Towards Formalizing Cyclic Tableaux and Interpolation for PDL in Lean

Malvin Gattinger

11 June 2025, TU Wien

Outline

Overview

What is Interpolation?

If god exists, then the world will never end and all humans and cats will live forever.

$\Rightarrow$ If god exists and Mia is a cat, then Mia will live forever.

© "Simon's Cat"

What is Interpolation?

If god exists, then the world will never end and all humans and cats will live forever.

$\Rightarrow$ If god exists, then all cats will live forever.

$\Rightarrow$ If god exists and Mia is a cat, then Mia will live forever.

Craig-Interpolation: Definition

Given:

- Λ – logic: set of validities, by semantics or proof system
- $L(\phi)$ – language of a formula
  e.g. $L(p \to ((r \lor p) \land q)) = \{p, q, r\}$

Definition

A logic Λ has *Interpolation* iff for any $\phi \to \psi \in \Lambda$, there is a θ s.t.:

- $L(\theta) \subseteq L(\phi) \cap L(\psi)$,
- $\phi \to \theta \in \Lambda$
- and $\theta \to \psi \in \Lambda$.

We call θ an *interpolant* for $\phi \to \psi$.

Craig-Interpolation: Definition

Given:

- Λ – logic: set of validities, by semantics or proof system
- $L(\phi)$ – language of a formula
  e.g. $L(p \to ((r \vee p) \wedge q)) = \{p, q, r\}$

Definition

A logic Λ has *Interpolation* iff for any $\phi \to \psi \in \Lambda$, there is a θ s.t.:

- $L(\theta) \subseteq L(\phi) \cap L(\psi)$,
- $\phi \to \theta \in \Lambda$
- and $\theta \to \psi \in \Lambda$.

We call θ an *interpolant* for $\phi \to \psi$.

Examples

Given $(q \vee (r \wedge s)) \;\to\; (\neg q \to (t \vee s))$ we find $\theta = q \vee s$.

Craig-Interpolation: Definition

Given:

- Λ – logic: set of validities, by semantics or proof system
- $L(\phi)$ – language of a formula
  e.g. $L(p \to ((r \vee p) \wedge q)) = \{p, q, r\}$

Definition

A logic Λ has *Interpolation* iff for any $\phi \to \psi \in \Lambda$, there is a θ s.t.:

- $L(\theta) \subseteq L(\phi) \cap L(\psi)$,
- $\phi \to \theta \in \Lambda$
- and $\theta \to \psi \in \Lambda$.

We call θ an *interpolant* for $\phi \to \psi$.

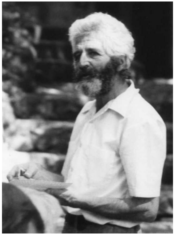
Examples

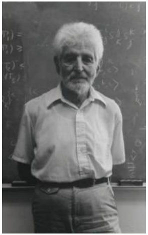
Given $(q \vee (r \wedge s)) \ \to \ (\neg q \to (t \vee s))$ we find $\theta = q \vee s$.

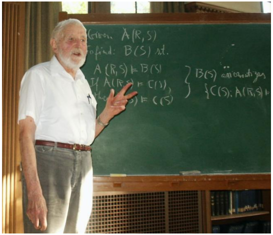
Given $(Eg \to Lw \wedge \forall x : (Hx \vee Cx \to Ix)) \ \to \ (Eg \to Cm \to Im)$ we find $\theta = Eg \to \forall x : (Cx \to Ix)$

Logics that have Craig Interpolation

1957: William Craig shows interpolation for First-Order-Logic

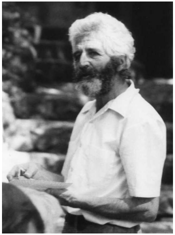
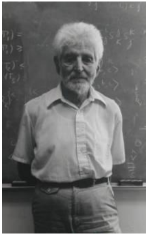
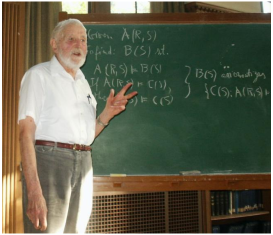






Logics that have Craig Interpolation

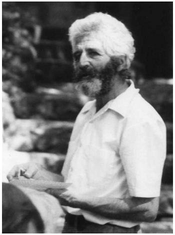
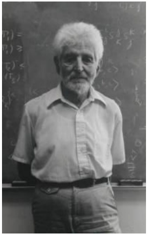
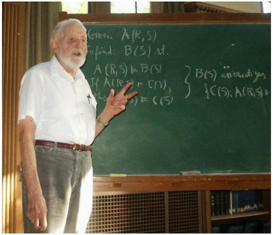
1957: William Craig shows interpolation for First-Order-Logic

- Propositional Logic ✓
- First-Order Logic ✓
- Intuitionistic Logic ✓
- Basic and Multi-modal logic ✓
- μ-Calculus ✓ (even has *uniform* interpolation)
- ...

Logics that have Craig Interpolation

1957: William Craig shows interpolation for First-Order-Logic

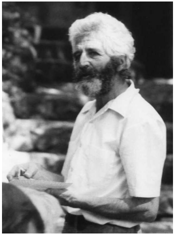
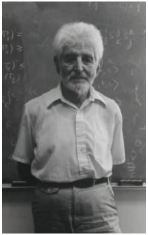
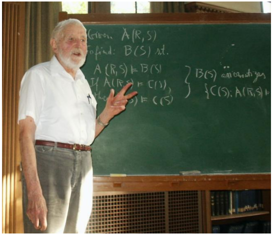




- Propositional Logic ✓
- First-Order Logic ✓
- Intuitionistic Logic ✓
- Basic and Multi-modal logic ✓
- μ-Calculus ✓ (even has *uniform* interpolation)
- ...

- Propositional Dynamic Logic (PDL) ?

Logics that have Craig Interpolation

1957: William Craig shows interpolation for First-Order-Logic

- Propositional Logic ✓
- First-Order Logic ✓
- Intuitionistic Logic ✓
- Basic and Multi-modal logic ✓
- μ-Calculus ✓ (even has *uniform* interpolation)
- ...

- Propositional Dynamic Logic (PDL) ?
  Yes, but the history is a mess.

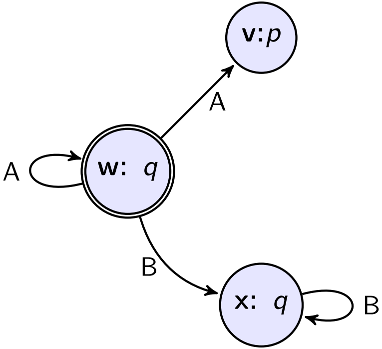
What is PDL

Propositional Dynamic Logic by Fischer and Ladner (1979)

"fundamental propositional logical system based on modal logic for describing correctness, termination and equivalence of programs."

Related to: regular expressions, automata theory, multi-agent knowledge, programming language semantics, . . .

