Proofs as Coalgebras

MSc Thesis *(Afstudeerscriptie)*

written by

Madeleine Gignoux

under the supervision of **Dr. Malvin Gattinger** and **Prof. Dr. Yde Venema**, and submitted to the Examinations Board in partial fulfillment of the requirements for the degree of

MSc in Logic

at the *Universiteit van Amsterdam.*

Date of the public defense:	**Members of the Thesis Committee:**
May 7th, 2026	Dr. Maria Aloni (chair)
	Dr. Malvin Gattinger (supervisor)
	Dr. Marianna Girlando
	Dr. Tobias Kappé
	Prof. Dr. Yde Venema (supervisor)

Institute for Logic, Language and Computation

Abstract

In this thesis we study ill-founded proof systems from a coalgebraic perspective. We present proofs as coalgebras, where correctness is enforced solely by path conditions on infinite branches, with no requirements of tree-like structure or rootedness. We apply this framework to the Gödel-Löb provability logic (GL) and the alternation-free μ-calculus (AFMC), and pursue three lines of investigation: *finitization*, *interpolation*, and *formalization*. For *finitization*, we establish exponential upper bounds on proof size in terms of closure size for both GL and AFMC. For *interpolation*, we obtain Lyndon interpolation for GL and syntactic Lyndon interpolation for AFMC; for the latter we additionally show that interpolant size is bounded exponentially by closure size. For *formalization*, we present a fully machine-verified proof of Craig interpolation for GL in Lean 4.

Contents

Chapter 1

Introduction

Our first introduction to proof theory, whether it be proof calculi or analytic tableaux, is likely one in which the mathematical object that constitutes a proof, typically a tree or a list, is inductively defined. These objects are easy to work with due to their well-founded structure. However, well-foundedness in proof systems is not inherently necessary, and not every logic has a well-founded proof system. In particular, logics with fixpoints introduce circular reasoning, and proof theorists have used ill-founded proofs to reason about this circularity [NW96; Bro06].

In practice, proofs for these systems are often still tree-like. The standard approaches are (1) using trees with infinite branches to represent proofs, referred to here as *infinite-tree* proofs. Or, (2) using finite trees, but allowing for certain leaves to be identified with an ancestor node labeled by the same sequent, referred to here as *cyclic* proofs. There is a third type of infinite proof, where proofs are finite trees with infinitely many premises, but since we are interested in ill-founded proof systems we do not explore this further.

an *infinite tree* proof a *cyclic* proof

Figure 1.1: The structures of infinite tree proofs and cyclic proofs. Inspired by [Klo26].

At first glance, a cycle or an infinite branch within a proof may seem problematic. In case of a cyclic proof, how can one claim to have proven φ by referring to the same proof of φ? And in case of an infinite tree, how can one verify a branch that never terminates? To answer both questions, one imposes conditions on infinite paths needed for soundness and completeness. These conditions may directly relate to the semantics of the logic, but can often be abstracted via annotations on nodes or formulas [Sti14; Jun10]. These annotations serve as visually verifiable witnesses of proof correctness.

The approach we will take in this thesis is to use path conditions as our sole structural requirement, discarding all underlying well-foundedness or tree-like structure, and viewing proofs as arbitrary graphs subject only to those conditions. Specifically, we will view proofs as coalgebras $(X, \alpha : X \to \mathcal{T}X)$ for some endofunctor $\mathcal{T}$. Taking a coalgebraic approach toward proofs was first explored by B. Sierra Miranda *et al.* in [SSZ24]. This work views proofs coalgebraically as a collection of finite trees stitched together. By contrast, our coalgebras will look at proofs step-by-step: an element $x \in X$ will be a node

of the proof and $\alpha(x)$ will encode the principal formulas, the non-principal formulas, and the rule label, as well as the premises of x.

This approach is deliberately unconstrained: in constructing a proof we have greater freedom in connecting nodes, as we need not maintain a tree-like structure or even rootedness. We test this approach along three axes: *finitization*, *interpolation*, and *formalization*, using coalgebraic proof systems for the Gödel-Löb provability logic (GL) and the alternation-free μ-calculus (AFMC) as case studies.

Transforming infinite mathematical objects into equivalent finite ones arises in model theory, modal logic, and even proof theory. Indeed, to find interpolants or to machine-verify proofs, we need a finite proof to work with, and as such, proof theorists often define transformations from infinite tree proof systems into cyclic ones. In our case, we will show that we can transform infinite coalgebras (X, α) which we deem to be proofs, into finite coalgebras (Y, β) which still meet our path conditions for being a proof. The core difficulty in finitizing proofs will revolve around preserving these infinite path conditions. To do so, we use *filtration*, a coalgebraic concept used to show finitization in modal-logic, and take inspiration from D. Kozen's proof of the finite model property for the modal μ-calculus [Koz88]. We also obtain stronger results by bounding the size of the resulting proofs by closure size rather than formula length, which is often the best-case scenario for cyclic proof systems.

We also want to show interpolation using our finitization results. In cyclic proof theory for fixpoint logics, interpolants are constructed inductively using Maehara's method, with fixpoints taken at non-axiomatic leaves or their companions. Since our proofs lack an underlying inductive structure, we take a different approach. For each of the aforementioned logics, we find interpolants by solving a system of equations, in which we relate each node in a proof to the form of its interpolant [Bor88; Sha14; AL19; MV21]. We will prove the correctness of the interpolants syntactically. Using our finitization results, we will be able to give bounds on the size of our interpolants and the syntactic interpolation results.

Finally, we test the feasibility of this approach in the functional programming language and interactive theorem prover Lean 4. Proof systems wherein proofs are inductively defined are straightforward to formalize in functional programming languages as inductive definitions are central to the underlying theory. However, formalizing ill-founded proofs is not as straightforward. To the best of our knowledge, the only formalization of an ill-founded proof system in Lean 4 to date is the ongoing work formalizing a cyclic proof system for PDL by M. Gattinger *et al.* [Bor+25]. Using our coalgebraic methodology and our approaches toward finitization and interpolation, we formalize a proof of Craig interpolation for GL, and evaluate the coalgebraic approach as a methodology for formalizing ill-founded proof systems.

The structure of this thesis is as follows: in Chapter 3 we introduce our coalgebraic notion of proof. In Chapter 4 we apply this coalgebraic methodology to the case study of GL, specifically for the ill-founded proof systems by D. Shamkanov in [Sha14], to obtain bounded finitization results via filtration in Section 4.1 and Lyndon interpolation in Section 4.2. Similarly, in Chapter 5 we treat the case study of AFMC using the ill-founded proof systems by J. Marti and Y. Venema in [MV21]. In Section 5.1 we achieve bounded finitization of the proof system, and in Section 5.2 we achieve syntactic Lyndon interpolation with an exponential upper bound on interpolant size for guarded formulas. Finally, in Chapter 6 we discuss formalization of the coalgebraic methodology in Lean 4 and present our formalization of Craig interpolation via this method for GL.

The work we do in Lean will be specific to GL but will follow the approach of this thesis directly. We use labels such as "`interpolation` ✓" next to a theorem and "`Proof`" next to a definition to link to the corresponding work in Lean.

Chapter 2

Preliminaries

2.1 Relations

Given a binary relation $R \subseteq X \times X$, we let R^+ and R^* denote the transitive and reflexive-transitive closure of R. We let R^n denote the n-fold relational composition of R, defined inductively by $R^0 := \{(x,x) \mid x \in X\}$ and $R^{n+1} := \{(x,z) \mid \exists y,\, xRy \text{ and } yR^nz\}$. Given $Y \subseteq X$, we let R_Y denote the restriction of R to Y and $R[Y]$ denote $\{x \mid \exists y \in Y, yRx\}$. Given $x, y \in X$, if xRy, we say y is a *premise* of x and if xR^*y we say y is a *successor* of x. We denote the set of all successors of x by $\uparrow x$, i.e. $\uparrow x := \{y \in X \mid xR^*y\}$. A finite path of length n is a finite sequence $(x_i)_{i<n}$ such that for all $i+1 < n$, x_iRx_{i+1}. An infinite R-path is a sequence $(x_i)_{i<\omega}$ such that for all $i < \omega$, x_iRx_{i+1}. A *R-cluster* C of X refers to an equivalence class of the relation

$$x \sim y \iff xR^*y \text{ and } yR^*x.$$

We call a cluster C *trivial* if $C = \{x\}$ for some x such that xR^+x does not hold. Given two clusters C and D we will let $\prec$ denote the relation given by

$$C \prec D \iff C \neq D \text{ and } \exists x \in C, \exists y \in D, xR^+y.$$

It is well known that this relation is well-founded if X is finite.

2.2 Fixpoint Logics

Within this thesis, we will explore proof systems for two fixpoint logics. These logics are the Gödel-Löb provability logic (GL) and the alternation-free μ-calculus (AFMC). To discuss the AFMC, we will first have to discuss the full modal μ-calculus (MC).

2.2.1 The Gödel-Löb Provability Logic

The Gödel-Löb Provability Logic is a modal logic which is the result of research from the 1930s to the 1960s exploring the notion of provability as an operator. At its core is the provability predicate in Peano arithmetic (PA), denoted $\mathrm{Prov}(\ulcorner\varphi\urcorner)$ for which M. Löb proved in [Löb55] that

$$\mathsf{PA} \vdash \mathrm{Prov}(\ulcorner\mathrm{Prov}(\ulcorner\varphi\urcorner) \to \varphi\urcorner) \to \mathrm{Prov}(\ulcorner\varphi\urcorner).$$

Viewing Prov($\ulcorner\cdot\urcorner$) as a modal operator leads us to what is known today as Löb's axiom (L) $\Box(\Box\varphi \to \varphi) \to \Box\varphi$. The logic GL is basic modal logic BML with Löb's axiom added.

Definition 2.1 (`Formula`)**.** The language of the basic modal logic $\mathcal{L}_\Box$ is defined in negation-normal form as follows.

$$\mathcal{L}_\Box \ni \varphi \quad := \quad \bot \mid \top \mid p \mid \overline{p} \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \Diamond\varphi \mid \Box\varphi$$

for $p \in \mathsf{Prop}$.

Definition 2.2 (Axiomatization of GL [Boo94])**.** GL is axiomatized by the following axioms

- All propositional tautologies,
- (K) $\Box(\varphi \to \psi) \to \Box\varphi \to \Box\psi$,
- (L) $\Box(\Box\varphi \to \varphi) \to \Box\varphi$,

and the inference rules

- Modus Ponens: If $\vdash_{\mathsf{GL}} \varphi$ and $\vdash_{\mathsf{GL}} \varphi \to \psi$ then $\vdash_{\mathsf{GL}} \psi$,
- Necessitation: If $\vdash_{\mathsf{GL}} \varphi$ then $\vdash_{\mathsf{GL}} \Box\varphi$,

along with substitution.

It is well known that GL is modally sound and complete with respect to the class of transitive and conversely well-founded Kripke frames [Boo94].

Key Definitions

We provide some key definitions and lemmas here that we will refer to throughout the text.

Definition 2.3 (`single`, `partial_`)**.** Given a formula φ, a set $Y \subseteq \mathsf{Prop}$ and a set of formulas ψ_y for each $y \in Y$ the simultaneous substitution $\varphi[\psi_y/y \mid y \in Y]$ is defined as follows:

$$\begin{aligned}
\delta[\psi_y/y \mid y \in Y] &:= \delta \text{ for } \delta \in \{\bot, \top\} \\
p[\psi_y/y \mid y \in Y] &:= p \text{ for } p \notin Y \\
z[\psi_y/y \mid y \in Y] &:= \psi_z \text{ for } z \in Y \\
\overline{p}[\psi_y/y \mid y \in Y] &:= \overline{p} \text{ for } p \notin Y \\
\overline{z}[\psi_y/y \mid y \in Y] &:= \overline{\psi_z} \text{ for } z \in Y \\
(\varphi \circ \psi)[\psi_y/y \mid y \in Y] &:= \varphi[\psi_y/y \mid y \in Y] \circ \psi[\psi_y/y \mid y \in Y] \text{ for } \circ \in \{\vee, \wedge\} \\
(\triangle\varphi)[\psi_y/y \mid y \in Y] &:= \triangle(\varphi[\psi_y/y \mid y \in Y]) \text{ for } \triangle \in \{\Diamond, \Box\}
\end{aligned}$$

When Y is a singleton, i.e. $Y = \{y\}$, we write this as $\varphi[\psi_y/y]$ or $\varphi(\psi_y)$ when y is understood. We also will denote simultaneous substitutions by maps $\sigma : Y \to \mathcal{L}_\Box$ which extend to maps $\hat{\sigma} : \mathcal{L}_\Box \to \mathcal{L}_\Box$ by

$$\hat{\sigma}(\varphi) := \varphi[\sigma(y)/y \mid y \in Y].$$

For readability, we will remove the hat superscript. Moreover, to avoid confusion, σ will never be used to denote a formula.

Definition 2.4 (`Formula.vocab`, `Sequent.vocab`)**.** Given $\varphi \in \mathcal{L}_\Box$, the vocabulary of φ, denoted $\mathsf{Voc}(\varphi)$, is defined inductively as follows:

$$\begin{array}{lcllcl} \mathsf{Voc}(\bot) & := & \emptyset & \mathsf{Voc}(\top) & := & \emptyset \\ \mathsf{Voc}(p) & := & \{p\} & \mathsf{Voc}(\overline{p}) & := & \{p\} \\ \mathsf{Voc}(\varphi \vee \psi) & := & \mathsf{Voc}(\varphi) \cup \mathsf{Voc}(\psi) & \mathsf{Voc}(\varphi \wedge \psi) & := & \mathsf{Voc}(\varphi) \cup \mathsf{Voc}(\psi) \\ \mathsf{Voc}(\Diamond\varphi) & := & \mathsf{Voc}(\varphi) & \mathsf{Voc}(\Box\varphi) & := & \mathsf{Voc}(\varphi) \end{array}$$

Given a sequent Δ, we define the vocabulary of Δ by $\mathsf{Voc}(\Delta) := \bigcup_{\varphi\in\Delta} \mathsf{Voc}(\varphi)$.

Definition 2.5 (`Formula.lit`, `Sequent.lit`)**.** Given $\varphi \in \mathcal{L}_\Box$, the literals of φ, denoted $\mathsf{Lit}(\varphi)$, are defined inductively as follows:

$$\begin{array}{lcllcl} \mathsf{Lit}(\bot) & := & \emptyset & \mathsf{Lit}(\top) & := & \emptyset \\ \mathsf{Lit}(p) & := & \{p\} & \mathsf{Lit}(\overline{p}) & := & \{\overline{p}\} \\ \mathsf{Lit}(\varphi \vee \psi) & := & \mathsf{Lit}(\varphi) \cup \mathsf{Lit}(\psi) & \mathsf{Lit}(\varphi \wedge \psi) & := & \mathsf{Lit}(\varphi) \cup \mathsf{Lit}(\psi) \\ \mathsf{Lit}(\Diamond\varphi) & := & \mathsf{Lit}(\varphi) & \mathsf{Lit}(\Box\varphi) & := & \mathsf{Lit}(\varphi) \end{array}$$

Given a sequent Δ, we define the literals of Δ by $\mathsf{Lit}(\Delta) := \bigcup_{\varphi\in\Delta} \mathsf{Lit}(\varphi)$.

