sets are disjunctions though, hence, in order to make this block true it suffices to satisfy one of the formula therein. An R-formula that gets unfolded infinitely often is—unlike an U-formula—indeed satisfied.

Example 16. Consider the strategy in Figure 2 again. It is not a winning strategy because its left branch contains a bad A-trace, i.e. the eventuality $\mathtt{FG}p$ is postponed for an infinite number of steps, which is the only thread contained in the trace. Since this thread is an U-thread, there is no R-thread contained in the trace.

Figure 3 shows a winning strategy for player 0 in the game on this formula $\mathtt{AFG}p \wedge \mathtt{EGEF}\neg p$. Infinite threads are being depicted using thin arrows. It is not hard to see that every A-trace contains a R-thread and that every E-trace only contains R-threads. Again, this strategy induces a canonic model, but this time a satisfying one because it is in fact a winning strategy:



Note that in this case, all paths starting in the leftmost state will eventually only visit states that satisfy p. Furthermore, there is a path—namely the loop on this state—on which every state is the beginning of a path—namely the one moving over to the right—on which $\neg p$ holds at some point.

Winning strategies, as opposed to ordinary strategies, exactly characterise satisfiability of CTL*-formulas in the following sense.

Theorem 17. For all $\vartheta \in \mathrm{CTL}^*$: ϑ is satisfiable iff player 0 has a winning strategy for the satisfiability game $\mathcal{G}_\vartheta$.

The proof is given in the following section.

4. Correctness Proofs

This section contains the proof of Theorem 17; both implications – soundness and completeness – are considered separately. The completeness proof is technically tedious but does not use any heavy machinery once the right invariants are found. Given a model for ϑ we use these invariants to construct a winning strategy for player 0 in a certain way. Soundness can be shown by collapsing a winning strategy into a tree-like transition system and verifying that it is indeed a model of ϑ.

4.1. Soundness.

Theorem 18. Suppose that player 0 has a winning strategy for the satisfiability game $\mathcal{G}_\vartheta$. Then ϑ is satisfiable.

Proof. We treat the winning strategy, say σ, as a tree T with nodes $\mathcal{V}$ and a root r. The nodes are labelled with configurations corresponding to the strategy. Thus, labels which belong to player 0 have at most one successor. Only a node which is the objective of the rule $(\mathtt{X_1})$ can have more successors.

Let $\mathcal{S}$ be those nodes which are leaves or on which the rules $(\mathtt{X_0})$ or $(\mathtt{X_1})$ are applied. The tree defines a transition system as described just before of Subsection 3.2. Formally, for any node s let $\widehat{s}$ be the oldest descendants —including s— of s in $\mathcal{S}$. Since player 0 owns all configurations besides those which rule $(\mathtt{X_1})$ can handle, Lemma 10 ensures that this assignment is total. The edge relation $\to \subseteq \mathcal{S} \times \mathcal{S}$ is defined as

$$\{(t, \widehat{s}) \mid s \text{ is a child of } t \text{ in } T\} \cup \{(s, s) \mid s \text{ is a leaf in } T\}.$$

The induced transition system is $\mathcal{T}_\vartheta = (\mathcal{S}, \widehat{r}, \to, \ell)$ where $\ell(s) = \mathcal{C} \cap \mathcal{P}$ for any $s \in S$ labelled with a configuration $\mathcal{C}$. Note that $\mathcal{T}_\vartheta$ is total. In the following, we omit the transition system $\mathcal{T}_\vartheta$ in front of the symbol $\models$. Moreover, we identify a node with its annotated configuration.

For the sake of a contradiction, assume that $\mathcal{T}_\vartheta, \widehat{r} \not\models \mathtt{E}\vartheta$. We will show that the winning strategy σ admits an infinite play which contains a bad trace. For this purpose, we simultaneously construct a maximal play $\mathcal{C}_0, \mathcal{C}_1, \ldots$ which conforms to σ, a maximal connected sequence of blocks $Q_0\Gamma_0, Q_1\Gamma_1, \ldots$ in this play, and a partial sequence π_i of paths in $\mathcal{T}_\vartheta$ such that the following properties hold for all indices i and for all formulas φ and ψ.

- (Ξ-1) If $Q_i = \mathtt{E}$ then $\widehat{\mathcal{C}_i} \not\models \mathtt{E}(\bigwedge \Gamma_i)$.
- (Ξ-2) If $Q_i = \mathtt{E}$, the rule $(\mathtt{EE})$ or $(\mathtt{EA})$ is applied to $\mathcal{C}_i$ with $\mathtt{E}\Gamma_i$ and φ as principals, and $\widehat{\mathcal{C}_i} \not\models \varphi$, then $Q_i\Gamma_i = \varphi$.
- (Ξ-3) If $Q_i = \mathtt{A}$ then π_i is defined, $\widehat{\mathcal{C}_i} = \pi_i(0)$, and $\pi_i \not\models \bigvee \Gamma_i$.
- (Ξ-4) If $Q_i = Q_{i+1} = \mathtt{A}$, and the rule $(\mathtt{X_0})$ or $(\mathtt{X_1})$ is applied to $\mathcal{C}_i$ then $\pi_{i+1} = \pi_i^1$ holds.
- (Ξ-5) If $Q_i = Q_{i+1} = \mathtt{A}$, and neither $(\mathtt{X_0})$ nor $(\mathtt{X_1})$ is applied to $\mathcal{C}_i$ then $\pi_{i+1} = \pi_i$ holds.
- (Ξ-6) If $Q_i\Gamma_i = \mathtt{A}(\varphi \mathtt{R} \psi, \Sigma)$, if the rule $(\mathtt{AR})$ is applied to $\mathcal{C}_i$ 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 is straight forward. We detail the proof for some cases, and thereto use formulas and notations as shown in Figure 1. As for the rule (EA), if $\widehat{\mathcal{C}}_i \not\models \mathtt{E}(\mathtt{A}\varphi \wedge \bigwedge \Pi)$ then $\widehat{\mathcal{C}}_i \not\models \mathtt{A}\varphi$ or $\widehat{\mathcal{C}}_i \not\models \mathtt{E}(\bigwedge \Pi)$. If the first case does not apply then the trace is continued with $\mathtt{E}\Pi$. Otherwise, $Q_{i+1}\Gamma_{i+1} = \mathtt{A}\varphi$ holds and π_{i+1} is an arbitrary path in $\mathcal{T}_\vartheta$ which starts at $\widehat{\mathcal{C}}_i$ and which fulfills $\pi_{i+1} \not\models \varphi$. As for the rule (AR), we have $\pi_i \not\models \varphi \mathtt{R} \psi \vee \bigvee \Sigma$. Using that $\pi_i = \pi_{i+1}$ and an unrolling of the R-operator, $\pi_{i+1} \not\models \psi$ or $\pi_{i+1} \not\models \varphi \vee \mathtt{X}(\varphi \mathtt{R} \psi)$ holds. 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: if the play is finite the last configuration consists of literals only. On the other hand, the last block of the sequence Ξ reaches this leaf. Therefore, the play must be infinite. In particular, the sequence Ξ is a trace, and by Lemma 12 it is either an E- or an A-trace.

Trace Ξ is an E-trace: Let n be minimal such that $(Q_i\Gamma_i, Q_{i+1}\Gamma_{i+1})$ is not spawning for all $i \geq n$. Therefore, all these quantifiers Q_is are E, and the set Γ_n is a singleton. By π we denote the subsequence of the play $(\mathcal{C}_i)_{i \geq n}$ which consists of nodes in $\mathcal{S}$ only. For a node $\mathcal{C}$ in the play, we write $\pi^{\mathcal{C}}$ to denote the suffix of π starting at $\widehat{\mathcal{C}}$. The trace contains a thread $\xi_0, \xi_1, \ldots$ such that

(ξ-1) $\pi^{\mathcal{C}_i} \not\models \xi_i$, and

(ξ-2) if $\xi_i = \varphi \mathtt{R} \psi$, the rule (ER) is applied to $\mathcal{C}_i$ with $\mathtt{E}\Gamma_i$ and ξ_i as principals, and $\pi^{\mathcal{C}_i} \not\models \psi$, then $\xi_{i+1} = \psi$.

for all $i \geq n$ and all formulas φ and ψ. Indeed, the thread can be constructed step by step. Obviously, there is a sequence of connected formulas $\xi_0, \ldots \xi_n$ within the trace because the set Γ_n is a singleton. The rules (E$\vee$), (E$\wedge$), (EU) and (ER) clearly preserve the properties (ξ-1) and (ξ-2). As for the rule (El), the formula ξ_i cannot be the principal literal because $\pi^{\mathcal{C}_i}$ is a countermodel of ξ_i but the literal survives until the next application of the model rules which defines the first state of $\pi^{\mathcal{C}_i}$. If the rule (EE) or (EA) is applied, the property (Ξ-2) keeps ξ_i from being the principal formula because the considered suffix is the trace is not spawning.

By Lemma 13, ξ is either a U- or an R-thread. In the first case, the thread ξ attests that the trace is bad although player 0 wins the play. Otherwise, suppose that ξ is an R-thread. By Lemma 14, there are $m \geq n$ and formulas φ and ψ such that $\xi_m = \varphi \mathtt{R} \psi$, and $\xi_i = \varphi \mathtt{R} \psi$ or $\xi_i = \mathtt{X}(\varphi \mathtt{R} \psi)$ for all $i \geq m$, Along the play $(\mathcal{C}_i)_{i \geq m}$, between any two consecutive applications of the rules ($\mathtt{X_0}$) or ($\mathtt{X_1}$), the rule (ER) must have been applied with $\xi_i = \varphi \mathtt{R} \psi$ and $Q_i\Gamma_i$ as principals for some $i \geq m$. The property (ξ-2) ensures that $\pi^{\mathcal{C}_i} \models \psi$. Since this is true for any such two consecutive applications, $\pi^{\mathcal{C}_i} \models \psi$ for all $i \geq m$. Therefore, $\pi^{\mathcal{C}_m}$ models $\varphi \mathtt{R} \psi$. But this situation contradicts the property (ξ-1) for i being one the infinity many positions on which the rule (ER) is applied to $Q_i\Gamma_i$ and ξ_i.

Trace Ξ is an A-trace: It suffices to show that Ξ is a bad trace. Suppose for the sake of a contradiction that Ξ contains an R-thread $(\xi_i)_{i \in \mathbb{N}}$. Let $n \in \mathbb{N}$ and $\varphi, \psi \in FL(\vartheta)$ such that $Q_i = \mathtt{A}$, $\xi_n = \varphi \mathtt{R} \psi$, and $\xi_i = \varphi \mathtt{R} \psi$ or $\xi_i = \mathtt{X}(\varphi \mathtt{R} \psi)$ for all $i \geq n$, cf. Lemma 14.