PDL: Example



$$\mathcal{M}, w \models \langle A \rangle q$$
$$\mathcal{M}, w \models \neg [A] q$$
$$\mathcal{M}, w \models \langle A; B \rangle q$$
$$\mathcal{M}, w \models [B^*] q$$
$$\mathcal{M}, w \models [B; A] \bot$$

$$\mathcal{M}, w \models \langle A \rangle (\langle A \rangle \neg q \land \langle B \rangle [B^*] q)$$

PDL: Basic Definitions

Syntax

Formulas and Programs:

$$\begin{array}{rcl} \phi & ::= & p \mid \neg\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \phi \to \phi \mid \langle\alpha\rangle\phi \\ \alpha & ::= & A \mid \alpha;\alpha \mid \alpha \cup \alpha \mid \alpha^* \mid \phi? \end{array}$$

Models

$\mathcal{M} = (W, \mathcal{R}, V)$ where

- W: set of worlds/states
- $\mathcal{R} = (R_a)_a$: family of binary relations on W
- V : Prop $\to \mathcal{P}(W)$: valuation function

PDL: Basic Definitions

Syntax

Formulas and Programs:

$$\begin{array}{lcl} \phi & ::= & p \mid \neg\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \phi \to \phi \mid \langle\alpha\rangle\phi \\ \alpha & ::= & A \mid \alpha;\alpha \mid \alpha \cup \alpha \mid \alpha^* \mid \phi? \end{array}$$

Models

$\mathcal{M} = (W, \mathcal{R}, V)$ where

- W: set of worlds/states
- $\mathcal{R} = (R_a)_a$: family of binary relations on W
- V : Prop $\to \mathcal{P}(W)$: valuation function

```
structure KripkeModel (W : Type) : Type where
  val  : W → Nat → Prop
  Rel  : Nat → W → W → Prop
```

PDL: Semantics

Semantics of formulas:

- $\mathcal{M}, w \vDash p$ iff $w \in V(p)$
- $\mathcal{M}, w \vDash \neg\phi$ iff $\mathcal{M}, w \nvDash \phi$
- $\mathcal{M}, w \vDash \phi \vee \psi$ iff $\mathcal{M}, w \vDash \phi$ or $\mathcal{M}, w \vDash \psi$
- $\mathcal{M}, w \vDash \phi \wedge \psi$ iff $\mathcal{M}, w \vDash \phi$ and $\mathcal{M}, w \vDash \psi$
- $\mathcal{M}, w \vDash \phi \to \psi$ iff $\mathcal{M}, w \nvDash \phi$ or $\mathcal{M}, w \vDash \psi$
- $\mathcal{M}, w \vDash [\alpha]\phi$ iff for all $w' \in W: \; wR_\alpha w' \; \Rightarrow \; \mathcal{M}, w' \vDash \phi$.

Semantics of programs:

- $R_{\chi;\xi} := R_\chi ; R_\xi$ (consecution)
- $R_{\chi \cup \xi} := R_\chi \cup R_\xi$ (union)
- $R_{\chi^*} := (R_\chi)^*$ (reflexive-transitive closure)
- $R_{\phi?} := \{w \in W \mid w \vDash \phi\}$ (reflexive test)

PDL: Language of a formula or program

$$
\begin{array}{lcl}
L(p) & := & \{p\} \\
L(\phi \wedge \psi) & := & L(\phi) \cup L(\psi) \\
L(\phi \vee \psi) & := & L(\phi) \cup L(\psi) \\
L(\phi \to \psi) & := & L(\phi) \cup L(\psi) \\
L(\langle \tau \rangle \phi) & = & L(\tau) \cup L(\phi)
\end{array}
$$

$$
\begin{array}{lcl}
L(a) & := & \{a\} \\
L(\sigma; \tau) & := & L(\sigma) \cup L(\tau) \\
L(\sigma \cup \tau) & := & L(\sigma) \cup L(\tau) \\
L(\sigma^*) & := & L(\sigma)
\end{array}
$$

Example: $L([a; b]p \to \langle c \rangle q) = \{a, b, c, p, q\}$

The Question

Does PDL have interpolation? 🤔

Example: $[(a \cup b)^*](p \wedge q) \rightarrow [b^*](q \vee r)$ is valid in PDL.

The Question

Does PDL have interpolation? 🤔

Example: $[(a \cup b)^*](p \wedge q) \rightarrow [b^*](q \vee r)$ is valid in PDL.

Interpolant: $[b^*]q$

The Question

Does PDL have interpolation? 🤔

Example: $[(a \cup b)^*](p \wedge q) \rightarrow [b^*](q \vee r)$ is valid in PDL.

Interpolant: $[b^*]q$

But how do we *always* find these *systematically*?

History: Does PDL have Interpolation?

- 1981 Daniel Leivant: Yes! (using sequent calculus)

History: Does PDL have Interpolation?

- 1981 Daniel Leivant: Yes! (using sequent calculus)
- 1988 Manfred Borzechowski: Ja! (mit Tableau-Kalkül)

History: Does PDL have Interpolation?

- 1981 Daniel Leivant: Yes! (using sequent calculus)
- 1988 Manfred Borzechowski: Ja! (mit Tableau-Kalkül)
- 1999 Marcus Kracht: cannot verify your arguments

History: Does PDL have Interpolation?

- 1981 Daniel Leivant: Yes! (using sequent calculus)
- 1988 Manfred Borzechowski: Ja! (mit Tableau-Kalkül)
- 1999 Marcus Kracht: cannot verify your arguments
- 2002 Tomasz Kowalski: Yes! (using super-amalgamation)

History: Does PDL have Interpolation?

- 1981 Daniel Leivant: Yes! (using sequent calculus)
- 1988 Manfred Borzechowski: Ja! (mit Tableau-Kalkül)
- 1999 Marcus Kracht: cannot verify your arguments
- 2002 Tomasz Kowalski: Yes! (using super-amalgamation)
- 2004 Tomasz Kowalski: Never mind, let me retract that.

History: Does PDL have Interpolation?

- 1981 Daniel Leivant: Yes! (using sequent calculus)
- 1988 Manfred Borzechowski: Ja! (mit Tableau-Kalkül)
- 1999 Marcus Kracht: cannot verify your arguments
- 2002 Tomasz Kowalski: Yes! (using super-amalgamation)
- 2004 Tomasz Kowalski: Never mind, let me retract that.



MG's personal PDL History

- 2013 Yde Venema in the *Model Theory* course:
  By the way, for PDL this is an open problem.
- 2014 I find out that Kracht is wrong about Leivant being wrong but Yde Venema and I find other gaps in the proof of Leivant none of us manages to fill.

MG's personal PDL History

- 2013 Yde Venema in the *Model Theory* course:

  By the way, for PDL this is an open problem.

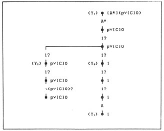
- 2014 I find out that Kracht is wrong about Leivant being wrong but Yde Venema and I find other gaps in the proof of Leivant none of us manages to fill.
- 2013/14 I also contact Borzechowski and get his thesis.

MG's personal PDL History

- 2013 Yde Venema in the *Model Theory* course:

  By the way, for PDL this is an open problem.

- 2014 I find out that Kracht is wrong about Leivant being wrong but Yde Venema and I find other gaps in the proof of Leivant none of us manages to fill.
- 2013/14 I also contact Borzechowski and get his thesis.



