A Proof from 1988 that PDL has Interpolation?

Manfred Borzechowski

EDV-Beratung Manfred Borzechowski
Berlin, Germany

Malvin Gattinger [1]

University of Groningen
Groningen, The Netherlands

Abstract

Multiple arguments that Propositional Dynamic Logic has Craig Interpolation have been published, but one has been revoked and the status of the others is unclear. Here we summarise a proof attempt originally written by the first author in German in 1988. We also make available the original text and an English translation.
The proof uses a tableau system with annotations. Interpolants are defined for partitioned nodes, going from leaves to the root with appropriate definitions for each rule. To prevent infinite branches generated by the $*$ operator, additional marking rules are used. In particular, nodes are also defined as end nodes when they have a predecessor with the same set of formulas along a branch with the same marking.
We end with open questions about the proof idea and connections to more recent related work on non-wellfounded proof systems.

Keywords: Propositional Dynamic Logic, Craig Interpolation, Tableau.

1 Introduction

Propositional Dynamic Logic (PDL) from [4] is a well-known modal logic which is both expressive and well-behaved. PDL can express common programming constructs such as conditionals and loops, but also has a small model property.

A logic has Craig Interpolation (CI) iff for any validity $\phi \to \psi$ there exists a formula θ in the vocabulary that is used both in ϕ and in ψ such that $\phi \to \theta$ and $\theta \to \psi$ are valid. The formula θ is then called an *interpolant.*

For PDL the vocabulary includes atomic propositions and atomic programs. For example, $[(A \cup B)^*](p \land q) \to [(B;B)^*](q \lor r)$ is valid in PDL and $[B^*]q$ is an interpolant for this validity. But whether such interpolants always exist, i.e. whether PDL has CI, has been studied for more than four decades and is still unknown. The key challenge is how to systematically find interpolants for

[1] Corresponding author: `malvin@w4eg.eu`

validities involving the star operator a^* which denotes arbitrary finite iteration of a program a. There have been at least the following three proof attempts:

- Daniel Leivant in [10] from 1981. This article presents a sequent calculus including a rule for $*$ with infinitely many premises. This rule is then replaced with a finitary rule and an intuitionistic variant of the system is defined. Interpolation is then shown in the intuitionistic system using Maeharas Method, defining interpolants for each node in a proof [12, p. 33]. Interpolants for $*$ are defined via fixed points of matrices of programs.

  In [9] it is said that it was not "possible to verify the argument" and claimed that the finitary rule is problematic. But the rule can be validated using the finite model property of PDL, as argued in [5]. Still, other parts of the argument, e.g. the translation to the intuitionistic system, seem problematic. As far as we know, the status of this argument is currently unknown [6].
- Manfred Borzechowski in [2] from 1988. The idea here is similar to [10], but using a tableau system instead of a sequent calculus. This text is also criticised in [9], but without any specific argument.
- Tomasz Kowalski in [7] from 2002. This algebraic proof was officially retracted [8] in 2004, after a flaw was pointed out by Yde Venema.

The correctness of the first two texts is still the subject of discussions. In this note we summarise the proof attempt from [2]. This diploma thesis was written under the supervision of Wolfgang Rautenberg at FU Berlin, but not published. Together with this summary we make available the original German text and an English translation at `https://malv.in/2020/borzechowski-pdl`. Page numbers refer to the German text, but are also shown in the translation.

We use the following notation: p, q, etc. are atomic propositional variables, P, Q, etc. are formulas from $P ::= p \mid \neg P \mid P \wedge Q \mid [a]P$. Moreover, A, B, etc. are atomic programs and a, b, etc. are programs from $a ::= A \mid a;a \mid a \cup a \mid a^* \mid P?$. We do not repeat the semantics for PDL here — see the original page 6 or [4].

Section 2 provides an overview of the tableau system, Section 3 describes the main idea how to define interpolants, and Section 4 lists open questions.

2 Tableaux for PDL

The system is defined below. We read rules top-down and use "$\ldots \mid \ldots$" for branches. The Boolean rules and those for PDL constructs besides $*$ are standard. The *critical rule* (At) for atomic programs uses $X_A := \{P \mid [A]P \in X\}$ which corresponds to a transition to another state in a Kripke model.

To deal with the $*$ operator and to prevent infinitely long branches, the system uses the following two non-standard features and extra condition 6.

Nodes with n-formulas. The $(\neg n)$ rule is essentially a diamond rule for the $*$ operator. It also replaces $*$ by the string '(n)'. Formulas with '(n)' are n-*formulas*, in contrast to *normal* formulas. An n-formula becomes normal again by extra condition 1, which applies iff an atomic modality is reached.

Markings. Formulas can be marked with other formulas as upper indices, using the loading rule $(M+)$. Nodes with marked formulas are called *loaded*, in contrast to *free*. Markings can be removed or changed by $(M-)$, $(\neg n)$ or $(\neg ?)$.

