Completeness and Decidability of Converse PDL in the Constructive Type Theory of Coq
Christian Doczkal, Joachim Bard

To cite this version:
Christian Doczkal, Joachim Bard. Completeness and Decidability of Converse PDL in the Constructive Type Theory of Coq. Certified Programs and Proofs, Jan 2018, Los Angeles, United States. hal-01646782

HAL Id: hal-01646782
https://hal.archives-ouvertes.fr/hal-01646782
Submitted on 23 Nov 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
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
cristian.doczkal@ens-lyon.fr

Joachim Bard
Saarland University

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.

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 $[\alpha]\phi$ is satisfied by some state if all $\alpha$-reachable states satisfy $\phi$. The language of programs includes a reflexive transitive closure operation $(\alpha^\star)$ 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 $\phi$ one can either construct a proof of $\neg\phi$ 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., $\vdash \phi \lor \nvdash \phi$ for all $\phi$). 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 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\?

- $\varphi, \psi ::= p \mid \bot \mid \varphi \rightarrow \psi \mid [\alpha]\varphi \quad (p : \mathcal{P})$

We define the remaining logical operations in terms of this syntax (i.e., $\neg \varphi ::= \varphi \rightarrow \bot$, $\varphi \land \psi ::= \neg(\varphi \rightarrow \neg \psi)$, $(\alpha)\varphi ::= \neg[\alpha]\neg \varphi$, $[\alpha]\ Empresa1=\neg[\alpha]\neg \varphi$).

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 $M$ hence consists of

- A (possibly infinite) type $|M|$ whose elements are called states
- A transition relation $\Rightarrow_M : |M| \rightarrow |M|$ Prop for every $a : \mathcal{A}$
- A labeling $L_M : \mathcal{P} \rightarrow |M| \rightarrow Prop$

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

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

![Figure 1. Semantics of formulas and programs](image_url)

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., $[\alpha^*][\varphi \rightarrow [\alpha]\varphi] \rightarrow [\alpha^*] \varphi$ with a rule (Ind) and omit $+ \varphi \rightarrow [\alpha][\alpha^*]\varphi \rightarrow [\alpha^*]\varphi$, 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 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 $M$ for which $\models$ is stable under double negation, i.e., a transition system satisfying

$\forall \varphi, \neg\neg (w \models \varphi) \rightarrow w \models \varphi$  

$^{(*)}$

Classical models can be understood in several ways. The condition $(\ast)$ 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 $M$ be a transition system. Then $\vdash$ is sound for $M$ (i.e., $\vdash \varphi$ implies $w \models \varphi$ for all $\varphi$ and all $w : M$) if and only if $M$ is a classical model.

$^{2}$Here, shallow refers to the fact that $w \models \varphi$ is just a proposition talking about some relations over an arbitrary type.
We now define the notion of subformula universe. A subformula universe \( \phi \) over every subformula universe \( U \) are a class of finite syntactic models designed such that for \( \models \) then follow as corollaries.

Completeness and Decidability of Converse PDL in Coq

\[
\begin{align*}
\vdash \phi & \to \psi \to \phi & (1) \\
\vdash (\theta \to \phi \to \psi) & \to (\theta \to \phi) \to \theta \to \psi & (2) \\
\vdash \neg \phi & \to \phi & (3) \\
\vdash [\alpha][\phi \to \psi] & \to [\alpha]\phi \to [\alpha]! & (4) \\
\vdash [\alpha]\phi & \to [\beta]\phi \to [\alpha + \beta]\phi & (5) \\
\vdash [\alpha + \beta]\phi & \to [\alpha]\phi & (6) \\
\vdash [\alpha + \beta]\phi & \to [\beta]\phi & (7) \\
\vdash [\alpha\beta]\phi & \to [\alpha][\beta]\phi & (8) \\
\vdash [\alpha\beta]\phi & \to [\alpha\beta]\phi & (9) \\
\vdash [\alpha^+]? & \to \phi & (10) \\
\vdash [\alpha^+]? & \to [\alpha][\alpha^+]? & (11) \\
\vdash [\phi]? & \psi \leftrightarrow (\phi \to \psi) & (12)
\end{align*}
\]

**Figure 2.** Hilbert system for PDL

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.

The central notions in the completeness proof are the Fisher-Ladner closure (i.e., \( \models \) whenever \( \models \)) 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 \( \models \phi \) for all unsatisfiable formulas \( \phi \) in \( U \).

