Completeness and Decidability Results for CTL in Coq

Christian Doczkal Gert Smolka

Published in Proc. of ITP 2014, Vienna, Austria, LNAI 8558, Springer, 2014
Revised June 17, 2015

We prove completeness and decidability results for the temporal logic CTL in Coq/Ssreflect. Our basic result is a constructive proof that for every formula one can obtain either a finite model satisfying the formula or a proof in a Hilbert system certifying the unsatisfiability of the formula. The proof is based on a history-augmented tableau system obtained as the dual of Brünnler and Lange's cut-free sequent calculus for CTL. We prove the completeness of the tableau system and give a translation of tableau refutations into Hilbert refutations. Decidability of CTL and completeness of the Hilbert system follow as corollaries.

1 Introduction

We are interested in a formal and constructive metatheory of the temporal logic CTL [6]. We start with the definitions of formulas, models, and a satisfiability relation relating models and formulas. The models are restricted such that the satisfiability relation is classical. We then formalize a Hilbert proof system and prove it sound for our models. Up to this point everything is straightforward. Our basic result is a constructive proof that for every formula one can obtain either a finite model satisfying the formula or a derivation in the Hilbert system certifying the unsatisfiability of the formula. As corollaries of this result we obtain the completeness of the Hilbert system, the finite model property of CTL, and the decidability of CTL.

Informal and classical proofs of our corollaries can be found in Emerson and Halpern's work on CTL [7, 5]. Their proofs are of considerable complexity as it comes to the construction of models and Hilbert derivations. As is, their completeness proof for the Hilbert system is not constructive and it is not clear how to make it constructive.

Brünnler and Lange [3] present a cut-free sequent system for CTL satisfying a finite subformula property. Due to the subformula property, the sequent system constitutes a decision method for formulas that yields finite counter-models for non-valid formulas. The sequent system is non-standard in that formulas are annotated with histories, which are finite sets of formulas. Histories are needed to handle eventualities (e.g., until formulas) with local rules.

We base the proof of our main result on a tableau system that we obtain by dualizing Brünnler and Lange's sequent system. This is the first tableau system for CTL employing only local rules. Existing tableau methods for CTL [7, 5] combine local rules with global model checking of eventualities. Given a formula, the tableau system either constructs a finite model satisfying the formula or a tableau refutation. We give a translation from tableau refutations to Hilbert refutations, thereby showing the completeness of the Hilbert system and the soundness of the tableau system. The translation is compositional in that it is defined by structural recursion on tableau refutations. For the translation it is essential that the tableau system has only local rules.

With our results it should not be difficult to obtain formal and constructive proofs of the soundness and completeness of Brünnler and Lange's original system.

The standard definition [5] of the satisfiability relation of CTL employs infinite paths, which are difficult to handle in a constructive setting. We avoid infinite paths by capturing the semantics of eventualities with induction and the semantics of co-eventualities with coinduction.

Our formal development consists of about 3500 lines of Coq/Ssreflect. There are three subtasks of considerable complexity. One complex subtask is the construction of finite models from intermediate structures we call demos. Our demos play the role of the pseudo-Hintikka structures in Emerson [5] and are designed such that they go well with the tableau system. Another complex subtask is the construction of a demo from the tableau-consistent clauses in a subformula universe. Finally, the translation of tableau refutations to Hilbert refutations is of considerable complexity, in particular as it comes to the application of the induction axioms of the Hilbert system.

Given the practical importance of CTL and the complex proofs of the metatheoretic results for CTL, we think that the metatheory of CTL is an interesting and rewarding candidate for formalization. No such formalization exists in the literature. In previous work [4] we have prepared this work by proving related results for a weaker modal logic. As it comes to eventualities, which are responsible for the expressiveness and the complexity of the logic, our previous work only captured the simplest eventuality saying that a state satisfying a given formula is reachable.

Our development is carried out in Coq [13] with the Ssreflect [9] extension. We build a library for finite sets on top of Ssreflect's countable types and use it to capture the subformula property. We also include a fixpoint theorem for finite sets and use it to show decidability of tableau derivability.

In each section of the paper, we first explain the mathematical ideas behind the proofs and then comment briefly on the difficulties we faced in the formalization. For additional detail, we refer the reader to Coq development.[1]

2 CTL in Coq

We define the syntax and semantics of CTL as we use it in our formalization. We fix a countable alphabet $\mathcal{AP}$ of atomic propositions p and define formulas as follows:

$$s, t \ := p \mid \bot \mid s \to t \mid \mathsf{AX}\, s \mid \mathsf{A}(s\, \mathsf{U}\, t) \mid \mathsf{A}(s\, \mathsf{R}\, t)$$

We define the remaining propositional connectives using $\to$ and $\bot$. We also use the following defined modal operators: $\mathsf{EX}\, s \equiv \neg\, \mathsf{AX}\, \neg s$, $\mathsf{A}^+(s\, \mathsf{U}\, t) \equiv \mathsf{AX}\, \mathsf{A}(s\, \mathsf{U}\, t)$, $\mathsf{E}(s\, \mathsf{U}\, t) \equiv \neg\, \mathsf{A}(\neg s\, \mathsf{R}\, \neg t)$, $\mathsf{E}^+(s\, \mathsf{U}\, t) \equiv \mathsf{EX}\, \mathsf{E}(s\, \mathsf{U}\, t)$, $\mathsf{E}(s\, \mathsf{R}\, t) \equiv \neg\, \mathsf{A}(\neg s\, \mathsf{U}\, \neg t)$, and $\mathsf{EG}\, t \equiv \mathsf{E}(\bot\, \mathsf{R}\, t)$.

The formulas of CTL are interpreted over transition systems where the states are labeled with proposition symbols. Unlike most of the literature on CTL [5, 7, 1], where the semantics of CTL formulas is defined in terms of infinite paths, we define the semantics of CTL using induction and coinduction. Our semantics is classically equivalent to the standard infinite path semantics but better suited for a constructive formalization.

Let W be a type, $R : W \to W \to \mathit{Prop}$ a relation, and $P, Q : W \to \mathit{Prop}$ predicates. We require that R is serial, i.e., that every $w : W$ has some R-successor. We define the eventuality AU ("always until") inductively as:

$$\frac{Q\, w}{\mathsf{AU}\, R\, P\, Q\, w} \qquad\qquad \frac{P\, w \qquad \forall v. R\, w\, v \Rightarrow \mathsf{AU}\, R\, P\, Q\, v}{\mathsf{AU}\, R\, P\, Q\, w}$$

Further, we define AR ("always release") coinductively.

$$\dfrac{Q\, w \qquad P\, w}{\overline{\mathsf{AR}\, R\, P\, Q\, w}} \qquad\qquad \dfrac{Q\, w \qquad \forall v. R\, w\, v \Rightarrow \mathsf{AR}\, R\, P\, Q\, v}{\overline{\mathsf{AR}\, R\, P\, Q\, w}}$$

Now let $L : \mathcal{AP} \to W \to \mathit{Prop}$ be a labeling function. We evaluate CTL formulas to

[1] `http://www.ps.uni-saarland.de/extras/itp14`.