Definition 2.1 A *PDL tableau* is a finite tree generated according to the following rules and in addition adhering to the seven extra conditions below.
The classical rules:

$$(\neg)\,\frac{X; \neg\neg P}{X; P} \qquad (\wedge)\,\frac{X; P \wedge Q}{X; P; Q} \qquad (\neg\wedge)\,\frac{X; \neg(P \wedge Q)}{X; \neg P \;\mid\; X; \neg Q}$$

The local rules:

$$(\neg\cup)\,\frac{X; \neg[a \cup b]P}{X; \neg[a]P \;\mid\; X; \neg[b]P} \qquad (\neg ?)\,\frac{X; \neg[Q?]P}{X; Q; \neg P} \qquad (\neg ;)\,\frac{X; \neg[a; b]P}{X; \neg[a][b]P}$$

$$(\cup)\,\frac{X; [a \cup b]P}{X; [a]P; [b]P} \qquad (?)\,\frac{X; [Q?]P}{x; \neg Q \;\mid\; X; P} \qquad (;)\,\frac{X; [a; b]P}{X; [a][b]P}$$

$$(\neg n)\,\frac{X; \neg[a^*]P}{X; \neg P \;\mid\; X; \neg[a][a^{(n)}]P} \qquad (n)\,\frac{X; [a^*]P}{X; P; [a][a^{(n)}]P}$$

The PDL rules:

$$(M+)\,\frac{X; \neg[a_0] \ldots [a_n]P}{X; \neg[a_0] \ldots [a_n]P^P}\ X \text{ free} \qquad \text{the } \textit{loading rule},$$

$$(M-)\,\frac{X; \neg[a]P^R}{X; \neg[a]P} \qquad \text{the } \textit{liberation rule},$$

$$(At)\,\frac{X; \neg[A]P^R}{X_A; \neg P^{R\setminus P}} \qquad \text{the } \textit{critical} \text{ rule.}$$

The marked rules (where $\ldots^{R\setminus P}$ indicates that R is removed iff $R = P$):

$$(\neg\cup)\,\frac{X; \neg[a \cup b]P^R}{X; \neg[a]P^R \;\mid\; X; \neg[b]P^R} \qquad (\neg ;)\,\frac{X; \neg[a; b]P^R}{X; \neg[a][b]P^R}$$

$$(\neg n)\,\frac{X; \neg[a^*]P^R}{X; \neg P^{R\setminus P} \;\mid\; X; \neg[a][a^{(n)}]P^R} \qquad (\neg ?)\,\frac{X; \neg[Q?]P^R}{X; Q; \neg P^{R\setminus P}}$$

1. Instead of a node $X; \neg[A]P$ or $X; [A]P$ with an n-formula P we always obtain the node $X; \neg[A]f(P)$ or $X; [A]f(P)$, respectively, where $f(P)$ is obtained by replacing (n) with $*$.
2. Instead of a node $X; [a^{(n)}]P$ we always obtain the node X.
3. A rule must be applied to an n-formula whenever it is possible.
4. No rule may be applied to a $\neg[a^{(n)}]$-node.
5. To a node obtained using $(M+)$ we may not apply $(M-)$.
6. If a normal node t has a predecessor s with the same formulas and the path $s \ldots t$ uses (At) and is loaded if s is loaded, then t is an end node.
7. Every loaded node that is not an end note by condition 6 has a successor.

Claim 2.2 *The system from Definition 2.1 is sound and complete for PDL.*

The full completeness proof is contained in sections 1.8 to 1.10 of the original text. The main idea is to construct a Kripke model from an open tableau.

3 Interpolation via Tableaux

We claim that the tableau system can be used to show interpolation. We first define interpolants for partitioned sets of formulas. A partitioned set X is a disjoint union of two subsets X_1, X_2. We write it as $X = X_1/X_2$.

Definition 3.1 A formula θ is an interpolant for a partitioned set X_1/X_2 iff θ is in the vocabulary of that is used in both X_1 and X_2 and the two sets $X_1 \cup \{\neg\theta\}$ and $\{\theta\} \cup X_2$ are both inconsistent.

Corollary 3.2 *A formula θ is an interpolant for a validity $\phi \to \psi$ iff θ is an interpolant for the partitioned set X_1/X_2 given by $X_1 = \{\phi\}$ and $X_2 = \{\neg\psi\}$.*

To find an interpolant for a validity $\phi \to \psi$ we start a tableau with $\phi/\neg\psi$ as its root. This tableau is built as usual from the root to the leaves, applying the rules to partitioned sets. Then we go in the opposite direction: starting at the leaves, we define an interpolant for each node. Depending on the rule which was applied, we use the interpolant(s) of the child node(s) to define a new interpolant for the parent node. In addition, the interpolant might depend on whether the active formula in a rule application is in the left or right side of the partition. As mentioned above, this is similar to Maehara's Method for sequent calculi [12, p. 33]. We discuss two rules as examples here.

Interpolating $(\neg\cup)$**.** Suppose we use $(\neg\cup)$ in the right set. Given two interpolants θ_a and θ_b for $X_1/X_2; \neg[a]P$ and $X_1/X_2; \neg[b]P$ respectively, we define the new interpolant $\theta := \theta_a \wedge \theta_b$ for the parent node $X_1/X_2; \neg[a \cup b]P$. Similarly, on the left side we would use $\theta := \theta_a \vee \theta_b$ for $X_1; \neg[a \cup b]P/X_2$.