Definition 2.6 (`Formula.neg`, `Sequent.neg`)**.** Given $\varphi \in \mathcal{L}_\Box$, the negation of φ, denoted $\overline{\varphi}$, is defined inductively as follows:

$$\begin{array}{rcllcl} \overline{\bot} & := & \top & \overline{\top} & := & \bot \\ \overline{p} & := & \overline{p} & \overline{\overline{p}} & := & p \\ \overline{\varphi \vee \psi} & := & \overline{\varphi} \wedge \overline{\psi} & \overline{\varphi \wedge \psi} & := & \overline{\varphi} \vee \overline{\psi} \\ \overline{\Diamond\varphi} & := & \Box\overline{\varphi} & \overline{\Box\varphi} & := & \Diamond\overline{\varphi} \end{array}$$

Given a sequent Δ, we define the negation of Δ by $\overline{\Delta} := \{\overline{\varphi} \mid \varphi \in \Delta\}$.

Definition 2.7 (`Formula.FL`, `Sequent.FL`)**.** Given $\varphi \in \mathcal{L}_\Box$, the (Fischer-Ladner) closure of φ, denoted $\mathrm{Clos}(\varphi)$, is defined inductively as follows:

$$\begin{array}{lcllcl} \mathrm{Clos}(\bot) & := & \{\bot\} & \mathrm{Clos}(\top) & := & \{\top\} \\ \mathrm{Clos}(p) & := & \{p\} & \mathrm{Clos}(\overline{p}) & := & \{\overline{p}\} \\ \mathrm{Clos}(\varphi \vee \psi) & := & \{\varphi \vee \psi\} \cup \mathrm{Clos}(\varphi) \cup \mathrm{Clos}(\psi) & \mathrm{Clos}(\varphi \wedge \psi) & := & \{\varphi \wedge \psi\} \cup \mathrm{Clos}(\varphi) \cup \mathrm{Clos}(\psi) \\ \mathrm{Clos}(\Diamond\varphi) & := & \{\Diamond\varphi\} \cup \mathrm{Clos}(\varphi) & \mathrm{Clos}(\Box\varphi) & := & \{\Box\varphi\} \cup \mathrm{Clos}(\varphi) \end{array}$$

Given a sequent Δ, the closure of Δ is given by $\mathrm{Clos}(\Delta) := \bigcup_{\varphi\in\Delta} \mathrm{Clos}(\varphi)$.

Since $\mathcal{L}_\Box$ does not have explicit fixpoints, $\mathrm{Clos}(\varphi)$ coincides with the subformulas of φ. We nonetheless use the term closure to stay uniform with the μ-calculus, which we will discuss later.

Definition 2.8. Given $\varphi \in \mathcal{L}_\Box$, the (closure) size of φ, denoted $|\varphi|_{\mathrm{Cl}}$, is $|\,\mathrm{Clos}(\varphi)|$ and given a sequent $\Delta \in \mathcal{P}_\omega(\mathcal{L}_\Box)$, the size of Δ, denoted $|\Delta|_{\mathrm{Cl}}$, is $|\,\mathrm{Clos}(\Delta)|$.

Lemma 2.9. *Given $\mathcal{L}_\Box$ formulas φ, ψ and $p \in \mathsf{Prop}$, it holds that $|\varphi[\psi/p]|_{\mathrm{Cl}} \leq |\varphi|_{\mathrm{Cl}} + |\psi|_{\mathrm{Cl}}$.*

Proof. This follows by proving $\mathrm{Clos}(\varphi[\psi/p]) \subseteq \{\chi[\psi/p] \mid \chi \in \mathrm{Clos}(\varphi)\} \cup \mathrm{Clos}(\psi)$ which follows by induction on φ. □

The Fixpoint Theorem

A key theorem about GL that we will use in this thesis is the De Jongh-Sambin Fixpoint Theorem. This is a result proven independently in 1975 by D. de Jongh (unpublished, see [vBe24]) and G. Sambin in 1976 [Sam76]. In general, a fixpoint of a formula $\varphi(q)$ is a formula ψ such that $\psi \equiv \varphi(\psi)$, where $\equiv$ denotes semantic equivalence. For GL, the fixpoint theorem restricts the formulas it considers to formulas $\varphi(p)$ where all occurrences of p in φ are in the scope of a modal operator.

Theorem 2.10 (The De Jongh-Sambin Fixpoint Theorem [vBe24; Sam76; Rei89])**.** *For every formula $\varphi(p)$, such that all occurrences of p in φ are in the scope of a modal operator, there exists a formula ψ such that*

$$\psi \equiv \varphi(\psi)$$

where $\mathsf{Voc}(\psi) \subseteq \mathsf{Voc}(\varphi)$ *and* $p \notin \mathsf{Voc}(\psi)$.

Our work will use a simplified version of this result specific to modal formulas, i.e. formulas of the form $\triangle\varphi(p)$ for $\triangle \in \{\Box, \Diamond\}$. This means that we can drop the condition that all occurrences of p in $\triangle\varphi(p)$ are in the scope of a modal operator, since this is immediate.

Lemma 2.11 (`fixed_point_theorem_modal` ✓)**.** *For every formula $\varphi(p)$ it holds that*

1. $\Box\varphi(\top) \equiv \Box\varphi(\Box\varphi(\top))$,
2. $\Diamond\varphi(\bot) \equiv \Diamond\varphi(\Diamond\varphi(\bot))$.

Proof. This is found by following the construction given by L. Reidhaar-Olson in [Rei89]. □

2.2.2 The Modal μ-Calculus

The modal μ-calculus (MC) was introduced by D. Kozen, building on foundational work of D. Scott and J. de Bakker, to reason about fixpoint properties of programs and transition systems. It is an extension of modal logic with two added operators, μ and ν, which will be interpreted to denote the least and greatest fixpoint of a formula respectively. The language is defined inductively as follows.

$$\mathcal{L}_\mu \ni \varphi \quad := \quad \bot \mid \top \mid p \mid \overline{p} \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \Diamond\varphi \mid \Box\varphi \mid \mu x.\psi \mid \nu x.\psi$$

for $p, x \in \mathsf{Prop}$ with x only occurring positively in ψ.

Key Definitions

There are some standard notions present in the literature on the modal μ-calculus that we do not define in detail. For a full exploration of the modal μ-calculus, we refer the reader to [BS07].

Definition 2.12. Given $\varphi \in \mathcal{L}_\mu$, the *free variables* of φ, denoted $FV(\varphi)$, are defined inductively as follows:

$$\begin{array}{lcllcl}
FV(\bot) & := & \emptyset & FV(\top) & := & \emptyset \\
FV(p) & := & \{p\} & FV(\overline{p}) & := & \{p\} \\
FV(\varphi \vee \psi) & := & FV(\varphi) \cup FV(\psi) & FV(\varphi \wedge \psi) & := & FV(\varphi) \cup FV(\psi) \\
FV(\Diamond\varphi) & := & FV(\varphi) & FV(\Box\varphi) & := & FV(\varphi) \\
FV(\mu x.\varphi) & := & FV(\varphi) \setminus \{x\} & FV(\nu x.\varphi) & := & FV(\varphi) \setminus \{x\}
\end{array}$$

Given a sequent Δ, we define the free variables of Δ by $FV(\Delta) := \bigcup_{\varphi\in\Delta} FV(\varphi)$.

Definition 2.13. Given $\varphi \in \mathcal{L}_\mu$, the *bound variables* of φ, denoted $BV(\varphi)$, are defined inductively as follows:

$$\begin{array}{lcllcl} BV(\bot) & := & \emptyset & BV(\top) & := & \emptyset \\ BV(p) & := & \emptyset & BV(\overline{p}) & := & \emptyset \\ BV(\varphi \vee \psi) & := & BV(\varphi) \cup BV(\psi) & BV(\varphi \wedge \psi) & := & BV(\varphi) \cup BV(\psi) \\ BV(\Diamond\varphi) & := & BV(\varphi) & BV(\Box\varphi) & := & BV(\varphi) \\ BV(\mu x.\varphi) & := & BV(\varphi) \cup \{x\} & BV(\nu x.\varphi) & := & BV(\varphi) \cup \{x\} \end{array}$$

Given a sequent Δ, we define the bound variables of Δ by $BV(\Delta) := \bigcup_{\varphi\in\Delta} BV(\varphi)$.

Definition 2.14. Given $\varphi \in \mathcal{L}_\mu$, the *literals* of φ, denoted $\mathsf{Lit}(\varphi)$, are defined inductively as follows:

$$\begin{array}{lcllcl} \mathsf{Lit}(\bot) & := & \emptyset & \mathsf{Lit}(\top) & := & \emptyset \\ \mathsf{Lit}(p) & := & \{p\} & \mathsf{Lit}(\overline{p}) & := & \{\overline{p}\} \\ \mathsf{Lit}(\varphi \vee \psi) & := & \mathsf{Lit}(\varphi) \cup \mathsf{Lit}(\psi) & \mathsf{Lit}(\varphi \wedge \psi) & := & \mathsf{Lit}(\varphi) \cup \mathsf{Lit}(\psi) \\ \mathsf{Lit}(\Diamond\varphi) & := & \mathsf{Lit}(\varphi) & \mathsf{Lit}(\Box\varphi) & := & \mathsf{Lit}(\varphi) \\ \mathsf{Lit}(\mu x.\varphi) & := & \mathsf{Lit}(\varphi) \setminus \{x\} & \mathsf{Lit}(\nu x.\varphi) & := & \mathsf{Lit}(\varphi) \setminus \{x\} \end{array}$$

Given a sequent Δ, we define the literals of Δ by $\mathsf{Lit}(\Delta) := \bigcup_{\varphi\in\Delta} \mathsf{Lit}(\varphi)$.

Definition 2.15. Given $\varphi \in \mathcal{L}_\mu$, the *negation* of φ is defined by first defining the *boolean dual*, denoted φ^∂, defined inductively as follows:

$$\begin{array}{lcllcl} \bot^\partial & := & \top & \top^\partial & := & \bot \\ p^\partial & := & p & \overline{p}^\partial & := & \overline{p} \\ (\varphi \vee \psi)^\partial & := & \varphi^\partial \wedge \psi^\partial & (\varphi \wedge \psi)^\partial & := & \varphi^\partial \vee \psi^\partial \\ (\Diamond\varphi)^\partial & := & \Box\varphi^\partial & (\Box\varphi)^\partial & := & \Diamond\varphi^\partial \\ (\mu x.\varphi)^\partial & := & \nu x.\varphi^\partial & (\nu x.\varphi)^\partial & := & \mu x.\varphi^\partial \end{array}$$

Then the negation of φ, denoted $\overline{\varphi}$ is defined by $\varphi^\partial[p \rightleftarrows \overline{p} \mid p \in FV(\varphi)]$, i.e. the formula φ^∂ with all free occurrences of p replaced with $\overline{p}$ and vice versa. Given a sequent Δ, we define the negation of Δ by $\overline{\Delta} := \{\overline{\varphi} \mid \varphi \in \Delta\}$.

We say φ is *tidy* if $BV(\varphi) \cap FV(\varphi) = \emptyset$ and *guarded* if every subformula $\eta x.\psi$ of φ is such that all free occurrences of x in ψ are in the scope of a modality. We will assume all formulas are tidy.

To prevent variable capture, we are only allowed to substitute ψ in the place of x in φ when two conditions are met:

1. x only occurs positively in φ,
2. $FV(\varphi) \cap BV(\psi) = \emptyset$.

We say ψ is *free for* x in φ if these conditions are met.

Definition 2.16 (Simultaneous Substitution [KMV20])**.** Given a formula φ, a set $Y \subseteq \mathsf{Prop}$ and a set of formulas ψ_y for each $y \in Y$ such that each ψ_y is free for y in φ, the *simultaneous substitution* $\varphi[\psi_y/y \mid y \in Y]$ is defined inductively as follows:

$$\begin{aligned}
\delta[\psi_y/y \mid y \in Y] &:= \delta \text{ for } \delta \in \{\bot, \top\} \\
p[\psi_y/y \mid y \in Y] &:= p \text{ for } p \notin Y \\
z[\psi_y/y \mid y \in Y] &:= \psi_z \text{ for } z \in Y \\
\overline{p}[\psi_y/y \mid y \in Y] &:= \overline{p} \text{ for } p \notin Y \\
(\varphi \circ \psi)[\psi_y/y \mid y \in Y] &:= \varphi[\psi_y/y \mid y \in Y] \circ \psi[\psi_y/y \mid y \in Y] \text{ for } \circ \in \{\vee, \wedge\} \\
(\triangle\varphi)[\psi_y/y \mid y \in Y] &:= \triangle(\varphi[\psi_y/y \mid y \in Y]) \text{ for } \triangle \in \{\Diamond, \Box\} \\
(\eta x.\varphi)[\psi_y/y \mid y \in Y] &:= \eta x.(\varphi[\psi_y/y \mid y \in Y \setminus \{x\}]) \text{ for } \eta \in \{\mu, \nu\}
\end{aligned}$$

When Y is a singleton, i.e. $Y = \{y\}$, we write this as $\varphi[\psi_y/y]$. We also will denote a simultaneous substitution by $\sigma : Y \to \mathcal{L}_\mu$ which extends to a partial map $\hat{\sigma} : \mathcal{L}_\mu \to \mathcal{L}_\mu$ by

$$\hat{\sigma}(\varphi) := \varphi[\sigma(y)/y \mid y \in Y]$$

for formulas φ such that ψ_y is free for y in φ for all $y \in Y$. For readability, we will remove the hat superscript. Moreover, to avoid confusion, σ will never be used to denote a formula.

Definition 2.17. The *unfolding* of a fixpoint formula $\eta x.\varphi$ for $\eta \in \{\mu, \nu\}$ is defined to be $\varphi[\eta x.\varphi/x]$, denoted $\mathrm{unf}(\eta x.\varphi)$.

