Constructive Completeness for Modal Logic with Transitive Closure

Christian Doczkal Gert Smolka

Published in Proc. of CPP 2012, Kyoto, Japan, LNCS 7679, Springer, 2012

Classical modal logic with transitive closure appears as a subsystem of logics used for program verification. The logic can be axiomatized with a Hilbert system. In this paper we develop a constructive completeness proof for the axiomatization using Coq with Ssreflect. The proof is based on a novel analytic Gentzen system, which yields a certifying decision procedure that for a formula constructs either a derivation or a finite countermodel. Completeness of the axiomatization then follows by translating Gentzen derivations to Hilbert derivations. The main difficulty throughout the development is the treatment of transitive closure.

Keywords: modal logic, completeness, decision procedures, constructive proofs, Hilbert Systems, Gentzen systems, Coq, Ssreflect.

1 Introduction

We are interested in a constructive and formal metatheory of decidable logics developed for program verification. In this paper we consider a logic K^+, which appears as a subsystem of PDL [8] and CTL [6]. K^+ extends the basic modal logic K with a modality for the transitive closure of the step relation. Our constructive account of K^+ is based on a Hilbert system and a class of models. Our main result is a constructive proof that the Hilbert system is complete for our class of models. The completeness proof comes in the form of a certifying decision procedure[1] that for a formula constructs either a derivation in the Hilbert system or a finite countermodel. This establishes the completeness of the Hilbert

[1] When we say decision procedure in this paper we do not claim that execution is feasible in practice.

system and the small model property of the logic. The main difficulty throughout the development is the treatment of transitive closure. The presence of the transitive closure modality for instance shows in the non-compactness of K^+.

We obtain our Hilbert system for K^+ from a Hilbert system for PDL [15]. Proving the completeness of the Hilbert system constructively turned out to be a challenge. The completeness proofs in the literature [15] are based on maximal consistent sets and are thus nonconstructive. The notable exception is a paper [1] by Ben-Ari, Pnueli, and Manna, where the completeness of a Hilbert system for UB (a logic subsuming K^+) is shown by extending a tableau-based decision procedure such that it yields a Hilbert refutation in case it fails to construct a model.

We refine the approach of Ben-Ari et al. [1] by replacing the tableau-based system with an analytic Gentzen system. In contrast to the tableau system, which constructs models, the Gentzen system constructs derivations, which can be translated to Hilbert derivations. Less directly, the Gentzen system also constructs models. The states of the models are obtained from the underivable sequents containing only subformulas of the input formula. Thus the Gentzen system gives us a certifying decision procedure that for an input formula s returns either a finite model of $\neg s$ or a derivation certifying that s is true in all models.

For propositional logic, the correspondence between tableau systems and Gentzen systems is immediate and well-known [17, 20]. For modal logic, the situation is less clear. Fitting gives a tableaux system for S4 [9] and a corresponding Gentzen system [10]. This system can be easily adapted to K, and the resulting system serves as the basis of our Gentzen system for K^+. Finding the missing rule for K^+ took some effort. Existing tableau systems for K^+ and related logics use local conditions to expand the tableau but also check global conditions like reachability on the constructed graph. For a corresponding Gentzen system these conditions must be reformulated as inference rules deriving valid sequents from valid sequents. For K^+ this leads to a rule we call compound rule. In contrast to the other rules, which are based on local properties, the compound rule in one step analyzes a strongly connected component of the search space for a model.

Once we have the Gentzen system for K^+, we translate Gentzen derivations into Hilbert derivations. To do so, we give for each Gentzen rule a function that for Hilbert derivations of the premises yields a Hilbert derivation of the conclusion. For all rules but the compound rule this is straightforward. The compound rule is the only rule dealing with the transitive closure modality. We handle the compound rule with an application of the induction-like Segerberg axiom of the Hilbert system to an invariant accounting for the strongly connected

component licensing the application of the compound rule. Ben-Ari et al. [1] use the Segerberg axiom in a similar way but their invariant did not work for us.

Our development [4] is carried out in Coq [19] with the Ssreflect [14] extension. We profit much from Ssreflect since our development requires computational finite types for subformulas, sequents, and sets of sequents.

We think that certifying decision procedures for logics used in program verification deserve more attention. The usual tableau-based decision procedures for such logics (e.g., [1, 16]) construct finite models for satisfiable formulas but do not construct certificates for unsatisfiable formulas. For K^+, we remedy this situation with a Gentzen system that constructs derivations for unsatisfiable formulas and models for satisfiable formulas.

In a previous paper [5] we give a constructive and formal proof of the decidability of an extension of K^+. There we rely on a pruning-based decision method and make no attempt to generate Hilbert proofs.

The paper is organized as follows. We first define the syntax, the models, and the Hilbert system for K^+. We then say a few things about finite types and finite sets in Ssreflect, which provide essential infrastructure for our development. Next we discuss how we formalize analytic Gentzen systems in Coq using Ssreflect. We then define a class of syntactic models we call demos. Demos represent the models produced by our Gentzen systems. Next we introduce propositional retracts, which we need for the formulation of the compound rule. We then define the Gentzen system for K^+ and prove that it yields a demo for every underivable sequent. Finally, we show how derivations in this system are translated to derivations in the Hilbert system.

