Completeness and Decidability of Converse PDL in the Constructive Type Theory of Coq

Christian Doczkal
Univ Lyon, CNRS, ENS de Lyon, UCB Lyon 1, LIP
France
christian.doczkal@ens-lyon.fr

Joachim Bard
Saarland University
Germany

Abstract

The completeness proofs for Propositional Dynamic Logic (PDL) in the literature are non-constructive and usually presented in an informal manner. We obtain a formal and constructive completeness proof for Converse PDL by recasting a completeness proof by Kozen and Parikh into our constructive setting. We base our proof on a Pratt-style decision method for satisfiability constructing finite models for satisfiable formulas and pruning refutations for unsatisfiable formulas. Completeness of Segerberg's axiomatization of PDL is then obtained by translating pruning refutations to derivations in the Hilbert system. We first treat PDL without converse and then extend the proofs to Converse PDL. All results are formalized in Coq/Ssreflect.

CCS Concepts • **Theory of computation → Modal and temporal logics**; **Constructive mathematics**; **Type theory**;

Keywords Propositional Dynamic Logic, Hilbert system, Completeness, Decidability, Constructive proofs, Pratt-style pruning, Coq, Ssreflect

ACM Reference Format:
Christian Doczkal and Joachim Bard. 2018. Completeness and Decidability of Converse PDL in the Constructive Type Theory of Coq. In *Proceedings of 7th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP'18).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3167088

1 Introduction

Propositional Dynamic Logic (PDL) [12, 15] is a modal logic developed for reasoning about programs with applications for instance in knowledge representation [3]. The modalities of PDL are given by regular programs (i.e., regular expressions with tests) describing binary relations on states. The formula $[\alpha]\varphi$ is satisfied by some state if all α-reachable states satisfy φ. The language of programs includes a reflexive transitive closure operation (α^*) causing PDL to be non-compact. Converse PDL (CPDL), also defined in [12], extends PDL with a converse operation on programs. Both PDL and CPDL have the small-model property [12] and are EXPTIME-complete [12, 21]. Axiomatizations of PDL and CPDL were first given by Segerberg [22] and shown complete independently by Gabbay [13] and Parikh [19].

Our main result is a machine-checked constructive proof that for every PDL or CPDL formula φ one can either construct a proof of $\neg\varphi$ from the respective axioms or a model of bounded size satisfying the formula. Completeness and the small-model property as well as decidability of satisfiability, validity and provability then follow as corollaries. The form of constructive completeness result established here is more informative than a classical completeness result in the sense that it provides an algorithm constructing proofs for valid formulas rather than merely showing the existence of such proofs.

The completeness proofs for (C)PDL in the literature either use non-standard canonical models and filtration [15, 19] or construct "canonical" models inside a finite syntactic universe [18]. Both techniques are non-constructive in the sense that they assume (at least) logical decidability of provability (i.e., $\vdash \varphi \lor \nvdash \varphi$ for all φ). While logical decidability follows from (computational) decidability, the easiest way to show decidability of provability is via completeness.

In order to obtain a constructive completeness result, we base the completeness proof on a decision method for satisfiability. We employ a variant of pruning [17, 21] extended to provide refutations for unsatisfiable formulas in addition to constructing models for satisfiable formulas. Translating these pruning refutations to proofs in the Hilbert system then yields the desired completeness result. The use of pruning as decision procedure underlying the completeness result naturally leads to a factorization of the proof into an algorithmic part for the decision method, a semantic argument for the model construction, and a syntactic translation from pruning refutations to Hilbert refutations. We believe that the constructive nature of the proof and the clear separation of concerns makes the proof particularly easy to follow.

The pruning-based approach is inspired by completeness proofs for the branching time logics UB ("unified branching") [5] and Computation Tree Logic (CTL) [11] and was

CPP'18, January 8–9, 2018, Los Angeles, CA, USA

ACM ISBN 978-1-4503-5586-5/18/01…$15.00
https://doi.org/10.1145/3167088

employed by the first author to obtain formal and constructive completeness proofs of Hilbert and sequent systems for CTL [8, 10]. For the construction of models using pruning, we follow the presentation of pruning in [16]. The translation from pruning refutations to Hilbert proofs builds on ideas in [18]. For CPDL, we show that the Hilbert system validates the conversion of formulas to converse normal form and then use pruning for converse normal formulas. This isolates the treatment of converse to a few select places in the proof. We are not aware of any other completeness proof treating converse this way.

The mathematical proofs described in this paper are constructive and designed with the formalization in mind. The accompanying Coq development [9] follows the proofs outlined in the paper and provides the details elided in the paper. The development is carried out using the Ssreflect [14] proof language and the Mathematical Component Libraries [25]. Our development builds upon the formal and constructive completeness proof for test-free PDL presented in [4].

The rest of the paper is organized as follows. Section 2 recalls the syntax, semantics, and Hilbert system of PDL and describes their representation in Coq. Sections 3 to 6 describe the constructive completeness result for PDL. Section 7 describes the changes required to extend our results to CPDL. Section 8 provides a high-level overview of the accompanying Coq development [9].

2 Propositional Dynamic Logic

We fix a countably infinite type of atomic programs $\mathcal{A}$ and a countably infinite type of atomic propositions $\mathcal{P}$. The letter a ranges over atomic programs and the letter p ranges over atomic propositions. We consider Propositional Dynamic Logic (PDL) with the following syntax for *programs* (denoted $\alpha, \beta, \ldots$) and *formulas* (denoted $\varphi, \psi, \ldots$).

$$\alpha, \beta := a \mid \alpha + \beta \mid \alpha\beta \mid \alpha^* \mid \varphi? \qquad (a : \mathcal{A})$$
$$\varphi, \psi := p \mid \bot \mid \varphi \to \psi \mid [\alpha]\varphi \qquad (p : \mathcal{P})$$

We define the remaining logical operations in terms of this syntax (i.e., $\neg\varphi := \varphi \to \bot$, $\varphi \wedge \psi := \neg(\varphi \to \neg\psi)$, $\langle\alpha\rangle\varphi := \neg[\alpha]\neg\varphi, \ldots$).[1] We write $|\varphi|$ and $|\alpha|$ for the sizes of formulas and programs (i.e., the size of the syntax tree).

Formulas are interpreted over transition systems where the states are labeled with atomic propositions and the transitions are labeled with atomic programs. A transition system $\mathcal{M}$ hence consists of

- A (possibly infinite) type $|\mathcal{M}|$ whose elements are called *states*
- A *transition relation* $\overset{a}{\Rightarrow}_{\mathcal{M}} : |\mathcal{M}| \to |\mathcal{M}| \to \mathsf{Prop}$ for every $a : \mathcal{A}$
- A *labeling* $L_{\mathcal{M}} : \mathcal{P} \to |\mathcal{M}| \to \mathsf{Prop}$.

[1] While $\varphi \to \psi$ could be defined as $[\varphi?]\psi$, we include it in the syntax since this allows us to define the Hilbert system with minimal reliance on defined logical operations.

$$\begin{aligned}
w \models p &:= L_{\mathcal{M}}\, p\, w \\
w \models \bot &:= \bot \\
w \models \varphi \to \psi &:= w \models \varphi \to w \models \psi \\
w \models [\alpha]\varphi &:= \forall v. w \overset{\alpha}{\Rightarrow} v \to v \models \varphi \\
\\
w \overset{a}{\Rightarrow} v &:= w \overset{a}{\Rightarrow}_{\mathcal{M}} v \\
w \overset{\alpha+\beta}{\Rightarrow} v &:= w \overset{\alpha}{\Rightarrow} v \vee w \overset{\beta}{\Rightarrow} v \\
w \overset{\alpha\beta}{\Rightarrow} v &:= \exists u. w \overset{\alpha}{\Rightarrow} u \wedge u \overset{\beta}{\Rightarrow} v \\
w \overset{\alpha^*}{\Rightarrow} v &:= w \,(\overset{\alpha}{\Rightarrow})^*\, v \\
w \overset{\varphi?}{\Rightarrow} v &:= w = v \wedge w \models \varphi
\end{aligned}$$

Figure 1. Semantics of formulas and programs

In the following, we write $\mathcal{M}$ for transitions systems as well as their underlying type of states.

