university of groningen / faculty of science and engineering

Implementing Maehara's Method for Star-Free Propositional Dynamic Logic.

Bachelor's Project Thesis

Francesca Perin, s2865300, f.perin@student.rug.nl
Supervisors: B.R.M. Gattinger

Abstract: Craig's interpolation theorem has been proven for different logics such as first-order logic, propositional and basic modal logic. However, it is still an open debate whether Propositional Dynamic Logic (PDL) has said property. This research does not aim to close such debate. Our goal is to implement an interpolation prover for propositional logic, basic modal logic, some of its extensions, up to star-free PDL. Our program randomly generates an implication and then the prover determines its validity by using sequent calculus. The prover then, given the valid sentence, also finds an interpolant using Maehara's partition interpolation method. The prover then outputs the sequent calculus proof and the computed interpolant. For propositional logic, all rules of inference and the computation of the interpolant are fully deterministic. For modal logic and star-free PDL, some of the rules are non-deterministic in which case the prover uses depth-first search to find one of the valid proofs. The prover is tested with 450.000 randomly generated formulas and also a specific set of formulas for all logics. For each randomly generated formula tested that was found valid a final interpolant which respected the definition was found. This was also the case for the specific set, only one formula was supposed to be non-valid but was found to be valid. We argue that this was due to a mistake as the proof for that formula is correct. The code for this project can be found in the GitHub repository: `https://github.com/FrancescaPerin/BScProject`

1 Introduction

Interpolation properties have long been of interest for logicians but more recently applications of interpolation have also been studied in formal verification (McMillan, 2018), computational complexity (Cook and Reckhow, 1979), and knowledge representation (Lutz and Wolter, 2011; ten Cate, Franconi, and Seylan, 2013), among others (Baŕcenas, Lavalle-Martiñez, Molero-Castillo, and Velaźquez-Mena). Craig interpolation (CI) states that given a logic Λ, as set of valid formulas, and given the language $L(\varphi)$ as the set of propositional atoms (and atomic programs for PDL) of formula φ, the logic Λ has Craig interpolation if and only if for any formula of the type $\varphi \to \psi \in \Lambda$ there exists at least one formula μ such that:

$$\varphi \to \mu \in \Lambda \text{ and } \mu \to \psi \in \Lambda \tag{1.1}$$

$$L(\mu) \subseteq L(\varphi) \cap L(\psi) \tag{1.2}$$

Formula μ is then called an interpolant for the implication $\varphi \to \psi$. One or more interpolants that respect the property might be found.

Craig interpolation was first proven for first-order logic by William Craig in 1957. Later it was shown that the property also holds for many other logics such as propositional logic, basic modal logic (D' Agostino, 2008), multi-modal logic (Madarász, 1995) and intuitionistic logic (D' Agostino, 2008).

With regard to Propositional Dynamic Logic (PDL) there have been several attempts to prove the CI property.

Daniel Leivant (1981) in his paper and Manfred Borzechowski (1988) in his thesis, aim for a proof-theoretic argument using respectively sequent calculus and tableaux (Leivant, 1981; Borzechowski, 1988). However both of their proofs have been criticized, in particular, Kracht argues that Leivant with his method only proves Craig interpolation for finitary variants of PDL (Kracht, 1999). Furthermore, Tomasz Kowalski (2002) tried to prove the theorem for PDL using duality results about free dynamic algebras, but a flaw was pointed out by Yde Venema, which lead to Kowalski retracting the paper (Kowalski, 2002, 2004).

PDL is a propositional variant of Dynamic Logic (DL) that does not use terms, predicates, and functions but only uses two syntactic categories: propositions (ϕ) and programs (α).

The syntactic definition of PDL is the following:

$$\phi ::= p \mid \neg\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \phi \rightarrow \phi \mid [\alpha]\phi \mid \top \mid \bot \tag{1.3}$$

$$\alpha ::= a \mid \alpha;\alpha \mid \alpha \cup \alpha \mid \phi? \mid \alpha^* \tag{1.4}$$

where p belongs to the set of propositional letters. Furthermore a is an atomic program and the remaining, such as $\alpha;\alpha$ are complex programs. We also define the $\langle\alpha\rangle$ operator in terms of necessity as follows $\langle\alpha\rangle\phi := \neg[\alpha]\neg\phi$.

We notice that PDL extends propositional logic. Furthermore, if we do not take in consideration the complex programs and only consider one modality $[a]$ for a single atomic program a, then PDL becomes basic modal logic. In other words, we can start from the definition of propositional logic, then we extend it by adding the operators for modality resulting in modal logic, then we extend it further by adding programs (α). To understand the concept of programs easily we consider the semantics of modal logic. In modal logic, we first define a frame as a pair $\langle W, R\rangle$ where W is a set of worlds, and R is a binary relation over W. This binary relation is also called accessibility relation. So given $w, w' \in W$, the notation wRw' is used to represent that the world w' is accessible from world w.

1.1 PDL semantics

The semantic definition of PDL is not used in this project, however, it is important to understand why it has been difficult to prove CI for PDL.

PDL semantics uses Labeled Transition Systems (LTS) which is a generalization of Kripke models. An LTS consists of worlds (or states) as mentioned for modal logic, but the accessibility relations (also called transitions) between worlds are labeled with the name of the atomic programs. A labelled transition a from world $w \in W$, to $w' \in W$, noted as wR_aw', or $(w, w') \in R_a$ indicates that starting from world w, we can execute program a to reach world $w' \in W$. The formula $[a]p$ therefore is true if, starting from a word $w \in W$, p is true in all worlds $w' \in W$ that are reachable with a labeled transition a. Formally:

$$M, w \models [a]\varphi \text{ iff for all } w', wR_aw' \Rightarrow M, W' \models \varphi \tag{1.5}$$

The complex programs can be easily understood at this stage by their intuitive meanings: the program $\alpha;\beta$ is a sequential composition, first, we perform transition α and then transition β, program $\alpha \cup \beta$ is a non-deterministic choice between performing the transition α or β. Tests $\phi?$ check if the formula ϕ is true in the current state (or world). Finally, α^* is repeating α any finite number of times.
The $\alpha*$ program is challenging when trying to prove the CI property due to being infinitary. To find an interpolant of an implication such as $\varphi \rightarrow \psi$ (using tableaux or sequent calculus) the full proof of its validity is needed. Leivant in his system introduces a $[\alpha*]$ introduction rule which needs infinitely many premises leading to a proof that is not finite (Leivant, 1981). This is hard to use in computation and is also difficult to find an interpolant because it is not clear how to combine the previous interpolants if there is an infinite number of them.

In this paper we do not consider the α^* program, thus treating only star-free PDL. We are also not going to take into account tests, because it has been proven that PDL without tests has the CI property if and only if PDL with test does. (Gattinger, 2014; Kracht, 1999, Theorem 10.6.2 p.495)

1.2 Sequent calculus and Maehara's method

The aim of this project is not to prove that PDL has the CI property but instead to create a theorem prover that shows the CI property for propositional logic, modal logic up to star-free PDL by implementing Maehara's method. The implemented program generates randomly a formula of the type $\varphi \rightarrow \psi$, which is then given to the prover which determines its validity and if it is valid generates an interpolant and checks whether it respects the definition.

Given the three different logics taken into consideration in this project and the definition of interpolants we need to find a method to compute interpolants. We use Maehara's partition interpolation method (Takeuti, 1975) in combination with sequent calculus. The latter is very similar and was chosen instead of analytic tableaux due to this project being heavily influenced by Leivant's paper, which as previously mentioned used sequent calculus.

Sequent calculus is a type of proof calculus, in which every line is a conditional tautology (also called sequent). The starting implication $\varphi \rightarrow \psi$ is transformed into the entailment $\varphi \vdash \psi$. This formula in the sequent calculus proof is placed at the bottom of the proof and the proof is read starting from the bottom moving upwards. Furthermore, on the left and right of the turnstile

symbol ($\vdash$) the sequent can contain more than one formula:

$$\varphi_1, \varphi_2, ..., \varphi_n \vdash \psi_1, \psi_2, ..., \psi_k \quad (1.6)$$

In sequents the φ_i are called the *antecedents* (or as called in this paper premises) and ψ_i are are called the *consequents* (or referred in this paper as conclusions). In sequent calculus notation the commas to the left of the turnstile can be interpreted as conjunctions ($\wedge$), and to the right as disjunctions ($\vee$). The sequent in 1.6 can thus be rewritten as:

$$\vdash \varphi_1 \wedge \varphi_2 \wedge ... \wedge \varphi_n \rightarrow \psi_1 \vee \psi_2 \vee ... \vee \psi_k \quad (1.7)$$

A sequent calculus proof is a reduction tree (removing connectives) where as mentioned we have at the bottom the starting sentence and at the top, there are (one or more) sequents with fully reduced formulas (also called leaves). If one or more of the formulas in the premises can also be found in the conclusions, then the leaf is called axiom. A sentence is valid only if all the leaves of the proof are axioms.

