One-Pass Context-Based Tableaux Systems for CTL and ECTL

Alex Abuin
Ikerlan Technology Research Centre, Basque Research and Technology Alliance (BRTA), Arrasate-Mondragón Gipuzkoa, Spain
aabuin@ikerlan.es

Alexander Bolotov
University of Westminster, London, UK
`https://www.westminster.ac.uk/about-us/our-people/directory/bolotov-alexander`
A.Bolotov@westminster.ac.uk

Montserrat Hermo
University of the Basque Country, San Sebastián, Spain
montserrat.hermo@ehu.es

Paqui Lucio
University of the Basque Country, San Sebastián, Spain
`http://www.sc.ehu.es/paqui`
paqui.lucio@ehu.eus

Abstract

When building tableau for temporal logic formulae, applying a two-pass construction, we first check the validity of the given tableaux input by creating a tableau graph, and then, in the second "pass", we check if all the eventualities are satisfied. In one-pass tableaux checking the validity of the input does not require these auxiliary constructions. This paper continues the development of one-pass tableau method for temporal logics introducing tree-style one-pass tableau systems for Computation Tree Logic (CTL) and shows how this can be extended to capture Extended CTL (ECTL). A distinctive feature here is the utilisation, for the core tableau construction, of the concept of a context of an eventuality which forces its earliest fulfilment. Relevant algorithms for obtaining a systematic tableau for these branching-time logics are also defined. We prove the soundness and completeness of the method. With these developments of a tree-shaped one-pass tableau for CTL and ECTL, we have formalisms which are well suited for the automation and are amenable for the implementation, and for the formulation of dual sequent calculi. This brings us one step closer to the application of one-pass context-based tableaux in certified model checking for a variety of CTL-type branching-time logics.

2012 ACM Subject Classification Theory of computation → Modal and temporal logics

Keywords and phrases Temporal logic, fairness, expressiveness, branching-time

Digital Object Identifier 10.4230/LIPIcs.TIME.2020.14

Funding *Alexander Bolotov*: This author has been supported by the European Union (FEDER funds) under grant TIN2017-86727-C2-2-R, and by the University of the Basque Country under Project LoRea GIU18-182.
Montserrat Hermo: This author has been supported by the European Union (FEDER funds) under grant TIN2017-86727-C2-2-R, and by the University of the Basque Country under Project LoRea GIU18-182.
Paqui Lucio: This author has been supported by the European Union (FEDER funds) under grant TIN2017-86727-C2-2-R, and by the University of the Basque Country under Project LoRea GIU18-182.

© Alex Abuin, Alexander Bolotov, Montserrat Hermo, and Paqui Lucio;
licensed under Creative Commons License CC-BY

1 Introduction

In this paper we continue our investigation of tableaux-based deductive techniques for temporal logic having in mind their potential application in model checking, more specifically, in certified model checking [16], which aims to generate proofs as certificates of the properties that are verified, as well as counterexamples for those properties that are invalidated. There are two known ways to build tableau constructions for temporal logic formulae (for the survey of tableau method for temporal logic we refer an interested reader to [13]). Two-pass constructions check the validity of the given tableaux input in two passes - in the first pass a tableau graph is obtained and the second "pass" checks the satisfiability of all eventualities. In one-pass tableaux checking the validity of the input does not require these auxiliary constructions. This paper continues the development of one-pass tableau method for temporal logics [11, 4], this time for Computation Tree Logic (CTL) and Extended Computation Tree Logic (ECTL) introduced, respectively, in [6] and [10]. The core tableau construction is based on the concept of *a context of an eventuality*, which is a set of formulae that "accompanies" the eventuality in the label of the node of a tableaux graph. Our specific tableau rules that involve context force the earliest fulfilment of eventualities. In previous works such a context-based one-pass tableaux approach has been developed for propositional linear-time temporal logic, PLTL [11], and for the branching-time logic ECTL$^{\#}$ [4], which introduces a new class of fairness constraints utilising the "until" temporal operator. It has also been shown how, in the linear-time case, the method, being mingled with a SAT solver, can be invoked as part of the certified model checking for PLTL [2]. Aiming at similar developments for branching-time cases, in particular for CTL, we make two observations.

Firstly, the satisfiability of a property φ in PLTL can be reduced to checking if a complete transition system satisfies $\neg\varphi$ (since any counter-model of $\neg\varphi$ is a model of φ) and both the satisfiability and model checking are PSPACE-complete [18]. However, the CTL satisfiability problem cannot be reduced to the CTL model checking. In particular, a model checking algorithm for CTL properties (for example [5] implemented in NuSMV) cannot be adapted for testing CTL satisfiability: the model checking problem for CTL is known to be P-complete [7], while the satisfiability problem for CTL is EXPTIME-complete [9]. However, any decision procedure of CTL satisfiability can be used to perform model checking tasks.

Secondly, note that in our previous work one-pass tableaux method was developed for a richer logic - ECTL$^{\#}$ [4]. However, the application of such model checking procedure for CTL simply based on the existing one-pass tableaux for ECTL$^{\#}$ would become too "non-intuitive" due to the complexity of its rules that are needed for this richer logic. We also note that the distinguished (and unavoidable) feature of one-pass technique for ECTL$^{\#}$ is the utilisation of two types of context, unlike in the case of PLTL. Here the so-called "outer" (similar to PLTL) context is a collection of state formulae, and is complemented by so called "inner" context, a collection of path formulae. Subsequently, the development of a simpler one-pass method for CTL is an important task. In our tableau method for CTL, similarly to PLTL, we only need the "outer" context, yet, similar to ECTL$^{\#}$ the generated tableaux are AND-OR trees. Our results provide an intuitive tableau method that serves as a decision procedure of CTL satisfiability and can also be used in certified model checking of CTL properties hence the method presented in the paper would enable a subsequent study and implementation of a certified model checker. With the development of tree-shaped one-pass tableaux for CTL and ECTL, this paper has proved the effectiveness of the approach which now covers both linear-time and a range of branching-time logics. Moreover, the results of this paper give us formalisms which are well suited for the automation and are amenable for the implementation, and for the formulation of a dual sequent calculi - all these bring us one step closer to the application of these developments in certified model checking for a variety of branching-time logics. Additionally, aiming at the extension of the certified model checking to the branching-time framework, a proof system, e.g. sequent calculus, is required to check the proof certificates in the branching-time setting.

Our extensive search for tableau methods for CTL has not shown a great variety of systems. For example, [3] presents a two-pass tableau, where in the first pass the tableau rules are applied creating a cyclic graph. In the second pass, "bad loops" are pruned (where a "bad loop" is a loop containing some eventuality that is not fulfilled along it). In [1, 12] the authors introduce a single-pass tableaux decision procedure for CTL. It is based on Schwendimann's one-pass procedure for PLTL [17]. This tableau method uses an additional mechanism for collecting information on the set of formulae in the nodes, and passing it, to subsequent nodes along branches. The information on previously generated nodes helps detecting "bad loops" without constructing the whole graph. Finally, we note that we have not found an explicit formulation of a tableaux (one or two pass) method for ECTL.

To ensure that the presentation of quite technical details in the paper is clear and self-contained, we supply all major technical details in the text. This determines the following structure of the paper. In §2 we give CTL and ECTL syntax and semantics as sublogics of $\mathsf{CTL}^\star$. The formulation of the tableau method is presented in §3, where we first give some preliminaries and then overview the tableau construction as an AND-OR tree and provide examples. A *systematic* tableau construction is introduced in §4. In §5 we show further extension of the method for ECTL. In §6 we draw the conclusions and prospects of future work that the presented results open. Finally, in Appendix A we briefly recall the cyclic models characterization of satisfiability in branching temporal logics. The soundness and completeness of our tableau methods are proved in Appendix B. Finally, in Appendix C we depict the complete tableau for the running example in the paper.

2 Syntax and Semantics of CTL and ECTL

The language of branching-time logic extends the language of classical propositional logic by future time temporal operators $\circ$ - "at the next moment of time", $\Diamond$ - "eventually", $\Box$ - "always" and $\mathcal{U}$ - "until", together with paths quantifiers A - "for all paths" quantifier, and E - "there exists a path" quantifier.

The hierarchy of CTL-type family of Branching-time logics (BTL) is defined by releasing restrictions on the concatenations of temporal operators and paths quantifiers which define classes of admissible state formulae distinguished for these logics. As in CTL every temporal operator must be preceded by a path quantifier, this logic cannot express fairness which requires at least the concatenation of $\Box$ and $\Diamond$. These are tackled by ECTL [9] which enables simple fairness constraints but not their Boolean combinations. ECTL^+ [10] further extends the expressiveness of ECTL allowing Boolean combinations of temporal operators and ECTL fairness constraints (but not permitting their nesting). The logic $\mathsf{ECTL}^\#$ [4] extends ECTL^+ by allowing the combinations $\Box(A\mathcal{U}B)$ or $A\mathcal{U}\Box B$, referred to as modalities $\Box\mathcal{U}$ and $\mathcal{U}\Box$. The logic $\mathsf{CTL}^\star$, often considered as the "full branching-time logic" overcomes all these restrictions on syntax allowing any arbitrary combinations of temporal operators and path quantifiers. For the sake of generality, as all logics we are interested in are subsumed by $\mathsf{CTL}^\star$, we first recall $\mathsf{CTL}^\star$ syntax and then, by restricting it, derive the syntax for each of $\mathsf{ECTL}^\#$, ECTL^+, ECTL and CTL.

▶ **Definition 1** (Syntax of $\mathsf{CTL}^\star$). *Given Prop is a fixed set of propositions, and $p \in$ Prop, we define sets of state (σ) and path (π) $\mathsf{CTL}^\star$ formulae over Prop as follows:* $\sigma ::= \mathbf{T} \mid p \mid \sigma_1 \wedge \sigma_2 \mid \neg\sigma \mid E\pi$ *and* $\pi_{\mathsf{CTL}^\star} ::= \sigma \mid \pi_1 \wedge \pi_2 \mid \neg\pi \mid \circ\pi \mid \pi\mathcal{U}\pi$.

Observe that in Definition 1 for the set of path formulae we deliberately used an index $_{\mathsf{CTL}^\star}$ and did not use any index for the set of state formulae: the syntax of $\mathsf{CTL}^\star$ sublogics we will define later, will be distinguished exactly by specific to these logics path formulae constructions, while the set of state formulae is preserved from the definition of $\mathsf{CTL}^\star$ syntax. Other usual Boolean operators can be derived from those introduced in the standard way while the "release" ($\mathcal{R}$), $\Diamond$ and $\Box$ operators can be defined as follows: $\varphi_1 \mathcal{R} \varphi_2 \equiv \neg(\neg\varphi_1 \mathcal{U} \neg\varphi_2)$, $\Diamond\varphi \equiv \mathbf{T}\mathcal{U}\varphi$, and $\Box\varphi \equiv \neg\Diamond\neg\varphi$.

We consider a Kripe-style semantics of $\mathsf{CTL}^\star$: a Kripke structure, $\mathcal{K}$, is a triple (S, R, L) where $S \neq \emptyset$ is a set of *states*, $R \subseteq S \times S$ is a total binary relation, called *the transition relation*, and $L : S \to 2^{\mathsf{Prop}}$ is a *labelling function.* Our Kripke structures are labelled directed graphs that correspond to Emerson's R-generable structures, i.e. the transition relation R is suffix, fusion and limit closed [8]. A *path* x through a Kripke structure $\mathcal{K}$ is an infinite sequence of states $s_i, s_{i+1}, s_{i+2} \ldots$ such that $(s_j, s_{j+1}) \in R$ for any $j \geq i$. A *fullpath* x through a Kripke structure $\mathcal{K}$ is an infinite sequence of states $s_0, s_1, s_2 \ldots$, where s_0 is the root. Given a path $x = s_i, s_{i+1}, \ldots$ and a state $s_k \in x$ such that $k > i$, we denote its finite prefix $x^{\leq k} = s_i, s_{i+1} \ldots, s_k$ and its infinite suffix $x^{\geq k} = s_k, s_{k+1}, \ldots$. The notation $\mathcal{K} \upharpoonright x(i)$ denotes a Kripke structure with the set of states of $\mathcal{K}$ restricted to those that are R-reachable from $x(i)$ and $\mathsf{fullpaths}(\mathcal{K})$ is the set of all fullpaths in $\mathcal{K}$. Given the structure $\mathcal{K} = (S, R, L)$, the relation $\models$ evaluates path formulae in a given path x and state formulae at the state index i of x and is defined on atoms by $\mathcal{K}, x, i \models p$ iff $p \in L(x(i))$. Omitting standard definitions for Booleans, we present the relation $\models$ for temporal connectives and path quantifier E:

$\mathcal{K}, x, i \models \mathsf{E}\pi$ iff there exists a path $y \in \mathsf{fullpaths}(\mathcal{K} \upharpoonright x(i))$ such that $\mathcal{K}, y \models \pi$.
$\mathcal{K}, x, i \models \circ\pi$ iff $\mathcal{K}, x, i+1 \models \pi$.
$\mathcal{K}, x, i \models \pi_1 \mathcal{U} \pi_2$ iff there exists $k \geq i$ with $\mathcal{K}, x^{\geq k} \models \pi_2$ and $\mathcal{K}, x^{\geq j} \models \pi_1$ for all $0 \leq j \leq k-1$.