2 Problem Statement

We assume a countable alphabet $\mathcal{P}$ of atomic propositions p and consider the formulas

$$s, t ::= \bot \mid p \mid s \to t \mid \Box s \mid \Box^+ s$$

To increase readability, we introduce a number of defined logical operations.

$$\neg s := s \to \bot \qquad s \wedge t := \neg(s \to \neg t) \qquad s \vee t := \neg s \to t \qquad \Box^* s := s \wedge \Box^+ s$$

Formulas are interpreted over transition systems consisting of a set of states $|\mathcal{M}|$, a transition relation $\to_{\mathcal{M}} \subseteq |\mathcal{M}| \times |\mathcal{M}|$, and a labeling $\Lambda_{\mathcal{M}} : |\mathcal{M}| \to 2^{\mathcal{P}}$. The **satisfaction relation** $\mathcal{M}, w \vDash s$ between transition systems, their states, and

formulas is defined as follows:

$$\begin{aligned}
\mathcal{M}, w &\not\models \bot \\
\mathcal{M}, w &\models p \iff p \in \Lambda_{\mathcal{M}}(w) \\
\mathcal{M}, w &\models s \to t \iff \mathcal{M}, w \models s \text{ implies } \mathcal{M}, w \models t \\
\mathcal{M}, w &\models \Box s \iff M, v \models s \text{ for all } v \text{ such that } w \to_{\mathcal{M}} v \\
\mathcal{M}, w &\models \Box^{+} s \iff M, v \models s \text{ for all } v \text{ such that } w \to^{+}_{\mathcal{M}} v
\end{aligned}$$

Here, $\to^{+}_{\mathcal{M}}$ is the transitive closure of the transition relation. In Coq, we represent transition systems as a record type:

```
Record ts : Type := TS {
  state :> Type ;
  trans : state -> state -> Prop ;
  label : state -> aprop -> Prop }.
```

We define the satisfaction relation as a function into Prop:

```
satisfies : forall (T : ts), T -> form -> Prop
```

Since we consider classical modal logic, we consider as **models** those transition systems, for which the satisfaction relation is stable under double negation.

```
Definition stable (X Y : Type) (R : X -> Y -> Prop) :=
  forall x y, ~ ~ R x y -> R x y.
Record model := Model { ts_of :> ts ; modelP : stable (satisfies ts_of) }.
```

A formula is **satisfiable** if it has a model, i.e., it holds at some state of some model. A formula s is **valid** if it holds at every state of every model. The Hilbert system for which we want to show completeness is shown in Figure 1. We write $\vdash s$ if s is provable in the Hilbert system. We represent the Hilbert system in Coq as an inductive predicate:

```
Inductive prv : form -> Prop :=
| r_mp s t     : prv (s ---> t) -> prv s -> prv t
| ax_k s t     : prv (s ---> t ---> s)
...
```

We can immediately show soundness.

Lemma 2.1 (Soundness) If s is provable, then s is valid.

Proof Induction on the derivation of $\vdash s$, using stability of the satisfaction relation to show the case for (DN). ∎

Our main result is the following.

Theorem 2.2 (Certified Decidability) For every formula s, we can either construct a finite model of $\neg s$ or a proof of s.

$$
\begin{array}{ll}
s \to t \to s & \text{(K)} \\
(s \to t \to u) \to (s \to t) \to (s \to u) & \text{(S)} \\
\neg\neg s \to s & \text{(DN)} \\
\Box(s \to t) \to \Box s \to \Box t & \text{(N)} \\
\Box^+(s \to t) \to \Box^+ s \to \Box^+ t & \text{(N+)} \\
\Box^+ s \to \Box s & \text{(T1)} \\
\Box^+ s \to \Box\Box^+ s & \text{(T2)} \\
\Box s \to \Box\Box^+ s \to \Box^+ s & \text{(T3)} \\
\Box s \to \Box^+(s \to \Box s) \to \Box^+ s & \text{(Segerberg)}
\end{array}
$$

$$
\frac{s \to t \qquad s}{t}\,\text{MP} \qquad \frac{s}{\Box s}\,\text{NEC} \qquad \frac{s}{\Box^+ s}\,\text{NEC}^+
$$

Figure 1: Hilbert System for Modal Logic with Transitive Closure

Corollary 2.3 (Completeness) If s is valid, then s is provable.

For the rest of this paper, we will mostly use mathematical notation to convey the ideas of the completeness proof. We present Coq code to show design choices and when the formal proof differs from the mathematical presentation. The reader is invited to browse the `coqdoc` proof outline and the source files [4].

3 Finite Types and Finite Sets in Ssreflect

Our formal proofs rely heavily on the Ssreflect extension to Coq, so we briefly describe the most important features we use. For technical details refer to [12, 13]. In Ssreflect, a counted type is a type with a boolean equality test and a choice operator for boolean predicates. A finite type is a counted type together with a finite list enumerating its elements. Finite types can be constructed from finite lists and finiteness is preserved by many type constructors. In particular, finite types are closed unter cartesian products and taking sets. Finite types come with boolean quantifiers `[forall x, p x]` and `[exists x, p x]` taking boolean predicates and returning booleans. The Coq syntax for the remaining operations we use is given in Figure 2.