After deriving the proof for a valid sentence we then use Maheara's partition interpolation method to compute an interpolant. We will discuss the method in more detail in the next section, but as a general idea, the method involves finding the interpolant for each axiom in the proof and given these interpolants compute the interpolant for the next sequent. How the interpolants are combined or modified depends on the rules applied in the sequent calculus proof. A valid sentence can have more than one interpolant as there are many possible proofs. Given a proof of a sentence the interpolant found is always the same.

2 Methods

In this section, we are going to first rewrite the definition of interpolant and sequent (for sequent calculus) to follow Maheara's method. Secondly, we are going to explain the sequent calculus rules and the rules for computing interpolants and how they were derived according to the new definition. We are going to start from the rules for propositional logic, then the rules for modal logic and finally for star-free PDL.

2.1 Maehara's method

Maehara's method is called partition interpolation. To use this method we need to give our definition of sequent in which the premises and conclusions are partitioned:

$$f^-; f^+ \vdash X^+; X^- \quad (2.1)$$

A partition of a set f of formulas is a pair f^-, f^+ such that $f^- \cup f^+ = f$ and $f^- \cap f^+ = \emptyset$. This is denoted as $f^-; f^+$. The conclusions are also partitioned, thus $X^+ \cup X^- = X$ and $X^+ \cap X^- = \emptyset$.

We rewrite the definiton of interpolant (1.1,1.2) according to the partitions as given in the sequent definition in formula 2.1. We say that Θ is an interpolant of 2.1 iff:

$$f^- \vdash X^-, \Theta \text{ and } f^+, \Theta \vdash X^+ \quad (2.2)$$

$$L(\Theta) = L(f^-, X^-) \cap L(f^+, X^+) \quad (2.3)$$

When starting a proof with an entailment of the type $\varphi \vdash \psi$ the premise φ is placed into f^- and the conclusion ψ is placed into X^+, resulting in the following entailment $\varphi; \vdash \psi;$, where thus both premise and conclusion are on the left side of the semicolons. In this case f^+ and X^- are empty.

The definition of interpolant in 2.2 and 2.3 is related to the definition in 1.1 and 1.2. Any formula of the type $\varphi \rightarrow \psi$ can be rewritten in sequent calculus notation as $\varphi; \vdash \psi;$. Furthermore, if we prove that Θ is an interpolant for the sequent $\varphi; \vdash \psi;$ (thus respecting the definitions 2.2 and 2.3) it also holds that Θ is an interpolant for $\varphi \rightarrow \psi$ and respects the definitions in 1.1 and 1.2.

The partitions are necessary to compute the interpolants. As mentioned in Section 1, an axiom is a sequent in which there is at least one formula that is common between the premises and the conclusions. To find the interpolant for an axiom we look in which partition the formula is in the premises and the conclusions. If there is more than one such formula we select one arbitrarily. Assuming that $\mathbf{a}$ is the common formula, we then have four possibilities:

- $f^-, \mathbf{a}; f^+ \vdash X^+, \mathbf{a}; X^-$ then $\Theta \coloneqq \mathbf{a}$
- $f^-; f^+, \mathbf{a} \vdash X^+; X^-, \mathbf{a}$ then $\Theta \coloneqq \neg\mathbf{a}$
- $f^-, \mathbf{a}; f^+ \vdash X^+; X^-, \mathbf{a}$ then $\Theta \coloneqq \bot$
- $f^-; f^+, \mathbf{a} \vdash X^+, \mathbf{a}; X^-$ then $\Theta \coloneqq \top$

where Θ is the interpolant of the axiom. Starting from the interpolants of the axioms in the proof we can then compute the interpolant for the sequent. Every rule used in the sequent calculus proof has a one to one relation with an interpolant rule. The latter specifies how to combine or modify the previous interpolant(s) such that the new one respects the definition for the next sequent. When the root sequent of the proof is reached the final interpolant is obtained.

There are also a few cases where an axiom does not contain a common formula but is nevertheless an axiom. If we have a sentence of the type $\bot \vdash \psi_1, \psi_2, ..., \psi_k$ (where k is a positive integer), the sentence is always valid because $\bot \vdash \top$ is valid but also $\bot \vdash \bot$. The same is also valid for $\varphi_1, \varphi_2, ..., \varphi_k \vdash \top$. When we encounter such axioms, to find the interpolant we look in which partition $\top$ or $\bot$ are. We have the following cases:

- $f^-, \bot; f^+ \vdash X^+; X^-$ then $\Theta = \bot$
- $f^-, \bot; f^+ \vdash X^+; X^-$ then $\Theta = \bot$
- $f^-; f^+, \bot \vdash X^+; X^-$ then $\Theta = \top$
- $f^-; f^+, \bot \vdash X^+; X^-$ then $\Theta = \top$

- $f^-; f^+ \vdash X^+, \top; X^-$ then $\Theta = \top$
- $f^-; f^+ \vdash X^+, \top; X^-$ then $\Theta = \top$
- $f^-; f^+, \top \vdash X^+; X^-, \top$ then $\Theta = \bot$
- $f^-; f^+ \vdash X^+; X^-, \top$ then $\Theta = \bot$

This is the case only if there is no common formula between premises and conclusions. Here f^-, f^+, X^+ and X^- cold be empty or contain one or more formulas. For the case in which we have $\bot$ then a formula is chosen from the conclusions, otherwise for $\top$ the formula is chosen from the premises.

2.2 Rules for Propositional logic

After rewriting the definition of sequent, we need to derive the rules of inference for sequent calculus accordingly to this definition (2.1).

We start with the rules for propositional logic. In this paper, we only show the derivation of two rules for the implication, but the rules were derived for all the connectives $\neg, \wedge, \vee, \rightarrow$ and can be found in Appendix A. One can decide to only define the rules for the connectives $\neg$ and $\rightarrow$, and define the remaining connectives in terms of the ones for which the rules are defined. Leivant (1981) uses this approach in his sequent calculus system D.

For each connective, there are two inference rules, due to the rules changing when applied in the premises or the conclusions. Additionally, in our sequent definition both premises and conclusions are partitioned. For each connective we have four rules, depending on if it is applied in the premise or conclusion, but also depending on which side of the partition, i.e. of the semicolon, the connective is found. The rules in the proof are labeled with a letter L if applied in the premises or R if in the conclusions, the connective, and a '+' symbol if applied to the left of the partition or '-' if applied to the right side of the partition for both the premises and conclusions (this is not kinked to the symbol '-' and '+' in, for example $f^- or X^+$).

2.2.1 The L$\rightarrow_{_}$ rule

In this section, we are going to derive the sequent calculus rule and the interpolant rule for the sequent in 2.6. The same is going to be performed for sequent 2.5 in Section 2.2.2.

We start from Leivant's rule for the left implication:

$$\frac{f \vdash A, X \qquad f, B \vdash X}{f, A \rightarrow B \vdash X}\,(\text{L} \rightarrow) \tag{2.4}$$

Take our definition of partitioned sequent in 2.1. We can add formula $A \rightarrow B$ in the left partition in one case and in the other in the right, obtaining the sequents:

$$f^-, A \rightarrow B; f^+ \vdash X^+; X^- \tag{2.5}$$

$$f^-; f^+, A \rightarrow B \vdash X^+; X^- \tag{2.6}$$

We look at the rule in formula 2.4. In the rule, removing the implication symbol in $A \rightarrow B$ results in two new sequents, one which keeps formula A, which is moved from the premise to the conclusion, and the other which keeps formula B in the conclusions. This is true also for our rule of inference but we need to make sure that formulas A and B are in the correct partitions.

Now we choose that formula A moves to the left partition (in the conclusions) and formula B to the right partition (in the premises). When the formula is applied we obtain the following sequents:

$$\frac{f^-; f^+ \vdash X^+, A; X^- \qquad f^-; f^+, B \vdash X^+; X^-}{f^-; f^+, A \rightarrow B \vdash X^+; X^-}\,(L \rightarrow_{\_}) \tag{2.7}$$

Now we assume that we have interpolants Θ_1 for the left child and Θ_2 for the right child. Now we apply the definition of interpolant in 2.2 and we write the resulting sequents for the two children and the original root sequent in 2.6.