Ergebnis der Konstruktion ist also, daß $I_0 = [A^*](p \vee [C]0)$ ein Interpoland für die Formeln $[(A;A)^*](p \wedge [A;(B \cup C)]0)$ und $[A^*](p \vee [C]q)$ ist. Auf Grund der vorgenommenen Überlegungen, einen nicht unnötig großen Interpolanden zu konstruieren, ist I_0 sogar frei von Tests. Dieses Ergebnis ist jedoch nicht immer erreichbar.

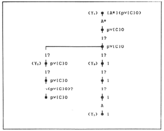
- 2020 I translate the German Diplomarbeit to English.

MG's personal PDL History

- 2013 Yde Venema in the *Model Theory* course:

  By the way, for PDL this is an open problem.

- 2014 I find out that Kracht is wrong about Leivant being wrong but Yde Venema and I find other gaps in the proof of Leivant none of us manages to fill.
- 2013/14 I also contact Borzechowski and get his thesis.



Ergebnis der Konstruktion ist also, daß I_0=[A*](p∨[C]0) ein Interpoland für die Formeln [(A;A)*](p∧[A;(B∪C)]0) und [A*](p∨[C]q) ist. Auf Grund der vorgenommenen Überlegungen, einen nicht unnötig großen Interpolanden zu konstruieren, ist I_0 sogar frei von Tests. Dieses Ergebnis ist jedoch nicht immer erreichbar.

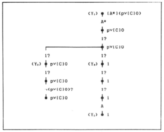
- 2020 I translate the German Diplomarbeit to English.
- 2022 to 2025: reading group with Helle Hvid Hansen, Revantha Ramanayake, Valentina Trucco Dalmas and Yde Venema (earlier also Johannes Marti, Jan Rooduijn and Guillermo Menéndez Turata)

MG's personal PDL History

- 2013 Yde Venema in the *Model Theory* course:

  By the way, for PDL this is an open problem.

- 2014 I find out that Kracht is wrong about Leivant being wrong but Yde Venema and I find other gaps in the proof of Leivant none of us manages to fill.
- 2013/14 I also contact Borzechowski and get his thesis.



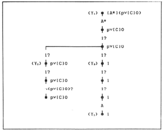
Ergebnis der Konstruktion ist also, daß I_0=[A*](p∨[C]0) ein Interpoland für die Formeln [(A;A)*](p∧[A;(B∪C)]0) und [A*](p∨[C]q) ist. Auf Grund der vorgenommenen Überlegungen, einen nicht unnötig großen Interpolanden zu konstruieren, ist I_0 sogar frei von Tests. Dieses Ergebnis ist jedoch nicht immer erreichbar.

- 2020 I translate the German Diplomarbeit to English.
- 2022 to 2025: reading group with Helle Hvid Hansen, Revantha Ramanayake, Valentina Trucco Dalmas and Yde Venema (earlier also Johannes Marti, Jan Rooduijn and Guillermo Menéndez Turata)
- 2025 April: we finish and submit our revised proof

  https://arxiv.org/abs/2503.13276

MG's personal PDL History

- 2013 Yde Venema in the *Model Theory* course:
  By the way, for PDL this is an open problem.
- 2014 I find out that Kracht is wrong about Leivant being wrong but Yde Venema and I find other gaps in the proof of Leivant none of us manages to fill.
- 2013/14 I also contact Borzechowski and get his thesis.



Ergebnis der Konstruktion ist also, daß $I_0=[A^*](p\vee[C]0)$ ein Interpoland für die Formeln $[(A;A)^*](p\wedge[A;(B\cup C)]0)$ und $[A^*](p\vee[C]q)$ ist. Auf Grund der vorgenommenen Überlegungen, einen nicht unnötig großen Interpolanden zu konstruieren, ist I_0 sogar frei von Tests. Dieses Ergebnis ist jedoch nicht immer erreichbar.

- 2020 I translate the German Diplomarbeit to English.
- 2022 to 2025: reading group with Helle Hvid Hansen, Revantha Ramanayake, Valentina Trucco Dalmas and Yde Venema (earlier also Johannes Marti, Jan Rooduijn and Guillermo Menéndez Turata)
- 2025 April: we finish and submit our revised proof
  🎉 `https://arxiv.org/abs/2503.13276`

All done?

History of the Formalization Project

- 2021 I start to learn Lean - it's a better Haskell!
- 2022 AiML: Interpolation for Basic Modal Logic in Lean 3
  `https://github.com/m4lvin/tablean`

History of the Formalization Project

- 2021 I start to learn Lean - it's a better Haskell!
- 2022 AiML: Interpolation for Basic Modal Logic in Lean 3
  `https://github.com/m4lvin/tablean`
- 2023 switch from Lean 3 to Lean 4 (mathport is amazing!)
- January 2024: MSc Logic project to get help from students
  Amos Nicodemus, Djanira dos Santos Gomes, Wietse Bosman, Haitian Wang, Xiaoshuang Yang, Jeremy Sorkin
- January 2025 and later: more MSc Logic students
  Noam Cohen, Eshel Yaron, Madeleine Gignoux

Overview

Interpolation via Tableaux

The procedure Borzechowski uses is known as Maehara's Method:

1. Define a sound and complete tableaux system.
2. For a valid implication, build a tableau (top-down).
3. Partition each node in the tableaux (top-down).
4. Define interpolants for all closed nodes / leaves.
5. Bottom-up define interpolants by combining those from child nodes.

Interpolation via Tableaux

The procedure Borzechowski uses is known as Maehara's Method:

1. Define a sound and complete tableaux system.
2. For a valid implication, build a tableau (top-down).
3. Partition each node in the tableaux (top-down).
4. Define interpolants for all closed nodes / leaves.
5. Bottom-up define interpolants by combining those from child nodes.

Demo: `https://w4eg.de/malvin/illc/tapdleau`

Why is this difficult?

The star.

Without star PDL is essentially multi-modal logic and easy to interpolate.

Why is this difficult?

The star.
Without star PDL is essentially multi-modal logic and easy to interpolate.

- The sequent calculus by Leivant has a rule with infinitely many premises:

$$*R: \quad \frac{\{f \vdash g, [\alpha]^n X\}_{n=0,1,\ldots}}{f \vdash g, [\alpha^*]X}$$

Why is this difficult?

The star.
Without star PDL is essentially multi-modal logic and easy to interpolate.

- The sequent calculus by Leivant has a rule with infinitely many premises:

$$*R: \quad \frac{\{f \vdash g, [\alpha]^n X\}_{n=0,1,\ldots}}{f \vdash g, [\alpha^*]X}$$

… and then argues that we can turn the system into a finitary one.

Why is this difficult?

The star.

Without star PDL is essentially multi-modal logic and easy to interpolate.

- The sequent calculus by Leivant has a rule with infinitely many premises:

$$*R: \quad \frac{\{f \vdash g, [\alpha]^n X\}_{n=0,1,\ldots}}{f \vdash g, [\alpha^*]X}$$

... and then argues that we can turn the system into a finitary one.