4 Analytic Gentzen Systems in Coq

For a constructive proof of Theorem 2.2, we need a decision procedure which for a given formula eiher constructs a countermodel or a Hilbert proof. We will

|| &&	Boolean disjunction and conjunction
x \in xs	Generic membership operation for lists, sets, etc.
seq_sub xs	The finite type whose elements are the members of the list xs
{set X}	The finite type of sets over the finite type X.
:|: :&:	Union and intersection
[set x:X | A]	The set $\{ x \in X \mid A \}$ as an element of the type {set X}.

Figure 2: Ssreflect's Syntax for Finite Types and Finite Sets

use an analytic Gentzen system for this purpose. Before we develop the Gentzen system for K^+, we first show how we represent analytic Gentzen systems in Coq. As an example we will use an analytic Gentzen for basic modal logic K which is adapted from Fitting's tableau system for S4 [9]. This Gentzen system for K will also serve as the starting point for the development of our Gentzen System for K^+.

We represent sequents as finite sets of **signed formulas** [17] we call **clauses**. For instance, the sequent $p, q \Rightarrow u, v$ is represented as the clause $\{p^+, q^+, u^-, v^-\}$. The letter C ranges over clauses. A state **satisfies** a signed formula s^σ if it satisfies $\lfloor s^\sigma \rfloor$ where $\lfloor s^+ \rfloor = s$ and $\lfloor s^- \rfloor = \neg s$. A state satisfies a clause, if it satisfies all signed formulas it contains. Accordingly, the **associated formula** of a clause C is $\bigwedge_{s^\sigma \in C} \lfloor s^\sigma \rfloor$.

A sound Gentzen system is now a deduction system that derives unsatisfiable clauses from unsatisfiable clauses. This is in harmony with the conventional view that a sound Gentzen system derives valid sequents from valid sequents since validity of the formula $p \wedge q \to u \vee v$ associated with the sequent $p, q \Rightarrow u, v$ is equivalent to unsatisfiability of the formula we associate to the clause $\{p^+, q^+, u^-, v^-\}$

Figure 3 shows our Gentzen system for basic modal logic K. The notation $C; s^\sigma$ is to be read as $C \cup \{s^\sigma\}$. The notation $\mathcal{R}C$ denotes the set $\{ s^+ \mid \Box s^+ \in C \}$, which we call the **request** of C. The system is **analytic** in the sense that if a clause C is derivable, it has a derivation employing only signed subformulas of the formulas in C. For analytic Gentzen systems derivability of clauses is decidable provided that rule instantiation is decidable.

We want to use analytic Gentzen systems as certifying decision procedures. Hence, we fix an "input" formula s_0 and parameterize our definitions by this formula. This allows us to only consider the finitely many signed subformulas of s_0, which in turn allows us to leverage Ssreflect's finite types and sets.

Writing sub s0 for the list of subformulas of s0, we represent the signed formulas as the finite type F and clauses as sets over F.

```
Definition F := (bool * seq_sub (sub s0)) %type.
```

$$\frac{}{C; s^+; s^-}\ \text{Ax} \qquad \frac{}{C; \bot^+}\ \bot$$

$$\frac{C; s^+; t^-}{C; s \to t^-}\ \to^- \qquad \frac{C; s^- \qquad C; t^+}{C; s \to t^+}\ \to^+ \qquad \frac{\mathcal{R}C; s^-}{C; \Box s^-}\ \text{Jump}$$

Figure 3: Analytic Gentzen System for K

```
Definition clause := {set F}.
```

This allows most properties of clauses and sets of clauses to be expressed as boolean predicates and reasoned about classically. The rules of a Gentzen system will be represented as a boolean predicate

```
rule : {set clause} -> clause -> bool
```

Thus, the type of the rule predicate ensures that the system is analytic. Using a boolean predicate to represent rules also captures our intuition that rule instantiation should be decidable.

The set of derivable clauses is the least fixpoint of one-step derivability:

```
Definition onestep_derivable_from (S : {set clause}) :=
  [set C | [exists D : {set clause}, (D \subset S) && rule D C]].
```

One-step derivability is monotone so the least fixpoint exists and can be computed by applying the onestep_derivable_from function n times to the empty set where n is the size of the type clause. Hence derivability is decidable for any boolean rule predicate.

5 Demos

We now define a class of syntactic models we call demos [16]. The states of demos will be clauses and the definition will be such that every demo satisfies all the clauses it contains. Further, we will design our Gentzen system for K^+ such that it is complete for demos, i.e., the underivable clauses contain a demo satisfying all underivable clauses.

A clause H is called a **Hintikka clause** if it satisfies the following conditions:

H1. $\bot^+ \notin H$

H2. If $p^- \in H$, then $p^+ \notin H$

H3. If $(s \to t)^+ \in H$, then $s^- \in H$ or $t^+ \in H$

H4. If $(s \to t)^- \in H$, then $s^+ \in H$ and $t^- \in H$

To express the Hintikka property as a boolean predicate we have to take care of the fact that the Coq representation of our signed formulas consist of three parts: a formula, a proof that this formula is a subformula of s_0, and a boolean sign. With sc : s \in sub s0 we write the signed formula s^- as [F s; sc; false]. We define projections on the membership proofs in sub s0:

```
Lemma pll s t (sc : s ---> t \in sub s0) : s \in sub s0.
```

With this in place, we can express the Hintikka property as follows:

```
Definition Hcond (t : F) (H : {set F}) :=
  match t with
  | [F s ---> t; sc; true] =>
      ([F s; pll sc; false] \in H) || ([F t; plr sc; true] \in H)
  | ...
  end.
Definition hintikka (H : {set F}) : bool := [forall t in H, Hcond t H].
```

We now come to the definition of demos. The states of demos are Hintikka clauses. For the transition relation we extend the notion of **request** to K^+.