Along the play $(\mathcal{C}_i)_{i \geq n}$, between any two consecutive applications of the rules ($\mathtt{X_0}$) or ($\mathtt{X_1}$), the rule (AR) must have been applied such that ξ_i and $Q_i\Gamma_i$ are principal for some $i \in \mathbb{N}$. In this situation, the formula ξ_i is $\varphi \mathtt{R} \psi$. Because ξ_{i+1} is either $\varphi \mathtt{R} \psi$ or $\mathtt{X}(\varphi \mathtt{R} \psi)$, 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 $\Sigma \subseteq FL(\vartheta)$. Hence, thanks to ($\Xi$-6) we have $\pi_i \models \psi$. Because the block quantifier

remains $\mathtt{A}$, the properties (Ξ-4) and (Ξ-5) show that $\pi_n^j \models \psi$ for all $j \in \mathbb{N}$. Therefore, $\pi_n \models \varphi \mathtt{R} \psi$ holds. As the formula $\varphi \mathtt{R} \psi$ is ξ_n, the path π_n satisfies $\bigvee \Gamma_n$. However, this situation contradicts the property (Ξ-3). Thus, the considered play contains Ξ as a bad trace. □

4.2. **Completeness.** To show completeness, we need a witness for satisfiable $\mathtt{E}$-formulas. For this purpose, let $\mathcal{T} = (\mathcal{S}, s^*, \to, \lambda)$ be a transition system, $s \in \mathcal{S}$ be a state and ψ be a formula such that $s \models \mathtt{E}\psi$. We may assume a well-ordering $\lhd_{\mathcal{T}}$ on the set of paths in $\mathcal{T}$ [Zer04]. 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 $\pi \lhd_{\mathcal{T}} \xi_{\mathcal{T}}(s, \psi)$, $\pi(0) = s$ and $\pi \models \psi$.

A *$\mathcal{T}$-labelled (winning) strategy* is a (winning) strategy with every configuration being labelled with a state such that the root is labelled with s^*, and for every s-labelled configuration and every s'-labelled successor configuration it holds that $s \to s'$ if the corresponding rule application is $(\mathtt{X_1})$ or $(\mathtt{X_0})$ and $s = s'$ otherwise.

Theorem 19. Let $\vartheta \in \mathrm{CTL}^*$ be satisfiable. Then player 0 has a winning strategy for the satisfiability game $\mathcal{G}_\vartheta$.

Proof. Let ϑ be a formula, $\mathcal{T} = (\mathcal{S}, s^*, \to, \lambda)$ be a transition system, and $s^* \in \mathcal{S}$ be a state s.t. $\mathcal{T}, s^* \models \mathtt{E}\vartheta$. In the following we may omit the system $\mathcal{T}$ in front of the symbol $\models$.

We inductively construct an $\mathcal{S}$-labelled strategy for player 0 as follows. Starting with the labelled configuration $s^* : \mathtt{E}\vartheta$, we apply the rules in an arbitrary but eligible ordering systematically by preserving $s \models \Phi$ for every state-labelled configuration $s : \Phi$ and by preferring small formulas. In particular, the strategy is defined the following properties.

- (S-1) If the rule application to follow Φ is $(\mathtt{Al})$, $(\mathtt{AE})$ or $(\mathtt{AA})$, with $\mathtt{A}(\psi, \Sigma)$ being the principal block in Φ and ψ being the principal (state) formula, and $s \models \psi$, then the successor configuration of Φ follows ψ and discards the original $\mathtt{A}$-block.
- (S-2) If the rule application to follow Φ is $(\mathtt{EU})$, with $\mathtt{E}(\varphi \mathtt{U} \psi, \Pi)$ being the principal block in Φ and $\varphi \mathtt{U} \psi$ being the principal formula, then the successor configuration of Φ follows ψ iff $\xi_{\mathcal{T}}(s, (\varphi \mathtt{U} \psi) \wedge \bigwedge \Pi) \models \psi$.
- (S-3) If the rule application to follow Φ is $(\mathtt{E}\vee)$, with $\mathtt{E}(\psi_1 \vee \psi_2, \Pi)$ being the principal block in Φ and $\psi_1 \vee \psi_2$ being the principal formula, and $\xi_{\mathcal{T}}(s, (\psi_1 \vee \psi_2) \wedge \bigwedge \Pi) \models \psi_i$ for some $i \in \{1, 2\}$, then the successor configuration of Φ follows ψ_i.
- (S-4) If the rule application to follow Φ is $(\mathtt{X_0})$ and its successor configuration is labelled with a state s' such that $s \to s'$ and successor configuration Φ' then player 0 labels this successor with the state s'.
- (S-5) If player 1 applies the rule $(\mathtt{X_1})$ to a configuration Φ which is labelled with a state s and obtains successor configuration $\mathtt{E}\Pi, \Phi'$ then player 0 labels this successor with the state $\xi_{\mathcal{T}}(s, \mathtt{X}\Pi)(1)$.

Such a strategy exists because the property $s : \Phi$ can be maintained. Note that every finite play ends in a node labelled with consistent literals only. Clearly, player 0 wins such a play.

For the sake of contradiction, assume that player 0 does not win if she follows the strategy. Hence, there is an infinite labelled play $s_0 : \Phi_0$, $s_1 : \Phi_1$, … (with $s_0 = s^*$ and $\Phi_0 = \mathtt{E}\vartheta$) containing a bad trace $B_0, B_1, \ldots$. We define a *lift operation* $\widehat{i}$ that selects the next modal rule application as follows.

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

Due to Lemma 10, $\widehat{i}$ is well-defined for every i. Additionally, we define the *modal distance*

$$\delta(i,k) := |\{j \mid i \leq j < k \text{ and } j = \widehat{j}\}|$$

as well that counts the number of modal rule application between i and k. Every position i induces a generic path π_i by

$$\pi_i \colon j \mapsto s_{\min\{k \mid k \geq i \text{ and } \delta(i,k)=j\}}$$

and note that the path π_i indeed is well-defined for every i.

By Lemma 12, the bad trace is either an $\mathtt{A}$- or an $\mathtt{E}$-trace that is eventually not spawning, i.e. there is a position n such that $B_i \equiv \mathtt{E}\Pi_i$ or $B_i \equiv \mathtt{A}\Sigma_i$ for all $i \geq n$ with (B_i, B_{i+1}) being not spawning. Let n be the least of such kind.

Next, the bad trace gives rise to a $\mathtt{U}$-thread in it that is satisfied by the transition system. For this purpose we construct a $\mathtt{U}$-thread $\phi_0, \phi_1, \ldots$ in $B_0, B_1, \ldots$ such that all $i \geq n$ satisfy the following properties.

(ϕ-1) $\pi_i \models \phi_i$.

(ϕ-2) For all formulas φ and ψ we have: If $\phi_i = \varphi \mathtt{U} \psi$, $\phi_i \neq \phi_{i+1}$ and if $\pi_i \models \psi$, then $\phi_{i+1} = \psi$.

The construction of the thread depends on the kind of the trace.

Trace $B_0, B_1, \ldots$ is an $\mathtt{E}$-trace: The paths π_i and $\xi_{\mathcal{T}}(s_i, \Pi_i)$ coincide for all $i \geq n$ for two reasons. First, whenever a rules besides $(\mathtt{X_0})$ and $(\mathtt{X_1})$ justifies the move from the configuration Φ_i to Φ_{i+1} for $i \geq n$, then $\xi_{\mathcal{T}}(s_i, \Pi_i)$ and $\xi_{\mathcal{T}}(s_{i+1}, \Pi_{i+1})$ are equal. Second, this $\mathtt{E}$-trace overcomes the application of the rules $(\mathtt{X_0})$ and $(\mathtt{X_1})$. Thus, the minimal paths $\xi_{\mathcal{T}}$ define the labels $s_n, s_{n+1}, \ldots$ and, in this way, the paths π.

Since n is the *least* index s.t. (B_i, B_{i+1}) is not spawning for all $i \geq n$, the set Π_n has to be a singleton. Define ϕ_n to be the single formula in Π_n. Because $s_n \models \mathtt{E}\Pi_n$, the path $\xi_{\mathcal{T}}(s_n, \Pi_n)$ satisfies ϕ_n.

As the trace is assumed to be bad, it contains a $\mathtt{U}$-thread, say $\phi_0, \phi_1, \ldots$. The construction of the strategy ensures that $\xi_{\mathcal{T}}(s_i, \Pi_i) \models \phi_i$ for all $i \geq n$. Hence, $\pi_i \models \phi_i$. Additionally, the constraint (S-2) yields the property (ϕ-2).

Trace $B_0, B_1, \ldots$ is an $\mathtt{A}$-trace: Since n is the least index such that (B_i, B_{i+1}) is not spawning $i \geq n$, the set Σ_n has to be a singleton. Define ϕ_n to be the single formula in Σ_n.

For $i \geq n$ the formula ϕ_{i+1} bases on ϕ_i as follows. If $\widehat{i} = i$, that is, one of the modal rules $(\mathtt{X_0})$ and $(\mathtt{X_1})$ is to be applied next, then set $\phi_{i+1} = \phi'$ where $\phi_i = \mathtt{X}(\phi')$ for some formula ϕ'. Otherwise, $\widehat{i} \neq i$ holds. If B_i or ϕ_i is not principal in the rule instance then set $\phi_{i+1} := \phi_i$. Because (B_i, B_{i+1}) is not spawning, ϕ_{i+1} belongs to B_{i+1}. Otherwise, B_i and ϕ_i are principal. The formula ϕ is neither a literal nor an $\mathtt{E}$- nor an $\mathtt{A}$-formula, because otherwise the property (S-1) together with $\pi_i \models \phi_i$ would entail the end of this sequence of blocks or would show that the connection (B_i, B_{i+1}) is spawning. Thus, the applied rule is either $(\mathtt{AR})$, $(\mathtt{AU})$, $(\mathtt{A}\wedge)$ or $(\mathtt{A}\vee)$. 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 $\phi_i = \varphi \mathtt{U} \psi$, then set $\phi_{i+1} := \psi$ iff $\pi_i \models \psi$ and, otherwise, set ϕ_{i+1} to the other successor, that is φ or $\mathtt{X}(\varphi \mathtt{U} \psi)$, of ϕ_i in B_{i+1}. Finally, if $\phi_i = \psi_1 \wedge \psi_2$ or $\phi_i = \psi_1 \vee \psi_2$, then set $\phi_{i+1} := \psi_k$ s.t. ψ_k is connected to ϕ_i in B_{i+1} and $\pi_i \models \psi_k$.

Putting suitable formulas in front of the sequence $\phi_n, \phi_{n+1}, \ldots$ entails a thread in the trace $B_0, B_1, \ldots, B_n, B_{n+1}, \ldots$. By assumption the trace is bad and by Lemma 13, the thread is a $\mathtt{U}$-thread.

Since $\phi_0, \phi_1, \ldots$ is a $\mathtt{U}$-thread, there are formulas φ_0 and φ_1 such that $\phi_i = \varphi_0 \mathtt{U} \varphi_1$ for infinitely many indices i. The set

$$A := \{i > n \mid \phi_{i-1} = \mathtt{X}(\varphi_1 \mathtt{U} \varphi_2) \text{ and } \phi_i = \varphi_1 \mathtt{U} \varphi_2\}$$

is infinite by Lemma 10 and 14. Let $i_0 < i_1 < \ldots$ be the ascending enumeration of A. Between every two immediately consecutive elements either the rule $(\mathtt{X_1})$ or $(\mathtt{X_0})$ is applied exactly once. Therefore, $\pi^1_{i_j} = \pi_{i_{j+1}}$ for all indices $j \geq 0$. By property (ϕ-1) we have $\pi_{i_0} \models \varphi_1 \mathtt{U} \varphi_2$. Hence, there is a $k \geq 0$ such that $\pi^k_{i_0} \models \varphi_2$. In particular, $\pi_{i_k} \models \varphi_2$ and so $\pi_{i_k} \models \varphi_1 \mathtt{U} \varphi_2$. For some ℓ between i_k and i_{k+1} the formula ϕ_ℓ must be turned from $\varphi_1 \mathtt{U} \varphi_2$ into $\mathtt{X}(\varphi_1 \mathtt{U} \varphi_2)$ to finally pass the application of $(\mathtt{X_0})$ and $(\mathtt{X_1})$ at position $i_{k+1} - 1$. However, the property (ϕ-2) shows that ϕ_ℓ is just φ_2. □

5. A Decision Procedure for CTL*

5.1. **Using Deterministic Automata to Check the Winning Condition.** Plays can be represented as infinite words over a certain alphabet, and we will show that the language of plays that are won by player 0 is ω-regular, i.e. recognisable by a nondeterministic Büchi automaton for instance.

