SAARLAND UNIVERSITY
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

BACHELOR'S THESIS

A FORMAL COMPLETENESS PROOF FOR TEST-FREE PDL

Author:
Joachim Bard

Advisor:
Dr. Christian Doczkal
Supervisor:
Prof. Dr. Gert Smolka

Reviewers:
Prof. Dr. Gert Smolka
Dr. Christian Doczkal

Submitted: 01st February 2017

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement in Lieu of an Oath:

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent:

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, 01st February, 2017

Abstract

In this thesis we give a formalization of test-free PDL based on results from the literature using the proof assistant Coq. We show soundness and completeness for a Hilbert system. In order to show completeness we use Hintikka sets and pruning. Pruning is a method yielding finite models for satisfiable formulas. For unsatisfiable formulas pruning yields abstract refutations. We obtain completeness of the Hilbert system by translating abstract refutations to proofs in the Hilbert system. As corollaries, we also obatin the small model property, plus that satisfiability, validity and Hilbert provability of a formula are decidable.

Acknowledgements

Foremost, I would like to thank my advisor Christian Doczkal for his guidance and patience. I am very grateful for his advice, support, and especially for improving my understanding of the given literature.

Moreover, I would like to thank my supervisor Prof. Gert Smolka, who taught me basically everything, I know about computational logic. I am also thankful for the possibility to write this thesis.

I would like to thank Prof. Smolka and Christian for reviewing my thesis.

I want to thank my family, friends and fellow students for their great advice and support. Special thank goes to Yannick for proofreading this thesis.

Contents

Chapter 1

Introduction

Propositional dynamic logic (PDL) was introduced by Fischer and Ladner [3]. PDL is a generalization of modal logic and is used to reason about regular programs α via the modality $[\alpha]s$. The meaning of the formula $[\alpha]s$ is that whenever one executes the program α in a state, formula s will hold upon termination. One also defines a dual modality $\langle\alpha\rangle s = \neg[\alpha]\neg s$. This dual modality states that there is a terminating execution of α, such that s will be true afterwards. Programs consist of atomic programs, concatenation $(\alpha;\beta)$, nondeterministic choice $(\alpha+\beta)$ and iteration (α^*), as well as tests ($?s$, where s is a formula). In addition to this regular PDL one does examine also restrictions and extensions. For the former it is common to remove iterations [6] or tests [7], because those give rise to some complexity. Popular extensions include a converse modality [5], nominals [4], or intersection of programs [1].

In this thesis we will give a formal completeness of a Hilbert system for test-free PDL. Completeness means we can establish a Hilbert proof for every formula, that is true on all states of all models. We will show the stronger result that for every formula s either yields a Hilbert proof of $\neg s$ or a model of bounded size satisfying s. This result is called informative completeness. In addition to completeness, we also obtain the existence of small models, plus decidability of validity, satisfiability and Hilbert provability for test-free PDL.

We state the basic definitions, including the Hilbert system, in chapter 2. The used Hilbert system is a variant of the one presented in [5]. We show soundness, i.e. every fomula provable in the Hilbert system is true on all states of all classical models. Classical models are a special class of models that are needed to show soundness constructively [2].

In chapter 3 we start from the subfomula closure [3] of a given fomula s. We construct a model, which satisfies all satisfiable subsets of that closure. This model, that we call demo, is built via pruning [8]. We begin by enumerating all maximal Hintikka subsets. Those subsets are connected by transitions, such that none of these transitions lead to a contradiction with a formula of the form $[a]s$. Still there can be subsets which contain formulas that violate a certain condition. Such subsets are one after another removed. Each removed subset is accompanied with an abstract refutation. The abstract refutation resembles the violated condition. So all unsatisfiable subsets are refuted.

The abstract refutations are translated into the Hilbert system in chapter 4. The essential part here is to dualize Kozen's and Parikh's "Lemma 1" of [7] to make it constructive. In combination with results from chapter 3 we obtain informative completeness.

In chapter 5 we will discuss, which problems need to be overcome to extend the approach presented in chapters 3 and 4 to full PDL.

Related Work

Fischer and Ladner [3] use quotient models to show that each satisfiable formula has a small model satisfying it. Pruning was developed by Pratt [8] to show that satisfiability is decidable in EXPTIME. He also states the Hintikka conditions we use. An elegant completeness proof was found by Kozen and Parikh [7]. Doczkal's PhD thesis [2] presents a Coq formalization of K and K^*. Both logics are sublogics of test-free PDL. The notions of informative completeness and abstract refutations are taken from there. We also reuse libraries for finite sets, pruning and propositional Hilbert proofs. The main difference is that there a demo consists of sets containing only literals. Literals can be seen as elementary formulas. A support relation is used to check, if such a set satisfies a general formula. While pruning can be phrased in terms of a support relation, we were unable to translate the resulting abstract refutations to proofs in the Hilbert system. Instead of literal sets and support we use Hintikka sets.

Contribution

To the best of our knowledge, we are the first to provide a formalization of test-free PDL together with its main results. We obtain these results by combining pruning with abstract refutations and tranlating abstract refutations into the Hilbert system.

Chapter 2

Test-free PDL

In this chapter we first state syntax and semantics for test-free PDL. We use a Hilbert system which is a variant of the one from [5]. We show soundness of this Hilbert system on classical models. That means every formula s, which can be proven in the Hilbert system, is true on all states of all classical models. The notion of a classical model is due to Doczkal [2]. This special class of models is needed to show soundness constructively.