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

For every set S of clauses, we define a **transition relation** $\to_S \subseteq S \times S$ as follows: $C \to_S D$ iff $\mathcal{R}C \subseteq D$ and $\{C, D\} \subseteq S$. We write $\to_S^+$ for the transitive closure of $\to_S$. A set $\mathcal{D}$ of Hintikka clauses is a **demo** if every clause $C \in \mathcal{D}$ satisfies the following conditions:

D1. If $\Box t^- \in C$, then there is a clause $D \in \mathcal{D}$ such that $C \to_{\mathcal{D}} D$ and $t^- \in D$.

D2. If $\Box^+ t^- \in C$, then there is a clause $D \in \mathcal{D}$ such that $C \to_{\mathcal{D}}^+ D$ and $t^- \in D$.

A **demo for a clause** C is a demo that contains a clause that extends C. Let $\mathcal{D}$ be a demo. The **model associated with** $\mathcal{D}$ takes as states the clauses in $\mathcal{D}$ and as transition relation the relation $\to_{\mathcal{D}}$. Moreover, a state C of the associated model is labeled with atomic proposition p if and only if $p \in C$.

Lemma 5.1 Let $\mathcal{D}$ be a demo, $\mathcal{M}$ its associated model, and s a formula. If $s^\sigma \in C \in \mathcal{D}$, then $\mathcal{M}, C \vDash \lfloor s^\sigma \rfloor$.

Proof Induction on s.

Note that, in contrast to the demo condition for $\Box t^-$, the demo condition for $\Box^+ t^-$ is non-local in the sense that it may take an arbitrary number of transitions to reach the clause containing t^-. We call a formula of the form $\Box^+ t^-$ an **eventuality** and say that a clause D satisfying (D2) **fulfills** the eventuality. Coming up with a Gentzen system whose underivable Hintikka clauses satisfy (D2) requires some work.

$$\frac{}{\{C\} \rhd C} \qquad \frac{}{\emptyset \rhd C}\,\bot \in C \text{ or } \{s^+, s^-\} \subseteq C$$

$$\frac{\mathcal{D}_1 \rhd C; s^- \qquad \mathcal{D}_2 \rhd C; t}{\mathcal{D}_1 \cup \mathcal{D}_2 \rhd C; s \to t^+} \qquad \frac{\mathcal{D} \rhd C; s; t^-}{\mathcal{D} \rhd C; s \to t^-}$$

Figure 4: Retracts

6 Propositional Retracts

We now begin the development of our Gentzen system vor K^+. To ease our notation, we will from now on write just s for a positively signed formula s^+.

We will design our system such that the set of underivable clauses will contain a demo for every underivable clause. Since demos only contain Hintikka clauses, we need to ensure that for every underivable clause there is an underivalbe Hintikka extension, i.e, an underivable Hintikka clause containing C.

We call a set of clauses $\mathcal{D}$ a **retract** of C, writen $\mathcal{D} \rhd C$, if C is derivable from $\mathcal{D}$ using propositional reasoning. More precisely, a retract of C is the frontier of a backwards derivation starting at C using the propositional rules from Figure 3. See Figure 4 for the precise definition of this notion. A retract $\mathcal{D}$ of C is called a **Hintikka retract** if every $H \in \mathcal{D}$ is a Hintikka extension of C.

Lemma 6.1 For every clause one can compute a Hintikka retract.

Proof Let C be a clause. We prove this by induction on the number of signed formulas not in C. If C is a Hintikka clause, then $\{C\}$ is a Hintikka retract of C. If C contains a formula both with positive and negative sign or $\bot^+$, we pick the empty retract. Otherwise, there is some implication $s \to t^\sigma$ whose Hintikka condition is not satisfied. We consider the case where $\sigma = -$; the other case is similar. We have $C \subsetneq C; s; t^-$ so by induction hypothesis we can compute a Hintikka retract H for $C; s; t^-$ which is also a Hintikka retract for C.

On the Coq side, this amounts to showing.

```
Lemma saturation C : { H | hretract H C }
```

Here, `{ H | retract H D}` is the type of dependent pairs of sets of clauses `H` and proofs that these are Hintikka retracts of `C`. We define a function which computes a Hintikka retract for every clause, by projecting out the first component of that pair:

```
Definition dret C := proj1_sig (saturation C)
```

We refer to the Hintikka retract computed by dret as the **default retract** of C. We now have the first rule of our Gentzen system for K^+, the **retract rule**:

$$\frac{C_1 \dots C_n}{C}\{C_1, \dots, C_n\} \text{ is the default Hintikka retract for } C$$

Note that the use of the default retract in the definition of the retract rule essentially fixes a single strategy in which the propositional rules can be applied. This allows us to express the retract rule as a boolean predicate without the tedium of expressing the retract relation as a boolean predicate. Mathematically, any Hintikka retract would suffice since we do not make use of any special properties of the default retract.