- The system by Borzechowski uses this rule for the star:

$$\begin{array}{c} [\alpha *]p \\ \Big| \; (*) \\ p; [\alpha][\alpha^n]p \end{array}$$

(where n is literally "n", not a number.)

Dealing with Repeats

With $*$ in a tableau we may arrive back at the same formulas.

$$[(Q?)^*]p$$

$$(*)$$

$$p; [Q?][(Q?)^*]p$$

$$(?) \qquad\qquad (?)$$

$$p; \neg Q \qquad\qquad p; [(Q?)^*]p$$

This is an example of a local repeat.

Dealing with Local Repeats: Local Unfolding

Idea: given a program α, compute all possible ways to execute it.

Example

$$\Phi_{\Diamond}((p? \cup a); b^*, \psi) = \{\{\neg[a][b^*]\psi\}, \{p, \neg\psi\}, \{p, \neg[b][b^*]\psi\}\}$$

Definition

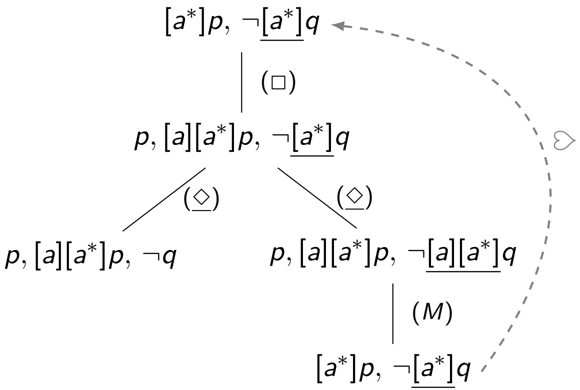
We define H to define one rule that "maximally takes apart" α in $\neg[\alpha]\phi$

$$\begin{array}{lcl}
H_a & := & \big\{(\emptyset, a)\big\} \\
H_{\tau?} & := & \big\{(\{\tau\}, \varepsilon)\big\} \\
H_{\alpha\cup\beta} & := & H_\alpha \cup H_\beta \\
H_{\alpha;\beta} & := & \big\{(X, \vec{\delta}\beta) \mid (X, \vec{\delta}) \in H_\alpha, \delta \neq \varepsilon\big\} \\
 & & \cup\big\{(X \cup Y, \vec{\delta}) \mid (X, \varepsilon) \in H_\alpha, (Y, \vec{\delta}) \in H_\beta\big\} \\
H_{\alpha^*} & := & \big\{(\emptyset, \varepsilon)\big\} \cup \big\{(X, \vec{\delta}\alpha^*) \mid (X, \vec{\delta}) \in H_\alpha, \vec{\delta} \neq \varepsilon\big\}
\end{array}$$

Here the $\delta \neq \epsilon$ ensures we do not create local repeats.

Global Repeats

The star can also lead to repeats like this:



To deal with this we use a *loading* (aka focus) mechanism and accept loaded-path repeats as closed end nodes of the tableau.
(See examples in the Demo.)

Main Ideas

- Soundness
  - All local rules (including unfolding) are sound and complete.
  - Lemma: loaded diamonds true in model can be imitated in tableau.
- Completeness
  - Define "model graphs" similar to canonical models;
  - Define a determined two-player game: prover vs builder;
  - Show that from winning strategy we can build
    - either a model graph
    - or a closed tableau.

Main Ideas

- Soundness
  - All local rules (including unfolding) are sound and complete.
  - Lemma: loaded diamonds true in model can be imitated in tableau.
- Completeness
  - Define "model graphs" similar to canonical models;
  - Define a determined two-player game: prover vs builder;
  - Show that from winning strategy we can build
    - either a model graph
    - or a closed tableau.
- Interpolation?
  - Maehara's method!
  - But what about the loaded-path repeats?

How to interpolate loops? Pseudo-interpolants with extra stuff!

Problem:

- Repeats are leaves, but they do not have interpolants yet!

How to interpolate loops? Pseudo-interpolants with extra stuff!

Problem:

- Repeats are leaves, but they do not have interpolants yet!

Two big insights by Borzechowski:

- We actually do not care about finding an interpolant for all nodes in that sub-tableau. Only the root.
- We do not have to use the given tableau to find the interpolants. Instead, define another tree, the pseudo-tableau, and run Maehara's method on that!

How to interpolate loops? Pseudo-interpolants with extra stuff!

Problem:

- Repeats are leaves, but they do not have interpolants yet!

Two big insights by Borzechowski:

- We actually do not care about finding an interpolant for all nodes in that sub-tableau. Only the root.
- We do not have to use the given tableau to find the interpolants. Instead, define another tree, the pseudo-tableau, and run Maehara's method on that!

Alternative explanation:

- Pseudo-interpolants: only together with extra stuff fulfil the conditions for being an interpolant.
- In the root node there is no extra stuff, so there we still get a standard interpolant.

How to interpolate loops? Pseudo-interpolants with extra stuff!

Problem:

- Repeats are leaves, but they do not have interpolants yet!

Two big insights by Borzechowski:

- We actually do not care about finding an interpolant for all nodes in that sub-tableau. Only the root.
- We do not have to use the given tableau to find the interpolants. Instead, define another tree, the pseudo-tableau, and run Maehara's method on that!

Alternative explanation:

- Pseudo-interpolants: only together with extra stuff fulfil the conditions for being an interpolant.
- In the root node there is no extra stuff, so there we still get a standard interpolant.

Yet alternative explanation: change the *system of equations*.

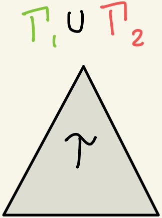
Dealing with Repeats: More Details

Following slides are by Valentina Trucco Dalmas, thank you!

Towards Maehara

$\Gamma_1 \cup \Gamma_2$
unsatisfiable

$\rightsquigarrow$

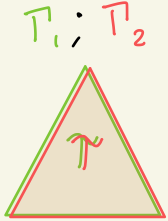


$\rightsquigarrow$ θ

Towards Maehara

$\Gamma_1 \cup \Gamma_2$

unsatisfiable $\rightsquigarrow$



$\rightsquigarrow$ θ

A split tableau system for PDL

$\Gamma_1 ; \Gamma_2$ Split Sequents

$$\frac{\varphi \wedge \psi, \Gamma_1 ; \Gamma_2}{\varphi, \psi, \Gamma_1 ; \Gamma_2} (\wedge)_\ell \qquad \frac{\Gamma_1 ; \Gamma_2, \varphi \wedge \psi}{\Gamma_1 ; \Gamma_2, \varphi, \psi} (\wedge)_r$$

$$\frac{\varphi \vee \psi, \Gamma_1 ; \Gamma_2}{\varphi, \Gamma_1 ; \Gamma_2 \quad | \quad \psi, \Gamma_1 ; \Gamma_2} (\vee)_\ell \qquad \frac{\Gamma_1 ; \Gamma_2, \varphi \vee \psi}{\Gamma_1 ; \Gamma_2, \varphi \quad | \quad \Gamma_1 ; \Gamma_2, \psi} (\vee)_r$$

Theorem: There exists a closed tableau for $\Gamma_1 ; \Gamma_2$ iff
(sound & complete) $\Gamma_1 \cup \Gamma_2$ is unsatisfiable.