2.1 Syntax

We define the syntax for test-free PDL:

$$\begin{aligned} s, t &::= x \mid \bot \mid s \to t \mid [\alpha]s && (x : \mathbb{N}) \\ \alpha, \beta &::= a \mid \alpha ; \beta \mid \alpha + \beta \mid \alpha^* && (a : \mathbb{N}) \end{aligned}$$

On top of this mimimal syntax we define derived operators:

$$\begin{aligned} \neg s &:= s \to \bot \\ \top &:= \neg \bot \\ s \vee t &:= \neg s \to t \\ s \wedge t &:= \neg(s \to \neg t) \\ \langle \alpha \rangle s &:= \neg [\alpha] \neg s \end{aligned}$$

We call formulas of the form x variables and programs of the form a atomic. Atomic programs correspond to one-step computations. Sequential programs $\alpha; \beta$ are executed by first executing α and then β. For programs like $\alpha + \beta$ one can choose to execute α or β. Iterated programs α^* execute α several times, including 0. Intuitively $[\alpha]s$ means, that whenever one executes program α in a state w, which terminates in state v, v satisfies formula s. Thus $\langle \alpha \rangle s$ requires a terminating execution of α ending in a state where s holds.

2.2 Models

Formulas are interpreted on transition systems, we call models. A model $\mathcal{M}$ consists of

- the type of states $|\mathcal{M}|$,
- a transition relation $\overset{a}{\Rightarrow}$ on these states for each atomic program a
- and a labeling Λ, which relates variables to states.

The transition relation $w \overset{a}{\Rightarrow} v$ states, that v is reachable from w by executing a. $\Lambda\, x\, w$ means, that the variable x holds at the given state w. For a model $\mathcal{M}$ we lift the transition relation for atomic programs to general programs:

$$\begin{aligned} w \overset{\alpha;\beta}{\Rightarrow} v &:= \exists u.\, w \overset{\alpha}{\Rightarrow} u \wedge u \overset{\beta}{\Rightarrow} v \\ w \overset{\alpha+\beta}{\Rightarrow} v &:= w \overset{\alpha}{\Rightarrow} v \vee w \overset{\beta}{\Rightarrow} v \\ w \overset{\alpha^*}{\Rightarrow} v &:= w(\overset{\alpha}{\Rightarrow})^* v \end{aligned}$$

$(\overset{\alpha}{\Rightarrow})^*$ is the reflexive, transitive closure of $\overset{\alpha}{\Rightarrow}$. Since this closure is defined inductively, we get an induction principle. We say v is α-reachable from w, if $w \overset{\alpha}{\Rightarrow} v$ holds. α-reachability corresponds to the execution of α. Using this notion of reachability we define the semantics of a formula on a state w of a model:

$$\begin{aligned} w \vDash x &:= \Lambda\, x\, w \\ w \vDash \bot &:= \bot \\ w \vDash s \to t &:= w \vDash s \to w \vDash t \\ w \vDash [\alpha]s &:= \forall v.\, w \overset{\alpha}{\Rightarrow} v \to v \vDash s \end{aligned}$$

We can now define classical and finite models. On classical models we know that $w \vDash s \vee w \nvDash s$ for all states w and all formulas s. This instance of excluded middle is needed to show soundness of the Hilbert system constructively ([2]). For finite models we demand that the transition relation and the labeling are decidable. Also finite models consist only of a finite number of states, hence the name finite. Because of the decidability requirements, finite models are also classical.

We say a model satisfies a fomula s, if there is a state of the model where s holds. A formula s is satisfiable, if there is a satisfying classical model. If all states of all classical models fulfill a formula s, then we call s valid.

2.3 Hilbert System and Soundness

In this section we state the Hilbert system for test-free PDL and prove it sound according to our semantics. The given Hilbert system (figure 2.1) is essentially from

$$\begin{array}{rl}
\mathsf{K} & \vdash s \to t \to s \\
\mathsf{S} & \vdash (u \to s \to t) \to (u \to s) \to u \to t \\
\mathsf{DN} & \vdash \neg\neg s \to s \\
\mathsf{N} & \vdash [\alpha](s \to t) \to [\alpha]s \to [\alpha]t \\
& \vdash [\alpha]s \to [\beta]s \to [\alpha + \beta]s \\
& \vdash [\alpha + \beta]s \to [\alpha]s \\
& \vdash [\alpha + \beta]s \to [\beta]s \\
& \vdash [\alpha;\beta]s \to [\alpha][\beta]s \\
& \vdash [\alpha][\beta]s \to [\alpha;\beta]s \\
& \vdash [\alpha^*]s \to s \\
& \vdash [\alpha^*]s \to [\alpha][\alpha^*]s \\
& \dfrac{\vdash s \to t \qquad \vdash s}{\vdash t}\,\mathsf{MP} \\
& \dfrac{\vdash s}{\vdash [\alpha]s}\,\mathsf{Nec} \\
& \dfrac{\vdash u \to [\alpha]u \qquad \vdash u \to s}{\vdash u \to [\alpha^*]s}\,\mathsf{SI}
\end{array}$$

Figure 2.1: Hilbert system for test-free PDL

[5] with some simplifications, since we don't need axioms for derived operators. The axioms K, S, DN and the modus ponens rule (MP) are well known from classical propositional logic. The normality axiom N allows us to apply modus ponens after doing some compution α. If a formula is always true, then it is clearly true after execution of a program. This is the statement of the necessity rule Nec. The star-induction rule SI is to be seen as follows. Provided we have an invariant u, that implies s and also carries over by running program α. That invariant guarantees, that after executing α several times s still holds. All other axioms give us the ability to reason about all kinds of programs.