Lemma 6.2 (Extension) If C is underivable, it has an underivable Hintikka extension.

Note that all clauses that are derivable using propositional reasoning have an empty default retract. Hence, the retract rule allows us to handle propositional reasoning in a single step and completely separately from modal reasoning.

7 The Gentzen System for K^+

The Gentzen system consisting of the retract rule and the jump rule from Figure 3 is already complete for formulas not involving $\Box^+$.

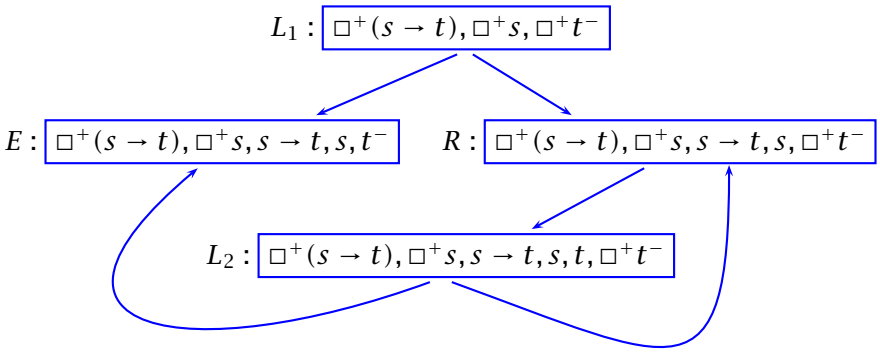
Lemma 7.1 The underivable Hintikka clauses satisfy condition (D1).

Hence, all we need is a rule that establishes (D2). A first candidate for a rule for the $\Box^+$ modality might be the following **split-jump rule**:

$$\frac{\mathcal{R}C; s^- \qquad \mathcal{R}C; \Box^+ s^-}{C; \Box^+ s^-}$$

The rule is motivated by the two ways in which we can satisfy the eventuality $\Box^+ s^-$: we either satisfy s^- at the next state (left branch of the rule) or we delay fulfilling the eventuality (right branch). However, the resulting Gentzen system would be incomplete as witnessed by the following example.

Example 7.2 The clause $\{\Box^+(s \to t), \Box^+ s, \Box^+ t^-\}$ corresponding to the axiom (N^+) is an example of an unsatisfiable but underivable clause. Alternating between applying the split-jump and the retract rule yields a looping derivation that can be visualized as a graph:



Here, $\{L_2\}$ is the unique Hintikka retract of R. The clause E can be proved by the retract rule, but the derivation "loops" around the clause L_2. □

To obtain a complete deduction system, we need a stronger rule for eventualities. As we have seen in the previous example, applying the retract and the jump rule can lead to looping derivations on unsatisfiable clauses. Thus, our rule for eventualities is a generalization of the split-jump rule that allows for certain looping derivations.

A **compound** is a triple $(s, \mathcal{L}, \mathcal{E})$ consisting of a formula s and two sets $\mathcal{L}$ and $\mathcal{E}$ of clauses such that every clause $C \in \mathcal{L}$ satisfies the following conditions:

1. $\Box^+ s^- \in C$
2. $\mathcal{R}C; s^- \in \mathcal{E}$
3. $\mathcal{L} \cup \mathcal{E}$ contains all clauses of the default retract of $\mathcal{R}C; \Box^+ s^-$.

If $(s, \mathcal{L}, \mathcal{E})$ is a compound, we call $\mathcal{L}$ the **loop** and the clauses in $\mathcal{E}$ **exit clauses**. We now formulate the remaining rule and call it **compound rule**:

$$\frac{C_1 \quad \cdots \quad C_n}{C} \quad (s, \mathcal{L}, \{C_1, \ldots, C_n\}) \text{ is a compound and } C \in \mathcal{L}$$

Starting from a clause C containing $\Box^+ s^-$, instances of the compound rule can be generated as follows: We set $\mathcal{L} := \{C\}$ and $\mathcal{E} := \emptyset$ and start by applying the split-jump rule to C. The left premise and those clauses in the default retract of $\mathcal{R}C; \Box^+ s^-$ that we want to derive by other means are added to $\mathcal{E}$. The remaining clauses that are not yet in $\mathcal{L}$ are added to $\mathcal{L}$, and we continue by applying the split-jump rule to these clauses. This process must terminate with a compound $(s, \mathcal{L}, \mathcal{E})$ since there are only finitely many clauses that can be added to

$\mathcal{L}$. The fact that one does not need to apply the split-jump rule to clauses that are already in $\mathcal{L}$ allows for looping derivations. The compound rule can easily be expressed as a boolean predicate.

Example 7.3 We can derive the clause $\{\Box^+(s \to t), \Box^+ s, \Box^+ t^-\}$ from Example 7.2 using the compound rule. Taking L_1, L_2, and E as in Example 7.2, the triple $(t, \{L_1, L_2\}, \{E\})$ is a compound and E can be derived using the retract rule. □

Lemma 7.4 The underivable Hintikka clauses satisfy condition (D2).