Lemma 2.18. *Let $Y \subseteq \mathsf{Prop}$, let $\varphi, \chi, \psi_y \in \mathcal{L}_\mu$ for $y \in Y$, and let $x \in \mathsf{Prop}$ with $x \notin Y$. Suppose x is free in φ but not in any ψ_y, that χ is free for x in φ, and that ψ_y is free for y in $\varphi[\chi/x]$ for each $y \in Y$. Then*

$$\varphi[\chi/x][\psi_y/y \mid y \in Y] = \varphi[\psi_y/y \mid y \in Y][(\chi[\psi_y/y \mid y \in Y])/x].$$

Proof. Follows by induction on φ. □

Lemma 2.19. *Let $Y \subseteq \mathsf{Prop}$, let $\varphi, \psi_y \in \mathcal{L}_\mu$ for $y \in Y$, and suppose ψ_y is free for y in φ and y occurs only positively in φ, for all $y \in Y$. Then*

$$\mathsf{Lit}(\varphi[\psi_y/y \mid y \in Y]) = (\mathsf{Lit}(\varphi) \setminus (Y \cap FV(\varphi))) \cup \bigcup_{y \in Y \cap FV(\varphi)} \mathsf{Lit}(\psi_y).$$

Proof. Follows by induction on φ. □

Definition 2.20. The (Fischer-Ladner) closure of a formula φ, denoted $\mathrm{Clos}(\varphi)$ is as defined in Definition 3.15 of [KMV20]. The closure of a sequent Δ is given by

$$\mathrm{Clos}(\Delta) := \bigcup_{\varphi \in \Delta} \mathrm{Clos}(\varphi).$$

Lemma 2.21. *For all tidy formulas φ, ψ, the following statements hold:*

1. $\mathrm{Clos}(\varphi \circ \psi) = \{\varphi \circ \psi\} \cup \mathrm{Clos}(\varphi) \cup \mathrm{Clos}(\psi)$ *for* $\circ \in \{\vee, \wedge\}$.
2. $\mathrm{Clos}(\triangle\varphi) = \{\triangle\varphi\} \cup \mathrm{Clos}(\varphi)$ *for* $\triangle \in \{\Diamond, \Box\}$.

3. $\mathrm{Clos}(\eta x.\varphi) = \{\eta x.\varphi\} \cup \{\chi[\eta x.\varphi/x] \mid \chi \in \mathrm{Clos}(\varphi)\}$ *for* $\eta \in \{\mu, \nu\}$.

4. *Given* ψ *and* $p \in \mathsf{Prop}$ *such that* $p \in FV(\varphi)$ *and* ψ *is free for* p *in* φ *then*

$$\mathrm{Clos}(\varphi[\psi/p]) = \{\chi[\psi/p] \mid \chi \in \mathrm{Clos}(\varphi)\} \cup \mathrm{Clos}(\psi).$$

Proof. Proposition 3.22 in [KMV20]. □

Lemma 2.22. *For all formulas* $\varphi, \psi, \chi \in \mathcal{L}_\mu$ *and* $p \in \mathsf{Prop}$*, if* $\mathrm{Clos}(\varphi) \subseteq \mathrm{Clos}(\psi)$ *then* $\mathrm{Clos}(\varphi[\chi/p]) \subseteq \mathrm{Clos}(\psi[\chi/p])$.

Proof. Follows from Lemma 2.21. □

Definition 2.23. The (closure) size of a formula φ, denoted $|\varphi|_{\mathrm{Cl}}$, is $|\mathrm{Clos}(\varphi)|$, and the (closure) size of a sequent Δ, denoted $|\Delta|_{\mathrm{Cl}}$ is $|\mathrm{Clos}(\Delta)|$.

Lemma 2.24. *For all* $\varphi, \psi \in \mathcal{L}_\mu$ *and* $p \in \mathsf{Prop}$*,* $|\varphi[\psi/p]|_{\mathrm{Cl}} \leq |\varphi|_{\mathrm{Cl}} + |\psi|_{\mathrm{Cl}}$.

Proof. Proposition 3.32 in [KMV20]. □

Least Ordinals

The notion of least ordinals for the modal μ-calculus formulas will be useful when we use the modal μ-calculus as a meta-language on proofs in Section 5.1.

As semantics are not our primary concern in this proof-theoretic work, the semantics we give here will be for the transfinite μ-calculus, an extension that gives rise to least ordinals. This extension adds the formula $\mu^\alpha x.\varphi$ and $\nu^\alpha x.\varphi$ for each ordinal α and φ positive in x. We interpret formulas on states s in Kripke models $\mathcal{M} = (M, R, V)$ as follows:

$s \vDash \bot$	never	$s \vDash \top$	always
$s \vDash p$	*iff* $s \in V(p)$	$s \vDash \overline{p}$	*iff* $s \notin V(p)$
$s \vDash \varphi \vee \psi$	*iff* $s \vDash \varphi$ or $s \vDash \psi$	$s \vDash \varphi \wedge \psi$	*iff* $s \vDash \varphi$ and $s \vDash \psi$
$s \vDash \Diamond\varphi$	*iff* $t \vDash \varphi$ for some $t \in R[s]$	$s \vDash \Box\varphi$	*iff* $t \vDash \varphi$ for all $t \in R[s]$
$s \vDash \mu^0 x.\varphi$	never	$s \vDash \nu^0 x.\varphi$	always
$s \vDash \mu^{\alpha+1} x.\varphi$	*iff* $s \vDash \varphi[\mu^\alpha x.\varphi/x]$	$s \vDash \nu^{\alpha+1} x.\varphi$	*iff* $s \vDash \varphi[\nu^\alpha x.\varphi/x]$
$s \vDash \mu^\lambda x.\varphi$	*iff* $s \vDash \mu^\beta x.\varphi$ for some $\beta < \lambda$	$s \vDash \nu^\lambda x.\varphi$	*iff* $s \vDash \nu^\beta x.\varphi$ for all $\beta < \lambda$
$s \vDash \mu x.\varphi$	*iff* $s \vDash \mu^\alpha x.\varphi$ for some α	$s \vDash \nu x.\varphi$	*iff* $s \vDash \nu^\alpha x.\varphi$ for all α

where λ is a limit ordinal.

Since ordinals are well-ordered, given a formula $\mu x.\varphi$ such that $s \vDash \mu x.\varphi$, we call the *least ordinal* of $\mu x.\varphi$ at s the least α such that $s \vDash \mu^\alpha x.\varphi$. Given a μ-formula φ (i.e. $\varphi = \mu x.\psi$), we let φ^α denote the formula $\mu^\alpha x.\psi$.

2.2.3 The Alternation-Free μ-Calculus

The alternation-free μ-calculus (AFMC) is a fragment of the modal μ-calculus which restricts the interaction between μ and ν operators in a given formula. We can phrase this as a condition on MC formulas, but because proving that formulas are alternation-free under this restriction is difficult, we use the following mutually-inductive definition of AFMC formulas.

Definition 2.25 (AFMC formulas [MV21])**.** The formulas of the alternation-free μ-calculus are given by the following mutually-inductive definition.

$$\begin{array}{rcl} \mathcal{L}_\mu^{\mathrm{af}} \ni \varphi & := & \bot \mid \top \mid p \mid \overline{p} \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \Diamond\varphi \mid \Box\varphi \mid \mu p.\varphi_{\{p\}}^{\mu} \quad \mid \nu p.\varphi_{\{p\}}^{\nu} \quad \mid \\ \mathcal{N}_Q^{\mu} \ni \varphi & := & \bot \mid \top \mid q \mid \quad \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \Diamond\varphi \mid \Box\varphi \mid \mu p.\varphi_{Q\cup\{p\}}^{\mu} \mid \qquad\qquad \mid \psi \\ \mathcal{N}_Q^{\nu} \ni \varphi & := & \bot \mid \top \mid q \mid \quad \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \Diamond\varphi \mid \Box\varphi \mid \qquad\qquad \mid \nu p.\varphi_{Q\cup\{p\}}^{\nu} \mid \psi \end{array}$$

where $p \in \mathsf{Prop}$, $q \in Q$, $\varphi_P^\eta \in \mathcal{N}_P^\eta$ for $P \subseteq \mathsf{Prop}$, and $\psi \in \mathcal{L}_\mu^{\mathrm{af}}$ such that $FV(\psi) \cap Q = \emptyset$. The set $\mathcal{N}_Q^\mu$ is referred to as the *noetherian segment* over Q, and $\mathcal{N}_Q^\nu$ is referred to as the *co-noetherian segment* over Q.

Key Properties

We define literals (Lit), negation ($\overline{\cdot}$), and closure (Clos) the same way as we do for MC. The properties of these operators that we introduced earlier will also remain. Still, there are some more key properties specific to AFMC formulas that we will now present.

Lemma 2.26. *Let $Q, Q' \subseteq \mathsf{Prop}$ such that $Q \subseteq Q'$. Then for $\eta \in \{\mu, \nu\}$,*

1. $\mathcal{N}_{Q'}^\eta \subseteq \mathcal{N}_Q^\eta$.
2. $\mathcal{N}_\emptyset^\eta = \mathcal{L}_\mu^{\mathrm{af}}$.

Proof. Proposition 2.4 in [MV21]. □

Lemma 2.27. *Let $Q \subseteq \mathsf{Prop}$, $p \in \mathsf{Prop}$ and suppose $\varphi \in \mathcal{N}_{Q\cup\{p\}}^\eta$ and $\psi \in \mathcal{N}_Q^\eta$. The following hold:*

1. *If $p \notin FV(\psi)$ then $\psi \in \mathcal{N}_{Q\cup\{p\}}^\eta$,*
2. *If ψ is free for p in φ then $\varphi[\psi/p] \in \mathcal{N}_Q^\eta$.*

Proof. Proposition 2.5 in [MV21]. □

Lemma 2.28. *Let $Q \subseteq \mathsf{Prop}$. If $\varphi \in \mathcal{L}_\mu^{\mathrm{af}}$ is not constructed using any fixpoint operators and no $q \in Q$ occurs negatively in φ, then $\varphi \in \mathcal{N}_Q^\eta$ for any $\eta \in \{\mu, \nu\}$.*

Proof. Follows from induction on φ. □

Lemma 2.29. *Let $Q, Y \subseteq \mathsf{Prop}$ and suppose $\varphi \in \mathcal{N}_Q^\eta$ and $\psi_y \in \mathcal{L}_\mu^{\mathrm{af}}$ for each $y \in Y$, with ψ_y free for y in φ. If $FV(\psi_y) \cap Q = \emptyset$ for each $y \in Y$ then $\varphi[\psi_y/y \mid y \in Y] \in \mathcal{N}_Q^\eta$.*

Proof. We proceed by induction on φ. We will list the interesting cases. If φ is z for $z \in Y$ then $\varphi[\psi_y/y \mid y \in Y] = \psi_z$ and $\psi_z \in \mathcal{L}_\mu^{\mathrm{af}}$ so $\psi_z \in \mathcal{N}_Q^\eta$ since $FV(\psi_z) \cap Q = \emptyset$. If φ is $\eta p.\chi$ where $\chi \in \mathcal{N}_{Q\cup\{p\}}^\eta$ then $(\eta p.\chi)[\psi_y/y \mid y \in Y] = \eta p.\chi[\psi_y/y \mid y \in Y \setminus \{p\}]$. By the induction hypothesis we have $\chi[\psi_y/y \mid y \in Y \setminus \{p\}] \in \mathcal{N}_{Q\cup\{p\}}^\eta$, so $(\eta p.\chi)[\psi_y/y \mid y \in Y] \in \mathcal{N}_Q^\eta$. Finally if $\varphi \in \mathcal{L}_\mu^{\mathrm{af}}$ and $FV(\varphi) \cap Q = \emptyset$ then by our induction hypothesis $\varphi[\psi_y/y \mid y \in Y] \in \mathcal{L}_\mu^{\mathrm{af}}$ and since $FV(\varphi[\psi_y/y \mid y \in Y]) \cap Q = \emptyset$ we have $\varphi[\psi_y/y \mid y \in Y] \in \mathcal{N}_Q^\eta$. □

Chapter 3

Coalgebras

Coalgebras will be the central mathematical object of this thesis. We will begin by introducing the basics of coalgebras, and then discuss the specific types of coalgebra we will use in this thesis and some useful properties we will refer to later. We will assume basic knowledge of category theory.

Definition 3.1 (Coalgebra)**.** Given a category $\mathbb{C}$ and an endofunctor $\mathcal{T} : \mathbb{C} \to \mathbb{C}$, a $\mathcal{T}$-coalgebra (X, α) is a pair consisting of X in $\mathrm{ob}(\mathbb{C})$ and a morphism $\alpha : X \to \mathcal{T}X$ in $\mathrm{mor}(\mathbb{C})$.

For our purposes from now on, all coalgebras discussed will be for Set endofunctors. Given a $\mathcal{T}$-coalgebra (X, α), we refer to X as the *base set*, and α as the *structure morphism*.

Coalgebras have many nice category theoretic properties and also give rise to the coalgebraic modal logic. See [Ven19; Jac16] for more on coalgebras and category theoretic properties. For this thesis, we will only need one property: filtration. Filtrations will appear as a useful tool for finitization of proofs in the coalgebraic proof systems we discuss. To start, we define the notion of filtration that we will use.

Definition 3.2 (Filtration)**.** Let (X, α) be a $\mathcal{T}$-coalgebra and let $\sim \subseteq X \times X$ be an equivalence relation on X. Fix a set $\hat{X}$ to be the set of representatives of the equivalence classes of X chosen by the choice function $q : X \to \hat{X}$. The filtration of (X, α) for the choice function q is the $\mathcal{T}$-coalgebra $(\hat{X}, \beta)$ where $\beta := \mathcal{T}q \circ \alpha \circ i$ and i is the embedding $\hat{X} \hookrightarrow X$.

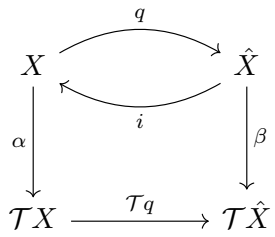

Figure 3.1: Explanatory diagram

Remark 3.3. For readers familiar with category theory and specifically coalgebras, our definition of filtration is different from the standard definition of filtration, which uses equivalence classes rather than their representatives.

Recall that we use coalgebras to define a general notion of a sequent-style proof. From our perspective, proofs will be coalgebras (X, α) where X is the set of nodes in the proof, and α assigns each node its rule label, formulas, and premises. The coalgebras we consider in this thesis will be coalgebras for the following endofunctor.

Definition 3.4. Given a language $\mathcal{L}$ and set of rule labels $\mathcal{R}$ we define the Set-endofunctor $\mathcal{T}^{\mathcal{L}}_{\mathcal{R}}$ as follows:

$$\begin{aligned}\mathrm{ob}(\mathsf{Set}) \ni X &\mapsto \mathcal{R} \times \mathcal{P}_\omega(\mathcal{L}) \times \mathcal{P}_\omega(\mathcal{L}) \times \mathcal{P}_\omega(X)\\ \mathrm{mor}(\mathsf{Set}) \ni f &\mapsto I_{\mathcal{R}} \times I_{\mathcal{P}_\omega(\mathcal{L})} \times I_{\mathcal{P}_\omega(\mathcal{L})} \times \mathcal{P}_\omega(f)\end{aligned}$$

where I_A is the identity morphism for A in Set and $\mathcal{P}_\omega$ is the finite powerset functor.

We will give an example of how this relates to proofs now.

Definition 3.5. Given a $\mathcal{T}^{\mathcal{L}}_{\mathcal{R}}$-coalgebra (X, α) we refer to an element $x \in X$ as a node. Letting $(r^\alpha, f^\alpha_p, f^\alpha_n, p^\alpha)$ denote the respective projections of α (i.e. $\alpha = (r^\alpha, f^\alpha_p, f^\alpha_n, p^\alpha)$), we refer to r^α as the rule, f^α_p as the principal formulas, f^α_n as the non-principal formulas, and p^α as the premises of a node.