Let $\mathcal{M}$ be a transition system. We define a satisfaction relation between states w of $\mathcal{M}$ and formulas φ, written $w \models \varphi$, and an interpretation of programs α as binary relations on $\mathcal{M}$, written $\overset{\alpha}{\Rightarrow}$ (with $\mathcal{M}$ implicit), by mutual recursion on formulas and programs (cf. fig. 1). Note that we use the same symbols (e.g. $\to$) both for the logical operators of PDL and for those of the type theory. This does not lead to ambiguity.

Our goal is to show soundness and completeness of the Hilbert system for PDL presented in fig. 2. The Hilbert system employed here is a variant of Segerberg's axiomatization as presented in [15]. We replace the original induction axiom (i.e., $[\alpha^*](\varphi \to [\alpha]\varphi) \to \varphi \to [\alpha^*]\varphi$) with a rule (Ind) and omit $\vdash \varphi \to [\alpha][\alpha^*]\varphi \to [\alpha^*]\varphi$, which is derivable by induction. We prefer the system from fig. 2 because, a priori, it appears weaker. Nevertheless, it is straightforward to show that the two systems are equivalent (see [9] for the details). Hence, all our results apply to both systems.

The satisfaction relation essentially constitutes a shallow[2] embedding of the classical object logic PDL into the constructive type theory of Coq. In order to ensure that the object logic is interpreted classically we restrict our attention to those transition systems for which the satisfaction relation is stable under double negation.

Definition 2.1. A *(classical) model* is a transition system $\mathcal{M}$ for which $\models$ is stable under double negation, i.e., a transition system satisfying

$$\forall w : \mathcal{M}\, \forall \varphi. \neg\neg(w \models \varphi) \to w \models \varphi \qquad (*)$$

[2] Here, shallow refers to the fact that $w \models \varphi$ is just a proposition talking about some relations over an arbitrary type.

$$\vdash \varphi \to \psi \to \varphi \tag{1}$$
$$\vdash (\theta \to \varphi \to \psi) \to (\theta \to \varphi) \to \theta \to \psi \tag{2}$$
$$\vdash \neg\neg\varphi \to \varphi \tag{3}$$
$$\vdash [\alpha](\varphi \to \psi) \to [\alpha]\varphi \to [\alpha]t \tag{4}$$
$$\vdash [\alpha]\varphi \to [\beta]\varphi \to [\alpha + \beta]\varphi \tag{5}$$
$$\vdash [\alpha + \beta]\varphi \to [\alpha]\varphi \tag{6}$$
$$\vdash [\alpha + \beta]\varphi \to [\beta]\varphi \tag{7}$$
$$\vdash [\alpha\beta]\varphi \to [\alpha][\beta]\varphi \tag{8}$$
$$\vdash [\alpha][\beta]\varphi \to [\alpha\beta]\varphi \tag{9}$$
$$\vdash [\alpha^*]\varphi \to \varphi \tag{10}$$
$$\vdash [\alpha^*]\varphi \to [\alpha][\alpha^*]\varphi \tag{11}$$
$$\vdash [\varphi?]\psi \leftrightarrow (\varphi \to \psi) \tag{12}$$

$$\text{MP}\ \frac{\vdash \varphi \to \psi \qquad \vdash \varphi}{\vdash \psi} \qquad \text{Nec}\ \frac{\vdash \varphi}{\vdash [\alpha]\varphi} \qquad \text{Ind}\ \frac{\vdash \varphi \to [\alpha]\varphi}{\vdash \varphi \to [\alpha^*]\varphi}$$

Figure 2. Hilbert system for PDL

Classical models can be understood in several ways. The condition (∗) localizes the classical assumptions required for soundness into the definition of models. In fact, classical models are the largest class of models for which one can constructively show soundness of the Hilbert system for PDL given in fig. 2.

Theorem 2.2 (Soundness). *Let $\mathcal{M}$ be a transition system. Then $\vdash$ is sound for $\mathcal{M}$ (i.e., $\vdash \varphi$ implies $w \models \varphi$ for all φ and all $w : \mathcal{M}$) if and only if $\mathcal{M}$ is a classical model.*

Note that every transition system is a classical model if one assumes excluded middle. Moreover, one can show constructively that all finite transition systems (with boolean transition relations and labeling) are classical models. This amounts to proving decidability of the model-checking problem – which in the case of PDL is straightforward. Since PDL has the small-model property, we only need to construct finite models to show completeness.

For the rest of the paper, the word *model* always means classical model. We say that a formula φ is *satisfiable* if $w \models \varphi$ for some state w of some model and *valid* if $w \models \varphi$ for every state w of every model.

The main result of this paper is a constructive proof that for every formula φ, one can either construct a finite model certifying the satisfiability of φ or a proof of $\neg\varphi$ from the axioms in fig. 2 (certifying the unsatisfiability of φ). Completeness of the Hilbert system (i.e., $\vdash \varphi$ whenever φ is valid) as well as decidability of satisfiability, validity and provability then follow as corollaries.

The central notions in the completeness proof are the notions of demo, subformula universe, and pruning. Demos are a class of finite syntactic models designed such that for every subformula universe U, there exists a largest demo over U satisfying all satisfiable formulas in U. We construct this largest demo using pruning in such a way that we obtain proofs of $\neg\varphi$ for all unsatisfiable formulas φ in U.

3 Subformula Universes

We now define the notion of subformula universe. A subformula universe for a formula φ is a finite set of signed formulas [23] containing all formulas that play a role in deciding satisfiability of φ.

Definition 3.1. We call a finite set F of formulas *subformula closed* if it satisfies the following closure properties:

S1. If $\varphi \to \psi \in F$, then $\{\varphi, \psi\} \subseteq F$.
S2. If $[a]\varphi \in F$, then $\varphi \in F$.
S3. If $[\alpha + \beta]\varphi \in F$, then $\{[\alpha]\varphi, [\beta]\varphi, \varphi\} \subseteq F$.
S4. If $[\alpha\beta]\varphi \in F$, then $\{[\alpha][\beta]\varphi, [\beta]\varphi, \varphi\} \subseteq F$.
S5. If $[\alpha^*]\varphi \in F$, then $\{[\alpha][\alpha^*]\varphi, \varphi\} \subseteq F$.
S6. If $[\psi?]\varphi \in F$, then $\{\psi, \varphi\} \subseteq F$.

Note that subformula closedness requires the presence of more than just the subformulas of every formula (e.g., $[\alpha][\alpha^*]\varphi$ when $[\alpha^*]\varphi \in F$).

It is a standard result about PDL that every formula is included in a subformula closed set of size linear in $|\varphi|$ called the *Fisher-Ladner closure* [12, 15] of φ. This closure be computed using two mutually recursive functions FL and $\mathsf{FL}_\Box$ as follows:

$$\begin{aligned}
\mathsf{FL}(p) &:= \{p\} \\
\mathsf{FL}(\bot) &:= \{\bot\} \\
\mathsf{FL}(\varphi \to \psi) &:= \{\varphi \to \psi\} \cup \mathsf{FL}(\varphi) \cup \mathsf{FL}(\psi) \\
\mathsf{FL}([\alpha]\varphi) &:= \mathsf{FL}_\Box(\alpha, \varphi) \cup \mathsf{FL}(\varphi)
\end{aligned}$$

$$\begin{aligned}
\mathsf{FL}_\Box(p, \varphi) &:= \{[p]\varphi\} \\
\mathsf{FL}_\Box(\alpha + \beta, \varphi) &:= \{[\alpha + \beta]\varphi\} \cup \mathsf{FL}_\Box(\alpha, \varphi) \cup \mathsf{FL}_\Box(\beta, \varphi) \\
\mathsf{FL}_\Box(\alpha\beta, \varphi) &:= \{[\alpha\beta]\varphi\} \cup \mathsf{FL}_\Box(\alpha, [\beta]\varphi) \cup \mathsf{FL}_\Box(\beta, \varphi) \\
\mathsf{FL}_\Box(\alpha^*, \varphi) &:= \{[\alpha^*]\varphi\} \cup \mathsf{FL}_\Box(\alpha, [\alpha^*]\varphi) \\
\mathsf{FL}_\Box(\psi?, \varphi) &:= \{[\psi?]\varphi\} \cup \mathsf{FL}(\psi)
\end{aligned}$$

In order to show that $\mathsf{FL}(\varphi)$ is subformula closed, we first establish the following transitivity property. The proof follows the presentation in [15].