$$\begin{array}{rl}
\mathsf{K} & s \to t \to s \\
\mathsf{S} & ((u \to s \to t) \to (u \to s) \to u \to t) \\
\mathsf{DN} & ((s \to \bot) \to \bot) \to s \\
\mathsf{N} & \mathsf{AX}(s \to t) \to \mathsf{AX}\, s \to \mathsf{AX}\, t \\
\mathsf{U1} & t \to \mathsf{A}(s\,\mathsf{U}\,t) \\
\mathsf{U2} & s \to \mathsf{AX}\,\mathsf{A}(s\,\mathsf{U}\,t) \to \mathsf{A}(s\,\mathsf{U}\,t) \\
\mathsf{R1} & \mathsf{A}(s\,\mathsf{R}\,t) \to t \\
\mathsf{R2} & \mathsf{A}(s\,\mathsf{R}\,t) \to (s \to \bot) \to \mathsf{AX}\,\mathsf{A}(s\,\mathsf{R}\,t) \\
\mathsf{AX} & \mathsf{AX}\,\bot \to \bot
\end{array}$$

$$\frac{s \qquad s \to t}{t}\,\mathsf{MP} \qquad \frac{s}{\mathsf{AX}\, s}\,\mathsf{Nec} \qquad \frac{t \to u \qquad s \to \mathsf{AX}\, u \to u}{\mathsf{A}(s\,\mathsf{U}\,t) \to u}\,\mathsf{AU_{ind}}$$

$$\frac{u \to t \qquad u \to (s \to \bot) \to \mathsf{AX}\, u}{u \to \mathsf{A}(s\,\mathsf{R}\,t)}\,\mathsf{AR_{ind}}$$

Figure 1: Hilbert Axiomatization of CTL

predicates on W:

$$eval\ p = L\,p \qquad eval\ (s \to t) = \lambda w.eval\ s\ w \implies eval\ t\ w$$
$$eval\ \bot = \lambda\_.False \qquad eval\ (\mathsf{AX}\, s) = \lambda w.\forall v.R\ w\ v \implies eval\ t\ v$$
$$eval\ (\mathsf{A}(s\,\mathsf{U}\,t)) = \mathsf{AU}\, R\ (eval\ s)\ (eval\ t)$$
$$eval\ (\mathsf{A}(s\,\mathsf{R}\,t)) = \mathsf{AR}\, R\ (eval\ s)\ (eval\ t)$$

We say w **satisfies** a formula s, written $w \vDash s$, if we have $eval\ s\ w$. Similar to [4], we consider as models only those serial transition systems (W, R, L) for which

$$\forall s \forall w \in W. w \vDash s \lor w \nvDash s \tag{1}$$

is provable. When $\mathcal{M}$ is a model, we write $\to_{\mathcal{M}}$ for the transition relation of $\mathcal{M}$ and $w \in \mathcal{M}$ if w is a state of $\mathcal{M}$.

Note that having to prove (1) severely restricts our ability to construct infinite models. However, since CTL has the small model property it suffices to construct finite models for our completeness results. For these models (1) is easy to prove. Formalizing models this way allows us to reason about the classical object logic CTL without assuming any classical axioms.

The Hilbert axiomatization we use in our formalization is a variant of the Hilbert system given by Emerson and Halpern [7]. The rules and axioms of the Hilbert axiomatization are given in Figure 1. We write $\vdash s$ if s is provable from the axioms and call a proof of $\neg s$ a **Hilbert refutation** of s.

Theorem 2.1 If $\vdash s$ then $w \vDash s$ for all models $\mathcal{M}$ and states $w \in \mathcal{M}$.

Proof Induction on the derivation of $\vdash s$, using (1) for the cases corresponding to DN and $\mathsf{AR_{ind}}$.

We are now ready to state our basic theorem.

Theorem 2.2 (Certifying Decision Method) For every formula we can construct either a finite model or a Hilbert refutation.

3 A History-Based Tableau System for CTL

The tableau system we use as the basis for our certifying decision method employs signed formulas [11]. A **signed formula** is either s^+ or s^- where s is a formula. Signs bind weaker than formula constructors, so $s \to t^+$ is to be read as $(s \to t)^+$. We write σ for arbitrary signs and $\overline{\sigma}$ for the sign opposite to σ. A state satisfies a signed formula s^σ if it satisfies $\lfloor s^\sigma \rfloor$ where $\lfloor s^+ \rfloor = s$ and $\lfloor s^- \rfloor = \neg s$.

We refer to positive until formulas and negative release formulas as **eventualities**. For the eventuality $\mathsf{A}(s\,\mathsf{R}\,t)^-$ to be satisfied at a state, there must be a path from this state to a state satisfying $\neg t$ that satisfies $\neg s$ on every state along the way.

A **clause** is a finite set of signed formulas and a **history** is a finite set of clauses. The letters C and D range over clauses and the letter H ranges over histories. For the rest of this paper, sets are always assumed to be finite. An **annotated eventuality** is a formula of the form

$$\mathsf{A}(s\,\mathsf{U}_H\,t)^+ \mid \mathsf{A}^+(s\,\mathsf{U}_H\,t)^+ \mid \mathsf{A}(s\,\mathsf{R}_H\,t)^- \mid \mathsf{A}^+(s\,\mathsf{R}_H\,t)^-$$

An **annotation** is either an annotated eventuality or the empty annotation "$\cdot$". The letter a ranges over annotations. An **annotated clause** is a pair $C|a$ of a clause C and an annotation a.

We give the semantics of annotated clauses by interpreting clauses, histories, and annotations as formulas. If an object with an associated formula appears in the place of a formula, it is to be interpreted as its associatend formula. The **associated formula** of a clause C is $\bigwedge_{s^\sigma \in C} \lfloor s^\sigma \rfloor$. The **associated formula** of a history H is the formula $\bigwedge_{C \in H} \neg C$. The **associated formula** of an annotation is defined as follows:

$$\begin{aligned}
\mathsf{af}(\cdot) &= \top \\
\mathsf{af}(\mathsf{A}(s\,\mathsf{U}_H\,t)^+) &= \mathsf{A}((s \wedge H)\,\mathsf{U}(t \wedge H)) \\
\mathsf{af}(\mathsf{A}^+(s\,\mathsf{U}_H\,t)^+) &= \mathsf{A}^+((s \wedge H)\,\mathsf{U}(t \wedge H)) \\
\mathsf{af}(\mathsf{A}(s\,\mathsf{R}_H\,t)^-) &= \mathsf{E}((\neg s \wedge H)\,\mathsf{U}(\neg t \wedge H)) \\
\mathsf{af}(\mathsf{A}^+(s\,\mathsf{R}_H\,t)^-) &= \mathsf{E}^+((\neg s \wedge H)\,\mathsf{U}(\neg t \wedge H))
\end{aligned}$$

The meaning of an annotated eventuality can be understood as follows: a state satisfies $\mathsf{A}(s\,\mathsf{U}_H\,t)^+$ if it satisfies $\mathsf{A}(s\,\mathsf{U}\,t)$ without satisfying any clause from H along the way. For $\mathsf{A}(s\,\mathsf{R}_H\,t)^-$ we push the negation introduced by the sign down to s and t before adding the history. A state satisfies the annotated clause $C|a$, if it satisfies the formula $C \wedge a$.

