Completeness and Decidability Results for CTL in Constructive Type Theory

Christian Doczkal Gert Smolka

March 3, 2016

Saarland University

Published in J. Autom. Reason., ITP 2014 Special Issue, DOI:10.1007/s10817-016-9361-9

We prove completeness and decidability results for the temporal logic CTL in Coq/Ssreflect. Our main 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 small model property of CTL and completeness of the Hilbert system follow as corollaries. Our proofs mostly refine the mathematical proofs given by Emerson and Halpern. One important deviation is our use of an inductive semantics for CTL to avoid reasoning about infinite paths. On finite models the inductive semantics agrees constructively with the standard path semantics. The proof amounts to the verification of a simple model checking algorithm. For general models, the agreement between the inductive semantics and the path semantics requires excluded middle and dependent choice.

1 Introduction

Computation tree logic (CTL) [11] is a temporal logic widely used in program verification [9]. The model checking problem for CTL is decidable in polynomial time [5, 13]. The logic has the small model property and satisfiability of formulas is EXPTIME-complete [12, 8]. There are several complete Hilbert axiomatizations for CTL in the literature. In addition to the original axiomatizations by Emerson and Halpern [12, 8] there is an axiomatization by Lange and Stirling [23] derived from a game-theoretic interpretation of CTL. The game-theoretic approach also led to the development of a cut-free sequent calculus for CTL [4].

We are interested in a formal and constructive metatheory of CTL. We define formulas, models, a satisfaction relation relating models and formulas, and a Hilbert proof system. Our main 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 and the decidability of satisfiability and provability (using soundness of the Hilbert system). Obtaining formal and constructive proofs for these metatheoretic results requires a careful reengineering of the original proofs.

In addition to the usual path semantics, we define an equivalent inductive semantics for CTL. The inductive semantics is inspired by the fixpoint characterization [10, 9] of path formulas used for model checking [13, 9]. For most of our development, we work with the inductive semantics. This avoids reasoning about infinite paths and leads to more natural proofs.

The rules of our Hilbert system directly correspond to the inductive semantics. We constructively prove the Hilbert system sound for the usual path semantics on finite models. The main issue is showing soundness of the double negation axiom. We obtain a constructive soundness proof by showing that on finite models the inductive semantics is decidable and agrees with the path semantics. This amounts to the verification of a simple model checking algorithm.

If we assume excluded middle and a weak form of choice, the soundness argument extends to the class of all models. Combined with the constructive completeness result, this establishes the small model property of CTL.

The original completeness proofs for Hilbert axiomatizations of CTL are of considerable complexity [12, 8]. One reason for this complexity is the non-compactness of the logic, caused by the ability to express transitive closure. This means that the Lindenbaum construction [16], which uses maximally consistent sets, does not extend to CTL. Moreover, the Fischer-Ladner quotient construction for PDL [15] does not extend to CTL because it does not preserve satisfaction of always-until formulas. This complicates the construction of finite models for CTL. The original completeness proofs for CTL [12, 8] are based on a decision procedure employing Pratt-style pruning [24, 20]. With some reengineering, the original proofs can be transformed into constructive proofs.

The overall structure of our completeness proof follows Emerson's handbook article [8]. We adapt the proofs to better match the inductive semantics of CTL. There are two subtasks of considerable complexity. One complex subtask is the construction of finite models from syntactic pseudo-models we call demos. The other complex subtask is the construction of Hilbert refutations for formulas not satisfied by any demo.

Our demos play the role of the pseudo-Hintikka structures employed by Emerson [8]. Demos sit at the heart of our decidability result and are designed to give

a good compromise between minimizing the effort for the model construction and minimizing the effort of constructing Hilbert refutations. We employ literal clauses and the notion of support [21] instead of Hintikka sets. We handle eventualities (i.e., until formulas) using inductively defined fulfillment predicates instead of the embedded fragments used by Emerson.

For the construction of finite models from demos, we unfold demos into fragments such that each fragment fulfills one eventuality. Following Emerson [8], we assemble the different fragments into a model.

For the construction of Hilbert refutations, we replace Emerson's [8] classical argument with a constructive one. The original proof is non-constructive in the sense that it assumes logical decidability of Hilbert provability to show completeness. While Hilbert provability is computationally decidable, and hence also logically decidable, the easiest way to show this is through completeness.

Our construction of Hilbert refutations is inspired by the work of Ben-Ari et al. [2], where a constructive proof is sketched for UB, a fragment of CTL. The construction of Hilbert derivations is of considerable complexity, in particular as it comes to the instantiation of the induction rules of the Hilbert system. Our induction invariants are adapted from Emerson [8].

Given the practical importance of CTL and the complexity of the proofs of the metatheoretic results for CTL, we think that the metatheory of CTL is an interesting and rewarding candidate for formalization. The original proofs [12, 8] are presented in a fairly informal manner. The gap between the informal proofs and our formalization is considerable, in particular as it comes to the generation of Hilbert refutations.

Our development [6] is carried out in Coq [27] with the Ssreflect [18] extension. It includes a library for finite sets over Ssreflect's countable types, which is used to formalize demos, pruning, and the model construction. In total, the development consists of about 3500 lines.

In previous work [7], we prove completeness of the Hilbert system for CTL using a variant of Brünnler and Lange's [4] sequent calculus as the underlying decision method. In the present work we follow Emerson [8] and base our completeness results on Pratt-style pruning [24, 20]. This simplifies the model construction and leads to a more natural construction of the required Hilbert refutations.

The paper is organized as follows: In Section 2 we recall the syntax and semantics of CTL and present the Hilbert system. Section 3 describes the inductive semantics we use in our proofs. Section 4 contains a brief description of the finite set library underlying the formal development. In Section 5 we prove the Hilbert system sound for finite models. In Section 6 we define clauses and the notion of support, which are essential for the definition of demos in Section 7. Sections 8 and 9 describe the construction of models from demos. Section 10

introduces the pruning procedure we use to constructs demos. In Sections 11 and 12 we extend the pruning procedure to also construct Hilbert derivations. In Section 13 we study general models and establish the small model property of CTL.

2 CTL

We briefly recall the syntax and semantics of CTL [11, 8, 1]. We fix a countable alphabet $\mathcal{A}$ of atomic propositions p and define **formulas** as follows:

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

We choose to work with a minimal set of operations for technical convenience. We make use of the following abbreviations for formulas:

$$\begin{aligned} \neg s &:= s \to \bot & \qquad \Diamond s &:= \neg \Box \neg s \\ s \vee t &:= \neg s \to t & \qquad \mathsf{E}(s\,\mathsf{U}\,t) &:= \neg\,\mathsf{A}(\neg s\,\mathsf{R}\,\neg t) \\ s \wedge t &:= \neg(s \to \neg t) & \qquad \mathsf{E}(s\,\mathsf{R}\,t) &:= \neg\,\mathsf{A}(\neg s\,\mathsf{U}\,\neg t) \\ s \leftrightarrow t &:= (s \to t) \wedge (t \to s) \end{aligned}$$

A **model** is a tuple $\mathcal{M} = (|\mathcal{M}|, \Rightarrow_{\mathcal{M}}, L_{\mathcal{M}})$ where

- $|\mathcal{M}|$ is a set of **states**
- $\Rightarrow_{\mathcal{M}} \subseteq |\mathcal{M}| \times |\mathcal{M}|$ is a serial **transition relation** (i.e., every state has at least one successor)
- $L_{\mathcal{M}} : \mathcal{A} \to 2^{|\mathcal{M}|}$ is a **labeling function**.

Intuitively, the formula $\mathsf{A}(s\,\mathsf{U}\,t)$ holds at some state w if for every infinite path starting at w the formula s holds at every state of the path until the path reaches a state where t holds. The formula $\mathsf{A}(s\,\mathsf{R}\,t)$ holds at some state w if for every infinite path starting at w either t holds at every state along the path or s holds at at some state on the path and t holds at every state up to (and including) that state.

For the following definitions, we fix some model $\mathcal{M}$. A **path** is a function $\pi : \mathbb{N} \to |\mathcal{M}|$ such hat $\pi\, n \Rightarrow_{\mathcal{M}} \pi(n+1)$ for all n. The letter π ranges over paths. The **satisfaction relation** $w \vDash s$ for states w of $\mathcal{M}$ and formulas s is defined as

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

$$
\frac{s \qquad s \to t}{t}\ \mathsf{MP} \qquad\qquad \frac{s}{\Box s}\ \mathsf{Nec}
$$

$$
\frac{t \to u \qquad s \to \Box u \to u}{\mathsf{A}(s\,\mathsf{U}\,t) \to u}\ \mathsf{UI} \qquad\qquad \frac{u \to t \qquad u \to \neg s \to \Box u}{u \to \mathsf{A}(s\,\mathsf{R}\,t)}\ \mathsf{RI}
$$

Figure 1: Hilbert System

follows:

$$
\begin{aligned}
w \vDash p &:= w \in L_{\mathcal{M}}\, p \\
w \vDash \bot &:= \bot \\
w \vDash s \to t &:= w \vDash s \to w \vDash t \\
w \vDash \Box s &:= \forall v.\ (w \Rightarrow_{\mathcal{M}} v) \to v \vDash s \\
w \vDash \mathsf{A}(s\,\mathsf{U}\,t) &:= \forall \pi.\ \pi\, 0 = w \to \exists n.\ \pi\, n \vDash t \wedge \forall m < n.\ \pi\, m \vDash s \\
w \vDash \mathsf{A}(s\,\mathsf{R}\,t) &:= \forall \pi.\ \pi\, 0 = w \to \forall n.\ \pi\, n \vDash t \vee \exists m < n.\ \pi\, m \vDash s
\end{aligned}
$$

We write $\mathcal{M} \vDash s$ if $w \vDash s$ for all states w of $\mathcal{M}$.

A formula is **valid** if it is satisfied by every state of every model. A formula is (finitely) **satisfiable** if it is satisfied by some state of some (finite) model.

