A Verified Proof of Craig Interpolation for Basic Modal Logic via Tableaux in Lean

Malvin Gattinger [1]

ILLC, University of Amsterdam

Abstract

We present a formalisation of tableaux for Basic Modal Logic (K) in the Lean proof assistant. We cover soundness, completeness, and Craig Interpolation via tableaux. In the future we aim to extend this proof from Basic Modal Logic to Propositional Dynamic Logic (PDL). Specifically, this is a first step towards a formal verification of Borzechowski (1988) which claims that PDL has Craig Interpolation.

Keywords: Modal Logic, Interpolation, Tableaux, Interactive Theorem Proving

1 Introduction

It is well-known that many modal logics have the Craig interpolation property (see Theorem 3.1 below). However, for Propositional Dynamic Logic (PDL) [4] this is considered an open question. Recently the proof attempt by [2] was translated from German to English [3]. As a first step towards a formal verification of this proof, here we formalise the Basic Modal Logic analogue of the PDL tableaux in [2] in the proof assistant *Lean*. We then show soundness, completeness and Craig interpolation via tableaux. The code is available at `https://github.com/m4lvin/tablean` and `https://doi.org/10.5281/zenodo.6628187`.

Lean. Lean can be considered both a programming language, a theorem prover and a proof assistant. It is based on Type Theory; its main developer is L. de Moura [10]. A large amount of mathematics has been formalised in Lean and is maintained by the *mathlib* project [13]. For this project we use the extended community version of Lean 3.42.1 and refer to `https://leanprover-community.github.io` for further details.

Related Work. The mathlib project does not contain any modal logic so far, but several formalisation projects of modal logics have been done in Lean. Most similar to ours is the work by Wu and Goré [14], who implement a verified decision procedure based on tableaux for the basic modal logics K, KT and S4. They do not discuss interpolation. Other related Lean projects (without tableaux) include the completeness of S5 [1], and formalisations of Public

[1] `malvin@w4eg.eu` · `https://malv.in`

Short presentation at *Advances in Modal Logic 2022*, Rennes, August 2022.

Announcement Logic [8,11]. Modal logics have also been formalised in other proof assistants, e.g. Epistemic Logic in Isabelle [5]. Another similar Isabelle project is a formalised proof of interpolation for classical propositional logic [9]. Finally, we also mention a tableaux prover in Coq for propositional logic [6].

Basic Definitions. We consider basic modal logic, given by the BNF syntax $\varphi ::= \bot \mid p \mid \neg\varphi \mid \varphi\wedge\varphi \mid \Box\varphi$ where $p \in \mathsf{P}$ for some set P of atomic propositions. We use the abbreviations $\varphi \to \psi := \neg(\varphi\wedge\neg\psi)$ and $\varphi\vee\psi := \neg(\neg\varphi\wedge\neg\psi)$. For any φ, we denote by $\mathsf{voc}(\varphi)$ its *vocabulary*. E.g., $\mathsf{voc}((p\vee q)\wedge(\Box q \to \Diamond r)) = \{p, q, r\}$.

We use standard Kripke models $\mathcal{M} = (W, R, V)$ where $R \subseteq W \times W$ and $V\colon W \to \mathsf{Prop}$. As our first example of Lean code, the following defines the standard semantics. Here `formula` is an inductive type for the grammar above, and `char` plays the role of the set P for atomic propositions.

```
structure kripkeModel (W : Type) : Type :=
  (val : W → char → Prop)
  (rel : W → W → Prop)

def evaluate {W : Type} : (kripkeModel W × W) → formula → Prop
| (M,w) ⊥       := false
| (M,w) (· p)   := M.val w p
| (M,w) (∼ φ)   := ¬ evaluate (M,w) φ
| (M,w) (φ ⅄ ψ) := evaluate (M,w) φ ∧ evaluate (M,w) ψ
| (M,w) (□ φ)   := ∀ v : W, (M.rel w v → evaluate (M,v) φ)
```

2 Tableaux

The tableaux system we use consists of the following five *local rules* plus one modal rule. Here X is a set of formulas and $\varnothing$ indicates a *closed* branch.