The goal is then to replace the global condition on plays of having only good traces by an annotation of the game configurations with automaton states and a global condition on these states. For instance, if the resulting automaton was of Büchi type, then the game would become a Büchi game: in order to solve the satisfiability game it suffices to check whether player 0 has a winning strategy in the game with the annotations in the sense that she can enforce plays which are accepted by the Büchi automaton for the annotations.

Now note that the automaton recognising such plays needs to be deterministic: suppose there are two plays uv and uw with a common prefix u s.t. both are accepted by an automaton $\mathcal{A}$. 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. This could be resolved by allowing two annotations on the nodes of the common prefix, but an infinite tree can have infinitely many branches and it is not clear how to bound the number of needed annotations. However, if $\mathcal{A}$ is deterministic then it has a unique run on the common prefix, and an annotation with a single state of a deterministic automaton suffices.

It is known that every ω-regular language can be recognised by a deterministic Muller [McN66], Rabin [Saf88] or parity automaton [Pit06]. A simple consequence of the last result is the fact that every game with an ω-regular winning condition can be reduced to a parity game. Thus, we could simply show that the winning conditions of the satisfiability games of Section 3 are ω-regular and appeal to this result as well as known algorithms for solving parity games in order to have a decision procedure for CTL*. While this does not seem avoidable entirely, it turns out that the application of this technique, which is not very efficient in practice, can be reduced to a minimum. The rest of this section is devoted to the analysis of the satisfiability games' winning conditions as a formal and ω-regular language with a particular focus on the question of determinisability.

In our proposed reduction to parity games we will use annotations with states from two different deterministic automata: one checks that all E-traces are good, the other one checks that all A-traces are good. The reason for this division is the fact that the former check is much simpler than the latter. It is possible to directly define a deterministic automaton that checks for absence of a bad E-trace. It is not clear at all though, how to directly define a deterministic automaton that checks for absence of a bad A-trace. We therefore use nondeterministic automata and known constructions for complementing and determinising them. The next part recalls the automata theory that is necessary for this, and in particular shows how these two automata used in the annotations can be merged into one.

5.2. **Büchi, co-Büchi and Parity Automata on Infinite Words.** We will particularly need the models of Büchi, co-Büchi and parity automata [GTW02].

Definition 20. A *nondeterministic parity automaton* (NPA) is a tuple $\mathcal{A} = (Q, \Sigma, q_0, \delta, \Omega)$ 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 $\Omega : Q \to \mathbb{N}$ a priority function. 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 $\max\{\Omega(q) \mid q = q_i \text{ for infinitely many } i \in \mathbb{N}\}$ is even, i.e. if the maximal priority of a state that is seen infinitely often in this run is even. The *language* of the NPA $\mathcal{A}$ is $L(\mathcal{A}) = \{w \mid$ there is an accepting run of $\mathcal{A}$ on $w\}$. The *index* of an NPA $\mathcal{A}$ is the number of different priorities occurring in it, i.e. $|\Omega[Q]|$. The *size* of $\mathcal{A}$, written as $|\mathcal{A}|$, is the number of its states.

Nondeterministic Büchi and *co-Büchi automata* (NBA / NcoBA) are special cases of NPA. An NBA is an NPA as above with $\Omega : Q \to \{1, 2\}$, and an NcoBA is an NPA with $\Omega : Q \to \{0, 1\}$. Hence, an accepting run of an NBA has infinitely many occurrences of a state with priority 2, and an accepting run of an NcoBA has almost only occurrences of states with priority 0. Traditionally, in an NBA the states with priority 2 are called the *final set*, and one defines an NBA as $(Q, \Sigma, q_0, \delta, F)$ where, in our terminology, $F := \{q \in Q \mid \Omega(q) = 2\}$. An NcoBA can equally defined with an acceptance set F rather than a priority function Ω, but then $F := \{q \in Q \mid \Omega(q) = 0\}$.

An NPA / NBA / NcoBA with transition relation δ is *deterministic* (DPA / DBA / DcoBA) if $|\{q' \mid (q, a, q') \in \delta\}| = 1$ for all $q \in Q$ and $a \in \Sigma$. In this case we may view δ as function from $Q \times \Sigma$ into Q.

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 [GTW02, Section 1.2] in the theory of ω-word automata.

Lemma 21. 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 $|\overline{\mathcal{A}}| = |\mathcal{A}|$.

In order to be able to turn presence of a bad trace—which may be easy to recognise using a nondeterministic automaton—into absence of such which is required by the winning condition, we need complementation of nondeterministic automata as well. Luckily, an NcoBA can be determinised into a DcoBA using the Miyano-Hayashi construction [MH84] which can easily be complemented into a DBA according to Lemma 21.

Theorem 22 ([MH84])**.** For every NcoBA $\mathcal{A}$ with n states there is a DBA $\overline{\mathcal{A}}$ with at most 3^n states s.t. $L(\overline{\mathcal{A}}) = \overline{L(\mathcal{A})}$.

NBA cannot be determinised into DBA, but into automata with stronger acceptance conditions [Saf88, Pit06, KW08, Sch09]. We are particularly interested in constructions that yield parity automata.

Theorem 23 ([Pit06])**.** For every NBA with n states there is an equivalent DPA with at most n^{2n+2} states and index at most $2n-1$.

For the decision procedure presented below we also need a construction that intersects a deterministic Büchi and a deterministic parity automaton. This will allow us to consider absence of bad `E`- and bad `A`-traces separately.

Lemma 24. For every DBA $\mathcal{A}$ with n states and DPA $\mathcal{B}$ with m states and index k there is a DPA $\mathcal{C}$ with at most $n \cdot m \cdot k$ many states and index at most $k+1$ s.t. $L(\mathcal{C}) = L(\mathcal{A}) \cap L(\mathcal{B})$.

Proof. Let $\mathcal{A} = (Q_1, \Sigma, q_1^0, \delta_1, F)$ and $\mathcal{B} = (Q_2, \Sigma, q_2^0, \delta_2, \Omega)$. Define $\mathcal{C}$ as $(Q_1 \times Q_2 \times \Omega[Q_2], \Sigma, (q_1^0, q_2^0, \Omega(q_2^0)), \delta, \Omega')$ where

$$\delta\big((q_1,q_2,p),a\big) \;:=\; \begin{cases} \big(\delta_1(q_1,a), \delta_2(q_2,a), \Omega(\delta_2(q_2,a))\big) & \text{, if } q_1 \in F \\ \big(\delta_1(q_1,a), \delta_2(q_2,a), \max\{p, \Omega(\delta_2(q_2,a))\}\big) & \text{, if } q_1 \notin F \end{cases}$$

Note that $\mathcal{C}$ simulates two runs of $\mathcal{A}$ and $\mathcal{B}$ in parallel on a word $w \in \Sigma^\omega$, and additionally records in its third component the maximal priority that has been seen in $\mathcal{B}$'s run since the last visit of a final state in the run of $\mathcal{A}$ if it exists. Thus, in order to determine whether or not both simulated runs are accepting it suffices to examine the priorities at those positions at which the $\mathcal{A}$-component is visiting a final state. In all other cases we choose a low odd priority.

$$\Omega'(q_1,q_2,p) \;:=\; \begin{cases} p+2 & \text{, if } q_1 \in F \\ 1 & \text{, if } q_1 \notin F \end{cases}$$

Then the highest priority occurring infinitely often in a run of $\mathcal{C}$ is even iff so is the one in the simulated run of $\mathcal{B}$ and $\mathcal{A}$ visits infinitely many final states at the same time.

It should be clear that the number of states in $\mathcal{C}$ is bounded by $n \cdot m \cdot k$, and that it uses at most one priority more than $\mathcal{B}$. □

To define an automaton which checks the absence of bad `A`-traces, we need the intersection of Büchi with co-Büchi automata as well as alphabet projections of Büchi automata.

Lemma 25. For every DBA $\mathcal{A}$ with n states and every DcoBA $\mathcal{B}$ with m states there is an NBA $\mathcal{C}$ with at most $n \cdot m \cdot 2$ states such that $L(\mathcal{C}) = L(\mathcal{A}) \cap L(\mathcal{B})$.

Proof. Let $\mathcal{A}$ be $(Q_1, \Sigma, q_1^0, \delta_1, F_1)$ and $\mathcal{B}$ be $(Q_2, \Sigma, q_2^0, \delta_2, F_2)$. Then define the NBA $\mathcal{C}$ as $(Q, \Sigma, (q_1^0, q_2^0, 0), \delta, F_1 \times F_2 \times \{1\})$ with $Q = (Q_1 \times Q_2 \times \{0\}) \cup (Q_1 \times F_2 \times \{1\})$, where δ realises the synchronous product of δ_1 and δ_2 on $Q_1 \times Q_2 \times \{0\}$ and on $Q_1 \times F_2 \times \{1\}$. In addition, for every transition from $(q_1, q_2, 0)$ to $(q_1', q_2', 0)$ there is also one with the same alphabet symbol to $(q_1', q_2', 1)$ if $q_2' \in F_2$. Note that this creates nondeterminism. □

Lemma 26. Let $\mathcal{C}$ be an NBA over the alphabet $\Sigma_A \times \Sigma_B$. There is a NBA $\mathcal{A}$ over the alphabet Σ_A such that $|\mathcal{A}| \leq |\mathcal{C}|$ and for all words $a_0 a_1 \ldots \in \Sigma_A^\omega$ it holds that

$$a_0 a_1 \ldots \in L(\mathcal{A}) \quad \text{iff} \quad \text{there is a word } b_0 b_1 \ldots \in \Sigma_B^\omega \text{ with } (a_0,b_0)(a_1,b_1)\ldots \in L(\mathcal{C})\,.$$

Proof. The automaton $\mathcal{C}$ is almost $\mathcal{A}$. Let δ_A be the transition relation of $\mathcal{A}$. Clearly, the set $\{(q,a,q') \mid (q,(a,b),q') \in \delta_A \text{ for some } b \in \Sigma_B\}$ is adequate as a transition relation for $\mathcal{C}$. □

5.3. **An Alphabet of Rule Applications.** Clearly, an infinite play in the game for some formula ϑ can be regarded as an ω-word over the alphabet of all possible configurations. This alphabet would have doubly exponential size in the size of the input formula. It is possible to achieve the goals stated above with a more concise alphabet.

Definition 27. A *rule application* in a play for ϑ is a pair of a configuration and one of its successors. Note that such a pair is entirely determined by the principal block and the principal formula of the configuration and a number specifying the successor. This enables a smaller symbolic encoding. For instance, the transition from the configuration $\mathtt{A}(\mathtt{E}\varphi, \Sigma), \Phi$ to the successor $\mathtt{A}\Sigma, \Phi$ in rule ($\mathtt{AE}$) can be represented by the quadruple $(\mathtt{A}, \{\mathtt{E}\varphi\} \cup \Sigma, \mathtt{E}\varphi, 1)$. The other possible successor 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 successor in rule ($\mathtt{X_1}$) is entirely determined by one of the $\mathtt{E}$-blocks in the configuration. Hence, let

$$\Sigma_\vartheta^{\mathsf{pl}} \;:=\; \big(\{\mathtt{A},\mathtt{E}\} \times 2^{FL(\vartheta)} \times FL(\vartheta) \times \{0,1\}\big) \;\cup\; \{\mathtt{Ett}, \mathtt{X}_0\} \;\cup\; \big(\{\mathtt{X}_1\} \times 2^{FL(\vartheta)}\big)$$

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

An infinite play $\pi = C_0, C_1, \ldots$ then induces a word $\pi' = r_0, r_1, \ldots \in (\Sigma_\vartheta^{\mathsf{pl}})^\omega$ in a straight-forward way: r_i is the symbolic representation of the configuration/successor pair (C_i, C_{i+1}). We will not formally distinguish between an infinite play π and its induced ω-word π' over $\Sigma_\vartheta^{\mathsf{pl}}$.