Validity of formulas can be axiomatized with a Hilbert system [8, 23]. We will work with the Hilbert system shown in Figure 1. We write $\vdash s$ if s is provable from the axioms. We say s is **refutable** if $\neg s$ is provable and call the proof of $\neg s$ a **Hilbert refutation**.

We will show soundness and completeness of the Hilbert system for finite models constructively. The completeness proof comes in the form of a certifying decision method. For every input formula s we either construct a finite model certifying the satisfiability of s or a Hilbert refutation of s certifying the unsatisfiability of s. Using excluded middle and dependent choice, we will also show that the Hilbert system is sound for all models.

3 Inductive Semantics

For our proofs, we employ an inductive semantics similar to the fixpoint semantics used with model checking [13, 9]. We interpret path formulas inductively (for $\mathsf{A}(s\,\mathsf{U}\,t)$) or coinductively (for $\mathsf{A}(s\,\mathsf{R}\,t)$) according to the following fixpoint characterizations [10, 9]:

$$\mathsf{A}(s\,\mathsf{U}\,t) \equiv \mu X.\ t \vee (s \wedge \Box\, X) \tag{3.1}$$

$$\mathsf{A}(s\,\mathsf{R}\,t) \equiv \nu X.\ t \wedge (s \vee \Box\, X) \tag{3.2}$$

Let $\mathcal{M}$ be a model. We define an inductive predicate AU of type

$$(|\mathcal{M}| \to \mathsf{Prop}) \to (|\mathcal{M}| \to \mathsf{Prop}) \to |\mathcal{M}| \to \mathsf{Prop}$$

and a coinductive predicate AR of the same type with the following rules:

$$\frac{Q\,w}{\mathsf{AU}\,P\,Q\,w} \qquad \frac{P\,w \qquad \forall v.(w \Rightarrow_{\mathcal{M}} v) \to \mathsf{AU}\,P\,Q\,v}{\mathsf{AU}\,P\,Q\,w}$$

$$\frac{P\,w \qquad Q\,w}{\mathsf{AR}\,P\,Q\,w} \qquad \frac{Q\,w \qquad \forall v.(w \Rightarrow_{\mathcal{M}} v) \to \mathsf{AR}\,P\,Q\,v}{\mathsf{AR}\,P\,Q\,w}$$

Based on the predicates AU and AR, we define a second satisfaction relation $w \vDash_i s$ between states w of $\mathcal{M}$ and formulas s by recursion on formulas:

$$\begin{aligned}
w \vDash_i p &:= w \in L_{\mathcal{M}}\,p\\
w \vDash_i \bot &:= \bot\\
w \vDash_i s \to t &:= w \vDash_i s \to w \vDash_i t\\
w \vDash_i \Box\, s &:= \forall v \in |\mathcal{M}|.\ (w \Rightarrow_{\mathcal{M}} v) \to v \vDash_i s\\
w \vDash_i \mathsf{A}(s\,\mathsf{U}\,t) &:= \mathsf{AU}\,(\lambda v.v \vDash_i s)\,(\lambda v.v \vDash_i t)\,w\\
w \vDash_i \mathsf{A}(s\,\mathsf{R}\,t) &:= \mathsf{AR}\,(\lambda v.v \vDash_i s)\,(\lambda v.v \vDash_i t)\,w
\end{aligned}$$

The rules and axioms of the Hilbert system in Figure 1 are motivated by the inductive characterizations of path formulas. The axioms U1 and U2 correspond to the introduction rules of the predicate AU and the rule UI corresponds to the respective induction principle. Dually, the axioms R1 and R2 correspond to inversion of the rules for AR and the rule RI corresponds to coinduction. Given the close correspondence between the inductive semantics and the Hilbert system, it is straightforward to show that the Hilbert system is sound for all models where $\vDash_i$ is logically decidable.

Lemma 3.1 Let $\mathcal{M}$ be a model such that $w \vDash_i s \vee (w \nvDash_i s)$ is provable for all formulas s and all states w of $\mathcal{M}$. Then $\mathcal{M} \vDash_i s$ if $\vdash s$.

Proof by induction on $\vdash s$. The case for DN follows since $w \vDash_i \neg\neg s$ is, by definition, equivalent to $\neg\neg(w \vDash_i s)$ and negation behaves classically on decidable properties. The cases for the inductive rules UI and RI are immediate with the inductive interpretation of path formulas. The remaining cases are obvious. ∎

Note that in the presence of excluded middle, the decidability premise of Lemma 3.1 is trivially satisfied. In Section 5 we will show constructively that $\vDash_i$ is decidable for finite models.

4 Decidability and Finite Sets

For constructive proofs, decidability properties play an important role since decidable propositions behave classically. In particular, decidability provides for crucial case distinctions in proofs.

Following Ssreflect [18], we say that a property $P : \mathsf{Prop}$ is **decidable**, if there exists a boolean expression p such that $p = \mathsf{true} \leftrightarrow P$. We call p the **boolean reflection** of P. Similarly, we call a predicate $P : X \to \mathsf{Prop}$ decidable if $P\,x$ is decidable for all $x : X$. Since we work in the constructive type theory of Coq without axioms, the existence of a boolean reflection implies computational decidability. We refer to boolean predicates $X \to \mathbb{B}$ as decidable predicates since they can be seen as their own boolean reflection. If a boolean p appears in the place of a proposition it is to be read as $p = \mathsf{true}$ and similarly for boolean predicates.

In the context of decidability, countable and finite types [17] play an important role. A **countable type** is a type with a decidable equality that can be enumerated (e.g., numbers or formulas). For countable types, a choice function for decidable properties can be constructed. This means for a countable type X there is a function

$$\mathsf{xchoose}_X : \forall p : X \to \mathbb{B}.\ (\exists x.\ p\,x) \to X$$

such that for every proof $E : (\exists x.\ p\,x)$ we have $p\,(\mathsf{xchoose}_X\,p\,E)$. A **finite type** is a countable type whose elements can be given with a list. Many properties that are not decidable in general are decidable over finite types. In particular, quantification over finite types preserves decidability.

In addition to finite types, we make extensive use of finite sets over finite and countable base types. For our purposes, finite sets are data structures. In particular, we only consider finite sets with decidable membership.

The Ssreflect libraries [17] provide finite sets but only over finite base types. For some parts of our development this is to restrictive, since we will work extensively with sets of formulas as well as sets of sets of formulas. For this we require a library providing finite sets over countable types providing the usual

operations including separation $\{x \in A \mid p\,x\}$, replacement $\{f\,x \mid x \in A\}$, and powerset. Since we could not find a library satisfying all our needs, we developed our own.

Our set type is realized as a constructive quotient over lists obtained with a normalization function. The normalization function is defined using a constructive choice operator and picks some canonical duplicate-free list to represent a given set. The extensional representation obtained this way ensures that set membership on all levels (sets, sets of sets, etc.) is just membership in the list representing the set. In the following, we write $\mathsf{set}\,X$ for the type of finite sets over a countable or finite type X.

We will use fixpoint iteration to show that certain inductive definitions over finite types and sets are decidable. Let X be a countable type, $U : \mathsf{set}\,X$, and $F : \mathsf{set}\,X \to \mathsf{set}\,X$ a function that is monotone (i.e. $\forall A\,B.\ A \subseteq B \to F\,A \subseteq F\,B$) and bounded by U (i.e., $\forall A.\ A \subseteq U \to F\,A \subseteq U$). We can compute the least fixpoint of F, written $\mathsf{lfp}\,F$, as $F^{|U|}\,\emptyset$, i.e., by iterating F on the empty set once for every element of U. Further, we can compute its greatest fixpoint, written $\mathsf{gfp}\,F$, as $C_U(\mathsf{lfp}\,(C_U \circ F \circ C_U))$, where C_U is the complement in U. In addition to the fixpoint equations $F(\mathsf{lfp}\,F) = \mathsf{lfp}\,F$ and $F(\mathsf{gfp}\,F) = \mathsf{gfp}\,F$, we show the following "induction" principles:

$$\forall P.P\,\emptyset \to (\forall A.P\,A \to P\,(F\,A)) \to P(\mathsf{lfp}\,F) \tag{4.1}$$

$$\forall P.P\,U \to (\forall A.P\,A \to P\,(F\,A)) \to P(\mathsf{gfp}\,F) \tag{4.2}$$

Note that if X is a finite type, then every function of type $\mathsf{set}\,X \to \mathsf{set}\,X$ is bounded by the full set over X.

5 Soundness for Finite Models

We now show that the Hilbert system in Figure 1 is sound for finite models and the path semantics. Given Lemma 3.1, it suffices to show that on finite models the inductive semantics is decidable and in agreement with the path semantics. This amounts to the verification of a simple model checking algorithm.

We formalize models as records comprised of a type of states, a transition relation, a labeling predicate, and a proof that the transition relation is serial:

$$\begin{array}{lll} \mathsf{model} := \{ & \mathsf{state} & : \mathsf{Type}; \\ & \mathsf{trans} & : \mathsf{state} \to \mathsf{state} \to \mathsf{Prop}; \\ & \mathsf{label} & : \mathcal{A} \to \mathsf{state} \to \mathsf{Prop}; \\ & \mathsf{serial} & : \forall w\,\exists v.\ \mathsf{trans}\,w\,v\} \end{array}$$

A finite model is a model where the type of states is a finite type. In addition, we require that the transition relation and the labeling are decidable for finite models.

We fix some finite model $\mathcal{M}$ for the rest of this section.

Lemma 5.1 $w \vDash_i s$ is decidable for every state w of $\mathcal{M}$ and every formula s.