For any set Σ of state formulae, $\mathcal{K}, x, i \models \Sigma$ iff $\mathcal{K}, x, i \models \sigma$, for all $\sigma \in \Sigma$. Moreover, if for any fullpath $x \in \mathsf{fullpaths}(\mathcal{K})$, we have $\mathcal{K}, x, 0 \models \Sigma$, then we simply write $\mathcal{K} \models \Sigma$. For a state formula φ, the set of its models, $\mathsf{Mod}(\varphi)$, is formed by all triples $(\mathcal{K}, x, i)$ such that $\mathcal{K}, x, i \models \varphi$. Then φ is *satisfiable* ($\mathsf{Sat}(\varphi)$) if $\mathsf{Mod}(\varphi) \neq \emptyset$, otherwise φ is *unsatisfiable* ($\mathsf{UnSat}(\varphi)$). For state formulae φ and φ', if $\mathsf{Mod}(\varphi) = \mathsf{Mod}(\varphi')$ then φ and φ' are logically equivalent denoted as $\varphi \equiv \varphi'$. Satisfiability and logical equivalence are generalised to sets of state formulae Σ, in the natural way (formally by substituting φ with Σ in the relevant definitions and stating that Σ is satisfied when all its formulae are satisfied).

For each of BTL logics $\mathsf{ECTL}^{\#}$, ECTL^{+}, ECTL and CTL its syntax is defined over a fixed set of propositions Prop, such that the definition of state formulae is the same as for $\mathsf{CTL}^\star$ (Def. 1), and the eventuality $\Diamond\varphi$ is the abbreviation for $\mathbf{T}\mathcal{U}\varphi$. The specific for these logics restrictions on the $\mathsf{CTL}^\star$ grammar in Definition 1 generate the corresponding sets for path formulae, as in Definition 2. For technical convenience, here we define $\Box$ as the basic language operator.

▶ **Definition 2** (Paths formulae for $\mathsf{ECTL}^{\#}$,ECTL^{+}, ECTL and CTL).

$$\begin{array}{lll}
\pi_{\mathsf{ECTL}^{\#}} & ::= & \sigma \mid \pi_1 \wedge \pi_2 \mid \neg\pi \mid \circ\sigma \mid \sigma\mathcal{U}(\sigma \wedge \Diamond\sigma) \mid \Box(\sigma \vee \Box\sigma) \mid \sigma\mathcal{U}(\Box\sigma) \mid \Box(\sigma\mathcal{U}\sigma) \\
\pi_{\mathsf{ECTL}^{+}} & ::= & \sigma \mid \pi_1 \wedge \pi_2 \mid \neg\pi \mid \circ\sigma \mid \sigma\mathcal{U}\sigma \mid \Box\sigma \mid \Box\Diamond\sigma \mid \Diamond\Box\sigma. \\
\pi_{ECTL} & ::= & \sigma \mid \neg\pi \mid \circ\sigma \mid \sigma\mathcal{U}\sigma \mid \Box\sigma \mid \Box\Diamond\sigma \mid \Diamond\Box\sigma. \\
\pi_{CTL} & ::= & \sigma \mid \neg\pi \mid \circ\sigma \mid \sigma\mathcal{U}\sigma \mid \Box\sigma.
\end{array}$$

▶ **Definition 3** (Literals). *Let* Prop *be a fixed set of CTL (ECTL) propositions, and let $\rho \in$ Prop. Then the set of CTL(ECTL) literals is defined as Lit ::=* $\mathbf{F} \mid \mathbf{T} \mid \rho \mid \neg\rho$.

It is well known that any given branching-time formula φ can be converted to a formula $\mathsf{NNF}(\varphi)$ - the Negation Normal Form of φ obtained by pushing negations inwards until they only apply to literals. The conversion is based on well-known equivalences which ensure that φ and $\mathsf{NNF}(\varphi)$ have exactly the same models, i.e. are logically equivalent. Consequently, we assume that inputs for the tableaux procedure in CTL and ECTL are given in NNF. For simplicity, we will write $\sim\varphi$ instead of $\mathsf{NNF}(\neg\varphi)$. Also, for a finite set $\Phi = \{\varphi_1, \dots, \varphi_n\}$, we let $\sim\Phi = \mathsf{NNF}(\neg \bigwedge_{i=1}^{n} \varphi_i)$.

Further, it is important to note that the nesting of "pure path formulae", totally unrestricted in CTL*, is now restricted in its sublogics by relevant grammar cases for paths formulae. For example, a CTL* formula (1) is not an ECTL$^\#$ formula. Rewriting it as $\mathsf{A}(\mathbf{T}\mathcal{U}(\circ p \wedge \mathsf{E}\circ\neg p))$ we can see

$$\mathsf{A}\Diamond(\bigcirc p \wedge \mathsf{E}\bigcirc\neg p) \tag{1}$$

that $\circ p \wedge \mathsf{E}\circ\neg p$ is neither a state formula nor of the form $\Box\sigma$. Note that the validity of (1) which is indicative for CTL*, is directly linked to the limit closure property [8]. Similarly, a ECTL$^\#$ formula $\mathsf{A}((p\mathcal{U}\Box q) \wedge (s\mathcal{U}\Box\neg q))$ is not an ECTL$^+$ formula because $p\mathcal{U}\Box q$ and $s\mathcal{U}\Box\neg q$, hence their conjunction, are not admissible ECTL$^+$ formulae. Further, an ECTL$^+$ formula (2) does not belong to ECTL

$$\mathsf{E}(\Box\Diamond q \wedge \Diamond\Box\neg q) \tag{2}$$

as $\Box\Diamond q \wedge \Diamond\Box\neg q$ is not an admissible ECTL path formula. Finally, the fairness constraint (3) expressible in ECTL cannot be constructed in CTL syntax as every temporal operator

$$\mathsf{E}\Box\Diamond q \tag{3}$$

in a CTL formula must be preceded by a path quantifier. Note that it is important to distinguish the problem if a formula of a superlogic belongs to a sublogic and the problem if a formula of a superlogic can be expressed in a sublogic. For example, $\mathsf{E}(\Box\Diamond q \vee \Diamond\Box\neg q)$, similarly to formula (2) does not belong to ECTL but unlike (2), it is expressible in this logic, as $\mathsf{E}(\Box\Diamond q \vee \Diamond\Box\neg q) \equiv \mathsf{E}\Box\Diamond q \vee \mathsf{E}\Diamond\Box\neg q$ which is an ECTL formula if we define $\vee$ via $\wedge$.

Table 1 Classification of context-based tableaux systems for CTL-type logics and relevant difficult cases of concatenations of temporal operators and path quantifiers.

BTL Logics	$\mathsf{E}\Box\Diamond q$	$\mathsf{E}(\Box\Diamond q \wedge \Diamond\Box\neg q)$	$\mathsf{A}((p\mathcal{U}\Box q) \vee (s\mathcal{U}\Box\neg r))$	$\mathsf{A}\Diamond(\circ p \wedge \mathsf{E}\circ\neg p)$	One-pass Tableaux
$\mathcal{B}(\mathcal{U}, \circ)$ (CTL)	X	X	X	X	This paper
$\mathcal{B}(\mathcal{U}, \circ, \Box\Diamond)$ (ECTL)	$\surd$	X	X	X	This paper
$\mathcal{B}^+(\mathcal{U}, \circ, \Box\Diamond)$ (ECTL$^+$)	$\surd$	$\surd$	X	X	$\surd$
$\boldsymbol{\mathcal{B}^+(\mathcal{U}, \circ, \mathcal{U}\Box)}$ (ECTL$^\#$)	$\surd$	$\surd$	$\surd$	X	$\surd$
$\mathcal{B}^\star(\mathcal{U}, \circ)$ (CTL*)	$\surd$	$\surd$	$\surd$	$\surd$	X

Table 1 represents BTL logics classified by their expressiveness using "$\mathcal{B}$" for "Branching", followed by the set of only allowed modalities as parameters; $\mathcal{B}^+$ indicates admissible Boolean combinations of the modalities and $\mathcal{B}^\star$ reflects "no restrictions" in either concatenations of the modalities or Boolean combinations between them following the notation initially proposed in [8] and further tuned in [15]. The top row of the figure represents the indicative formulae (1)-(3) for the listed logics. The last column in Table 1 reflects the development of the context-based one-pass tableaux technique for CTL-type logics: the method has been developed for $\mathsf{ECTL}^{\#}$ ([4] where the motivation was to tackle complex cases of fairness). In this paper we introduce the technique for CTL and ECTL, while the case of ECTL^{+} can be tackled effectively by the technique developed for $\mathsf{ECTL}^{\#}$. Indeed, ECTL^{+} and $\mathsf{ECTL}^{\#}$ have similar cases of the Boolean combination of eventualities in the scope of A and E, namely disjunctions of the eventualities in the scope of the A quantifier and conjunctions of eventualities in the scope of the E quantifier, see [4] for details. Thus, Table 1 also reflects syntactical cases of concatenations of temporal operators and path quantifiers that are difficult for context-based one-pass tableaux. To tackle these cases, in addition to α- and β-rules, that are standard to the tableaux, novel β^{+}-rules which use the context to force the eventualities to be fulfilled as soon as possible, were introduced. As $\mathsf{ECTL}^{\#}$ is more expressive than ECTL^{+} in allowing new type of fairness constraints that use the $\mathcal{U}$ operator, the relevant rules introduced in [4] would cover all difficult concatenations of operators in ECTL^{+}. Hence, simply treating the case of one-pass context-based tableaux for ECTL^{+} as solved by the relevant development for a richer logic $\mathsf{ECTL}^{\#}$, in this paper we concentrate on bridging the gap in our roadmap in supplying BTL logics by this technique, by developing the method for CTL and ECTL. The ultimate target of this roadmap - the one-pass context-based tableaux for $\mathsf{CTL}^{\star}$ remains extremely difficult and an open problem.

3 Context-based One-pass Tableau Method for CTL

We precede the presentation of the method by the introduction of a number of important concepts. Firstly, we introduce a concept of *basic modality* which reflects the restrictions on forming the basic admissible combinations of temporal operators in the scope of a path quantifier. Recall that formulae of CTL and ECTL logics are written in NNF. Abbreviating by Q either of the path quantifiers A or E, we consider a basic modality of CTL or ECTL logic to be of the form QT, where T is a temporal operator. The structure QT is generated by the grammar rules for these logics in Def. 2. We can identify all basic modalities in a given formula φ by finding its most embedded modality(es), say M_1, then looking at the next basic modality in which M_1 is embedded, etc. For example, basic modalities for CTL are structures $\mathsf{Q}\circ, \mathsf{Q}\mathcal{U}$, and $\mathsf{Q}\Box$ while for ECTL these will be $\mathsf{Q}\circ, \mathsf{Q}\mathcal{U}$, $\mathsf{Q}\Box$, $\mathsf{Q}\Diamond\Box$ and $\mathsf{Q}\Box\Diamond$. If we analyse a CTL formula $\mathsf{E}\circ\mathsf{A}\circ p$ then the most embedded basic modality, M_1, would be $\mathsf{A}\circ p$, which is embedded as $\mathsf{E}\circ M_1$. These are generalised in Definition 4.