We can now prove our first theorem, namely soundness. Soundness states, that each Hilbert provable formula is true on all states of all classical models.

Theorem 2.3.1 (Soundness). *If $\vdash s$, then $w \vDash s$ for all classical models $\mathcal{M}$ and all states w of $\mathcal{M}$.*

Proof. The proof follows by induction on $\vdash s$. In the case for $\vdash \neg\neg s \to s$ we need exactly our additional assumption of classical models. The case for the star-induction rule concludes by a nested induction on $(\overset{\alpha}{\Rightarrow})^*$. ■

We will show that Hilbert provablity and validity coincide. One direction is soundness (theorem 2.3.1). The other direction, from validity to a Hilbert proof, is called completeness. Completeness will be established across the next two chapters.

Chapter 3

Demos and Pruning

In this chapter we build a small model which satisfies all satisfiable finite sets C of a given universe. This model, called demo, is constructed iteratively. First we enumerate all finite sets of possible combinations of subformulas and their negations. For this we use the Fischer-Ladner closure [3]. Then we connect those sets with as many transitions as possible. By possible we mean that the adding of an transition does not lead to a contradiction with a fomula of the form $[a]s$. Now there might be sets C that contain formulas of the form $\neg[\alpha]s$. These formulas put a condition on the fragment of α-reachable sets. If that condition is not fulfilled, we have to remove C. We repeat that process until we cannot remove any clause. This construction of a demo is called pruning and was developed by Pratt [8]. Additionally we accompany each removed clause C with an abstract refutation. That abstract refutation states the reason for removing C. So this construction yields either an abstract refutation for C or a small model that satisfies C.

3.1 Demos

For convenience we annotate our formulas s with signs, i.e. s^+ or s^-. Each annotated formula s^σ can be transformed back into a formula by adding an extra negation for a negative sign:

$$\lfloor s^+ \rfloor := s \qquad \lfloor s^- \rfloor := \neg s$$

Also we define the reverse $\overline{\sigma}$ of a sign σ:

$$s^{\overline{+}} := s^- \qquad s^{\overline{-}} := s^+$$

We call finite sets of signed formulas clauses. A clause C is a Hintikka set, if it has the following properties:

$$\begin{aligned}
\bot^+ &\notin C \\
s^\sigma \in C &\to s^{\overline{\sigma}} \notin C \\
s \to t^+ \in C &\to s^- \in C \,\vee\, t^+ \in C \\
s \to t^- \in C &\to s^+ \in C \,\wedge\, t^- \in C \\
[\alpha;\beta]s^\sigma \in C &\to [\alpha][\beta]s^\sigma \in C \\
[\alpha+\beta]s^+ \in C &\to [\alpha]s^+ \in C \,\wedge\, [\beta]s^+ \in C \\
[\alpha+\beta]s^- \in C &\to [\alpha]s^- \in C \,\vee\, [\beta]s^- \in C \\
[\alpha^*]s^+ \in C &\to s^+ \in C \,\wedge\, [\alpha][\alpha^*]s^+ \in C \\
[\alpha^*]s^- \in C &\to s^- \in C \,\vee\, [\alpha][\alpha^*]s^- \in C
\end{aligned}$$

The first two requirements guarantee, that C is locally consistent. The ones concerning programs correspond to unfolding these programs by one step. These Hintikka conditions are taken from [8]. For Hintikka clauses C we have:

$$(\neg\neg s)^\sigma \in C \to s^\sigma \in C$$

This automatic removal of double negations is the reason for using signed formulas. We call a Hintikka set maximal according to a set of formulas F, if it contains each formula $s \in F$ positively (s^+) or negatively (s^-).

For $C, D \in S$ we define reachability in S, which is very similar to our reachability on models:

$$\begin{aligned}
C \overset{a}{\rightsquigarrow}_S D &:= \mathcal{R}_a\, C \subseteq D \qquad (\mathcal{R}_a\, C := \{s \mid [a]s \in C\}) \\
C \overset{\alpha;\beta}{\rightsquigarrow}_S D &:= \exists E \in S.\, C \overset{\alpha}{\rightsquigarrow}_S E \wedge E \overset{\beta}{\rightsquigarrow}_S D \\
C \overset{\alpha+\beta}{\rightsquigarrow}_S D &:= C \overset{\alpha}{\rightsquigarrow}_S D \vee C \overset{\beta}{\rightsquigarrow}_S D \\
C \overset{\alpha^*}{\rightsquigarrow}_S D &:= C(\overset{\alpha}{\rightsquigarrow}_S)^* D
\end{aligned}$$

$\mathcal{R}_a\, C$ is called the request of C, because it contains all formulas, that need to be fulfilled by all a-successors of C. Our definition is arranged such, that we don't miss a potential a-successor by taking every request fulfilling clause D as successor. So we connect the clauses in S by an a-transition if and only if that transition does not lead to a contradiction to a formula of the form $[a]s$.