Interpolating (At)**.** Suppose we use (At) in the left set to go from a parent node $\neg[A]\phi; Y_1/Y_2$ to a child node $\neg\phi; (Y_1)_A/(Y_2)_A$. Suppose θ_A is an interpolant for the child node. Then $\neg\phi; (Y_1)_A; \neg\theta_A$ and $(Y_2)_A; \theta_A$ are inconsistent. We now want an interpolant for the parent, i.e. a θ such that $\neg[A]\phi; Y_1; \neg\theta$ and $Y_2; \theta$ are inconsistent. A solution is to set $\theta := \langle A \rangle \theta_A$, unless Y_2 is empty, in which case we are not allowed to use A, so we ignore θ_A and let $\theta := \bot$. Similarly, if (At) is applied in the right set we use $\theta := [A]\theta_A$, unless Z_1 is empty, in which case we let $\theta := \top$.

We refer to the original text for two examples. A closed tableau for the set $\{\neg[(A \cup p?)^*]q, [A^*]q\}$ is given on page 29 and an interpolant for $[(A;A)^*](p \wedge [A;(B \cup C)]0) \to [A^*](p \vee [C]q)$ is computed in Section 2.4: $[A^*](p \vee [C]0)$.

4 Open Questions

The previous two sections provide only a high-level overview of the argument. To verify it completely we will further study the following two main questions:

- How exactly are the existence lemma and completeness of the system shown? In particular, what is the role of *first free normal successor nodes*?

- How are interpolants defined for end nodes due to condition 6? The original text uses the extra tableaux $\mathcal{T}^I$ and $\mathcal{T}^J$ for this, what is their role?

If the proof can be verified, there are of course further questions:

- Can we simplify the proof to only consider test-free PDL?
- How does the system compare to recent work on infinitary and non-wellfounded systems, such as [1] for μ-calculus and [3] for PDL?
- Can interpolation be efficiently implemented into an automated prover? We have started to implement parts of the given system, similar to how the star-free fragment of [10] was implemented by [11].

To conclude, we hope that this summary will help to further scrutinise the proof and encourage the interested participant of AiML 2020 to contact us.

References

[1] Afshari, B., G. Jäger and G. E. Leigh, *An infinitary treatment of full mu-calculus*, in: R. Iemhoff, M. Moortgat and R. de Queiroz, editors, *Workshop on Logic, Language, Information, and Computation (WoLLIC)*, 2019, pp. 17–34.
URL `https://doi.org/10.1007/978-3-662-59533-6_2`

[2] Borzechowski, M., *Tableau-Kalkül für PDL und Interpolation* (1988), Diplomarbeit.
URL `https://malv.in/2020/borzechowski-pdl/`

[3] Docherty, S. and R. N. S. Rowe, *A non-wellfounded, labelled proof system for propositional dynamic logic*, in: S. Cerrito and A. Popescu, editors, *Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 2019)*, Lecture Notes in Computer Science **11714**, 2019, pp. 335–352.
URL `http://arxiv.org/abs/1905.06143`

[4] Fischer, M. J. and R. E. Ladner, *Propositional dynamic logic of regular programs*, Journal of Computer and System Sciences **18** (1979), pp. 194–211.
URL `https://doi.org/10.1016/0022-0000(79)90046-1`

[5] Gattinger, M., *Craig Interpolation of PDL – A report on the proof by Daniel Leivant (1981)* (2014).
URL `https://w4eg.de/malvin/illc/pdl.pdf`

[6] Gattinger, M. and Y. Venema, *Interpolation for PDL: an open problem?*, in: *Circularity in Syntax and Semantics*, 2019, p. 26, talk and abstract only.
URL `http://www.cse.chalmers.se/~bahafs/CiSS2019/CiSS2019BoA.pdf`

[7] Kowalski, T., *PDL has interpolation*, Journal of Symbolic Logic **67** (2002), pp. 933–946.
URL `https://doi.org/10.2178/jsl/1190150141`

[8] Kowalski, T., *Retraction note for "PDL has interpolation"*, Journal of Symbolic Logic **69** (2004), pp. 935–936.
URL `https://doi.org/10.2178/jsl/1096901777`

[9] Kracht, M., "Tools and Techniques in Modal Logic," 1999.
URL `https://wwwhomes.uni-bielefeld.de/mkracht/html/tools/book.pdf`

[10] Leivant, D., *Proof theoretic methodology for propositional dynamic logic*, in: J. Díaz and I. Ramos, editors, *Formalization of Programming Concepts (ICFPC 1981)*, Lecture Notes in Computer Science **107**, 1981, p. 356–373.
URL `https://doi.org/10.1007/3-540-10699-5_111`

[11] Perin, F., *Implementing Maehara's method for star-free Propositional Dynamic Logic* (2019), Bachelor's Thesis, University of Groningen.
URL `https://fse.studenttheses.ub.rug.nl/20770/`

[12] Takeuti, G., "Proof Theory," Studies in logic and the foundations of mathematics **81**, North-Holland, 1975.