▶ **Definition 4** ($\mathsf{ECTL}^{\#}$,ECTL^{+}, ECTL and CTL Basic Modalities).

$$M_{ECTL} \quad ::= \quad c \mid Q\circ M \mid Q(M\mathcal{U}M) \mid Q\Box M \mid Q\Box\Diamond M \mid Q\Diamond\Box M.$$
$$M_{CTL} \quad ::= \quad c \mid Q\circ M \mid Q(M\mathcal{U}M) \mid Q\Box M.$$

where c stands for a purely classical formula (we can consider a purely classical formula as a zero-degree basic modality) and M stands for any basic modality of CTL in the definition of $\mathsf{M}_{\mathsf{CTL}}$ and of ECTL in the definition of $\mathsf{M}_{\mathsf{ECTL}}$. Note that we have "derived" cases of basic modalities for $\Diamond\mathsf{M}$ and $\mathsf{M}\,\mathcal{R}\,\mathsf{M}$. In what follows, every CTL modality $\mathsf{Q}\mathcal{U}$ or $\mathsf{Q}\Diamond$ is called *eventuality*.

CTL tableau rules are based on fixpoint characterisation of its basic modalities: (in the equations below ν and μ stand for "minimal fixpoint" and "maximal fixpoint" operators, respectively)

$$\begin{array}{ll} \mathsf{E}\Box\varphi = \nu\rho(\varphi \wedge \mathsf{E}\circ\rho) & \mathsf{E}(\varphi\,\mathcal{R}\,\psi) = \nu\rho(\psi \wedge (\varphi \vee \mathsf{E}\circ\rho)) \\ \mathsf{A}\Box\varphi = \nu\rho(\varphi \wedge \mathsf{A}\circ\rho) & \mathsf{A}(\varphi\,\mathcal{R}\,\psi) = \nu\rho(\psi \wedge (\varphi \vee \mathsf{A}\circ\rho)) \end{array} \tag{4}$$

$$\begin{array}{ll} \mathsf{E}\Diamond\varphi = \mu\rho(\varphi \vee \mathsf{E}\circ\rho) & \mathsf{E}(\varphi\mathcal{U}\psi) = \mu\rho(\psi \vee (\varphi \wedge \mathsf{E}\circ\rho)) \\ \mathsf{A}\Diamond\varphi = \mu\rho(\varphi \vee \mathsf{A}\circ\rho) & \mathsf{A}(\varphi\mathcal{U}\psi) = \mu\rho(\psi \vee (\varphi \wedge \mathsf{A}\circ\rho)) \end{array} \tag{5}$$

This fixpoint characterisation of basic CTL and ECTL modalities as maximal or minimal fixpoints give rise to their analytical classification as α- or β-formulae which are associated, in the tableau with α- and β-rules: $\mathsf{Q}\Box$, and $\mathsf{Q}\,\mathcal{R}$ as maximal fixpoints are classified as α-formulae while $\mathsf{Q}\Diamond$ and $\mathsf{Q}\mathcal{U}$ as minimal fixpoints are β-formulae. This is also reflected in the known equivalences:

$$\begin{array}{ll} \mathsf{E}\Box\varphi = \varphi \wedge \mathsf{E}\circ\mathsf{E}\Box\varphi & \mathsf{E}(\varphi\,\mathcal{R}\,\psi) = \psi \wedge (\varphi \vee \mathsf{E}\circ\mathsf{E}(\varphi\,\mathcal{R}\,\psi)) \\ \mathsf{A}\Box\varphi = \varphi \wedge \mathsf{A}\circ A\Box\varphi & \mathsf{A}(\varphi\,\mathcal{R}\,\psi) = \psi \wedge (\varphi \vee \mathsf{A}\circ\mathsf{A}(\varphi\,\mathcal{R}\,\psi)) \end{array} \tag{6}$$

$$\begin{array}{ll} \mathsf{E}\Diamond\varphi = \varphi \vee \mathsf{E}\circ\mathsf{E}\Diamond\varphi & \mathsf{E}(\varphi\mathcal{U}\psi) = \psi \vee (\varphi \wedge \mathsf{E}\circ\mathsf{E}(\varphi\mathcal{U}\psi)) \\ \mathsf{A}\Diamond\varphi = \varphi \vee \mathsf{A}\circ\mathsf{A}\Diamond\varphi & \mathsf{A}(\varphi\mathcal{U}\psi) = \psi \vee (\varphi \wedge \mathsf{A}\circ\mathsf{A}(\varphi\mathcal{U}\psi)) \end{array} \tag{7}$$

The tableau method determines if a given set of CTL state formulae is satisfiable or not. We precede the formal introduction of the technique by its informal overview. The initial node of the tableaux is labelled by a CTL formula in NNF. To expand the root, and any subsequent node, we apply one of the following rules: α- and β-rules, the "next-state" rule, which reflects a "jump" from a "state" to a "pre-state", and, finally, characteristic to our approach, β^+-rules, where the use of the context (of an eventuality) is essential. The use of the context in these rules, which is a collection of state formulae accompanying the eventuality in the label of the node, forces the soonest fulfillment of eventualities. We apply α-, β-, and β^+-rules repeatedly until we reach a node labelled by $\mathbf{F}$ or by an inconsistent set of formulae, or a node whose labels have already occurred within the path under consideration. In the former case the expansion of the given branch terminates with $\perp$ as its leaf. In the latter case, a repetitive node in the branch means that the branch has a loop – where some subformulae of the given formula are satisfied forever – which could be "bad" or "good". A loop is "bad" when it has a node which contains an unfulfilled eventuality, i.e. none of the nodes of the loop satisfies it. In our procedure, the application of β^+-rules to eventualities is essential to distinguish between "good" and "bad" loops - if β^+-rules have already been applied to every eventuality occurring in the branch then we have a 'good loop' and this branch represents a model for the given formula. Otherwise, we choose an eventuality to which a corresponding β^+-rule has not been applied.

▶ **Definition 5** (Syntactically Consistent Set of Formulae). *A set Σ of state formulae σ is syntactically consistent abbreviated as $\Sigma_\top$ if $\mathbf{F} \notin \Sigma$ and $\{\sigma, \sim\sigma\} \not\subseteq \Sigma$ for any σ; otherwise, Σ is inconsistent denoted as $\Sigma_\perp$.*

▶ **Definition 6** (Tableau, Consistent Node, Closed branch). *A tableau for a set of CTL state formulae Σ is a labelled tree $\langle T, \tau, \Sigma\rangle$, where T is a tree, and τ is a mapping of the nodes of T to the state formulae, elements of Σ, such that the following two conditions hold: (i) The root is labelled by the set Σ. (ii) For any other node $m \in T$, its label $\tau(m)$ is a set of state formulae obtained as the result of the application of one of the rules in Figures 1, 2 and 4 to its parent node n. Given the applied rule is R, we term m an R-successor of n. A node n of* *a tree T is consistent, abbreviated as $n_\top$, if its label, $\tau(n)$, is a syntactically consistent set of formulae (see Def. 5), else n is inconsistent, abbreviated as $n_\bot$. If a branch b of τ, contains $n_\bot \in b$, then b is closed else b is open.*

$$(\wedge)\ \frac{\Sigma, \sigma_1 \wedge \sigma_2}{\Sigma, \sigma_1, \sigma_2} \qquad (\mathsf{Q}\Box)\ \frac{\Sigma, \mathsf{Q}\Box\sigma}{\Sigma, \sigma, \mathsf{Q}\circ\mathsf{Q}\Box\sigma}$$

$$(\vee)\ \frac{\Sigma, \sigma_1 \vee \sigma_2}{\Sigma, \sigma_1 \ \mid\ \Sigma, \sigma_2} \qquad (\mathsf{Q}\mathcal{U})\ \frac{\Sigma, \mathsf{Q}(\sigma_1 \mathcal{U} \sigma_2)}{\Sigma, \sigma_2 \ \mid\ \Sigma, \sigma_1, \mathsf{Q}\circ\mathsf{Q}(\sigma_1 \mathcal{U} \sigma_2)}$$

$$(\mathsf{Q}\mathcal{R})\ \frac{\Sigma, \mathsf{Q}(\sigma_1 \mathcal{R} \sigma_2)}{\Sigma, \sigma_1, \sigma_2 \ \mid\ \Sigma, \sigma_2, \mathsf{Q}\circ\mathsf{Q}(\sigma_1 \mathcal{R} \sigma_2)} \qquad (\mathsf{Q}\Diamond)\ \frac{\Sigma, \mathsf{Q}\Diamond\sigma}{\Sigma, \sigma \ \mid\ \Sigma, \mathsf{Q}\circ\mathsf{Q}\Diamond\sigma}$$

Figure 1 α- and β-Rules.

The rules in Figure 1 follow the standard for the tableaux classification of rules into α-rules and β-rules that for the formulae with CTL modalities are based on their analytic classification reflected in Equations (6)-(7). Thus, if a node, n, in the tableau graph is labelled by a set of formulae, Σ, φ, and a designated formula for the application of tableau rules, φ, is an α-formula - $\mathsf{Q}\Box$ or $\mathsf{Q}\mathcal{R}$, then a corresponding α-rule applies, while if φ is a β-formula - $\mathsf{Q}\Diamond$ or $\mathsf{Q}\mathcal{U}$ then a corresponding β-rule applies. In the latter case we treat Σ as a (possibly empty) context for the eventuality φ. These applications of α- and β-rules generate a set of formulae in the conclusion as a label for the successor node, $n+1$, in case of an α-rule, or as labels of two successors of n, in case of a β-rule.

When a node n is labelled by an *elementary set of formulae* – i.e. a set which exclusively formed by literals and formulae of the form $Q\circ\sigma$ – then this structure is analogous to the construction to a "state" in the terminology of [19]; it enables us to construct the successors of n corresponding to "pre-states" [19]. According to the next proposition we are guaranteed to reach such a tree structure, where the last node of every branch, at this stage of the construction, is a state.

▶ **Proposition 7.** *Any set of CTL state formulae has a tableau T such that the last node of every branch is labelled by an elementary set of state formulae.*

Proof. Repeatedly apply to every expandable node any applicable α- or β-rule until all expandable nodes are elementary. Then, the next-state rule must be applied to every expandable node. ◀

$$(\mathsf{Q}\circ)\ \frac{\Sigma, \mathsf{A}\circ\sigma_1, \dots, \mathsf{A}\circ\sigma_\ell, \mathsf{E}\circ\sigma'_1, \dots, \mathsf{E}\circ\sigma'_k,}{\sigma_1, \dots, \sigma_\ell, \sigma'_1 \ \&\ \dots\ \&\ \sigma_1, \dots, \sigma_\ell, \sigma'_k} \quad \text{where } \Sigma \text{ is a set of literals.}$$

Figure 2 Next-state Rule. ("&" joins AND-successors in the conclusion.)

Proposition 7 enables the application of the so-called "next-state rule" depicted in Figure 2. Applying this rule we split the current branch at node n where the set Σ,$\mathsf{A}\circ\sigma_1, \dots, \mathsf{A}\circ\sigma_\ell$, $\mathsf{E}\circ\sigma'_1$, $\dots, \mathsf{E}\circ\sigma'_k$ is satisfied, into k branches (i.e. into the number of branches equal to the number of $\mathsf{E}\circ$ constraints) where the successors of n along these branches are AND-successors, and are labelled each by a different set $\sigma_1, \dots, \sigma_\ell, \sigma'_i$, for $i \in \{1, \dots, k\}$. This rule splits branches in a "conjunctive" way, and we use the symbol & to represent the generation of AND-successors of node n. Thus, the graphs generated by the tableaux with the application of the "next-state" rule are AND-OR trees. The subsequent construction of a tableau, additionally, involves rules that are applied to so called "uniform sets of formulae".

▶ **Definition 8** (Uniform Set of Formulae). *A set of CTL state formulae Σ is uniform iff Σ is exclusively formed by literals and basic CTL modalities and it has at most one E-formula.*

Applying Proposition 7 to construct a tableau with all its expandable nodes labelled by elementary sets of formulae, and then applying the rule $(\mathsf{Q}\circ)$ (to every expandable node), and finally, repeatedly applying (to every expandable node) the rules $(\wedge)$ and $(\vee)$, we can prove Proposition 9 which states that we can also reach the stage where the last nodes of tableaux branches are labelled by uniform sets of formulae.