Definition 3.1.1 (Demo)**.** *We call a finite set of maximal Hintikka clauses S a demo if:*

($D\Box$) *If $C \in S$ and $[\alpha]s^- \in C$, then $\exists D \in S.\; C \overset{\alpha}{\rightsquigarrow}_S D \,\wedge\, s^- \in D$.*

Such a demo S supports a clause C, if it contains a superset D of C.
Written: $S \triangleright C := \exists D \in S.\, C \subseteq D$

A demo S can be seen as a model in the following sense:

$$|\mathcal{M}_S| := S$$
$$C \overset{a}{\Rightarrow} D := \mathcal{R}_a C \subseteq D$$
$$\Lambda\, x\, C := x^+ \in C$$

Every state C of this model should satisfy each contained signed formula $s^\sigma \in C$. In order to get that result we show two simple lemmas.

Lemma 3.1.2. $C \overset{\alpha}{\Rightarrow} D \leftrightarrow C \overset{\alpha}{\rightsquigarrow}_S D$

Proof. In both directions the proof works by induction on α. ■

Lemma 3.1.3. *If* $[\alpha]s^+ \in C$ *and* $C \overset{\alpha}{\rightsquigarrow}_S D$*, then* $s^+ \in D$*.*

Proof. This follows easily by induction on α and construction of $\overset{\alpha}{\rightsquigarrow}_S$. ■

Lemma 3.1.4. *For every formula s we have $\mathcal{M}_S, C \models \lfloor s^\sigma \rfloor$, whenever $s^\sigma \in C$.*

Proof. The proof goes by induction on s. In the case for $[\alpha]s^+$ one uses the lemmas 3.1.3 and 3.1.2. The case for $[\alpha]s^-$ uses the demo condition to obtain a reachable D, which contains s^-. This suffices to complete the proof. ■

3.2 Subformula Closure

The subformula closure contains in addition to all trivial subformulas also some formulas, which are obtained by unfolding occuring programs according to the semantics. This closure was introduced by Fischer and Ladner ([3]), hence the name Fischer-Ladner closure. It can be computed as shown below, where `sub` s recurses on s and

$\mathsf{sub}_\Box\,[\alpha]s$ on α. Therefore sub and $\mathsf{sub}_\Box$ are structural recursive.

$$\begin{aligned}
\mathsf{sub}\,x &:= \{x\}\\
\mathsf{sub}\,\bot &:= \{\bot\}\\
\mathsf{sub}\,s \to t &:= \{s \to t\} \cup \mathsf{sub}\,s \cup \mathsf{sub}\,t\\
\mathsf{sub}\,[\alpha]s &:= \mathsf{sub}\,s \cup \mathsf{sub}_\Box\,[\alpha]s\\
\mathsf{sub}_\Box\,[a]s &:= \{[a]s\}\\
\mathsf{sub}_\Box\,[\alpha;\beta]s &:= \{[\alpha;\beta]s\} \cup \mathsf{sub}_\Box\,[\alpha][\beta]s \cup \mathsf{sub}_\Box\,[\beta]s\\
\mathsf{sub}_\Box\,[\alpha+\beta]s &:= \{[\alpha+\beta]s\} \cup \mathsf{sub}_\Box\,[\alpha]s \cup \mathsf{sub}_\Box\,[\beta]s\\
\mathsf{sub}_\Box\,[\alpha^*]s &:= \{[\alpha^*]s\} \cup \mathsf{sub}_\Box\,[\alpha][\alpha^*]s\\
\mathsf{sub}\,F &:= \bigcup_{s\in F} \mathsf{sub}\,s
\end{aligned}$$

A set of formulas F is subformula closed, if it satisfies the following conditions:

- If $s \to t \in F$, then $\{s,t\} \subseteq F$.
- If $[a]s \in F$, then $s \in F$.
- If $[\alpha;\beta]s \in F$, then $\{[\alpha][\beta]s, [\beta]s, s\} \subseteq F$.
- If $[\alpha+\beta]s \in F$, then $\{[\alpha]s, [\beta]s, s\} \subseteq F$.
- If $[\alpha^*]s \in F$, then $\{[\alpha][\alpha^*]s, s\} \subseteq F$.

We define a subformula universe U of a subformula closed set F as

$$U := \bigcup_{s\in F} \{s^+, s^-\}.$$

We show now some basic facts about the Fischer-Ladner closure.

Fact 3.2.1.

- $s \in \mathsf{sub}\,s$
- $|\mathsf{sub}\,s| \leq |s|$
- $\mathsf{sub}\,s$ *is subformula closed*
- $|U| \leq 2|F|$

Because of those facts, we can construct a subformula universe U, using the Fischer-Ladner closure. Its size is bounded, i.e. $|U| \leq 2|s|$.

$$\frac{C \subseteq U \qquad \text{coref } S \qquad S \not\triangleright C}{\text{ref } C}\,\mathsf{R1}$$

$$\frac{[\alpha]s^- \in C \qquad S \subseteq S_0 \qquad \text{coref } S \qquad \nexists D \in S.\, C \overset{\alpha}{\rightsquigarrow}_S D \wedge s^- \in D}{\text{ref } C}\,\mathsf{R2}$$

$$\text{coref } S := \forall C \in S_0 \setminus S.\ \text{ref } C$$

Figure 3.1: Abstract Refutations

3.3 Pruning and Refutations

We fix a subformula closed set of formulas F and its corresponding universe U. The goal of pruning is to build a demo. To that end we start with the set of all maximal Hintikka sets S_0. Then we check the demo condition $(D\Box)$ on all finitely many clauses $C \in S$. If this check is fulfilled, then we found a demo. Otherwise we remove a clause C violating the demo condition and start the check again. Since our Hintikka system is finite, we eventually stop removing clauses. Then there is no clause contradicting to the demo condition and thus we computed a demo. This process of checking and removing clauses is called pruning [8]. Doczkal formalized a general pruning method for his PhD thesis [2], which I reuse in my own formalization.