Lemma 3.2. *1. $\varphi \in \mathsf{FL}(\varphi)$ and $[\alpha]\varphi \in \mathsf{FL}_\Box(\alpha, \varphi)$.*
2. If $\psi \in \mathsf{FL}(\varphi)$, then $\mathsf{FL}(\psi) \subseteq \mathsf{FL}(\varphi)$.
3. If $\psi \in \mathsf{FL}_\Box(\alpha, \varphi)$, then $\mathsf{FL}(\psi) \subseteq \mathsf{FL}_\Box(\alpha, \varphi) \cup \mathsf{FL}(\varphi)$.

Proof. Claim (1) is trivial. Claims (2) and (3) follow by mutual induction on φ for (2) and α for (3). □

Lemma 3.3. *$\mathsf{FL}(\varphi)$ is subformula closed.*

Proof. Immediate with lemma 3.2(1) and lemma 3.2(2). □

Lemma 3.4. $|\mathsf{FL}(\varphi)| \le |\varphi|$ *and* $|\mathsf{FL}_\Box(\alpha,\psi)| \le |\alpha|$.

Proof. By mutual induction on φ and α. □

For the definition of subformula universes, we employ signed formulas. A *signed formula* has the form φ^σ where φ is a formula and $\sigma \in \{-,+\}$ is its *sign*. Signs never occur within a formula and bind weaker than formula constructors, e.g, $[\alpha]\varphi^+$ is to be read as $([\alpha]\varphi)^+$. Semantically, negative signs correspond to top-level negations. That is, $w \models \varphi^-$ is to be read as $w \not\models \varphi$ and $w \models \varphi^+$ as $w \models \varphi$. In particular, we have:

$$w \models [\alpha]\varphi^- \;\Leftrightarrow\; \exists v. w \overset{\alpha}{\Rightarrow} v \wedge w \models \varphi^- \;\Leftrightarrow\; w \models \langle\alpha\rangle\neg\varphi$$

Definition 3.5 (Subformula Universe)**.** Let F be a subformula closed set. We refer to the set $\{\, \varphi^\sigma \mid \varphi \in F, \sigma \in \{+,-\} \,\}$ as the *subformula universe over* F. For formulas φ, we write $U(\varphi)$ for the subformula universe over $\mathsf{FL}(\varphi)$.

Signed formulas are a technical device allowing us to describe demos and pruning, which are usually described using negation-normal formulas, in terms of our minimal syntax for PDL.

4 Demos

A *clause* is a finite set of signed formulas. Demos [16, 17] are certain sets of clauses that can be seen as models in such a way that every state satisfies all signed formulas it contains.

The first requirement on demos is that all states are Hintikka sets.

Definition 4.1. A clause H is called a *Hintikka set* if satisfies the following closure conditions.

H1. $\bot^+ \notin H$.
H2. There is no formula φ such that $\{\varphi^+, \varphi^-\} \subseteq H$.
H3. If $(\varphi \to \psi)^+ \in H$, then $\varphi^- \in H$ or $\psi^+ \in H$.
H4. If $(\varphi \to \psi)^- \in H$, then $\varphi^+ \in H$ and $\psi^- \in H$.
H5. If $[\alpha\beta]\varphi^\sigma \in H$, then $[\alpha][\beta]\varphi^\sigma \in H$.
H6. If $[\alpha+\beta]\varphi^+ \in H$, then $[\alpha]\varphi^+ \in H$ and $[\beta]\varphi^+ \in H$.
H7. If $[\alpha+\beta]\varphi^- \in H$, then $[\alpha]\varphi^- \in H$ or $[\beta]\varphi^- \in H$
H8. If $[\alpha^*]\varphi^+ \in H$, then $\varphi^+ \in H$ and $[\alpha][\alpha^*]\varphi^+ \in H$.
H9. If $[\alpha^*]\varphi^- \in H$, then $\varphi^- \in H$ or $[\alpha][\alpha^*]\varphi^- \in H$.
H10. If $[\varphi?]\psi^+ \in H$, then $\varphi^- \in H$ or $\psi^+ \in H$.
H11. If $[\varphi?]\psi^- \in H$, then $\varphi^+ \in H$ and $\psi^- \in H$.

Intuitively, Hintikka sets are clauses that are consistent with respect to propositional reasoning and simple equivalences (e.g., $\vdash [\alpha\beta]\varphi \leftrightarrow [\alpha][\beta]\varphi$). Note that if $\varphi^\sigma \in U$ for some subformula universe U, then all formulas mentioned in the Hintikka condition for φ^σ are also in U. The use of signs avoids the need to close subformula universes under adding/removing top-level negations (as is, for instance, done in [11]) thus simplifying the reasoning about the closure properties of the subformula universes.

Every finite set of clauses S, can be seen as a model $\mathcal{M}(S)$ in the following way:

$$|\mathcal{M}(S)| := S$$
$$H \overset{a}{\Rightarrow}_{\mathcal{M}(S)} H' := \{\, \varphi \mid [a]\varphi \in H \,\} \subseteq H'$$
$$L_{\mathcal{M}(S)}\, p\, H := p^+ \in H$$

Even for sets of Hintikka sets, the states of $\mathcal{M}(S)$ will generally not satisfy all the formulas they contain. To see this, recall that $[a]p^-$ corresponds to the diamond formula $\langle a\rangle\neg p$, and consider the case where $S := \{\{[a]p^-, p^+\}\}$. Then the only state of $\mathcal{M}(S)$ lacks the a-successor satisfying p^- required by $[a]p^-$.

Definition 4.2. Let S be a finite set of clauses. We interpret programs as relations on S in the following way:

$$C \overset{a}{\rightsquigarrow}_S D := \{\, \varphi^+ \mid [a]\varphi^+ \in C \,\} \subseteq D$$
$$C \overset{\alpha+\beta}{\rightsquigarrow}_S D := C \overset{\alpha}{\rightsquigarrow}_S D \vee C \overset{\beta}{\rightsquigarrow}_S D$$
$$C \overset{\alpha\beta}{\rightsquigarrow}_S D := \exists E \in S. C \overset{\alpha}{\rightsquigarrow}_S E \wedge E \overset{\beta}{\rightsquigarrow}_S D$$
$$C \overset{\alpha^*}{\rightsquigarrow}_S D := C\, (\overset{\alpha}{\rightsquigarrow}_S)^*\, D$$
$$C \overset{\varphi?}{\rightsquigarrow}_S D := C = D \wedge \varphi \in C$$

Remark 1. *The set library employed in the formalization only allows us to write down sets that are finite by construction. However, finiteness of the unrestricted comprehension* $\{\, \varphi^+ \mid [a]\varphi^+ \in C \,\}$ *depends on the injectivity of the box constructor. In Coq, we instead use replacement and separation (i.e.,* $\{\, \mathrm{body}\ \varphi \mid \varphi \in \{\, \psi \in C \mid \mathrm{isBox}\ \psi \,\} \,\}$*, where isBox tests for the shape* $[\alpha]\varphi^+$ *and body strips away the outer box) yielding a set that is finite by construction.*

Note that $\overset{\alpha}{\rightsquigarrow}_S$, does not mention the satisfaction relation. This allows us to use $\overset{\alpha}{\rightsquigarrow}_S$ to phrase the condition ensuring that $\overset{\alpha}{\Rightarrow}$ and $\models$ behave as required.

Definition 4.3 (Demo)**.** A finite set $\mathcal{D}$ of Hintikka sets is called a *demo* if it satisfies the following condition:

(D) If $[\alpha]\varphi^- \in C \in \mathcal{D}$, then $C \overset{\alpha}{\rightsquigarrow}_{\mathcal{D}} D$ and $\varphi^- \in D$ for some $D \in \mathcal{D}$.

We now show that for demos $\mathcal{D}$, every state of $\mathcal{M}(\mathcal{D})$ satisfies all formulas it contains. We fix some demo $\mathcal{D}$ for the rest of this section. The proof follows [16].