▶ **Proposition 9.** *Any set of CTL state formulae Σ has a tableau T such that labels of all its expandable nodes are uniform sets of formulae.*

▶ **Definition 10** (Uniform Tableau). *For any set Σ of CTL state formulae, the tableau for Σ provided by Proposition 9 is denoted Uniform_Tableau(Σ).*

Now we illustrate the procedure by a running example and in the subsequent text will gradually explain its main steps with illustrative figures for some parts. The whole tableau is depicted in Appendix C.

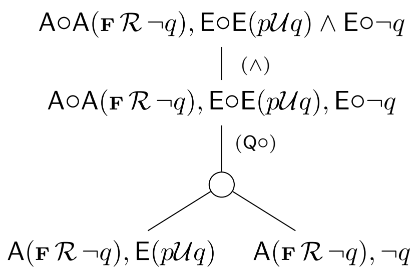

$\mathsf{A}\circ\mathsf{A}(\mathbf{F}\,\mathcal{R}\,\neg q), \mathsf{E}\circ\mathsf{E}(p\,\mathcal{U}q) \wedge \mathsf{E}\circ\neg q$
$(\wedge)$
$\mathsf{A}\circ\mathsf{A}(\mathbf{F}\,\mathcal{R}\,\neg q), \mathsf{E}\circ\mathsf{E}(p\,\mathcal{U}q), \mathsf{E}\circ\neg q$
$(\mathsf{Q}\circ)$
$\mathsf{A}(\mathbf{F}\,\mathcal{R}\,\neg q), \mathsf{E}(p\,\mathcal{U}q)$
$\mathsf{A}(\mathbf{F}\,\mathcal{R}\,\neg q), \neg q$

Figure 3 Example of uniform tableau.

▶ **Example 11.** The given set of formulae $\{\mathsf{A}\circ\mathsf{A}(\mathbf{F}\,\mathcal{R}\,\neg q), \mathsf{E}\circ\mathsf{E}(p\,\mathcal{U}q) \wedge \mathsf{E}\circ\neg q\}$ is not uniform. Hence, by applying the rules $(\wedge)$ and $(\circ)$, we obtain the tableau in Figure 3. The two AND-successors created by the "next-state" rule $(\mathsf{Q}\circ)$ are respectively labelled by the uniform sets: $\mathsf{A}(\mathbf{F}\,\mathcal{R}\,\neg q), \mathsf{E}(p\,\mathcal{U}q)$ and $\mathsf{A}(\mathbf{F}\,\mathcal{R}\,\neg q), \neg q$.

We extend our set of tableau rules with the new two rules named as β^+-rules (Figure 4). Note that the $(\mathsf{Q}\diamondsuit)^+$ rule can be derived from the application of the $(\mathsf{Q}\mathcal{U})^+$ to the CTL formula $\mathbf{T}\mathcal{U}\sigma$. These rules, similarly to β-rules, also split a branch into two branches. These two β^+-rules are the only rules in our system that make use of the context - their application force the eventualities to be satisfied as soon as possible (from the point of the tableau construction where an eventuality is selected to be expanded with a β^+-rule). The context is given by the sets Σ that contain state formulae. In the conclusion of a β^+-rule we add to the conclusion of the corresponding β-rule, a conjunct $\sim\Sigma'$. Recall that this is an NNF of the negation of the conjunction of all formulae in Σ' that are left from Σ after performing the set-theoretical difference constraint indicated in the formulation of the rule. The idea now is that $\sim\Sigma'$ should also be satisfied until σ_2 becomes satisfied. This prevents the repetition of the context while σ_2 is "delayed". Note that Σ' does not include the $A\Box$ (with any prefix of sequence of $\mathsf{A}\circ$) because these formulas would be necessarily repeated along any branch - indeed, if we use $\sim\Sigma$ instead of $\sim\Sigma'$ we will generate a branch for each $\mathsf{A}\Box$ that will be immediately closed.

$$(\mathsf{Q}\mathcal{U})^+ \ \frac{\Sigma, \mathsf{Q}(\sigma_1\mathcal{U}\sigma_2)}{\Sigma, \sigma_2 \ \mid \ \Sigma, \sigma_1, \mathsf{Q}\circ\mathsf{Q}((\sigma_1\wedge\sim\Sigma')\mathcal{U}\sigma_2)} \qquad (\mathsf{Q}\Diamond)^+ \ \frac{\Sigma, \mathsf{Q}\Diamond\sigma}{\Sigma, \sigma \ \mid \ \Sigma, \mathsf{Q}\circ\mathsf{Q}((\sim\Sigma')\mathcal{U}\sigma)}$$

where $\Sigma' = \Sigma \setminus \{(\mathsf{A}\circ)^i\mathsf{A}\Box\sigma \mid i \geq 0 \text{ and } (\mathsf{A}\circ)^i\mathsf{A}\Box\sigma \in \Sigma\}$ and $(A\circ)^i$ stands for i times $\mathsf{A}\circ$.

Figure 4 β^+-Rules.

▶ **Definition 12** (Next-Step Variant). *A state formula $\mathsf{Q}(\sim\Sigma'\mathcal{U}\sigma)$ obtained by the application of a β^+-rule to formula $\mathsf{Q}(\sigma_1\mathcal{U}\sigma_2)$ or $\mathsf{Q}\Diamond\sigma$ is called the next-step variant of $\mathsf{Q}(\sigma_1\mathcal{U}\sigma_2)$.*

$$\begin{tikzcd}
 & \mathsf{A}(\mathbf{F}\,\mathcal{R}\,\neg q), \boxed{\mathsf{E}(p\mathcal{U}q)} \arrow[dl, dash] \arrow[dr, dash, "(\mathsf{E}\mathcal{U})^+"'] & \\
\mathsf{A}(\mathbf{F}\,\mathcal{R}\,\neg q), q & & \mathsf{A}(\mathbf{F}\,\mathcal{R}\,\neg q), p, \mathsf{E}\circ\boxed{\mathsf{E}((p\wedge\mathsf{E}(\mathbf{T}\mathcal{U}q))\mathcal{U}q)}
\end{tikzcd}$$

Figure 5 Application of rule $(\mathsf{E}\mathcal{U})^+$ (we mark in grey the eventuality to which the β^+-rule applies).

▶ **Example 13.** Figure 5 reflects the application of the rule $(\mathsf{E}\mathcal{U})^+$ to the left-most expandable node in Figure 3 (labelled by $\mathsf{E}(p\mathcal{U}q), \mathsf{A}(F\,\mathcal{R}\,\neg q)$). Here, the context of the eventuality $\mathsf{E}(p\mathcal{U}q)$ is the $\mathsf{A}\,\mathcal{R}$-formula. The rule $(\mathsf{E}\mathcal{U})^+$ splits the tableau into two branches. The left successor is labelled by $q, \mathsf{A}(F\,\mathcal{R}\,\neg q)$ and the right successor is labelled by $p, \mathsf{E}\circ\mathsf{E}(p\wedge \mathsf{E}(\mathbf{T}\mathcal{U}q))\mathcal{U}q), \mathsf{A}(F\,\mathcal{R}\,\neg q)$, where the middle formula $\mathsf{E}\circ\mathsf{E}(p\wedge\mathsf{E}(\mathbf{T}\mathcal{U}q))\mathcal{U}q)$ is the next-step variant of the eventuality $\mathsf{E}(p\mathcal{U}q)$ and it contains the NNF of the negation of the context for this eventuality, i.e. $\mathsf{E}(\mathbf{T}\mathcal{U}q) = \mathsf{NNF}\neg\mathsf{A}(F\,\mathcal{R}\,\neg q)$.

4 Systematic Tableau Construction

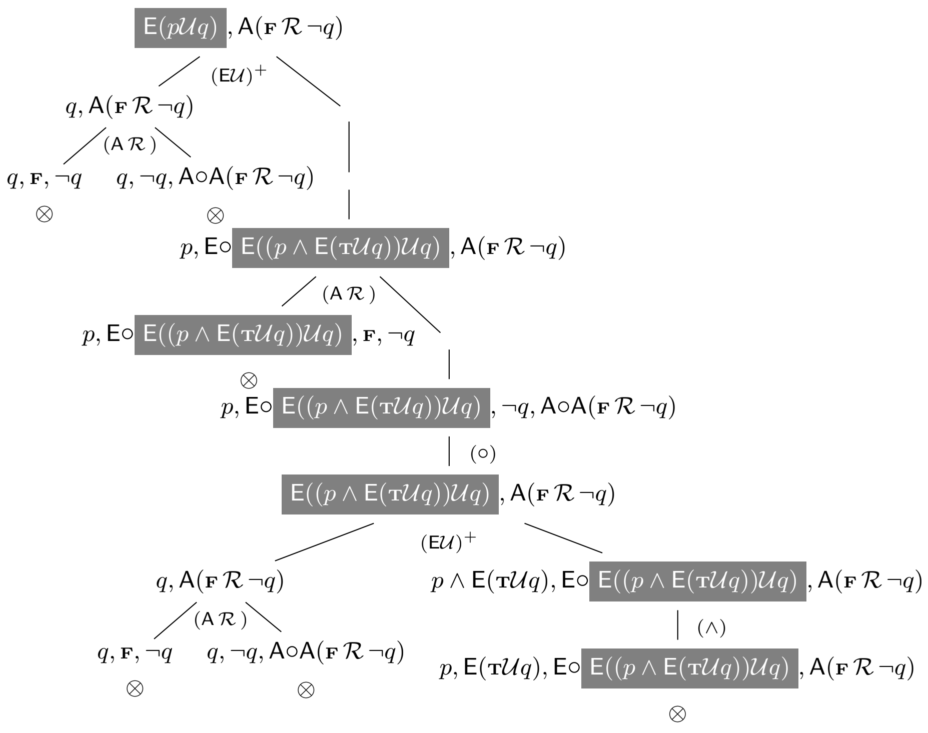
In this section we define an algorithm, $\mathcal{A}^{sys}$, that constructs a *systematic tableau.* Let us observe that, due to the rule $(\mathsf{Q}\circ)$, any open tableau should have a collection of open branches including all the $(\mathsf{Q}\circ)$-successors of any node labelled by an elementary set of formulae. These collections of branches are called *bunches.* Any open bunch of the systematic tableau, constructed by the algorithm $\mathcal{A}^{sys}$ introduced in this section, enables the construction of a model for the initial set of formulae.

The algorithm $\mathcal{A}^{sys}$ constructs an *expanded* tableau (see Definition 25) for the given input. $\mathcal{A}^{sys}$ applied to the input Σ_0, denoted as $\mathcal{A}^{sys}(\Sigma_0)$, returns a systematic tableau $\mathcal{A}^{sys}_{\Sigma_0}$. Intuitively, "expanded" means "complete" in the sense that any possible rule has been already applied at every node. Though the best way to implement this algorithm is a depth-first construction, for clarity, we formulate it as a breadth-first construction of a collection of subtrees. The procedure Uniform_Tableau, in the above Algorithm 1, was introduced in Definition 10 along with the notion of a uniform set of state formulae. The notation $T_1[\ell \leftarrow T_2]$ stands for the tableau T_1 where the expandable ℓ is substituted by the tableau T_2. In particular, $T[\ell \leftarrow$Uniform_Tableau$(\Sigma)]$ is the tableau T where the expandable ℓ is substituted by the Uniform_Tableau(Σ). Procedure Eventuality_Selection chooses an

Algorithm 1 Systematic Tableau Construction.

```
1: procedure SYSTEMATIC_TABLEAU($\Sigma_0$)          ▷ where $\Sigma_0$: set of CTL state formulae
2:     if $\Sigma_0$ is not uniform then $T$ := Uniform_Tableau($\Sigma_0$)
3:     while T has at least one expandable node do
4:                     ▷ Invariant: Any expandable node of $T$ is labelled by an uniform set
5:         Choose any node $\ell$ in $T$ such that $\tau(\ell)$ is expandable          ▷ $\tau(\ell)$ is uniform
6:         if there is no eventuality in $\tau(\ell)$ then $T$ := $T[\ell \leftarrow$Uniform_Tableau($\tau(\ell)$)]
7:         else
8:             Eventuality_Selection($\tau(\ell)$)
9:             Apply_$\beta^+$-rule($\tau(\ell)$)
10:            Let $\ell_1, \ell_2$ the two children of $\ell$
11:            for i = 1 .. 2 do
12:                if $\ell_i$ is expandable and $\tau(\ell_i)$ is not uniform then
13:                    $T$ := $T[\ell_i \leftarrow$Uniform_Tableau($\tau(\ell_i)$)]
```

eventuality to which the corresponding β^+-rule ($(\mathsf{Q}\mathcal{U})^+$ or $(\mathsf{Q}\Diamond)^+$) can be applied. Procedure Apply_β^+-rule(Σ) applies the corresponding β^+-rule to the selected eventuality, it also keeps as . the next-step variant (Definition 12) of such eventuality.