The **request** of a clause is the set $\mathcal{R}\,C := \{\, s^+ \mid \mathsf{AX}\,s^+ \in C \,\}$. The request of annotations is defined such that $r\,(\mathsf{A}^+(s\,\mathsf{U}_H\,t)) = \mathsf{A}(s\,\mathsf{U}_H\,t)$ and $r\,a = \cdot$ for all other annotations. The intuition behind requests is that if a state satisfies $C|a$, then every successor state must satisfy $\mathcal{R}\,C|r\,a$.

Our tableau calculus derives unsatisfiable clauses. The rules of the calculus can be found in Figure 2. The notation C, s^σ is to be read as $C \cup \{s^\sigma\}$. If C, s^σ appears in the conclusion of a rule, we refer to C as the **context** and to s^σ as the **active formula**. The tableau system is essentially dual to the sequent calculus CT [3]. While CT derives valid disjunctions, our tableau calculus derives unsatisfiable conjunctions. Aside from syntactic changes, the main difference between CT and the tableau calculus is that in CT all the rules carry the proviso that the active formula in the conclusion does not appear in the context. We impose no such restriction. The reason for this is simply convenience. Our completeness proof does not rely on this added flexibility.

The history mechanism (last two rows in Figure 2) works by recording all contexts encountered while trying to fulfill one eventuality. If a context reappears further up in the derivation, we can close this branch since every eventuality that can be fulfilled, can be fulfilled without going through cycles. If all branches lead to cycles, the eventuality cannot be fulfilled and the clause is unsatisfiable.

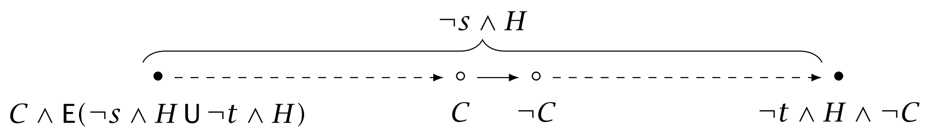
In our formalization, we do not argue soundness of the tableau system directly using models. Instead, we show the following translation theorem:

Theorem 3.1 If $C|a$ is tableau derivable, then $\vdash \neg(C \wedge a)$.

Corollary 3.2 If $C|a$ is tableau derivable, then $C|a$ is unsatisfiable.

We defer the proof of Theorem 3.1 to Section 6.

Even though it is not part of our formal development, we still argue soundness of the tableau system informally (and classically) to give some intuition how the history mechanism works. Soundness of all the rules except $\mathsf{A_H}$ and $\mathsf{R_H}$ is easy to see. The case for $\mathsf{A_H}$ is argued (in the dual form) by Brünnler and Lange [3]. So we argue soundness of $\mathsf{R_H}$ here. Assume that $C|A(s\,\mathsf{R}_H\,t)^-$ is satisfiable and $C, t^-|\cdot$ is unsatisfiable. Then the situation looks as follows:


$\neg s \wedge H$
$C \wedge \mathsf{E}(\neg s \wedge H\,\mathsf{U}\,\neg t \wedge H)$
C
$\neg C$
$\neg t \wedge H \wedge \neg C$

$$\frac{}{C, p^+, p^- | a} \qquad \frac{}{C, \bot^+ | a} \qquad \frac{C, s^- | a \qquad C, t^+ | a}{C, s \to t^+ | a}\,{\to^+} \qquad \frac{C, s^+, t^- | a}{C, s \to t^- | a}\,{\to^-}$$

$$\frac{\mathcal{R}\, C | r\, a}{C | a}\,\mathsf{X} \qquad \frac{\mathcal{R} C, u^- | r\, a}{C, \mathsf{AX}\, u^- | a}\,\mathsf{AX}^- \qquad \frac{\mathcal{R} C \,|\, \mathsf{A}(s\, \mathsf{U}_H\, t)^-}{C \,|\, \mathsf{A}^+(s\, \mathsf{U}_H\, t)^-}\,\mathsf{R}_\mathsf{H}^+$$

$$\frac{C, t^+ | a \qquad C, s^+, \mathsf{A}^+(s\, \mathsf{U}\, t)^+ | a}{C, \mathsf{A}(s\, \mathsf{U}\, t)^+ | a}\,\mathsf{U}^+ \qquad \frac{C, t^-, s^- | a \qquad C, t^-, \mathsf{A}^+(s\, \mathsf{U}\, t)^- | a}{C, \mathsf{A}(s\, \mathsf{U}\, t)^- | a}\,\mathsf{U}^-$$

$$\frac{C, s^+, t^+ | a \qquad C, t^+, \mathsf{A}^+(s\, \mathsf{R}\, t)^+ | a}{C, \mathsf{A}(s\, \mathsf{R}\, t)^+ | a}\,\mathsf{R}^+ \qquad \frac{C, t^- | a \qquad C, s^-, \mathsf{A}^+(s\, \mathsf{R}\, t)^- | a}{C, \mathsf{A}(s\, \mathsf{R}\, t)^- | a}\,\mathsf{R}^-$$

$$\frac{C \,|\, \mathsf{A}(s\, \mathsf{U}_\emptyset\, t)^+}{C, \mathsf{A}(s\, \mathsf{U}\, t)^+ \,|\, \cdot}\,\mathsf{A}_\emptyset \qquad \frac{C, t^+ \,|\, \cdot \qquad C, s^+ \,|\, \mathsf{A}^+(s\, \mathsf{U}_{H,C}\, t)^+}{C \,|\, \mathsf{A}(s\, \mathsf{U}_H\, t)^+}\,\mathsf{A}_\mathsf{H} \qquad \frac{}{C \,|\, \mathsf{A}(s\, \mathsf{U}_{H,C}\, t)^+}\,\overline{\mathsf{A}}$$

$$\frac{C \,|\, \mathsf{A}(s\, \mathsf{R}_\emptyset\, t)^-}{C, \mathsf{A}(s\, \mathsf{R}\, t)^- \,|\, \cdot}\,\mathsf{R}_\emptyset \qquad \frac{C, t^- \,|\, \cdot \qquad C, s^- \,|\, \mathsf{A}^+(s\, \mathsf{R}_{H,C}\, t)^-}{C | A(s\, \mathsf{R}_H\, t)^-}\,\mathsf{R}_\mathsf{H} \qquad \frac{}{C \,|\, \mathsf{A}(s\, \mathsf{R}_{H,C}\, t)}\,\overline{\mathsf{R}}$$

Figure 2: Tableau System for CTL

There exists some state satisfying $C \wedge \mathsf{E}(\neg s \wedge H\, \mathsf{U} \neg t \wedge H)$. Hence, there exists a path satisfying $\neg s \wedge H$ at every state until it reaches a state satisfying $\neg t \wedge H$. Since $C, t^- | \cdot$ is unsatisfiable, this state must also satisfy $\neg C$. Therefore, the path consists of at least 2 states. The last state on the path that satisfies C (left circle) also satisfies $\neg s$ and $\mathsf{E}^+((\neg s \wedge H \wedge \neg C)\, \mathsf{U}(\neg t \wedge H \wedge \neg C))$ and therefore $C, s^- \,|\, \mathsf{A}^+(s\, \mathsf{R}_{H,C}\, t)^-$.

Note that, although the R_H rule looks similar to the local rule R^-, the soundness argument is non-local; if there is state satisfying the conclusion of the rule, the state satisfying one of the premises may be arbitrarily far away in the model.