Lemma 4.4. *Let* $[\alpha]\varphi^+ \in C$ *such that* $C \overset{\alpha}{\Rightarrow}_{\mathcal{M}(\mathcal{D})} D$ *and for all* $E \in S$ *and all* ψ *with* $|\psi| < |\alpha|$ *we have that* $t^- \in E$ *implies* $E \not\models \psi$*. Then* $\varphi^+ \in D$*.*

Proof. By induction on α. The assumption on formulas of size smaller than α is required to handle the case for tests. □

Lemma 4.5. *Let* $\{C, D\} \subseteq \mathcal{D}$ *such that* $C \overset{\alpha}{\rightsquigarrow}_{\mathcal{D}} D$ *and assume that* $\psi^+ \in E$ *implies* $E \models \psi$ *for all* $E \in \mathcal{D}$ *and all* ψ *such that* $|\psi| < |\alpha|$*. Then* $C \overset{\alpha}{\Rightarrow}_{\mathcal{M}(\mathcal{D})} D$*.*

Theorem 4.6 (Demo Theorem). *Let $\varphi^\sigma \in C \in \mathcal{M}(\mathcal{D})$. Then $C \models \varphi^\sigma$.*

Proof. By complete induction on $|\varphi|$. The case for $\varphi = [\alpha]\psi^+$ follows with lemma 4.4 using the induction hypothesis to establish the condition on formulas smaller than $|\alpha|$. Similarly, the case for $\varphi = [\alpha]\psi^-$ follows with the demo condition and lemma 4.5. All other cases follow by induction using the Hintikka properties of C. □

5 Pruning

Pruning [17, 21] starts from a given set of Hintikka sets and removes clauses violating the demo condition until a demo is reached. We will show that when starting from the set of all maximal Hintikka sets over some subformula universe U, this process terminates with a demo satisfying all satisfiable formulas from U. Moreover, we will obtain pruning refutations for all removed clauses.

We fix some subformula closed set F and write U for the subformula universe over F.

Definition 5.1. A Hintikka set $C \subseteq U$ is called *maximal* if for all $\varphi \in F$ either $\varphi^+ \in C$ or $\varphi^- \in C$.

The pruning function is defined recursively as follows:

$$\mathsf{prune}\, S := \begin{cases} \mathsf{prune}(S \setminus \{C\}) & \exists [\alpha]\varphi^- \in C. \neg\exists D \in S. \\ & \quad C \overset{\alpha}{\rightsquigarrow}_S D \wedge \varphi^- \in D \\ S & S \text{ is a demo} \end{cases}$$

Remark 2. *The definition above does not specify which Hintikka set is to be removed if several violate the demo condition. In the Coq development, we use a choice operator for finite sets to deterministically pick a clause to remove.*

We now define a demo over U as follows:

$$S_0 := \{\, C \subseteq U \mid C \text{ maximal and hintikka} \,\}$$
$$\mathcal{D} := \mathsf{prune}\, S_0$$

Lemma 5.2. *$\mathcal{D}$ is a demo contained in S_0.*

We say that a set of clauses S *supports* a clause C, written $S \rhd C$, if there exists some Hintikka set $D \in S$ such that $C \subseteq D$. We have already established that a formula $\varphi \in F$ is satisfiable whenever $\mathcal{D} \rhd \{\varphi^+\}$ (theorem 4.6). In order to obtain completeness, it remains to show $\vdash \neg\varphi$ whenever $\mathcal{D} \not\rhd \{\varphi^+\}$. To prove this, we need to generalize from single formulas to clauses, i.e., prove $\vdash \neg C$ whenever $\mathcal{D} \not\rhd C$. When a clause C appears in the place of a formula, as in $\vdash \neg C$ above, it is to be read as the sign respecting conjunction of the formulas it contains, i.e., C is to be read as $\bigwedge_{\varphi^\sigma \in C} \lfloor \varphi^\sigma \rfloor$ where $\lfloor \varphi^- \rfloor := \neg\varphi$ and $\lfloor \varphi^+ \rfloor := \varphi$. If $\vdash \neg C$, we call C *(Hilbert) refutable.*

We will show that all clauses over U that are not supported by S_0 are refutable. Moreover, we will show that this is an invariant that is preserved during pruning. That is, when a Hintikka clause is removed from S_0 and therefore some clause

$$\text{P1}\ \frac{\mathsf{pcoref}\, S \qquad C \subseteq U \qquad S \not\rhd C}{\mathsf{pref}\, C}$$

$$\text{P2}\ \frac{\begin{array}{cc} S \subseteq S_0 & \mathsf{pcoref}\, S \\ [\alpha]\varphi^- \in C & \neg\exists D \in S. C \overset{\alpha}{\rightsquigarrow}_S D \wedge \varphi^- \in D \end{array}}{\mathsf{pref}\, C}$$

$$\mathsf{pcoref}\, S := \forall C \in S_0 \setminus S. \mathsf{pref}\, C$$

Figure 3. Pruning Refutations

$C \subseteq U$ is no longer supported by the remaining clauses, we can prove $\vdash \neg C$, possibly using proofs constructed at an earlier stage.

To abstract from the algorithmic details of pruning, we give an inductive characterization of the clauses over U that are not supported by $\mathcal{D}$ and then translate derivations of this inductive definition to proofs in the Hilbert system. The rules are given in fig. 3. If $\mathsf{pref}\, C$, we say that C is *pruning refutable* and if $\mathsf{pcoref}\, S$ for some $S \subseteq S_0$, we say that S is *pruning corefutable*. In both rules, the set S corresponds to some intermediate stage of pruning and the premise $\mathsf{pcoref}\, S$ captures the intuition that we have already established $\mathsf{pref}\, C$ for all preciously removed Hintikka clauses. The rule P1 then expresses the fact a clause cannot be supported by $\mathcal{D}$ if all clauses that could possibly support it have already been removed and the rule P2 corresponds to the pruning condition.

Lemma 5.3. *$\mathcal{D}$ is pruning corefutable.*

Theorem 5.4 (Pruning Completeness). *Let $C \subseteq U$, then C is either pruning refutable or satisfied by a model with at most $2^{|U|}$ states.*

Proof. By case analysis on $\mathcal{D} \rhd C$ using theorem 4.6 and lemmas 5.2 and 5.3. □

6 Hilbert Refutations

We now establish completeness of the Hilbert system by showing that pruning refutable clauses are also Hilbert refutable. The proof is compositional in the sense that we show the rules for pruning refutations admissible for the Hilbert system.

For sets of clauses A, we abbreviate $\bigvee_{C \in A} C$ as $\bigvee A$. We continue to work with the subformula universe U from the previous section. We say that a set $S \subseteq S_0$ is *(Hilbert) corefutable* if $\vdash \neg C$ for all clauses in $S_0 \setminus S$.

Lemma 6.1. *Let $C \subseteq U$ be maximal but not a Hintikka set. Then $\vdash \neg C$.*

Proof. By case analysis on the Hintikka condition being violated using propositional reasoning and axioms (5-12). □

We remark that lemma 6.1 encapsulates most of the propositional reasoning required to prove completeness.

Lemma 6.2 (Extension). *Let $S \subseteq S_0$ be corefutable and let $C \subseteq U$ be a clause. Then $\vdash C \to \bigvee\{D \in S \mid C \subseteq D\}$.*

Proof. Since S is corefutable, it suffices to show $\vdash C \to \bigvee\{D \in S_0 \mid C \subseteq D\}$. The claim follows by induction on $|U| - |C|$. If C is maximal, then either $C \in S_0$ or $\vdash \neg C$ (lemma 6.1). Both cases are trivial. If C is not maximal then $\vdash C \to C \cup \{\varphi^+\} \vee C \cup \{\varphi^-\}$ for some $\varphi \in F$ such that $\{\varphi^+, \varphi^-\} \cap C = \emptyset$ and the claim follow by induction hypothesis. □

Lemma 6.3 (Admissibility of P1). *Let S be corefutable and let $C \subseteq U$ such that $S \not\vdash C$. Then $\vdash \neg C$.*

Proof. Immediate with lemma 6.2. □

Before we can translate the rule P2, we need a few more auxiliary lemmas.