$\mathsf{E}(p\mathcal{U}q), \mathsf{A}(\mathbf{F}\,\mathcal{R}\,\neg q)$
$(\mathsf{E}\mathcal{U})^+$
$q, \mathsf{A}(\mathbf{F}\,\mathcal{R}\,\neg q)$
$(\mathsf{A}\mathcal{R})$
$q, \mathbf{F}, \neg q$
$q, \neg q, \mathsf{A}\circ\mathsf{A}(\mathbf{F}\,\mathcal{R}\,\neg q)$
$\otimes$
$\otimes$
$p, \mathsf{E}\circ\mathsf{E}((p \wedge \mathsf{E}(\mathbf{T}\mathcal{U}q))\mathcal{U}q), \mathsf{A}(\mathbf{F}\,\mathcal{R}\,\neg q)$
$(\mathsf{A}\mathcal{R})$
$p, \mathsf{E}\circ\mathsf{E}((p \wedge \mathsf{E}(\mathbf{T}\mathcal{U}q))\mathcal{U}q), \mathbf{F}, \neg q$
$\otimes$
$p, \mathsf{E}\circ\mathsf{E}((p \wedge \mathsf{E}(\mathbf{T}\mathcal{U}q))\mathcal{U}q), \neg q, \mathsf{A}\circ\mathsf{A}(\mathbf{F}\,\mathcal{R}\,\neg q)$
$(\circ)$
$\mathsf{E}((p \wedge \mathsf{E}(\mathbf{T}\mathcal{U}q))\mathcal{U}q), \mathsf{A}(\mathbf{F}\,\mathcal{R}\,\neg q)$
$(\mathsf{E}\mathcal{U})^+$
$q, \mathsf{A}(\mathbf{F}\,\mathcal{R}\,\neg q)$
$p \wedge \mathsf{E}(\mathbf{T}\mathcal{U}q), \mathsf{E}\circ\mathsf{E}((p \wedge \mathsf{E}(\mathbf{T}\mathcal{U}q))\mathcal{U}q), \mathsf{A}(\mathbf{F}\,\mathcal{R}\,\neg q)$
$(\mathsf{A}\mathcal{R})$
$(\wedge)$
$q, \mathbf{F}, \neg q$
$q, \neg q, \mathsf{A}\circ\mathsf{A}(\mathbf{F}\,\mathcal{R}\,\neg q)$
$p, \mathsf{E}(\mathbf{T}\mathcal{U}q), \mathsf{E}\circ\mathsf{E}((p \wedge \mathsf{E}(\mathbf{T}\mathcal{U}q))\mathcal{U}q), \mathsf{A}(\mathbf{F}\,\mathcal{R}\,\neg q)$
$\otimes$
$\otimes$
$\otimes$

Figure 6 A closed tableau for $\{\mathsf{E}(p\mathcal{U}q), \mathsf{A}(\mathbf{F}\,\mathcal{R}\,\neg q)\}$.

▶ **Example 14.** The application of the Algorithm 1 to the set $\{\mathsf{E}(p\mathcal{U}q), \mathsf{A}(\mathbf{F}\,\mathcal{R}\,\neg q)\}$ shown in Figure 6 selects the eventuality $\mathsf{E}(p\mathcal{U}q)$ and applies the rule $(\mathsf{E}\mathcal{U})^+$ as explained in Example 13. The left successor node is labelled by $q, \mathsf{A}(F\,\mathcal{R}\,\neg q)$. Further expansion of this node by applying the rule $(\mathsf{A}\mathcal{R})$ generates two inconsistent successor nodes. Applying the $\mathsf{A}\mathcal{R}$-rule to the right successor, we obtain the left successor node which is inconsistent and a right successor whose label is an elementary set. Thus, we apply the "next-state" rule generating the successor labelled by the set of two formulae - arguments of $\mathsf{E}\circ$ and $\mathsf{A}\circ$. In this "pre-state" we select the eventuality $\mathsf{E}\mathcal{U}$ and generate two successor nodes applying again the β^+-rule. The left successor is subsequently expanded by two inconsistent successors of the $\mathsf{A}\mathcal{R}$-rule. The right successor is expanded by the $\wedge$-rule and then, since $\mathsf{NNF}(\neg\mathsf{E}(\mathbf{T}\mathcal{U}q)) = \mathsf{A}(\mathbf{F}\mathcal{R}\neg q)$, the node is syntactically inconsistent, because it contains $\mathsf{E}(\mathbf{T}\mathcal{U}q))$ and $\sim \mathsf{E}(\mathbf{T}\mathcal{U}q))$ (see Definition 5).

A tableau for $\mathsf{E}(p\mathcal{U}q), \mathsf{A}(F\mathcal{R}\neg q)$ is also exhibited in [1, 12]. Note the direct correspondence between our context-based tableau (Figure 6) and the one in [1, 12]- they have exactly the same nodes. The right-most branch, in our case, closes by (syntactical) inconsistency, likewise all the other branches. The difference is that, in this branch, the inconsistency comes from the use of the context in the selected eventuality. The corresponding branch in the tableau in [1, 12] is closed by the detection of a "bad loop". Intuitively, whenever the tableau in [1, 12] detects a "bad loop", our tableau is closed by contradiction.

When the input is a satisfiable set, the systematic tableau aims to obtain a loop-node that makes branches eventuality-covered. Next, we define both concepts.

▶ **Definition 15** (Loop-node). *Let b be a tableau branch and $n_i \in b$ $(0 \leq i)$. Then n_i is a loop-node if there exists $n_j \in b$ $(0 \leq j < i)$ such that $\tau(n_i) \subseteq \tau(n_j)$. We say that n_j is a companion node of n_i.*

▶ **Definition 16** (Eventuality-covered Branch). *A tableau branch $b = n_0, n_1, \ldots, n_i$ is eventuality-covered if n_i is a loop-node, with a companion node n_j $(0 \leq j < i)$, both labelled by a uniform set Σ such that every eventuality in $\tau(n_i)$ is selected in some node n_k $(j \leq k < i)$.*

The procedure Eventuality_Selection performs in some fair way that ensures that any open branch will ever be *eventuality-covered.*

▶ **Definition 17** (Non-expandable Node). *A node n is non-expandable if $\tau(n) = \Sigma_\perp$ or n is a loop-node of branch b which is eventuality-covered. Otherwise, n is expandable.*

Consequently, an expandable node is either a node that is not a loop-node or a loop-node whose branch is not eventuality-covered.

▶ **Definition 18** (Bunch in a Tableau, Closed Bunch and Tableau). *A bunch b is a collection of branches that is maximal with respect to $(\mathsf{Q}\circ)$-successor, i.e. every $(\mathsf{Q}\circ)$-successor of any node in b is also in b. A bunch b is a closed bunch if, and only if, at least one of its branches is closed, otherwise it is open. A tableau is closed if, and only if, all its bunches are closed.*

Therefore, any open tableau has at least one open bunch, formed by one or more open branches. Open branches are ended in a loop node. Open bunches represent models, specifically *cyclic models* as defined in Appendix A.

5 Extending the Tableau from CTL to ECTL

In this section we explain a (relatively easy) way to extend the CTL tableau method to the more expressive logic ECTL. This is achieved by adding the new rules given in Figure 7. The α-rule $(\mathsf{Q}\Box\Diamond)$ and the β-rule $(\mathsf{Q}\Diamond\Box)$ that respectively correspond to the following logical equivalences for the basic modalities that extend CTL to ECTL:

$$\begin{array}{ll} \mathsf{E}\Box\Diamond\sigma \equiv \mathsf{E}\Diamond\sigma \wedge \mathsf{E}\circ\mathsf{E}\Box\Diamond\sigma & \mathsf{E}\Diamond\Box\sigma \equiv \mathsf{E}\Box\sigma \vee (\mathsf{E}\Diamond\sigma \wedge \mathsf{E}\circ\mathsf{E}\Diamond\Box\sigma) \\ \mathsf{A}\Box\Diamond\sigma \equiv \mathsf{A}\Diamond\sigma \wedge \mathsf{A}\circ\mathsf{A}\Box\Diamond\sigma & \mathsf{A}\Diamond\Box\sigma \equiv \mathsf{A}\Box\sigma \vee (\mathsf{A}\Diamond\sigma \wedge \mathsf{A}\circ\mathsf{A}\Diamond\Box\sigma) \end{array} \quad (8)$$

There are no additional β^+-rules: eventualities $\mathsf{Q}\Diamond\sigma$ introduced by the rules in Figure 7 are CTL-modalities handled by the β^+-rules of the method for CTL.

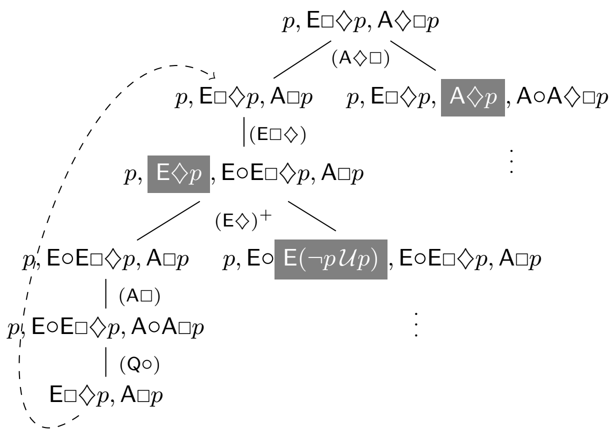
$$(\mathsf{Q}\Box\Diamond)\ \frac{\Sigma, \mathsf{Q}\Box\Diamond\sigma}{\Sigma, \mathsf{Q}\Diamond\sigma, \mathsf{Q}\circ\mathsf{Q}\Box\Diamond\sigma} \qquad (\mathsf{Q}\Diamond\Box)\ \frac{\Sigma, \mathsf{Q}\Diamond\Box\sigma}{\Sigma, \mathsf{Q}\Box\sigma \ \mid\ \Sigma, \mathsf{Q}\Diamond\sigma, \mathsf{Q}\circ\mathsf{Q}\Diamond\Box\sigma}$$

Figure 7 Rules for extending CTL to ECTL.

To complete the extension, let us recall that in [4] some (subsumption-like) simplification rules are needed to ensure the termination of the tableau method for the logic ECTL$^\#$. Though the method for CTL does not need any of such rules, the handling of the more expressive modalities in ECTL – by the rules in Figure 7 – combined with our β^+-rules, requires the following simplification rule:

$$(\sqsubset \mathsf{Q}\mathcal{U})\ \{\mathsf{Q}((\sigma_1 \wedge \chi)\mathcal{U}\sigma_2), \mathsf{Q}(\sigma_1\mathcal{U}\sigma_2)\} \longrightarrow \{\mathsf{Q}((\sigma_1 \wedge \chi)\mathcal{U}\sigma_2)\} \tag{9}$$

By means of this rule, any next-step variant of an eventuality φ subsumes the original eventuality φ that could appear repeatedly after the application of one of the rules in Figure 7.



$p, \mathsf{E}\Box\Diamond p, \mathsf{A}\Diamond\Box p$
$(\mathsf{A}\Diamond\Box)$
$p, \mathsf{E}\Box\Diamond p, \mathsf{A}\Box p$
$p, \mathsf{E}\Box\Diamond p, \mathsf{A}\Diamond p, \mathsf{A}\circ\mathsf{A}\Diamond\Box p$
$(\mathsf{E}\Box\Diamond)$
$p, \mathsf{E}\Diamond p, \mathsf{E}\circ\mathsf{E}\Box\Diamond p, \mathsf{A}\Box p$
$(\mathsf{E}\Diamond)^+$
$p, \mathsf{E}\circ\mathsf{E}\Box\Diamond p, \mathsf{A}\Box p$
$p, \mathsf{E}\circ\mathsf{E}(\neg p\mathcal{U}p), \mathsf{E}\circ\mathsf{E}\Box\Diamond p, \mathsf{A}\Box p$
$(\mathsf{A}\Box)$
$p, \mathsf{E}\circ\mathsf{E}\Box\Diamond p, \mathsf{A}\circ\mathsf{A}\Box p$
$(\mathsf{Q}\circ)$
$\mathsf{E}\Box\Diamond p, \mathsf{A}\Box p$

Figure 8 ECTL tableau for $\{p, \mathsf{E}\Box\Diamond p, \mathsf{A}\Diamond\Box p\}$ ($\vdots$ means this branch expansion is not depicted).