As noted by Brünnler and Lange [3], the calculus is sound for all annotated clauses but only complete for clauses with the empty annotation. Consider the clause $\emptyset \,|\, \mathsf{A}(p\, \mathsf{U}_{\{\{p^+\}\}}\, p)^+$. The clause is underivable, but its associated formula is equivalent to the unsatisfiable formula $\mathsf{A}((p \wedge \neg p)\, \mathsf{U}(p \wedge \neg p))$. To obtain a certifying decision method, completeness for history-free clauses is sufficient.

3.1 Decidability of Tableau Derivability

For our certifying decision method, we need to show that tableau derivability is decidable. The proof relies on the subformula property, i.e., the fact that backward application of the rules stays within a finite syntactic universe. We call a set of signed formulas **subformula closed**, if it satisfies the following conditions:

S1. If $(s \to t)^\sigma \in \mathcal{F}$, then $\{s^{\overline{\sigma}}, t^\sigma\} \subseteq \mathcal{F}$.

S2. If $\mathsf{AX}\, s^\sigma \in \mathcal{F}$, then $s^\sigma \in \mathcal{F}$.

S3. If $\mathsf{A}(s\,\mathsf{U}\,t)^\sigma \in \mathcal{F}$, then $\{s^\sigma, t^\sigma, \mathsf{A}^+(s\,\mathsf{U}\,t)^\sigma\} \subseteq \mathcal{F}$.

S4. If $\mathsf{A}(s\,\mathsf{R}\,t)^\sigma \in \mathcal{F}$, then $\{s^\sigma, t^\sigma, \mathsf{A}^+(s\,\mathsf{R}\,t)^\sigma\} \subseteq \mathcal{F}$.

It is easy to define a recursive function *ssub* that computes for a signed formula s^σ a finite subformula closed set containing s^σ. The **subformula closure** of a clause C is defined as $\mathit{sfc}\, C := \bigcup_{s \in C} \mathit{ssub}\, s$ and is always a subformula closed extension of C. Now let $\mathcal{F}$ be a subformula closed set. The **annotations for $\mathcal{F}$**, written $\mathcal{A}(\mathcal{F})$, consist of $\cdot$ and eventualities from $\mathcal{F}$ annotated with histories $H \subseteq \mathcal{P}(\mathcal{F})$, where $\mathcal{P}(\mathcal{F})$ is the powerset of $\mathcal{F}$. We define the **universe for $\mathcal{F}$** as $\mathcal{U}(\mathcal{F}) := \mathcal{P}(\mathcal{F}) \times \mathcal{A}(\mathcal{F})$.

Lemma 3.3 1. If $\mathcal{F}$ is subformula closed, the set $\mathcal{U}(\mathcal{F})$ is closed under backward application of the tableau rules.

2. For every annotated clause $C|a$ there exists a subformula closed set $\mathcal{F}$, such that $C|a \in \mathcal{U}(\mathcal{F})$.

3. Derivability of annotated clauses is decidable.

Proof Claim (1) follows by inspection of the individual rules. For (2) we reason as follows: If $a = \cdot$, we take $\mathcal{F}$ to be $\mathit{sfc}\, C$. If $a = \mathsf{A}(s\,\mathsf{U}_H\,t)$, one can show that $C|\,\mathsf{A}(s\,\mathsf{U}_H\,t) \in \mathcal{U}(\mathit{sfc}\,(C, \mathsf{A}(s\,\mathsf{U}\,t) \cup \bigcup_{D \in H} D))$. All other cases are similar.

For (3) consider the annotated clause $C|a$. By (2) we know that $C|a \in \mathcal{U}(\mathcal{F})$ for some $\mathcal{F}$. We now compute the least fixpoint of one-step tableau derivability inside $\mathcal{U}(\mathcal{F})$. By (1) the annotated clause $C|a$ is derivable iff it is contained in the fixpoint.

3.2 Finite Sets in Coq

To formalize the tableau calculus and the decidability proof, we need to formalize clauses and histories. The Ssreflect libraries [8] contain a library for finite sets. However, the type of sets defined there requires that the type over which the sets are formed is a finite type, i.e., a type with finitely many elements. This is clearly not the case for the type of signed formulas.

We want a library providing extensional finite sets over countable types (e.g., signed formulas) providing all the usual operations including separation ($\{\, x \in$

$A \mid p\,x\,\}$), replacement ($\{\,f\,x \mid x \in A\,\}$), and powerset. We could not find a library satisfying all our needs, so we developed our own.

Our set type is a constructive quotient over lists. We use the choice operator provided by Ssreflect to define a normalization function that picks some canonical duplicate-free list to represent a given set. This normalization function is the main primitive for constructing sets. On top of this we build a library providing all the required operations. Our lemmas and notations are inspired by Ssreflect's finite sets and we port most of the lemmas that apply to the setting with infinite base types. We instantiate Ssreflect's big operator library [2], which provides us with indexed unions.

Our library also contains a least fixpoint construction. For every bounded monotone function from sets to sets we construct its least fixpoint and show the associated induction principle. This is used in the formalization of Lemma 3.3 to compute the set of derivable clauses over a given subformula universe.

4 Demos

We now define demos. In the completeness proof of the tableau calculus, demos serve as the interface between the model construction and the tableau system. Our demos are a variant of the pseudo-Hintikka structures used by Emerson [5]. Instead of Hintikka clauses, we use literal clauses and the notion of support [10].

A signed formula is a **literal**, if it is of the form p^σ, $\bot^\sigma$, or $\mathsf{AX}\,s^\sigma$. A **literal clause** is a clause containing only literals. A literal clause is **locally consistent** if it contains neither $\bot^+$ nor both p^+ and p^- for any p. A clause **supports** a signed formula, written $C \rhd s^\sigma$, if

$$\begin{array}{rcll}
C \rhd l & \iff & l \in C & \text{if } l \text{ is a literal} \\
C \rhd (s \to t)^+ & \iff & C \rhd s^- \ \vee\ C \rhd t^+ & \\
C \rhd (s \to t)^- & \iff & C \rhd s^+ \ \wedge\ C \rhd t^- & \\
C \rhd \mathsf{A}(s\,\mathsf{U}\,t)^+ & \iff & C \rhd t^+ \ \vee\ (C \rhd s^+ \ \wedge\ C \rhd \mathsf{A}^+(s\,\mathsf{U}\,t)^+) & \\
C \rhd \mathsf{A}(s\,\mathsf{U}\,t)^- & \iff & C \rhd t^- \ \wedge\ (C \rhd s^- \ \vee\ C \rhd \mathsf{A}^+(s\,\mathsf{U}\,t)^-) & \\
C \rhd \mathsf{A}(s\,\mathsf{R}\,t)^+ & \iff & C \rhd t^+ \ \wedge\ (C \rhd s^+ \ \vee\ C \rhd \mathsf{A}^+(s\,\mathsf{R}\,t)^+) & \\
C \rhd \mathsf{A}(s\,\mathsf{R}\,t)^- & \iff & C \rhd t^- \ \vee\ (C \rhd s^- \ \wedge\ C \rhd \mathsf{A}^+(s\,\mathsf{R}\,t)^-) &
\end{array}$$

We define $C \rhd D := \forall s^\sigma \in D.\ C \rhd s^\sigma$.