From the left child in rule 2.7 we get:

$$f^- \vdash X^-, \Theta_1 \tag{2.8}$$

$$f^+, \Theta_1 \vdash X^+, A \tag{2.9}$$

From the right child in rule 2.7 we get:

$$f^- \vdash X^-, \Theta_2 \tag{2.10}$$

$$f^+, B, \Theta_2 \vdash X^+ \tag{2.11}$$

For the root sequent in rule 2.7 we want to find a Θ such that:

$$f^- \vdash X^-, \Theta \tag{2.12}$$

$$f^+, A \to B, \Theta \vdash X^+ \tag{2.13}$$

If formula 2.7 is correct then it is the case that we can apply rule 2.4 to sequent 2.13 and obtain the sequents from 2.9 and 2.11 :

$$\frac{f^+, \Theta_1 \vdash X^+, A \qquad f^+, B, \Theta_2 \vdash X^+}{f^+, A \to B, \Theta \vdash X^+} (L \to) \tag{2.14}$$

We can define Θ as the conjunction of Θ_1 and Θ_2 ($\Theta = \Theta_1 \wedge \Theta_2$). Now we rewrite the definitions of 2.12 and 2.13 as following:

$$f^- \vdash X^-, \Theta_1 \wedge \Theta_2 \tag{2.15}$$

$$f^+, A \to B, \Theta_1 \wedge \Theta_2 \vdash X^+ \tag{2.16}$$

Now we can also rewrite formula 2.14 using the above definitions:

$$\frac{f^+, \Theta_1 \wedge \Theta_2 \vdash X^+, A \qquad f^+, B, \Theta_1 \wedge \Theta_2 \vdash X^+}{f^+, A \to B, \Theta_1 \wedge \Theta_2 \vdash X^+} (L \to_+, L\wedge_+) \tag{2.17}$$

Rule 2.17 is an instance of rule 2.4 applied to sequent in 2.16 which further proves that the inference rule is correct.

We just defined Θ as being $\Theta_1 \wedge \Theta_2$, but we can show that this is the case. Starting from axioms in 2.9 and in 2.11 we want to be able to construct a proof that reaches the sequent 2.16 as such:

$$\frac{\dfrac{\dfrac{f^+, \Theta_1 \vdash X^+, A}{f^+, \Theta_1, \Theta_2 \vdash X^+, A}\text{(WEAK L)} \qquad \dfrac{f^+, B, \Theta_2 \vdash X^+}{f^+, B, \Theta_1, \Theta_2 \vdash X^+}\text{(WEAK L)}}{f^+, (A \to B), \Theta_1, \Theta_2 \vdash X^+}(L \to)}{f^+, (A \to B), (\Theta_1 \wedge \Theta_2) \vdash X^+}(L\wedge) \tag{2.18}$$

Starting from the axioms we work backward, we apply the weakening rule (which is used to add one or more formulas to the premises or conclusions) so that in both sequents we have both interpolants Θ_1 and Θ_2. Then the left implication rule is applied to A and B to obtain $A \to B$ in the premises, finally, we need to find a rule which combines Θ_1 and Θ_2 with one (or more) connective. The rule is applied to combine the two interpolants and thus defines Θ in terms of Θ_1 and Θ_2. In the proof the only rule that can be used to combine the two interpolants is the conjunction rule, leading to Θ being defined as $\Theta_1 \wedge \Theta_2$.

Proof in 2.18 only uses one of the two sequents in the definition of interpolant (2.2). A proof is also required for the second sequent and can be used to double check that the definition of Θ in terms of Θ_1 and Θ_2 is correct. We start the proof from the two axioms which are the formulas in 2.8 and 2.10. We then check if Θ_1 and Θ_2 can be combined with the connective found in the other proof, in this case $\wedge$.

$$\frac{f^- \vdash X^-, \Theta_1 \qquad f^- \vdash X^-, \Theta_2}{f^- \vdash X^-, (\Theta_1 \wedge \Theta_2)} (R\wedge) \tag{2.19}$$

As shown in 2.19 it is the case, as found in the other proof, that the $L\wedge$ is used to join the two sequents in one formula, thus the definition is still valid. We also know that every time the rule $L \to_-$ (formula in 2.7) is applied in the sequent calculus proof, the interpolant of the next sequent is the conjunction of the interpolants found for the two children.

Now we showed that the inference rule is correct and that the definition of the interpolant Θ in 2.2 holds, however, it is also necessary to show that language condition in 2.3 also holds.

Regarding the language of the left sequent we know that:

$$L(\Theta_1) \subseteq L(f^-, X^-) \cap L(f^+, X^+, A) \tag{2.20}$$

for the language of the right sequent we know that:

$$L(\Theta_2) \subseteq L(f^-, X^-) \cap L(f^+, B, X^+) \tag{2.21}$$

for the language of the root sequent we know that :

$$L(\Theta) = L(f^-, X^-) \cap L(f^+, A \to B, X^+) \tag{2.22}$$

and for our definiton of Θ then:

$$L(\Theta) = L(\Theta_1 \wedge \Theta_2) \tag{2.23}$$

$$\begin{aligned} L(\Theta) &= L(\Theta_1) \cup L(\Theta_2) \\ &\subseteq (L(f^-, X^-) \cap L(f^+, X^+, A)) \\ &\quad \cup (L(f^-, X^-) \cap L(f^+, B, X^+)) \\ &= (L(f^+, X^+, A) \cup L(f^+, B, X^+)) \cap L(f^-, X^-) \\ &= L(f^+, X^+, A \to B) \cap L(f^-, X^-) \end{aligned} \tag{2.24}$$

Therefore the language condition also holds.

We fully derived the rule $L \to_-$ and showed how to derive the correspondent rule for computing the new interpolant.

2.2.2 The L$\to_+$ rule

The way of deriving the partition rules for sequent calculus always follows the same principles shown in the previous section 2.2.1. However, there are two rules in which a switch in the partition is required. These rules are the $L \to_+$ rule and the $L\neg_+$ rule (both applied to the premises and left side of the partition). In this section we are only going to show the derivation of the

rule L $\rightarrow_+$ and corresponding interpolant rule. This because it is a slightly more complex rule since it involves a binary operator. Because the process to derive the second rule is the same it will not be discussed in details. The full proof for the $L\neg_+$ rule can be found in Appendix A.

In the case of a switch in partition the notation of the sequent becomes the following:

$$f^+; f^- \vdash X^-; X^+ \tag{2.25}$$

We note that this notation is never used in the starting sequent of a proof as a switch in the partition can occur only when either the L$\rightarrow_+$ rule or the $L\neg_+$ rule is applied.

To find an interpolant Θ for this specific case the two definitions given in 2.2 and 2.3 are also changed, consistent with the changes in the partition:

$$f^+ \vdash X^+, \Theta \text{ and } f^-, \Theta \vdash X^- \tag{2.26}$$

$$L(\Theta) = L(f^+, X^+) \cap L(f^-, X^-) \tag{2.27}$$

As shown previously we start from the implication rule 2.4 and we try different possibilities.
The inference rule that is derived is the following:

$$\frac{f^+; f^- \vdash X^-, A; X^+ \qquad f^-, B; f^+ \vdash X^+; X^-}{f^-, A \rightarrow B; f^+ \vdash X^+; X^-} L \rightarrow_+ \tag{2.28}$$

It is important to notice that in 2.28 the left child is the only sequent in which there is the switch in partition that we explained before, hence the definition of interpolant with the switch in partition is only used when working with that sequent, and for the other ones we use the normal definition.

Looking at the root sequent of 2.28 (or 2.5) we use the first sequent in 2.2 to derive the sequent for the right child and we use the first sequent in 2.26 to derive the sequent for the left child, these are the axioms of 2.29.

$$\frac{\dfrac{f^-, \Theta_1 \vdash A, X^- \qquad f^-, B \vdash X^-, \Theta_2}{f^-, (A \rightarrow B), \Theta_1 \vdash X^-, \Theta_2} (L \rightarrow)}{f^-, (A \rightarrow B) \vdash X^-, (\Theta_1 \rightarrow \Theta_2)} (R \rightarrow) \tag{2.29}$$

In the proof, by looking from top to bottom, we can notice that formulas A and B are unified to form one formula with the implication connective which is correct as we are applying the $L \rightarrow_+$ rule, but also the two interpolants Θ_1 and Θ_2 are unified to form the next interpolant using the implication connective. The interpolant for the root sequent is thus $\Theta_1 \rightarrow \Theta_2$. One important thing to notice is that we take the interpolant Θ_1 from the sequent which contains formula A and we take the interpolant Θ_2 from the sequent which contains formula B.