▶ **Example 19.** Figure 8 shows an open tableau with the application of the two rules added to extend CTL to ECTL. We outline a single branch where the ECTL-rules of Figure 7 exclusively apply in the first two steps whilst the rest of steps always apply CTL-rules. We only show the left-most branch because it is an expanded open branch from which the model $\langle p\rangle^\omega$ can be constructed.

6 Conclusion

We introduced a one-pass context-based tableau method for temporal logics CTL and ECTL, providing the soundness and completeness arguments and illustrating the method on a number of examples. The distinctive feature of the method presented in the paper, is that the core tableau construction is based on the concept of a context of an eventuality. The method developed in the paper is much simpler than the analogous technique obtained earlier for a richer logic - $\mathsf{ECTL}^{\#}$ where two types of context (both outer and inner contexts) are used. The construction in this paper only uses the "outer" context, however, similar to $\mathsf{ECTL}^{\#}$, generates tableaux as AND-OR trees.

Our results provide intuitive tableau methods that serve as decision procedures of CTL and ECTL satisfiability. The results of this paper also give us formalisms which are well suited for the automation and are amenable for the implementation, and for the formulation of a dual sequent calculi. All these enable a potential application of the developed tableau methods in certified model checking.

The two tableau methods presented here have double-exponential time worst case complexity. Indeed, a trivial adaptation of [11] allows us to say that the so-called *closure* – the set of all formulas that could appear in a tableau – has in the worst case size $O(2^{O(2^n)})$, where n is the input formula size (this complexity characterisation matches the one of [1, 12]). However, in practice the worst case is very unusual. More often, for example when the context of an eventuality mostly contains modalities $\mathsf{A}\Box$ (which is typical in reactive systems specifications), the number of possible contexts is much smaller and consequently performance is much better.

References

1 P. Abate, R. Goré, and F. Widmann. One-pass tableaux for computation tree logic. In N. Dershowitz and A. Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning*, pages 32–46, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. doi:10.1007/978-3-540-75560-9_5.

2 A. Abuin, A. Bolotov, U. Díaz-de-Cerio, M. Hermo, and P. Lucio. Towards certified model checking for PLTL using one-pass tableaux. In Johann Gamper, Sophie Pinchinat, and Guido Sciavicco, editors, *26th International Symposium on Temporal Representation and Reasoning, TIME 2019, October 16-19, 2019, Málaga, Spain*, volume 147 of *LIPIcs*, pages 12:1–12:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.TIME.2019.12.

3 M. Ben-Ari, A. Pnueli, and Z. Manna. The temporal logic of branching time. *Acta Inf.*, 20(3):207–226, September 1983. doi:10.1007/BF01257083.

4 A. Bolotov, M. Hermo, and P. Lucio. Branching-time logic ECTL# and its tree-style one-pass tableau: Extending fairness expressibility of ECTL+. *Theoretical Computer Science*, 813:428–451, 2020. doi:10.1016/j.tcs.2020.02.015.

5 J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 1020 states and beyond. *Information and Computation*, 98(2):142–170, 1992. doi:10.1016/0890-5401(92)90017-A.

6 E. M. Clarke and E. A. Emerson. Using Branching Time Temporal Logic to Synthesise Synchronisation Skeletons. *Science of Computer Programming*, pages 241–266, 1982. doi:10.1016/0167-6423(83)90017-5.

7 E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, 8(2):244–263, 1986. doi:10.1145/567067.567080.

8 E. A. Emerson. Temporal and modal logic. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science (Vol. B)*, pages 995–1072. MIT Press, Cambridge, USA, 1990.

9 E. A. Emerson and J. Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *Journal of Computer and System Sciences*, 30(1):1–24, 1985. doi:10.1016/0022-0000(85)90001-7.

10 E. A. Emerson and J. Y. Halpern. Sometimes and not never revisited: On branching versus linear time temporal logic. *J. ACM*, 33(1):151–178, 1986. doi:10.1145/4904.4999.

11 J. Gaintzarain, M. Hermo, P. Lucio, M. Navarro, and F. Orejas. Dual systems of tableaux and sequents for PLTL. *Journal of Logic and Algebraic Programming*, 78(8):701–722, 2009. doi:10.1016/j.jlap.2009.05.001.

12 R. Goré. And-or tableaux for fixpoint logics with converse: LTL, CTL, PDL and CPDL. In Stéphane Demri, Deepak Kapur, and Christoph Weidenbach, editors, *Automated Reasoning - 7th International Joint Conference, IJCAR 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 19-22, 2014. Proceedings*, volume 8562 of *Lecture Notes in Computer Science*, pages 26–45. Springer, 2014. doi:10.1007/978-3-319-08587-6_3.

13 R. Goré. Tableau methods for modal and temporal logics. In Marcello D'Agostino, Dov M. Dov Gabbay, Reiner Hähnle, and Joachim Posegga, editors, *Handbook of Tableau Methods*, pages 297–396. Springer, Netherlands, Dordrecht, 1999.

14 R. Kashima. An axiomatization of ECTL. *J. Log. Comput.*, 24(1):117–133, 2014. doi:10.1093/logcom/ext005.

15 N. Markey. Temporal logics. Course notes, Master Parisien de Recherche en Informatique, Paris, France, 2013. URL: http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/NM-coursTL13.pdf.

16 A. Mebsout and C. Tinelli. Proof certificates for SMT-based model checkers for infinite-state systems. In *Proceedings of the 16th Conference on Formal Methods in Computer-Aided Design*, FMCAD '16, pages 117–124, 2016. doi:10.1109/FMCAD.2016.7886669.

17 Stefan Schwendimann. A new one-pass tableau calculus for pltl. In *International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, pages 277–291. Springer, 1998. doi:10.1007/3-540-69778-0_28.

18 A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*, 32(3):733–749, 1985. doi:10.1145/3828.3837.

19 P. Wolper. The tableau method for temporal logic: An overview. *Logique Et Analyse*, 28(110-111):119–136, 1985.

A Interpretation of CTL-type Logics Over Cyclic Structures

In this appendix we define cyclic models and discuss their ability to characterise satisfiability in branching temporal logics.

▶ **Definition 20** (Cyclic Sequence, Cyclic Path and Cyclic Kripke structure)**.** *Let z be a finite sequence of states $z = s_0, s_1, \ldots, s_j$ such that, for every $0 \leq k < j$, $(s_k, s_{k+1}) \in R$. Then, z is cyclic iff there exists s_i, $0 \leq i \leq j$ such that $(s_j, s_i) \in R$. Let z be a finite cyclic sequence, the subsequence $s_i, \ldots, s_j$ of z is called a loop and s_i is called the cycling element. We denote the loop as $\langle s_i, \ldots, s_j\rangle^\omega$. A cyclic path over z is an infinite sequence $\mathsf{path}(z) = s_0, s_1, \ldots, s_{i-1}\langle s_i, s_{i+1}, \ldots, s_j\rangle^\omega$. A Kripke structure $\mathcal{K}$ is cyclic if every fullpath is a cyclic path over a cyclic sequence of states.*

Cyclic paths are also known as *ultimately periodic* paths.

The fact that CTL (ECTL) satisfiability can be reduced to the interpretation over cyclic models only is derived from the existence of the finite model property [9], see also [14]. Hence, for any CTL (ECTL) formula φ, such that $\mathsf{Mod}(\varphi) \neq \emptyset$, there always exists a model $\mathcal{K} \in \mathsf{Mod}(\varphi)$ such that $\mathcal{K}$ is cyclic. Therefore, when speaking about the satisfiability in CTL (hence ECTL) we can consider *cyclic* Kripke structures.

B Soundness and Completeness

Since CTL and ECTL are sublogics of ECTL$^{\#}$ and the tableau method presented here is the adaptation of the method in [4], in this section we essentially adapt to CTL the soundness and completeness proofs developed in [4]. We firstly prove the soundness and completeness of the tableau method for CTL, and then we extend both results to ECTL.

To prove the soundness of our tableau method for CTL (Theorem 22), we show that every tableau rule in Figures 1, 2 and 4 are sound (or preserve satisfiability) in the sense of the next Lemma 21.

▶ **Lemma 21** (Soundness of the Tableau Rules for CTL). *Consider all the rules in Figures 1, and 2 and 4.*

1. *For any α-rule $\frac{\Sigma}{\Sigma'}$: $\mathsf{Sat}(\Sigma)$ if and only if $\mathsf{Sat}(\Sigma')$.*
2. *For any β- and β^{+}-rule $\frac{\Sigma}{\Sigma_1 | \Sigma_2}$: $\mathsf{Sat}(\Sigma)$ if and only if $\mathsf{Sat}(\Sigma_1)$ or $\mathsf{Sat}(\Sigma_2)$.*
3. *If Σ is a set of consistent literals, then $\mathsf{Sat}(\Sigma, \mathsf{A}\circ\sigma_1, \dots, \mathsf{A}\circ\sigma_\ell, \mathsf{E}\circ\sigma'_1, \dots, \mathsf{E}\circ\sigma'_k)$ if and only if $\mathsf{Sat}(\sigma_1, \dots, \sigma_\ell, \sigma'_i)$ for all $1 \leq i \leq k$.*

Proof. All the items follows very easily by the "systematic" application of the semantic definitions of the modalities, except the "if" direction for the two of β^{+}-rules. We will prove here the "if" direction of the rules $(\mathsf{Q}\mathcal{U})^{+}$ for $\mathsf{Q} = \mathsf{E}$ and $\mathsf{Q} = \mathsf{A}$, because the rules $(\mathsf{Q}\Diamond)^{+}$ are particular cases by abbreviation $\Diamond\sigma = \mathbf{T}\mathcal{U}\sigma$.
For the "if" direction of rule $(\mathsf{E}\mathcal{U})^{+}$, let $\mathcal{K} \models \Sigma, \mathsf{E}(\sigma_1\mathcal{U}\sigma_2)$ and let x be the path in $\mathcal{K}$ such that $\mathcal{K}, x, i \models \Sigma, \mathsf{E}(\sigma_1\mathcal{U}\sigma_2)$. Then, let j be the least $i \geq 0$ such that $\mathcal{K}, x, i \models \sigma_2$. If $j = i = 0$, then $\mathcal{K}, x, 0 \models \Sigma, \sigma_2$. Otherwise, if $j > 0$ then $\mathcal{K}, x, m \models \sigma_1$, for all $0 \leq m < j$. Consider k to be the greatest such m for which $\mathcal{K}, x, k \models \Sigma$. Hence, $\mathcal{K}, x, h \models \sim\Sigma$, for all h such that $k + 1 \leq h < j$. In particular, by definition of Σ' (obtained from Σ) it is easy to see that $\mathcal{K}, x, h \models \sigma$ for every $\sigma \in (\Sigma \setminus \Sigma')$ and for all h such that $0 \leq h < j$. Therefore, $\mathcal{K}, x, h \models \sim\Sigma'$, for all h such that $k + 1 \leq h < j$. Thus, $\mathcal{K}, x, k \models, \sigma_1, \mathsf{E}\circ\mathsf{E}((\sigma_1 \wedge \sim\Sigma')\mathcal{U}\sigma_2)$.
For the "if" direction of rule $(\mathsf{A}\mathcal{U})^{+}$, let us suppose that

$$\mathsf{UnSat}(\Sigma, \sigma_2) \text{ and } \mathsf{UnSat}(\Sigma, \sigma_1, \mathsf{A}\circ\mathsf{A}((\sigma_1 \wedge \sim\Sigma')\mathcal{U}\sigma_2)).$$