A **fragment** is a finite, rooted, and acyclic directed graph labeled with clauses. If G is a fragment, we write $x \in G$ to say that x is a node of G and $x \to_G y$ if there is a G-edge from x to y. A node $x \in G$ is **internal** if it has some successor and a **leaf** otherwise. If $x \in G$, we write Λ_x for the literal clause labeling x.

We also write x_{root} for the root of a graph if the graph can be inferred from the context. A fragment is **nontrival** if its root is not a leaf.

We fix some subformula closed set $\mathcal{F}$ for the rest of this section. and write $\mathcal{L}$ for the set of locally consistent literal clauses over $\mathcal{F}$. We also fix some set $\mathcal{D} \subseteq \mathcal{L}$. Let $L \in \mathcal{D}$ be a clause. A fragment G is a **$\mathcal{D}$-fragment for L** if:

F1. If $x \in G$ is a leaf, then $\Lambda_x \in \mathcal{D}$ and $\Lambda_x \in \mathcal{L}$ otherwise.

F2. The root of G is labeled with L.

F3. If $x \to_G y$, then $\Lambda_y \rhd \mathcal{R}(\Lambda_x)$.

F4. If $x \in G$ is internal and $\mathsf{AX}\,s^- \in \Lambda_x$, then $x \to_G y$ and $\Lambda_y \rhd \mathcal{R}(\Lambda_x), s^-$ for some $y \in G$.

A $\mathcal{D}$-fragment G for L is a **$\mathcal{D}$-fragment for L and u** if whenever $L \rhd u$ then:

E1. If $u = \mathsf{A}(s\,\mathsf{U}\,t)^+$, then $L \rhd t^+$ or $\Lambda_x \rhd s^+$ for every internal $x \in G$ and $\Lambda_y \rhd t^+$ for all leaves $y \in G$.

E2. If $u = \mathsf{A}(s\,\mathsf{R}\,t)^-$, then $L \rhd t^-$ or $\Lambda_x \rhd s^-$ every internal $x \in G$ and $\Lambda_y \rhd t^-$ for some $y \in G$.

Note that if u is an eventuality and $L \rhd u$, then u is fulfilled in every $\mathcal{D}$-fragment for L and u. The conditions $L \rhd t^+$ in (E1) and $L \rhd t^-$ in (E2) are required to handle the case of an eventuality that is fulfilled in L and allow for the construction of nontrivial fragments in this case. A **demo** for $\mathcal{D}$ is an indexed collection of nontrivial fragments $(G(u,L))_{u \in \mathcal{F}, L \in \mathcal{D}}$ where each $G(u,L)$ is a $\mathcal{D}$-fragment for L and u.

4.1 Demos to Finite Models

Assume that we are given some demo $(G(u,L))_{u \in \mathcal{F}, L \in \mathcal{D}}$. We construct a model $\mathcal{M}$ satisfying all labels occurring in the demo. If $\mathcal{F}$ is empty, there is nothing to show, so we can assume that $\mathcal{F}$ is nonempty.

The states of $\mathcal{M}$ are the nodes of all the fragments in the demo, i.e., every state of $\mathcal{M}$ is a dependent triple (u, L, x) with $u \in \mathcal{F}$, $L \in \mathcal{D}$, and $x \in G(u,L)$. A state (u,L,x) is labeled with atomic proposition p iff $p^+ \in \Lambda_x$.

To define the transitions of $\mathcal{M}$, we fix an ordering $u_0, \dots, u_n$ of the signed formulas in $\mathcal{F}$. We write u_{i+1} for the successor of u_i in this ordering. The successor of u_n is taken to be u_0. The transitions of $\mathcal{M}$ are of two types. First, we lift all the internal edges of the various fragments to transitions in $\mathcal{M}$. Second, if x is a leaf in $G(u_i, L_j)$ that is labeled with L, we add transitions from (u_i, L_j, x) to all successors of the root of $G(u_{i+1}, L)$. Thus, the fragments in the demo can be thought of as arranged in a matrix as shown in Figure 3 where the L_i are the clauses in $\mathcal{D}$. Note that every root has at least one successor, since demos contain only nontrivial fragments. Thus, the resulting transition system is serial

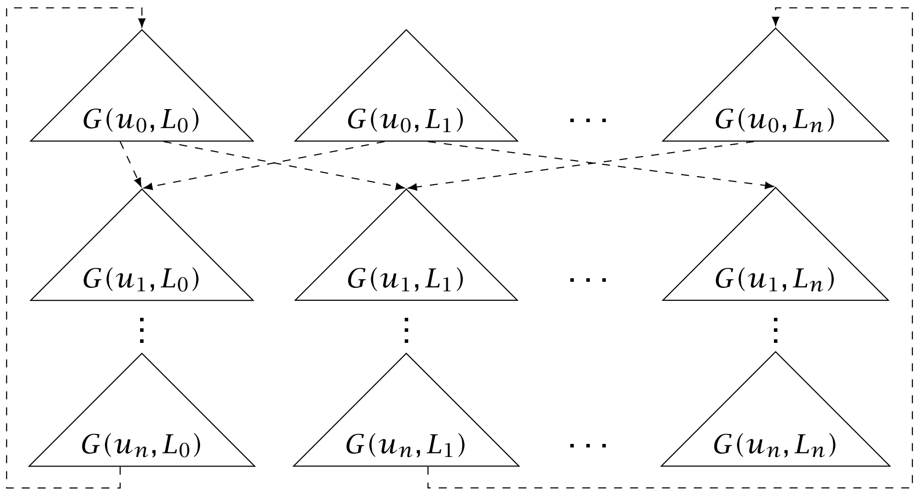


Figure 3: Matrix of Fragments

and hence a model. We then show that every state of $\mathcal{M}$ satisfies all signed formulas it supports.

Lemma 4.1 If $(u, L, x) \in \mathcal{M}$ and $\Lambda_x \rhd s^\sigma$, then $(u, L, x) \vDash \lfloor s^\sigma \rfloor$.

Proof The proof goes by induction on s. We sketch the case for $A(s \,\mathsf{U}\, t)^+$. The case for $\mathsf{A}(s \,\mathsf{R}\, t)^-$ is similar and all other cases are straightforward.

Let $w = (u_i, L_j, x) \in \mathcal{M}$ and assume $\Lambda_x \rhd \mathsf{A}(s \,\mathsf{U}\, t)^+$. By induction hypothesis it suffices to show $\mathsf{AU}_\mathcal{M}\, s\, t\, w$ where

$$\mathsf{AU}_\mathcal{M}\, s\, t\, w := \mathsf{AU}\, (\rightarrow_\mathcal{M})\, (\lambda(\_,\_,y).\Lambda_y \rhd s^+)\, (\lambda(\_,\_,y).\Lambda_y \rhd t^+)\, w$$

To show $\mathsf{AU}_\mathcal{M}\, s\, t\, w$ it suffices to show $\mathsf{AU}_\mathcal{M}\, s\, t\, (u_{i+1}, L, x_{root})$ for all L satisfying $L \rhd \mathsf{A}(s \,\mathsf{U}\, t)^+$ since by (F3) the property of supporting $\mathsf{A}(s \,\mathsf{U}\, t)^+$ gets propagated down to the leaves of $G(u_i, L_j)$ on all paths that do not support t^+ along the way.