We also define $f^\alpha := f^\alpha_n \cup f^\alpha_p$ which we refer to as the sequent of a node. We will drop the α superscript when the context is clear.

Example 3.6. A rule application often found in disjunctive-style sequent calculi is an and rule explaining how to handle formulas of the form $\varphi \wedge \psi$. These rule applications look like the following figure to some degree.

$$\frac{(y)\ \Gamma, \varphi \qquad (z)\ \Gamma, \psi}{(x)\ \Gamma, \varphi \wedge \psi}\ \mathsf{and}$$

Figure 3.2: Example and rule application.

Let us assume this is one step in a larger proof whose information we are not privy to. This is common in proof theory, and from our coalgebraic perspective we still have all the information about the node x that we need. Let us discuss what $\alpha(x)$ should be.

The first, second, and third projection of $\alpha(x)$ into $\mathcal{R}$, $\mathcal{P}_\omega(\mathcal{L})$, and $\mathcal{P}_\omega(\mathcal{L})$ respectively should label the node x with the rule, principal formula, and non-principal formulas. Therefore, this will be $(\mathsf{and}, \{\varphi \wedge \psi\}, \Gamma)$. Lastly, the fourth projection of α into $\mathcal{P}_\omega(X)$ should explain how x fits into the larger proof by connecting it to its premise nodes. In this case $p^\alpha(x)$ would be $\{y, z\}$. Putting this all together, we get $\alpha(x) = (\mathsf{and}, \{\varphi \wedge \psi\}, \Gamma, \{y, z\})$.

There is a natural edge relation on $\mathcal{T}^{\mathcal{L}}_{\mathcal{R}}$-coalgebras given as follows.

Definition 3.7. Given a $\mathcal{T}^{\mathcal{L}}_{\mathcal{R}}$-coalgebra (X, α), let the relation $\lhd^\alpha \subseteq X \times X$ be the relation given by

$$x \lhd^\alpha y \text{ if and only if } y \in p^\alpha(x).$$

We will drop the α superscript when the context is clear.

Of course, the coalgebraic approach does not give us any assertion that rules are applied correctly, meaning that not every coalgebra is a proof.

Definition 3.8. Given a logic with language $\mathcal{L}$ and a rule set $\mathcal{R}$, a $\mathcal{T}^{\mathcal{L}}_{\mathcal{R}}$ (coalgebraic) proof system A is the class of $\mathcal{T}^{\mathcal{L}}_{\mathcal{R}}$-coalgebras (X, α) such that for each $x \in X$ a set of node, step, and path conditions about x are satisfied.

In Chapter 4 and Chapter 5 we will have concrete proof systems to work with. For now, we will denote arbitrary coalgebraic proof systems using the variables A and B.

Example 3.9. Let us return to the example and rule application from Figure 3.2 before and discuss what conditions arise. Node conditions will be conditions about a node itself, therefore they must be conditions about the rule label and the principal and non-principal formulas. For instance, in Figure 3.2, a node condition that arises will be that the principal formula must be a conjunction if the rule label at x is and.

For an example of step conditions, we have to look one step ahead and ask ourselves: *if the rule application is* and*, what do we know about our premises?* Indeed, it tells us that x must have (at most) two premise nodes whose sequents are Γ, φ and Γ, ψ.

Lastly, there are path conditions, which are conditions about infinite paths in a coalgebra (X, α). Unlike node and step conditions, these are conditions not encoded directly in the rule applications of the system. For example, a path condition in ill-founded proof theory that we will encounter later in this thesis is the stipulation that on every infinite path, a modal rule is applied infinitely many times. These will make a small appearance in Chapter 4, but will get discussed in greater detail in Chapter 5.

We define some important terminology for arbitrary proofs, such as what it means for a sequent $\Delta \in \mathcal{P}_\omega(\mathcal{L})$ for some language $\mathcal{L}$ to be provable.

Definition 3.10 (Proves)**.** Given an arbitrary proof system A for the endofunctor $\mathcal{T}_{\mathcal{R}}^{\mathcal{L}}$ and a proof (X, α) in A, we say (X, α) proves a sequent $\Delta \in \mathcal{P}_\omega(\mathcal{L})$ in A if there exists $x \in X$ such that $f(x) = \Delta$. In shorthand we write $(X, \alpha) \vdash_{\mathsf{A}} \Delta$. Moreover, we say Δ is provable (notation: $\vdash_{\mathsf{A}} \Delta$) if there exists a proof (X, α) of Δ.

We are interested in finitizing proofs, i.e. given a proof of a sequent Δ, finding a proof of Δ such that the underlying set of the coalgebra is finite. Let us make clear what we mean by a finite proof.

Definition 3.11. A proof (X, α) in system A is finite if X is finite.

Our goal will be as follows: given an A-proof of a sequent Δ, we want to show that a finite proof of Δ exists. To approach finitization, we start by showing a weaker result for which we need the following definition.

Definition 3.12. A proof (X, α) in system A is f-finite if $f[X]$ is finite.

This is indeed a weaker result since every finite proof is f-finite. However, proving this intermediate result will help us prove the stronger result we desire. This approach will be explored with the case studies of coalgebraic proof systems for GL and AFMC in Section 4.1 and Section 5.1 respectively.

Given a proof (X, α) of a sequent Δ witnessed by x in system A, we will find f-finite proofs by restricting our focus to the part of the proof used in proving Δ, or in other words, we will only look at successors of x. This will help us to achieve f-finiteness as we will see in practice in Theorem 4.7 and Theorem 5.8.

Definition 3.13. Given a $\mathcal{T}_{\mathcal{R}}^{\mathcal{L}}$-coalgebra (X, α) and a node $x \in X$, let $\alpha_x : \uparrow x \to \mathcal{T}_{\mathcal{R}}^{\mathcal{L}}(\uparrow x)$ (where $\uparrow x := \{y \mid x \lhd^* y\}$) be the map given by $y \mapsto \alpha(y)$. We call the $\mathcal{T}_{\mathcal{R}}^{\mathcal{L}}$-coalgebra $(\uparrow x, \alpha_x)$ the *point-generated coalgebra* around x.

We note that the map $\alpha_x : \uparrow x \to \mathcal{T}^{\mathcal{L}}_{\mathcal{R}}(\uparrow x)$ given by $y \mapsto \alpha(y)$ is well-defined since if $z \in p(y)$ for $y \in \uparrow x$ then $z \in \uparrow x$.

Of course, showing that given a proof (X, α) in system A and a node $x \in X$, the point-generated coalgebra around x is still a proof, and is in fact f-finite, is not guaranteed. This is something that must be verified individually for each system.

3.1 Sets of Equations

In this thesis, we will define interpolants by solving a system of equations. We include this section here to clarify what we mean by a system of equations and a solution, and refer back to it later.

The standard method for finding interpolants in a well-founded proof system is Maehara's method [Mae61]. In short, given a proof of $\neg\varphi, \psi$ for which we are trying to find interpolants, the sequent at the root is split into the partition $\neg\varphi \mid \psi$, and these partitions are defined for all nodes in the proof upward. Next, via downward induction on the proof structure: one assumes an interpolant exists for each premise node and defines the interpolant from there. In cyclic proof systems, where there is an underlying well-founded tree structure, this process becomes trickier and requires care when handling back edges.

We aim to find interpolants by solving a system of equations. To begin with, we assume that we can assign for each node x in a coalgebra (X, α) a fresh atomic variable q_x. We will let Λ_X denote the set of all free variables for the nodes in X (i.e. $\Lambda_X := \{q_x \mid x \in X\}$).

Definition 3.14 (Set of Equations)**.** A set of equations on a coalgebra (X, α) is a function $\chi : X \to \mathcal{L}$.

For shorthand and clarity we will always write $\chi(x)$ as χ_x. The terminology "set" refers to the fact that we view the set of equations χ as the set $\{q_x \approx \chi_x \mid x \in X\}$ where $\approx$ is a meaningless symbol. As an example, given the rule application in Figure 3.3, we view q_x as a variable standing for the interpolant for x, and the equation for the interpolant for this rule application would be $q_x \approx q_y \vee q_z$ (i.e. $x \mapsto q_y \vee q_z$). In a well-founded system, we would define the interpolant at each node x, denoted ι_x directly by the shape of the rule, setting $\iota_x = \iota_y \vee \iota_z$, but in ill-founded proof systems repeats prevent this: the interpolant at a node may depend on itself, so we settle for $\iota_x \equiv \iota_y \vee \iota_z$. Rather than imposing an explicit priority for the order in which we solve these equations, we instead prove that a solution exists by constructing partial solutions for each $Y \subseteq X$. We now clarify what we mean by a solution.

Definition 3.15. Given a coalgebra (X, α) and a set of equations $\chi : X \to \mathcal{L}$, a solution to χ is a substitution $\sigma : \Lambda_X \to \mathcal{L}$ such that the following conditions are met:

1. For all $x \in X$, $\sigma(q_x) \equiv \sigma(\chi_x)$ (where $\equiv$ denotes semantic equivalence),
2. For all $x \in X$, $FV(\sigma(q_x)) \cap \Lambda_X = \emptyset$.

3.2 Partial Proofs

A standard notion in (cyclic and non-cyclic) sequent calculi is that of a partial proof. This is a "snippet" of a proof for which some assumptions are left open.

$$\frac{(z)\ \Gamma, \varphi \mid \Delta \qquad (y)\ \Gamma, \psi \mid \Delta}{(x)\ \Gamma, \varphi \wedge \psi \mid \Delta}\ \mathsf{and} \qquad\qquad \frac{\dfrac{\dfrac{\Gamma, \varphi \mid \iota_y}{\Gamma, \varphi \mid \iota_y, \iota_z}\ \mathsf{wk} \quad \dfrac{\Gamma, \psi \mid \iota_z}{\Gamma, \psi \mid \iota_y, \iota_z}\ \mathsf{wk}}{\Gamma, \varphi \wedge \psi \mid \iota_y, \iota_z}\ \mathsf{and}}{\Gamma, \varphi \wedge \psi \mid \iota_y \vee \iota_z}\ \mathsf{or}$$

Figure 3.3: Interpolant correctness partial proof (right) for and rule application (left).

In practice, in well-founded proof systems these snippets are not always explicitly connected. Instead, by induction on the proof structure, one assumes interpolants exist for the premises and constructs each partial proof downward, constructing the entire proof by induction.

Our approach does not have structural induction to benefit from. Yet, induction is not necessary: as seen in Figure 3.3, we can define each snippet separately and later reconnect them. This is exactly the approach we take. These snippets will be part of a proof system with an added rule as allowing for non-axiomatic leaf nodes called assumptions.

Definition 3.16. Given a rule set $\mathcal{R}$, we define $\mathcal{R}_{\mathsf{as}}$ to be $\mathcal{R} \cup \{\mathsf{as}\}$.

Definition 3.17. Let A be an arbitrary proof system. An A_{par}-proof is a $\mathcal{T}^{\mathcal{L}}_{\mathcal{R}_{\mathsf{as}}}$-coalgebra (X, α) such that the following node, step, and path conditions are met for each $x \in X$.

(node) One of the following conditions hold:

1. $(r \times f_p)(x)$ is of the form $(\mathsf{as}, \emptyset)$,
2. The node conditions of A hold for x.

(step) The following conditions hold:

1. if $(r \times f_p)(x) = (\mathsf{as}, \emptyset)$ then $p(x) = \emptyset$,
2. The step conditions of A hold for x.

(path) The path conditions of A hold at x.

Our goal is to combine a collection of partial proofs to eliminate assumptions. We will accomplish this by connecting the open assumptions of the partial proofs to nodes in other partial proofs. To do so, we will replace the rule as with the rule skip as follows.

$$\frac{\Gamma}{\Gamma}\ \mathsf{skip}$$

Figure 3.4: The skip rule application.

Remark 3.18. We will add the aforementioned skip rule shown in Figure 3.4 to an arbitrary proof system. The main advantage of skip is that it provides a simple means to connect partial proofs. There are pros and cons to adding this rule to an arbitrary proof system. It is locally sound, and a complete proof system extended with the skip rule remains complete. However, for soundness we must add a path condition to ensure that skip is not applied cofinitely many times on an infinite path, as illustrated in Figure 3.5 below. The rule is clearly admissible, and if it cannot be used cofinitely often, it is also eliminable.

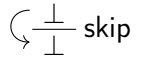

Figure 3.5: A dangerous "proof" using skip.

Definition 3.19. Given a set of rules $\mathcal{R}$, define $\mathcal{R}_{\mathsf{skip}}$ by $\mathcal{R} \cup \{\mathsf{skip}\}$.

Definition 3.20. Let A be an arbitrary proof system for the endofunctor $\mathcal{T}^{\mathcal{L}}_{\mathcal{R}}$. A $\mathcal{T}^{\mathcal{L}}_{\mathcal{R}_{\mathsf{skip}}}$-coalgebra (X, α) is an $\mathsf{A}_{\mathsf{skip}}$-proof if for each $x \in X$ the following node, step, and path conditions are met.

(node) One of the following conditions hold:

1. $(r \times f_p)(x) = (\mathsf{skip}, \emptyset)$,
2. The node conditions of A hold for x.

(step) The following conditions hold:

1. if $(r \times f_p)(x) = (\mathsf{skip}, \emptyset)$ then $f[p(x)] = f(x)$ and $|p(x)| = 1$,
2. The step conditions of A hold for x.

(path) The following conditions hold:

1. On every infinite $\lhd^{\alpha}$-path from x, skip does not occur cofinitely many times,
2. The path conditions of A hold for x.

We now explain how to assemble a collection of partial proofs into a $\mathcal{T}^{\mathcal{L}}_{\mathcal{R}_{\mathsf{skip}}}$-coalgebra. To do so, we first need a compatibility condition on the collection that ensures that there exists a way to connect these partial proofs.

Definition 3.21. A collection of partial $\mathcal{T}^{\mathcal{L}}_{\mathcal{R}}$-coalgebras $\{(P_i, \beta_i)\}_{i \in \mathcal{I}}$, for some index set $\mathcal{I}$, is called *structured* if

- For all $i \in \mathcal{I}$ there is a node $p_i \in P_i$ which we refer to as the root node.
- For all $i \in \mathcal{I}$ and $z \in P_i$, if $r^{\beta_i}(z) = \mathsf{as}$ then there is a chosen $j \in \mathcal{I}$ such that $f^{\beta_i}(z) = f^{\beta_j}(p_j)$.

Definition 3.22 (Proof Transformation)**.** Given a structured collection of partial $\mathcal{T}^{\mathcal{L}}_{\mathcal{R}}$-coalgebras $\{P_i\}_{i \in \mathcal{I}}$ for some arbitrary index $\mathcal{I}$, we define the proof transformation of this collection to be the $\mathcal{T}^{\mathcal{L}}_{\mathcal{R}_{\mathsf{skip}}}$-coalgebra (Y, β) defined as follows.

$$Y := \biguplus_{i \in \mathcal{I}} P_i = \{(i, z) \mid i \in \mathcal{I}, z \in P_i\}$$

$$\beta(i, z) := \begin{cases} (\mathsf{skip}, f_p^{\beta_i}(z), f_n^{\beta_i}(z), \{(j, p_j)\}) & \text{if } r^{\beta_i}(z) = \mathsf{as}, \\ (r^{\beta_i}(z), f_p^{\beta_i}(z), f_n^{\beta_i}(z), \{i\} \times p^{\beta_i}(z)) & \text{else.} \end{cases}$$

where p_j denotes the root of (P_j, β_j) and j is the index guaranteed by Definition 3.21.