$$(\bot)\ \frac{X, \bot}{\varnothing} \qquad (\neg)\ \frac{X, \varphi, \neg\varphi}{\varnothing} \qquad (\neg\neg)\ \frac{X, \neg\neg\varphi}{X, \varphi} \qquad (\wedge)\ \frac{X, \varphi\wedge\psi}{X, \varphi, \psi} \qquad (\neg\wedge)\ \frac{X, \neg(\varphi\wedge\psi)}{X, \neg\varphi \mid X, \neg\psi}$$

The five local rules are represented in Lean as follows.

```
inductive localRule : finset formula → finset (finset formula) → Type
| bot {X    } (h : ⊥          ∈ X) : localRule X ∅
| not {X φ  } (h : φ ∈ X ∧ ∼φ ∈ X) : localRule X ∅
| neg {X φ  } (h : ∼∼φ        ∈ X) : localRule X { X \ {∼∼φ}    ∪ {φ}   }
| con {X φ ψ} (h : φ ⅄ ψ      ∈ X) : localRule X { X \ {φ⅄ψ}    ∪ {φ,ψ} }
| nCo {X φ ψ} (h : ∼(φ ⅄ ψ)   ∈ X) : localRule X { X \ {∼(φ⅄ψ)} ∪ {∼φ}
                                                 , X \ {∼(φ⅄ψ)} ∪ {∼ψ}  }
```

The *modal rule* uses the *projection* $X_\Box := \{\varphi \mid \Box\varphi \in X\}$. It may only be applied when X is *simple*, i.e. consists of (negated) atoms and boxed formulas.

$$(\neg\Box)\ \frac{X, \neg\Box\varphi}{X_\Box, \neg\varphi}$$

Example 2.1 Fig. 1 shows a tableau for $(\Box\Box(p\wedge q)\vee\Box\bot) \to (\Box(r\vee\Box q)\vee\Box r)$. It consists of four local tableaux, connected by three applications of the modal rule, marked with dotted lines. We spell out all abbreviations using $\neg$ and $\wedge$.

In Lean we define tableaux as two inductive types. For a `localTableau` for X we need either a local rule applicable to X and local tableaux for the `next` nodes, or that X is `simple`. A `closedTableau` for X consists either of a local tableau together with closed tableaux for all its (simple) end nodes, or an application of the modal rule and a closed tableau for the result.

```
inductive localTableau : finset formula → Type
| byLocalRule {X B}
    (_ : localRule X B) (next : Π Y ∈ B, localTableau Y) : localTableau X
| sim {X} : simple X → localTableau X

inductive closedTableau : finset formula → Type
| loc {X} (lt : localTableau X) :
    (Π Y ∈ endNodesOf ⟨X, lt⟩, closedTableau Y) → closedTableau X
| atm {X φ} : ∼□φ ∈ X → simple X →
    closedTableau (projection X ∪ {∼φ}) → closedTableau X
```

Theorem 2.2 (Soundness and Completeness) *Let X be a finite set of formulas. There is a closed tableau for X if and only if X is unsatisfiable.*

For soundness it suffices to show that all rules preserve satisfiability. For the completeness proof (by contraposition) we use open tableaux (i.e. not ending in $\varnothing$) to build a model satisfying X. Note that having one open tableau does not imply that there is no closed tableau, because the modal rule allows a choice between different formulas. For example, in Fig. 1 in the left branch the first application of $\neg\Box$ on the formula $\neg\Box r$ would lead to an open tableau.

The formal completeness proof in Lean is by induction on the size of X. A crucial part of the completeness proof is Lemma 1 from [2]: a simple set is satisfiable iff it is not closed and all its projections are satisfiable.

```
lemma almostCompleteness :
  Π n X, lengthOfSet X = n → consistent X → satisfiable X

lemma Lemma1_simple_sat_iff_all_projections_sat {X : finset formula} :
  simple X → ( satisfiable X ↔
      (¬ closed X ∧ ∀ R, (∼□R) ∈ X → satisfiable (projection X ∪ {∼R})) )
```