Complementing pruning, we want to give an abstract refutation calculus in figure 3.1. These refutations serve as reason, that S does not satisfy a clause C. Rule R2 mirrors exactly the demo condition $(D\Box)$. Therefore it gives the reason for removing C from S. It is also possible, that C is not a maximal Hintikka clause. But C might be extended to a maximal Hintikka clause $D \in S$. In order to be unsatisfied by S, such a $D \in S$ should not exist. Thus we have rule R1, which transfers the refutations to arbitrary clauses. Additionally we call S corefutable, if all so far removed clauses C are refutable.

Lemma 3.3.1. *Every demo is corefutable.*

Proof. This is due to the fact, that we only remove refutable clauses. ■

Theorem 3.3.2 (Pruning Completeness)**.** *Let $C \subseteq U$. Then either C is refutable, or there exists a finite model with at most $2^{2|F|}$ states, satisfying C.*

Proof. We do a case analysis whether the demo supports C. If so the demo gives us the desired model by lemma 3.1.4 and fact 3.2.1. Otherwise we can refute C by definition and using the fact that our demo is corefutable 3.3.1. ■

The previous theorem 3.3.2 will yield completeness, if we can translate abstract refutations into the Hilbert system. This translation will be shown in the next chapter.

Chapter 4

Hilbert Refutations

In this chapter we want to prove our main result, namely completeness of the Hilbert system. This means we have to provide a Hilbert proof $\vdash s$ for every valid formula s. We will show a stronger result, yielding either a Hilbert proof for $\neg s$ or a small model that satisfies s. Completeness is then obtained as a corollary. Using the last theorem from the previous chapter we just have to translate abstract refutations into the Hilbert system. The essential part of that proof is to dualize Kozen's and Parikh's "Lemma 1" of [7].

4.1 Translation of R1

A Hilbert refutation of a formula s means, we can show $\vdash \neg s$. Therefore S corefutable means, $\forall C \in S_0 \setminus S.\ \vdash \neg C$. Recall that S_0 is the set of all maximal Hintikka clauses of a subformula universe. If a clause C occurs in the Hilbert system, it should be read as $\bigwedge_{s^\sigma \in C} \lfloor s^\sigma \rfloor$.

In order to translate rule R1 of the abstract refutation calculus (figure 3.1) into the Hilbert system, we first show some basic lemmas.

Lemma 4.1.1. $\vdash C \to (\{s^+\} \cup C) \vee (\{s^-\} \cup C)$

Proof. Works by propositional reasoning, because we can show $\vdash s \vee \neg s$. ∎

Lemma 4.1.2. *Every maximal clause $C \subseteq U$, that is not a Hintikka clause, can be refuted.*

Proof. We do case analysis on a formula $s^\sigma \in C$ that contradicts the Hintikka conditions. In each case we have $t^{\sigma'} \notin C$ for a direct subformula t of s. Maximality yields $\overline{t^{\sigma'}} \in C$. Then the goal $\vdash \neg C$ follows by propositional reasoning and using the axioms from figure 2.1. ∎

We prove now a connection of arbitrary clauses and clauses of a corefutable set S.

Lemma 4.1.3. *Let S be corefutable. For each clause $C \subseteq U$ from our universe, we have $\vdash C \to \bigvee\{D \in S \mid C \subseteq D\}$*

Proof. It suffices to show $\vdash C \to \bigvee\{D \in S_0 \mid C \subseteq D\}$, because S is corefutable. Now we extend C with formulas using lemma 4.1.1 as long as the clause is not maximal. This yields a maximal clause $E \supseteq C$, such that $\vdash C \to E$. If E is a Hintikka clause, then $E \in S_0$ and so $\vdash E \to \bigvee\{D \in S_0 \mid C \subseteq D\}$. If E is not a Hintikka clause then lemma 4.1.2 will solve the goal. ∎

We show now that rule R1 of the abstract refutation calculus (figure 3.1) can be translated into the Hilbert system.

Lemma 4.1.4. *Let S be corefutable. Every clause $C \subseteq U$, that is not supported by S, can be refuted.*

Proof. We have to show $\vdash C \to \bot$. By lemma 4.1.3 we know $\vdash C \to \bigvee\{D \in S \mid C \subseteq D\}$. Since C is not supported by S, the big disjunction is empty and therefore equivalent to $\bot$. ∎

4.2 Translation of R2 and Completeness

We first show some useful axioms and rules, which can be derived in the Hilbert system.

Lemma 4.2.1.