Lemma 6.4. *Let $C, D \subseteq U$ be maximal. Then $\vdash C \to \neg D$ whenever $C \neq D$.*

Lemma 6.5. *Let $S \subseteq S_0$ be corefutable. Then*

1. *$\vdash \bigvee S$.*
2. *$\vdash \neg(\bigvee A) \to \bigvee(S \setminus A)$ for all $A \subseteq S$.*

Proof. Claim (1) follows immediately with lemma 6.2. For (2) it suffices to show $\vdash C \to \neg(\bigvee A) \to \bigvee(S \setminus A)$ for $C \in S$ (Claim (1)). If $C \in A$ we obtain a contradiction with $\neg \bigvee A$. Otherwise, the claim is trivial. □

In addition to the lemmas above, we also make use of the following facts.

Lemma 6.6.
1. *If $\vdash \varphi \to \psi$, then $\vdash [\alpha]\varphi \to [\alpha]\psi$ and $\langle\alpha\rangle\varphi \to \langle\alpha\rangle\psi$.*
2. *$\vdash \neg\langle\alpha\rangle\bot$*
3. *$\vdash \langle\alpha\rangle\varphi \to [\alpha]\psi \to \langle\alpha\rangle(\varphi \wedge \psi)$*
4. *$\vdash \neg[\alpha]\varphi \leftrightarrow \langle\alpha\rangle\neg\varphi$*
5. *$\vdash [\alpha]\varphi \to [\alpha]\psi \to [\alpha](\varphi \wedge \psi)$*
6. *$\vdash \langle\alpha\rangle(\varphi \vee \psi) \to \langle\alpha\rangle\varphi \vee \langle\alpha\rangle\psi$*
7. *If $\vdash \psi \to \varphi$ and $\vdash \langle\alpha\rangle\psi \to \psi$, then $\vdash \langle\alpha^*\rangle\psi \to \varphi$*

Note that lemma 6.6(1) justifies rewriting with implications underneath of modalities, and we will do so without explicit mention.

In the following we present Hilbert proofs in "backward style" where each line is obtained from the previous line by applying (or rewriting with) some lemma or by propositional reasoning (usually the introduction or elimination of some big conjunction or disjunction). We chose this presentation, rather than the traditional forward chaining, because it closely matches the way the proofs are obtained in Coq.

The next lemma is the core of the completeness proof. In particular, this is the place where the induction rule for α^* is used.

Lemma 6.7. *Let $S \subseteq S_0$ be corefutable and let $C, D \in S$. Then $\vdash C \to [\alpha]\neg D$ whenever $C \not\xrightarrow{\alpha}_S D$.*

Proof. By induction on α.

Case $\alpha = a$**:** By assumption, there exists some $[a]\varphi^+ \in C$ such that $\varphi^+ \notin D$. Hence $\varphi^- \in D$ since D is maximal. To show $\vdash C \to [a]\neg D$ it therefore suffices to show $\vdash [a]\varphi \to [a]\neg\neg\varphi$ which follows with lemma 6.6(1).

Case: $\alpha = \psi?$**:** By axiom (12), it suffices to show $\vdash C \to \psi \to \neg D$. Since $C \not\xrightarrow{\psi?}_S D$, we either have $C \neq D$ and the claim follows with lemma 6.4 or $C = D$ and $\psi^+ \notin C$. But then, $\psi^- \in C$ since C is maximal. Therefore $\vdash C \to \neg\psi$ and the claim follows with propositional reasoning.

Case $\alpha = \beta + \gamma$**:** By induction hypothesis we have both $\vdash C \to [\beta]\neg D$ and $\vdash C \to [\gamma]\neg D$. The claim then follows with axiom (5).

Case $\alpha = \beta\gamma$**:** We reason as follows:

$$
\begin{aligned}
&\vdash C \to [\beta\gamma]\neg D && \\
\Leftarrow\ &\vdash C \to [\beta][\gamma]\neg D && \text{axiom (9)} \\
\Leftarrow\ &\vdash C \to \langle\beta\rangle\langle\gamma\rangle D \to \bot && \text{lemma 6.6(4)} \\
\Leftarrow\ &\vdash C \to \langle\beta\rangle((\bigvee S) \wedge \langle\gamma\rangle D) \to \bot && \text{lemma 6.5(1)} \\
\Leftarrow\ &\vdash C \to \langle\beta\rangle(\bigvee_{E \in S}(E \wedge \langle\gamma\rangle D)) \to \bot && \\
\Leftarrow\ &\vdash C \to \bigvee_{E \in S}(\langle\beta\rangle(E \wedge \langle\gamma\rangle D)) \to \bot && \text{lemma 6.6(6)} \\
\Leftarrow\ &\vdash C \to \langle\beta\rangle(E \wedge \langle\gamma\rangle D) \to \bot && (\text{for } E \in S)
\end{aligned}
$$

By assumption, we have either $C \not\xrightarrow{\beta}_S E$ or $E \not\xrightarrow{\gamma}_S D$. By induction hypothesis, we obtain either $\vdash C \to [\beta]\neg E$ or $\vdash E \to [\gamma]\neg D$. The claim then follows with lemma 6.6(3) and lemma 6.6(2).

Case: $\alpha = \beta^*$**:** We want to apply the induction rule with a suitable invariant. We define $I := \{E \in S \mid C \overset{\beta^*}{\rightsquigarrow}_S E\}$. We clearly have $C \in I$ and therefore $\vdash C \to \bigvee I$. By assumption, $D \notin I$ and therefore $\vdash \bigvee I \to \neg D$ since for every $E \in I$ we have $E \neq D$ and therefore $\vdash E \to \neg D$ (lemma 6.4). Using the induction rule with ψ set to $\bigvee I$, it suffices to show $(\bigvee I) \to [\beta](\bigvee I)$. We reason as follows:

$$
\begin{aligned}
&\vdash (\bigvee I) \to [\beta](\bigvee I) && \\
\Leftarrow\ &\vdash E \to [\beta](\bigvee I) && E \in I \\
\Leftarrow\ &\vdash E \to \langle\beta\rangle(\neg \bigvee I) \to \bot && \text{lemma 6.6(4)} \\
\Leftarrow\ &\vdash E \to \langle\beta\rangle(\bigvee S \setminus I) \to \bot && \text{lemma 6.5(2)} \\
\Leftarrow\ &\vdash E \to (\bigvee_{F \in S \setminus I} \langle\beta\rangle F) \to \bot && \text{lemma 6.6(6)} \\
\Leftarrow\ &\vdash E \to \langle\beta\rangle F \to \bot && F \in S \setminus I \\
\Leftarrow\ &\vdash E \to [\beta]\neg F && \text{lemma 6.6(4)}
\end{aligned}
$$

Since $E \not\xrightarrow{\beta}_S F$ by the definition of I, the last claim follows by induction hypothesis. □

Remark 3. *The previous lemma can be seen as a generalization of the contrapositive of [18, Lemma 1] from the collection of maximally consistent clauses (over a given universe) to arbitrary corefutable collections of maximal Hintikka sets (i.e., all possible intermediate states of pruning). We need the generalization in order to incrementally construct Hilbert derivations and the contrapositive since at the current point in the development there is no easy way to show that provability of formulas is decidable. Decidability of provability does follow once we have established decidability of satisfiability and completeness (cf. corollary 6.12).*

Lemma 6.8 (Admissibility of P2). *Let $S \subseteq S_0$ be corefutable and let $C \in S$ with $[\alpha]\varphi^- \in C$ such that $\neg\exists D \in S. C \overset{\alpha}{\rightsquigarrow}_S D \wedge \varphi^- \in D$. Then $\vdash \neg C$.*

Proof. Let $X := \{ D \in S \mid \varphi^- \in D \}$. In order to prove $\neg C$, we show that C implies both $[\alpha](\bigwedge_{D\in X} \neg D)$ and $\langle\alpha\rangle \bigvee X$ and that that these two consequences are contradictory. For the first implication we reason as follows:

$$\begin{aligned} &\vdash C \to [\alpha](\bigwedge_{D\in X} \neg D) \\ \Leftarrow\ &\vdash C \to \bigwedge_{D\in X} [\alpha]\neg D && \text{lemma 6.6(5)} \\ \Leftarrow\ &\vdash C \to [\alpha]\neg D && D \in X \end{aligned}$$

The last claim follows with lemma 6.7 since $C \overset{\alpha}{\not\rightsquigarrow}_S D$ by assumption. For the second implication we have:

$$\begin{aligned} &\vdash C \to \langle\alpha\rangle \bigvee X \\ \Leftarrow\ &\vdash \neg[\alpha]\varphi \to \langle\alpha\rangle \bigvee X && \neg[\alpha]\varphi \in C \\ \Leftarrow\ &\vdash \langle\alpha\rangle\neg\varphi \to \langle\alpha\rangle \bigvee X && \text{lemma 6.6(4)} \\ \Leftarrow\ &\vdash \{\varphi^-\} \to \bigvee X && \text{lemma 6.6(1)} \end{aligned}$$

This time, the last claim follows with lemma 6.2.