Proof by Induction on s. Since $\mathcal{M}$ is a finite model, $L_{\mathcal{M}}\, p\, w$ and $w \Rightarrow_{\mathcal{M}} v$ are decidable for all w, v, and p. Implication and the finite quantification in the definition of $w \vDash_i \Box s$ preserve decidability. Hence, it suffices to show that for decidable predicates $P, Q : |\mathcal{M}| \to \mathbb{B}$ the predicates $\mathsf{AU}\, P\, Q$ and $\mathsf{AR}\, P\, Q$ are decidable. We construct boolean reflections for $\mathsf{AU}\, P\, Q$ and $\mathsf{AR}\, P\, Q$ using fixpoint iteration. For this we express the introduction rules for $\mathsf{AU}\, P\, Q$ and $\mathsf{AR}\, P\, Q$ as monotone functions on $\mathsf{set}\, |\mathcal{M}|$:

$$F_{\mathsf{AU}}(A) := \{\, w \in |\mathcal{M}| \mid Q\, w \,\} \cup \{\, w \in |\mathcal{M}| \mid P\, w \wedge \forall v.(w \Rightarrow_{\mathcal{M}} v) \to v \in A \,\}$$
$$F_{\mathsf{AR}}(A) := \{\, w \in |\mathcal{M}| \mid P\, w \wedge Q\, w \,\} \cup \{\, w \in |\mathcal{M}| \mid Q\, w \wedge \forall v.(w \Rightarrow_{\mathcal{M}} v) \to v \in A \,\}$$

Now it suffices to show

$$\mathsf{AU}\, P\, Q\, w \leftrightarrow w \in \mathsf{lfp}\, F_{\mathsf{AU}} \qquad (*)$$
$$\mathsf{AR}\, P\, Q\, w \leftrightarrow w \in \mathsf{gfp}\, F_{\mathsf{AR}} \qquad (**)$$

For $(*)$, the direction from left to right follows by induction on AU and the converse direction follows with (4.1). For $(**)$, the direction from left to right follows with (4.2) and the converse direction follows by coinduction. ∎

It remains to show that the inductive semantics agrees with the path semantics. The interesting part is showing the correctness of AU and AR.

Lemma 5.2 If P and Q are decidable predicates $|\mathcal{M}| \to \mathbb{B}$ and w is a state of $\mathcal{M}$, then

$$\mathsf{AU}\, P\, Q\, w \leftrightarrow \forall \pi.\pi\, 0 = w \to \exists n.Q\, (\pi\, n) \wedge \forall m < n.P\, (\pi\, m)$$

Proof The direction from left to right follows by induction on $\mathsf{AU}\, P\, Q\, w$. Since AU is decidable, we can show the direction from right to left by showing its contrapositive. Assume we have $\neg\mathsf{AU}\, P\, Q\, w$. We construct a path that contradicts the right hand side. Consider the following decidable subrelation of $\Rightarrow_{\mathcal{M}}$:

$$u \rightharpoonup v := u \Rightarrow_{\mathcal{M}} v \wedge (\neg\mathsf{AU}\, P\, Q\, u \to p\, u \to \neg\mathsf{AU}\, P\, Q\, v)$$

Since $\Rightarrow_{\mathcal{M}}$ is serial, the relation $\rightharpoonup$ is serial as well. Using constructive choice we construct a function $f : |\mathcal{M}| \to |\mathcal{M}|$ that selects for every state of $\mathcal{M}$ a $\rightharpoonup$-successor. We define $\pi\, n := f^n\, w$ and show

$$\forall n.\neg\mathsf{AU}\, P\, Q\, (\pi\, n) \vee \exists m < n.\neg P\, (\pi\, m)$$

by induction on n. This yields $\neg Q\, (\pi\, n) \vee \exists m < n.\neg P\, (\pi\, m)$ for all n, contradicting the right hand side as required. ∎

Lemma 5.3 If P and Q are decidable predicates $|\mathcal{M}| \to \mathbb{B}$ and w is a state of $\mathcal{M}$, then

$$\mathsf{AR}\,P\,Q\,w \leftrightarrow \forall\pi.\,\pi\,0 = w \to \forall n.Q\,(\pi\,n) \vee \exists m < n.P\,(\pi\,m)$$

Proof For the direction from left to right, we assume $\mathsf{AR}\,P\,Q\,w$ and fix some path π such that $\pi\,0 = w$. We prove

$$\forall n.\,\mathsf{AR}\,P\,Q\,(\pi\,n) \vee \exists m < n.P\,(\pi\,m)$$

by induction on n. The base case follows by assumption, the induction step by inversion on $\mathsf{AR}\,P\,Q\,(\pi\,n)$. The claim then follows since $\mathsf{AR}\,P\,Q\,(\pi\,n)$ implies $Q\,(\pi\,n)$.

For the converse direction, we abbreviate the right hand side as $\mathsf{AR}'\,w$. We first show that the path characterization satisfies the inversion properties of AR.

$$\forall v.\,\mathsf{AR}'\,v \to Q\,v \tag{$*$}$$

$$\forall u\,v.\,\mathsf{AR}'\,u \to \neg P\,u \to (u \Rightarrow_{\mathcal{M}} v) \to \mathsf{AR}'\,v \tag{$**$}$$

For $(*)$, we use constructive choice to obtain some path through the model. Property $(**)$ is easy to verify. The claim then follows by coinduction using $(*)$ and $(**)$. ∎

Lemma 5.4 (Finite Agreement) Let w be a state of $\mathcal{M}$. Then $w \vDash s$ iff $w \vDash_i s$.

Proof by induction on s using Lemma 5.2 and Lemma 5.3. ∎

Soundness for finite models is now a direct consequence of the previous lemmas.

Theorem 5.5 (Finite Soundness) $\mathcal{M} \vDash s$ if $\vdash s$ and $\mathcal{M}$ is a finite model.

Proof Let $\mathcal{M}$ be finite model and $\vdash s$. By Lemma 5.4 it suffices to show $w \vDash_i s$ for all states w of $\mathcal{M}$. Since $w \vDash_i s$ is decidable by Lemma 5.1, the claim follows with Lemma 3.1. ∎

6 Clauses and Support

The central notion in the design of our certifying decision method is the notion of a demo. Demos are a class of syntactic pseudo-models, a variation of the pseudo-Hintikka structures used by Emerson [8]. We will show that every demo can be turned into a model and that every formula that is not supported by a demo is refutable.

For our demos we use literal clauses and the notion of support [21, 22] instead of the more traditional notion of Hintikka sets [24, 12, 8, 20]. Clausal demos appear in [21] as evident branches. We use signed formulas [25] to express top-level negations. Signs are a formal device that leads to a simple definition of subformula closure (Section 10) based on our minimal syntax.

A **signed formula** is either s^+ or s^- where s is a formula. Signs bind weaker than formula constructors, so $\Box\, s^+$ is to be read as $(\Box\, s)^+$. 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 $\lfloor s^- \rfloor = \neg s$. Hence, negative signs can be thought of as top-level negations. We have the following equivalences:

$$\begin{aligned} \lfloor \Box\, s^- \rfloor &\leftrightarrow \Diamond \neg s \\ \lfloor \mathsf{A}(s\,\mathsf{U}\,t)^- \rfloor &\leftrightarrow \mathsf{E}(\neg s\,\mathsf{R}\,\neg t) \\ \lfloor \mathsf{A}(s\,\mathsf{R}\,t)^- \rfloor &\leftrightarrow \mathsf{E}(\neg s\,\mathsf{U}\,\neg t) \end{aligned}$$

A **clause** is a finite set of signed formulas. A state satisfies a clause if it satisfies all its members. A signed formula is a **literal** if it is of the form p^σ, $\bot^\sigma$, or $\Box\, s^\sigma$. 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. We refer to locally consistent literal clauses as **base clauses**.

We define the **support relation** $C \rhd s^\sigma$ between base clauses C and signed formulas s^σ by recursion on formulas:

$$\begin{aligned} C \rhd s^\sigma &:= s^\sigma \in C && \text{if } s^\sigma \text{ is a literal} \\ C \rhd (s \to t)^+ &:= C \rhd s^- \;\vee\; C \rhd t^+ \\ C \rhd (s \to t)^- &:= C \rhd s^+ \;\wedge\; C \rhd t^- \\ C \rhd \mathsf{A}(s\,\mathsf{U}\,t)^+ &:= C \rhd t^+ \;\vee\; (C \rhd s^+ \;\wedge\; \Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+ \in C) \\ C \rhd \mathsf{A}(s\,\mathsf{U}\,t)^- &:= C \rhd t^- \;\wedge\; (C \rhd s^- \;\vee\; \Box\,\mathsf{A}(s\,\mathsf{U}\,t)^- \in C) \\ C \rhd \mathsf{A}(s\,\mathsf{R}\,t)^+ &:= C \rhd t^+ \;\wedge\; (C \rhd s^+ \;\vee\; \Box\,\mathsf{A}(s\,\mathsf{R}\,t)^+ \in C) \\ C \rhd \mathsf{A}(s\,\mathsf{R}\,t)^- &:= C \rhd t^- \;\vee\; (C \rhd s^- \;\wedge\; \Box\,\mathsf{A}(s\,\mathsf{R}\,t)^- \in C) \end{aligned}$$

We also define $C \rhd D := \forall s^\sigma \in D.\ C \rhd s^\sigma$. The support relation can be understood as a restricted form of entailment justified by propositional reasoning and the equivalences (3.1) and (3.2).

We employ base clauses and support rather than Hintikka sets because we find the structurally recursive definition of support easier to work with than sets with closure conditions.

7 Demos

Let C and D be a clauses. We call the set $\mathcal{R}^\Box C := \{ s^+ \mid \Box s^+ \in C \}$ the **box request** of C. If $D \rhd \mathcal{R}^\Box C$, we say that there is a **possible transition** from C to D or that D is a **possible successor** of C.

A demo will be a set of base clauses S that can be turned into a finite model where the states are clauses from S, possibly appearing multiple times, and the transition relation is a selection of possible transitions. The model will be constructed such that every state satisfies all formulas it supports. For this, we need to ensure that every clause in S has enough possible successors in S to satisfy all literals of the form $\Box s^-$. We also need to ensure that all path formulas are satisfied.