Proof Let $\mathcal{U}$ be the set of underivable Hintikka clauses and $\Box^+ s^- \in C \in \mathcal{U}$. We define

$$\mathcal{L} := \{\, C' \in \mathcal{U} \mid \Box^+ s^- \in C \text{ and } C \to_{\mathcal{U}}^* C' \,\}$$

Let $\mathcal{D}$ be the set of derivable clauses. Since $C \in \mathcal{L}$ and C is not derivable, we know that $(s, \mathcal{L}, \mathcal{D})$ is not a compound. However, compound conditions (1) and (3) hold. Hence there is a clause $D \in \mathcal{L}$ such that $\mathcal{R}D; s^- \notin \mathcal{D}$. Thus $\mathcal{R}D; s^-$ is not derivable and therefore has a Hintikka extension $D' \in \mathcal{U}$ (Lemma 6.2). Thus $C \to_{\mathcal{U}}^* D \to_{\mathcal{U}} D'$ and $t^- \in D'$.

We now have a complete Gentzen system for K^+.

Theorem 7.5 The set of underivable Hintikka clauses is a demo for every underivable clause. Thus every underivable clause is satisfiable.

Proof Follows with Lemma 7.1, Lemma 7.4, and Lemma 5.1.

8 Translating Gentzen Derivations to Hilbert Proofs

So far we did not consider soundness of the Gentzen system. However, soundness of the Gentzen system will follow as a by-product of a translation of Gentzen derivations to Hilbert proofs, which we need anyway to show completeness of the Hilbert system.

We associate with every signed formula and every clause a formula as defined in Section 4. If a clause appears in the place of a formula, the clause is to be understood as notation for its associated formula.

We aim for a translation theorem of the following form:

Theorem 8.1 If C is derivable in the Gentzen system, then $\vdash \neg C$.

We call a Hilbert proof of $\neg C$ a (Hilbert) **refutation** of C. A constructive proof of the translation theorem can be seen as a translation from Gentzen derivations

to Hilbert refutations. To prove this theorem by induction on the Gentzen derivation, we need a number of lemmas corresponding to the rules of the Gentzen system.

For the retract rule we have the following Lemma which we will also use in the translation of the compound rule.

Lemma 8.2 If $\mathcal{D} \triangleright C$, then $\vdash C \to \bigvee \mathcal{D}$.

Proof Induction on the definition of retract.

For the jump rule we have:

Lemma 8.3 If $\vdash \neg(\mathcal{R}C; s^-)$, then $\vdash \neg(C; \Box s^-)$.

Proof We reason as follows:

1.	$\vdash \neg(\mathcal{R}C; s^-)$	assumption
2.	$\vdash \mathcal{R}C \to s$	propositional reasoning
3.	$\vdash \Box\mathcal{R}C \to \Box s$	NEC, N
4.	$\vdash C \to \Box s$	$\vdash C \to \Box\mathcal{R}C$
5.	$\vdash \neg(C; \Box s^-)$	propositional reasoning

Note that the Gentzen system consisting only of the retract rule and the jump rule corresponds very closely to the Gentzen system in 3 and is complete for formulas not involving $\Box^+$. So giving a constructive completeness proof for K is not difficult. The difficulty of giving a constructive completeness proof for K^+ lies entirely in the treatment of transitive closure.

9 Generating Hilbert Proofs

Before we turn to the translation of the compound rule, we first note that to formalize this kind of translation argument, we need to develop some infrastructure for generating Hilbert proofs in Coq as finding Hilbert proofs in the bare Hilbert system can be a difficult task. Of course, Hilbert systems are well understood and there are many techniques to come up with Hilbert proofs. We merely mention two techniques that are easy to set up and help significantly in generating Hilbert proofs in Coq.

It is well known that the entailment relation (i.e., `prv (s ---> t)` in Coq) defines a preorder on formulas and that the logical operations have certain **monotonicity properties** with respect to this preorder. For example:

$$\frac{\vdash s' \to s \qquad \vdash t \to t'}{\vdash (s \to t) \to (s' \to t')} \qquad \frac{\vdash s \to s' \qquad \vdash t \to t'}{\vdash (s \wedge t) \to (s' \wedge t')} \qquad \frac{\vdash s \to s'}{\vdash \Box s \to \Box s'} \qquad \ldots$$

$$\frac{s \in xs}{\bigwedge xs \to s} \qquad \frac{\bigwedge s :: xs \to t}{\bigwedge xs \to s \to t} \qquad \frac{\bigwedge nil \to s}{s}$$

$$\frac{\bigwedge xs \to s \to t \qquad \bigwedge xs \to s}{\bigwedge xs \to t}$$

Figure 5: Assumption Lemmas

We make these monotonicity properties known to Coq's extended (setoid) rewriting tactic [18]. This allows us to freely rewrite with the entailment relation to strengthen claims or to weaken assumptions, thereby contracting many mechanical reasoning steps into a single rewrite.

Another major hindrance, in particular to finding propositional proofs, is the lack of assumption management in the Hilbert system. However, we can simulate natural deduction style reasoning inside the Hilbert system using a few lemmas on big conjunctions.