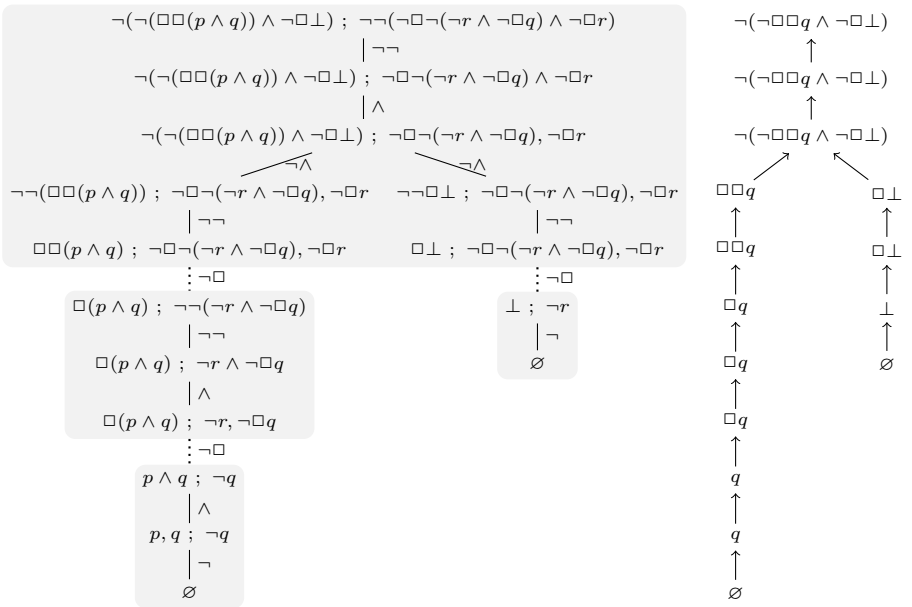

Fig. 1. Example tableau (left) and corresponding tree of interpolants (right).

3 Interpolation via Tableaux

In [12] it was shown that tableaux systems such as the one defined in the previous section can be used to show the following (see also [7, Section 3.8]).

Theorem 3.1 (Craig Interpolation) *For any formulas φ and ψ, if $\varphi \to \psi$ is a tautology, then there exists a formula θ (called* interpolant*) such that $\varphi \to \theta$ and $\theta \to \psi$ are tautologies and $\mathsf{voc}(\theta) \subseteq \mathsf{voc}(\varphi) \cap \mathsf{voc}(\psi)$.*

To show interpolation via tableaux we first provide a more general definition of interpolants for partitions. A *partition* of X is a pair of sets of formulas (X_1, X_2) such that $X = X_1 \cup X_2$. We write $X_1; X_2$ for (X_1, X_2), as in Fig. 1.

Definition 3.2 A formula θ is an *interpolant for a partition $X_1; X_2$* iff $\mathsf{voc}(\theta) \subseteq \mathsf{voc}(X_1) \cap \mathsf{voc}(X_2)$ and both $X_1 \cup \{\neg\theta\}$ and $X_2 \cup \{\theta\}$ are not satisfiable.

This definition is given in Lean as follows:

```
def partInterpolant (X1 X2 : finset formula) (θ : formula) :=
  voc θ ⊆ (voc X1 ∩ voc X2)  ∧  ¬ satisfiable ( X1 ∪ {∼θ} )
                             ∧  ¬ satisfiable ( X2 ∪ { θ} )
```

Lemma 3.3 *A formula $\varphi \to \psi$ is a tautology iff $\{\varphi, \neg\psi\}$ is not satisfiable. Moreover, θ is an interpolant for $\varphi \to \psi$ iff θ is an interpolant for $(\{\varphi\}, \{\neg\psi\})$.*

To prove Theorem 3.1 we first use completeness to get a closed tableau for $\varphi; \neg\psi$ and then use it to define interpolants. While a tableau is built starting at the root, the corresponding tree of interpolants is generated starting at the leaves. Fig. 1 shows how we obtain the interpolant $\Box\Box q \vee \Box\bot$ for Example 2.1.

How interpolants from child nodes are combined to define interpolants for their parent nodes depends on the rule, and the side(s) of the partition where it is applied. A large part of our Lean formalisation is about the choice of interpolants, and showing correctness of the choice. For example, in Fig. 1 we apply $(\neg\wedge)$ on the left side of the partition (`X1`) and take the disjunction of the old interpolants θ_a and θ_b to define the new one. This uses the following.

```
lemma nCoInterpolantX1 {X1 X2 φ ψ θa θb} :
  ∼(φ∧ψ) ∈ X1 →
    partInterpolant (X1 \ {∼(φ∧ψ)} ∪ {∼φ}) (X2 \ {∼(φ∧ψ)}) θa →
      partInterpolant (X1 \ {∼(φ∧ψ)} ∪ {∼ψ}) (X2 \ {∼(φ∧ψ)}) θb →
        partInterpolant X1 X2 (∼(∼θa ∧ ∼θb))
```

Finally, the proof that interpolants as in Definition 3.2 always exist is by a structural induction on the `closedTableaux`, and a second induction on the size of the set of formulas for local tableaux.