We refer to the signed formulas $\mathsf{A}(s \,\mathsf{U}\, t)^+$ and $\mathsf{A}(s \,\mathsf{R}\, t)^-$ as **eventualities**. To fulfill the eventuality $\mathsf{A}(s \,\mathsf{R}\, t)^-$, we need to ensure that there is a path of possible transitions in S supporting s^- at every state until finally supporting t^-. To fulfill $\mathsf{A}(s \,\mathsf{U}\, t)^+$, we need to ensure that there exists a directed acyclic graph with labels from S supporting s^+ at internal nodes and t^+ at the leaves. Dealing with eventualities is the main technical difficulty in designing demos.

Let C be a clause. We define the **requests** of C as follows:

$$\mathcal{R}\, C := \{\mathcal{R}^\Box C\} \cup \{\, \mathcal{R}^\Box C, s^- \mid \Box s^- \in C \,\}$$

One necessary condition for a clause C to be satisfied by some state w is that all its requests are satisfied by immediate successors of w. The singleton $\{\mathcal{R}^\Box C\}$ captures the fact that models are serial and ensures that every clause has at least one request. Note that the requests need not be base clauses. We say that a clause C is **supported by** S, written $S \rhd C$, if for some clause $D \in S$ we have $D \rhd C$. For S to be a demo we require that every request of a clause $C \in S$ is supported by S.

It remains to define the fulfillment conditions for eventualities. Since base clauses contain only literals, it is technically convenient to phrase the fulfillment condition in terms of **eventuality literals**, i.e., $\Box\, \mathsf{A}(s \,\mathsf{R}\, t)^-$ and $\Box\, \mathsf{A}(s \,\mathsf{R}\, t)^+$.

We inductively define **fulfillment relations** $S, C \rhd e$ between sets of clauses S, clauses C and eventuality literals. A clause $C \in S$ fulfills $\Box\, \mathsf{A}(s \,\mathsf{R}\, t)^-$ if there exists a possible successor E of C such that either $E \rhd t^-$ or $E \rhd s^-$ and E fulfills $\Box\, \mathsf{A}(s \,\mathsf{R}\, t)^-$. The precise rule defining $S, C \rhd \Box\, \mathsf{A}(s \,\mathsf{R}\, t)^-$ inductively is:

$$\frac{E \in S \qquad E \rhd \mathcal{R}^\Box C \qquad E \rhd t^- \vee (E \rhd s^- \wedge\ S, E \rhd \Box\, \mathsf{A}(s \,\mathsf{R}\, t)^-)}{S, C \rhd \Box\, \mathsf{A}(s \,\mathsf{R}\, t)^-}$$

To fulfill the eventuality literal $\Box\, \mathsf{A}(s \,\mathsf{U}\, t)^+$, every successor needs to fulfill $\mathsf{A}(s \,\mathsf{U}\, t)^+$. For the model construction, we therefore want to introduce as few successors as possible. However, we need to introduce at least one successor for

every request. Thus, we need to ensure that one of the possible successors for every request fulfills the eventuality.

A clause $C \in S$ fulfills $\Box \mathsf{A}(s \mathsf{U} t)^+$ if for every request D of C there exists a clause $E \in S$ such that $E \rhd D$ and either $E \rhd t^+$ or $E \rhd s^+$ and E fulfills $\Box \mathsf{A}(s \mathsf{U} t)^+$. The precise rule defining $S, C \rhd \Box \mathsf{A}(s \mathsf{U} t)^+$ inductively is:

$$\frac{\forall D \in \mathcal{R}\, C\ \exists E \in S.\ E \rhd D \wedge (E \rhd t^+ \vee (E \rhd s^+ \wedge\ S, E \rhd \Box \mathsf{A}(s \mathsf{U} t)^+))}{S, C \rhd \Box \mathsf{A}(s \mathsf{U} t)^+}$$

The notion of inductive fulfillment arises naturally from the inductive semantics of CTL. Inductive fulfillment is a sufficient criterion for the existence of the paths and trees mentioned above and eliminates the need for embedded fragments as used in [8].

Definition 7.1 (Demo) A set of base clauses S is called a **demo** if it satisfies the following conditions:

D1. If $C \in S$ and $D \in \mathcal{R}C$, then $S \rhd D$.

D2. If e is an eventuality literal and $e \in C \in S$, then $S, C \rhd e$.

8 Model Construction

We now show how to turn demos into models. The proof consists of two parts. Fist we unwind the demo into a collection of clause-labeled graphs called fragments such that each fragment fulfills one eventuality in one clause. Following [8], we then assemble the fragments into a finite model.

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 $x \Rightarrow_G y$ 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 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 set of base clauses S for the rest of this section.

Let $C \in S$ be a clause. A fragment G is an **fragment for C** if:

F1. $\Lambda_x \in S$ for all $x \in G$.

F2. The root of G is labeled with C.

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

F4. If $x \in G$ is internal and $D \in \mathcal{R}(\Lambda_x)$, then there exists some $y \in G$ such that $x \Rightarrow_G y$ and $\Lambda_y \rhd D$.

Let u be a formula. A fragment G for C is a **fragment for C and u** if $u \notin C$ or

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

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

Note that if $C \rhd \mathsf{A}(s\,\mathsf{U}\,t)^+$, then $\mathsf{A}(s\,\mathsf{U}\,t)^+$ is fulfilled in every fragment for C and $\Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+$. Either $C \rhd t^+$ and $\mathsf{A}(s\,\mathsf{U}\,t)^+$ is fulfilled directly at the root or $C \rhd s^+$ and $\Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+ \in C$ and $\mathsf{A}(s\,\mathsf{U}\,t)^+$ is fulfilled according to (E1). For $\mathsf{A}(s\,\mathsf{R}\,t)^-$ the situation is similar.

Definition 8.1 (Fragment Demo) Let $V := \bigcup_{C \in S} C$. A **fragment demo** (for S) is an indexed collection of nontrivial fragments $(G(u,C))_{u \in V, C \in S}$ where each $G(u,C)$ is a fragment for C and u.

Lemma 8.2 If S is a demo, then there exists a fragment demo for S such that every fragment has most $2 \cdot |S|$ nodes.

Proof We obtain a fragment for $C \in S$ and $u \in V$ as follows. We distinguish three cases.

1. $u = \Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+ \in C$ and $C \rhd s^+$: We define:

$$A_l := \{\, D \in S \mid S, D \rhd \Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+ \,\}$$
$$A_r := \{\, D \in S \mid D \rhd t^+ \,\}$$

   The fulfillment relation for $\Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+$ is an inductive predicate over a finite set. Thus, A_l can be computed as the least fixpoint of some monotone function F bounded by S. For every clause $C \in A_l$, we define the level of C to be the smallest n such that $C \in F^{n+1}\emptyset$. We define a terminating relation $\rightharpoonup$ on the disjoint (tagged) union $A_l \uplus A_r$ such that

$$\mathsf{inl}(D) \rightharpoonup \mathsf{inl}(E) \;\leftrightarrow\; E \rhd \mathcal{R}^\Box D \wedge E \rhd s^+ \wedge \mathsf{level}\,E < \mathsf{level}\,D$$
$$\mathsf{inl}(D) \rightharpoonup \mathsf{inr}(E) \;\leftrightarrow\; E \rhd \mathcal{R}^\Box D$$

   There are no transitions within A_r or from A_r to A_l. Since S is a demo, we have $S, C \rhd \Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+$ and thus $C \in A_l$. The subgraph reachable from $\mathsf{inl}(C)$ yields a fragment for C and u of the required size. It is immediate that the fragment satisfies (F1–3). Conditions (F4) and (E1) follow with the definition of fulfillment.

2. $u = \Box\,\mathsf{A}(s\,\mathsf{R}\,t)^- \in C$ and $C \rhd s^-$: We define:

$$A_l := \{\, D \in S \mid S, D \rhd \Box\,\mathsf{A}(s\,\mathsf{R}\,t)^- \,\}$$

   As above, we define levels for the clauses in A_l and define a terminating relation $\rightharpoonup$ on the disjoint union $A_l \uplus S$ such that

$$\mathsf{inl}(D) \rightharpoonup \mathsf{inl}(E) \;\leftrightarrow\; E \rhd \mathcal{R}^\Box D \wedge E \rhd s^- \wedge \mathsf{level}\,E < \mathsf{level}\,D$$
$$\mathsf{inl}(D) \rightharpoonup \mathsf{inr}(E) \;\leftrightarrow\; E \rhd \mathcal{R}^\Box D$$

Again, the subgraph reachable from $\mathsf{inl}(C)$ yields a fragment for C and u of the required size. Conditions (F1–4) are easy to verify. Condition (E2) follows by induction on the level of C using the definition of fulfillment.

3. In all other cases it suffices to construct a fragment for C. The fragment consists of a root node labeled with C and leaves labeled with all possible successors of C in S. ∎

The levels employed in the proof of Lemma 8.2 are similar to the ranks employed by Ben-Ari et al. [2]. However, their model construction follows a different approach.

We now show how to assemble a fragment demo into a model. The construction is adapted from Emerson's handbook article [8]. We verify the construction using the inductive semantics of CTL instead of the path semantics.

Assume that we are given some fragment demo $(G(u,C))_{u\in V, C\in S}$ for S. We construct a finite model $\mathcal{M}$ satisfying all labels occurring in the demo. If V is empty, there is nothing to show, so we assume that V 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,C,x) with $u \in V$, $C \in S$, and $x \in G(u,C)$. A state (u,C,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 V. 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 $G(u_i,C_j)$, we add transitions from (u_i,C_j,x) to all successors of the root of $G(u_{i+1},\Lambda_x)$. This leads to the following definition:

$$\begin{aligned}(u_i,C,x) \Rightarrow_{\mathcal{M}} (v,D,y) := {}& (v = u_i \wedge D = C \wedge x \Rightarrow_{G(v,D)} y) \vee \\ & (\mathsf{leaf}\, x \wedge v = u_{i+1} \wedge D = \Lambda_x \wedge x_{root} \Rightarrow_{G(v,D)} y)\end{aligned}$$

The fragments in the demo can be thought of as arranged in a matrix as shown in Figure 2 where the C_i are the clauses in S. Since fragment demos consist of nontrivial fragments only, the resulting transition system is serial and we have a model.