We perform the same procedure looking at the root sequent of 2.28 (or 2.5) and using the second sequent in 2.2 for the right child and the second sequent in 2.26 for the left child. As before, we derive the two new sequents which are the axioms in the following proof:

$$\frac{f^+ \vdash X^+, \Theta_1 \qquad f^+, \Theta_2 \vdash X^+}{f^+, (\Theta_1 \rightarrow \Theta_2) \vdash X^+} (L \rightarrow) \tag{2.30}$$

Considering 2.29 this suggests that when the rule $L \rightarrow_+$ is applied, we should combine the interpolants by $\theta := \theta_1 \rightarrow \theta_2$.

Finally, we confirm that this way of combining interpolants respects the language condition:

$$L(\Theta_1) \subseteq L(f^+, X^+) \cap L(f^-, X^-, A) \tag{2.31}$$

$$L(\Theta_2) \subseteq L(f^-, X^-, B) \cap L(f^+, X^+) \tag{2.32}$$

$$\begin{aligned} L(\Theta_1 \rightarrow \Theta_2) &= L(\Theta_1) \cup L(\Theta_2) \\ &\subseteq (L(f^+, X^+) \cap L(f^-, X^-, A)) \\ &\quad \cup (L(f^-, X^-, B) \cap L(f^+, X^+)) \\ &= (L(f^-, X^-, A) \\ &\quad \cup L(f^-, X^-, B)) \cap L(f^+, X^+) \\ &= L(f^-, X^-, A \rightarrow B) \cap L(f^+, X^+) \end{aligned} \tag{2.33}$$

These are only two of the 16 logical rules of inference for propositional logic that were derived for our system. The full set of rules can be found in Appendix A. For propositional logic, all rules of inference are fully deterministic.

2.3 Rules for Modal logic and star-free PDL

For modal logic, we add two rules: weakening and the modality rule. For star-free PDL we add the semicolon rules and the union rules.

2.3.1 Modality and weakening rules

The weakening rule is used to add formulas in the premises or conclusions (if we consider the rule going from top to bottom, if we consider it going from bottom to top we are removing them). The formal definition of the rule can be seen in 2.34 where, $f^- \subseteq f^{-\prime}$, $f^+ \subseteq f^{+\prime}$, $X^+ \subseteq X^{+\prime}$ and $X^- \subseteq X^{-\prime}$.

$$\frac{f^-; f^+ \vdash X^+; X^-}{f^{-\prime}; f^{+\prime} \vdash X^{+\prime}; X^{-\prime}} \text{ Weak} \tag{2.34}$$

This rule is the only non-deterministic rule used in the prover. The prover always checks if the deterministic rules can be applied first, and only if no other rule can be applied then the weakening

rule is used. When the rule is applied the prover uses a depth-first search approach to find one application of the rule which leads to a valid proof. It is possible to implement this rule such that the prover removes one or more formulas (and try all possible combinations) from the premises and/or conclusions in the sequent and try to prove the resulting sequents. However, we decided to implement the weakening rule by trying all combinations where only one formula is removed. If necessary the rule can be applied multiple times in succession, which allows all possibilities to be tested. This implementation deviates from the original rule as in the proofs we have, for instance, n weakening rules applied consecutively, instead of only one rule which removes n formulas. We used this implementation as it resulted in overall faster performance of the prover.

For this rule we don't have four cases as for the rules for propositional logic since the rule is not applied to a formula in the sequent but to the sequent itself. When this rule is applied the interpolant does not change because if we have the sequent $f^-; f^+ \vdash X^+; X^-$ and Θ is an interpolant then Θ is also an interpolant for $f^-, A; f^+, B \vdash X^+, C; X^-, D$ where A,B,C and D could be empty or contain multiple formulas.

The second rule is the modality rule:

$$\frac{f \vdash X}{[\alpha]f \vdash [\alpha]X}\ \text{Mod} \tag{2.35}$$

it has an important restriction that it can only be applied if the sequent has conclusions either empty or with one formula with modality. The rule is used to add (or remove depending on how we read it) the $[\alpha]$ modality to all formulas in the premises and if present the formula in the conclusion. When this rule is applied the modality $[\alpha]$ is also added to the interpolant, thus the new interpolant becomes $[\alpha]\Theta$. If it's the case that the interpolant is $\top$ the interpolant remains unchanged. This is because $[\alpha]\top$ is the same as $\top$. For this rule, we have only one case (not L/R or +/-) because as the weakening rule we can consider the rule being applied to the entire sequent rather than to a formula in the sequent.

2.3.2 Semicolon and union rules

For PDL we use six more rules (also present in Leivant's system D) the ; (semicolon) rules (shown in 2.36 and 2.37) and the $\cup$ (union) rules (shown in 2.38, 2.39, 2.40 and in 2.41). These rules are mainly used to change the notation of a sequent from star-free PDL notation to multi-modal logic notation.

The two semicolon rules are essentially the same if applied in the premises or conclusions and also when applied in the left or right partition. We could give the rules using the partitioned sequent but this would result in 4 essentially equal rules so we decided to use the sequent definition without partitions :

$$\frac{f, [\alpha][\beta]A \vdash X}{f, [\alpha;\beta]A \vdash X}\ \text{L}\ ; \tag{2.36}$$

$$\frac{f \vdash X, [\alpha][\beta]A}{f \vdash X, [\alpha;\beta]A}\ \text{R}\ ; \tag{2.37}$$

When any of the semicolon rules is used the interpolant does not change.

The union rules can be interpreted as a conjunction of programs so the rule changes if applied to premises or the conclusions. When applied to the premises it is not sensitive to partitions, however, when applied to the conclusions it is, the rule for the sequent calculus remains the same but the rule for computing the interpolants changes (see also Appendix A).

$$\frac{f^-, [\alpha]A, [\beta]A; f^+ \vdash X^+; X^-}{f^-, [\alpha \cup \beta]A; f^+ \vdash X^+; X^-}\ L\cup_+ \tag{2.38}$$

For the above rule the interpolant does not change.

$$\frac{f^-; f^+, [\alpha]A, [\beta]A, \vdash X^+; X^-}{[\alpha \cup \beta]A, f^-; f^+ \vdash X^+; X^-}\ L\cup_- \tag{2.39}$$

For the above rule the interpolant does not change.

$$\frac{f^-; f^+ \vdash X^+, [\alpha]A; X^- \qquad f^-; f^+ \vdash X^+, [\beta]A; X^-}{f^-; f^+ \vdash X^+, [\alpha \cup \beta]A; X^-}\ R\cup_+ \tag{2.40}$$

$$\frac{f^-; f^+ \vdash X^+; X^-, [\alpha]A, \qquad f^-; f^+ \vdash X^+; X^-, [\beta]A}{f^-; f^+ \vdash X^+; X^-, [\alpha \cup \beta]A}\ R\cup_- \tag{2.41}$$

For the remaining rules, however, the interpolant changes. As mentioned before, the union of two programs can be interpreted as the conjunction of the two. In fact $[\alpha \cup \beta]A \leftrightarrow [\alpha]A \wedge [\beta]A$ (Gattinger, 2014). To know how to combine the interpolants one can simply look at how interpolants are combined in the $\wedge L_+$ and $\wedge L_-$ rules (shown in Appendix A).

For $\cup R_+$ rule 2.40 the interpolant Θ of the root sequent is the conjunction of the interpolants Θ_1 and Θ_2 of the left and right child. Instead for $\cup R_-$ rule 2.41 the interpolant Θ is the disjunction of the interpolants Θ_1 and Θ_2.

2.4 Implementation in Python

The prover using sequent calculus as proof system and Maehara's partition interpolation method for computing the interpolant was built

on python 3.7.3. The full code of this project can be found in the GitHub repository: `https://github.com/FrancescaPerin/BScProject`

Python was chosen as programming language because it is higher level, this allowed us not to take into account various aspects, such as memory management or algorithms to store arrays which deviate from the scope of implementing sequent calculus and Maehara's Method. Another aspect of Python which was useful in this project is that the code is more readable, this is a good advantage especially for the rules because by looking at them one can easily understand how the rule is being executed. This choice has an impact on the overall performance of the program. Another possible choice of language was Prolog, we chose Python instead as we could build our own architecture. The prover was implemented first only for propositional logic then extended to modal logic and then further extended to star-free PDL. This language enabled us to create an architecture and then modifying it to allow for the various extensions required to reach star-free PDL starting from propositional logic. This also enabled us to keep one program which can deal with all three logics.

The main file of the prover is sequentCalc.py in this file we define all inference rules for all logics and also all rules for computing the interpolants.