(1) $\vdash [\alpha]s \to [\alpha]t$ *and* $\vdash \langle\alpha\rangle s \to \langle\alpha\rangle t$, *if* $\vdash s \to t$

(2) $\vdash \neg\langle\alpha\rangle\bot$

(3) $\vdash \langle\alpha\rangle s \to [\alpha]t \to \langle\alpha\rangle(s \wedge t)$

(4) $\vdash \neg[\alpha]s \to \langle\alpha\rangle\neg s$

(5) $\vdash [\alpha]s \wedge [\alpha]t \to [\alpha](s \wedge t)$

(6) $\vdash \langle\alpha\rangle(s \vee t) \to \langle\alpha\rangle s \vee \langle\alpha\rangle t$

Proof. All of them are proven easily. ∎

Bevor we translate the rule R2 of the abstract refutation calculus (figure 3.1), we show some basic lemmas.

Lemma 4.2.2. *If $S \subseteq S_0$ is corefutable, then $\vdash \bigvee_{C \in S} C$*

Proof.

$$\begin{aligned} & \vdash \bigvee_{C \in S} C \\ \Leftarrow & \vdash \bigvee \{C \in S \mid \emptyset \subseteq C\} \\ \Leftarrow & \vdash \emptyset \to \{C \in S \mid \emptyset \subseteq C\} \text{ and } \vdash \emptyset \end{aligned}$$

The resulting goal is proven by lemma 4.1.3 and the fact that the empty clause $\emptyset$ is equivalent to $\top$. ■

Lemma 4.2.3. *Let $S \subseteq S_0$ be corefutable. Then $\vdash \neg \bigvee_{C \in A} C \to \bigvee_{C \in S \setminus A} C$ holds for all subsets $A \subseteq S$ of S.*

Proof. By lemma 4.2.2 we get some $C \in S$ and we have to show $\vdash C \to \neg \bigvee_{C \in A} C \to \bigvee_{C \in S \setminus A} C$. If $C \in A$, we contradict to $\neg \bigvee_{C \in A} C$. Otherwise we conclude by propositional reasoning. ■

Lemma 4.2.4. *Given two different clauses $C \neq D$ from $S \subseteq S_0$. Then $\vdash C \to \neg D$.*

Proof. We have the two cases for $C \not\subseteq D$ and $D \not\subseteq C$. Since both are symmetrical, we can w.l.o.g. assume $C \not\subseteq D$. So there must be a $s^\sigma \in C$, such that $s^\sigma \notin D$. But then $s^{\overline{\sigma}} \in D$, because D is maximal. Therefore C and D are contradictory. ■

The next lemma is the dualized version of Kozen's and Parikh's "Lemma 1" in [7]. Thus it relies on maximality.

Lemma 4.2.5. *Let $S \subseteq S_0$ be corefutable. For each two clauses $C, D \in S$ from S, where $C \overset{\alpha}{\not\to}_S D$, we know $\vdash C \to [\alpha]\neg D$.*

Proof. By induction on α.

Case a**:** Since $\mathcal{R}_a C \not\subseteq D$, there is a $s^+ \in \mathcal{R}_a C$, such that $s^+ \notin D$. We have $[a]s^+ \in C$. $s^- \in D$ follows by maximality of D.

$$\begin{aligned} & \vdash C \to [a]\neg D \\ \Leftarrow & \vdash C \to \neg\langle a\rangle D \\ \Leftarrow & \vdash [a]s \to \langle a\rangle\neg s \to \bot \\ \Leftarrow & \vdash \langle a\rangle(\neg s \wedge s) \to \bot && \text{by lemma 4.2.1(3)} \\ \Leftarrow & \vdash \neg s \wedge s \to \bot \text{ and } \vdash \neg\langle a\rangle\bot && \text{by lemma 4.2.1(1)} \end{aligned}$$

The remaining goal is proven by propositional reasoning and lemma 4.2.1(2).

Case $\alpha;\beta$: For each $E \in S$ we have $C \overset{\alpha}{\not\rightsquigarrow}_S E$ or $E \overset{\beta}{\not\rightsquigarrow}_S D$.

$$\begin{aligned}
&\vdash C \to [\alpha;\beta]\neg D \\
\Leftarrow &\vdash C \to [\alpha][\beta]\neg D \\
\Leftarrow &\vdash C \to (\langle\alpha\rangle\neg[\beta]\neg D) \to \bot \\
\Leftarrow &\vdash C \to \langle\alpha\rangle((\bigvee_{E\in S} E) \wedge \langle\beta\rangle D) \to \bot && \text{by lemma 4.2.2} \\
\Leftarrow &\vdash C \to \langle\alpha\rangle(\bigvee_{E\in S} (E \wedge \langle\beta\rangle D)) \to \bot \\
\Leftarrow &\vdash C \to \bigvee_{E\in S} ((\langle\alpha\rangle(E \wedge \langle\beta\rangle D)) \to \bot && \text{by lemma 4.2.1(6)} \\
\Leftarrow &\vdash C \to \langle\alpha\rangle(E \wedge \langle\beta\rangle D) \to \bot \quad \text{for all } E \in S
\end{aligned}$$

If $C \overset{\alpha}{\not\rightsquigarrow}_S E$, then we have to show $\vdash [\alpha]\neg E \to \langle\alpha\rangle(E \wedge \langle\beta\rangle D) \to \bot$ by induction hypothesis for α. By lemma 4.2.1(1) it suffices to show $\vdash [\alpha]\neg E \to \langle\alpha\rangle E \to \bot$, which is easily proven. If $E \overset{\beta}{\not\rightsquigarrow}_S D$, then we have to show $\vdash \langle\alpha\rangle(E \wedge \langle\beta\rangle D) \to \bot$. It suffices to show $\vdash E \to \langle\beta\rangle D \to \bot$ by lemmas 4.2.1(1) and 4.2.1(2). By induction hypothesis we get $\vdash [\beta]\neg D \to \langle\beta\rangle D \to \bot$, which is easily proven.