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

Lemma 8.3 If $(u,C,x) \in |\mathcal{M}|$ and $\Lambda_x \rhd s^\sigma$, then $(u,C,x) \models_i \lfloor s^\sigma \rfloor$.

Proof 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 $w = (u_i,C_j,x) \in |\mathcal{M}|$ 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

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

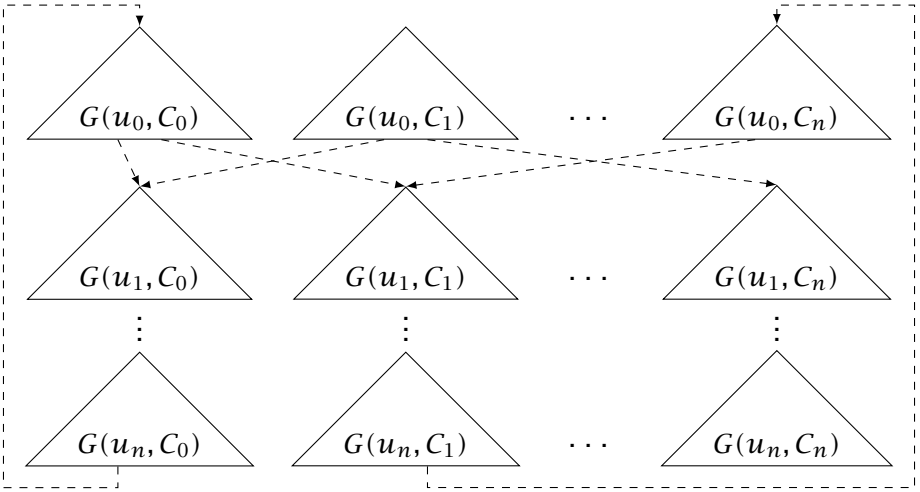


Figure 2: Matrix of Fragments (adapted from [8])

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

If $C \rhd t^+$, then $\mathsf{AU}_{\mathcal{M}}\, s\, t\, (u_{i+1}, C, x_{root})$ trivially holds. Otherwise, we have $C \rhd s^+$ and $\Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+ \in C$. In particular, we have $\Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+ \in V$. We prove $\mathsf{AU}_{\mathcal{M}}\, s\, t\, (u_{i+1}, C, x_{root})$ by induction on the distance from u_{i+1} to $\Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+$ according to the order on V. If $u_{i+1} = \Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+$, we have $\mathsf{AU}_{\mathcal{M}}\, s\, t\, (u_{i+1}, C, x_{root})$ by (E1). Otherwise, the claim follows by induction, deferring to the next row of the matrix as we did above. ∎

Theorem 8.4 (Model Existence) If S is a demo and V is nonempty, then there exists a model with at most $2 \cdot |S|^2 \cdot |V|$ states satisfying every clause supported by S.

Proof Follows immediately with Lemma 8.2 and Lemma 8.3. ∎

9 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.

The levels employed in the proof of Lemma 8.2 are computed using a specialization of constructive choice for natural numbers which chooses minimal witnesses for inhabited and decidable predicates.

Let $\{x, y\} \subseteq \mathsf{lfp}\, F$ for some monotone and bounded function F. To formalize the proof of Lemma 8.2, we only need two facts to characterize levels:

$$x \in F^{\mathsf{level}\, x+1} \emptyset$$
$$y \in F^{\mathsf{level}\, x} \emptyset \to \mathsf{level}\, y < \mathsf{level}\, x$$

We remark that although the fulfillment relations are conceptually inductive definitions, the formal proofs only employ the fixpoint characterization of fulfillment.

Once we have constructed a fragment demo, we turn the finite set $V \times S$ into a finite type I. Except for the transitions connecting the leaves of one row to the next row, the model is then obtained as the disjoint union of a collection of graphs indexed by I. Let $G : I \to \mathsf{graph}$ be such a collection. We lift the internal edges of G by defining a predicate

$$\mathsf{liftEdge} \;:\; (\Sigma i{:}I.\, G\, i) \to (\Sigma i{:}I.\, G\, i) \to \mathbb{B}$$

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

$$\mathsf{liftEdge}\, (i, x)\, (i, y) \;\leftrightarrow\; x \Rightarrow_{G\, i} y$$
$$i \neq j \;\to\; \neg \mathsf{liftEdge}\, (i, x)\, (j, y)$$

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

The proof of Lemma 8.3 contains a nested induction on the distance from u_{i+1} to $\Box \mathsf{A}(s \,\mathsf{U}\, t)^+$ according to the order on V. This distance is formalized using two functions[1] $\mathsf{dist} : V \to V \to \mathbb{N}$ and $\mathsf{next} : V \to V$ satisfying

$$\mathsf{dist}\, s\, t = 0 \;\to\; s = t$$
$$\mathsf{dist}\, s\, t = n + 1 \;\to\; \mathsf{dist}\, (\mathsf{next}\, x)\, y = n$$

Note that the distance defined this way is asymmetric. The definitions of dist and next employ an enumeration of V and arithmetic modulo $|V|$. The notation u_{i+1} is formalized as $\mathsf{next}\, u_i$.

We remark that in [8] 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

[1] we also write V for the finite type of elements of V

row to the successors of the equally labeled root of the next row. This way, we avoid the Σ-type construction at the cost of obtaining a slightly weaker bound on the size of the constructed model. The construction makes use of the fact that CTL formulas cannot distinguish different states that are labeled with the same atomic propositions and have the same successors.

10 Pruning and Subformula Closure

On a given input formula our certifying decision method starts by constructing a subformula closed clause U containing the input formula. We then use Pratt-style pruning [24, 20] to construct the largest demo over U. We will show that the demo constructed in this way supports all satisfiable formulas in U and that all unsupported formulas from U are refutable.

We call a clause U **subformula closed**, if it satisfies the following conditions:

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

S2. If $\Box\, s^\sigma \in U$, then $s^\sigma \in U$.

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

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

For every signed formula s^σ, one can compute a smallest subformula closed clause containing s^σ. We refer to this clause as the **subformula closure** of s^σ.

We write $|s|$ for the **size** of the formula s.

Lemma 10.1 The subformula closure of s^σ has size at most $2 \cdot |s|$.

We fix some subformula closed clause U for the rest of this section.

Let $C \subseteq U$ be a base clause. Note that all requests of C are themselves contained in U. Also, if $s^\sigma \in U$, then every signed formula involved in deciding $C \rhd s^\sigma$ is in U. Let us remark that our minimal syntax combined with signs provides for a very simple notion of subformula closure.

It turns out that there exists a unique demo $\mathcal{D}$ that contains exactly the satisfiable base clauses over U. We compute $\mathcal{D}$ using Pratt-style pruning. We start with the set

$$S_0 := \{\, C \subseteq U \mid C \text{ base clause} \,\}$$

We then successively remove clauses that violate one of the demo conditions until we are left with a demo. This is possible since (D1) and (D2) are decidable. As noted in the proof of Lemma 8.2, the fulfillment relations can be decided using fixpoint iteration and all other predicates that occur are obviously decidable.

By Theorem 8.4, we already know that a formula $s^\sigma \in U$ is satisfiable if $\mathcal{D} \rhd \{s^\sigma\}$. It remains to show that $\lfloor s^\sigma \rfloor$ is refutable if $\mathcal{D} \not\rhd \{s^\sigma\}$. The construction of $\mathcal{D}$ using pruning gives rise to a number of abstract refutation conditions.

Definition 10.2 (Refutation Predicate) A **refutation predicate** is a predicate ref satisfying the following conditions for every clause C:

R1. If S is **corefutable** (i.e., $\forall D \in S_0 \setminus S.\,\mathsf{ref}\,D$), $C \subseteq U$, and $S \not\triangleright C$, then $\mathsf{ref}\,C$.

R2. If $\mathsf{ref}\,D$ for some $D \in \mathcal{R}\,C$, then $\mathsf{ref}\,C$.

R3. If S is corefutable, $e \in C \in S$ for some eventuality literal e, and $S, C \not\triangleright e$, then $\mathsf{ref}\,C$.

A refutation predicate ref is **sound** if $\mathsf{ref}\,C$ implies that C is not satisfied by any finite model. Note that if ref is sound then $\mathcal{D} \triangleright C$ and $\mathsf{ref}\,C$ are mutually exclusive.

Lemma 10.3 Let ref be a refutation predicate and let $C \subseteq U$. If $\mathcal{D} \not\triangleright C$, then $\mathsf{ref}\,C$.

Proof The set S_0 is clearly corefutable. If S is corefutable, any clause violating (D1) is refutable by (R1) and (R2). Further, any clause violating (D2) is refutable by (R3). Thus, $\mathcal{D}$ is corefutable and the claim follows with (R1). ∎

Lemma 10.4 Let ref be a sound refutation predicate and let $C \subseteq U$. Then $\mathsf{ref}\,C$ is decidable.

Proof Follows with Theorem 8.4, Lemma 10.3, and the fact hat $\mathcal{D} \triangleright C$ is decidable. ∎

Unsatisfiability in finite models yields a sound refutation predicate. However, we do not prove this directly. Instead we will show the stronger claim that Hilbert refutability yields a sound refutation predicate.

11 Informative Completeness

We now finish the constructive completeness proof by showing that the Hilbert axiomatization realizes a refutation predicate. We deviate from the original proofs [12, 8] because they are non-constructive in that they assume that Hilbert provability is logically decidable (i.e., $\forall s.\, \vdash s \vee \nvdash s$). Our construction of Hilbert refutations is inspired by the work of Ben-Ari et al. [2], where a similar constructive proof is sketched for UB, a fragment of CTL.