Without loss of generality, we can assume $\mathsf{A}(s \,\mathsf{U}\, t)^+ \in \mathcal{F}$. Thus, we can prove $\mathsf{AU}_\mathcal{M}\, s\, t\, (u_{i+1}, L, x_{root})$ by induction on the distance from u_{i+1} to $\mathsf{A}(s \,\mathsf{U}\, t)^+$ according to the ordering of $\mathcal{F}$. If $u_{i+1} = \mathsf{A}(s \,\mathsf{U}\, t)^+$, we have $\mathsf{AU}_\mathcal{M}\, s\, t\, (u_{i+1}, L, x_{root})$ by (E1). Otherwise, the claim follows by induction, deferring to the next row of the matrix as we did above.

Theorem 4.2 If $(G_{u,L})_{u \in F, L \in \mathcal{D}}$ is a demo for $\mathcal{D}$, there exists a finite model satisfying every label occurring in $(G_{u,L})_{u \in F, L \in \mathcal{D}}$.

4.2 Formalizing the Model Construction

Our representation of fragments is based on finite types. We represent finite labeled graphs as relations over some finite type together with a labeling function. We then represent fragments using clause labeled graphs with a distinguished root element.

We turn the finite set $\mathcal{F} \times \mathcal{D}$ into a finite type I. Except for the transitions connecting the leaves of one row to the next row, the model is then just the disjoint union of a collection of graphs indexed by I. Let $G : I \to \mathit{graph}$ be such a collection. We lift the internal edges of G by defining a predicate

$$\mathit{liftEdge} \;:\; (\Sigma i{:}I.\, G\, i) \to (\Sigma i{:}I.\, G\, i) \to \mathit{bool}$$

on the dependent pairs of an index and a node of the respective graph satisfying

$$\mathit{liftEdge}\,(i,x)\,(i,y) \iff x \to_{G\,i} y$$
$$i \neq j \implies \neg\mathit{liftEdge}\,(i,x)\,(j,y)$$

The definition of *liftEdge* uses dependent types in a form that is well supported by Ssreflect.

Our model construction differs slightly from the construction used by Emerson and Halpern [5]. In Emerson's handbook article, every leaf of a fragment is replaced by the root with the same label on the next level. Thus, only the internal nodes of every fragment become states of the model. This would amount to using a Σ-type on the vertex type of every dag. In our model construction, we connect the leaves of one row to the successors of the equally labeled root of the next row, thus, avoiding a Σ-type construction. This makes use of the fact that CTL formulas cannot distinguish different states that have the same labels and the same set of successors.

5 Tableaux to Demos

An annotated clause is **consistent** if it is not derivable, and a clause C is consistent if $C|\cdot$ is consistent. Let $\mathcal{F}$ be a subformula closed set. We now construct a demo for the consistent literal clauses over $\mathcal{F}$. We define

$$\mathcal{D} := \{\, L \subseteq \mathcal{F} \mid L \text{ consistent}, L \text{ literal} \,\}$$

We now have to construct for every pair $(u, L) \in \mathcal{F} \times \mathcal{D}$ a nontrivial $\mathcal{D}$-fragment for L and u. We will construct a demo, where all the fragments are trees. To bridge the gap between the tableau, which works over arbitrary clauses, and $\mathcal{D}$-fragments, which are labeled with literal clauses only, we need the following lemma:

Lemma 5.1 If $C|a \in \mathcal{U}(\mathcal{F})$ is consistent, we can construct a literal clause $L \subseteq \mathcal{F}$ such that $L \rhd C$ and $L|a$ is consistent.

Proof The proof proceeds by induction on the total size of the non-literal formulas in C. If this total size is 0, then C is a literal clause and there is nothing to show. Otherwise there exists some non-literal formula $u^\sigma \in C$. Thus $C|a = C \setminus \{u^\sigma\}, u^\sigma|a$ and we can apply the local rule for u^σ. Consider the case where $u^\sigma = s \to t^+$. By rule $\to^+$ we know that $C \setminus \{s \to t^+\}, s^-|a$ or $C \setminus \{s \to t^+\}, t^+$ is consistent. In either case we obtain a literal clause L supporting C by induction hypothesis. The other cases are similar. ∎

Before we construct the fragments, we need one more auxiliary definition

$$\mathcal{R}^- C := \mathcal{R}C, \{\, \mathcal{R}C, s^- \mid \mathsf{AX}\, s^- \in C \,\}$$

The set of clauses $\mathcal{R}^- C$ serves the dual purpose of the request $\mathcal{R}C$. It contains all the clauses that must be supported at the successors of C to satisfy (F4).

The demo for $\mathcal{D}$ consists of three kinds of fragments. The easiest fragments are those for a pair (u, L) where $L \not\rhd u$ or u is not an eventuality. In this case, a $\mathcal{D}$-fragment for L is also a $\mathcal{D}$-fragment for L and u.

Lemma 5.2 If $L \in \mathcal{D}$, we can construct a nontrivial $\mathcal{D}$-fragment for L.

Proof By assumption $L|\cdot$ is consistent. According to rules X and AX^-, $C|\cdot$ is consistent for every clause $C \in R^- L$. Note that there is at least one such clause. By Lemma 5.1, we can obtain for every $C \in \mathcal{R}^- L$ some clause $L_C \in \mathcal{D}$. The $\mathcal{D}$-fragment for L consists of a single root labeled with L and one successor labeled with L_C for every $C \in \mathcal{R}^- L$.

Next, we deal with the case of a pair $(\mathsf{A}(s \,\mathsf{U}\, t)^+, L)$ where $L \rhd \mathsf{A}(s \,\mathsf{U}\, t)^+$. This is the place where we make use of the history annotations.

Lemma 5.3 If $C|A(s \,\mathsf{U}_H\, t)^+ \in \mathcal{U}(\mathcal{F})$ is consistent, we can construct a $\mathcal{D}$-fragment G for L such that $\Lambda_x \rhd s^+$ for every internal node $x \in G$ and $\Lambda_y \rhd t^+$ for all leaves $y \in G$ where L is some clause supporting $C, \mathsf{A}(s \,\mathsf{U}\, t)^+$.

Proof Induction on the slack of H, i.e., the number of clauses from $\mathcal{P}(\mathcal{F})$ that are not in H. Since $C|A(s \,\mathsf{U}_H\, t)^+$ is consistent, we know $C \notin H$. According to rule $\mathsf{A_H}$, there are two cases to consider:

- $C, t^+|\cdot \in \mathcal{U}(\mathcal{F})$ is consistent: By Lemma 5.1 we obtain a literal clause L such that $L \rhd C, t^+$ and $L|\cdot$ is consistent. The trivial fragment with a single node labeled with L satisfies all required properties.

- $C, s^+ | A^+(s \,\mathsf{U}_{H,C}\, t)^+ \in \mathcal{U}(\mathcal{F})$ is consistent: By Lemma 5.1 we obtain a literal clause L such that $L \rhd C, s^+$ and $L | A^+(s \,\mathsf{U}_{H,C}\, t)^+$ is consistent. In particular, $L, \mathsf{A}^+(s \,\mathsf{U}\, t)^+$ is locally consistent and supports C as well as $\mathsf{A}(s \,\mathsf{U}\, t)^+$. By induction hypothesis, we obtain a $\mathcal{D}$-fragment for every clause in $\mathcal{R}^- L$. Putting everything together, we obtain a $\mathcal{D}$-fragment for $L, \mathsf{A}^+(s \,\mathsf{U}\, t)^+$ satisfying all required properties ■

Lemma 5.4 If $L \in \mathcal{D}$, we can construct a nontrivial $\mathcal{D}$-fragment for L and $\mathsf{A}(s \,\mathsf{U}\, t)^+$.