For every $r \in \Sigma_\vartheta^{\mathsf{pl}}$ let $\mathrm{con}_r^{\mathtt{E}}(\cdot)$ be a partial function from $\mathtt{E}$-blocks to $\mathtt{E}$-blocks which satisfies the connection relation $\rightsquigarrow$ and avoids spawning connections. Thus, the function is undefined for $r = \mathtt{Ett}$ and the argument $\mathtt{E}\emptyset$, only. For all other parameters and arguments the function is uniquely defined.

5.4. **DPA for the Absence of Bad $\mathtt{A}$-Traces.** An $\mathtt{A}$-trace-marked play is a (symbolic representation of a) play together with an $\mathtt{A}$-trace therein. It can be represented as an infinite word over the extended alphabet

$$\Sigma_\vartheta^{\mathsf{tmp}} := \Sigma_\vartheta^{\mathsf{pl}} \times \{\mathtt{A},\mathtt{E}\} \times 2^{FL(\vartheta)}\,.$$

The second and the last components of the alphabet simply name a block on the marked trace. Note that these components are half a step behind the first component because the latter links between two consecutive configuration. Remember that an $\mathtt{A}$-trace can proceed through finitely many $\mathtt{E}$-blocks before it gets trapped in $\mathtt{A}$-blocks only. We define a co-Büchi automaton $\mathcal{C}_\vartheta^{\mathtt{A}}$ which recognises exactly those $\mathtt{A}$-trace-marked plays which contain an $\mathtt{R}$-thread in the marked trace. It is $\mathcal{C}_\vartheta^{\mathtt{A}} = (\{\mathtt{W}\} \cup FL_{\mathtt{R}}(\vartheta), \Sigma_\vartheta^{\mathsf{tmp}}, \mathtt{W}, \delta, FL_{\mathtt{R}}(\vartheta))$. We describe the transition relation δ intuitively. A formal definition can easily be derived from this. Starting in the waiting state $\mathtt{W}$ it eventually guesses a formula of the form $\psi_1 \mathtt{R} \psi_2$ which occurs in the marked $\mathtt{A}$-trace. It then tracks this formula in its state for as long as it is unfolded with rule ($\mathtt{AR}$) and remains in the marked trace. If it leaves the marked trace in the sense that the trace proceeds through a block which does not contain this subformula anymore, or an $\mathtt{E}$-block occurs as part of the marked trace then $\mathcal{C}_\vartheta^{\mathtt{A}}$ simply stops. The following proposition is easily seen to be true using Definition 9 and Lemma 14.

Lemma 28. Let $w \in (\Sigma_\vartheta^{\mathsf{tmp}})^\omega$ be an $\mathtt{A}$-trace-marked play of a game for ϑ. Then $w \in L(\mathcal{C}_\vartheta^{\mathtt{A}})$ iff the marked trace of w contains an $\mathtt{R}$-thread.

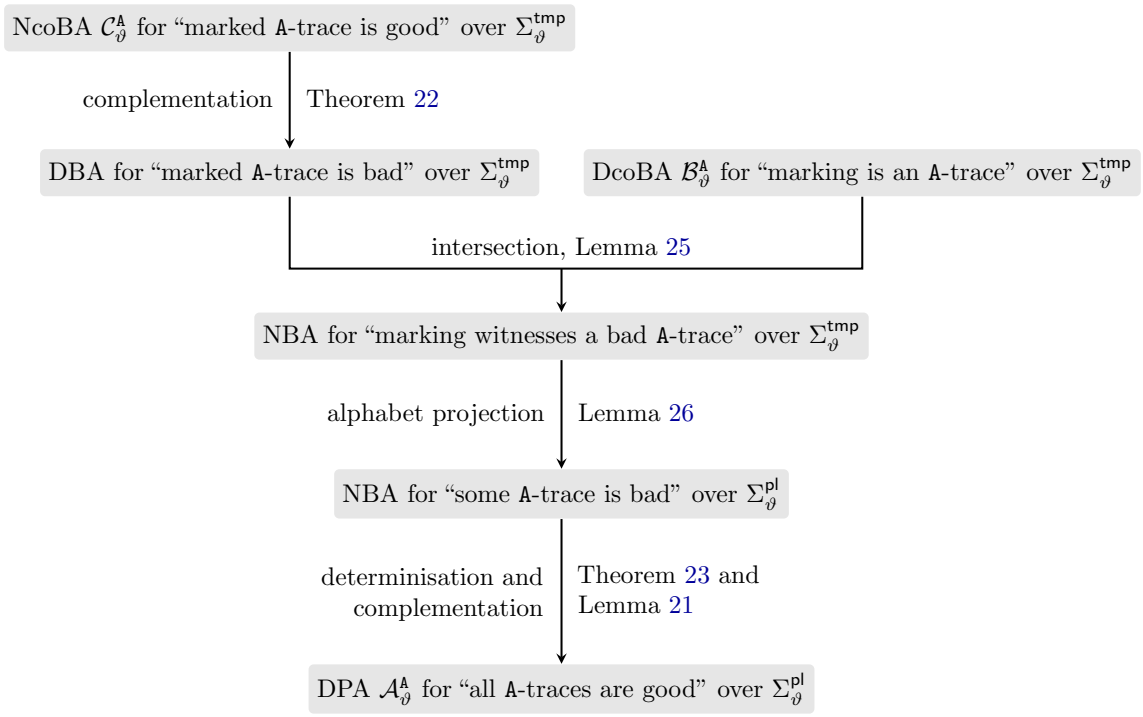


Figure 4: Construction of the DPA for Theorem 30.

On the way to construct an automaton which recognises plays without bad A-traces we need to eliminate the restriction on w in the previous lemma. In other words, an automaton is needed which decides whether or not the annotated sequence of blocks is an A-trace.

Lemma 29. There is a DcoBA $\mathcal{B}^{\mathtt{A}}_\vartheta$ over $\Sigma^{\mathsf{tmp}}_\vartheta$ of size $\mathcal{O}(2^{|\vartheta|})$ such that the equivalence

$$((r_i, Q_i, \Delta_i))_{i\in\mathbb{N}} \in L(\mathcal{B}^{\mathtt{A}}_\vartheta) \quad \text{iff} \quad (Q_i\Delta_i)_{i\in\mathbb{N}} \text{ is an A-trace in the play } (r_i)_{i\in\mathbb{N}}$$

holds for all infinite plays $r_0, r_1, \ldots \in \Sigma^{\mathsf{pl}}_\vartheta$ and all sequences of blocks $(Q_i\Delta_i)_{i\in\mathbb{N}}$.

Proof. Take as $\mathcal{B}^{\mathtt{A}}_\vartheta$ the deterministic co-Büchi automaton with states

$$Q := \{\mathtt{E}, \mathtt{A}\} \times 2^{FL(\vartheta)},$$

initial state $(\mathtt{E}, \{\vartheta\})$ and final states $\{\mathtt{A}\} \times 2^{FL(\vartheta)}$. The automaton verifies that the last two components of the input indeed form an A-trace. For this purpose, the state bridges between two successive blocks in the input sequence. Due to the co-Büchi acceptance condition, the input is accepted if the block quantifier eventually remains A. However, these properties define an A-trace.

Formally, given a state (Q_0, Δ_0) and a letter (r, Q_1, Δ_1), a move into the state (Q_2, Δ_2) is only possible iff $Q_0 = Q_1$, $\Delta_0 = \Delta_1$, and the rule instance r transfers the block $Q_1\Delta_1$ into the block $Q_2\Delta_2$. Note that the sequence of blocks might end if the rules $(\mathtt{Ett})$ and $(\mathtt{X_1})$ are applied. In such a situation, the automaton gets stuck and rejects thereby. □

Figure 4 explains how the previously defined automata $\mathcal{C}^{\mathtt{A}}_\vartheta$ and $\mathcal{B}^{\mathtt{A}}_\vartheta$ can then be transformed into a deterministic parity automaton, called $\mathcal{A}^{\mathtt{A}}_\vartheta$, that checks for presence of an R-thread in all A-traces of a given play. It is obtained using complementation twice, intersection and the

projection of the alphabet $\Sigma_\vartheta^{\mathsf{tmp}}$ to $\Sigma_\vartheta^{\mathsf{pl}}$. The four automata shown at the top are defined over the extended alphabet of plays with marked traces, whereas the others work on the alphabet $\Sigma_\vartheta^{\mathsf{pl}}$ of symbolic rule applications only. Almost all operations keep the automata small besides the determinisation. All in all, we obtain the following property.

Theorem 30. For every CTL*-formula ϑ there is a DPA $\mathcal{A}_\vartheta^{\mathtt{A}}$ of size $2^{2^{\mathcal{O}(|\vartheta|)}}$ and of index $2^{\mathcal{O}(|\vartheta|)}$ s.t. for all plays $\pi \in (\Sigma_\vartheta^{\mathsf{pl}})^\omega$ we have: $\pi \in L(\mathcal{A}_\vartheta^{\mathtt{A}})$ iff π does not contain a bad $\mathtt{A}$-trace.

5.5. **DBA for the Absence of Bad $\mathtt{E}$-Traces.** Remember that a bad $\mathtt{E}$-trace is one that contains a $\mathtt{U}$-thread. It is equally possible to construct an NcoBA which checks in a play for such a trace and then use complementation and determinisation constructions as it is done for $\mathtt{A}$-traces. However, it is also possible to define a DBA $\mathcal{A}_\vartheta^{\mathtt{E}}$ directly which accepts a play iff it does not contain a bad $\mathtt{E}$-trace. This requires a bit more insight into the combinatorics of plays but leads to smaller automata in the end.