In addition to the abbreviations defined in Section 2, we define "big" conjunctions and disjunctions indexed by lists. We use big conjunctions to reason about clauses inside the Hilbert system. If C is a clause, we refer to $\bigwedge_{s^\sigma \in C} \lfloor s^\sigma \rfloor$ as its **associated formula**.[2] When a clause occurs in the place of a formula, it is to be read as its associated formula.

[2] We convert finite sets to lists as required.

UE	$\vdash \mathsf{A}(s\,\mathsf{U}\,t) \leftrightarrow t \vee (s \wedge \Box\,\mathsf{A}(s\,\mathsf{U}\,t))$
RE	$\vdash \mathsf{A}(s\,\mathsf{R}\,t) \leftrightarrow t \wedge (s \vee \Box\,\mathsf{A}(s\,\mathsf{R}\,t))$
$\mathsf{R}^\Box$	$\vdash C \to \Box(\mathcal{R}^\Box\, C)$
R	If $D \in \mathcal{R}C$, then $\vdash C \to \Diamond D$
Reg	If $\vdash s \to t$, then $\vdash \Box s \to \Box t$ and $\vdash \Diamond s \to \Diamond t$

Figure 3: CTL Lemmas

We will show that $\lambda C.\ \vdash \neg C$ is a sound refutation predicate. Soundness follows with Theorem 5.5. To show the refutation conditions we need a number of lemmas about the Hilbert system. The most important ones are shown in Figure 3. We defer the discussion of the technical issues involved in proving these lemmas to the next section. We continue to work with the subformula closed clause U from the previous section.

We start by showing refutation condition (R1). For this we show that every clause $C \subseteq U$ implies the disjunction of all clauses in S that support C. We define the **base of C in S** to be the set

$$\mathcal{B}_S\, C := \{\, D \in S \mid D \rhd C \,\}$$

If A is a set of clauses, we abbreviate $\bigvee_{C \in A} C$ as $\bigvee A$. We also abbreviate $C \cup \{s^\sigma\}$ as C, s^σ.

Lemma 11.1 Let $S \subseteq S_0$ be corefutable and $C \subseteq U$. Then $\vdash C \to \bigvee \mathcal{B}_S\, C$.

Proof by induction sum of the sizes of the non-literal formulas in C. Assume there exists some non-literal formula $u \in C$. We consider the case where $u = s \to t^+$. We have

$$\vdash C \to (C \setminus \{u\}), s^- \vee (C \setminus \{u\}), t^+$$

by propositional reasoning. By induction hypothesis this yields

$$\vdash C \to \bigvee \mathcal{B}_S((C \setminus \{u\}), s^-) \vee \bigvee \mathcal{B}_S((C \setminus \{u\}), t^+)$$

The claim then follows since $\mathcal{B}_S\,((C \setminus \{u\}), s^-) \cup \mathcal{B}_S((C \setminus \{u\}), t^+) \subseteq \mathcal{B}_S\, C$. All other cases are similar. The cases for $\mathsf{A}(s\,\mathsf{U}\,t)^\sigma$ and $\mathsf{A}(s\,\mathsf{R}\,t)^\sigma$ follow with UE and RE respectively.

If C contains only literals, there are three cases: C is not locally consistent, $C \in S$, or $C \in S_0 \setminus S$. In each case the claim follows with propositional reasoning. ∎

The refutation condition (R1) is an immediate consequence of the previous lemma.

Lemma 11.2 Let $S \subseteq S_0$ be corefutable and $C \subseteq U$. Then $\vdash \neg C$ if $S \not\vdash C$.

To show the refutation conditions for eventuality literals we need to use the induction rules of the Hilbert system. We do not use the rules UI and RI directly. Instead, we use derived rules tailored for the refutation of eventuality literals.

Lemma 11.3 1. If $\vdash u \to \Box(t \wedge (\neg s \to u))$, then $\vdash u \to \Box\,\mathsf{A}(s\,\mathsf{R}\,t)$.
2. If $\vdash u \to \Diamond(\neg t \wedge (s \to u))$, then $\vdash u \to \neg\Box\,\mathsf{A}(s\,\mathsf{U}\,t)$.

Proof We prove (2), the proof for (1) is similar but simpler. Assume $\vdash u \to \Diamond(\neg t \wedge (s \to u))$. Let $u' := \neg(\neg t \wedge (s \to u))$. We reason as follows:

$$\begin{array}{lll} & \vdash u \to \neg\Box\,\mathsf{A}(s\,\mathsf{U}\,t) & \\ \Leftarrow & \vdash \neg\Box\,u' \to \neg\Box\,\mathsf{A}(s\,\mathsf{U}\,t) & \text{assumption, def-}\Diamond \\ \Leftarrow & \vdash \mathsf{A}(s\,\mathsf{U}\,t) \to u' & \text{prop., Reg} \\ \Leftarrow & \vdash t \to u' \text{ and } \vdash s \to \Box\,u' \to u' & \mathsf{AI} \end{array}$$

The left claim follows by propositional reasoning. The right claim is equivalent to $\vdash s \to (\neg t \wedge s \to u) \to \Diamond(\neg t \wedge s \to u)$ and hence to $\vdash s \to (\neg t \wedge s \to u) \to u$. ∎

Note that Lemma 11.3(2) turns the induction rule for $\mathsf{A}(s\,\mathsf{U}\,t)$ into a coinductive refutation rule. We will use the coinduction rules established above to refute clauses with unsupported eventuality literals.

The crucial point in the construction of refutations for eventualities is to obtain a suitable coinduction invariant (i.e., the formula u in Lemma 11.3). Consider some corefutable set $S \subseteq S_0$. For both eventuality literals, the invariant is the disjunction of the clauses in S that do not fulfill the eventuality literal. Intuitively, this means some required path (in the case of $\Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+$) or all possible paths (in the case of $\Box\,\mathsf{A}(s\,\mathsf{R}\,t)^-$) stay within this collection of clauses. Thus, it is impossible to satisfy the eventuality literal. The invariants are adapted from [8].

Lemma 11.4 Let $S \subseteq S_0$ be corefutable and $C \in S$. Then $\vdash \neg C$ if $\Box\,\mathsf{A}(s\,\mathsf{R}\,t)^- \in C$ and $S, C \not\vdash \Box\,\mathsf{A}(s\,\mathsf{R}\,t)^-$

Proof We define

$$I := \{\, D \in S \mid S, D \not\vdash \Box\,\mathsf{A}(s\,\mathsf{R}\,t)^- \,\} \qquad\qquad u := \bigvee I$$

For every $D \in I$, we have

$$S \not\vdash \mathcal{R}^{\Box} D, t^- \tag{$*$}$$
$$\mathcal{B}_S(\mathcal{R}^{\Box} D, s^-) \subseteq I \tag{$**$}$$

since violating either condition would allow us to prove $S, D \rhd \Box\,\mathsf{A}(s\,\mathsf{R}\,t)^-$.

Since $\Box\,\mathsf{A}(s\,\mathsf{R}\,t)^- \in C \in I$, it suffices to show:

$$
\begin{array}{lll}
 & \vdash u \to \Box\,\mathsf{A}(s\,\mathsf{R}\,t) & \\
\Leftarrow & \vdash u \to \Box(t \wedge (\neg s \to u)) & \text{Lemma 11.3} \\
\Leftarrow & \vdash D \to \Box(t \wedge (\neg s \to u)) & D \in I \\
\Leftarrow & \vdash \Box(\mathcal{R}^{\Box} D) \to \Box(t \wedge (\neg s \to u)) & \mathsf{R}^{\Box} \\
\Leftarrow & \vdash \mathcal{R}^{\Box} D \to t \text{ and } \vdash \mathcal{R}^{\Box} D \wedge \neg s \to u & \mathsf{Reg}
\end{array}
$$

The left claim is equivalent to $\vdash \neg\mathcal{R}^{\Box} D, t^{-}$ and follows with $(*)$ and Lemma 11.2. Likewise, the right claim is equivalent to $\vdash \mathcal{R}^{\Box} D, s^{-} \to u$ and follows with $(**)$ and Lemma 11.1. ∎

Lemma 11.5 Let $S \subseteq S_0$ be corefutable and $C \in S$. Then $\vdash \neg C$ if $\Box\,\mathsf{A}(s\,\mathsf{U}\,t)^{+} \in C$ and $S, C \not\vdash \Box\,\mathsf{A}(s\,\mathsf{U}\,t)^{+}$.

Proof We define

$$
I := \{\, D \in S \mid S, D \not\vdash \Box\,\mathsf{A}(s\,\mathsf{U}\,t)^{+} \,\} \qquad\qquad u := \bigvee I
$$

For every $D \in I$ there exists some $E \in \mathcal{R}\,D$ such that

$$
S \not\vdash E, t^{+} \wedge \mathcal{B}_S\,(E, s^{+}) \subseteq I \qquad (*)
$$

since otherwise we would have $S, D \rhd \Box\,\mathsf{A}(s\,\mathsf{U}\,t)^{+}$. Since $\Box\,\mathsf{A}(s\,\mathsf{U}\,t)^{+} \in C \in I$, it suffices to show:

$$
\begin{array}{lll}
 & \vdash u \to \neg\,\Box\,\mathsf{A}(s\,\mathsf{U}\,t) & \\
\Leftarrow & \vdash u \to \Diamond(\neg t \wedge (s \to u)) & \text{Lemma 11.3} \\
\Leftarrow & \vdash D \to \Diamond(t \wedge (s \to u)) & D \in I \\
\Leftarrow & \vdash \Diamond E \to \Diamond(\neg t \wedge (s \to u)) & E \text{ as given by } (*),\ \mathsf{R} \\
\Leftarrow & \vdash E, t^{+} \to \bot \text{ and } \vdash E, s^{+} \to u & \mathsf{Reg}, \text{ prop.}
\end{array}
$$

Both claims follow immediately with $(*)$ and Lemma 11.1. ∎

Lemma 11.6 The predicate $\lambda C.\ \vdash \neg C$ is a sound refutation predicate.

Proof Soundness follows with Theorem 5.5. Condition (R2) follows with R. The remaining refutation conditions follow with Lemmas 11.2, 11.4, and 11.5. ∎

We now have everything in place for our main result.