Now it suffices to show:

$$\begin{aligned} &\vdash [\alpha](\bigwedge_{D\in X} \neg D) \to \langle\alpha\rangle(\bigvee X) \to \bot \\ \Leftarrow\ &\vdash \langle\alpha\rangle((\bigwedge_{D\in X} \neg D) \wedge \bigvee X) \to \bot && \text{lemma 6.6(3)} \\ \Leftarrow\ &\vdash \neg((\bigwedge_{D\in X} \neg D) \wedge \bigvee X) && \text{lemma 6.6(2)} \end{aligned}$$

The last claim follows with propositional reasoning. Thus we obtain $\vdash \neg C$. □

Lemma 6.9. *$\vdash \neg C$ whenever C is pruning refutable.*

Proof. By induction on pref C using lemmas 6.3 and 6.8. □

We are now in the position to prove our main result for PDL.

Theorem 6.10 (Informative Completeness). *For every PDL formula φ, one can either construct a proof of $\neg\varphi$ or a model with at most $2^{2|\varphi|}$ states satisfying φ.*

Proof. Fix some formula φ. By theorem 5.4, we either obtain a model for φ of size $2^{|U(\varphi)|}$ and the claim follows with lemma 3.4 or we have pref C and the claim follows with lemma 6.9. □

In Coq, theorem 6.10 takes the form of a function having the (dependent) type

$$\forall\varphi.\ (\Sigma\mathcal{M}\,(x : \mathcal{M}).\ |\mathcal{M}| < 2^{2|\varphi|} \wedge w \models \varphi) + (\vdash \neg\varphi)$$

Corollary 6.11 (Completeness). *$\vdash \varphi$ whenever φ is valid.*

We say a predicate $P : X \to \mathrm{Prop}$ is *decidable* if there is a function $p : X \to \mathbb{B}$ such that $Px \leftrightarrow (p\,x = \mathrm{true})$ for all $x : X$ (i.e., p decides P). With respect to this (shallow) notion of decidability, we also obtain:

Corollary 6.12 (Decidability). *Satisfiability, validity and provability of formulas are decidable.*

Proof. Immediate with soundness and completeness. □

Corollary 6.13 (Small-Model Property). *Let φ be satisfiable. Then φ is satisfied by a model with at most $2^{2|\varphi|}$ states.*

Proof. Immediate with soundness. □

Note that we prove the small-model property (smp) using a decision procedure rather than proving decidability using the smp as is often done in classical arguments.

7 Converse

We now extend the informative completeness result from PDL to CPDL. The proofs remain largely the same as for PDL. Therefore we only describe the parts that need to be changed. The formalization accompanying this paper [9] includes separate developments for the two logics.

We extend the syntax with a new program construct $\alpha^\smile$. The satisfaction relation for CPDL is defined as for PDL with the interpretation of $\alpha^\smile$ defined as:

$$w \overset{\alpha^\smile}{\Rightarrow} v := v \overset{\alpha}{\Rightarrow} w$$

Following [15], the Hilbert system for CPDL extends the Hilbert system from fig. 2 with two axioms:

$$\vdash \varphi \to [\alpha]\langle\alpha^\smile\rangle\varphi \tag{13}$$

$$\vdash \varphi \to [\alpha^\smile]\langle\alpha\rangle\varphi \tag{14}$$

By duality we obtain:

Lemma 7.1. *$\vdash \langle\alpha\rangle[\alpha^\smile]\varphi \to \varphi$ and $\vdash \langle\alpha^\smile\rangle[\alpha]\varphi \to \varphi$.*

The main problem in extending the proof to CPDL is to adapt the proof of lemma 4.4. It turns out that the Hintikka conditions (definition 4.1) are insufficient to handle the case for converse appearing on top of other programs. We resolve this by showing that the Hilbert system validates a conversion to *converse normal form* (i.e., formulas where converse is only applied to atomic programs) and then restricting to converse normal formulas.

$$\mathsf{cnf}\, p := p \qquad \mathsf{cnf}\,(\varphi \to \psi) := \mathsf{cnf}\,\varphi \to \mathsf{cnf}\,\psi$$
$$\mathsf{cnf}\,\bot := \bot \qquad \mathsf{cnf}\,([\alpha]\varphi) := [\mathsf{cnp}\,\mathsf{false}\,\alpha](\mathsf{cnf}\,\varphi)$$

$$\begin{aligned}
\mathsf{cnp}\,\mathsf{false}\,a &:= a\\
\mathsf{cnp}\,\mathsf{true}\,a &:= a^\smile\\
\mathsf{cnp}\,\mathsf{false}\,(\alpha\beta) &:= (\mathsf{cnp}\,\mathsf{false}\,\alpha)(\mathsf{cnp}\,\mathsf{false}\,\beta)\\
\mathsf{cnp}\,\mathsf{true}\,(\alpha\beta) &:= (\mathsf{cnp}\,\mathsf{true}\,\beta)(\mathsf{cnp}\,\mathsf{true}\,\alpha)\\
\mathsf{cnp}\,b\,(\alpha+\beta) &:= \mathsf{cnp}\,b\,\alpha + \mathsf{cnp}\,b\,\beta\\
\mathsf{cnp}\,b\,(\alpha^*) &:= (\mathsf{cnp}\,b\,\alpha)^*\\
\mathsf{cnp}\,b\,(\varphi?) &:= (\mathsf{cnf}\,\varphi)?\\
\mathsf{cnp}\,b\,(\alpha^\smile) &:= \mathsf{cnp}\,(\neg b)\,\alpha
\end{aligned}$$

Figure 4. Converse Normalization

We start by computing converse normal forms. We want to exhaustively apply the following transformations to programs.

$$(\alpha+\beta)^\smile \mapsto \alpha^\smile + \beta^\smile \qquad \alpha^{*\smile} \mapsto \alpha^{\smile *}$$
$$(\alpha\beta)^\smile \mapsto \beta^\smile\alpha^\smile \qquad \alpha^{\smile\smile} \mapsto \alpha$$

As with the computation of the Fisher-Ladner closure, we define two functions

$$\mathsf{cnf} : \mathsf{formula} \to \mathsf{formula}$$
$$\mathsf{cnp} : \mathsf{bool} \to \mathsf{program} \to \mathsf{program}$$

by mutual recursion on formulas and programs as shown in fig. 4. The boolean argument for cnp serves as a flag signaling whether we are currently pushing down a converse operation. This allows for a simple structurally recursive definition of converse normalization.