Case $\alpha + \beta$: We have $C \overset{\alpha}{\not\rightsquigarrow}_S D$ and $C \overset{\beta}{\not\rightsquigarrow}_S D$.

$$\begin{aligned}
&\vdash C \to [\alpha+\beta]\neg D \\
\Leftarrow &\vdash [\alpha]\neg D \to [\beta]\neg D \to [\alpha+\beta]\neg D && \text{by IH for } \alpha \text{ and } \beta
\end{aligned}$$

This is an instance of an axiom in figure 2.1.

Case α^*: Let $I := \{E \in S \mid C \overset{\alpha^*}{\rightsquigarrow}_S E\}$ be the set of all reachable clauses from C. Note that $D \notin I$. We have to show $C \to [\alpha^*]\neg D$. So we use the star induction rule with $u := \bigvee_{E\in I} E$ as invariant.

- $\vdash C \to u$, since $C \in I$.
- $\vdash u \to \neg D$ follows by lemma 4.2.4, because every $E \in I$ is different from $D \notin I$.

- $\vdash u \to [\alpha]u$, since for all $E \in I$ we have:

$$\begin{array}{lll} & \vdash E \to [\alpha]u & \\ \Leftarrow & \vdash E \to (\langle\alpha\rangle\neg \bigvee_{G\in I} G) \to \bot & \\ \Leftarrow & \vdash E \to (\langle\alpha\rangle \bigvee_{G\in S\setminus I} G) \to \bot & \text{by lemma 4.2.3} \\ \Leftarrow & \vdash E \to (\bigvee_{G\in S\setminus I} \langle\alpha\rangle G) \to \bot & \text{by lemma 4.2.1(6)} \\ \Leftarrow & \vdash E \to \langle\alpha\rangle G \to \bot \quad \text{for an arbitrarily } G \in S \setminus I & \\ \Leftarrow & \vdash [\alpha]\neg G \to \langle\alpha\rangle G \to \bot & \text{by IH using } E \text{ for } C \\ & & \text{and } G \text{ for } D \end{array}$$

The remaining goal is proven by propositional reasoninig.

■

Using the lemma above we prove that rule R2 of the abstract refutation calculus (figure 3.1) can be translated into the Hilbert system.

Lemma 4.2.6. *Let $S \subseteq S_0$ be corefutable and $C \in S$, such that $[\alpha]s^- \in C$. Then C is refutable, whenever $\nexists D \in S.\, C \overset{\alpha}{\rightsquigarrow}_S D \wedge s^- \in D$.*

Proof. Let $X := \{D \in S \mid \{s^-\} \subseteq D\}$ be the set of all clauses that contain s^-. We first show:

$$\begin{array}{lll} & \vdash C \to [\alpha] \bigwedge_{D\in X} \neg D & \\ \Leftarrow & \vdash C \to \bigwedge_{D\in X} [\alpha]\neg D & \text{by lemma 4.2.1(5)} \\ \Leftarrow & \vdash C \to [\alpha]\neg D \quad \text{for all } D \in X & \end{array}$$

This holds by the previous proven lemma 4.2.5, because $s^- \in D$ and $C \overset{\alpha}{\not\rightsquigarrow}_S D$.

Secondly we proof:

$$\begin{array}{lll} & \vdash C \to \langle\alpha\rangle \bigvee_{D\in X} D & \\ \Leftarrow & \vdash \neg[\alpha]s \to \langle\alpha\rangle \bigvee_{D\in X} D & \\ \Leftarrow & \vdash \neg s \to \bigvee_{D\in X} D & \text{by lemmas 4.2.1(4) and 4.2.1(1)} \end{array}$$

Lemma 4.1.3 closes that proof.

Using these two Hilbert proofs, we can refute C:

$$\begin{aligned}
&\vdash C \to \bot \\
\Leftarrow &\vdash \langle\alpha\rangle \bigvee_{D\in X} D \to [\alpha] \bigwedge_{D\in X} \neg D \to \bot \\
\Leftarrow &\vdash \langle\alpha\rangle (\bigvee_{D\in X} D \wedge \bigwedge_{D\in X} \neg D) \to \bot && \text{by lemma 4.2.1(3)} \\
\Leftarrow &\vdash (\bigvee_{D\in X} D \wedge \bigwedge_{D\in X} \neg D) \to \bot && \text{by lemmas 4.2.1(1) and 4.2.1(2)}
\end{aligned}$$

Propositional reasoning closes the proof. ■

Lemma 4.2.7. *Every abstract refutation can be translated into a Hilbert refutation.*

Proof. By lemmas 4.1.4 and 4.2.6, we can translate abstract into Hilbert refutations. ■

We show now the main theorem.