In Coq, we realize big conjunctions and disjunctions over lists of formulas using Ssreflect's canonical big operators [2]. We also need big conjunctions indexed by finite sets, which we represent by fixing an arbitrary enumeration of the elements. The most prominent use of this construction is for the associated formulas of clauses. For a big conjunction of the form $\bigwedge_{x \in A} x$ we will just write $\bigwedge A$ and likewise for disjunctions.

The lemmas we use to simulate natural deduction style reasoning are displayed as rules in Figure 5. Here xs ranges over lists of formulas and $::$ is the cons operator. Note that there are no rules corresponding to NEC and NEC$^+$ since these rules would clearly be unsound in the presence of assumptions. This essentially restricts reasoning with assumptions to the propositional fragment, which is sufficient for our purposes.

Building on these rules, we define a set of tactics simulating the behavior of basic Coq tactics like `intros` and `apply` on the level of Hilbert proofs. For additional detail, we refer the reader to the theory files. Using setoid rewriting and these tactics the various modal logic lemmas that we need for our translation proof can be proved easily. Those modal logic lemmas to which we will refer explicitly can be found in Figure 6.

$\vdash \Box s \vee \Box t \to \Box(s \vee t)$ (D2)

$\vdash \Box^* s \to \Box\Box^* s$ (S1)

$\vdash \Box^+ s \to \Box\Box^* s$ (S2)

$\vdash C \to \Box\mathcal{R}C$ (R1)

If $\vdash s \to t$, then $\vdash \Box^+ s \to \Box^+ t$ (REG$^+$)

Figure 6: Basic Modal Logic Lemmas

10 Translation Method for the Compound Rule

We now turn to the last missing piece in our formal completeness proof, the translation method for the compound rule.

Lemma 10.1 Let $(s, \mathcal{L}, \mathcal{E})$ be a compound such that $\vdash \neg D$ for all $D \in \mathcal{E}$. Then for every $C \in \mathcal{L}$, we have $\vdash \neg C$.

Proof Let $C_0 \in \mathcal{L}$. It suffices to show $\vdash C_0 \to \Box^+ s$ since $\Box^+ s^- \in C_0$ by the definition of compound. We define

$$I := \Box^* s \vee \bigvee_{C \in \mathcal{L}} \mathcal{R}C$$

and show the following properties of I:

(i) $\vdash C_0 \to \Box I$

(ii) $\vdash I \to s$

(iii) $\vdash I \to \Box I$

Once we have shown (i) to (iii), we can finish the proof as follows:

1. $\vdash \Box^+(I \to \Box I)$ (iii), NEC$^+$
2. $\vdash \Box I \to \Box^+ I$ Segerberg
3. $\vdash \Box I \to \Box^+ s$ (ii), REG$^+$
4. $\vdash C_0 \to \Box^+ s$ (i)

For (i - iii) we reason as follows:

(i) We have:

1. $\vdash C_0 \to \Box\mathcal{R}C_0$ (R1)
2. $\vdash C_0 \to \Box I$ monotonicity

(ii) It suffices to show $\vdash \mathcal{R}C \to s$ for every $C \in \mathcal{L}$. For every such C we have:

1. $\vdash \neg(\mathcal{R}C; s^-)$ — Def. compound, assumption
2. $\vdash \mathcal{R}C \to s$ — propositional reasoning

(iii) We show that every disjunct of I implies $\Box I$. By (S1) it suffices to show $\vdash \mathcal{R}C \to \Box I$ for every $C \in \mathcal{L}$. Let $C \in \mathcal{L}$ and let $\mathcal{D} \subseteq \mathcal{L} \cup \mathcal{E}$ be the default retract of $\mathcal{R}C; \Box^+ s^-$.

1. $\vdash \mathcal{R}C; \Box^+ s^- \to \bigvee \mathcal{D}$ — Lemma 8.2
2. $\vdash \mathcal{R}C; \Box^+ s^- \to \bigvee(\mathcal{D} \cap \mathcal{L})$ — $\vdash \neg D$ for all $D \in \mathcal{E}$
3. $\vdash \mathcal{R}C; \Box^+ s^- \to \bigvee_{L \in \mathcal{L}} L$ — propositional reasoning
4. $\vdash \mathcal{R}C; \Box^+ s^- \to \bigvee_{L \in \mathcal{L}} \Box \mathcal{R}L$ — (R1), monotonicity
5. $\vdash \mathcal{R}C; \Box^+ s^- \to \Box \bigvee_{L \in \mathcal{L}} \mathcal{R}L$ — (D2)
6. $\vdash \mathcal{R}C \to \Box^+ s \vee \Box \bigvee_{L \in \mathcal{L}} \mathcal{R}L$ — propositional reasoning
7. $\vdash \mathcal{R}C \to \Box(\Box^* s \vee \bigvee_{L \in \mathcal{L}} \mathcal{R}L)$ — (S2), (D2)
8. $\vdash \mathcal{R}C \to \Box I$ — Def. I

Now we can prove the translation theorem.

Proof (of Theorem 8.1) Let C be derivable. We prove the claim by induction on the derivation of C. The case for the compound rule follows immediately with Lemma 10.1. The cases for the jump rule and the retract rule follow with Lemma 8.3 and Lemma 8.2 respectively.