Lemma 7.2. *1.* $\vdash \varphi \leftrightarrow \mathsf{cnf}\,\varphi$
2. $\vdash [\mathsf{cnp}\,\mathsf{true}\,\alpha]\psi \leftrightarrow [\alpha^\smile]\psi$
3. $\vdash [\mathsf{cnp}\,\mathsf{false}\,\alpha]\psi \leftrightarrow [\alpha]\psi$

Proof. We show claim (1) and the conjunction of (2) and (3) by mutual induction on φ and α. Most cases follow immediately with the respective induction hypotheses. It remains to show $\vdash [\alpha]\varphi \leftrightarrow [\beta]\varphi$ whenever $\alpha \mapsto \beta$. We show one direction of the case for $\alpha^{*\smile} \mapsto \alpha^{\smile *}$.

$$\begin{aligned}
&\vdash [\alpha^{\smile *}]\varphi \to [\alpha^{*\smile}]\varphi\\
\Leftarrow\ &\vdash [\alpha^{*\smile}]\langle\alpha^*\rangle[\alpha^{\smile *}]\varphi \to [\alpha^{*\smile}]\varphi && \text{axiom (14)}\\
\Leftarrow\ &\vdash \langle\alpha^*\rangle[\alpha^{\smile *}]\varphi \to \varphi && \text{lemma 6.6(1)}\\
\Leftarrow\ &\vdash \langle\alpha^*\rangle[\alpha^{\smile *}]\varphi \to [\alpha^{\smile *}]\varphi && \text{axiom (10)}\\
\Leftarrow\ &\vdash \langle\alpha\rangle[\alpha^{\smile *}]\varphi \to [\alpha^{\smile *}]\varphi && \text{lemma 6.6(7)}\\
&\text{(and } \vdash [\alpha^{\smile *}]\varphi \to [\alpha^{\smile *}]\varphi\text{)}\\
\Leftarrow\ &\vdash \langle\alpha\rangle[\alpha^\smile][\alpha^{\smile *}]\varphi \to [\alpha^{\smile *}]\varphi && \text{axiom (11)}
\end{aligned}$$

The last claim follows with lemma 7.1. □

Note that in the proof above, we need to generalize the claim using axiom (10) before applying the induction lemma (lemma 6.6(7)).

We extend the definition of $\mathsf{FL}_\square$ with an additional clause for converse

$$\mathsf{FL}_\square(\alpha^\smile, \varphi) := \{[\alpha^\smile]\varphi\} \cup \mathsf{FL}_\square(\alpha, \varphi)$$

Lemma 7.3. *If φ is converse normal, then all $\psi \in \mathsf{FL}(\varphi)$ are converse normal.*

The only place in the proof where we need to exploit the fact that we can restrict to converse normal formulas is when adapting the proof of lemma 4.4. We adapt the transition relation on sets of clauses by changing the case for atomic programs to respect converses of atomic programs

$$C \overset{a}{\rightsquigarrow}_S D := \{\,\varphi^+ \mid [a]\varphi^+ \in C\,\} \subseteq D \quad \wedge \quad \{\,\varphi^+ \mid [a^\smile]\varphi^+ \in D\,\} \subseteq C$$

and adapt the definition of $\mathcal{M}(S)$ accordingly.

Lemma 7.4. *Let $\mathcal{D}$ be a demo containing only converse normal formulas and let $[\alpha]\varphi^+ \in C$ such that $C \overset{\alpha}{\Rightarrow}_{\mathcal{M}(\mathcal{D})} D$ and for all $E \in S$ and all ψ with $|\psi| < |\alpha|$ we have that $t^- \in E$ implies $E \not\models \psi$. Then $\varphi^+ \in D$.*

Proof. By induction on α. The case for $a^\smile$ is symmetric to the case for a. All other cases are essentially the same as in the proof of lemma 4.4. □

The translation to Hilbert refutations requires the addition of two new cases to the proof of lemma 6.7. Firstly, $C \overset{a}{\rightsquigarrow}_S D$ can now also fail because $\varphi^+ \notin C$ for some formula $[a^\smile]\varphi \in D$. Secondly, we need to handle the case where $C \overset{\alpha^\smile}{\not\rightsquigarrow}_S D$. Both cases are straightforward.

Theorem 7.5. *For every CPDL formula φ, one can either construct a proof of $\neg\varphi$ or a model with at most $2^{4|\varphi|}$ states satisfying φ.*

Proof. Let φ be some formula. Then $\vdash \varphi \leftrightarrow \mathsf{cnf}\varphi$ (lemma 7.2) and $|U(\varphi)| \le 4|\varphi|$ since $|\mathsf{cnf}\,\varphi| \le 2|\varphi|$. The claim then follows analogously to the proof of theorem 6.10. □

The corollaries from the previous section carry over to CPDL as expected.

We remark that, since CPDL has both more syntactic constructs and a Hilbert system with more axioms, the completeness result for CPDL does not subsume the completeness result for PDL.

8 Remarks on the Formalization

The Coq development accompanying this paper [9] follows the mathematical development fairly closely and provides the details elided in the paper. The development consists of about 1300 lines for CPDL and 1000 lines for PDL split roughly half-and-half between specifications and proofs. This conciseness

is achieved by relying on the mathematical component libraries [25] as well as two libraries developed in [8]. The latter two account for another 2000 lines.

The first library is for reasoning about finite sets over countable base types (e.g., formulas) and used to define the Fisher-Ladner closure, clauses, demos, and pruning. While the mathematical component libraries do contain a library for extensional finite sets, this library only provides sets over finite types, which is too restrictive for our purposes. While we use finite sets as data structure underlying the pruning method for deciding PDL satisfiability, the purpose is to allow for a constructive proof rather than actually running the procedure. Consequently, we implement finite sets without regard for computational costs. We obtain an extensional representation by representing each set using some canonical duplicate free list. In addition to the usual operations (e.g., separation, replacement, and powerset), the library also features a number of constructions not needed here (e.g., fixpoint operators for bounded monotone functions) and comes with rudimentary automation based on a tableau calculus implemented in Ltac [2]. We remark that there is another library for finite sets over countable types [7], currently being developed with the aim of integrating it into the mathematical component libraries, incorporating some of the design decisions underlying our finite set library.

The second library underlying the development is a library for constructing Hilbert derivations. It is folklore that reasoning inside deeply embedded proof systems (i.e., where the proof system is represented using an inductive definition) is cumbersome in Coq. This is particularly true for reasoning inside a bare Hilbert system due to the lack of assumption management. The libraries developed in [8] provide tactics for assumption management for any Hilbert system extending classical propositional logic. Further, it provides the instances to enable (setoid) rewriting [24] with the preorder $\varphi \prec \psi := \vdash \varphi \to \psi$. The facilities for assumption management are mainly used when proving basic, often propositional, facts. For the more high-level lemmas (e.g., lemma 6.7) rewriting with $\prec$ is the main source of automation. We remark that rewriting with equivalences alone would be too restrictive for our purposes since many important facts (e.g., axioms (13) and (14)) are only implications.

Morally, we see proofs as computational objects that can be inspected and manipulated. However, the need for setoid rewriting described above forces us to formalize the Hilbert systems as predicates (i.e., $\vdash$: formula $\to$ Prop) rather than as families of types (i.e., $\vdash$: formula $\to$ Type). While the introduction of universe polymorphism in recent versions of Coq allows, in principle, to use setoid rewriting also for type families, we ran into technical problems that we were, so far, unable to resolve when trying to turn the Hilbert systems into type families.

From the engineering point of view, it is also unfortunate that we were forced to create two separate developments for PDL and CPDL even though there is a considerable overlap between the two proofs. We could have obtained a limited amount of sharing by proving basic facts about the Hilbert systems (e.g., lemma 6.6) for a structure hiding the inductive nature of the Hilbert system (i.e., the fact that there are no other axioms) and then instantiating this structure with the Hilbert systems for both PDL and CPDL. This approach was used in [8] to build a hierarchy of Hilbert systems including propositional logic, K, K*, and CTL. However, given the large number of syntactic constructs for PDL and the fact that we only need about a dozen of these basic facts, the gains would be marginal at best. One option to merge the two developments might have been the introduction of a boolean flag, along the lines of the formalization underlying [20], signaling the presence or absence of converse in definitions and lemma statements. This could provide for significant sharing, at the cost of some technical overhead and slightly less natural definitions and lemma statements.

9 Conclusion

We have given formal and constructive completeness proofs for PDL and CPDL by combining ideas and techniques from a variety of sources [15, 16, 18]. We consider the constructive argument given here more informative than a classical completeness proof in the sense that it provides an algorithm constructing both finite models for satisfiable formulas and proofs for valid formulas.[3] In addition to basing the proof on an algorithm, we also prove the correctness of this algorithm without classical assumptions. The reason for this is twofold. First, classical assumptions (besides those localized to classical models) are simply not necessary. Moreover, working without axioms allows us to appeal to the normalization property of the logic of Coq. This, for instance, yields that the shallow notion of decidability used in corollary 6.12 entails computational decidability in the usual sense.