For propositional rules we define, for each connective and for left and right, a class which contains four main functions: the function `canApply()`, the functions `stepRight()`, `stepLeft()`, and finally the function `interpolate()`.

Function `canApply()` receives as input a set of formulas and checks if a connective (specific for the rule) is present in one of the formulas. If the connective is present and can be simplified the function returns a list with True only at the position of the formula in which the connective was found. This function is called with the right and left premises or with the right and left conclusions.

Functions `stepLeft()` and `stepRight()` take the formula that was found by `canApply()` and a sequent and applies the actual inference rule, in general removing the formula from the sequent, obtaining the left and right operands and inserting them back in the sequent in the correct partitions. Function `stepLeft()` is used for the left partition and function `stepRight()` is used for the right partition.

Finally function `interpolate()` takes as input the previous interpolant(s) and a boolean. The boolean is to know which function was called either `stepLeft()` or `stepRight()`. This enables the `interpolate()` function to know given the previous interpolant(s) which rule to apply to compute the next one.

For modal logic and star-free PDL rules, the idea is the same we always have a function `canApply()` that checks if a inference rule can be applied to a sequent, a function `step()` that actually applies the rule, and function `interpolate()` which given one or two interpolants computes the next.

The function responsible for determining the validity of a sentence is the recursive function `solve()`. This function takes the starting sequent that we want to prove and checks if any `canApply()` function returns a list with one index being 'True'. To have a more compact proof the function first checks all rules with branching factor of one between the propositional rules, then the propositional rules with branching factor two, then the modality, semicolon and union rules and lastly weakening. Weakening is checked last because we want the prover to apply the weakening rule only if no other rule can be applied.

If `canApply()` returns a list with one index being 'True' then the corresponding function of the type `step()` is called and the new child or children sequents are stored. Furthermore, we also store in `self.__rule` the `interpolate()` function that has to be called in relation to the specific rule applied and a boolean value which is also going to be needed for `interpolate()` as previously mentioned.

If all `canApply()` return an empty list, it's the case that no rule can be applied and the program has reached a leaf (fully reduced formula), thus the function returns true if all leaves are axioms otherwise false. The output of the `solve()` function is the validity of the tested sequent.

If `solve()` returns True then the recursive function `calcInterpolant()` is called. The rule takes the original sequent and is called recursively on the children. It checks if the sequent is an axiom with the function `isAxiom()`, if it's not then it moves to the next sequent, if it is then it calls the function `axiomInterpolant()` which computes the interpolant of the axiom as explained in Section 2.1 plus a few more cases. As mentioned, in `self.__rule` we store all interpolant rules to be used and in the correct order. Once the interpolants of the axioms are computed recursion starts moving backward. Recall that we compute

the interpolants moving down the proof. We call each function `interpolate()` stored giving it the old computed interpolant such that it will compute the next, until all rules are applied, thus reaching the final interpolant.

Once the final interpolant is reached, the two new entailments of the definition are generated and checked using the same `solve()` function.

The program generates a .tex file with the LaTeX code of the proof and a PDF image. This must be manually saved by the user because if the program is run a second time the files get overwritten with the new proof. This is why the prover only generates the PDF image when formulas are tested manually and not randomly. For more information on how to test the code see the README.md file in the GitHub repository.

3 Results

The aim of this project is to show that Craig interpolation holds for propositional logic, modal logic up to star-free PDL, by implementing a theorem prover (based on Maehara's method) that generates interpolants and checks their correctness.

The prover was tested with 3 sets of 150.000 randomly generated formulas, one set for each logic: propositional logic, basic modal logic, and star-free PDL logic. For propositional logic 44.120 were found to be valid and in all cases, the final computed interpolant was correct and respected the definition and constraints. For basic modal logic and for star-free PDL the valid sentences found were respectively 23.815 and 23.503. For all valid formulas, the final computed interpolant was correct and respected the definition and constraints.

Because modal logic extends propositional logic and the formulas are generated randomly it is the case that, when testing formulas for modal logic, several propositional sentences are also tested. This is also the case for star-free PDL, which can generate and test formulas for propositional logic and basic modal logic.

The prover was further manually tested with the sets of sentences below. For this sets of sentences, the validity was already known so we used them to test that the prover would indeed give the same output.

3.1 Test set for propositional logic

Provable:

1. $\top \to \top$
2. $\bot \to \top$
3. $\bot \to p$
4. $p \to \top$
5. $\top \to p \vee \neg p$
6. $\top \to \neg(p \wedge \neg p)$
7. $p \to p$
8. $(p \wedge q) \to (q \vee r)$
9. $(p \wedge q \wedge r) \to (q \vee r \vee s)$
10. $(p \wedge q \wedge r) \to (r \vee s \vee t)$
11. $(p \to q) \to (\neg q \to \neg p)$
12. $((p \vee t) \to (q \vee r)) \to ((\neg r \wedge s) \to (\neg q \to \neg p))$
13. $((p \vee t) \to (q \vee r)) \to (\neg r \to (\neg q \to \neg p))$

Not provable:

1. $p \to q$
2. $q \to \ q$
3. $\top \to q \vee (r \wedge s)$
4. $(p \vee q \vee r) \to (r \wedge s \wedge t)$
5. $((p \wedge t) \to (q \wedge r)) \to (\neg r \to (\neg q \to \neg p))$

The prover gave the correct output and found a valid interpolant for all sentences. Below we show the sequent calculus proof for formula 13 (of the provable set) and the interpolant computed for each sequent in the proof.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{p;\overset{p}{\vdash} p,t;r,q}R\vee_+ \quad \dfrac{}{q;p\overset{q}{\vdash} r,q; \qquad r;p\overset{r}{\vdash} r,q;}L\vee_+}{p;\overset{p}{\vdash}(p\vee t);r,q \qquad\qquad (q\vee r);p \overset{(q\vee r)}{\vdash} r,q;}L\to_+}{((p\vee t)\to(q\vee r));p \overset{(p\to(q\vee r))}{\vdash} r,q;}L\neg_-}{((p\vee t)\to(q\vee r));\neg(q),p \overset{(p\to(q\vee r))}{\vdash} r;}L\neg_-}{((p\vee t)\to(q\vee r));\neg(r),\neg(q),p \overset{(p\to(q\vee r))}{\vdash} ;}R\neg_+}{((p\vee t)\to(q\vee r));\neg(r),\neg(q) \overset{(p\to(q\vee r))}{\vdash} \neg(p);}R\to_+}{((p\vee t)\to(q\vee r));\neg(r) \overset{(p\to(q\vee r))}{\vdash} (\neg(q)\to\neg(p));}R\to_+}{((p\vee t)\to(q\vee r)); \overset{(p\to(q\vee r))}{\vdash} (\neg(r)\to(\neg(q)\to\neg(p)));}R\to_+ \tag{3.1}$$

The final interpolant for the formula $((p \vee t) \to (q \vee r)) \to (\neg r \to (\neg q \to \neg p))$ is $p \to (q \vee r)$.

3.2 Test set for modal logic

Provable:

1. $[a](p \wedge q) \rightarrow ([a]p \wedge [a]q)$
2. $[a](p \rightarrow q) \rightarrow ([a]p \rightarrow [a]q)$
3. $\neg[a](p \vee q) \rightarrow (\neg[a]p \vee \neg[a]r)$
4. $\neg[a]\neg(p \vee q) \rightarrow (\neg[a]\neg p \vee \neg[a]\neg q)$
5. $(\neg[a]\neg p \vee \neg[a]\neg q) \rightarrow \neg[a]\neg(p \vee q)$
6. $[a](p \wedge [a]q) \rightarrow [a][a]q$
7. $[a](q \rightarrow r) \rightarrow (\neg[a]r \rightarrow \neg[a]q)$

Not provable:

1. $[a](p \vee q) \rightarrow ([a]p \vee [a]q)$
2. $[a][a]p \rightarrow p$
3. $[a]p \rightarrow q$
4. $\neg[a]\neg p \rightarrow [a](p \wedge p)$

The prover gave the correct output for all sentences and also found a correct interpolant for all valid sentences.