We will show that $\mathsf{UnSat}(\Sigma, \mathsf{A}(\sigma_1\mathcal{U}\sigma_2))$. For that, let us consider any arbitrary $\mathcal{K}$ such that $\mathcal{K} \models \Sigma$ to show that $\mathcal{K} \not\models \mathsf{A}(\sigma_1\mathcal{U}\sigma_2)$. By the above unsatisfiability hypothesis, if $\mathcal{K} \models \Sigma$, then both $\mathcal{K} \not\models \sigma_2$ and $\mathcal{K} \not\models \sigma_1 \wedge \mathsf{A}\circ\mathsf{A}((\sigma_1 \wedge \sim\Sigma')\mathcal{U}\sigma_2)$. Then, there are two possible cases. First, if $\mathcal{K} \models \neg\sigma_1 \wedge \neg\sigma_2$, then it is obvious that $\mathcal{K} \not\models \mathsf{A}(\sigma_1\mathcal{U}\sigma_2)$. Second, if $\mathcal{K} \models \neg\sigma_2 \wedge \neg\mathsf{A}\circ\mathsf{A}((\sigma_1 \wedge \sim\Sigma')\mathcal{U}\sigma_2)$, then there exists $x_1 \in \mathsf{fullpaths}(\mathcal{K})$ and $i_1 > 0$ that satisfy both $\mathcal{K}, x_1, j \models \neg\sigma_2$ for all j such that $0 \leq j \leq i_1$, and $\mathcal{K}, x_1, i_1 \models \neg\sigma_1 \vee \Sigma'$. Since all the formulae in $\Sigma \setminus \Sigma'$ are satisfied in all states along all paths, indeed $\mathcal{K}, x_1, i_1 \models \neg\sigma_1 \vee \Sigma$. Therefore, if $\mathcal{K}, x_1, i_1 \models \neg\sigma_1$, then obviously $\mathcal{K} \not\models \mathsf{A}(\sigma_1\mathcal{U}\sigma_2)$. Otherwise, if $\mathcal{K}, x_1, i_1 \models \Sigma$, applying the same reasoning for $\mathcal{K}{\restriction}x_1(i_1)$ as we did above for $\mathcal{K}$, we can conclude that there should be a path $x_2 \in \mathsf{fullpaths}(\mathcal{K}{\restriction}x_1(i_1))$ and some $i_2 > 0$ such that either $\mathcal{K}{\restriction}x_1(i_1), x_2, j \models \neg\sigma_2$ for all j such that $i_1 \leq j \leq i_2$ and $\mathcal{K}{\restriction}x_1(i_1), x_2, i_2 \models \neg\sigma_1 \vee \Sigma$. Hence, if $\mathcal{K}{\restriction}x_1(i_1), x_2, i_1 \models \neg\sigma_1$, then trivially $\mathcal{K} \not\models \mathsf{A}(\sigma_1\mathcal{U}\sigma_2)$. Otherwise, $\mathcal{K}{\restriction}x_1(i_1), x_2, i_1 \models \Sigma$. Hence, there are two possible scenarios: 1.) After a finite number of iterations we get a path $y = x_1^{\leq i_1} x_2^{\leq i_2} \cdots x_k^{\leq i_k}$ such that $\mathcal{K}, y, j \models \neg\sigma_2$ for all j such that $0 \leq j \leq i_k$ and $\mathcal{K}, y, i_k \models \neg\sigma_1$. 2.) The infinite iteration of the second case yields a path $y = x_1^{\leq i_1} x_2^{\leq i_2} \cdots x_k^{\leq i_k} \cdots$ (that exists by the limit closure property) such that $\mathcal{K}, y, i \models \neg\sigma_2$ for all $i \geq 0$. In both scenarios we have $\mathcal{K} \not\models \mathsf{A}(\sigma_1\mathcal{U}\sigma_2)$ holds for any arbitrary $\mathcal{K}$ that satisfies Σ. Thus, $\mathsf{UnSat}(\Sigma, \mathsf{A}(\sigma_1\mathcal{U}\sigma_2))$. ◀

▶ **Theorem 22** (Soundness of the Tableau Method for CTL). *Given any set of state formulae Σ, if there exists a closed tableau for Σ then $\mathsf{UnSat}(\Sigma)$.*

Proof. In a closed tableau for Σ, the set of formulae labelling at least one leaf in each bunch is inconsistent and therefore unsatisfiable. Then, by Lemma 21, the labelling of the root node, Σ, is unsatisfiable. ◀

Next, we prove the refutational completeness of the tableau method for CTL (Theorem 29). For that, we firtsly define the notion of stage and prove some auxiliary properties on the stages and bunches of the systematic tableau, that are necessary to prove that every open bunch in the systematic tableau represents a model of the initial set of formulae (Lemma 28).

▶ **Definition 23** (Stage). *Given a branch, b of a tableau T, a stage in T is every maximal subsequence of successive nodes $n_i, n_{i+1}, \ldots, n_j$ in b such that $\tau(n_k)$ is not a $(\mathsf{Q}\circ)$-child of $\tau(n_{k-1})$, for all k such that $i < k \leq j$. We denote by $\mathsf{stages}(b)$ the sequence of all stages of b. The successor relation on $\mathsf{stages}(b)$ is induced by the successor relation on b. The labelling function τ is extended to stages as the union of the original τ applied to every node in a stage.*

▶ **Definition 24** ($\alpha\beta^+$-saturated Stage). *We say that a stage $s = n_i, \ldots, n_j$ in $\mathcal{A}^{sys}_{\Sigma}$ is $\alpha\beta^+$-saturated if and only if it satisfies the following conditions:*

1. *For all $\sigma_1 \wedge \sigma_2 \in \tau(s)$: $\{\sigma_1, \sigma_2\} \subseteq \tau(s)$.*
2. *For all $\mathsf{Q}\Box\sigma \in \tau(s)$: $\{\sigma, \mathsf{Q}\circ\mathsf{Q}\Box\sigma\} \subseteq \tau(s)$.*
3. *For all $\sigma_1 \vee \sigma_2 \in \tau(s)$: $\sigma_1 \in \tau(s)$ or $\sigma_2 \in \tau(s)$.*
4. *For all $\mathsf{Q}(\sigma_1 \mathcal{R} \sigma_2) \in \tau(s)$: $\{\sigma_1, \sigma_2\} \subseteq \tau(s)$ or $\{\sigma_2, \mathsf{Q}\circ\mathsf{Q}(\sigma_1 \mathcal{R} \sigma_2)\} \subseteq \tau(s)$.*
5. *For all $\mathsf{Q}(\sigma_1\mathcal{U}\sigma_2) \in \tau(s)$: $\sigma_2 \in \tau(s)$ or $\{\sigma_1, \mathsf{Q}\circ\mathsf{Q}(\sigma_1\mathcal{U}\sigma_2)\} \subseteq \tau(s)$ or*
   $\{\sigma_1, \mathsf{Q}\circ\mathsf{Q}((\sigma_1 \wedge \sim\Sigma')\mathcal{U}\sigma_2)\} \subseteq \tau(s)$
   where $\Sigma' = (\tau(n_i) \setminus \{\mathsf{Q}(\sigma_1\mathcal{U}\sigma_2)\}) \setminus \{(\mathsf{A}\circ)^i\mathsf{A}\Box\varphi \mid i \geq 0$ and $(\mathsf{A}\circ)^i\mathsf{A}\Box\varphi \in \tau(n_i)\}$.
6. *For all $\mathsf{Q}(\Diamond\sigma) \in \tau(s)$: $\sigma \in \tau(s)$ or $\{\mathsf{Q}\circ\mathsf{Q}(\Diamond\sigma)\} \subseteq \tau(s)$ or $\{\mathsf{Q}\circ\mathsf{Q}((\sim\Sigma')\mathcal{U}\sigma)\} \subseteq \tau(s)$*
   where $\Sigma' = (\tau(n_i) \setminus \{\mathsf{Q}\Diamond\sigma\}) \setminus \{(\mathsf{A}\circ)^i\mathsf{A}\Box\varphi \mid i \geq 0$ and $(\mathsf{A}\circ)^i\mathsf{A}\Box\varphi \in \tau(n_i)\}$.

▶ **Definition 25** (Expanded Bunch and Tableau). *An open branch b is expanded if each stage $s \in \mathsf{stages}(b)$ is $\alpha\beta^+$-saturated and b is eventuality-covered. A bunch is expanded if all its open branches are expanded. A tableau is expanded if all its open bunches are expanded.*

The construction of the systematic tableau applies exactly one β^+-rule to exactly one selected eventuality (if any) at the first node of the stage, and then applies exhaustively all the applicable α- and β-rules to the formulas in the stage, until the branch closes, or its leaf is labelled by an elementary set, or it contains a loop-node. Consequently, the following Proposition 26 holds which can be trivially proved by construction.

▶ **Proposition 26.** *Given any set of state formulae Σ, the systematic tableau $\mathcal{A}^{sys}_{\Sigma}$ is expanded.*

Next we prove a crucial property of the systematic tableau management of eventualities by means of the selection policy.

▶ **Proposition 27.** *Let b be an open branch of $\mathcal{A}^{sys}_{\Sigma}$ and let $\mathsf{Q}(\sigma_1\mathcal{U}\sigma_2)$ be a formula that is selected at some stage $s_i \in \mathsf{stages}(b)$. Then, there exists some stage $s_k \in \mathsf{stages}(b)$ (for some $k \geq i$) such that $\sigma_2 \in \tau(s_k)$ and $\sigma_1 \in \tau(s_j)$ for all $j \in \{i, \ldots, k-1\}$.*

Proof. By construction, the uniform set labelling the first node at each stage s_j $(j \geq i)$ of b has the form $\Sigma_{s_j}, \mathsf{Q}((\sigma_1 \wedge (\sim\Sigma_{s_i} \wedge \cdots \wedge \sim\Sigma_{s_{j-1}}))\mathcal{U}\sigma_2)$ where each Σ_{s_j} is the context of the selected formula containing the next-step variant of $\mathsf{Q}(\sigma_1\mathcal{U}\sigma_2)$ at the first node of each stage s_j. Since no other β^+-rule is applied each Σ_{s_j} is a subset of the finite set formed by all state formulae that are subformulae of some formula in Σ_{s_i} and their negations. Hence, there are a finite number of different Σ_{s_j}. Therefore, after finitely many applications of the β^+-rule, $\Sigma_{s_h} = \Sigma_{s_j}$, for some $h >= i$, for some $j \in \{i, \dots, h-1\}$, and $\sigma_1 \wedge (\sim \Sigma_{s_i} \wedge \cdots \wedge \sim \Sigma_{s_{h-1}}) \in \tau(s_h)$. In particular, $\sim \Sigma_{s_h} \in \tau(s_h)$, hence, Σ_{s_h} must be inconsistent. Since b is open, this is a contradiction. This means that, for some $k \geq i$ the application of the corresponding β^+-rule should force that $\sigma_2 \in \tau(s_k)$. In addition, by Proposition 26 and Definition 24(5), $\sigma_1 \in \tau(s_j)$ for all $j \in \{i, \dots, k-1\}$. ◀

▶ **Lemma 28** (Model Existence). *Let Σ be any set of formulae. For any expanded bunch H of $\mathcal{A}^{sys}_{\Sigma}$, there exists a Kripke structure $\mathcal{K}_H$ such that $\mathcal{K}_H \models \Sigma$.*