The completeness proof is designed to be constructive while reusing ideas from the literature wherever possible. The desire to work constructively essentially rules out the approach in [15], where completeness is established by using filtration on an infinite non-standard canonical model. In fact, even the construction of a finite "canonical" model for a given subformula universe in [18] is non-constructive in the sense that it requires decidability of Hilbert provability in order to determine which formulas are contained in which state. Of course, Hilbert provability is decidable (corollary 6.12) However, since the Hilbert system is not analytic, the easiest way to establish this is via completeness. This motivates basing the proof on a decision method. We use pruning since the maximal demo it constructs corresponds closely to the model employed in [18]. This allows us to obtain one of the key lemmas in the translation from pruning refutations to

[3] More precisely, it provides an algorithm that is, while still impractical, more informative than blindly enumerating proofs.

Hilbert refutations by adapting the corresponding lemma in [18]. Altogether, we obtain a natural factorization of the proof into an algorithmic part for the decision method, a semantic argument for the model construction, and a syntactic translation from pruning refutations to Hilbert refutations.

By using converse normalization, we were able to adapt the proofs for PDL to CPDL with only a few local changes. We remark that proving the commutation properties underlying the correctness of converse normalization (lemma 7.2) turned out to be surprisingly tricky, in particular as it comes to $\vdash [\alpha^{*\smile}]\varphi \leftrightarrow [\alpha^{\smile *}]\varphi$ where both directions require a generalization of the statement before the induction rule is applied. Our attempts to find the relevant arguments in the literature were unsuccessful. The Coq development [9] contains all arguments in their entirety.

There are a number of methodical differences between the proofs presented here and the constructive completeness proofs for CTL in [8, 10]. The most fundamental one is the use of the more traditional Hintikka sets in favor of literal clauses and support. Here, literal clauses are clauses containing only formulas of the form $[a]\varphi^\sigma$ and p^σ and the support relation is a recursively-defined decidable predicate corresponding to the Hintikka conditions, e.g.,

$$C \rhd (\varphi \to \psi)^+ := C \rhd \varphi^- \vee C \rhd \psi^+$$

That is, the support relation is defined such that a literal clause supports all its possible Hintikka extensions. In [8], where pruning refutations are also translated to derivations of the sequent system for CTL presented in [6], the support relation provides a natural fit for the destructive reading of the sequent rules (i.e., the reading where the active formula is removed when applying a rule and *next state* rules are applied to literal clauses only). We would have preferred to extend the methodology employed for CTL also to PDL. However, as observed in [1, 16], a naive recursive definition of support for PDL, employing

$$C \rhd [\alpha^*]\varphi^+ := C \rhd \varphi^+ \wedge C \rhd [\alpha][\alpha^*]\varphi$$

to handle transitive closure, does not terminate on $[a^{**}]p^+$. This is sometimes called the *nested star problem*. While a notion of support can be defined for PDL [16], it is not clear whether the Hilbert system is expressive enough for constructing derivations based on this definition. When moving from literals and support to Hintikka sets, the recursive definition of the support relation is replaced with the checking of closure properties for clauses sets, thus avoiding the nested star problem.

Brünnler and Lange [6] suggest that, following the same methodology as for CTL, it should be possible to obtain an analytic sequent system for PDL. To the best of our knowledge, the details have not been worked out yet. It would be interesting to see if such an analytic sequent system can indeed be derived for PDL and whether a pruning based argument can be used to show its completeness.

Acknowledgments

The first author has been funded by the European Research Council (ERC) under the European Union's Horizon 2020 programme (CoVeCe, grant agreement No 678157).

This work was supported by the LABEX MILYON (ANR-10-LABX-0070) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007) operated by the French National Research Agency (ANR).

References

[1] Pietro Abate, Rajeev Goré, and Florian Widmann. 2009. An On-the-Fly Tableau-Based Decision Procedure for PDL-Satisfiability. In *Proc. 5th Workshop on Methods for Modalities (M4M-5) (Electr. Notes Theor. Comput. Sci.)*, Carlos Areces and Stéphane Demri (Eds.), Vol. 231. Elsevier, 191–209.

[2] Alexander Anisimov. 2015. *Proof Automation for Finite Sets.* B.Sc. Thesis. Saarland University.

[3] Franz Baader and Carsten Lutz. 2007. Description Logic. In *Handbook of Modal Logic*, Patrick Blackburn, Johan van Benthem, and Frank Wolter (Eds.). Studies in Logic and Practical Reasoning, Vol. 3. Elsevier, 757–820.

[4] Joachim Bard. 2017. *A Formal Completeness Proof for Test-free PDL.* B.Sc. Thesis. Saarland University.

[5] Mordechai Ben-Ari, Amir Pnueli, and Zohar Manna. 1983. The Temporal Logic of Branching Time. *Acta Inf.* 20 (1983), 207–226.

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

[7] Cyril Cohen. 2017. A finset and finmap DRAFT library. https://github.com/math-comp/finmap. (Nov. 2017). Accessed Nov. 17th, 2017.

[8] Christian Doczkal. 2016. *A Machine-Checked Constructive Metatheory of Computation Tree Logic.* Ph.D. Dissertation. Saarland University.

[9] Christian Doczkal and Joachim Bard. 2017. Coq development accompanying this paper. https://github.com/chdoc/comp-dec-pdl. (2017).

[10] Christian Doczkal and Gert Smolka. 2016. Completeness and Decidability Results for CTL in Constructive Type Theory. *J. Autom. Reasoning* 56, 3 (2016), 343–365.

[11] E. Allen Emerson and Joseph Y. Halpern. 1985. Decision Procedures and Expressiveness in the Temporal Logic of Branching Time. *J. Comput. System Sci.* 30, 1 (1985), 1–24.

[12] Michael J. Fischer and Richard E. Ladner. 1979. Propositional Dynamic Logic of Regular Programs. *J. Comput. System Sci.* 18 (1979), 194–211. Issue 2.

[13] Dov M. Gabbay. 1977. Axiomatization of Logic Programs. (1977). Text of a letter to V. Pratt.

[14] Georges Gonthier, Assia Mahboubi, and Enrico Tassi. 2016. *A Small Scale Reflection Extension for the Coq system.* Research Report RR-6455. INRIA. http://hal.inria.fr/inria-00258384/en/

[15] David Harel, Dexter Kozen, and Jerzy Tiuryn. 2000. *Dynamic Logic.* The MIT Press.

[16] Mark Kaminski. 2012. *Incremental Decision Procedures for Modal Logics with Nominals and Eventualities.* Ph.D. Dissertation. Saarland University.

[17] Mark Kaminski, Thomas Schneider, and Gert Smolka. 2011. Correctness and Worst-Case Optimality of Pratt-Style Decision Procedures for Modal and Hybrid Logics. In *TABLEAUX 2011 (LNCS (LNAI))*, Kai Brünnler and George Metcalfe (Eds.), Vol. 6793. Springer, 196–210.

[18] Dexter Kozen and Rohit Parikh. 1981. An Elementary Proof of the Completness of PDL. *Theor. Comput. Sci.* 14 (1981), 113–118.

[19] Rohit Parikh. 1978. The Completeness of Propositional Dynamic Logic. In *Mathematical Foundations of Computer Science (LNCS)*, Józef Winkowski (Ed.), Vol. 64. Springer, 403–415.

[20] Damien Pous. 2013. Kleene Algebra with Tests and Coq Tools for while Programs. In *Interactive Theorem Proving (ITP 2013) (LNCS)*, Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie (Eds.), Vol. 7998. Springer, 180–196.

[21] Vaughan R. Pratt. 1979. Models of Program Logics. In *Proc. 20th Annual Symp. on Foundations of Computer Science (FOCS'79)*. IEEE Computer Society Press, 115–122.

[22] Krister Segerberg. 1977. A Completeness Theorem in the Modal Logic of Programs. *Notices Amer. Math. Soc.* 24 (1977), A–552.

[23] Raymond M. Smullyan. 1963. A Unifying Principal in Quantification Theory. *Proceedings of the National Academy of Sciences* 49 (1963), 828–832.

[24] Matthieu Sozeau. 2009. A New Look at Generalized Rewriting in Type Theory. *J. Form. Reason.* 2, 1 (2009), 41–62.

[25] The Mathematical Components team. 2017. Mathematical Components. (2017). http://math-comp.github.io/math-comp/