3 Subformula Universes

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

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

1. If \( \phi \to \psi \in F \), then \( \{ \phi, \psi \} \subseteq F \).
2. If \( [\alpha]\phi \in F \), then \( \phi \in F \).
3. If \( [\alpha + \beta]\phi \in F \), then \( \{[\alpha]\phi, [\beta]\phi, \phi\} \subseteq F \).
4. If \( [\alpha\beta]\phi \in F \), then \( \{[\alpha][\beta]\phi, [\beta]\phi, \phi\} \subseteq F \).
5. If \( [\alpha^+]? \phi \in F \), then \( \{[\alpha][\alpha^+]? \phi, \phi\} \subseteq F \).
6. If \( [\psi]? \phi \in F \), then \( \{\psi, \phi\} \subseteq F \).

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

It is a standard result about PDL that every formula is included in a subformula closed set of size linear in \( |\phi| \) called the Fisher-Ladner closure [12, 15] of \( \phi \). This closure be computed using two mutually recursive functions \( FL \) and \( FL_\omega \) as follows:

\[
\begin{align*}
FL(p) & := \{p\} \\
FL(\bot) & := \{\bot\} \\
FL(\phi \to \psi) & := \{\phi \to \psi\} \cup FL(\phi) \cup FL(\psi) \\
FL([\alpha]\phi) & := FL_\omega([\alpha, \phi]) \cup FL(\phi) \\
FL([\alpha + \beta]\phi) & := \{[\alpha + \beta]\phi\} \cup FL_\omega([\alpha, \phi], [\beta, \phi]) \cup FL(\beta, \phi) \\
FL([\alpha\beta]\phi) & := \{[\alpha\beta]\phi\} \cup FL_\omega([\alpha, [\beta]\phi]) \cup FL(\beta, \phi) \\
FL([\alpha^+]? \phi) & := \{[\alpha^+]? \phi\} \cup FL_\omega([\alpha, [\alpha^+]? \phi]) \\
FL([\psi]? \phi) & := \{[\psi]? \phi\} \cup FL(\psi)
\end{align*}
\]

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

**Lemma 3.2.**
1. \( \phi \in FL(\phi) \) and \( [\alpha]\phi \in FL_\omega([\alpha, \phi]) \).
2. If \( \psi \in FL(\phi) \), then \( FL(\psi) \subseteq FL(\phi) \).
3. If \( \psi \in FL_\omega([\alpha, \phi]) \), then \( FL(\psi) \subseteq FL_\omega([\alpha, \phi]) \cup FL(\phi) \).

**Proof.** Claim (1) is trivial. Claims (2) and (3) follow by mutual induction on \( \phi \) for (2) and \( \alpha \) for (3). □

**Lemma 3.3.** \( FL(\phi) \) is subformula closed.

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

**Lemma 3.4.** \( |FL(\phi)| \leq |\phi| \) and \( |FL_\omega([\alpha, \psi])| \leq |\alpha| \).

**Proof.** By mutual induction on \( \phi \) and \( \alpha \). □

For the definition of subformula universes, we employ signed formulas. A signed formula has the form \( \phi^\sigma \) where \( \phi \) is a formula and \( \sigma \in \{-, +\} \) is its sign. Signs never occur within a formula and bind weaker than formula constructors, e.g., \( [\alpha]\phi^+ \) is to be read as \( ([\alpha]\phi)^+ \). Semantically, negative signs correspond to top-level negations. That is, \( w \models \phi^- \) is
to be read as \( w \models \neg \varphi \) and \( w \models \varphi^+ \) as \( w \models \varphi \). In particular, we have:

\[
\begin{align*}
w \models [\alpha] \varphi^- & \iff \exists v. w \xrightarrow{\alpha} v \land w \models \varphi^- \iff w \models \langle \alpha \rangle \neg \varphi \\
\text{Definition 3.5 (Subformula Universe). Let } F \text{ be a subformula closed set. We refer to the set } \{\varphi^\sigma \mid \varphi \in F, \sigma \in \{+, -\}\} \text{ as the subformula universe over } F. \text{ For formulas } \varphi, \text{ we write } U(\varphi) \text{ for the subformula universe over } FL(\varphi). \\
\text{Signed formulas are a technical device allowing us to describe demos and pruning, which are usually described using negation-normal formulas, in terms of our minimal syntax for PDL.}
\]

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.

\( H_1. \) \( \bot \not\in H \).

\( H_2. \) There is no formula \( \varphi \) such that \( \{\varphi^+, \varphi^-\} \subseteq H \).

\( H_3. \) If \( \langle \varphi \Rightarrow \varphi^+ \rangle \in H \), then \( \varphi^- \in H \) or \( \varphi^+ \in H \).