4 Future Work

Refactoring and optimization. Our project currently has around 2900 lines of code, including around 1700 unique lines. Checking the proof takes several minutes on an Intel i7 CPU with 4.7 GHz. The next step will thus be to avoid general "expensive" proof tactics such as `finish` and to reduce duplicate code.

Towards PDL. Our formalisation is unnecessarily complicated for basic modal logic. For example, separating `closedTableau` and `localTableau` is

not needed. But, as mentioned above, our long term goal is to move from BML to PDL and formally verify the proof attempt from [2,3]. Hence we use the same definitions, ideas and high-level proof structure. [2] The next intermediate steps will be multi-modal logic and star-free PDL. Another shortcut could be to leave out tests, as PDL has interpolation if test-free PDL has interpolation [7, Theorem 10.6.2], unless tests are needed for the construction given in [2].

Finally, we did not have space here for a comparison to [14], but we also aim to study whether interpolation could be shown formally in their framework.

Acknowledgements For helpful advice I would like to thank A.J. Best, J. Blanchette, R. Brasca, E. Brakkee, K. Buzzard, M. Carneiro, Y. Dillies, P. Johnson, T. Kappé, K. Miller, E. Rodriguez, R. Van de Velde, E. Wieser, others at leanprover.zulipchat.com, and the anonymous AiML reviewers.

References

[1] Bentzen, B., *A Henkin-Style Completeness Proof for the Modal Logic S5*, in: P. Baroni, C. Benzmüller and Y. Wáng, editors, *CLAR 2021*, 2021, pp. 459–467.
URL https://doi.org/10.1007/978-3-030-89391-0_25

[2] Borzechowski, M., *Tableau-Kalkül für PDL und Interpolation* (1988), Diplomarbeit.

[3] Borzechowski, M. and M. Gattinger, *A Proof from 1988 that PDL has Interpolation?* (2020), short talk at Advances in Modal Logic 2020.
URL https://malv.in/2020/borzechowski-pdl/

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

[5] From, A. H., *Epistemic Logic: Completeness of Modal Logics*, Archive of Formal Proofs (2018), https://isa-afp.org/entries/Epistemic_Logic.html.

[6] Galois, L., *A verfied tableau prover for classical propositional logic (in nnf)* (2018).
URL https://github.com/LudvikGalois/coq-CPL-NNF-tableau

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

[8] Li, J., *Formalization of PAL·S5 in Proof Assistant* (2020).
URL https://arxiv.org/abs/2012.09388

[9] Michaelis, J. and T. Nipkow, *Formalized Proof Systems for Propositional Logic*, in: *Types for Proofs and Programs (TYPES 2017)*, 2018, pp. 5:1–5:16.
URL https://doi.org/10.4230/LIPICS.TYPES.2017.5

[10] Moura, L. d., S. Kong, J. Avigad, F. v. Doorn and J. v. Raumer, *The Lean theorem prover (system description)*, in: *CADE-25*, 2015, pp. 378–388.

[11] Neeley, P., *Results in modal and dynamic epistemic logic*, talk at Lean Together 2021.
URL https://youtu.be/kXCB5wzQTKc

[12] Rautenberg, *Modal tableau calculi and interpolation*, Philosophical Logic **12** (1983), pp. 402–423.
URL https://www.jstor.org/stable/30226284

[13] The mathlib Community, *The Lean mathematical library*, in: *CPP*, 2020, pp. 367–381.
URL https://doi.org/10.1145/3372885.3373824

[14] Wu, M. and R. Goré, *Verified Decision Procedures for Modal Logics*, in: J. Harrison, J. O'Leary and A. Tolmach, editors, *ITP 2019*, LIPIcs **141**, 2019, pp. 31:1–31:19.
URL https://doi.org/10.4230/LIPIcs.ITP.2019.31

[2] An exception is that [2] uses model graphs to explicitly build counter models from open tableaux. Our formalisation defines model graphs, but they are not yet used.