Chapter 4

The Gödel-Löb Provability Logic

We want to test the feasibility of our coalgebraic notion of proof as a generalization of what it means to be a proof. To do so, we unify existing cyclic and ill-founded proof systems via our coalgebraic definition of proof, and test two processes central to proof theory: finitizing proofs, and proving the logic has interpolation via the proof system.

The first logic we will explore is GL. For a full introduction to the logic GL, we recommend [Boo94], but for our purposes, we will rely on the preliminaries outlined in Section 2.2 and some easily derivable properties. We formalized a majority of the results for GL from this section in the interactive theorem prover and functional programming language Lean 4. The formalization we developed as part of this thesis is discussed in Chapter 6.

There are multiple sequent calculi that are sound and complete with respect to GL. Perhaps the most well-known is the system GLSC ("GL Sequent Calculus") introduced by G. Sambin and S. Valentini in a two-sided form in [SV80]. A single sequent form of the aforementioned system called $\mathsf{GLS}_{\mathrm{seq}}$ is presented by D. Shamkanov in [Sha14] and is shown in Figure 4.1.

$$\frac{}{\top, \Gamma}\ \mathsf{top} \qquad \frac{}{p, \overline{p}, \Gamma}\ \mathsf{ax} \qquad \frac{\varphi, \psi, \Gamma}{\varphi \vee \psi, \Gamma}\ \mathsf{or} \qquad \frac{\varphi, \Gamma \qquad \psi, \Gamma}{\varphi \wedge \psi, \Gamma}\ \mathsf{and} \qquad \frac{\varphi, \Diamond\overline{\varphi}, \Diamond\Gamma, \Gamma}{\Box\varphi, \Diamond\Gamma, \Sigma}\ \mathsf{box}^*$$

Figure 4.1: Rule applications of the system $\mathsf{GLS}_{\mathrm{seq}}$.

Proofs in this system are finite trees, but, as noted by D. Shamkanov, there is seemingly no known method for finding Lyndon interpolants from proofs in this system due to the negation in the box^* rule application [Sha14], although it is well known that GL has Lyndon interpolation [Sha11]. To remedy this, D. Shamkanov augmented the aforementioned system $\mathsf{GLS}_{\mathrm{seq}}$ into a circular and ill-founded system in [Sha14] by replacing the rule box^* in Figure 4.1 with the rule box as follows.

$$\frac{}{\top, \Gamma}\ \mathsf{top} \qquad \frac{}{p, \overline{p}, \Gamma}\ \mathsf{ax} \qquad \frac{\varphi, \psi, \Gamma}{\varphi \vee \psi, \Gamma}\ \mathsf{or} \qquad \frac{\varphi, \Gamma \qquad \psi, \Gamma}{\varphi \wedge \psi, \Gamma}\ \mathsf{and} \qquad \frac{\varphi, \Diamond\Gamma, \Gamma}{\Box\varphi, \Diamond\Gamma, \Sigma}\ \mathsf{box}$$

Figure 4.2: Rule applications of the system $\mathsf{GL}_{\mathrm{inf}}$ and $\mathsf{GL}_{\mathrm{cyc}}$.

For those familiar with proof theory, the rule box as shown above is locally sound for logics which satisfy the (**4**) axiom (logics with transitive Kripke frames such as GL [BRV01]).

Definition 4.1 (`RuleApp`)**.** We define the rule set $\mathcal{R}_{\mathsf{GL}} = \{\mathsf{top}, \mathsf{ax}, \mathsf{or}, \mathsf{and}, \mathsf{box}\}$.

The notion of being a proof will now become a condition on $\mathcal{T}^{\mathcal{L}_\Box}_{\mathcal{R}_{\mathsf{GL}}}$-coalgebras (see Definition 3.4). In particular, we make all implicit node and step conditions represented in Figure 4.2, explicit. This leads to our following definition of proof.

Definition 4.2 (`Proof`)**.** A $\mathcal{T}^{\mathcal{L}_\Box}_{\mathcal{R}_{\mathsf{GL}}}$-coalgebra (X, α) is called a $\mathsf{G}^{\mathrm{gen}}$-proof if

(node) for all $x \in X$, $(r \times f_p)(x)$ is of one of the following forms:

1. $(\mathsf{top}, \{\top\})$ 2. $(\mathsf{ax}, \{q, \overline{q}\})$ 3. $(\mathsf{or}, \{\varphi \vee \psi\})$ 4. $(\mathsf{and}, \{\varphi \wedge \psi\})$ 5. $(\mathsf{box}, \{\Box\varphi\})$

where $q \in \mathsf{Prop}$ and $\varphi, \psi \in \mathcal{L}_\Box$.

(step) for all $x \in X$, the following statements hold:

1. If $f_p(x) = \{\top\}$ then $p(x) = \emptyset$.
2. If $f_p(x) = \{q, \overline{q}\}$ then $p(x) = \emptyset$.
3. If $f_p(x) = \{\varphi \vee \psi\}$ then $|p(x)| = 1$ and $f[p(x)] = \{(f_n(x) \setminus f_p(x)) \cup \{\varphi, \psi\}\}$.
4. If $f_p(x) = \{\varphi \wedge \psi\}$ then $|p(x)| \leq 2$ and
$$f[p(x)] = \big\{\{(f_n(x) \setminus f_p(x)) \cup \{\varphi\}\}, \{(f_n(x) \setminus f_p(x)) \cup \{\psi\}\}\big\}.$$
5. If $f_p(x) = \{\Box\varphi\}$ then $|p(x)| = 1$ and $f[p(x)] = \{\Box_4^{-1}(f_n(x)) \cup \{\varphi\}\}$ where
$$\Box_4^{-1}(\Theta) = \{\varphi \mid \Diamond\varphi \in \Theta\} \cup \{\Diamond\varphi \mid \Diamond\varphi \in \Theta\}.$$

In the case of $\mathsf{G}^{\mathrm{gen}}$-proofs, we can completely forget the rule projection r, by looking at f_p to determine which rule is applied. As a result, we could completely remove the rule set $\mathcal{R}_{\mathsf{GL}}$ from our endofunctor. This is not possible in the future proof systems we discuss, so for the sake of uniformity, we leave it here. We now justify why this system, which is a generalized form of D. Shamkanov's two systems $\mathsf{GL}_{\mathrm{cyc}}$ and $\mathsf{GL}_{\mathrm{inf}}$, is modally sound and complete.

Theorem 4.3 (`soundness` ✓, `completeness` ✓)**.** *For all sequents* Δ*,*

$$\vdash_{\mathsf{G}^{\mathrm{gen}}} \Delta \text{ if and only if } \vDash \bigvee \Delta$$

where $\vdash_{\mathsf{G}^{\mathrm{gen}}}$ *is as defined in Definition 3.10 and* $\vDash$ *denotes semantic validity on transitive and converse well-founded Kripke models.*

Proof. The work in [Sha14] establishes soundness and completeness for the ill-founded proof system $\mathsf{GL}_{\mathrm{inf}}$ by transforming proofs in $\mathsf{GL}_{\mathrm{inf}}$ to and from proofs in $\mathsf{GLS}_{\mathrm{seq}}$ using ultrametric spaces. Soundness and completeness then come as a corollary since $\mathsf{GLS}_{\mathrm{seq}}$ is sound and complete. Soundness and completeness for the coalgebraic notion of proof $\mathsf{G}^{\mathrm{gen}}$ can be seen by observing that every $\mathsf{GL}_{\mathrm{inf}}$-proof is a $\mathsf{G}^{\mathrm{gen}}$-proof, and every $\mathsf{G}^{\mathrm{gen}}$-proof can be turned into a $\mathsf{GL}_{\mathrm{inf}}$-proof via unraveling, giving our desired result. □

However, we want to avoid this detour via the $\mathsf{GLS}_{\mathrm{seq}}$ system and ultrametric spaces in our formalization work in Lean. For soundness, we will use the same reasoning as the soundness proof by G. Menéndez Turata in [Men24, Prop. 2.2.8], and for completeness we will prove the result ourselves by taking a game-theoretic approach. This will be discussed in more detail in Chapter 6.

A necessary property of the system, which is used for proving soundness, is the stipulation that on every infinite $\lhd$-path $(x_i)_{i<\omega}$ there are infinitely many i such that $r(x_i) = \mathsf{box}$. This is an example of a path condition, a condition about infinite paths within the proof, although for $\mathsf{G}^{\mathrm{gen}}$-proofs, this property is implicit.

Lemma 4.4 (`inf_path_has_inf_boxes` ✓)**.** *Given a $\mathsf{G}^{\mathrm{gen}}$-proof (X, α), on every infinite $\lhd$-path $(x_i)_{i<\omega}$, there are infinitely many i such that $r(x_i) = \mathsf{box}$.*

Proof. Let $(x_i)_{i<\omega}$ be an infinite $\lhd$-path. Suppose to the opposite that there are only finitely many $i < \omega$ such that $r(x_i) = \mathsf{box}$. Take the maximal such i, let this be n. Then observe that in all rule applications shown in Figure 4.2, the size of the premises of a sequent is smaller than the size of the sequent itself unless the rule application is box. Thus $|f(x_{m+1})|_{\mathrm{Cl}} < |f(x_m)|_{\mathrm{Cl}}$ for all $m \geq n$, a contradiction. □

4.1 Finitization

We want to show that our system $\mathsf{G}^{\mathrm{gen}}$ exhibits a *finite proof property* in the sense that we can turn a $\mathsf{G}^{\mathrm{gen}}$-proof of a sequent Δ into a $\mathsf{G}^{\mathrm{gen}}$-proof of Δ with a finite number of nodes. Even better, we will show a stronger version of this property, a *small proof property*, where we give bounds on the size of the resultant finite proofs. This will be necessary in future sections since we will need finiteness for our interpolation results, and the bound on the size of the proof will give us a bound on the size of our interpolants.

4.1.1 Point-Generated Proofs

To show that every proof of a sequent Δ can be turned into an f-finite proof of Δ, we begin by relating our proofs to something well-known to be finite: the (Fischer-Ladner) closure of Δ.

A common property of many cut-free sequent calculi is that all formulas are contained in the closure of the root node. Of course, in our coalgebraic setting we have no underlying tree structure and therefore no root of our proofs. However, for similar reasons we do have the following nice property about finite paths in our proofs.

Lemma 4.5 (`path_in_FL` ✓)**.** *Let (X, α) be a $\mathsf{G}^{\mathrm{gen}}$-proof and $x, y \in X$. If $x \lhd^* y$ then $f(y) \subseteq \mathrm{Clos}(f(x))$.*

Proof. Fix a $\mathsf{G}^{\mathrm{gen}}$-proof (X, α) and $x, y \in X$ such that $x \lhd^* y$. Since $x \lhd^* y$, we know there is some n such that $x \lhd^n y$. We proceed by induction on n. If $n = 0$ then clearly $f(x) \subseteq \mathrm{Clos}(f(x))$. If $n = k + 1$ then there is some $z \in X$ such that $x \lhd^k z$ and $z \lhd y$. We know by our induction hypothesis that $f(z) \subseteq \mathrm{Clos}(f(x))$ and since Clos is monotone and idempotent, we have $\mathrm{Clos}(f(z)) \subseteq \mathrm{Clos}(f(x))$. To get our desired result it suffices to show $f(y) \subseteq \mathrm{Clos}(f(z))$. We do so by checking cases on $r(z)$.

1. If $r(z)$ is top we have a contradiction since $y \in p(z)$, and by (step) 1. we know if $r(z)$ is top then $p(z) = \emptyset$.
2. If $r(z)$ is ax we have the same contradiction as in (1).
3. If $r(z)$ is or then there are $\varphi, \psi \in \mathcal{L}_\Box$ such that $f_p(z) = \{\varphi \vee \psi\}$ and by (step) 3. we know that $f(y) = (f_n(z) \setminus \{\varphi \vee \psi\}) \cup \{\varphi, \psi\}$ and recalling that $f(z) = f_p(z) \cup f_n(z)$ by definition of f, we know that $f(z) = f_n(z) \cup \{\varphi \vee \psi\}$. All that remains is to show that $(f_n(z) \setminus \{\varphi \vee \psi\}) \cup \{\varphi, \psi\} \subseteq \mathrm{Clos}(f_n(z) \cup \{\varphi \vee \psi\})$ which follows from the definition of Clos.

4. If $r(z)$ is and then by an analogous argument to (3) we get our desired result.

5. If $r(z)$ is box then there is $\varphi \in \mathcal{L}_\Box$ such that $f_p(z) = \{\Box\varphi\}$ and by (step) 5. we know that $f(y) = \Box_4^{-1}(f_n(z)) \cup \{\varphi\}$ where we recall that $\Box_4^{-1}(\Delta) = \{\Diamond\varphi \mid \Diamond\varphi \in \Delta\} \cup \{\varphi \mid \Diamond\varphi \in \Delta\}$. Thus it remains to show that $\Box_4^{-1}(f_n(z)) \cup \{\varphi\} \subseteq \mathrm{Clos}(f_n(z) \cup \{\Box\varphi\})$ which follows from the definition of Clos. □

As seen in the above lemma, by only looking forward in a proof, we have a nice property that relates to the Fischer-Ladner closure. Likewise, our notion of proof only looks forward in the sense that the *node* conditions can be checked on the node itself regardless of the edge relation induced by α, and the *step* conditions are checked by looking forward one step in the proof, hence the name. Therefore, intuitively fixing a point x in an arbitrary $\mathsf{G}^{\mathrm{gen}}$-proof, the point-generated coalgebra around x should still be a proof. This is what we will assert now for $\mathsf{G}^{\mathrm{gen}}$-proofs.

Lemma 4.6 (`pointGeneratedProof` ✓)**.** *Given a $\mathsf{G}^{\mathrm{gen}}$-proof (X, α) and some node $x \in X$, the point-generated $\mathcal{T}^{\mathcal{L}_\Box}_{\mathcal{R}_{\mathsf{GL}}}$-coalgebra around x (see Definition 3.13) given by $(\uparrow x, \alpha_x)$ is a $\mathsf{G}^{\mathrm{gen}}$-proof.*

Proof. We must check that the (node) and (step) conditions given in Definition 4.2 are satisfied. To begin with, observe that for all $y \in \uparrow x$ it holds that $\alpha_x(y) = \alpha(y)$.

(node) Take $y \in \uparrow x$. Since $(r^{\alpha_x} \times f_p^{\alpha_x})(y) = (r^\alpha \times f_p^\alpha)(y)$ and (X, α) is a $\mathsf{G}^{\mathrm{gen}}$-proof, one of the forms listed in the (node) conditions of Definition 4.2 will be satisfied.