\( H_4. \) If \( \langle \varphi \Rightarrow \varphi^- \rangle \in H \), then \( \varphi^+ \in H \) and \( \varphi^- \in H \).

\( H_5. \) If \( [\alpha | \beta] \varphi^\sigma \in H \), then \( [\alpha][\beta] \varphi^\sigma \in H \).

\( H_6. \) If \( [\alpha +] \varphi^+ \in H \), then \( [\alpha] \varphi^+ \in H \) and \( [\beta] \varphi^+ \in H \).

\( H_7. \) If \( [\alpha +] \varphi^- \in H \), then \( [\alpha] \varphi^- \in H \) or \( [\beta] \varphi^- \in H \).

\( H_8. \) If \( [\alpha] \varphi^+ \in H \), then \( \varphi^+ \in H \) and \( [\alpha][\alpha] \varphi^+ \in H \).

\( H_9. \) If \( [\alpha] \varphi^- \in H \), then \( \varphi^- \in H \) or \( [\alpha][\alpha] \varphi^- \in H \).

\( H_{10.} \) If \( [\varphi^\beta] \varphi^+ \in H \), then \( \varphi^- \in H \) or \( \varphi^+ \in H \).

\( H_{11.} \) If \( [\varphi^\beta] \varphi^- \in H \), then \( \varphi^+ \in H \) and \( \varphi^- \in H \).

Intuitively, Hintikka sets are clauses that are consistent with respect to state-local reasoning. Note that if \( \varphi^\sigma \in U \) for some subformula universe \( U \), and all formulas mentioned in the Hintikka condition for \( \varphi^\sigma \) 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 \( M(S) \) in the following way:

\[
\begin{align*}
|M(S)| & := S \\
H \xrightarrow{a} M(S) H' & := \{\varphi \mid [a] \varphi \in H\} \subseteq H' \\
L_{M(S)} \rho H & := \rho^+ \in H
\end{align*}
\]

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

\[
\begin{align*}
C \xrightarrow{a} S D \ &:= \{\varphi^+ \mid [a] \varphi^+ \in C\} \subseteq D \\
C \xrightarrow{\alpha +} S D \ &:= C \xrightarrow{a} S D \cup C \xrightarrow{\alpha} S D \\
C \xrightarrow{\alpha} S D \ &:= \exists E \in S. C \xrightarrow{\alpha} S E \land E \xrightarrow{\beta} S D \\
C \xrightarrow{\alpha -} S D \ &:= C (\xrightarrow{\alpha} S)^\ast D \\
C \xrightarrow{\alpha} S D \ &:= C = D \land \varphi \in C
\end{align*}
\]

**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., \{body \varphi \mid \varphi \in \{ \psi \in C \mid \text{isBox } \psi \}\}), where isBox tests for the shape \([a] \varphi^+ \) and body strips away the outer box yielding a set that is finite by construction.

Note that \( \xrightarrow{\alpha} S \), does not mention the satisfaction relation. This allows us to use \( \xrightarrow{\alpha} S \) to phrase the condition ensuring that \( \Rightarrow \) and \( \models \) behave as required.

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

\( (D) \) If \( [\alpha] \varphi^- \in C \in D \), then \( C \xrightarrow{\alpha} D \) and \( \varphi^- \in D \) for some \( D \in D \).

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

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

**Proof.** By induction on \( \alpha \). The assumption on formulas smaller than \( \alpha \) is required to handle the case for tests. \( \square \)

**Lemma 4.5.** Let \( \{C, D\} \subseteq D \) such that \( C \xrightarrow{\alpha} D \) and assume that \( \varphi^+ \in E \) implies \( E \not\models \psi \) for all \( E \in D \) and all \( \psi \) such that \( |\psi| < |\alpha| \). Then \( C \xrightarrow{\alpha} M(D) \).

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

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

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 \( S \subseteq U \) is called maximal if for all \( \phi \in F \) either \( \phi^* \subseteq C \) or \( \phi^- \subseteq C \).