3.3 Test set for star-free PDL

Provable:

1. $[a \cup b]q \rightarrow [b \cup a]q$
2. $[a][b](p \wedge q) \rightarrow [a \ ; \ b](q \vee r)$
3. $[(a \ ; \ b \ ; \ c) \cup (a \ ; \ d \ ; \ c)]p \rightarrow [a \ ; \ (b \cup d) \ ; \ c]p$
4. $[a \cup b](p \wedge q) \rightarrow ([c]\bot \rightarrow [b \cup c](q \vee r))$
5. $(p \wedge q) \rightarrow [a]\top$
6. $((p \vee s) \rightarrow [a \ ; \ b](q \rightarrow \neg q)) \rightarrow$
   $(\neg[a \ ; \ \mathrm{b}]\neg q \rightarrow \neg(p \vee s))$

Not provable:

1. $((p \vee s) \rightarrow [a \ ; \ b](q \rightarrow \neg q)) \rightarrow$
   $(\neg[a \ ; \ b]q \rightarrow \neg(p \vee r))$
2. $(p \wedge q) \rightarrow [a][a](q \vee r)$
3. $[a \ ; \ b]p \rightarrow [b \ ; \ a](p \vee r)$
4. $(p \rightarrow [a \ ; \ b]q) \rightarrow ([a \ ; \ b]p \rightarrow q)$

The prover gave the correct output for all sentences and also found a correct interpolant for all valid sentences.

3.4 Example

We tested our prover with the formula (in basic modal logic):

$$[a]((q \rightarrow r)) \rightarrow (\neg([a](r)) \rightarrow \neg([a](q))) \tag{3.2}$$

The formula is rewritten in sequent notation and is the root sequent of the proof.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{q;\overset{q}{\vdash} q;r} \quad \dfrac{}{r;q\overset{r}{\vdash} r;}}{(q \rightarrow r);q \overset{(q\rightarrow r)}{\vdash} r;}L \rightarrow_{+}}{[a]((q \rightarrow r));[a](q) \overset{[a]((q\rightarrow r))}{\vdash} [a](r);}Mod[a]}{[a]((q \rightarrow r));\neg([a](r)),[a](q) \overset{[a]((q\rightarrow r))}{\vdash} ;}L\neg_{-}}{[a]((q \rightarrow r));\neg([a](r)) \overset{[a]((q\rightarrow r))}{\vdash} \neg([a](q));}R\neg_{+}}{[a]((q \rightarrow r)); \overset{[a]((q\rightarrow r))}{\vdash} (\neg([a](r)) \rightarrow \neg([a](q)));}R \rightarrow_{+} \tag{3.3}$$

For this example the prover computed as final interpolant of the root sequent the formula $[a](q \rightarrow r)$. After computing the interpolant the prover checks that the sequents

$$[a](q \rightarrow r);\vdash [a](q \rightarrow r); \tag{3.4}$$

$$[a](q \rightarrow r);\vdash (\neg([a](r)) \rightarrow \neg[a](q)); \tag{3.5}$$

are also valid. This is the case thus the interpolant is valid for the original formula in 3.2 and it holds that the two entailments are valid:

$$[a]((q \rightarrow r)) \rightarrow [a](q \rightarrow r) \tag{3.6}$$

$$[a](q \rightarrow r) \rightarrow (\neg([a](r)) \rightarrow \neg([a](q))) \tag{3.7}$$

and that the language constrain is also valid:

$$\begin{aligned} L([a](q \rightarrow r)) \subseteq\ & L([a](q \rightarrow r)) \\ & \cap L(\neg([a](r)) \rightarrow \neg([a](q))) \end{aligned} \tag{3.8}$$

The proof above (3.3) was taken directly from our prover, which in case of a valid sentence, generates the proof in LaTeX code and also a pdf version.

A second example of proof for star-free PDL can be found in Appendix B.

4 Discussion

The prover was manually tested with a small set of formulas for which the validity was known. This was done to ensure that the prover would give the same results. The results were promising as the prover provided the same results for all cases, with one exception (see Section 3.1), and the interpolants computed were all valid. With regard to the 450.000 formulas generated randomly and then tested the results were also positive.

One of the limitations of this project is that the sets of formulas (for which the validity was known) were very small. Therefore, it was not possible to test in depth the reliability of the prover, thus further testing is required. One possibility would be to compare our prover with a different one (that could use a different system such as analytic tableaux). Given a sentence, the two provers would determine the validity independently and then the results compared. If our prover is fully reliable we should find the same set of valid and non-valid formulas.

It would also be interesting to compare our prover with one which also computes interpolants. However, this could be more challenging because as mentioned there could be more than one interpolant for a sentence, as the final interpolant depends on the proof of validity. Thus the two provers could both find an interpolant that is valid, but they might result in two different formulas and might not be comparable.

Some of the undesired cases that could arise are those in which the prover is wrong in determining the validity of the sentence, but also if the prover finds a valid sentence but the interpolant found does not satisfy its constraints. For example, it does not satisfy one or both of the implications in 2.2 or the language constraint in 2.3. If this happens when further testing the code, in our opinion is more likely to be a mistake in terms of implementation of the rules in the code or a case that was not foreseen and thus not implemented, rather than the derived rules being wrong or the method used not suitable. This opinion is based on the fact that all inference rules and most related interpolant rules were also proven. The rules were also double-checked by rewriting a connective in terms of different ones. For instance, rewriting $A \rightarrow B$ as $\neg A \vee B$, thus applying the rule for implication or in the other case the negation and disjunction rules results in the same sequents. The same can be also performed for the interpolant rules. This is another way to ensure that all rules are congruent as a whole and as a system.

One of the interpolant rules which was not fully derived and proven is the modality rule. In our rule, we mentioned that the inference rule and the related interpolant rule are not sensitive to partitions. However, the case that modality is not added if the interpolant is $\bot$, but it is if the interpolant is $\top$, suggests that it might be the case that where the formulas with modality are found in the premises and/or conclusion (possibly even with respect to partitions) plays a role in determining the rule for modifying the interpolant, where either modality is added or the interpolant remains the same.

5 Conclusions

Star-free PDL logic can be seen as a notation variant of multi-modal logic, for which Craig's interpolation property has been proven. In this project, we wanted to implement Maehara's partition interpolation method to compute the interpolants and use sequent calculus as proof system.

The results were promising as, for both the tests sets and the randomly generated sentences, if the formula was found to be valid the final interpolant was correct and respected the definition. However, because the prover was not tested heavily it is not possible to determine if it is fully reliable in determining the validity of a sentence and computing the interpolants. For this reason, we cannot exclude the possibility of our prover proving a sentence correct when its not, the opposite case or finding a non correct interpolant. In which case, as we argued in Section 4, it is more likely to be a problem of implementing the rules in the program or as we discussed a problem in the interpolant rule for the modality rule which could possibly depend on partitions.

References

E. Bańcenas, J. Lavalle-Martińez, G. Molero-Castillo, and A. Velaźquez-Mena. Craig interpolation on the logic of knowledge. Unpublished. URL `http://ceur-ws.org/Vol-2264/paper2.pdf`.

M. Borzechowski. Tableaukalkül für PDL und interpolation. Diplomarbeit, Department of Mathematics, FU Berlin. Unpublished, 1988.

S.A. Cook and R.A Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44(1):36–50, 1979. doi:10.2307/2273702.

G. D' Agostino. Interpolation in non-classical logics. *Synthese*, 164(3):421–435, 2008. doi:10.1007/s11229-008-9359-x.

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

T. Kowalski. PDL has interpolation. *Journal of Symbolic Logic*, 67(3):933–946, 2002. URL `https://www.jstor.org/stable/3648548`.

T. Kowalski. Retraction note for PDL has interpolation. *Journal of Symbolic Logic*, 69(3):935, 2004. doi:10.2178/jsl/1096901777.

M. Kracht. *Tools and techniques in modal logic.*, volume 142. Springer Berlin, 1999. URL `https://wwwhomes.uni-bielefeld.de/mkracht/html/tools/book.pdf`.

D. Leivant. Proof theoretic methodology for propositional dynamic logic. In J. Díaz and I. Ramos, editors, *Formalization of Programming Concepts*, pages 356–373, Berlin, Heidelberg, 1981. Springer Berlin Heidelberg. doi:10.1007/3-540-10699-5_111.

C. Lutz and F. Wolter. Foundations for uniform interpolation and forgetting in expressive description logics. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, volume 2, pages 989–995. AAAI Press, 2011. URL `https://www.ijcai.org/Proceedings/11/Papers/170.pdf`.

J. X. Madarász. The craig interpolation theorem in multi-modal logics. *Bulletin of the Section of Logic*, 3(24):147–151, 1995. doi:10.1007/s11229-008-9359-x. URL `https://www.filozof.uni.lodz.pl/bulletin/pdf/24_3_5.pdf`.

K.L. McMillan. Interpolation and model checking. In E.M. Clarke, T.A. Henzinger, H. Veith, and R. Bloem, editors, *Handbook of Model Checking.*, pages 421–446. Springer Berlin, 2018. doi:10.1007/978-3-319-10575-8_14.