Let $\varphi_0 \mathtt{U} \psi_0, \ldots, \varphi_{k-1} \mathtt{U} \psi_{k-1}$ be an enumeration of all $\mathtt{U}$-formulas in ϑ. The DBA $\mathcal{B}_\vartheta$ consists of the disjoint union of k components $C_0, \ldots, C_{k-1}$ with $C_i = \{i\} \cup \{i\} \times 2^{FL(\vartheta)}$. In the i-th component, state i is used to wait for either of two occurrences: the i-th $\mathtt{U}$-formula gets unfolded or one of the rules for $\mathtt{X}$-formulas is being seen. In the first case the automaton starts to follow the thread of this particular $\mathtt{U}$-formula. In the second case, the automaton starts to look for the next $\mathtt{U}$-formula in line to check whether it forms a thread. Hence, the transitions in state i are the following.

$$\delta(i,r) = \begin{cases} (i,\Pi) & \text{if } r = (\mathtt{E},\Pi,\varphi_i \mathtt{U} \psi_i, 1) \\ (i+1) \bmod k & \text{if } r = \mathtt{X}_0 \text{ or } r = (\mathtt{X}_1,\Pi) \text{ for some } \Pi \\ i & \text{otherwise} \end{cases}$$

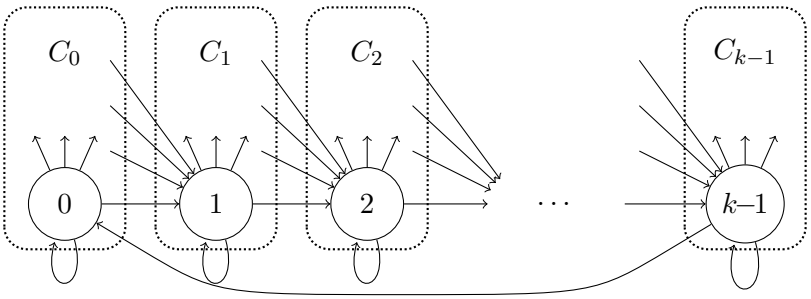
In order to follow a thread of the i-th $\mathtt{U}$-formula, the automaton uses the states of the form $\{i\} \times 2^{FL(\vartheta)}$ in which it can store the block that the current formula on the thread occurs in. It then only needs to compare this block to the principal block of the next rule application to decide whether or not this block has been transformed. If it has been then the automaton changes its state accordingly, otherwise it remains in the same state because the next rule application has left that block unchanged. Once a rule application terminates the possible thread of the i-th $\mathtt{U}$-formula, the automaton starts observing the next $\mathtt{U}$-formula in line. There are two possibilities for this: either the next rule application fulfils the $\mathtt{U}$-formula, or the $\mathtt{E}$-trace simply ends, for instance through an application of rule $(\mathtt{X}_1)$.

$$\delta((i,\Pi),r) = \begin{cases} (i+1) \bmod k & \text{if } r = (\mathtt{E},\Pi,\varphi_i \mathtt{U} \psi_i, 0) \\ (i,\Pi') & \text{otherwise, if } \mathrm{con}_r^{\mathtt{E}}(\mathtt{E}\Pi) = \mathtt{E}\Pi' \\ (i+1) \bmod k & \text{otherwise} \end{cases}$$

where $\mathrm{con}_r^{\mathtt{E}}$ is defined at the end of Subsection 5.3. The function δ is always defined as the second component of the state contains $\varphi_i \mathtt{U} \psi_i$ or $\mathtt{X}(\varphi_i \mathtt{U} \psi_i)$ whenever the first component is i.

Note that there is no transition for the case of the next rule being $(\mathtt{X}_0)$ because it only applies when there is no $\mathtt{E}$-block which is impossible if the automaton is following an $\mathtt{U}$-formula inside an $\mathtt{E}$-trace.

It is helpful to depict the transition structure graphically.



Note that every occurrence of rule ($\mathtt{X_0}$) or ($\mathtt{X_1}$) sends this automaton from any state i into the next component modulo k. Furthermore, when unfolding the i-th $\mathtt{U}$-formula in state i, it moves up into the component C_i where it follows the $\mathtt{E}$-trace that it is in. From this component it can only get to state $i+1 \bmod k$ if this $\mathtt{U}$-formula gets fulfilled.

Thus, since any infinite play must contain infinitely many applications of rule ($\mathtt{X_0}$) or ($\mathtt{X_1}$), there are only two possible types of runs of this automaton on such plays: those that eventually get trapped in some component $C_i \setminus \{i\}$, and those that visit all of $0, 1, \ldots, k-1$ infinitely often in this order.

It remains to be seen that this automaton—equipped with a suitable acceptance condition—recognises exactly those plays that do not contain a bad $\mathtt{E}$-trace.

Theorem 31. For every CTL* formula ϑ with k $\mathtt{U}$-subformulas there is a DBA $\mathcal{A}^{\mathtt{E}}_\vartheta$ of size at most $k \cdot (1 + 2^{|FL(\vartheta)|})$ s.t. for all plays $\pi \in \Sigma^\omega_\vartheta$: $\pi \in L(\mathcal{A}^{\mathtt{E}}_\vartheta)$ iff π does not contain a bad $\mathtt{E}$-trace.

Proof. As above, suppose that $\varphi_0 \mathtt{U} \psi_0, \ldots, \varphi_{k-1} \mathtt{U} \psi_{k-1}$ are all the $\mathtt{U}$-formulas occurring in $FL(\vartheta)$. Let $\mathcal{A}^{\mathtt{E}}_\vartheta := (C_0 \cup \ldots \cup C_{k-1}, \Sigma_\vartheta, 0, \delta, \{0\})$ be a Büchi automaton whose state set is the (disjoint) union of the components defined above and whose transition relation δ is also as defined above. It is easy to check that $\mathcal{A}^{\mathtt{E}}_\vartheta$ is indeed deterministic and of the size that is stated above. It remains to be seen that it is correct.

Let π be play. First we prove completeness, i.e. suppose that $\pi \notin L(\mathcal{A}^{\mathtt{E}}_\vartheta)$. Observe that in states of the form i it can always react to any input symbol whereas in states of the form (i, Π) it can react to all input symbols apart from ($\mathtt{X_0}$). However, such states are only reachable from states of the former type by reading a symbol of the form $(\mathtt{E}, \Pi, \varphi \mathtt{U} \psi, 1)$ which is only possible when there is an $\mathtt{E}$-block to which this rule is being applied. Furthermore, the automaton only stays in such states for as long as this block still contains this $\mathtt{U}$-formula, and $\mathtt{E}$-blocks can only disappear with rule ($\mathtt{Ett}$) when they become empty. Thus, $\mathcal{A}^{\mathtt{E}}_\vartheta$ has a (necessarily unique) run on every play, and π can therefore only be rejected if this run does not contain infinitely many occurrences of state 0.

Next we observe that $\mathcal{A}^{\mathtt{E}}_\vartheta$ cannot get trapped in a state of the form i because every infinite play contains infinitely many applications of rule ($\mathtt{X_0}$) or ($\mathtt{X_1}$)—cf. Lemma 10—which send it to state $(i+1) \bmod k$. Thus, in order not to accept π it would have to get trapped in some component of states of the form (i, Π) for a fixed i. However, it only gets there when the i-th $\mathtt{U}$-formula gets unfolded inside an $\mathtt{E}$-block, and it leaves this component as soon as this formula gets fulfilled. Thus, if it remains inside such a component forever, there must be an $\mathtt{U}$-thread inside $\mathtt{E}$-blocks, i.e. a bad $\mathtt{E}$-trace.

For soundness suppose that π contains a bad $\mathtt{E}$-trace. We claim that $\mathcal{A}^{\mathtt{E}}_\vartheta$ must get trapped in some component $C_i \setminus \{i\}$. Since this does not contain any final states, it will not accept π. Now note that at any moment in a play, all $\mathtt{U}$-formulas which are top-level in some

$\mathtt{E}$-block need to be unfolded with rule ($\mathtt{EU}$) before rule ($\mathtt{X_0}$) or ($\mathtt{X_1}$) can be applied. Thus, if $\mathcal{A}^{\mathtt{E}}_{\vartheta}$ is in some state i, and the i-th $\mathtt{U}$-formula occurs inside an $\mathtt{E}$-block at top-level position, then it will move to the component $C_i \setminus \{i\}$ instead of to $(i+1) \bmod k$ because the latter is only possible with a rule that occurs later than the rule which triggers the former transition.

As observed above, $\mathcal{A}^{\mathtt{E}}_{\vartheta}$ cannot remain in the only final state 0 forever. In order to visit it infinitely often, it has to visit all states $0, 1, \ldots, k-1$ infinitely often in this order. Thus, if there is a bad $\mathtt{E}$-trace with an $\mathtt{U}$-thread formed by the i-th $\mathtt{U}$-formula then there will eventually be a moment in which this i-th $\mathtt{U}$-formula gets unfolded and $\mathcal{A}^{\mathtt{E}}_{\vartheta}$ is trapped in some component $C_j \setminus \{j\}$ for $j \neq i$ and the rest of the run, or it is in state i. If the latter is the case then it gets trapped in $C_i \setminus \{i\}$ for the rest of the run before the next application of rule ($\mathtt{X_0}$) or ($\mathtt{X_1}$). In either case, π is not accepted. □

5.6. **The Reduction to Parity Games.** A *parity game* is a game $\mathcal{G} = (V, V_0, E, v_0, \Omega)$ s.t. (V, E) is a finite, directed graph with total edge relation E. V_0 denotes the set of nodes owned by player 0, and we write $V_1 := V \setminus V_0$ for its complement. The node $v_0 \in V$ 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 determinacy meaning that there is even a positional winning strategy for node v for one of the two player [EJ91]. The problem of *solving* a parity game is to determine which player has a winning strategy for v_0. It is solvable [Sch07] in time polynomial in $|V|$ and exponential in $|\Omega[V]|$.

Definition 32. Let ϑ be a state formula, $\mathcal{A}^{\mathtt{A}}_{\vartheta}$ be the DPA deciding absence of bad $\mathtt{A}$-traces according to Theorem 30, $\mathcal{A}^{\mathtt{E}}_{\vartheta}$ be the DBA deciding absence of bad $\mathtt{E}$-traces according to Theorem 31 and $\mathcal{A}_{\vartheta} = (Q, \Sigma^{\mathsf{pl}}_{\varphi}, q_0, \delta, \Omega)$ the DPA recognising the intersection of the languages of $\mathcal{A}^{\mathtt{A}}_{\vartheta}$ and $\mathcal{A}^{\mathtt{E}}_{\vartheta}$ according to Lemma 24. The *satisfiability parity game* for ϑ is $\mathcal{P}_{\vartheta} = (V, V_0, v_0, E, \Omega')$, defined as follows.

- $V := \mathit{Conf}(\vartheta) \times Q$
- $V_1 := \{(C, q) \in V \mid \text{rule } (\mathtt{X_1}) \text{ applies to } C\}$
- $V_0 := V \setminus V_1$
- $v_0 := (\mathtt{E}\vartheta, q_0)$
- $((C, q), (C', q')) \in E$ iff (C, C') is an instance of a rule application which is symbolically represented by $r \in \Sigma^{\mathsf{pl}}_{\vartheta}$ 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, winning strategies in the satisfiability games and the satisfiability parity games basically coincide.

Theorem 33. Player 0 has a winning strategy for $\mathcal{P}_\vartheta$ iff player 0 has a winning strategy for $\mathcal{G}_\vartheta$.

Proof. Let π be a play $(C_0, q_0), (C_1, q_1), \ldots$ of $\mathcal{P}_\vartheta$, and let $\pi' = C_0, C_1, \ldots$ be its projection onto the first components which ends at the first configuration on which no rule can be applied. The sequence π' is indeed a play in $\mathcal{G}_\vartheta$. Note that this projection is invertible: for every play π' in $\mathcal{G}_\vartheta$ there is a unique annotation with states of the deterministic automaton $\mathcal{A}_\vartheta$ leading to a play π in $\mathcal{P}_\vartheta$. Now we have the following.

$$\begin{aligned} \pi \text{ is won by player 0} &\Leftrightarrow \pi' \text{ is accepted by } \mathcal{A}_\vartheta, \text{ or } \pi' \text{ ends in a consistent set of literals} \\ &\Leftrightarrow \pi' \text{ is won by player 0} \end{aligned}$$

Thus, the projection of a winning strategy for player 0 in $\mathcal{P}_\vartheta$ is a winning strategy for her in $\mathcal{G}_\vartheta$, and conversely, every winning strategy there can be annotated with automaton states in order for form a winning strategy for her in $\mathcal{P}_\vartheta$. □

Corollary 34. Deciding satisfiability for some $\vartheta \in \mathrm{CTL}^*$ is in 2EXPTIME.