8

Maehara's Method

$$
\begin{tikzcd}[column sep=small]
 & p \vee (q \wedge r) \,;\, \neg p, \neg r \arrow[dl] \arrow[dr, "(\vee)_l"'] & \\
p \,;\, \neg p, \neg r & & q \wedge r \,;\, \neg p, \neg r \arrow[d, "(\wedge)_l"] \\
 & & q, r \,;\, \neg p, \neg r
\end{tikzcd}
$$

Maehara's Method

$$p \vee (q \wedge r) \,;\, \neg p, \neg r : p \vee r$$

$$(\vee)_\ell$$

$$p \,;\, \neg p, \neg r : p \qquad\qquad q \wedge r \,;\, \neg p, \neg r : r$$

$$(\wedge)_\ell$$

$$q, r \,;\, \neg p, \neg r : r$$

Recall:

- $T_1 \models \theta$ and $T_2 \models \neg \theta$
- $\mathrm{Voc}(\theta) \subseteq \mathrm{Voc}(T_1) \cap \mathrm{Voc}(T_2)$

Maehara's Method

$$\neg p, \neg r \,;\, p \vee (q \wedge r) : \neg p \wedge \neg r$$

$(\vee)_r$

$$\neg p, \neg r \,;\, p : \neg p \qquad\qquad \neg p, \neg r \,;\, q \wedge r : \neg r$$

$(\wedge)_r$

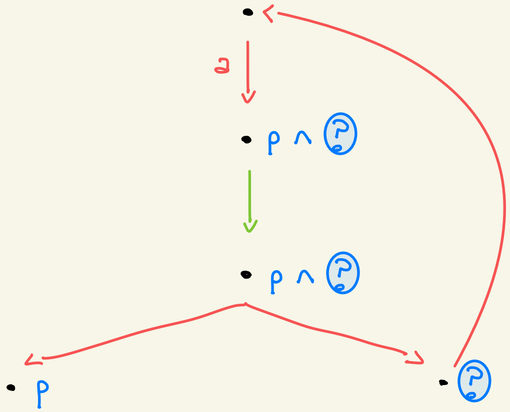
$$\neg p, \neg r \,;\, q, r : \neg r$$

Recall:

- $\Pi_1 \models \theta$ and $\Pi_2 \models \neg\theta$
- $Voc(\theta) \subseteq Voc(\Pi_1) \cap Voc(\Pi_2)$

Cyclic Maehara for PDL

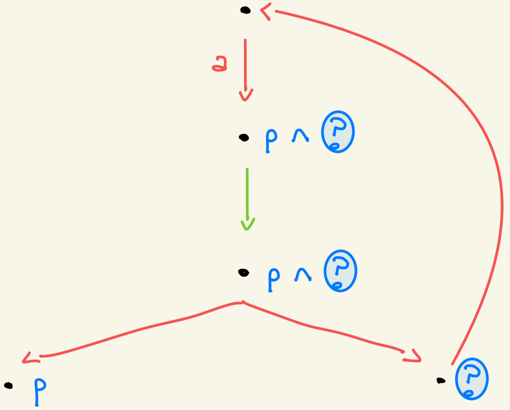
$\nu ?.[a](p \wedge ?)$



17

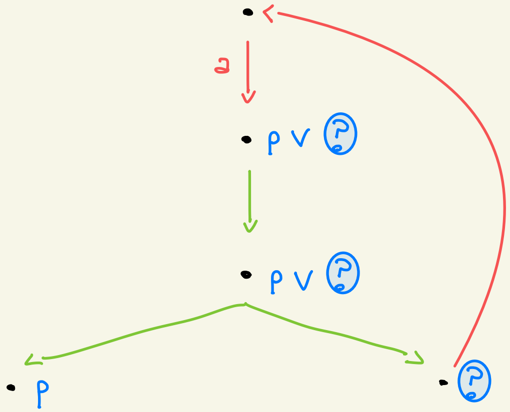
Cyclic Maehara for PDL

$$\nu ?.\,[a](p \wedge ?) \equiv [a^*][a]p$$



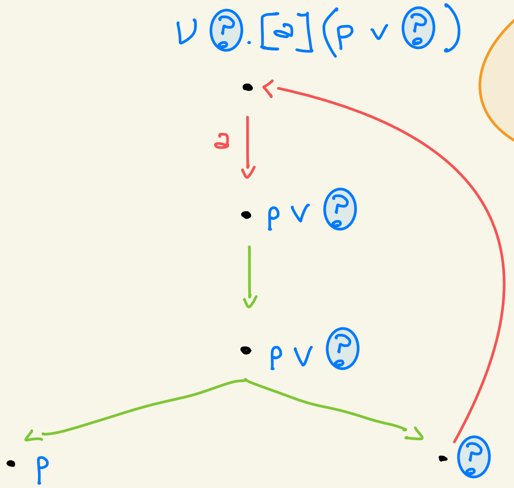
Cyclic Maehara for PDL

$\nu ?. [a](p \vee ?)$



18

Cyclic Maehara for PDL

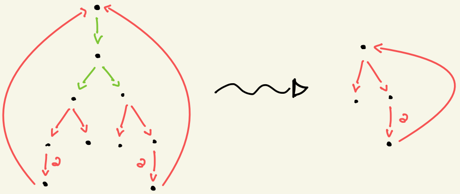


No solution in general in PDL

[CV14]

Quasitableaux

$$\begin{tikzcd}[column sep=tiny]
& & & {[a^*](p\vee\bot)\,;\,\neg[a^*]p} \arrow[d, "(*)_\ell"] & & & \\
& & & {p\vee\bot,[a][a^*](p\vee\bot)\,;\,\neg[a^*]p} \arrow[dll, "(\vee)_\ell"'] \arrow[drr] & & & \\
& {p,[a][a^*](p\vee\bot)\,;\,\neg[a^*]p} \arrow[dl, "(\neg*)_r"'] \arrow[dr] & & & & {\bot,[a][a^*](p\vee\bot)\,;\,\neg[a^*]p} \arrow[dl, "(\neg*)_r"'] \arrow[dr] & \\
{p,[a][a^*](p\vee\bot)\,;\,\neg[a][a^*]p} \arrow[d, "(M)_r"] & & {p,[a][a^*](p\vee\bot)\,;\,\neg p} & & {\bot,[a][a^*](p\vee\bot)\,;\,\neg p} & & {\bot,[a][a^*](p\vee\bot)\,;\,\neg[a][a^*]p} \arrow[d, "(M)_r"] \\
{[a^*](p\vee\bot)\,;\,\neg[a^*]p} \arrow[uuuurrr, bend left] & & & & & & {[a^*](p\vee\bot)\,;\,\neg[a^*]p} \arrow[uuuulll, bend right]
\end{tikzcd}$$