(step) Take $y \in \uparrow x$. Since $f_p^{\alpha_x}(z) = f_p^\alpha(z)$ for all $z \in \uparrow x$ and (X, α) satisfies (step) in Definition 4.2, we have our desired result. □

We end this section with its main theorem.

Theorem 4.7. *For any sequent Δ, if there is a $\mathsf{G}^{\mathrm{gen}}$-proof (X, α) of Δ then there is an f-finite proof of Δ where $|f[X]|$ is bounded by $2^{|\Delta|_{\mathrm{Cl}}}$.*

Proof. Let (X, α) be a $\mathsf{G}^{\mathrm{gen}}$-proof of Δ witnessed by x. Take the point-generated coalgebra around x given by $(\uparrow x, \alpha_x)$. By Lemma 4.6 this is a $\mathsf{G}^{\mathrm{gen}}$-proof. Moreover, by definition of α_x (see Definition 3.13) $f^\alpha(y) = f^{\alpha_x}(y)$ for all $y \in \uparrow x$, so we have that $(\uparrow x, \alpha_x)$ is a $\mathsf{G}^{\mathrm{gen}}$-proof of Δ (witnessed by x). Finally, take $y \in \uparrow x$, then by Lemma 4.5, $f^\alpha(y) \subseteq \mathrm{Clos}(\Delta)$, so $f^{\alpha_x}[\uparrow x] \subseteq \mathcal{P}_\omega(\mathrm{Clos}(\Delta))$ thus $|f^{\alpha_x}[\uparrow x]| \leq 2^{|\Delta|_{\mathrm{Cl}}}$. □

4.1.2 Filtration

The ultimate goal of this subsection will be to show that we can transform any f-finite proof of a sequent Δ into a finite proof of Δ. The method we will employ will use filtrations of coalgebras as defined in Definition 3.2.

Definition 4.8 (`f_eq_equi_rel`)**.** Given a $\mathsf{G}^{\mathrm{gen}}$-proof (X, α) define the relation $\sim^\alpha$ on X by

$$x \sim^\alpha y \text{ if and only if } f^\alpha(x) = f^\alpha(y).$$

The relation $\sim^\alpha$ for any proof (X, α) is clearly an equivalence relation. We will drop the α superscript when α is clear from the context.

Lemma 4.9 (`filtration` ✓)**.** *If $(\hat{X}, \hat{\alpha})$ is a filtration of a G^{gen}-proof (X, α) through the equivalence relation $\sim^{\alpha}$ then $(\hat{X}, \hat{\alpha})$ is a G^{gen}-proof.*

Proof. Recall from Definition 3.2 that $\hat{X}$ is the set of representatives of the equivalence classes of X determined by the choice function $q : X \to \hat{X}$ and $\beta := (\mathcal{T}^{\mathcal{L}_\Box}_{\mathcal{R}_{\mathsf{GL}}} q) \circ \alpha \circ i$ where $i : \hat{X} \to X$ is the embedding.

We check the (node) and (step) conditions given in Definition 4.2. To begin with, observe that for all $y \in \hat{X}$,

$$(r^\beta(y), f^\beta_p(y), f^\beta_n(y), p^\beta(y)) = (r^\alpha(y), f^\alpha_p(y), f^\alpha_n(y), q[p^\alpha(y)]). \tag{4.1}$$

(node) Take $y \in \hat{X}$. Since $(r^\beta \times f^\beta_p)(y) = (r^\alpha \times f^\alpha_p)(y)$ and (X, α) is a G^{gen}-proof, one of the forms listed in the (node) conditions of Definition 4.2 will be satisfied.

(step) Take $y \in \hat{X}$. To begin with, we show that for any $y \in \hat{X}$,

$$f^\beta[p^\beta(y)] = f^\alpha[p^\alpha(y)]. \tag{4.2}$$

Rewriting the left-hand side using (4.1) gives us $f^\beta[p^\beta(y)] = f^\alpha[q[p^\alpha(y)]] = (f^\alpha \circ q)[p^\alpha(y)]$. Since $f^\alpha(x) = f^\alpha \circ q(x)$ for all $x \in X$, we get $f^\beta[p^\beta(y)] = f^\alpha[p^\alpha(y)]$. We now check conditions (1)-(5).

1. If $f^\beta_p(y) = \{\top\}$, then $f^\alpha_p(y) = \{\top\}$ and $p^\alpha(y) = \emptyset$, thus $p^\beta(y) = q[p^\alpha(y)] = \emptyset$.
2. Follows analogously from (1)
3. If $f^\beta_p(y) = \{\varphi \vee \psi\}$ for $\varphi, \psi \in \mathcal{L}_\Box$, then $f^\alpha_p(y) = \{\varphi \vee \psi\}$ and $f^\alpha[p^\alpha(y)] = \{(f^\alpha_n(y) \setminus f^\alpha_p(y)) \cup \{\varphi, \psi\}\}$. Thus by (4.1) and (4.2) we have $f^\beta[p^\beta(y)] = \{(f^\beta_n(y) \setminus f^\beta_p(y)) \cup \{\varphi, \psi\}\}$. Finally, $|p^\alpha(y)| = 1$ so $|q[p^\alpha(y)]| = 1$.
4. If $f^\beta_p(y) = \{\varphi \wedge \psi\}$ then by the same reasoning as (3) we know $f^\beta[p^\beta(y)] = \{(f^\beta_n(y) \setminus f^\beta_p(y)) \cup \{\varphi\}, (f^\beta_n(y) \setminus f^\beta_p(y)) \cup \{\psi\}\}$. Moreover $|p^\alpha(y)| \leq 2$ so $|q[p^\alpha(y)]| \leq 2$.
5. Follows analogously from (3). □

Theorem 4.10. *For all sequents Δ, if there is a f-finite proof (X, α) of Δ then there is a finite proof (Y, β) of Δ such that $|Y| = |f^\beta[Y]|$.*

Proof. Let (X, α) be a f-finite proof of Δ, i.e. $|f[X]|$ is finite. Let $(\hat{X}, \hat{\alpha})$ be a filtration of (X, α) through the equivalence relation $\sim^\alpha$. Observe there is a bijection $\hat{X} \to f^{\hat{\alpha}}[\hat{X}]$ given by $y \mapsto f^{\hat{\alpha}}(y)$. This map is injective since if two representatives y_1 and y_2 of different equivalence classes satisfy $f^{\hat{\alpha}}(y_1) = f^{\hat{\alpha}}(y_2)$, then $f^\alpha(y_1) = f^\alpha(y_2)$ so $y_1 \sim^\alpha y_2$, a contradiction. Thus $|\hat{X}| = |f^{\hat{\alpha}}[\hat{X}]|$. Finally, let x witness the proof of Δ in (X, α). Let $\hat{x}$ denote the representative of the equivalence class x is in (i.e. $\hat{x} = q(x)$). Then $f^{\hat{\alpha}}(\hat{x}) = f^\alpha(x)$ and $f^\alpha(x) = \Delta$ so $(\hat{X}, \hat{\alpha})$ proves Δ, witnessed by $\hat{x}$. □

We can now combine Theorem 4.7 and Theorem 4.10 to prove the main result of the section.

Theorem 4.11. *For all sequents Δ, if there is a G^{gen}-proof of Δ then there is a finite proof of Δ whose size is bounded by $2^{|\Delta|_{\text{CI}}}$.*

Proof. This follows from Theorem 4.7 and Theorem 4.10. □

4.2 Interpolation

4.2.1 The Split Proof System

In his work in [Sha14], D. Shamkanov introduces a cyclic split sequent variant of his system $\mathsf{GL}_{\mathrm{cyc}}$, and shows a transformation from $\mathsf{GL}_{\mathrm{cyc}}$-proofs to the cyclic split sequent system. He then defines interpolants inductively on the structure of proofs in this split sequent system and proves the correctness of these interpolants semantically.

To approach interpolation from a coalgebraic perspective we start by defining a coalgebraic notion of proof mimicking the split sequent system given by D. Shamkanov. Formulas in the split sequent system will occur on the left- or right-hand side of a split sequent. For this reason, elements of the split sequent will be elements of $s\mathcal{L}_\Box := \mathcal{L}_\Box \times \{l, r\}$, where l denotes *left* and r denotes *right*. For shorthand, we let φ_l denote (φ, l) and φ_r denote (φ, r). Moreover, for $i \in \{l, r\}$ and $\Delta \in \mathcal{P}_\omega(\mathcal{L}_\Box)$, we define $\Delta^i := \{\varphi \mid (\varphi, i) \in \Delta\}$. We may denote a split sequent Δ using the notation $\Delta^l \mid \Delta^r$.

$$\frac{}{\top_i, \Gamma}\ \mathsf{top} \qquad \frac{}{p_i, \overline{p}_j, \Gamma}\ \mathsf{ax} \qquad \frac{\varphi_i, \psi_i, \Gamma}{(\varphi \vee \psi)_i, \Gamma}\ \mathsf{or} \qquad \frac{\varphi_i, \Gamma \qquad \psi_i, \Gamma}{(\varphi \wedge \psi)_i, \Gamma}\ \mathsf{and} \qquad \frac{\varphi_i, \Diamond\Gamma, \Gamma}{(\Box\varphi)_i, \Diamond\Gamma, \Sigma}\ \mathsf{box}$$

Figure 4.3: Rule applications of the $\mathsf{G}^{\mathrm{split}}$ system.

This is not the last time we will annotate formulas φ with extra information. To keep notation clean, given an annotated sequent Δ of formulas annotated with extra information, let the operators $\mathsf{Lit}(\Delta)$, $\mathrm{Clos}(\Delta)$, and $\overline{\Delta}$ denote the literals, closure, and negation of the underlying formulas in the sequent Δ respectively.

Making these conditions explicit we get the following notion of proof.

Definition 4.12 (`Split.Proof`)**.** A $\mathcal{T}^{s\mathcal{L}_\Box}_{\mathcal{R}_{\mathsf{GL}}}$-coalgebra (X, α) is called a $\mathsf{G}^{\mathrm{split}}$-proof if the following statements hold:

(node) for all $x \in X$, $(r \times f_p)(x)$ is of one of the following forms:

1. $(\mathsf{top}, \{\top_i\})$ 2. $(\mathsf{ax}, \{q_i, \overline{q}_j\})$ 3. $(\mathsf{or}, \{(\varphi \vee \psi)_i\})$ 4. $(\mathsf{and}, \{(\varphi \wedge \psi)_i\})$ 5. $(\mathsf{box}, \{(\Box\varphi)_i\})$

where $i, j \in \{l, r\}$, $q \in \mathsf{Prop}$ and $\varphi, \psi \in \mathcal{L}_\Box$.

(step) For all $x \in X$, the following statements hold:

1. If $f_p(x) = \{\top_i\}$ then $p(x) = \emptyset$.
2. If $f_p(x) = \{q_i, \overline{q}_j\}$ then $p(x) = \emptyset$.
3. If $f_p(x) = \{(\varphi \vee \psi)_i\}$ then $|p(x)| = 1$ and $f[p(x)] = \{(f_n(x) \setminus f_p(x)) \cup \{\varphi_i, \psi_i\}\}$.
4. If $f_p(x) = \{(\varphi \wedge \psi)_i\}$ then $|p(x)| \leq 2$ and
$$f[p(x)] = \big\{\{(f_n(x) \setminus f_p(x)) \cup \{\varphi_i\}\}, \{(f_n(x) \setminus f_p(x)) \cup \{\psi_i\}\}\big\}.$$
5. If $f_p(x) = \{(\Box\varphi)_i\}$ then $|p(x)| = 1$ and $f[p(x)] = \{\Box^{-1}_{4,\mathrm{split}}(f_n(x)) \cup \{\varphi_i\}\}$ where
$$\Box^{-1}_{4,\mathrm{split}}(\Theta) = \{(\varphi, i) \mid (\Diamond\varphi, i) \in \Theta\} \cup \{(\Diamond\varphi, i) \mid (\Diamond\varphi, i) \in \Theta\}.$$

We should note that the same finitization result about $\mathsf{G}^{\mathrm{gen}}$-proofs applies to $\mathsf{G}^{\mathrm{split}}$-proofs using the same technique of point generation and filtration.

Theorem 4.13 (`Split.finite_proof_of_proof` ✓). *Given a* $\mathsf{G}^{\text{split}}$*-proof* (X, α) *of* Δ*, there exists a finite* $\mathsf{G}^{\text{split}}$*-proof* (Y, β) *of* Δ *such that* $|Y| \leq 2^{|\Delta|_{\text{Cl}}+1}$.

Proof. Following the same method of finitization for G^{gen}-proofs in Section 4.1, we get our result by the same method of proving f-finitization via point generation and finitization via filtration of f-finite proofs using the same equivalence relation. We get a bound of $2^{|\Delta|_{\text{Cl}}+1}$ rather than $2^{|\Delta|_{\text{Cl}}}$ since we have to account for formulas showing up on both the left- and right-hand side of a split sequent. □

Moreover, this system is modally sound and complete by the same argument as for the G^{gen}-proof system.

Theorem 4.14 (`Split.soundness` ✓, `Split.completeness` ✓). *For all split sequents* Δ, Γ*,*

$$\vdash_{\mathsf{G}^{\text{split}}} \Delta \mid \Gamma \text{ if and only if } \vDash \bigvee(\Delta \cup \Gamma)$$

where $\vdash_{\mathsf{G}^{\text{split}}}$ *is as defined in Definition 3.10 and* $\vDash$ *denotes semantic validity on transitive and converse well-founded Kripke models.*

Proof. Soundness follows from an equivalent argument to the one given by G. Menéndez Turata in [Men24, Prop. 2.2.8]. Completeness for the $\mathsf{G}^{\text{split}}$ system follows from the game-theoretic approach we discuss in Section 6.3. □

The implicit infinite path condition stated in Lemma 4.4 for G^{gen}-proofs is also true for $\mathsf{G}^{\text{split}}$-proofs, and will be useful later.

Lemma 4.15 (`Split.inf_path_has_inf_boxes` ✓). *Given a* $\mathsf{G}^{\text{split}}$*-proof* (X, α)*, on every infinite* $\lhd$*-path* $(x_i)_{i<\omega}$*, there are infinitely many* i *such that* $r(x_i) = \mathsf{box}$*.*

Proof. Follows from Lemma 4.4. □

We will need the following property about literals in $\mathsf{G}^{\text{split}}$-proofs to prove that the interpolants we compute satisfy the vocabulary condition for Lyndon interpolants.