Proof Without loss of generality we can assume that $L \rhd s^+$ and $\mathsf{A}^+(s \,\mathsf{U}\, t)^+ \in L$. All other cases are covered by Lemma 5.2. Using rules X, AX^-, and $\mathsf{A}_\emptyset$, we show for every $C \in \mathcal{R}^- L$ that $C | \mathsf{A}(s \,\mathsf{U}_\emptyset\, t)^+$ is a consistent clause in $\mathcal{U}(\mathcal{F})$. By Lemma 5.3 we obtain a $\mathcal{D}$-fragment for every such clause. Putting a root labeled with L on top as in the proof of Lemma 5.2, we obtain a nontrivial $\mathcal{D}$-fragment for L and $\mathsf{A}(s \,\mathsf{U}\, t)^+$ as required.

Lemma 5.5 If $L \in \mathcal{D}$, we can construct a nontrivial $\mathcal{D}$-fragment for L and $\mathsf{A}(s \,\mathsf{R}\, t)^-$.

The proof of Lemma 5.5 is similar to the proof of Lemma 5.4 and uses a similar auxiliary lemma.

Theorem 5.6 1. There exists a $\mathcal{D}$-demo.

2. If $C | \cdot$ is consistent, then $w \vDash C$ for some finite model $\mathcal{M}$ with $w \in \mathcal{M}$.

Note that by Theorem 4.2 all the locally consistent labels of the internal nodes of the constructed fragments are satisfiable and hence must be consistent. However, at the point in the proof of Lemma 5.3 where we need to show local consistency of $L, \mathsf{A}^+(s \,\mathsf{U}\, t)$ from consistency of $L | \mathsf{A}^+(s \,\mathsf{U}_H\, t)$ showing local consistency is all we can do.

All fragments constructed in this section are trees. In the formalization, we state the lemmas from this section using an inductively defined tree type, leaving the sets of nodes and edges implicit. Thus, trees can be composed without updating an edge relation or changing the type of vertices. Even using this tailored representation, the formalization of Lemma 5.3 is one of the most complex parts of our development.

For Theorem 5.6, we convert the constructed trees to rooted dags. To convert a tree T to a dag, we turn the list of subtrees of T into a finite type and use this as the type of vertices. We then add edges from every tree to its immediate subtrees. This construction preserves all fragment properties even though identical subtrees of T are collapsed in into a single vertex.

A1	$\vdash A(s\,\mathsf{U}\,t) \leftrightarrow t \vee s \wedge A^+(s\,\mathsf{U}\,t)$
A2	$\vdash \mathsf{EG}\,\neg t \to \neg A(s\,\mathsf{U}\,t)$
A3	$\vdash \mathsf{A}((s \wedge u)\,\mathsf{U}(t \wedge u)) \to u$
E1	$\vdash t \to \mathsf{E}(s\,\mathsf{U}\,t)$
E2	$\vdash s \to \mathsf{E}^+(s\,\mathsf{U}\,t) \to \mathsf{E}(s\,\mathsf{U}\,t)$
AE	$\vdash \mathsf{AX}\,s \to \mathsf{EX}\,t \to \mathsf{EX}(s \wedge t)$
$\mathsf{EU_{ind}}$	If $\vdash t \to u$ and $\vdash s \to \mathsf{EX}\,u \to u$, then $\vdash \mathsf{E}(s\,\mathsf{U}\,t) \to u$
$\mathsf{EG_{ind}}$	If $\vdash u \to s$ and $\vdash u \to \mathsf{EX}\,u$, then $\vdash u \to \mathsf{EG}\,s$

Figure 4: Basic CTL Lemmas

6 Tableau Refutations to Hilbert Refutations

We now return to the proof of Theorem 3.1. For this proof, we will translate the rules of the tableau calculus to lemmas in the Hilbert calculus. For this we need a number of basic CTL lemmas. The lemmas to which we will refer explicitly can be found in Figure 4. In formulas, we let $s\,\mathsf{U}_H\,t$ abbreviate $(s \wedge H)\,\mathsf{U}(t \wedge H)$.

We present the translation lemmas for the rules $\mathsf{A_H}$ and $\mathsf{R_H}$. Given the non-local soundness argument sketched in Section 3, it should not come as a surprise that the translation of both rules requires the use of the corresponding induction rule from the Hilbert axiomatization. For both lemmas we use the respective induction rule in dualized form as shown in Figure 4.

Lemma 6.1 If $\vdash t \to \neg C$ and $\vdash \mathsf{E}^+(s\,\mathsf{U}_{H,C}\,t) \to s \to \neg C$,then $\vdash \neg(C \wedge \mathsf{E}(s\,\mathsf{U}_H\,t))$.

Proof Assume we have (a) $\vdash t \to \neg C$ and (b) $\vdash \mathsf{E}^+(s\,\mathsf{U}_{H,C}\,t) \to s \to \neg C$. By propositional reasoning, it suffices to show

$$\vdash \mathsf{E}(s\,\mathsf{U}_H\,t) \to \neg C \wedge \mathsf{E}(s\,\mathsf{U}_{H,C}\,t)$$

Applying the $\mathsf{EU_{ind}}$ rule leaves us with two things to prove. The first one is $\vdash t \wedge H \to \neg C \wedge E(s\,\mathsf{U}_{H,C}\,t)$ and can be shown using (a) and E1. The other is

$$\vdash s \wedge H \to \mathsf{EX}(\neg C \wedge \mathsf{E}(s\,\mathsf{U}_{H,C}\,t)) \to \neg C \wedge \mathsf{E}(s\,\mathsf{U}_{H,C}\,t)$$

The second assumption can be weakened to $\mathsf{E}^+(s\,\mathsf{U}_{H,C}\,t)$. Thus, we also have $\neg C$ by assumption (b). Finally, we obtain $\mathsf{E}(s\,\mathsf{U}_{H,C}\,t)$ using Lemma E2.

Lemma 6.2 If $\vdash C \to \neg t$ and $\vdash C \to s \to \neg A^+(s\,\mathsf{U}_{H,C}\,t)$, then $\vdash \neg(C \wedge \mathsf{A}(s\,\mathsf{U}_H\,t))$.

Proof Assume we have (a) $\vdash C \to \neg t$ and (b) $\vdash C \to s \to \neg A^+(s\,\mathsf{U}_{H,C}\,t)$. We set $u := \neg t \wedge \mathsf{A}^+(s\,\mathsf{U}_H\,t) \wedge \neg\,\mathsf{A}^+(s\,\mathsf{U}_{H,C}\,t)$. We first argue that it suffices to show (1) $\vdash u \to \mathsf{EG}\,\neg t$. Assume we have C and $\mathsf{A}(s\,\mathsf{U}_H\,t)$. By (a) we also know $\neg t$ and thus we have $s \wedge H$ and $\mathsf{A}^+(s\,\mathsf{U}_H\,t)$ by $\mathsf{A1}$. Using (b) and (1), we obtain $\mathsf{EG}\,\neg t$ which contradicts $\mathsf{A}(s\,\mathsf{U}_H\,t)$ according to $\mathsf{A2}$.