Theorem 4.2.8 (Informative Completeness). *For every formula s we have $\vdash \neg s$ or there is a finite model with at most $2^{2^{|s|}}$ states satisfying s.*

Proof. We use our pruning completness theorem 3.3.2 with $\mathsf{sub}\, s$ for the subformula closed set F and $\{s^+\}$ for C. By lemma 4.2.7 we translate the abstract into a Hilbert refutation. ■

Corollary 4.2.9 (Completeness). *If s is valid, then $\vdash s$.*

Proof. By informative completeness 4.2.8 for $\neg s$ we have two cases to consider. The case $\vdash \neg\neg s$ yields $\vdash s$ by the double negation axiom (DN) and modus ponens (MP). If there is a model satisfying $\neg s$, then we contradict to the assumption that s is valid. ■

Using soundness (theorem 2.3.1) we can also show several decidabilty results and the small model property.

Corollary 4.2.10 (Decidability). *For each formula s, satisfiability, validity and Hilbert provability are decidable.*

Corollary 4.2.11 (Small Model Property). *Every satisfiable formula s is satisfied by a model with at most $2^{2^{|s|}}$ states.*

Chapter 5

PDL

In the previous chapters we examined test-free PDL. Now the natural question arises, whether the presented techniques also work, when dropping this restriction. We will discuss in this chapter, which problems need to be overcome to extend the approach to full PDL. Also nothing from this chapter is part of the formalization.

First we define formulas and programs of PDL:

$$s, t ::= x \mid \bot \mid [\alpha]s \qquad (x : \mathbb{N})$$
$$\alpha, \beta ::= a \mid \alpha;\beta \mid \alpha + \beta \mid \alpha^* \mid ?s \qquad (a : \mathbb{N})$$

We call programs of the form $?s$ tests. Note that we can define implications using tests:

$$s \to t := [?s]t$$

Due to $[\alpha]s$, formulas depend on programs. And also programs depend on formulas because of $?s$. Thus we cannot seperate the definition of formulas from the definition of programs. This mutual dependency leads to the fact that definitions have to treat formulas and programs in parallel. For example, the definition of models and reachability cannot be seperated:

$$w \overset{?s}{\Rightarrow} w := w \vDash s$$
$$w \vDash [\alpha]s := \forall v.\, w \overset{\alpha}{\Rightarrow} v \to v \vDash s$$

For simplifications we omit the other cases. Observe that the execution of $?s$ does not change the state. Therefore $w \vDash [?s]t$ is equivalent to $w \vDash s \to w \vDash t$. This justifies the definition of implication.

The mutual dependency does not only appear in definitions but can also occur in proofs. If we want to show a statement using induction on a formula, then we have to show the case for $[\alpha]s$. For test-free PDL one often proves this by a nested induction on α. This will not work for full PDL. But we can prove the original statement and the lemma at once. Then the induction hypothesis will be available for each subexpression, no matter if it is a formula or a program. However it will increase the technicality of the proofs.

Beside that technicality we are very confident that the presented approach can be extended to full PDL. In chapter 3 we build a demo using pruning. Since Pratt [8] used pruning on full PDL, we should be able to include tests in the definition of the demo. The crucial part of the translation of abstract refutations into the Hilbert system (chapter 4) was the dualized version of "Lemma 1" of [7]. Kozen and Parikh state that their completeness proof also works for full PDL.

Overall the main problem will probably be, how well Coq behaves on mutual inductive definitions. It might be interesting to analyze, in what degree Coq is suited for such definitions.

Chapter 6

Conclusion

We gave a formalization of test-free PDL and showed soundness and completeness of a Hilbert system. Completeness was obtained from demos and pruning. Pruning was completed by abstract refutations. Those abstract refutations were translated into the Hilbert system. Additionally this construction yielded the small model property, and also that satisfiability, validity and Hilbert provability are decidable.

All results (except for chapter 5) were carried out in the proof assistant Coq. The development[1] consits of about 1000 lines, not inlcuding any library. Nearly half of them are specifications and the rest of them being part of proofs. It was compiled using Coq 8.5 and the Ssreflect[2] library version 1.6. We could also reuse Doczkal's libraries [2] on finite sets, pruning and Hilbert proofs.

[1]Available at `https://www.ps.uni-saarland.de/~bard/bachelor.php`
[2]Ssreflect can be found at `http://math-comp.github.io/math-comp/`

Bibliography

[1] Ryszard Danecki. Nondeterministic propositional dynamic logic with intersection is decidable. In *Computation Theory - Fifth Symposium, Zaborów, Poland, December 3-8, 1984, Proceedings*, pages 34–53, 1984.

[2] Christian Doczkal. *A Machine-Checked Constructive Metatheory of Computation Tree Logic*. PhD thesis, Saarland University, Mar 2016.

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

[4] Mark Kaminski. *Incremental Decision Procedures for Modal Logics with Nominals and Eventualities*. PhD thesis, Saarland University, Feb 2012.

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

[6] Dexter Kozen. A representation theorem for models of *-free PDL. In *Automata, Languages and Programming, 7th Colloquium, Noordweijkerhout, The Netherland, July 14-18, 1980, Proceedings*, pages 351–362, 1980.

[7] Dexter Kozen and Rohit Parikh. An elementary proof of the completeness of PDL. *Theor. Comput. Sci.*, 14:113–118, 1981.

[8] Vaughan R. Pratt. Models of program logics. In *20th Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 29-31 October 1979*, pages 115–122, 1979.