Proof. The number of states in $\mathcal{P}_\vartheta$ is bounded by

$$|\mathit{Conf}(\vartheta)| \cdot |Q| = 2^{2^{\mathcal{O}(|\vartheta|)}} \cdot 2^{2^{\mathcal{O}(|\vartheta|)}} \cdot 2^{\mathcal{O}(|\vartheta|)} \cdot |\vartheta| \cdot (1 + 2^{\mathcal{O}(|\vartheta|)}) = 2^{2^{\mathcal{O}(|\vartheta|)}}$$

Note that the out-degree of the parity game graph is at most $2^{|\vartheta|}$ because of rule $(\mathtt{X}_1)$. The game's index is $2^{\mathcal{O}(|\vartheta|)}$. It is known that parity games of size m and index k can be solved in time $m^{\mathcal{O}(k)}$ [Sch07] from which the claim follows immediately. □

5.7. **Model Theory.**

Corollary 35. Any satisfiable CTL* formula ϑ has a model of size at most $2^{2^{\mathcal{O}(|\vartheta|)}}$ and branching-width at most $2^{|\vartheta|}$.

Proof. Suppose ϑ is satisfiable. According to Theorems 17 and 33, player 0 has a winning strategy for $\mathcal{P}_\vartheta$. It is well-known that she then also has a positional winning strategy [Zie98]. A positional strategy can be represented as a finite graph of size bounded by the size of the game graph. A model for ϑ can be obtained from this winning strategy as it is done exemplarily in Section 3 and in detail in the proof of Theorem 18. The upper-bound on the branching-width is given by the fact that rule $(\mathtt{X}_1)$ can have at most $2^{|\vartheta|}$ many successors. □

The exponential branching-width stated in Corollary 35 can be improved to a linear one by restricting the rule applications. 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 configuration which do not have $\mathtt{X}$ as their outermost connectives. Following the proof of Theorem 19, any ordering on the rules does not affect the completeness.

As a measure of a configuration 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 configuration and such that $\mathtt{E}\{\varphi\}$ is not a block in this configuration. This measure is bounded by $|\vartheta|+1$ at the initial configuration $\mathtt{E}\{\vartheta\}$ and at every successor of the rules $(\mathtt{X_0})$ and $(\mathtt{X_1})$.

The size restriction ensures that any rule instance apart from $(\mathtt{X_0})$ and $(\mathtt{X_1})$ weakly decreases the measure. First, we consider the contribution of formulas to the measure. An inspection of the rules entails that any subformula $\mathtt{E}\varphi$ which contributes to the measure of the configuration at the top of a rule occurs at the bottom as a subformula. For the sake of contradiction, assume that $\mathtt{E}\varphi$ does not contribute to the measure of the configuration at the bottom. 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. Secondly, only the rules $(\mathtt{EE})$ and $(\mathtt{AE})$ can produce new $\mathtt{E}$-blocks. If a formula $\mathtt{E}\varphi$ is excluded from the measure of the configuration at the bottom then and only then $\mathtt{E}\{\varphi\}$ is a block in this configuration. Therefore, in the positive case this block is not new at the top. And in the negative case the new block at the top is paid by the formula at the bottom and prevents other instances of this formula at the top from being counted.

Putting this together with the argumentation in Corollary 35 yields the following.

Corollary 36. Any satisfiable CTL* formula ϑ has a model of size at most $2^{2^{\mathcal{O}(|\vartheta|)}}$ and branching-width at most $|\vartheta|$.

These upper bounds are asymptotically optimal, c.f. the proof of the 2EXPTIME–lower-bound [VS85] and the satisfiable formula $\bigwedge_{i=1}^{n} \mathtt{EX}(\neg p_i \wedge p_{i+1}) \wedge \bigwedge_{i=1}^{n} \mathtt{AX}(p_i \rightarrow p_{i+1})$ which forces any model to be of branching-width n.

6. On Fragments of CTL*

The logic CTL* has two prominent fragments: CTL$^+$ and CTL. These logics allow refining the decision procedure detailed in Section 5. The obtained procedures are conceptionally simpler and have an optimal time-complexity.

6.1. The Fragment CTL$^+$.

The satisfiability problem for CTL$^+$ is 2EXPTIME-hard [JL03] and hence —as a fragment of CTL*— it is also 2EXPTIME-complete. Nevertheless, CTL$^+$ is as expressive as CTL [EH85]. Hence, the question arises whether the lower expressivity compared to CTL* leads to a simpler decision procedure.

As CTL$^+$ is a fragment of CTL* we can apply the introduced games. However, the occurring formulas will not necessarily be CTL$^+$-formulas again, because the fixpoint rules can prefix an $\mathtt{X}$-constructor to the respective $\mathtt{U}$- or $\mathtt{R}$-formula. Nevertheless, the grammar for CTL$^+$ can be expanded accordingly. The new kinds are attached to line (2.4).

$$\psi \quad ::= \quad \varphi \mid \psi \vee \psi \mid \psi \wedge \psi \mid \mathtt{X}\varphi \mid \varphi \mathtt{U} \varphi \mid \varphi \mathtt{R} \varphi \mid \mathtt{X}(\varphi \mathtt{R} \varphi) \mid \mathtt{X}(\varphi \mathtt{U} \varphi) \tag{2.4'}$$

The lines (2.3) and (2.4') now define the grammar which every game follows. The usage of these new formulas does not affect any of the used asymptotic measures. The restriction to CTL$^+$ does not allow major simplification for the automata $\mathcal{A}^{\mathtt{E}}_{\vartheta}$ constructed in Subsection 5.5. However, the automata $\mathcal{A}^{\mathtt{A}}_{\vartheta}$ which rejects plays containing bad $\mathtt{A}$-traces can be essentially simplified: The refined construction bases on a coBüchi- instead of a Büchi-determinisation, and hence leads to a simpler acceptance condition. Due to Theorem 22 it suffices to construct an exponentially sized NcoBA which detects an $\mathtt{A}$-trace which does not contain any $\mathtt{R}$-thread.

For the rest of the subsection, fix a CTL$^+$-formula ϑ and consider an infinite play in the game $\mathcal{G}_\vartheta$. Let $(Q_i\Delta_i)_{i\in\mathbb{N}}$ be a trace in this play. A position i_0 in this trace is called $\mathtt{X}$-*stable* iff —firstly— the index i_0 addresses some top configuration either of the rule $(\mathtt{X_0})$ or of $(\mathtt{X_1})$, and —secondly— the connection $Q_i\Delta_i \rightsquigarrow Q_{i+1}\Delta_{i+1}$ is not spawning for every $i \geq i_0$. By Lemma 10 and 12 every trace has infinitely many $\mathtt{X}$-stable indices.

Lemma 37. Let $(Q_i\Delta_i)_{i\in\mathbb{N}}$ be a trace, let i_0 be one of its $\mathtt{X}$-stable positions, let $N \in \mathbb{N}$ and let $(\psi_i)_{i\leq N}$ be a sequence of connected formulas in the trace. If there is an $i_1 \geq i_0$ such that ψ_{i_1} is a state formula then ψ_j is a state formula for all $j \geq i_1$.

Proof. Every state formula in this trace eventually either disappears entirely —by the rule $(\mathtt{Al})$ for instance—, forms a new block *outside* the trace —by rule $(\mathtt{EE})$ for instance—, or get decomposed into a smaller state formula —by rule $(\mathtt{E}\vee)$ for instance—. One of these cases must happen before the rules $(\mathtt{X_0})$ or $(\mathtt{X_1})$ are applied. Finally, one of the two modal rules must be applied eventually due to Lemma 10. □

For every thread Lemma 14 reveals a position which describes the corresponding suffix of the thread. Next, we can strengthen this position to an $\mathtt{X}$-stable position.

Lemma 38. Let $(Q_i\Delta_i)_{i\in\mathbb{N}}$ be a trace and let i_0 be one of its $\mathtt{X}$-stable positions. Every thread $(\psi_i)_{i\in\mathbb{N}}$ in the trace satisfies: $\psi_i = \psi_{i_0}$ or $\psi_i = \mathtt{X}\psi_{i_0}$, for all $i \geq i_0$.

Proof. The thread cannot hit any state formula, because by Lemma 37 the thread would violate Lemma 14. The application of the rule $(\mathtt{X_0})$ or $(\mathtt{X_1})$ to the configuration at index $i_0 - 1$ entails that ψ_{i_0} is a $\mathtt{U}$- or an $\mathtt{R}$-formula. In particular along the remaining suffix, the thread must not hit a state formula. Therefore, the formula ψ_i is either ψ_{i_0} or $\mathtt{X}\psi_{i_0}$ for all $i \geq i_0$. □

Theorem 39. Let $(Q_i\Delta_i)_{i\in\mathbb{N}}$ be an $\mathtt{A}$-trace and let i_0 be one of its $\mathtt{X}$-stable positions. We have that: the trace is bad, iff Δ_i does not contain any $\mathtt{R}$- or $\mathtt{XR}$-formula for some $i \geq i_0$.

Proof. It suffices to show that the trace contains an $\mathtt{R}$-thread iff Δ_i contains a $\mathtt{R}$- or $\mathtt{XR}$-formula for every $i \geq i_0$. The "only if" direction is a consequence of Lemma 38. As for the "if" direction, every $\mathtt{R}$- or $\mathtt{XR}$-formula can be reached from the initial configuration of the game by a connected sequence of formulas. Due to König's lemma there is a corresponding infinite sequence. By Lemma 13, this sequence is either a $\mathtt{U}$- or an $\mathtt{R}$-thread. If the latter case applies, we are done. In the first case, infinitely many of the said $\mathtt{R}$- and $\mathtt{XR}$-formulas are reachable from a $\mathtt{U}$-formula. Due to the grammar, a state formula must occur between the $\mathtt{U}$-formula and each of the considered $\mathtt{R}$- and $\mathtt{XR}$-formulas. However, this situation contradicts Lemma 37. □

The previous theorem is specific for CTL$^+$. For CTL* an $\mathtt{A}$-trace $(Q_i\Delta_i)_{i\in\mathbb{N}}$ can be good, even if Δ_i does not contain any $\mathtt{R}$- or $\mathtt{XR}$-formula for some $i \geq i_0$. Indeed, the $\mathtt{R}$-formula witnessing that the trace is good might be hosted within a $\mathtt{U}$-formula. A play might delay the fulfillment of this $\mathtt{U}$-formula by several applications of $(\mathtt{X_0})$ or $(\mathtt{X_1})$.

Theorem 39 allows us to do without the determinisation of Büchi-automata as used to construct $\mathcal{A}^{\mathtt{A}}_\vartheta$ in Subsection 5.4. Indeed, there is a NcoBA which accepts every trace which contains a bad $\mathtt{A}$-traces. Define the NcoBA $\mathcal{C}^{\mathtt{A},\mathrm{CTL}^+}_\vartheta$ by $(Q, \Sigma^{\mathsf{br}}_\vartheta, \mathtt{W}, \delta, F)$ where

$$Q \;:=\; \{\mathtt{W}\} \cup \Big(2^{FL(\vartheta)} \times \{0,1,2\}\Big), \quad \text{and} \quad F \;:=\; 2^{FL(\vartheta)} \times \{2\}\,.$$

The automaton starts in the waiting state $\mathtt{W}$. Every $\mathtt{A}$-trace contains a spawning connection for the last time — at least one such connection occurs because the initial configuration is an $\mathtt{E}$-block. This connection is generated either by the rule $(\mathtt{AA})$ or by $(\mathtt{EA})$. Thus, $\mathcal{C}_\vartheta^{\mathtt{A},\mathrm{CTL}^+}$ eventually jumps after the corresponding input symbol, that is $(\mathtt{A}, \_, \mathtt{A}\varphi, 0)$ or $(\mathtt{E}, \_, \mathtt{A}\varphi, \_)$, into the state $(\{\varphi\}, 0)$. Then, $\mathcal{C}_\vartheta^{\mathtt{A},\mathrm{CTL}^+}$ tries to successively guess an $\mathtt{A}$-trace using the first component. If the block sequence stops or is spawning then the automaton rejects. The value 0 in the second component indicates the range between the last spawning connection and the first application of rules $(\mathtt{X_0})$ and $(\mathtt{X_1})$ afterwards. This application marks an $\mathtt{X}$-stable position. The flags 1 and 2 are responsible for the remaining sequence starting with value 1. The value is switched to 2 iff a block contains neither an $\mathtt{R}$- nor a $\mathtt{XR}$-formula. In such a situation, the automaton has to verify that the sequence does not break down. Therefore, the final states of the NcoBA is defined as stated above.