The pruning function is defined recursively as follows:

\[
\text{prune } S := \begin{cases} 
\text{prune}(S \setminus \{ C \}) & \exists [\alpha] \phi^- \in C, \lnot \exists D \in S. \\
C \not\subseteq S \land \phi^- \in D & S \\
S & \text{S is a demo}
\end{cases}
\]

**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:

\[
S_0 := \{ C \subseteq U \mid C \text{ maximal and hintikka set} \} \\
D := \text{prune } S_0
\]

**Lemma 5.2.** \( D \) is a demo contained in \( S_0 \).

We say that a set of clauses \( S \) supports a clause \( C \), written \( S \vdash C \), if there exists some Hintikka set \( D \subseteq S \) such that \( C \subseteq D \). We have already established that a formula \( \phi \in F \) is satisfiable whenever \( D \vdash \{ \phi^+ \} \) (Theorem 4.6). In order to obtain completeness, it remains to show \( \lnot \phi \) whenever \( D \not\vdash \{ \phi^+ \} \). To prove this, we need to generalize from single formulas to clauses, i.e., prove \( \lnot \phi \) whenever \( D \not\vdash \phi \). When a clause \( C \) appears in the place of a formula, and in \( \lnot \phi \) above, it is to be read as the sign respecting conjunction of the formulas it contains, i.e., \( C \) is to be read as \( \land_{\phi^+ \in \phi} [\phi] \) where \( [\phi^-] := \lnot \phi \) and \( [\phi^+] := \phi \). If \( \lnot \phi \) is not maximal, 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 some clause \( C \subseteq U \) is no longer supported by the remaining clauses, we can prove \( \lnot C \), 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 \( D \) and then translate derivations of this inductive definition to proofs in the Hilbert system. The rules are given in fig. 3. If \( \text{pref } C \), we say that \( C \) is pruning refutable and if \( \text{pcoref } S \) for some \( S \subseteq S_0 \), say that \( S \) is pruning corefutable. In both rules, the set \( S \) corresponds to some intermediate stage of pruning and the premise \( \text{pcoref } S \) captures the intuition that we have already established \( \text{pref } C \) for all previously removed Hintikka clauses. The rule \( P1 \) then expresses the fact a clause cannot be supported by \( D \) if all clauses that could possibly support it have already been removed and the rule \( P2 \) corresponds to the pruning condition.

**Lemma 5.3.** \( D \) 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 \( D \vdash C \) 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 \( \lor_{C \in A} C \) as \( \lor A \). 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 \( \lnot C \) for all clauses in \( S \). \( \not\vdash \) \( S \setminus S \).

**Lemma 6.1.** Let \( C \subseteq U \) be maximal but not a Hintikka set. Then \( \vdash \lnot C \).

**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 sate-local 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 \( \vdash C \rightarrow \lor \{ D \subseteq S_0 \mid C \subseteq D \} \).

**Proof.** Since \( S \) is corefutable, it suffices to show \( \vdash C \rightarrow \lor \{ D \subseteq S_0 \mid C \subseteq D \} \). The claim follows by induction on \( |U| - |C| \). If \( C \) is maximal, then either \( C \subseteq S_0 \) or \( \lnot C \) (Lemma 6.1). Both cases are trivial. If \( C \) is not maximal then \( \vdash C \rightarrow C \lor \{ \phi^+ \} \lor C \lor \{ \phi^- \} \) for some \( \phi \in F \) such that \( \{ \phi^+, \phi^- \} \cap C = \emptyset \) 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 \( \lnot C \).
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 \( \vdash C \rightarrow \neg D \) whenever \( C \not\subseteq D \).

**Lemma 6.5.** Let \( S \subseteq S_0 \) be corefutable. Then
1. \( \vdash \psi \rightarrow \psi \).
2. \( \vdash (\psi \land \psi) \rightarrow \psi \).
3. \( \vdash (\psi \lor \psi) \rightarrow \psi \).
4. \( \vdash (\psi \rightarrow \psi) \rightarrow \psi \).
5. \( \vdash (\psi \rightarrow \psi) \rightarrow \psi \).
6. \( \vdash (\psi \lor \psi) \rightarrow (\psi \lor \psi) \).
7. \( \vdash (\psi \rightarrow \psi) \rightarrow \psi \).

Proof. (Claim 1) follows immediately with lemma 6.2. For (2) it suffices to show \( \vdash C \rightarrow \neg (\psi \land \psi) \) for all \( A \subseteq S \).