Quasitableaux

$$\Delta_x = \neg[a^*]p$$

$$[a^*](p\vee\bot) ; \neg[a^*]p$$

$$\downarrow (*)_\ell$$

$$p\vee\bot, [a][a^*](p\vee\bot) ; \neg[a^*]p$$

$$(\vee)_\ell$$

$$p, [a][a^*](p\vee\bot) ; \neg[a^*]p \qquad \bot, [a][a^*](p\vee\bot) ; \neg[a^*]p$$

$$(\neg *)_r \qquad\qquad (\neg *)_r$$

$$p, [a][a^*](p\vee\bot) ; \neg[a][a^*]p \qquad p, [a][a^*](p\vee\bot) ; \neg p \qquad \bot, [a][a^*](p\vee\bot) ; \neg p \qquad \bot, [a][a^*](p\vee\bot) ; \neg[a][a^*]p$$

$$\downarrow (M)_r \qquad p \qquad \bot \qquad \downarrow (M)_r$$

$$[a^*](p\vee\bot) ; \neg[a^*]p \qquad\qquad [a^*](p\vee\bot) ; \neg[a^*]p$$

$$\boxed{\begin{array}{l} \big[[a^*](p\vee\bot)\big] \vee \\ \big[(p\vee\bot)\wedge[a][a^*](p\vee\bot)\big] \vee \\ \big[p\wedge[a][a^*](p\vee\bot)\big] \vee \\ \big[[a^*](p\vee\bot)\big] \vee \\ \big[\bot\wedge[a][a^*](p\vee\bot)\big] \vee \\ \big[[a^*](p\vee\bot)\big] \end{array}} = \rho_x ; \neg[a^*]p$$

26

Quasitableaux

$[a^*](p \vee \bot)$

$$[a^*](p\vee\bot)\,;\,\neg[a^*]p$$

$$\Big\downarrow (*)_\ell$$

$$p\vee\bot,\,[a][a^*](p\vee\bot)\,;\,\neg[a^*]p$$

$$(\vee)_\ell$$

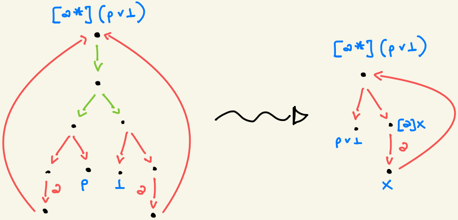
$$p,\,[a][a^*](p\vee\bot)\,;\,\neg[a^*]p \qquad\qquad \bot,\,[a][a^*](p\vee\bot)\,;\,\neg[a^*]p$$

$$(\neg *)_r \qquad\qquad\qquad\qquad (\neg *)_r$$

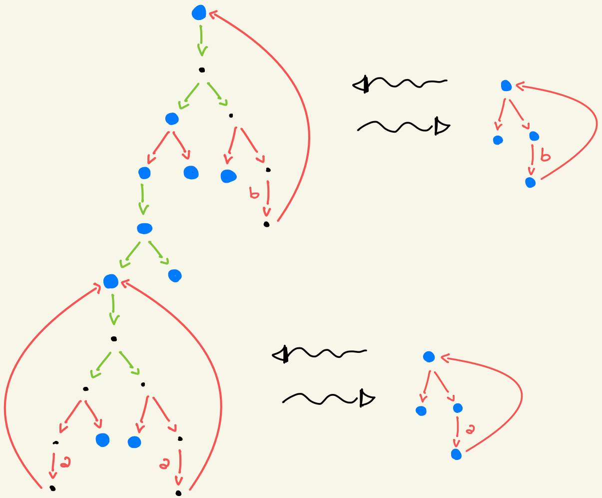
$$p,\,[a][a^*](p\vee\bot)\,;\,\neg[a][a^*]p \quad p,\,[a][a^*](p\vee\bot)\,;\,\neg p \quad \bot,\,[a][a^*](p\vee\bot)\,;\,\neg p \quad \bot,\,[a][a^*](p\vee\bot)\,;\,\neg[a][a^*]p$$

$$\Big\downarrow (M)_r \qquad\qquad\qquad\qquad\qquad\qquad \Big\downarrow (M)_r$$

$$[a^*](p\vee\bot)\,;\,\neg[a^*]p \qquad\qquad\qquad\qquad [a^*](p\vee\bot)\,;\,\neg[a^*]p$$



Putting it all together



More slides by Valentina Trucco Dalmas available at
`https://events.illc.uva.nl/llama/`

Overview

Lean: What

Lean is both

- an interactive theorem prover aka proof assistant
- a functional programming language

Similar systems: Rocq, Isabelle/HOL, agda, etc.

Lean: What

Lean is both

- an interactive theorem prover aka proof assistant
- a functional programming language

Similar systems: Rocq, Isabelle/HOL, agda, etc.

The key idea underlying formalized proofs: Propositions as Types
Propositions $\leftrightarrow$ Types
Proofs $\leftrightarrow$ Terms

Lean: What

Lean is both

- an interactive theorem prover aka proof assistant
- a functional programming language

Similar systems: Rocq, Isabelle/HOL, agda, etc.

The key idea underlying formalized proofs: Propositions as Types
Propositions $\leftrightarrow$ Types
Proofs $\leftrightarrow$ Terms

Example:

```
theorem exists_example {α : Type}  {F G : α → Prop} :
    ((∃ x, F x) ∧ (∀ x, F x → G x)) → ∃ x, G x :=
  fun ⟨⟨x,Fx⟩, F_sub_G⟩ => ⟨x, F_sub_G x Fx⟩
```

The last line is the term (= proof).
Checking that this function has this type is the same as checking the proof.

Lean: Why?

- Absolute certainty.
  (Especially when three times before a proof has been claimed ;-)

Lean: Why?

- Absolute certainty.
  (Especially when three times before a proof has been claimed ;-)
- (Really) understanding the proof.

Lean: Why?

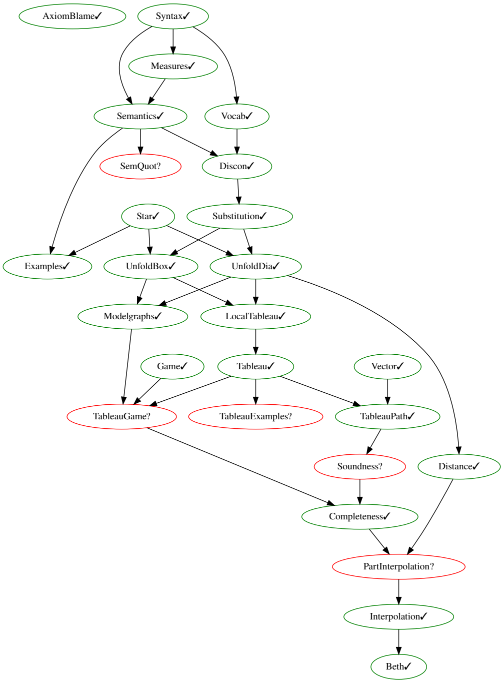
- Absolute certainty.
  (Especially when three times before a proof has been claimed ;-)
- (Really) understanding the proof.
- Bothering your co-authors with annoying details.