Proof. Let H be any expanded bunch of $\mathcal{A}^{sys}_{\Sigma}$. We define $\mathcal{K}_H = (S, R, L)$ such that $S = \bigcup_{b \in H} \mathsf{stages}(b)$ and for any $s \in S$: $L(s) = \{p \mid p \in \tau(n) \cap \mathsf{Prop}$ for some node $n \in s\}$; and R is the relation induced in $\mathsf{stages}(b)$ for each $b \in H$. Any branch in $b \in H$ is open, hence b ends in a loop-node. Moreover, every eventuality has been selected in some stage of b. Hence, there exists a (possibly empty) uniform set Σ_ℓ such that for some $i \geq 0$: $b = s_0, s_1, \dots, s_{i-1}, s_i, s_{i+1}, \dots, s_j, n_\ell$, where each s_h stands for a stage and n_ℓ is a non-expandable loop-node labelled by Σ_ℓ whose companion node is the first node at stage s_i. We are going to prove the following fact:

$$\mathcal{K}_H, s_a, 0 \models \sigma \text{ for any } a \in \{0, \dots, j\} \text{ and any formula } \sigma \text{ in } L(s_a)$$

by structural induction on the formula σ.
The base of the induction, for $\sigma = p \in \mathsf{Prop}$, follows by definition of $\mathcal{K}_H$.
The cases where σ has one of the forms $\sigma_1 \wedge \sigma_2$, $\mathsf{Q}\Box\sigma$, $\sigma_1 \vee \sigma_2$ and $\mathsf{Q}(\sigma_1 \mathcal{R} \sigma_2)$ are trivial by Definition 24 and the induction hypothesis. Hence, to complete the inductive proof we will show that $\mathcal{K}_H, s_a, 0 \models \mathsf{Q}(\sigma_1 \mathcal{U} \sigma_2)$ for any $\mathsf{Q}(\sigma_1 \mathcal{U} \sigma_2) \in L(s_a)$. The case for all $\mathsf{Q}\Diamond\sigma \in L(s_a)$ follows as a particular case by $\Diamond\sigma \equiv \mathbf{T}\mathcal{U}\sigma$.
Consider any $\mathsf{Q}(\sigma_1 \mathcal{U} \sigma_2) \in L(s_a)$. Since b is eventuality-covered and n_ℓ is a loop-node, $\mathsf{Q}(\sigma_1 \mathcal{U} \sigma_2)$ must be the selected eventuality at some node between the states s_a and s_j. Hence, by Proposition 27 and the definition of $\mathcal{K}_H$, there should be a state $s_k \in S$ (for some $a \leq k \leq j$) such that $\sigma_2 \in L(s_k)$ and $\sigma_1 \in L(s_z)$ for all $z \in \{a, \dots, k-1\}$. Then, by induction hypothesis, $\mathcal{K}_H, s_k, 0 \models \sigma_2$ and $\mathcal{K}_H, s_z, 0 \models \sigma_1$ for all $z \in \{a, \dots, k-1\}$. Therefore, $\mathcal{K}_H, s_a, 0 \models \mathsf{Q}(\sigma_1 \mathcal{U} \sigma_2)$.
To complete the proof, we show that the successor relation between states in $\mathcal{K}_H$ is well-defined. For that, consider any tableau node in any stage s_a that is labelled by an elementary set

$$\{\Sigma, \mathsf{A}\circ\sigma_1, \dots, \mathsf{A}\circ\sigma_n, \mathsf{E}\circ\sigma'_1, \dots, \mathsf{E}\circ\sigma'_k\}$$

where Σ is a consistent set of literals, by rule $(\mathsf{Q}\circ)$, s_a has (in $\mathcal{K}_H$) a successor state s^i_{a+1}, for each $i \in \{1, \dots, k\}$, such that $L(s^i_{a+1}) = \{\sigma_1, \dots, \sigma_n, \sigma'_i\}$. We can assume (by the above proved fact) that $\mathcal{K}_H, s^i_{a+1}, 0 \models \{\sigma_1, \dots, \sigma_n, \sigma'_i\}$ for all $i \in \{1, \dots, k\}$. Therefore, we can infer that $\mathcal{K}_H, s_a, 0 \models \{\Sigma, \mathsf{A}\circ\sigma_1, \dots, \mathsf{A}\circ\sigma_n, \mathsf{E}\circ\sigma'_1, \dots, \mathsf{E}\circ\sigma'_k\}$. ◀

Next, we prove the refutational completeness of the tableau method.

▶ **Theorem 29** (Refutational Completeness for CTL). *For any set of state formulae Σ, if $\mathsf{UnSat}(\Sigma)$ then there exists a closed tableau for Σ.*

Proof. Suppose the contrary, that there exists no closed tableau for Σ. Then the systematic tableau $\mathcal{A}_\Sigma^{sys}$ is open. Hence, there is at least one expanded bunch H in $\mathcal{A}_\Sigma^{sys}$. By Lemma 28, there exists a Kripke structure $\mathcal{K}_H$ such that $\mathcal{K}_H \models \Sigma$. Consequently, $\mathsf{Sat}(\Sigma)$. ◀

Finally, we prove the completeness of our tableau method for CTL.

▶ **Theorem 30** (Termination of the Tableau Method for CTL)**.** *For any set of state formulae Σ , the construction of the expanded tableau $\mathcal{A}_\Sigma^{sys}$ terminates.*

Proof. Tableau rules produce a finite branching, hence König's Lemma, applies. Therefore, it suffices to prove that every branch is finite. By Proposition 27, the application of a β^+-rule to a selected formula stops after a finite number of steps. Since the number of selectable eventualities in any open branch is finite, any open branch is eventuality-covered after a finite number of eventuality selections. Recall that we assume the eventuality selection strategy to be fair. ◀

▶ **Theorem 31** (Completeness of the Tableau Method for CTL)**.** *For any set of state formulae Σ, if Σ is satisfiable then there exists a (finite) open expanded tableau for Σ.*

Proof. The existence of the systematic tableau $\mathcal{A}_\Sigma^{sys}$ suffices to prove this fact, by Theorem 30. ◀

Now, we explain how the proofs of these metatheorems for CTL can be extended to ECTL. Firstly, we extend the soundness of the tableau rules, in the sense of Lemma 21, to the rules in Figure 7.

▶ **Lemma 32.** *For any ECTL set of state formulae Σ and any state formula σ:*

1. *$\mathsf{Sat}(\Sigma, \mathsf{Q}\Box\Diamond\sigma)$ if and only if $\mathsf{Sat}(\Sigma, \mathsf{Q}\Diamond\sigma, \mathsf{Q}\circ\mathsf{Q}\Box\Diamond\sigma)$.*
2. *$\mathsf{Sat}(\Sigma, \mathsf{Q}\Diamond\Box\sigma)$ if and only if $\mathsf{Sat}(\Sigma, \mathsf{Q}\Box\sigma)$ or $\mathsf{Sat}(\Sigma, \mathsf{Q}\Diamond\sigma, \mathsf{Q}\circ\mathsf{Q}\Diamond\Box\sigma)$.*

Proof. It follows by "systematic" application of the semantic definitions of the modalities $\mathsf{Q}\Box\Diamond$ and $\mathsf{Q}\Diamond\Box$ given by the equivalences (8) in Section 5. ◀

To extend the refutational completeness result to ECTL, we firstly extend the Definition 24 with the following additional conditions for a stage to be $\alpha\beta^+$-saturated:

7. For all $\mathsf{Q}\Box\Diamond\sigma \in \tau(s)$: $\{\mathsf{Q}\Diamond\sigma, \mathsf{Q}\circ\mathsf{Q}\Box\Diamond\sigma\} \subseteq \tau(s)$.
8. For all $\mathsf{Q}\Diamond\Box\sigma \in \tau(s)$: $\{\mathsf{Q}\Box\sigma\} \subseteq \tau(s)$ or $\{\mathsf{Q}\Diamond\sigma, \mathsf{Q}\circ\mathsf{Q}\Diamond\Box\sigma\} \subseteq \tau(s)$.

It is obvious that these two additional conditions are satisfied in any stage of the systematic tableau by construction. Using these conditions, it is routine to prove that $\mathcal{K}_H$ defined in Lemma 28 satisfies the fact that: $\mathcal{K}_H, s_a, 0 \models \sigma$ for any $a \in \{0, \ldots, j\}$ and any formula of the forms $\mathsf{Q}\Box\Diamond\sigma, \mathsf{Q}\Diamond\Box\sigma$ that belongs to $L(s_a)$. Therefore, refutational completeness (i.e. Theorem 29) extends to ECTL.

Finally, to extend the termination result (see proof of Theorem 30), it suffices to ensure that the rules in Figure 7 do not affect the behaviour of the β^+-rules on the selected eventualities in the sense that Proposition 27 is preserved. Since each application of a rule in Figure 7 introduces a new $\mathsf{Q}\Diamond\sigma$, in Section 5 we have introduced the simplification rule $(\sqsubset \mathsf{Q}\mathcal{U})$ (see (9) to subsume any occurrence of the eventuality φ by any next-step variant of φ. Therefore, Proposition 27 holds and hence Theorem 30 trivially extends to ECTL.

C The Running Example Tableau

In Figure 9 we depict the whole tableau for the running example we use in the paper. Note that the Q rule at step 2, denoted with a big circle generates two AND-successors, where the left successor has the closed tableau - this is explained in the paper. Hence, the bunch is closed, in spite of the open tableau at the right successor of the AND-node.

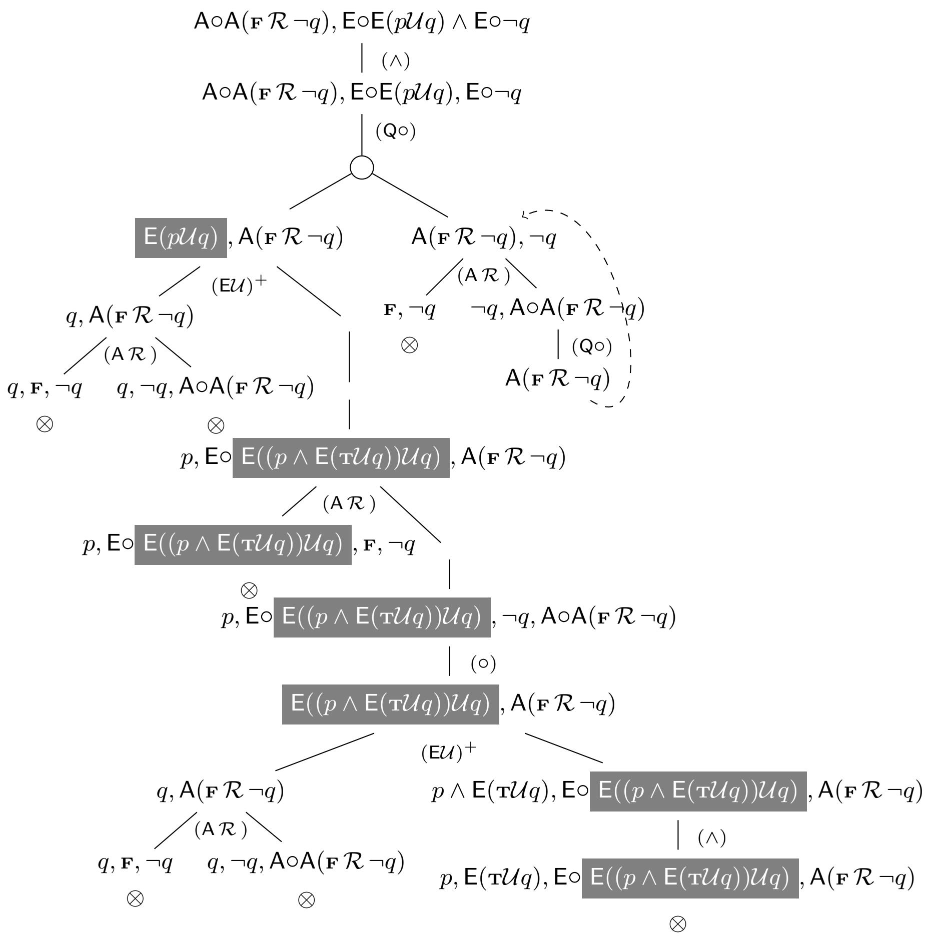

A∘A(F R ¬q), E∘E(pUq) ∧ E∘¬q
(∧)
A∘A(F R ¬q), E∘E(pUq), E∘¬q
(Q∘)
E(pUq), A(F R ¬q)
A(F R ¬q), ¬q
(EU)+
(A R)
q, A(F R ¬q)
F, ¬q
¬q, A∘A(F R ¬q)
(A R)
(Q∘)
q, F, ¬q
q, ¬q, A∘A(F R ¬q)
A(F R ¬q)
p, E∘ E((p ∧ E(TUq))Uq), A(F R ¬q)
(A R)
p, E∘ E((p ∧ E(TUq))Uq), F, ¬q
p, E∘ E((p ∧ E(TUq))Uq), ¬q, A∘A(F R ¬q)
(∘)
E((p ∧ E(TUq))Uq), A(F R ¬q)
(EU)+
q, A(F R ¬q)
p ∧ E(TUq), E∘ E((p ∧ E(TUq))Uq), A(F R ¬q)
(A R)
(∧)
q, F, ¬q
q, ¬q, A∘A(F R ¬q)
p, E(TUq), E∘ E((p ∧ E(TUq))Uq), A(F R ¬q)

Figure 9 A closed tableau for $\{\mathsf{A}\circ\mathsf{A}(\mathbf{F}\,\mathcal{R}\,\neg q), \mathsf{E}\circ\mathsf{E}(p\mathcal{U}q) \wedge \mathsf{E}\circ\neg q\}$.