G. Takeuti. *Proof Theory.* North-Holland Amsterdam/Oxford, 1975. URL `http://www.getcited.org/pub/101589483`.

B. ten Cate, E. Franconi, and I Seylan. Beth definability in expressive description logics. *Journal of Artificial Intelligence Research*, 48:347–414, 2013. doi:10.1613/jair.4057.

A Appendix

A.1 LEFT IMPLICATION RULE

- Left partition: (note partition have changed in one of the children)

$$\frac{f^+; f^- \vdash X^-, A; X^+ \qquad f^-, B; f^+ \vdash X^+; X^-}{f^-, A \to B; f^+ \vdash X^+; X^-}\,(L \to_+)$$

Interpolant: $\Theta_1 \to \Theta_2$
Proof:

$$\frac{\dfrac{f^-, \Theta_1 \vdash A, X^- \qquad f^-, B \vdash X^-, \Theta_2}{f^-, (A \to B), \Theta_1 \vdash X^-, \Theta_2}\,(L \to)}{f^-, (A \to B) \vdash X^-, (\Theta_1 \to \Theta_2)}\,(R \to)$$

$$\frac{f^+ \vdash X^+, \Theta_1 \qquad f^+, \Theta_2 \vdash X^+}{f^+, (\Theta_1 \to \Theta_2) \vdash X^+}\,(L \to)$$

$L(\Theta_1) = L(f^+, X^+) \cap L(f^-, X^-, A)$
$L(\Theta_2) = L(f^-, X^-, B) \cap L(f^+, X^+)$

$$\begin{aligned} L(\Theta_1 \to \Theta_2) &= L(\Theta_1) \cup L(\Theta_2) \\ &= (L(f^+, X^+) \cap L(f^-, X^-, A)) \cup (L(f^-, X^-, B) \cap L(f^+, X^+)) \\ &= (L(f^-, X^-, A) \cup L(f^-, X^-, B)) \cap L(f^+, X^+) \\ &= L(f^-, X^-, A \to B) \cap L(f^+, X^+) \end{aligned}$$

- Right partition:

$$\frac{f^-; f^+ \vdash X^+, A; X^- \qquad f^-; B, f^+ \vdash X^+; X^-}{f^-; A \to B, f^+ \vdash X^+; X^-}\,(L \to_-)$$

Interpolant: $\Theta_1 \wedge \Theta_2$
Proof:

$$\frac{f^- \vdash X^-, \Theta_1 \qquad f^- \vdash X^-, \Theta_2}{f^- \vdash X^-, (\Theta_1 \wedge \Theta_2)}\,(L\wedge)$$

$$\frac{\dfrac{\dfrac{f^+, \Theta_1 \vdash X^+, A}{f^+, \Theta_1, \Theta_2 \vdash X^+, A}\,(\text{WEAK L}) \qquad \dfrac{f^+, B, \Theta_2 \vdash X^+}{f^+, B, \Theta_1, \Theta_2 \vdash X^+}\,(\text{WEAK L})}{f^+, (A \to B), \Theta_1, \Theta_2 \vdash X^+}\,(L \to)}{f^+, (A \to B), (\Theta_1 \wedge \Theta_2) \vdash X^+}\,(L\wedge)$$

$L(\Theta_1) = L(f^-, X^-) \cap L(f^+, X^+, A)$
$L(\Theta_2) = L(f^-, X^-) \cap L(f^+, B, X^+)$

$$\begin{aligned} L(\Theta_1 \to \Theta_2) &= L(\Theta_1) \cup L(\Theta_2) \\ &= (L(f^-, X^-) \cap L(f^+, X^+, A)) \cup (L(f^-, X^-) \cap L(f^+, B, X^+)) \\ &= (L(f^+, X^+, A) \cup L(f^+, B, X^+)) \cap L(f^-, X^-) \\ &= L(f^+, X^+, A \to B) \cap L(f^-, X^-) \end{aligned}$$

A.2 LEFT AND RULE

- Left partition:

$$\frac{f^-, A, B; f^+ \vdash X^+; X^-}{f^-, A \wedge B; f^+ \vdash X^+; X^-}\,(L\wedge_+)$$

Interpolant: remains the same just rewriting

- Right partition:

$$\frac{f^-; A, B, f^+ \vdash X^+; X^-}{f^-; A \wedge B, f^+ \vdash X^+; X^-}\,(L\wedge_-)$$

Interpolant: remains the same just rewriting

A.3 LEFT OR RULE

- Left partition:

$$\frac{f^-, A; f^+ \vdash X^+; X^- \qquad f^-, B; f^+ \vdash X^+; X^-}{f^-, A \vee B; f^+ \vdash X^+; X^-}\,(L\vee_+)$$

Interpolant: $\Theta_1 \vee \Theta_2$

Proof:

$$\dfrac{\dfrac{\dfrac{f^-, A \vdash X^-, \Theta_1}{f^-, A \vdash X^-, \Theta_1, \Theta_2}\,(\text{WEAK R}) \qquad \dfrac{f^-, B \vdash X^-, \Theta_2}{f^-, B \vdash X^-, \Theta_1, \Theta_2}\,(\text{WEAK R})}{f^-, (A \vee B) \vdash X^-, \Theta_1, \Theta_2}\,(L\vee)}{f^-, (A \vee B) \vdash X^-, (\Theta_1 \vee \Theta_2)}\,(R\vee)$$

$$\dfrac{\dfrac{f^+, \Theta_1 \vdash A, B, X^+ \qquad f^+, \Theta_2 \vdash A, B, X^+}{f^+, (\Theta_1 \vee \Theta_2) \vdash A, B, X^+}\,(L\vee)}{f^+, (\Theta_1 \vee \Theta_2) \vdash (A \vee B), X^+}\,(R\vee)$$

- Right partition:

$$\frac{f^-; A, f^+ \vdash X^+; X^- \qquad f^-; B, f^+ \vdash X^+; X^-}{f^-; A \vee B, f^+ \vdash X^+; X^-}\,(L\vee_-)$$

Interpolant: $\Theta_1 \wedge \Theta_2$

Proof:

$$\frac{f^- \vdash X^-, \Theta_1 \qquad f^- \vdash X^-, \Theta_2}{f^- \vdash X^-, \Theta_1 \wedge \Theta_2}\,(R\wedge)$$

$$\dfrac{\dfrac{\dfrac{f^+, A, \Theta_1 \vdash X^+}{f^+, A, \Theta_1, \Theta_2 \vdash X^+}\,(\text{WEAK L}) \qquad \dfrac{f^+, B, \Theta_2 \vdash X^+}{f^+, B, \Theta_1, \Theta_2, \vdash X^+}\,(\text{WEAK L})}{f^+, (A \vee B), \Theta_1, \Theta_2 \vdash X^+}\,(L\vee)}{f^+, (A \vee B), (\Theta_1 \wedge \Theta_2) \vdash X^+}\,(L\wedge)$$

A.4 LEFT NOT RULE

- Left partition: (note change in partition)

$$\frac{f^+; f^- \vdash X^-, A; X^+}{f^-, \neg A; f^+ \vdash X^+; X^-}\,(L\neg_+)$$

Interpolant: $\neg\Theta_1$
Proof:

$$\frac{\dfrac{f^-, \Theta_1 \vdash X^-, A}{f^-, \neg A, \Theta_1 \vdash X^-}\,(L\neg)}{f^-, \neg A, \vdash X^-, \neg\Theta_1}\,(R\neg)$$

$$\frac{f^+ \vdash X^+, \Theta_1}{f^+, \neg\Theta_1 \vdash X^+}\,(L\neg)$$

- Right partition:

$$\frac{f^-; f^+ \vdash A, X^+; X^-}{f^-; \neg A, f^+ \vdash X^+; X^-}\,(L\neg_-)$$

Interpolant: Θ_1 interpolant does not change
Proof:

$$\frac{f^+, \Theta_1 \vdash X^+}{f^+, \neg A, \Theta_1 \vdash X^+}\,(L\neg)$$

$$f^- \vdash X^-, \Theta_1$$

A.5 RIGHT IMPLICATION RULE

- Left partition:

$$\frac{f^-; A, f^+ \vdash X^+, B; X^-}{f^-; f^+ \vdash X^+, A \to B; X^-}\,(R \to_+)$$