PDL in Lean: Goal

```
def Interpolant (φ : Formula) (ψ : Formula) (θ : Formula) :=
    θ.voc ⊆ φ.voc ∩ ψ.voc ∧ tautology (φ ↣ θ) ∧ tautology (θ ↣ ψ)

theorem interpolation {φ ψ : Formula} :
    tautology (φ ↣ ψ) → ∃ θ : Formula, Interpolant φ ψ θ := by
  [...]
```

PDL in Lean: Overview



Overview

Selected Issues

1. Mutual Induction / Recursion
2. Local Tableaux are finite (even with local unfolding)
3. How to represent proof trees with repeats?

(There are of course more — we are not even doing interpolation yet.)

Mutual induction: Syntax of formulas and programs

PDL really likes/needs mutual double recursion:

- programs in formulas
- formulas in programs

```
mutual
  inductive Formula : Type
    | bottom : Formula
    | atom_prop : Nat → Formula
    | neg : Formula → Formula
    | and : Formula → Formula → Formula
    | box : Program → Formula → Formula
  deriving Repr,DecidableEq
  inductive Program : Type
    | atom_prog : Nat → Program
    | sequence : Program → Program → Program
    | union : Program → Program → Program
    | star : Program → Program
    | test : Formula → Program
  deriving Repr,DecidableEq
end
```

Mutual induction: Semantics

```
mutual
  @[simp]
  def evaluate {W : Type} : KripkeModel W → W → Formula → Prop
    | _, _, ⊥ => False
    | M, w, ·c => M.val w c
    | M, w, ~φ => Not (evaluate M w φ)
    | M, w, φ⋀ψ => evaluate M w φ ∧ evaluate M w ψ
    | M, w, ⌈α⌉ φ => ∀ v : W, relate M α w v → evaluate M v φ
  @[simp]
  def relate {W : Type} : KripkeModel W → Program → W → W → Prop
    | M, ·c, w, v => M.Rel c w v
    | M, α;'β, w, v => ∃ y, relate M α w y ∧ relate M β y v
    | M, α∪β, w, v => relate M α w v ∨ relate M β w v
    | M, ∗α, w, v => Relation.ReflTransGen (relate M α) w v
    | M, ?'φ, w, v => w = v ∧ evaluate M w φ
end
```

Issue 2: Proofs about Mutual Induction

Unfortunately the `induction` tactic in Lean does not work for mutually inductive types. 😢

Issue 2: Proofs about Mutual Induction

Unfortunately the `induction` tactic in Lean does not work for mutually inductive types. 😢

This was my occasion to learn:

$$\text{Induction} = \text{Recursion}$$

Induction proofs are the same as recursive functions!
(Recall that theorems are functions.)

Issue 2: Proofs about Mutual Induction

Unfortunately the `induction` tactic in Lean does not work for mutually inductive types. 😢

This was my occasion to learn:

$$\text{Induction} = \text{Recursion}$$

Induction proofs are the same as recursive functions!
(Recall that theorems are functions.)

So, instead of using `induction` we can recursively call the theorem to obtain our induction hypotheses. 🙂

Issue 2: Termination Issues

The local unfolding rules for boxes and diamonds may generate an arbitrary number of formulas 🤔

Issue 2: Termination Issues

The local unfolding rules for boxes and diamonds may generate an arbitrary number of formulas 🤔

Usual approaches to show that tableaux are finite / terminate now fail:

- the sum of the length of formulas goes up ...
- while the maximum of the length of formulas may stay stithe same.

Issue 2: Termination Issues

The local unfolding rules for boxes and diamonds may generate an arbitrary number of formulas 🤔

Usual approaches to show that tableaux are finite / terminate now fail:

- the sum of the length of formulas goes up ...
- while the maximum of the length of formulas may stay stithe same.

- Solution in paper: subformula property via Fischer-Ladner closure.
- But in Lean we do not (yet) have a computable FL-closure 🤔

Issue 2 solution: The DM-ordering (work by Haitian Wang)

Solution: the Dershowitz-Manna Ordering on Multisets.

```
def IsDershowitzMannaLT [Preorder α] (M N : Multiset α) : Prop :=
  ∃ X Y Z,
      Z ≠ ∅
    ∧ M = X + Y
    ∧ N = X + Z
    ∧ ∀ y ∈ Y, ∃ z ∈ Z, y < z
```

Issue 2 solution: The DM-ordering (work by Haitian Wang)

Solution: the Dershowitz-Manna Ordering on Multisets.

```
def IsDershowitzMannaLT [Preorder α] (M N : Multiset α) : Prop :=
  ∃ X Y Z,
      Z ≠ ∅
    ∧ M = X + Y
    ∧ N = X + Z
    ∧ ∀ y ∈ Y, ∃ z ∈ Z, y < z

theorem wellFounded_isDershowitzMannaLT [WellFoundedLT α] :
  WellFounded
    (IsDershowitzMannaLT : Multiset α → Multiset α → Prop)
```

This was not in Mathlib, but there existed a ~~Coq~~ Rocq formalisation that ~~we~~ Haitian used as inspiration.

Side note: getting DM into mathlib

After finishing the proof of the DM Ordering Theorem in Lean, it took still quite some effort to make the code stable and maintainable, so that it was accepted into mathlib:
`https://github.com/leanprover-community/mathlib4/pull/14411`

Lesson: There is *formalizing in Lean* and there is *formalizing for mathlib*.

Issue 3: cyclic proofs

The inductive type `Tableau` should basically represent trees...

... but: loaded-path repeats need to see what is above them 🤔

Issue 3: cyclic proofs

The inductive type `Tableau` should basically represent trees...

... but: loaded-path repeats need to see what is above them 🤔

Solution: Dependent types 🙂

```
inductive Tableau : History → Sequent → Type
  | loc [..]
  | pdl (r : PdlRule X Y)
        (next : Tableau (X :: Hist) Y) : Tableau Hist X
  | lrep (lpr : X ∈ Hist ∧ ...) : Tableau Hist X
```

(heavily simplified code)

Overview

Summary

PDL has Craig Interpolation.

- The oldest proof has problems and gaps.
- The newest already was officially revoked.
- The middle one apparently nobody read, but did have the right ideas!

Summary

PDL has Craig Interpolation.

- The oldest proof has problems and gaps.
- The newest already was officially revoked.
- The middle one apparently nobody read, but did have the right ideas!

Lessons:

- Whether something is an open question can be an open question.
- German to English is easy, even without understanding it.
- English to Haskell is tricky, English to Lean is hard.
- Formalizing is ~~hard~~ fun, formalizing for Mathlib is harder.
- If you want to help, please contact me! 🙂

Summary

PDL has Craig Interpolation.

- The oldest proof has problems and gaps.
- The newest already was officially revoked.
- The middle one apparently nobody read, but did have the right ideas!

Lessons:

- Whether something is an open question can be an open question.
- German to English is easy, even without understanding it.
- English to Haskell is tricky, English to Lean is hard.
- Formalizing is ~~hard~~ fun, formalizing for Mathlib is harder.
- If you want to help, please contact me! 🙂

Dreaming bigger:

- ~~Why is there no Modal Logic in Mathlib?~~
  Why is there no Mathlib for Modal Logic?