The size of the automaton $\mathcal{C}_\vartheta^{\mathtt{A},\mathrm{CTL}^+}$ is exponential in $|\vartheta|$. Hence, the complement of its Miyano-Hayashi determinisation is of double-exponential size —c.f. Theorem 22— and can be used in Subsection 5.6 instead of the general DPA $\mathcal{A}_\vartheta^{\mathtt{A}}$. Thus the time complexity of the whole decision procedure is double-exponential.

The advantage of this approach tailored to CTL^+ is the Miyano-Hayashi determinisation. Their construction is simple to implement because it bases on an elaborated subset-construction only compared to known determinisation procedures for general Büchi automata [Saf88, Pit06].

Because the small-formula strategy in Subsection 5.7 is indepenent of the fragement, Corollary 36 also holds for CTL^+. The lower bound for the size is also doubly exponential [Lan08].

6.2. **The Fragment CTL.** The satisfiability problem for CTL is EXPTIME-complete. Again, the question arises whether the lower expressivity compared to CTL^* leads to a simpler decision procedure.

As CTL is a fragment of CTL^* we could apply the introduced satisfiability game. However, this would lead to games of doubly exponential size, resulting in an unoptimal decision procedure.

Hence, we define a new set of configurations and games rules that handle CTL-formulas in an optimal way. Due to the fact that subformulas of fixpoints in CTL are always state formulas, there is no need to keep the immediate subformulas in the respective block after unfolding. By placing them at the top-level of the configurations, we can do without the concept of blocks, since every block contains exactly one subformula. Hence, these blocks can be understood as CTL-formulas.

Here, a *configuration (for ϑ)* is a non-empty set of state formulas of the set $\{\varphi, \mathtt{EX}\varphi, \mathtt{AX}\varphi \mid \varphi \in \mathit{Sub}(\vartheta)\}$. The additional formulas $\mathtt{EX}\varphi$ and $\mathtt{AX}\varphi$ will be generated when unfolding fixpoints. In return, the Fischer-Ladner closure is replaced with the set of subformulas. The definition of consistency etc. is exactly the same as before.

Again, we write $\mathit{Conf}(\vartheta)$ for the set of all consistent configurations for ϑ. Note that this is a finite set of at most exponential size in $|\vartheta|$.

Definition 40. The satisfiability game for a CTL-formula ϑ is a directed graph $\mathcal{G}_\vartheta = (\mathit{Conf}(\vartheta), V_0, E, v_0, L)$ whose nodes are all possible configurations and whose edge relation is given by the game rules in Figure 5. It is understood that the formulas which are stated

$$(\wedge)\ \frac{\varphi_1, \varphi_2, \Phi}{\varphi_1 \wedge \varphi_2, \Phi} \qquad (\vee)\ \frac{\varphi_1, \Phi \ \mid\ \varphi_2, \Phi}{\varphi_1 \vee \varphi_2, \Phi}$$

$$(\mathtt{EU})\ \frac{\varphi_2, \Phi \ \mid\ \varphi_1, \mathtt{EXE}(\varphi_1 \mathtt{U} \varphi_2), \Phi}{\mathtt{E}(\varphi_1 \mathtt{U} \varphi_2), \Phi} \qquad (\mathtt{AU})\ \frac{\varphi_2, \Phi \ \mid\ \varphi_1, \mathtt{AXA}(\varphi_1 \mathtt{U} \varphi_2), \Phi}{\mathtt{A}(\varphi_1 \mathtt{U} \varphi_2), \Phi}$$

$$(\mathtt{ER})\ \frac{\varphi_1, \varphi_2, \Phi \ \mid\ \varphi_2, \mathtt{EXE}(\varphi_1 \mathtt{R} \varphi_2), \Phi}{\mathtt{E}(\varphi_1 \mathtt{R} \varphi_2), \Phi} \qquad (\mathtt{AU})\ \frac{\varphi_1, \varphi_2, \Phi \ \mid\ \varphi_2, \mathtt{AXA}(\varphi_1 \mathtt{R} \varphi_2), \Phi}{\mathtt{A}(\varphi_1 \mathtt{R} \varphi_2), \Phi}$$

$$(\mathtt{X_0})\ \frac{\varphi_1, \ldots, \varphi_n}{\mathtt{AX}\varphi_1, \ldots, \mathtt{AX}\varphi_n, \Lambda} \qquad (\mathtt{X_1})\ \frac{\varphi'_1, \varphi_1, \ldots, \varphi_n \ \mid\ \cdots \ \mid\ \varphi'_m, \varphi_1, \ldots, \varphi_n}{\mathtt{EX}\varphi'_1, \ldots, \mathtt{EX}\varphi'_m, \mathtt{AX}\varphi_1, \ldots, \mathtt{AX}\varphi_n, \Lambda}$$

Figure 5: The game rules for CTL.

explicitly under the line do not occur in the sets Λ or Φ. The symbol ℓ stands for an arbitrary literal. The initial configuration is $v_0 = \vartheta$. The winning condition L will be described next.

Again, we need to track the infinite behaviour of eventualities. However, the situation is much easier here. First, we can do without the concept of blocks, implying that we can do without the concept of traces as well. Second, there is no structural difference in tracking bad threads contained in E- or A-blocks, any infinite trace contains exactly one thread, i.e. existential quantification and universal quantification over threads in traces are interchangeable.

The definition of principal formulas and plays is the same as before, and we again have the definition of connectedness and write $(\mathcal{C}, \varphi) \rightsquigarrow (\mathcal{C}', \varphi')$ to indicate that $\varphi \in \mathcal{C}$ if connected to the subsequent formula $\varphi' \in \mathcal{C}'$. There are still infinitely many applications of rules $(\mathtt{X_0})$ or $(\mathtt{X_1})$ in a play.

Definition 41. Let $\mathcal{C}_0, \mathcal{C}_1, \ldots$ be an infinite play. A *thread* t within $\mathcal{C}_0, \mathcal{C}_1, \ldots$ again is an infinite sequence of formulas $\varphi_0, \varphi_1, \ldots$ s.t. for all $i \in \mathbb{N}$: $(\mathcal{C}_i, \varphi_i) \rightsquigarrow (\mathcal{C}_{i+1}, \varphi_{i+1})$.

Again, such a thread t is called a U-thread, resp. an R-thread if there is a formula $\varphi \mathtt{U} \psi \in \mathit{Sub}(\vartheta)$, resp. $\varphi \mathtt{R} \psi \in \mathit{Sub}(\vartheta)$ s.t. $\psi_j = \varphi \mathtt{U} \psi$, resp. $\psi_j = \varphi \mathtt{R} \psi$ for infinitely many j.

Again, every play contains a thread and every thread is either an U-thread or a R-thread.

Definition 42. An infinite play $\pi = C_0, C_1, \ldots$ belongs to the winning condition L of $\mathcal{G}_\vartheta = (\mathit{Conf}(\vartheta), V_0, E, v_0, L)$ if π does not contain a U-thread.

The following can be shown in similar way as Theorem 17:

Theorem 43. For all $\vartheta \in$ CTL: ϑ is satisfiable iff player 0 has a winning strategy for the satisfiability game $\mathcal{G}_\vartheta$.

As decision procedure, we again propose to apply a reduction to parity games, similar to the one of Subsection 5.6. The parity game is constructed the same way by using the reduced configuration set of this section. Additionally, we can construct a much simpler DPA for checking the winning conditions.

Due to the fact that there are no traces anymore resp. every trace now contains a thread-singleton, we can either apply an automaton construction similar to the one of Subsection 5.4 or to the one of Subsection 5.5. We follow the latter approach here.

Remember that the automaton of Subsection 5.5 was composed of the disjoint union of k components $C_0, \ldots, C_{k-1}$ with $C_i = \{i\} \cup \{i\} \times 2^{FL(\vartheta)}$, where $\varphi_0 \mathtt{U} \psi_0, \ldots, \varphi_{k-1} \mathtt{U} \psi_{k-1}$ was an enumeration of all U-formulas in ϑ. We can simplify the automaton dramatically here

by considering the components $C_i = \{i\} \cup \{i\} \times \{\varphi, \mathtt{EX}\varphi, \mathtt{AX}\varphi \mid \varphi \in \mathit{Sub}(\vartheta)\}$ instead. The transition function is updated accordingly, following now single formulas instead of blocks. We can get a result similar to Theorem 31:

Theorem 44. For every CTL formula ϑ with k $\mathtt{U}$-subformulas there is a DBA $\mathcal{A}$ of size at most $k \cdot (1 + 3|\vartheta|)$ s.t. for all plays π: $\pi \in L(\mathcal{A})$ iff π does not contain a $\mathtt{U}$-thread.

By attaching this automaton to our parity game, we obtain an optimal decision procedure for CTL:

Corollary 45. Deciding satisfiability for some $\varphi \in$ CTL is in EXPTIME.

Proof. The number of states in the constructed parity game is bounded by

$$2^{\mathcal{O}(|\vartheta|)} \cdot |\vartheta| \cdot (1 + 3|\vartheta|) \;=\; 2^{\mathcal{O}(|\vartheta|)}.$$

Note that the out-degree of the parity game graph is at most $|\vartheta|$ because of rule $(\mathtt{X}_1)$ which is bounded by the number of $\mathtt{E}$-formulas in ϑ. The game's index is 2 which makes it, in fact, a Büchi game. It is well-known [CHP06] that Büchi games with n states and m edges can be solved in time $\mathcal{O}(n \cdot m)$ from which the claim follows immediately. □

The previous upper bound is optimal because the satisfiability problem for CTL-fragment PDL is EXPTIME-hard [FL79]. Since each block in the configurations is mainly a subformula of ϑ, the branching-width is bounded by $|\vartheta|$. This bound is independent of the strategy as compared with Corollary 36.

7. Comparison with Existing Methods

7.1. **CTL*.** We compare the game-based approach with existing decision procedures for CTL*, namely Emerson/Jutla's tree automata [EJ00], Kupferman/Vardi's automata reduction [KV05], Reynolds' proof system [Rey01], and Reynolds' tableaux [Rey09] with respect to several aspects like computational optimality, availability of an implementation etc., c.f. Table 1.

Emerson/Jutla's procedure transforms a CTL*-formula φ 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 in Subsection 5.1. The game-based approach presented here does not use tree-automata as such, but player-0-strategies resemble runs of a tree automaton. The crucial difference is the separation between the use of machinery for the characterisation of satisfiability in CTL* and the use of automata only in order to make the abstract winning conditions effectively decidable. 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 rules. Furthermore, this separation enables the branching-width of models of φ to be flexible; it is given by the number of successors of the rule $(\mathtt{X}_1)$. In a tree automaton setting it is a priori fixed to a number which is linear in the size of the input formula. While this does not increase the asymptotic worst-case complexity, it may have an effect on the efficiency in practice. Not surprisingly, we do not know of any attempt to implement the tree-automata approach.