Proof (of Theorem 2.2) Derivability in our Gentzen system is decidable, so we consider two cases:
The clause $\{s^-\}$ *is derivable.* By Lemma 8.1, we have $\vdash \neg\neg s$ and hence $\vdash s$.
The clause $\{s^-\}$ *is underivable.* By Lemma 7.5, the set of underivable clauses over the subformulas of s yields a model for $\{s^-\}$ and hence for $\neg s$.

Note that in the proof above, the size of the countermodel can be bounded by 2^{2n} where n is the number of subformulas of s. Hence, we have also shown that K^+ has the small model property.

Organizing the proof as we do, we obtain a development of rather moderate size. It consists of less than 1000 lines specific to our proof, about half of which are Hilbert infrastructure and a collection of basic modal logic lemmas. On top of this we need a couple of hundred lines of generic constructions like the fixpoint computation described in Section 4.

11 Related Work

Only after submitting the initial version of this paper, we became aware of the work of Brünnler and Lange [3] presenting analytic sequent calculi for LTL and CTL. In their calculi one can focus on an eventuality formula and keep a history of the contexts in which a rule has been applied to this eventuality. If an eventuality occurs in a context that is already in the history, the sequent is provable.

Adapting Brünnler and Lange's rules to K^+, we obtain a system where one of the eventuality formulas in a clause can be annotated with a finite set of plain, i.e., annotation-free clauses. The rules for $\Box^+$ then look as follows:

$$\frac{\mathcal{R}C; s^- \qquad \mathcal{R}C; \Box^+_{\mathcal{R}C} s^-}{C; \Box^+ s^-} \qquad \frac{\mathcal{R}C; s^- \qquad \mathcal{R}C; \Box^+_{H;\mathcal{R}C} s^-}{C; \Box^+_H s^-} \qquad \frac{}{C; \Box^+_{H;\mathcal{R}C} s^-}$$

An annotated eventuality $\Box^+_H s^-$ is satisfied at a state w if there is a path from w to a state satisfying s^- that has at least one transition and after the first transition no state on the path satisfies a clause in H. The system consisting of the rules above and the rules from Figure 3 is sound for this semantics and complete for plain clauses.

LTL and CTL can express the semantics of annotated eventualities as a formula using the until operator. So given complete Hilbert systems for LTL [11] or CTL [7] it should be possible to translate derivations in Brünnler and Lange's calculi to Hilbert proofs. Unlike LTL and CTL, K^+ cannot express the semantics of an annotated eventuality as a formula. Hence, it is not clear how to translate the rules above to Hilbert refutations in the Hilbert system.

The motivation for our Gentzen system was the need for a simple inductive characterization of unsatisfiability that can be translated to Hilbert refutations to constructively show the completeness of the Hilbert system. In fact, our monolithic compound rule allows us to directly read off the instantiation of the Segerberg axiom. So while for constructive completeness proofs for Hilbert systems for LTL and CTL history-based Gentzen systems seem promising, a system with a compound rule appears essential for weaker logics like K^+.

Acknowledgments.

We thank Chad Brown for many helpful discussions and the suggestion to consider $\Box^+ s$ rather than $\Box^* s$ as primitive. We also thank the anonymous referees for their helpful comments.

References

[1] Ben-Ari, M., Pnueli, A., Manna, Z.: The temporal logic of branching time. Acta Inf. 20, 207–226 (1983)

[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.: Coq formalization accompanying this paper, `www.ps.uni-saarland.de/extras/cpp12/`

[5] Doczkal, C., Smolka, G.: Constructive formalization of hybrid logic with eventualities. In: Jouannaud, J.P., Shao, Z. (eds.) Certified Programs and Proofs. LNCS, Springer (Dec 2011)

[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] Fischer, M.J., Ladner, R.E.: Propositional dynamic logic of regular programs. J. Comput. System Sci. pp. 194–211 (1979)

[9] Fitting, M.: Intuitionistic logic, model theory and forcing. Studies in Logic, North-Holland Pub. Co. (1969)

[10] Fitting, M.: Proof Methods for Modal and Intuitionistic Logics. Reidel (1983)

[11] Gabbay, D.M., Pnueli, A., Shelah, S., Stavi, J.: On the temporal analysis of fairness. In: Abrahams, P.W., Lipton, R.J., Bourne, S.R. (eds.) POPL. pp. 163–173. ACM Press (1980)

[12] Garillot, F., Gonthier, G., Mahboubi, A., Rideau, L.: Packaging mathematical structures. In: Berghofer, S., Nipkow, T., Urban, C., Wenzel, M. (eds.) TPHOLs. LNCS, vol. 5674, pp. 327–342. Springer (2009)

[13] 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)

[14] 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/

[15] Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. The MIT Press (2000)

[16] Kaminski, M., Schneider, T., Smolka, G.: Correctness and worst-case optimality of Pratt-style decision procedures for modal and hybrid logics. In: Brünnler, K., Metcalfe, G. (eds.) TABLEAUX 2011. LNCS (LNAI), vol. 6793, pp. 196–210. Springer (2011)

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

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

[19] The Coq Development Team: coq.inria.fr

[20] Troelstra, A.S., Schwichtenberg, H.: Basic proof theory (2nd ed.). Cambridge University Press, New York, NY, USA (2000)