Main References

[1] Manfred Borzechowski, Malvin Gattinger, Helle Hvid Hansen, Revantha Ramanayake, Valentina Trucco Dalmas, Yde Venema: Propositional Dynamic Logic has Craig Interpolation: a tableau-based proof. Preprint submitted to LMCS, March 2025. `https://arxiv.org/abs/2503.13276`

[2] Malvin Gattinger, Amos Nicodemus, Djanira dos Santos Gomes, Noam Cohen, Wietse Bosman, Madeleine Gignoux, Haitian Wang, Eshel Yaron, Xiaoshuang Yang, Jeremy Sorkin: Tableaux for Propositional Dynamic Logic in Lean 4. Work in progress since October 2023. `https://github.com/m4lvin/lean4-pdl`

[3] Haitian Wang: Pull request 14411 for mathlib: add Dershowitz-Manna Ordering and Theorem. Merged into mathlib in January 2025. `https://github.com/leanprover-community/mathlib4/pull/14411`

Other related code:

- Basic Modal Logic in Lean 3: `https://github.com/m4lvin/tablean`
- PDL Prover in Haskell: `https://w4eg.de/malvin/illc/tapdleau`

More References and Links

- Daniel Leivant: *Proof theoretic methodology for propositional dynamic logic*. LNCS, 1981.
- Manfred Borzechowski (1988): *Tableau–Kalkül für PDL und Interpolation*. Diplomarbeit, FU Berlin, 1988.
  - English Translation (2020): `https://malv.in/2020/borzechowski-pdl`
- Marcus Kracht: *Tools and Techniques in Modal Logic*, 1999.
- Tomasz Kowalski: *PDL has interpolation*. JSL, 2002. Revoked in 2004.
- D'Agostino & Hollenberg: *Logical questions concerning the μ-Calculus: Interpolation, Lyndon and Łoś-Tarski*. JSL, 2000.
- Facundo Carreira, Yde Venema: *PDL inside the μ calculus: A syntactic and an automata-theoretic characterization*. AiML, 2014.
- Johan van Benthem: *The many faces of Interpolation*. Synthese, 2008.

Overview

1999 Kracht about Leivant and Borzechowski

Marcus Kracht writes the book *Tools and Techniques in Modal Logic* and says in Section 10.6 "The Unanswered Section" (p. 493):

> *"[T]he problem of interpolation for PDL is one of the major open problems in this area. Twice a solution has been announced [. . .], but in neither case was it possible to verify the argument.*

1999 Kracht about Leivant and Borzechowski

Marcus Kracht writes the book *Tools and Techniques in Modal Logic* and says in Section 10.6 "The Unanswered Section" (p. 493):

> *"[T]he problem of interpolation for PDL is one of the major open problems in this area. Twice a solution has been announced [. . .], but in neither case was it possible to verify the argument. The argument of Leivant makes use of the fact that if* $\phi \vdash_{\mathbf{PDL}} \psi$ *then we can bound the size of a possible countermodel so that the star* α^* *only needs to search up to a depth d which depends on* ϕ *and* ψ*. [. . .] However, this is tantamount to the following. Abbreviate by* PDL^n *the strengthening of* PDL *by axioms of the form* $[a^*]p \leftrightarrow [a^{\leq n}]p$ *for all a. Then, by the finite model property of* PDL, PDL *is the intersection of the logics* PDL^n*. Unfortunately, it is not so that interpolation is preserved under intersection."*

Note: Kracht does not explain why Borzechowski should be wrong.

2020 Translation

<u>Definition.</u> Ein Tableau $\mathcal{T}$ heißt *geschlossen*, wenn alle normalen freien Endknoten von $\mathcal{T}$ geschlossen sind.

Im verbleibenden Teil von Abschnitt 1 zeigen wir, daß der hiermit definierte PDL-Kalkül vollständig ist; das heißt, daß jede Formelmenge $X \subseteq \mathcal{F}$ genau dann erfüllbar ist, wenn kein geschlossenes Tableau für X existiert.

Doch zunächst geben wir ein Beispiel eines geschlossenen Tableaus für die Menge $X=\{\neg[(A\cup p?)^*]q, [A^*]q\}$.

- $\neg[(A\cup p?)^*]q;$ $[A^*]q$
  - (M+) $\neg[(A\cup p?)^*]q^q;$ $[A^*]q$
    - (n) $\neg[(A\cup p?)^*]q^q;$ $q;[A][A^*]q$
      - (¬n) $\neg q;$ $q;[A][A^*]q$
      - (¬n) $\neg[A\cup p?][(A\cup p?)^{(n)}]q^q;$ $q;[A][A^*]q$
        - (¬∪) $\neg[A][(A\cup p?)^*]q^q;$ $q;[A][A^*]q$
          - (At) $\neg[(A\cup p?)^*]q^q;$ $[A^*]q$
        - (¬∪) $\neg[p?][(A\cup p?)^{(n)}]q^q;$ $q;[A][A^*]q$
          - (¬?) $p;\neg[(A\cup p?)^{(n)}]q^q;$ $q;[A][A^*]q$

Definition 16. *A tableau* $\mathcal{T}$ *is called* closed *when all normal free end nodes of* $\mathcal{T}$ *are closed.*

In the remainder of part 1 we show that the PDL-Calculus we defined in the previous sections is complete; that is, any set of formulas $X \subset \mathcal{F}$ is satisfiable if and only if there is no closed tableau for X.

However, before that we give an example of a closed tableau for the set $X = \{\neg[(A\cup p?)^*]q, [A^*]q\}$.

- $\neg[(A\cup p?)^*]q \;;\; [A^*]q$
  - $(M+)$ $\neg[(A\cup p?)^*]q^q \;;\; [A^*]q$
    - (n) $\neg[(A\cup p?)^*]q^q \;;\; q \;;\; [A][A^*]q$
      - $(\neg n)$ $\neg q \;;\; q \;;\; [A][A^*]q$
      - $(\neg n)$ $\neg[(A\cup p?)][(A\cup p?)^{(n)}]q^q \;;\; q \;;\; [A][A^*]q$
        - $(\neg\cup)$ $\neg[A][(A\cup p?)^*]q^q \;;\; q \;;\; [A][A^*]$
          - (At) $\neg[(A\cup p?)^*]q^q \;;\; [A^*]$
        - $(\neg\cup)$ $\neg[p?][(A\cup p?)^{(n)}]q \;;\; q \;;\; [A][A^*]$
          - $(\neg ?)$ $p \;;\; \neg[(A\cup p?)^{(n)}]q \;;\; q \;;\; [A][A^*]q^q$

2023 Haskell Prover

In 2023 I resurrected my 2016 attempt of a tableaux prover in Haskell, using the proof rules from Borzechowski (1988).
Now with a better understanding of the proof thanks to the reading group, I did manage to make a prover that seems to work for all examples.

See `https://w4eg.de/malvin/illc/tapdleau` for our system, and `https://w4eg.de/malvin/illc/tapdleau-borzechowski` for the version with n-formulas.

2023 January Meeting in Berlin

I managed to meet Manfred Borzechowski in person!