Lemma 4.16. *Given a* $\mathsf{G}^{\text{split}}$*-proof* (X, α) *and* $x, y \in X$ *such that* $x \lhd^* y$*,*

$$\mathsf{Lit}(\overline{f(y)^l}) \cap \mathsf{Lit}(f(y)^r) \subseteq \mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)$$

Proof. Let (X, α) and x, y be as specified. Suppose $q \in \mathsf{Lit}(\overline{f(y)^l}) \cap \mathsf{Lit}(f(y)^r)$ and proceed by induction on $x \lhd^* y$. If $x = y$ we immediately have our result. So suppose $x \lhd z$ and $z \lhd^* y$. By the induction hypothesis we know $q \in \mathsf{Lit}(\overline{f(z)^l}) \cap \mathsf{Lit}(f(z)^r)$ so it remains to show that $q \in \mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)$. This can be seen by checking cases on $(r \times f_p)(x)$ as this determines $f(z)$. From the (step) conditions it can be observed that no new formulas are introduced when making a forward step within the proof and the rule applications of the $\mathsf{G}^{\text{split}}$-proof system do not negate any formulas, giving us our desired result. □

4.2.2 Finding Interpolants

The standard technique for finding interpolants from a sequent calculus is to use the well-founded tree-like structure of the proof system to define interpolants from the leaves downward. This means

special care has to be taken in cyclic proof systems to handle the repeats. However, as previously noted, the edge relation induced by our proof system is not necessarily well-founded. We will follow the approach towards interpolation laid out in Section 3.1.

Definition 4.17 (`encodeVar`)**.** Let (X, α) be a finite $\mathsf{G}^{\text{split}}$-proof. For every $x \in X$, let q_x be a unique atom such that q_x does not appear in $\bigcup_{x \in X} FV(f(x))$ (i.e. q_x does not appear in any of the sequents of (X, α)).

We note that since (X, α) is finite, there will only be a finite number of atoms in the entirety of the proof, so our set of free variables is well-defined.

Definition 4.18 (`equation`)**.** Given a $\mathsf{G}^{\text{split}}$-proof (X, α) we define a set of equations for interpolation (see Definition 3.14) $\chi : X \to \mathcal{L}_\Box$ such that χ is defined as follows

1. if $f_p(x) = \{\top_l\}$ then $\chi_x := \bot$;
2. if $f_p(x) = \{\top_r\}$ then $\chi_x := \top$;
3. if $f_p(x) = \{q_l, \overline{q}_l\}$ then $\chi_x := \bot$;
4. if $f_p(x) = \{q_l, \overline{q}_r\}$ then $\chi_x := \overline{q}$;
5. if $f_p(x) = \{q_r, \overline{q}_l\}$ then $\chi_x := q$;
6. if $f_p(x) = \{q_r, \overline{q}_r\}$ then $\chi_x := \top$;
7. if $f_p(x) = \{(\varphi \vee \psi)_l\}$ then $\chi_x := q_y$ where y is the single element in $p(x)$;
8. if $f_p(x) = \{(\varphi \vee \psi)_r\}$ then $\chi_x := q_y$ where y is the single element in $p(x)$;
9. if $f_p(x) = \{(\varphi \wedge \psi)_l\}$ then $\chi_x := \bigvee_{y \in p(x)} q_y$;
10. if $f_p(x) = \{(\varphi \wedge \psi)_r\}$ then $\chi_x := \bigwedge_{y \in p(x)} q_y$;
11. if $f_p(x) = \{(\Box\varphi)_l\}$ then $\chi_x := \Diamond q_y$ where y is the single element in $p(x)$;
12. if $f_p(x) = \{(\Box\varphi)_r\}$ then $\chi_x := \Box q_y$ where y is the single element in $p(x)$.

We should note that χ is well-defined since all of the aforementioned cases are unique and cover all possibilities for $f_p(x)$ in a $\mathsf{G}^{\text{split}}$-proof.

Lemma 4.19 (`var_in_equation` ✓)**.** *Let (X, α) be a $\mathsf{G}^{\text{split}}$-proof and take $x \in X$ and $q \in \mathsf{Lit}$. If $q \in \mathsf{Lit}(\chi_x)$ then one of the following statements holds:*

1. $q \in \mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)$,
2. $q = q_y$ *for some* $y \in X$ *such that* $x \lhd y$.

Proof. Let (X, α) be a $\mathsf{G}^{\text{split}}$-proof and let $x \in X$ and $q \in \mathsf{Prop}$ be as specified. By taking cases on $(r \times f_p)(x)$ we get our desired result from the definition of χ and f. □

Our approach to interpolation is to view an interpolant for a node x in a G^{gen}-proof (X, α) as a solution σ to the set of equations $\{q_x \approx \chi_x \mid x \in X\}$ applied to q_x. In fact, Lyndon interpolants require a stronger vocabulary condition, which says

$$\mathsf{Lit}(\sigma(q_x)) \subseteq \mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)$$

By solving a set of equations in the form defined in Definition 4.18, we find the interpolant not for just one node but for all nodes in a finite proof. Our method to prove that such a solution exists is to prove the following stronger claim which enforces the stronger vocabulary condition as well as a size condition.

Lemma 4.20 (`interpolant_strong_prop` ✓)**.** *Given a finite* $\mathsf{G}^{\text{split}}$*-proof* (X, α) *and a set of equations for interpolation* χ *as defined above, for every* $Y \subseteq X$ *there is a map* $\sigma_Y : \Lambda_Y \to \mathcal{L}_\Box$ *such that for all* $x \in Y$*, the following conditions are met.*

1. *One of the following statements hold:*
   (a) $\sigma_Y(q_x) = \sigma_Y(\chi_x)$,
   (b) $\sigma_Y(q_x) \equiv \sigma_Y(\chi_x)$, $r(x) = \mathsf{box}$, *and* $x \lhd_Y^+ x$.
2. *For all* $z \in X$*, if* $q_z \in \mathsf{Voc}(\sigma_Y(q_x))$ *then* $x \lhd^+ z$.
3. $\mathsf{Lit}(\sigma_Y(q_x)) \subseteq (\mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)) \cup \Lambda_{X \setminus Y}$.
4. $|\sigma_Y(q_x)|_{\text{Cl}} \leq 3 \times 2^{|Y|}$.

Proof. We will prove the existence of σ_Y by induction on the size of Y. Proving that this substitution satisfies conditions (1) through (4) requires many small checks, and we provide the full proof in the appendix in Lemma A.1.

The base case is immediate. Suppose $Y \neq \emptyset$, we inspect the structure of $(Y, \lhd_Y)$ where $\lhd_Y$ is the restriction of $\lhd$ to Y. We distinguish cases depending on whether there is a node $y \in Y$ such that $y \lhd_Y^+ y$, i.e. whether there is a cycle within Y.

- Suppose that no loops occur in Y. Then since the graph $(Y, \lhd_Y)$ is finite there must exist a leaf y with respect to $\lhd_Y$. Let $Z = Y \setminus \{y\}$. Then, by the induction hypothesis we know that there must be $\sigma_Z : \Lambda_Z \to \mathcal{L}_\Box$ with the properties listed above. We define $\sigma_Y : \Lambda_Y \to \mathcal{L}_\Box$ by

$$\sigma_Y(q_x) = \sigma_Z(q_x)[\chi_y / q_y].$$

- Now suppose that a loop does occur in Y. By Lemma 4.15 there must be a box node on this cycle, let it be y. Then $\chi_y = \triangle q_{y'}$ where y' is the unique premise of y and $\triangle \in \{\Box, \Diamond\}$. Let $Z = Y \setminus \{y\}$, by the induction hypothesis we know that there must be $\sigma_Z : \Lambda_Z \to \mathcal{L}_\Box$ satisfying all of the properties specified above. Let $\delta \in \{\top, \bot\}$ be $\top$ if $\triangle = \Box$ and $\bot$ if $\triangle = \Diamond$. Recall from Lemma 2.11 that we know $\sigma_Z(\chi_y)[(\sigma_Z(\chi_y)[\delta/q_y])/q_y] \equiv \sigma_Z(\chi_y)[\delta/q_y]$. We define $\sigma_Y : \Lambda_Y \to \mathcal{L}_\Box$ by

$$\sigma_Y(q_x) = \sigma_Z(q_x)[(\sigma_Z(\chi_y)[\delta/q_y])/q_y]. \qquad \Box$$

The following result is a corollary of the above lemma.

Theorem 4.21 (`interpolant_prop` ✓)**.** *Given a finite* $\mathsf{G}^{\text{split}}$*-proof* (X, α) *and a set of equations for interpolation* χ *as defined above, there is a map* $\sigma : \Lambda_X \to \mathcal{L}_\Box$ *such that for all* $x \in X$*, the following conditions are met.*

1. *One of the following hold:*
   (a) $\sigma(q_x) = \sigma(\chi_x)$,
   (b) $\sigma(q_x) \equiv \sigma(\chi_x)$ *and* $r(x) = \mathsf{box}$.
2. $\mathsf{Lit}(\sigma(q_x)) \subseteq \mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)$.
3. $|\sigma(q_x)|_{\text{Cl}} \leq 3 \times 2^{|X|}$.

Proof. Follows from Lemma 4.20. □

Definition 4.22 (`interpolant`)**.** Given a finite $\mathsf{G}^{\text{split}}$-proof (X, α), choose a solution σ to a set of equations for interpolation as guaranteed by Theorem 4.21. We define the *interpolant* for a node $x \in X$, denoted ι_x, to be $\sigma(q_x)$.

4.2.3 Partial Interpolation Proofs

In the last section we showed how to take a finite $\mathsf{G}^{\text{split}}$-proof (X, α) and find formulas ι_x for each $x \in X$ which we referred to as the *interpolants*. We want to show that we are justified in calling these formulas interpolants by proving the split sequents $f(x)^l \mid \iota_x$ and $\overline{\iota_x} \mid f(x)^r$ for each $x \in X$. The work by D. Shamkanov in [Sha14] proves this semantically using the underlying well-founded structure of GL_{cyc}. We do not have this well-foundedness to fall back on, but we will show that it is not necessary. The proof method we will employ will also be different.

Instead of a semantic argument, we will create proofs of the aforementioned sequents $f(x)^l \mid \iota_x$ and $\overline{\iota_x} \mid f(x)^r$ in a coalgebraic proof system by creating partial proofs and combining them. The proof system we base these partial proofs on will be an extension of D. Shamkanov's split sequent system with an added cut and weakening rule.

$$\frac{}{\top_i, \Gamma}\,\mathsf{top} \qquad \frac{}{p_i, \overline{p}_j, \Gamma}\,\mathsf{ax} \qquad \frac{\varphi_i, \psi_i, \Gamma}{(\varphi \vee \psi)_i, \Gamma}\,\mathsf{or} \qquad \frac{\varphi_i, \Gamma \qquad \psi_i, \Gamma}{(\varphi \wedge \psi)_i, \Gamma}\,\mathsf{and} \qquad \frac{\varphi_i, \Diamond\Gamma, \Gamma}{(\Box\varphi)_i, \Diamond\Gamma, \Sigma}\,\mathsf{box}$$

$$\frac{\Gamma}{\varphi_i, \Gamma}\,\mathsf{wk} \qquad \frac{\Gamma^l \mid \varphi \qquad \overline{\varphi} \mid \Gamma^r}{\Gamma}\,\mathsf{cut}$$

Figure 4.4: Rule applications for the G^{ext} system.

The set of rules for our extended system will therefore be given by $\mathcal{R}_{\text{ext}} := \mathcal{R}_{\mathsf{GL}} \cup \{\mathsf{wk}, \mathsf{cut}\}$.

Definition 4.23. A $\mathcal{T}^{s\mathcal{L}_\Box}_{\mathcal{R}_{\text{ext}}}$-coalgebra (X, α) is a G^{ext}-proof if for all $x \in X$, the following conditions hold

(node) One of the following holds

1. $(r \times f_p)(x) = (\mathsf{wk}, \{\varphi_i\})$,
2. $(r \times f_p)(x) = (\mathsf{cut}, \{\varphi_i\})$,
3. The (node) conditions of $\mathsf{G}^{\text{split}}$ hold for x.

(step) The following conditions hold

1. If $(r \times f_p)(x) = (\mathsf{wk}, \{\varphi_i\})$ then $f[p(x)] = \{f_n(x)\}$ and $|p(x)| = 1$.
2. If $(r \times f_p)(x) = (\mathsf{cut}, \{\varphi_i\})$ then

$$f[p(x)] = \{(f(x)^l \times \{l\}) \cup \{\varphi_r\}, (f(x)^r \times \{r\}) \cup \{\overline{\varphi}_l\}\} \text{ and } |p(x)| \leq 2.$$

(path) On every infinite $\lhd$-path $(x_i)_{i<\omega}$ from x there are infinitely many i such that $r(x_i) = \mathsf{box}$.

Some systems require the use of *analytic* cut to prove interpolation. We do not have this restriction since we only use the extended split system to prove the correctness of interpolants. However, our use of cut in our interpolation correctness proofs will be analytic and restricted to the use of cut to cut fixpoint formulas of the very specific form given in Lemma 2.11.

Lemma 4.24 (`partialInterpolationLeft`, `partialInterpolationRight`)**.** *Given a* $\mathsf{G}^{\mathrm{split}}$*-proof* (X, α) *and* $x \in X$ *there exists* $\mathsf{G}^{\mathrm{ext}}_{\mathrm{par}}$*-proofs* (L_x, β_x) *and* (R_x, γ_x) *such that the following properties hold:*

1. *There exists a node* $l_x \in L_x$ $(r_x \in R_x)$ *such that* $f^{\beta_x}(l_x) = f(x)^l \mid \iota_x$ $(f^{\gamma_x}(r_x) = \overline{\iota_x} \mid f(x)^r)$*.*
2. *For all* $z \in L_x$ $(z \in R_x)$*, if* $r(z) = \mathsf{as}$ *then there exists* $y \in p^\alpha(x)$ *such that* $f^{\beta_x}(z) = f^\alpha(y)^l \mid \iota_y$ $(f^{\gamma_x}(z) = \overline{\iota_y} \mid f^\alpha(y)^r)$*.*
3. *If* $r^\alpha(x) = \mathsf{box}$ *then* (L_x, β_x) $((R_x, \gamma_x))$ *has a* box *rule application on every path from* l_x (r_x) *to a non-axiomatic leaf.*

Proof. The interesting cases are when x is a box node, and $\iota_x \neq \triangle\iota_y$ where y is the unique premise of x. Table 4.1 covers the other cases for the choice of χ made clear by the "χ_x" column of the table. The three aforementioned properties can be checked via a direct inspection.