Interpolant: $\Theta_1 = \Theta_2$ no change
Proof:

$$\frac{f^+, A, \Theta_1 \vdash X^+, B}{f^+, \Theta_2 \vdash X^+, (A \to B)}\,(R \to)$$

$$f^- \vdash X^-, \Theta_1$$
$$f^- \vdash X^-, \Theta_2$$

- Right partition:

$$\frac{f^-, A; f^+ \vdash X^+; B, X^-}{f^-; f^+ \vdash X^+; A \to B, X^-}\,(R \to_-)$$

Interpolant: $\Theta_1 = \Theta_2$ no change
Proof:

$$\frac{f^-, A, \Theta_1 \vdash X^-, B}{f^-, \Theta_2 \vdash X^-, (A \to B)}\,(R \to)$$

$$f^+, \Theta_1 \vdash X^+$$
$$f^+, \Theta_2 \vdash X^+$$

A.6 RIGHT AND RULE

- Left partition:

$$\frac{f^-; f^+ \vdash X^+, A; X^- \qquad f^-; f^+ \vdash X^+, B; X^-}{f^-; f^+ \vdash X^+, A \wedge B; X^-}\,(R\wedge_+)$$

Interpolant: $\Theta_1 \wedge \Theta_2$
Proof:

$$\frac{f^- \vdash X^-, \Theta_1 \qquad f^- \vdash X^-, \Theta_2}{f^- \vdash X^-, (\Theta_1 \wedge \Theta_2)}\,(R\wedge)$$

$$\frac{\dfrac{\dfrac{f^+, \Theta_1 \vdash X^+, A}{f^+, \Theta_1, \Theta_2 \vdash X^+, A}\,(\text{WEAK L}) \qquad \dfrac{f^+, \Theta_2 \vdash X^+, B}{f^+, \Theta_1, \Theta_2 \vdash X^+, B}\,(\text{WEAK L})}{f^+, \Theta_1, \Theta_2 \vdash X^+, (A \wedge B)}\,(R\wedge)}{f^+, (\Theta_1 \wedge \Theta_2) \vdash X^+, (A \wedge B)}\,(L\wedge)$$

- Right partition:

$$\frac{f^-; f^+ \vdash X^+; A, X^- \qquad f^-; f^+ \vdash X^+; B, X^-}{f^-; f^+ \vdash X^+; A \wedge B, X^-}\,(R\wedge_-)$$

Interpolant: $\Theta_1 \vee \Theta_2$
Proof:

$$\frac{\dfrac{\dfrac{f^- \vdash X^-, A, \Theta_1}{f^- \vdash X^-, A, \Theta_1, \Theta_2}\,(\text{WEAK R}) \qquad \dfrac{f^- \vdash X^-, B, \Theta_2}{f^- \vdash X^-, B, \Theta_1, \Theta_2}\,(\text{WEAK R})}{f^- \vdash X^-, (A \wedge B), \Theta_1, \Theta_2}\,(R\wedge)}{f^- \vdash X^-, (A \wedge B), (\Theta_1 \vee \Theta_2)}\,(R\vee)$$

$$\frac{f^+, \Theta_1 \vdash X^+ \qquad f^+, \Theta_2 \vdash X^+}{f^+, (\Theta_1 \vee \Theta_2) \vdash X^+}\,(L\vee)$$

A.7 RIGHT OR RULE

- Left partition:

$$\frac{f^-; f^+ \vdash X^+, A, B; X^-}{f^-; f^+ \vdash X^+, A \vee B; X^-}\,(R\vee_+)$$

Interpolant: remains the same just rewriting

- Right partition:

$$\frac{f^-; f^+ \vdash X^+; A, B, X^-}{f^-; f^+ \vdash X^+; A \vee B, X^-}\,(R\vee_-)$$

Interpolant: remains the same just rewriting

A.8 RIGHT NOT RULE

- Left partition:

$$\frac{f^-; A, f^+ \vdash X^+; X^-}{f^-; f^+ \vdash X^+, \neg A; X^-}\,(R\neg_+)$$

Interpolant: $\Theta_1 = \Theta_2$ no change

Proof:

$$\frac{f^+, A, \Theta_1 \vdash X^+}{f^+, \Theta_2 \vdash X^+, \neg A}\,(R\neg)$$

$$\begin{array}{c} f^- \vdash X^-, \Theta_1 \\ f^- \vdash X^-, \Theta_2 \end{array}$$

- Right partition:

$$\frac{f^-, A; f^+ \vdash X^+; X^-}{f^-; f^+ \vdash X^+; \neg A, X^-}\,(R\neg_-)$$

Interpolant: $\Theta_1 = \Theta_2$ no change
Proof:

$$\frac{f^-, A \vdash X^-, \Theta_1}{f^- \vdash X^-, \neg A, \Theta_2}\,(R\neg)$$

$$\begin{array}{c} f^+, \Theta_1 \vdash X^+, A \\ f^+, \Theta_2 \vdash X^+, A \end{array}$$

A.9 WEAKENING

$$\frac{f \vdash X}{f' \vdash X'}\,(\text{Weak})$$

where $f \subseteq f'$ and $X \subseteq X'$

Interpolant: remains the same just rewriting

A.10 MODALITY

$$\frac{f \vdash X}{[\alpha]f \vdash [\alpha]X}\,(\text{Mod})$$

Interpolant:$[\alpha]\Theta_1$ where Θ_1 is the interpolants of the child. If Θ is $\perp$ the interpolant remains the same.

A.11 LEFT SEMICOLON

$$\frac{[\alpha][\beta]A, f \vdash X}{[\alpha;\beta]A, f \vdash X}\,(;)$$

Interpolant: remains the same just rewriting

A.12 RIGHT SEMICOLON

$$\frac{f \vdash [\alpha][\beta]A, X}{f \vdash [\alpha;\beta]A, X}\,(;)$$

Interpolant: remains the same just rewriting

A.13 LEFT UNION

$$\frac{[\alpha]A, [\beta]A, f \vdash X}{[\alpha \cup \beta]A, f \vdash X} \ (\cup \text{ L})$$

Interpolant: remains the same just rewriting

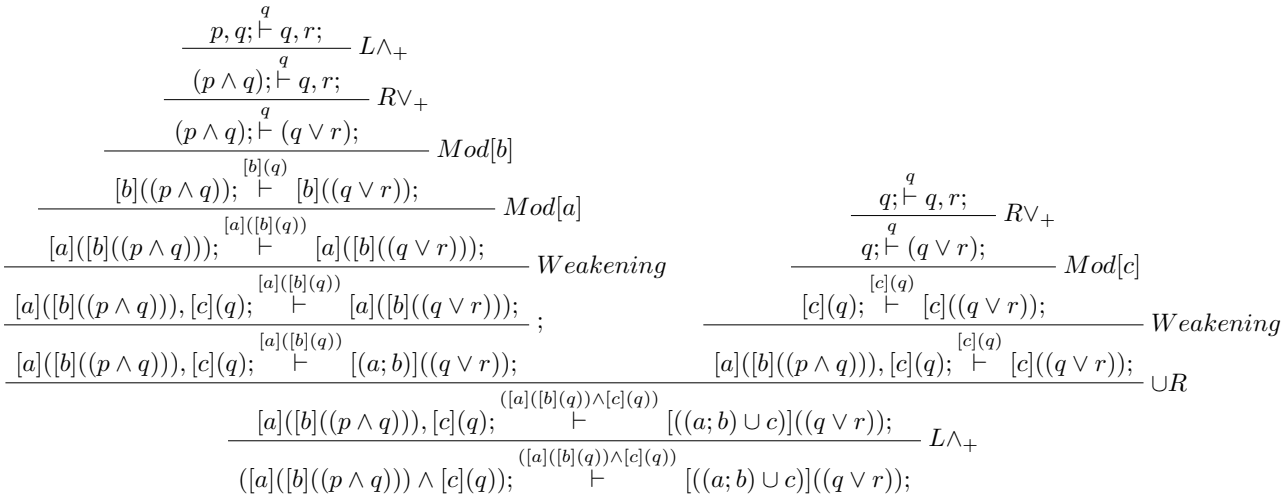
A.14 RIGHT UNION

$$\frac{f \vdash [\alpha]A, X \qquad f \vdash [\beta]A, X}{f \vdash [\alpha \cup \beta]A, X} \ (\cup \text{R})$$

- Left partition: interpolant is $\Theta_1 \wedge \Theta_2$ where Θ_1 and Θ_2 are the interpolants of the children
- Right partition: interpolant is $\Theta_1 \vee \Theta_2$ where Θ_1 and Θ_2 are the interpolants of the children

B Appendix

Example of sequent calculus proof and computation of interpolant for formula in star-free PDL.



The code for this project can be found in the GitHub repository: `https://github.com/FrancescaPerin/BScProject`