We show (1) using the $\mathsf{EG_{ind}}$ rule. Showing $\vdash u \to \neg t$ is trivial so it remains to show $\vdash u \to \mathsf{EX}\,u$. Assume u. By Lemma AE we have $\mathsf{EX}(\mathsf{A}(s\,\mathsf{U}_H\,t) \wedge \neg A(s\,\mathsf{U}_{H,C}\,t))$. It remains to show

$$\vdash \mathsf{A}(s\,\mathsf{U}_H\,t) \wedge \neg A(s\,\mathsf{U}_{H,C}\,t) \to u$$

We reason as follows:

1.	$\mathsf{A}(s\,\mathsf{U}_H\,t)$	assumption
2.	$\neg\,\mathsf{A}(s\,\mathsf{U}_{H,C}\,t)$	assumption
3.	$\neg t \vee \neg H \vee C$	2, A1
4.	$\neg s \vee \neg H \vee C \vee \neg\,\mathsf{A}^+(s\,\mathsf{U}_{H,C}\,t)$	2, A1
5.	H	1,A3
6.	$\underline{\neg t}$	3, 5, (a)
7.	$s \wedge \underline{\mathsf{A}^+(s\,\mathsf{U}_H\,t)}$	1,6,A1
8.	$\underline{\neg\,\mathsf{A}^+(s\,\mathsf{U}_{H,C}\,t)}$	4, 5, 7, (b)

This finishes the proof.

Proof (of Theorem 3.1) Let $C|a$ be derivable. We prove the claim by induction on the derivation of $C|a$. All cases except those for the rules $\mathsf{R_H}$ and $\mathsf{A_H}$ are straightforward. The former follows with Lemma 6.1 the latter with Lemma 6.2.

To formalize this kind of translation argument, we need some infrastructure for assembling Hilbert proofs as finding proofs in the bare Hilbert system can be a difficult task. We extend the infrastructure we used in our previous work [4] to CTL. We use conjunctions over lists of formulas to simulate context. We also use Coq's generalized (setoid) rewriting [12] with the preorder $\{\,(s,t) \mid\ \vdash s \to t\,\}$.

Putting our results together we obtain a certifying decision method for CTL.

Proof (of Theorem 2.2) By Lemma 3.3, derivability of the clause $s^+|\cdot$ is decidable. If $s^+|\cdot$ is derivable we obtain a proof of $\neg s$ with Theorem 3.1. Otherwise, we obtain a finite model satisfying s with Theorem 5.6 and Theorem 4.2. By Theorem 2.1, the two results are mutually exclusive.

Corollary 6.3 (Decidability) Satisfiability of formulas is decidable.

Corollary 6.4 (Completeness) If $\forall \mathcal{M}.\forall w \in \mathcal{M}.w \models s$, then $\vdash s$.

7 Conclusion

Our completeness proof for the tableau calculus differs considerably from the corresponding completeness proof for the sequent system given by Brünnler and Lange [3]. Their proof works by proving the completeness of another more restrictive sequent calculus which is ad-hoc in the sense that it features a rule whose applicability is only defined for backward proof search. We simplify the proof by working directly with the tableau rules and by using the model construction of Emerson [5].

The proof of Theorem 3.1 relies on the ability to express the semantics of history annotations in terms of formulas. This allows us to show the soundness of the tableau calculus by translating the tableau derivations in a compositional way. While this works well for CTL, this is not necessarily the case for other modal logics. As observed previously [4], the tableau system for CTL can be adapted to modal logic with transitive closure (K^+). However, K^+ cannot express the “until” operator used in the semantics of annotated eventualities. It therefore appears unlikely that the individual rules of a history-augmented tableau system for K^+ can be translated one by one to the Hilbert axiomatization. The tableau system we used to obtain a certifying decision method for K^+ [4] uses a complex compound rule instead of the more fine-grained history annotations employed here. Thus, even though the logic CTL is more expressive than K^+, the results presented here do not subsume our previous results.

An alternative to our construction of a certifying decision method could be to replace the tableau calculus with a pruning-based decision procedure like the one described by Emerson and Halpern [7]. In fact Emerson and Halpern’s completeness proof for their Hilbert axiomatization of CTL is based on this algorithm. While their proof is non-constructive, we believe that it can be transformed into a constructive proof. In any case, we believe that the formal analysis of our history-augmented tableau calculus is interesting in its own right.

The proofs we present involve a fair amount of detail, most of which is omitted in the paper for reasons of space. Having a formalization thus not only ensures that the proofs are indeed correct, but also gives the reader the possibility to look at the omitted details.

For our formal development, we profit much from Ssreflect’s handling of countable and finite types. Countable types form the basis for our set library and finite types are used heavily when we assemble the fragments of a demo into a finite model. Altogether our formalization consists of roughly 3500 lines. The included set library consists of about 700 lines, the remaining lines are split almost evenly over the proofs of Theorems 3.1, 5.6, and 4.2 and the rest of the development.

References

[1] Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press (2008)

[2] Bertot, Y., Gonthier, G., Biha, S.O., Pasca, I.: Canonical big operators. In: Mohamed, O.A., Muñoz, C., Tahar, S. (eds.) TPHOLs. LNCS, vol. 5170, pp. 86–101. Springer (2008)

[3] Brünnler, K., Lange, M.: Cut-free sequent systems for temporal logic. J. Log. Algebr. Program. 76(2), 216–225 (2008)

[4] Doczkal, C., Smolka, G.: Constructive completeness for modal logic with transitive closure. In: Hawblitzel, C., Miller, D. (eds.) CPP 2012. LNCS, vol. 7679, pp. 224–239. Springer (2012)

[5] Emerson, E.A.: Temporal and modal logic. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science: Formal Models and Sematics, vol. B, pp. 995–1072. Elsevier (1990)

[6] Emerson, E.A., Clarke, E.M.: Using branching time temporal logic to synthesize synchronization skeletons. Sci. Comput. Programming 2(3), 241–266 (1982)

[7] Emerson, E.A., Halpern, J.Y.: Decision procedures and expressiveness in the temporal logic of branching time. J. Comput. System Sci. 30(1), 1–24 (1985)

[8] Gonthier, G., Mahboubi, A., Rideau, L., Tassi, E., Théry, L.: A modular formalisation of finite group theory. In: Schneider, K., Brandt, J. (eds.) TPHOLs. LNCS, vol. 4732, pp. 86–101. Springer (2007)

[9] Gonthier, G., Mahboubi, A., Tassi, E.: A small scale reflection extension for the Coq system. Research Report RR-6455, INRIA (2008), `http://hal.inria.fr/inria-00258384/en/`

[10] Kaminski, M., Smolka, G.: Terminating tableaux for hybrid logic with eventualities. In: Giesl, J., Hähnle, R. (eds.) IJCAR 2010. LNCS, vol. 6173, pp. 240–254. Springer (2010)

[11] Smullyan, R.M.: First-Order Logic. Springer (1968)

[12] Sozeau, M.: A new look at generalized rewriting in type theory. Journal of Formalized Reasoning 2(1) (2009)

[13] The Coq Development Team: `http://coq.inria.fr`