Theorem 11.7 (Informative Completeness) For every formula s we can construct either a Hilbert proof of $\neg s$ or a model satisfying s having at most $|s| \cdot 2^{4 \cdot |s|+2}$ states.

Proof Fix some formula s and let U be the subformula closure of s^{+}. Let $\mathcal{D}$ be the demo over U constructed using pruning. If $\mathcal{D} \not\vdash \{s^{+}\}$ we obtain a refutation of s with Lemma 10.3 and Lemma 11.6. Otherwise, we obtain a model of the required size with Theorem 8.4 and Lemma 10.1. ∎

Together with Theorem 5.5, we obtain the following corollaries:

Corollary 11.8 (Finite Agreement) $\vdash s$ iff $\mathcal{M} \vDash s$ for every finite model $\mathcal{M}$.

Corollary 11.9 (Decidability) Hilbert provability and finite satisfiability are decidable.

12 Constructing Hilbert Derivations in Coq

In order to formalize the results from the previous section, we need to construct many derivations in the Hilbert system. Given that the Hilbert system is low-level and not goal-directed, this is a difficult task. The problem of constructing Hilbert derivations is in many respects similar to the construction of proof terms. We use Coq's tactic language to provide goal management, rewriting, and assumption management for the construction of Hilbert derivations.

We use setoid rewriting [26] to rewrite with provable equivalences (e.g., De Morgan laws, UE, RE) inside formulas. While rewriting with equivalences is convenient, it is often too restrictive for our purposes because several important lemmas (e.g., $\mathsf{R}^{\Box}$ and R) are implications. Therefore, we also use setoid rewriting to rewrite with provable implications using compatibility rules such as

$$\frac{\vdash s' \to s \qquad \vdash t \to t'}{\vdash (s \to t) \to (s' \to t')}\ \mathsf{C}_{\to} \qquad\qquad \frac{\vdash s \to s' \qquad \vdash t \to t'}{\vdash s \wedge t \to s' \wedge t'}\ \mathsf{C}_{\wedge}$$

$$\frac{\vdash s \to s' \qquad \vdash t \to t'}{\vdash \mathsf{A}(s\,\mathsf{U}\,t) \to \mathsf{A}(s'\,\mathsf{U}\,t')}\ \mathsf{C}_{\mathsf{AU}}$$

Setoid rewriting helps significantly in obtaining shorter and more linear proof scripts. For example, assume we know $\vdash t \to t'$. This allows us to reduce the claim $\vdash s \wedge t \to u$ to $\vdash s \wedge t' \to u$ with a single setoid rewrite. Performing the same reduction manually requires a non-linear derivation:

$$\dfrac{\dfrac{\dfrac{}{\vdash s \to s} \qquad \dfrac{}{\vdash t \to t'}}{\vdash s \wedge t \to s \wedge t'}\ \mathsf{C}_{\wedge} \qquad \vdash s \wedge t' \to u}{\vdash s \wedge t \to u}\ \mathsf{Trans}$$

In addition to rewriting, we use big conjunctions [3] to provide for an ND-style assumption management. Let A be a list of formulas. We abbreviate $\bigwedge_{s \in A} s$ as $\bigwedge A$ and define an **entailment relation** as follows:

$$A \vdash s := \vdash \bigwedge A \to s$$

$$\frac{s \in A}{A \vdash s}\ \mathsf{Asm} \qquad \frac{A, s \vdash t}{A \vdash s \to t}\ \mathsf{Intro} \qquad \frac{A \vdash t \qquad A, t \vdash s}{A \vdash s}\ \mathsf{Cut} \qquad \frac{s, A \vdash t}{A, s \vdash t}\ \mathsf{Rot}$$

$$\frac{A \vdash s_1 \quad \ldots \quad A \vdash s_n}{A, s_1 \to \ldots \to s_n \to t \vdash t}\ \mathsf{App} \qquad \frac{\vdash s_1 \to \ldots \to s_n \to t \qquad A \vdash s_1 \quad \ldots \quad A \vdash s_n}{A \vdash t}\ \mathsf{App_H}$$

Figure 4: Assumption Management

For the entailment relation, we use the derived rules shown in Figure 4. The rules in the upper row are realized with lemmas, whereas the rules in the lower row are realized with Ltac tactics. Besides the fact that the rules App and $\mathsf{App_H}$ are difficult to state as lemmas, applying the rules would be cumbersome. With Ltac we can use Coq's unification to find instances where n is minimal. The rule $\mathsf{App_H}$ is used to apply previously established facts, and we use unification to instantiate universally quantified variables in the leftmost premise. Using these derived rules we define a collection of tactics that correspond to the Coq tactics `intro`, `apply`, and `assert`.

In the formalization of the results from Section 11 the infrastructure for assumption management is mainly used to establish basic (e.g., propositional) facts. For the more high-level proofs (e.g., Lemmas 11.4 and 11.5) setoid rewriting is the main source of automation.

We remark that the assumption management rules shown in Figure 4 are not specific to the Hilbert system for CTL. The construction works for all Hilbert systems extending classical propositional logic. Let F be some type of formulas. We call a predicate $\vdash : F \to \mathsf{Prop}$ a **propositional system** if it satisfies the following conditions:

P1. There exist formulas $\bot : F$ and $\to : F \to F \to F$.

P2. If $\vdash s \to t$ and $\vdash s$, then $\vdash t$.

P3. $\vdash s \to t \to s$

P4. $\vdash ((u \to s \to t) \to (u \to s) \to u \to t)$

P5. $\vdash ((s \to \bot) \to \bot) \to s$

The rules in Figure 4 are available for every propositional system.

13 General Models

We now extend the soundness result for the Hilbert system from the class of finite models to the class of all models. This requires classical assumptions. We

use the axioms of excluded middle (XM) and dependent choice (DC) [19]:

$$\mathsf{XM} := \forall P : \mathsf{Prop}.\ P \vee \neg P$$

$$\mathsf{DC} := \forall X (R : X \to X \to \mathsf{Prop}).$$
$$(\forall x \exists y.\ R\,x\,y) \to \forall x \exists f : \mathbb{N} \to X.\ f\,0 = x \wedge \forall n.\ R\,(f\,n)(f\,(n+1))$$

The soundness proof for arbitrary models is completely analogous to the case for finite models, using axioms where the constructive reasoning from Section 5 fails.

The decidability premise of Lemma 3.1 trivially holds for all models in the presence of XM. Lemma 5.2 and Lemma 5.3 extend to infinite models if we use XM instead of decidability to justify case distinctions and DC instead of constructive choice to obtain infinite paths.

Theorem 13.1 (Soundness) Assume XM and DC. If $\vdash s$, then $\mathcal{M} \vDash s$ for all models $\mathcal{M}$.

Soundness for general models allows us to obtain the small model property of CTL from the informative completeness result.

Theorem 13.2 (Small Models) Assume XM and DC. If s is satisfiable, then s is satisfied in a model having at most $|s| \cdot 2^{4 \cdot |s|+2}$ states.

Proof Assume s is satisfiable. By Theorem 13.1 it is impossible that s is refutable. The claim follows with Theorem 11.7. ∎

We remark that the "bottom-up" fragment construction used in the proof of Lemma 8.2 is mainly motivated by the need to obtain "small" fragments. In our previous work [7], we obtained tree-shaped fragments using a construction that roughly corresponds to a "top-down" induction on the fulfillment relation. This eliminates all sharing and therefore yields exponentially larger fragments.

The semantics with respect to all models is essentially a shallow embedding of CTL into the type theory of Coq (cf. Section 5). Hence, we can show that XM and DC are not only sufficient but also necessary to prove soundness for general models.

Theorem 13.3 If $\vdash s$ implies $\mathcal{M} \vDash s$ for all models $\mathcal{M}$, then XM and DC are provable.

Proof To show XM it suffices so show $\neg\neg P \to P$ for all propositions P. We construct a model $\mathcal{M}$ with a single state w_0, such that $L_{\mathcal{M}}\,p\,w_0 \leftrightarrow P$. The claim is then immediate with the soundness of DN.

For DC, fix some serial relation $R : X \to X \to \mathsf{Prop}$ over some type X. This defines a model (the labeling is irrelevant). We have $\vdash \mathsf{E}(\bot\,\mathsf{R}\,\top)$. In the presence of XM, soundness yields an infinite path as required. ∎

14 Remark on Release

The semantics of the release modality can be defined in a number of classically equivalent but intuitionistically different ways (cf. [1], p. 256). The definition in Section 2 was chosen because one can show constructively, as we have done, that it coincides with the coinductive characterization on finite models. We now show that for other formulations, such as the commonly found

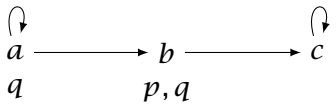
$$w \vDash \mathsf{A}(s\,\mathsf{R}\,t) \leftrightarrow \forall \pi.\pi\,0 = w \to (\forall n.\pi\,n \vDash t) \vee (\exists n.\pi\,n \vDash s \wedge \forall m \le n.\pi\,m \vDash t) \tag{14.1}$$

this is not the case. We construct a finite model such that if the equivalence above were to hold in this model, we could prove a constructively non-provable proposition called "limited principle of omniscience" [14].

$$\mathsf{LPO} := \forall f : \mathbb{N} \to \mathbb{B}.\ (\forall n.f\,n = \bot) \vee (\exists n.f\,n = \top)$$

Intuitively, LPO cannot be provable constructively since a constructive proof of LPO would correspond to a procedure that decides whether a given function f returns $\top$ for some argument. This would allow us to solve the halting problem (take f to be the function that checks whether a given Turing machine halts within n steps).

We construct a finite model $\mathcal{M}_3$ as follows:



Theorem 14.1 Assume (14.1) holds in $\mathcal{M}_3$. Then LPO is provable.