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 \( \neg \exists D \in S, C \varphi^\alpha D \land \varphi \in D \). Then \( \vdash \neg C \).
Completeness and Decidability of Converse PDL in Coq

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

$$\vdash C \rightarrow [\alpha](\land_{D \in X} \neg D)$$
$$\iff \vdash C \rightarrow \bigwedge_{D \in X} [\alpha] \neg D \quad \text{lemma 6.6(5)}$$
$$\iff \vdash C \rightarrow [\alpha] \neg D \quad D \in X$$

The last claim follows with lemma 6.7 since $C \not\vdash \neg C$ by assumption. For the second implication we have:

$$\vdash [\alpha](\land_{D \in X} \neg D) \rightarrow \langle \alpha \rangle X \rightarrow \bot$$
$$\iff \vdash \neg [\alpha] \varphi \rightarrow \langle \alpha \rangle X \quad \neg [\alpha] \varphi \in C$$
$$\iff \vdash \langle \alpha \rangle \neg \varphi \rightarrow \langle \alpha \rangle X \quad \text{lemma 6.6(4)}$$
$$\iff \vdash \{ \varphi^- \} \rightarrow \langle \alpha \rangle X \quad \text{lemma 6.6(1)}$$

This time, the last claim follows with lemma 6.2. Now it suffices to show

$$\vdash [\alpha](\land_{D \in X} \neg D) \rightarrow \langle \alpha \rangle(X) \rightarrow \bot$$
$$\iff \vdash \langle \alpha \rangle((\land_{D \in X} \neg D) \land \langle \alpha \rangle X) \rightarrow \bot \quad \text{lemma 6.6(3)}$$
$$\iff \vdash \langle \alpha \rangle((\land_{D \in X} \neg D) \land \langle \alpha \rangle X) \rightarrow \bot \quad \text{lemma 6.6(2)}$$

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 $\text{pref} C$ 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 $\varphi$, one can either construct a proof of $\neg \varphi$ or a model with at most $2^{2|\varphi|}$ states satisfying $\varphi$.

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

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

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

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

We say a predicate $P : X \rightarrow \text{Prop}$ is decidable if there is a function $p : X \rightarrow \mathbb{B}$ such that $P x \leftrightarrow (p x = \text{true})$ 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 $\varphi$ be satisfiable. Then $\varphi$ 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 $\alpha^-$. The satisfaction relation for CPDL is defined as for PDL with the interpretation of $\alpha^-$ defined as:

$$\alpha^-> \vdash w \Rightarrow v := v \Rightarrow w$$

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

$$\vdash \varphi \rightarrow [\alpha](\alpha^-)\varphi \quad (13)$$
$$\vdash \varphi \rightarrow [\alpha^-](\alpha)\varphi \quad (14)$$

By duality we obtain:

**Lemma 7.1.** $\vdash [\alpha^-] \varphi \rightarrow \varphi$ and $\vdash [\alpha] \varphi \rightarrow \varphi$.

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.

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

$$(\alpha \land \beta)^- \mapsto \alpha^- \land \beta^- \quad (\alpha\lnot) \mapsto \alpha^-$$

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

$$\text{cnf} : \text{formula} \rightarrow \text{formula}$$
$$\text{cnp} : \text{bool} \rightarrow \text{program} \rightarrow \text{program}$$

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.
with the respective induction hypotheses. It remains to show (lemma 6.6(7)). □

The last claim follows with lemma 7.1.

Lemma 7.3. Let \( D \) be a demo containing only converse normal formulas and let \([\alpha] \varphi^* \in C\) such that \( C \models D \) and for all \( E \in S\) and all \( \psi \) with \(|\psi| < |\alpha|\) we have that \( t^- \in E \) implies \( E \not\models \psi \). Then \( \varphi^* \in D \).

Proof. By induction on \( \alpha \). The case for \( \alpha^- \) is symmetric to the case for \( \alpha \). 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, \( C \models \) if \( D \) can now also fail because \( \varphi^* \notin C \) for some formula \([\alpha^-] \varphi \in D\). Secondly, we need to handle the case where \( C \not\models \) for \( D \). Both cases are straightforward.

Theorem 7.5. For every CPDL formula \( \varphi \), one can either construct a proof of \( \neg \varphi \) or a model with at most \(|\varphi|\) states satisfying \( \varphi \).

Proof. Let \( \varphi \) be some formula. Then \( \varphi \leftrightarrow \text{cnf} (\varphi) \) (lemma 7.2) and \(|U(\varphi)| \leq 4|\varphi|\) since \( [\text{cnf} (\varphi)] \leq 2|\varphi| \). 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 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. 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
We remark that rewriting with equivalences alone 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.\textsuperscript{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 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 \( \vdash [\alpha^* \neg \psi] \leftrightarrow [\neg \psi] \) 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 $[a]p^+$ and $p^+$ and the support relation is a recursively-defined decidable predicate corresponding to the Hintikka conditions, e.g.,

$$C \models (\phi \rightarrow \psi)^+ := C \models \phi^- \lor C \models \psi^+$$

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

$$C \models [a^*]p^+ := C \models \phi^- \land C \models [a][a^*]p$$

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 Hintikka 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.

References