Aspect \ Method	Emerson & Jutla [EJ00]	Reynolds [Rey01]	Kupferman & Vardi (& Wolper) [KVW00, KV05]	here
Concept	automata	tableaux	automata-reduction	games
Worst-case complexity	2EXPTIME	2EXPTIME	2EXPTIME	2EXPTIME
Implementation available	no	yes	no	yes[2]
Model construction	yes	yes	no	yes
Out-degree	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
Requires small model property	no	yes	no	no
Derives small model property	$2^{2^{\mathcal{O}(n)}}$	—	$2^{2^{\mathcal{O}(n)}}$	$2^{2^{\mathcal{O}(n)}}$
Needs Büchi determinisation	yes	no	no	yes

Table 1: Comparison of the main decision methods for satisfiability in CTL*.

Kupferman/Vardi's approach is not just a particular decision procedure for CTL*. Instead, it is a general approach to solving the emptiness problem for alternating parity tree automata. While this can generally be done using determinisation of Büchi automata as in Emerson/Jutla's approach, Kupferman/Vardi have found a way to avoid Büchi determinisation by using universal co-Büchi automata instead. These are translated into alternating weak tree automata and, finally, into nondeterministic Büchi tree automata. Emptiness of the latter is relatively easy to check. In the case of CTL*, a formula φ can be translated into a hesitant alternating automaton of size $\mathcal{O}(|\varphi| \cdot 2^{|\varphi|})$ [KVW00] whose emptiness can be checked in time that is doubly exponential in $|\varphi|$.

The price to pay, though, is the use of a reduction that is only satisfiability-preserving. Thus, their approach reduces the satisfiability problem for branching-time temporal logics that can be translated into alternating parity tree automata to the emptiness problem for tree automata which accept some tree iff the input formula is satisfiable. The translation does not preserve models, though. There is a way of turning a tree model for the nondeterministic Büchi automaton back into a tree model for the branching-time temporal logic formula because the alphabet that the universal co-Büchi automaton uses is just a projection of the hesitant alternating tree automaton's alphabet. Still, this procedure does not seem to keep a close connection between the subformulas of the input formulas and the structure of the resulting tree automaton which is being checked for emptiness.

Reynolds' proof system [Rey01] 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 lacks the subformula property and it is therefore not even clear how a decision procedure, i.e. proof search could be done. In comparison, the game-based calculus has the subformula property—formulas in blocks of successor configurations are subformulas of those in the blocks of the preceding one—and comes with

[2] https://github.com/oliverfriedmann/mlsolver

Aspect \ Method	Emerson & Halpern [EH85]	Vardi & Wolper [VW86]	Abate et al. [AGW07]	here
Concept	filtration	automata	tableaux	games
Worst-case complexity	EXPTIME	EXPTIME	2EXPTIME	EXPTIME
Implementation available	no	no	yes	yes
Model construction	yes	yes	yes	yes
Requires small model property	yes	no	no	no
Derives small model property	—	$2^{\mathcal{O}(n)}$	$2^{2^{\mathcal{O}(n)}}$	$2^{\mathcal{O}(n)}$

Table 2: Comparison of the main decision methods for satisfiability of CTL-formulas.

an implementable decision procedure. The only price to pay for this is the characterisation of satisfiability through infinite objects instead.

Reynold's tableau system [Rey11] shares some similarities with the games presented here. 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 the games come with relatively straight-forward game 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.*" The game-based method comes with a non-trivial repetition checker: it is given by the annotated automata.

7.2. **The Fragments CTL$^+$ and CTL.** To the best of our knowledge, there are no decision procedures that are especially tailored towards CTL$^+$. Thus, the restriction of the satisfiability games to CTL$^+$ as presented in Section 6.1 is the first decision procedure for this logic which does not also decide the whole of CTL*.

The situation for CTL is entirely different. The first decision procedure for CTL was given by Emerson and Halpern [EH85] using filtration. It starts with a graph of Hintikka sets and successively removes edges from this graph in order to exclude unfulfilled eventualities. This is similar to the game-based approach in that the game rules for Boolean connectives mimic the rules for being a Hintikka set. On the other hand, the machinery for excluding unfulfilled eventualities is an entirely different one.

There is a purely automata-theoretic decision procedure for CTL [VW86]: as such, it constructs a tree automaton which recognises all tree-unfoldings of models of the input formula. In order to obtain an asymptotically optimal decision procedure for CTL, Vardi/Wolper use a new type of acceptance condition resulting in *eventuality automata* whose emptiness problem can be decided in polynomial time. An exponential translation

from CTL into such automata then yields a decision procedure for CTL. There are certain similarities to the game-based approach presented here: the design of the simpler type of acceptance condition is reminiscent of the manual creation of deterministic automata that check the winning conditions.

There is a tableau-based decision procedure for CTL [AGW07]. As with Reynold's tableaux for CTL*, the main difference to the game-based (and also automata-theoretic) approach is the fact that the tableau calculi do not separate the decision procedure into a syntactical characterisation (e.g. winning strategy) and an algorithm deciding existence of such objects. This leads to correctness proofs which are even more complicated than the ones for the CTL* games presented here. Also, this method does not yield a common framework for dealing with unfulfilled eventualities which is given by the different types of (deterministic) automata which are being used here in order to characterise the winning conditions.

The work that is most closely related to the one presented here consists of the focus game approach to CTL [LS01]. These are also satisfiability games, and the rules there extend the rules here with a focus on a particular subformula which is under player 1's control. The focus game approach does not explicitly give an algorithm for deciding satisfiability. A close analysis shows that the focus can be seen as an annotation with a nondeterministic co-Büchi automaton to the game configurations, and a decision procedure could be obtained by determinising this automaton. In this respect, the games presented here improve over the focus games by showing how small deterministic Büchi automata suffice for this task.

Table 2 tabulates the comparison of the CTL satisfiability games with these other approaches.

8. Further Work

The results of the previous section show that the game/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

[AGW07] P. Abate, R. Goré, and F. Widmann. One-pass tableaux for computation tree logic. In *Proc. 14th Int. Conf. on Logic for Programming, Artificial Intelligence and Reasoning, LPAR'07*, volume 4790 of *LNCS*, pages 32–46. Springer, 2007.

[AI01] M. Adler and N. Immerman. An $n!$ lower bound on formula size. In *Proc. 16th Symp. on Logic in Computer Science, LICS'01*, pages 197–208, Boston, MA, USA, 2001. IEEE.

[BDF99] A. Bolotov, C. Dixon, and M. Fisher. Clausal resolution for CTL*. In *Proc. 24th Int. Symp. on Mathematical Foundations of Computer Science, MFCS'99*, volume 1672 of *LNCS*, pages 137–148. Springer, 1999.

[BF99] A. Bolotov and M. Fisher. A clausal resolution method for CTL branching-time temporal logic. *J. Exp. Theor. Artif. Intell*, 11(1):77–93, 1999.

[BVW94] O. Bernholtz, M.Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. In D. L. Dill, editor, *Proc. 6th Conf. on Computer Aided Verification, CAV'94*, volume 818 of *LNCS*, pages 142–155, Stanford, 1994. Springer.

[CE81] E. M. Clarke and E. A. Emerson. Synthesis of synchronization skeletons for branching time temporal logic. In *Logics of Programs: Workshop*, volume 131 of *LNCS*, pages 52–71, Yorktown Heights, New York, 1981. Springer.

[CHP06] Krishnendu Chatterjee, Tom Henzinger, and Nir Piterman. Algorithms for buchi games. In *GDV 06*, August 2006.

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

[EH86] 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.

[EJ91] 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.

[EJ00] 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.

[Eme90] 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.

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

[Fis91] M. Fisher. A resolution method for temporal logic. In *Proc. 12th Int. Joint Conference on Artificial Intelligence, IJCAI'91*, pages 99–104. Morgan Kaufmann, 1991.

[FL79] M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211, 1979.

[FL09] 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.

[FL10] O. Friedmann and M. Lange. A solver for modal fixpoint logics. In *Proc. 6th Workshop on Methods for Modalities, M4M-6*, volume 262 of *Elect. Notes in Theor. Comp. Sc.*, pages 99–111, 2010.

[FLL10] O. Friedmann, M. Latte, and M. Lange. A Decision Procedure for CTL* Based on Tableaux and Automata. In Jürgen Giesl and Reiner Hähnle, editors, *Proc. of the 5th Int. Joint Conference on Automated Reasoning*, volume 6173 of *Lecture Notes in Computer Science*, pages 331–345. Springer, 2010.

[GP08] 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.

[GTW02] E. Grädel, W. Thomas, and Th. Wilke, editors. *Automata, Logics, and Infinite Games*, volume 2500 of *LNCS*. Springer, 2002.

[JL03] J. Johannsen and M. Lange. CTL$^+$ is complete for double exponential time. In *Proc. 30th Int. Coll. on Automata, Logics and Programming, ICALP'03*, volume 2719 of *LNCS*, pages 767 – 775. Springer, 2003.

[KV05] O. Kupferman and M. Y. Vardi. Safraless decision procedures. In *Proc. 46th Ann. IEEE Symp. on Foundations of Computer Science, FOCS'05*, pages 531–542. IEEE, 2005.

[KVW00] O. Kupferman, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, 2000.

[KW08] 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.

[Lan08] M. Lange. A purely model-theoretic proof of the exponential succinctness gap between CTL$^+$ and CTL. *Information Processing Letters*, 108:308–312, 2008.

[LS01] M. Lange and C. Stirling. Focus games for satisfiability and completeness of temporal logic. In *Proc. 16th Symp. on Logic in Computer Science, LICS'01*, Boston, MA, USA, 2001. IEEE.

[LSS+05] 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.

[McN66] R. McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9(5):521–530, 1966.

[MH84] S. Miyano and T. Hayashi. Alternating finite automata on omega-words. *TCS*, 32(3):321–330, 1984.

[MS87] D. E. Muller and P. E. Schupp. Alternating automata on infinite trees. *TCS*, 54(2-3):267–276, 1987.

[Pit06] 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.

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

[PR88] 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.

[Rey01] M. Reynolds. An axiomatization of full computation tree logic. *Journal of Symbolic Logic*, 66(3):1011–1057, 2001.

[Rey09] 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.

[Rey11] M. Reynolds. A tableau-based decision procedure for CTL*. *Journal of Formal Aspects of Computing*, pages 1–41, 2011.

[Saf88] S. Safra. On the complexity of ω-automata. In *Proc. 29th Symp. on Foundations of Computer Science, FOCS'88*, pages 319–327. IEEE, 1988.

[Sch07] 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.

[Sch09] 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.

[VS85] 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.

[VW86] M. Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logic of programs. *Journal of Computer and System Sciences*, 32:183–221, 1986.

[Wil99] T. Wilke. CTL+ is exponentially more succinct than CTL. In *Proc. 19th Conf. on Foundations of Software Technology and Theoretical Computer Science, FSTTCS'99*, volume 1738 of *LNCS*, pages 110–121. Springer, 1999.

[Zer04] E. Zermelo. Beweis, daß jede Menge wohlgeordnet werden kann. *Mathematische Annalen*, 59:514–516, 1904.

[ZHD10] L. Zhang, U. Hustadt, and C. Dixon. CTL-RP: A computation tree logic resolution prover. *AI Communications*, 23(2-3):111–136, 2010.

[Zie98] W. Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *TCS*, 200(1–2):135–183, 1998.