Proof It is easy to verify that $a \vDash_i \mathsf{A}(p\,\mathsf{R}\,q)$. Hence, we have $a \vDash \mathsf{A}(p\,\mathsf{R}\,q)$ by Lemma 5.4. We fix some function $f : \mathbb{N} \to \mathbb{B}$. We define a path π that starts at a and evaluates $f\,n$ before the n-th transition. The path leaves for b if $f\,n = \top$ and otherwise stays at a. The claim then follows with (14.1). ∎

Since the inductive semantics coincides with the path semantics on finite models, Theorem 14.1 shows that the inductive semantics and a path semantics using (14.1) as definition for always release do not coincide constructively on finite models.

15 Conclusion

In this paper we have investigated the metatheory of CTL from a constructive and formal-ization-oriented point of view. As it turns out, most of the metatheoretic

results about CTL can be obtained completely constructively. The notable exception is the soundness of the Hilbert system for the class of all models, which we have shown equivalent to XM and DC. Soundness for finite models is interesting from the constructive point of view. As shown in Section 14, the soundness result for finite models depends crucially on the particular definition of the release modality given in Section 2.

Algorithms play an important role for our constructive development even though we are not interested in executability. The soundness proof for finite models amounts to the verification of a model checking algorithm, and completeness of the Hilbert system is obtained using a decision method for satisfiability. These are just two of many instances where the decidability properties required for constructive proofs are obtained using algorithms that are interesting in their own right.

The original proofs [12, 8] of the results covered in this paper are presented in an informal manner, and the gap between the original proofs and the formalization is considerable. This is particularly true for the generation of Hilbert refutations. In the original completeness proofs only a few important lemmas about Hilbert provability are mentioned. In fact, this is also the case for the proof presented in Section 11. To formalize the construction of Hilbert refutations we require the reasoning infrastructure described in Section 12 and a library with about 100 lemmas about Hilbert provability. Although all these lemmas have fairly simple proofs, coming up with the reasoning infrastructure and the right selection of lemmas took considerable effort.

The construction of models from fragments in Section 8 is a minor variation of the construction in [8]. We choose to formalize the declarative "matrix" construction from the handbook article [8] rather than the original recursive construction [12] because it seems better suited for formalization. The inductive semantics avoids the need to argue about infinite paths and allows for a fairly concise correctness proof for the model construction.

Demos serve as the interface connecting the construction of fragments with the construction of refutations. In Section 1, we remarked that our demos are designed to give a good compromise between minimizing the effort for the model construction and minimizing the effort of constructing Hilbert refutations. With all results in place, we can expand on this remark. As it turns out, the definition of fulfillment for eventualities has a large impact on the complexity of the proofs. For the model construction one needs to construct fragments for fulfilled eventualities, and for the construction of Hilbert refutations one needs to establish closure properties for the collection of clauses that do not fulfill a given eventuality.

In Emerson's handbook article [8] the fulfillment condition for eventualities requires the existence of fragments embedded in the demo (there called pseudo-

Hintikka structure). Thus, the existence of small fragments for the model construction is immediate. However, the closure conditions for clauses with unfulfilled eventualities are more difficult to obtain. For $\Box \mathsf{A}(s \mathsf{U} t)^+ \in C$ one needs to show that if there were embedded fragments fulfilling $\mathsf{A}(s \mathsf{U} t)^+$ for every request of C, then there would also exist an embedded fragment for C and $\Box \mathsf{A}(s \mathsf{U} t)^+$. While it is easy to see that the fragments for the successors can be combined into a fragment for C and $\Box \mathsf{A}(s \mathsf{U} t)^+$, showing that the resulting fragment can be embedded in the demo requires nodes with identical labels to be merged.

For our definition of demos, the closure properties required to refute unfulfilled eventualities are easy to obtain. On the other hand, we have to spend some effort to obtain fragments for fulfilled eventualities. We believe that the declarative fragment construction described in the proof of Lemma 8.2 is easier to formalize than the iterative process of merging nodes with identical labels used by Emerson [8].

Another difference between our fulfillment predicates and the fragment test by Emerson is that we define fulfillment for eventuality literals. This is technically convenient because it ensures that all arising fragments are nontrivial. It also allows us to avoid the problem of eventualities that are not fulfilled but could be fulfilled locally in a consistent extension of the clause under consideration. Emerson solves this problem by defining demos only for maximal clauses. This requires a subformula universe that is closed under adding top-level negations. This also means that demos must be defined relative to a given subformula-closed clause which is not the case for our definition of demo.

We remark that the model construction in our Coq formalization is slightly more general than what is described in Section 8. Condition (F1) can be weakened to allow arbitrary base clauses as labels for internal nodes. This generalization is not used in the proofs presented here, but it is needed to prove the completeness of the Gentzen system employed in previous work [7]. The weakened fragment condition allows for a corresponding weakening of the fulfillment relations. It appears likely that one can obtain a simplified completeness proof for the Gentzen system by showing that it satisfies the refutation conditions arising from the weakened fulfillment relations.

The proofs in the Coq development involve a fair amount of detail much of which is omitted in the paper. Our formalization is carefully structured to complement the mathematical development. The formalization consists of about 2700 lines plus about 800 lines for a finite set library. For our formal development, we profit much from Ssreflect's handling of countable and finite types [17]. Countable types form the basis for our set library and finite types are used when we assemble the fragments of a demo into a finite model.

References

[1] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.

[2] Mordechai Ben-Ari, Amir Pnueli, and Zohar Manna. The temporal logic of branching time. *Acta Inf.*, 20(3):207–226, 1983.

[3] Yves Bertot, Georges Gonthier, Sidi Ould Biha, and Ioana Pasca. Canonical big operators. In Otmane Aït Mohamed, César Muñoz, and Sofiène Tahar, editors, *Theorem Proving in Higher Order Logics (TPHOLs 2008)*, volume 5170 of *LNCS*, pages 86–101. Springer, 2008.

[4] Kai Brünnler and Martin Lange. Cut-free sequent systems for temporal logic. *J. Log. Algebr. Program.*, 76(2):216–225, 2008.

[5] Edmund M. Clarke, E. Allen Emerson, and A. Prasad Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, 8(2):244–263, 1986.

[6] Christian Doczkal and Gert Smolka. Coq formalization accompanying this paper (Online Resource 1).

[7] Christian Doczkal and Gert Smolka. Completeness and decidability results for CTL in Coq. In G. Klein and R. Gamboa, editors, *Interactive Theorem Prving (ITP 2014)*, volume 8558 of *LNAI*, pages 226–241. Springer, 2014.

[8] E. Allen Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science: Formal Models and Sematics*, volume B, pages 995–1072. Elsevier, 1990.

[9] E. Allen Emerson. The beginning of model checking: A personal perspective. In Orna Grumberg and Helmut Veith, editors, *25 Years of Model Checking*, volume 5000 of *LNCS*, pages 27–45. Springer, 2008.

[10] E. Allen Emerson and Edmund M. Clarke. Characterizing correctness properties of parallel programs using fixpoints. In J. W. de Bakker and Jan van Leeuwen, editors, *Automata, Languages and Programming*, volume 85 of *LNCS*, pages 169–181. Springer, 1980.

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

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

[13] E. Allen Emerson and Chin-Laung Lei. Efficient model checking in fragments of the propositional mu-calculus (extended abstract). In *Proceedings, Symp. on Logic in Computer Science, 16-18 June 1986, Cambridge, Massachusetts, USA*, pages 267–278. IEEE Computer Society, 1986.

[14] Martín Escardó. Infinite sets that satisfy the principle of omniscience in any variety of constructive mathematics. *J. Symb. Log.*, 78(3):764–784, 2013.

[15] Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *J. Comput. System Sci.*, 18(2):194–211, 1979.

[16] Melvin Fitting. Modal proof theory. In Patrick Blackburn, Johan van Benthem, and Frank Wolter, editors, *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*, pages 85–138. Elsevier, 2007.

[17] Georges Gonthier, Assia Mahboubi, Laurence Rideau, Enrico Tassi, and Laurent Théry. A modular formalisation of finite group theory. In Klaus Schneider and Jens Brandt, editors, *Theorem Proving in Higher Order Logics (TPHOLs 2007)*, volume 4732 of *LNCS*, pages 86–101. Springer, 2007.

[18] Georges Gonthier, Assia Mahboubi, and Enrico Tassi. A small scale reflection extension for the Coq system. Research Report RR-6455, INRIA Saclay, 2008.

[19] Hugo Herbelin. A constructive proof of dependent choice, compatible with classical logic. In *27th Annual ACM/IEEE Symp. on Logic in Computer Science (LICS)*, pages 365–374. IEEE Computer Society, 2012.

[20] Mark Kaminski, Thomas Schneider, and Gert Smolka. Correctness and worst-case optimality of Pratt-style decision procedures for modal and hybrid logics. In Kai Brünnler and George Metcalfe, editors, *Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 2011)*, volume 6793 of *LNAI*, pages 196–210. Springer, 2011.

[21] Mark Kaminski and Gert Smolka. Terminating tableaux for hybrid logic with eventualities. In Jürgen Giesl and Reiner Hähnle, editors, *Automated Reasoning (IJCAR 2010)*, volume 6173 of *LNCS*, pages 240–254. Springer, 2010.

[22] Mark Kaminski and Gert Smolka. A goal-directed decision procedure for hybrid PDL. *J. Autom. Reason.*, 52(4):407–450, 2014.

[23] Martin Lange and Colin Stirling. Focus games for satisfiability and completeness of temporal logic. In *16th Annual ACM/IEEE Symp. on Logic in Computer Science (LICS)*, pages 357–365. IEEE Computer Society, 2001.

[24] Vaughan R. Pratt. Models of program logics. In *20th Annual Symp. on Foundations of Computer Science (FOCS'79)*, pages 115–122. IEEE Computer Society, 1979.

[25] Raymond M. Smullyan. *First-Order Logic*. Springer, 1968.

[26] Matthieu Sozeau. A new look at generalized rewriting in type theory. *J. Form. Reason.*, 2(1), 2009.

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