The principal formula occurs on the left side of the proof. If the principal formula occurs on the left side of the proof then $\iota_x \equiv \Diamond\iota_y$ and $f_p(x) = \Box\varphi_l$ for some formula φ by Theorem 4.21. By completeness of the split system, this means that there exists a $\mathsf{G}^{\mathrm{split}}$-proof $\mathbb{D}_1$ of $\overline{\Diamond\iota_y} \mid \iota_x$ and a $\mathsf{G}^{\mathrm{split}}$-proof $\mathbb{D}_2$ of $\overline{\iota_x} \mid \Diamond\iota_y$. The $\mathsf{G}^{\mathrm{ext}}_{\mathrm{par}}$-proofs (L_x, β_x) and (R_x, γ_x) are given as follows:

$$\dfrac{\dfrac{\dfrac{\varphi, \Box_4^{-1}(\Gamma^l) \mid \iota_y}{\varphi, \Box_4^{-1}(\Gamma^l) \mid \Diamond\iota_y, \iota_y}\,\mathsf{wk}}{\Box\varphi, \Gamma^l \mid \Diamond\iota_y}\,\mathsf{box} \qquad \overline{\Diamond\iota_y} \mid \iota_x}{\Box\varphi, \Gamma^l \mid \iota_x}\,\mathsf{cut} \qquad \dfrac{\dfrac{\overline{\iota_y} \mid \Box_4^{-1}\Gamma^r}{\Box\overline{\iota_y} \mid \Gamma^r}\,\mathsf{box} \qquad \overline{\iota_x} \mid \Diamond\iota_y}{\overline{\iota_x} \mid \Gamma^r}\,\mathsf{cut}$$

The principal formula occurs on the right side of the proof. If the principal formula occurs on the right side of the proof then $\iota_x \equiv \Box\iota_y$ and $f_p(x) = \Box\varphi_r$ by Theorem 4.21. By completeness of the split system, this means that there exists a $\mathsf{G}^{\mathrm{split}}$-proof $\mathbb{D}_1$ of $\overline{\Box\iota_y} \mid \iota_x$ and a $\mathsf{G}^{\mathrm{split}}$-proof $\mathbb{D}_2$ of $\overline{\iota_x} \mid \Box\iota_y$. The $\mathsf{G}^{\mathrm{ext}}_{\mathrm{par}}$-proofs (L_x, β_x) and (R_x, γ_x) are given as follows:

$$\dfrac{\dfrac{\Box_4^{-1}(\Gamma^l) \mid \iota_y}{\Gamma^l \mid \Box\iota_y}\,\mathsf{box} \qquad \overline{\Box\iota_y} \mid \iota_x}{\Gamma^l \mid \iota_x}\,\mathsf{cut} \qquad \dfrac{\dfrac{\dfrac{\overline{\iota_y} \mid \varphi, \Box_4^{-1}\Gamma^r}{\overline{\iota_y}, \Diamond\overline{\iota_y} \mid \varphi, \Box_4^{-1}\Gamma^r}\,\mathsf{wk}}{\Diamond\overline{\iota_y} \mid \Box\varphi, \Gamma^r}\,\mathsf{box} \qquad \overline{\iota_x} \mid \Box\iota_y}{\overline{\iota_x} \mid \Box\varphi, \Gamma^r}\,\mathsf{cut}$$

□

Representation of $\alpha(x)$	χ_x	Left Proof (L_x, β_x)	Right Proof (R_x, γ_x)
$\top, \Gamma^l \mid \Gamma^r$	$\chi_x = \bot$	$\top, \Gamma^l \mid \iota_x$	$\overline{\iota_x} \mid \Gamma^r$
$\Gamma^l \mid \top, \Gamma^r$	$\chi_x = \top$	$\Gamma^l \mid \iota_x$	$\overline{\iota_x} \mid \top, \Gamma^r$
$q, \overline{q}, \Gamma^l \mid \Gamma^r$	$\chi_x = \bot$	$q, \overline{q}, \Gamma^l \mid \iota_x$	$\overline{\iota_x} \mid \Gamma^r$
$q, \Gamma^l \mid \overline{q}, \Gamma^r$	$\chi_x = \overline{q}$	$q, \Gamma^l \mid \iota_x$	$\overline{\iota_x} \mid \overline{q}, \Gamma^r$
$\overline{q}, \Gamma^l \mid q, \Gamma^r$	$\chi_x = q$	$\overline{q}, \Gamma^l \mid \iota_x$	$\overline{\iota_x} \mid q, \Gamma^r$
$\Gamma^l \mid q, \overline{q}, \Gamma^r$	$\chi_x = \top$	$\Gamma^l \mid \iota_x$	$\overline{\iota_x} \mid q, \overline{q}, \Gamma^r$
$\dfrac{\varphi, \psi, \Gamma^l \mid \Gamma^r}{\varphi \vee \psi, \Gamma^l \mid \Gamma^r}\,\mathsf{or}$	$\chi_x = \chi_y$	$\dfrac{\varphi, \psi, \Gamma^l \mid \iota_y}{\varphi \vee \psi, \Gamma^l \mid \iota_x}\,\mathsf{or}$	$\overline{\iota_x} \mid \Gamma^r$
$\dfrac{\Gamma^l \mid \varphi, \psi, \Gamma^r}{\Gamma^l \mid \varphi \vee \psi, \Gamma^r}\,\mathsf{or}$	$\chi_x = \chi_y$	$\Gamma^l \mid \iota_x$	$\dfrac{\overline{\iota_y} \mid \varphi, \psi, \Gamma^r}{\overline{\iota_x} \mid \varphi \vee \psi, \Gamma^r}\,\mathsf{or}$
$\dfrac{\varphi, \Gamma^l \mid \Gamma^r \qquad \psi, \Gamma^l \mid \Gamma^r}{\varphi \wedge \psi, \Gamma^l \mid \Gamma^r}\,\mathsf{and}$	$\chi_x = \chi_y \vee \chi_z$	$\dfrac{\dfrac{\dfrac{\varphi, \Gamma^l \mid \iota_y}{\varphi, \Gamma^l \mid \iota_y, \iota_z}\,\mathsf{wk} \qquad \dfrac{\psi, \Gamma^l \mid \iota_z}{\psi, \Gamma^l \mid \iota_y, \iota_z}\,\mathsf{wk}}{\varphi \wedge \psi, \Gamma^l \mid \iota_y, \iota_z}\,\mathsf{and}}{\varphi \wedge \psi, \Gamma^l \mid \iota_x}\,\mathsf{or}$	$\dfrac{\overline{\iota_y} \mid \Gamma^r \qquad \overline{\iota_z} \mid \Gamma^r}{\overline{\iota_x} \mid \Gamma^r}\,\mathsf{and}$
$\dfrac{\Gamma^l \mid \varphi, \Gamma^r \qquad \Gamma^l \mid \psi, \Gamma^r}{\Gamma^l \mid \varphi \wedge \psi, \Gamma^r}\,\mathsf{and}$	$\chi_x = \chi_y \wedge \chi_z$	$\dfrac{\Gamma^l \mid \iota_y \qquad \Gamma^l \mid \iota_z}{\Gamma^l \mid \iota_x}\,\mathsf{and}$	$\dfrac{\dfrac{\dfrac{\overline{\iota_y} \mid \varphi, \Gamma^r}{\overline{\iota_y}, \overline{\iota_z} \mid \varphi, \Gamma^r}\,\mathsf{wk} \qquad \dfrac{\overline{\iota_z} \mid \psi, \Gamma^r}{\overline{\iota_y}, \overline{\iota_z} \mid \psi, \Gamma^r}\,\mathsf{wk}}{\overline{\iota_y}, \overline{\iota_z} \mid \varphi \wedge \psi, \Gamma^r}\,\mathsf{and}}{\overline{\iota_x} \mid \varphi \wedge \psi, \Gamma^r}\,\mathsf{or}$
$\dfrac{\varphi, \Box_4^{-1}(\Gamma^l) \mid \Box_4^{-1}(\Gamma^r)}{\Box\varphi, \Gamma^l \mid \Gamma^r}\,\mathsf{box}$	$\chi_x = \Diamond\chi_y$	$\dfrac{\dfrac{\varphi, \Box_4^{-1}(\Gamma^l) \mid \iota_y}{\varphi, \Box_4^{-1}(\Gamma^l) \mid \Diamond\iota_y, \iota_y}\,\mathsf{wk}}{\Box\varphi, \Gamma^l \mid \iota_x}\,\mathsf{box}$	$\dfrac{\overline{\iota_y} \mid \Box_4^{-1}(\Gamma^r)}{\overline{\iota_x} \mid \Gamma^r}\,\mathsf{box}$
$\dfrac{\Box_4^{-1}(\Gamma^l) \mid \varphi, \Box_4^{-1}(\Gamma^r)}{\Gamma^l \mid \Box\varphi, \Gamma^r}\,\mathsf{box}$	$\chi_x = \Box\chi_y$	$\dfrac{\Box_4^{-1}(\Gamma^l) \mid \iota_y}{\Gamma^l \mid \iota_x}\,\mathsf{box}$	$\dfrac{\dfrac{\overline{\iota_y} \mid \varphi, \Box_4^{-1}(\Gamma^r)}{\Diamond\overline{\iota_y}, \overline{\iota_y} \mid \varphi, \Box_4^{-1}(\Gamma^r)}\,\mathsf{wk}}{\overline{\iota_x} \mid \Box\varphi, \Gamma^r}\,\mathsf{box}$

Table 4.1: Partial Interpolation Proofs

Lemma 4.25. *Given a finite* $\mathsf{G}^{\text{split}}$*-proof* (X, α)*, the collection of* $\mathsf{G}^{\text{ext}}_{\text{par}}$*-proofs* $\{(L_x, \beta_x)\}_{x \in X}$ *given in Lemma 4.24 is structured.*

Proof. This essentially follows from Lemma 4.24. To show that the collection of partial proofs is structured we have to show:

1. For every $x \in X$ there is a node $l_x \in L_x$ which we refer to as the root node,
2. For all $x \in X$ and $z \in L_x$, if $r^{\beta_x}(z) = \mathsf{as}$ then there is $y \in X$ such that $f^{\beta_x}(z) = f^{\beta_y}(l_y)$.

For (1) we define l_x to be a node guaranteed by Lemma 4.24 (1), and then (2) follows since given $x \in X$ and $z \in L_x$ such that $r^{\beta_x}(z) = \mathsf{as}$ by Lemma 4.24 (2) we know $f^{\beta_x}(z) = f^{\alpha}(y)^l \mid \iota_y$ for $y \in p^{\alpha}(x)$. Finally, by Lemma 4.24 (1) we know that $f^{\beta_y}(l_y) = f^{\alpha}(y)^l \mid \iota_y$. □

Lemma 4.26. *Given a finite* $\mathsf{G}^{\text{split}}$*-proof* (X, α)*, the collection of* $\mathsf{G}^{\text{ext}}_{\text{par}}$*-proofs* $\{(R_x, \gamma_x)\}_{x \in X}$ *given in Lemma 4.24 is structured.*

Proof. Analogous to Lemma 4.25. □

4.2.4 Proof Transformations

Theorem 4.27 (`proofTransformation` ✓)**.** *Given a* $\mathsf{G}^{\text{split}}$*-proof* (X, α)*, and a structured collection of* $\mathsf{G}^{\text{ext}}_{\text{par}}$*-proofs* $\{(P_x, \beta_x)\}_{x \in X}$*, let the root of* P_x *be denoted by* p_x*; a proof transformation* (Y, β) *(see Definition 3.22) is a* $\mathsf{G}^{\text{ext}}_{\text{skip}}$*-proof if for all* $x \in X$*, the following conditions hold:*

1. *If* $r^{\alpha}(x) = \mathsf{box}$ *then* P_x *has a* box *rule application on every path from* p_x *to the non-axiomatic leaves.*
2. *For all* $z \in P_x$ *such that* $r^{\beta_x}(z) = \mathsf{as}$*, the chosen* $y \in X$ *such that* $f^{\beta_x}(z) = f^{\beta_y}(p_y)$ *is in* $p^{\alpha}(x)$*.*

Proof. To begin with, recall by definition of (Y, β) in Definition 3.22 that

$$(f_p^{\beta}(x, z), f_n^{\beta}(x, z)) = (f_p^{\beta_x}(z), f_n^{\beta_x}(z)) \tag{4.3}$$

and by condition (2) we know that

$$(x, z) \lhd^{\beta} (x', z') \text{ if and only if } x \lhd^{\alpha} x' \text{ or, } x = x' \text{ and } z \lhd^{\beta_x} z'$$

We check that the (node), (step), and (path) conditions for the $\mathsf{G}^{\text{ext}}_{\text{skip}}$ system are met.

(node) Take $(x, z) \in Y$. We split cases depending on whether $r^{\beta_x}(z) = \mathsf{as}$. If so, then $(r^{\beta} \times f_p^{\beta})(x, z) = (\mathsf{skip}, \emptyset)$ by Definition 3.22, so we have our desired result. If $r^{\beta_x}(z) \neq \mathsf{as}$ then by Definition 3.17, the (node) condition of G^{ext} holds, our desired result.

(step) Take $(x, z) \in Y$. We split cases depending on whether $r^{\beta_x}(z) = \mathsf{as}$. If so, we know $f_p^{\beta_x}(z) = \emptyset$ by Definition 3.17 and $(r^{\beta} \times f_p^{\beta})(x, z) = (\mathsf{skip}, \emptyset)$ by Definition 3.22. By Definition 3.20 we must show $f^{\beta}[p^{\beta}(x, z)] = f^{\beta}(x, z)$ and $|p^{\beta}(x)| = 1$. This follows since the unique premise of (x, z) is (y, p_y) by Definition 3.22 and

$$f^{\beta}[p^{\beta}(x, z)] = f^{\beta}(y, p_y) = f^{\beta_y}(p_y) = f^{\beta_x}(z) = f^{\beta}(x, z).$$

The second and fourth equalities follow from (4.3) and the third equality follows from the structure condition. If $r^{\beta_x}(z) \neq \mathsf{as}$ then by Definition 3.17, the (step) condition of A holds, our desired result.

(path) Recall that the path conditions of the $\mathsf{G}^{\text{ext}}_{\text{skip}}$ system are the following:

1. On every infinite $\lhd^\beta$-path $(x_i, z_i)_{i<\omega}$ there are infinitely many i such that $r^\beta(x_i, z_i) = \mathsf{box}$,
2. On every infinite $\lhd^\beta$-path $(x_i, z_i)_{i<\omega}$ there are not cofinitely many i such that $r^\beta(x_i, z_i) = \mathsf{skip}$.

Recall that $Y = \{(x, z) \mid x \in X, z \in P_x\}$. We will show (1) as (1) will imply (2). This follows since if there are infinitely many i such that $r^\beta(x_i, z_i) = \mathsf{box}$ then there cannot be cofinitely many i such that $r^\beta(x_i, z_i) = \mathsf{skip}$. To show (1), let $(x_i, z_i)_{i<\omega}$ be an infinite path in (Y, β). We split cases depending on whether there exists n such that for all $i \geq n$, $x_i = x_n$.

- If this is the case, then we can find an infinite $\lhd^{\beta_{x_n}}$-path in P_{x_n} given by $(z_j)_{j\geq n}$. Since P_{x_n} is a $\mathsf{G}^{\text{ext}}_{\text{par}}$-proof, we know there are infinitely many j such that $r^{\beta_{x_j}}(z_j) = \mathsf{box}$. By (4.3) we know $r^{\beta_{x_j}}(z_j) = r^\beta(x_j, z_j)$, so on this path there are infinitely many nodes such that $r^\beta(x_j, z_j) = \mathsf{box}$.
- If this is not the case, then observe that $(x_i, z_i)_{i<\omega}$ induces an infinite path in (X, α). In other words, we can write our path as

$$(x_0, z_{0,0}), \ldots, (x_0, z_{0,m_0}), (x_1, z_{1,0}), \ldots, (x_1, z_{1,m_1}), \ldots$$

where $x_i \lhd^\alpha x_{i+1}$ for all i, z_{i,m_i} is a non-axiomatic leaf for all i, and $z_{i,0}$ is the root node of P_{x_i} for all $i \neq 0$. Since $(x_i)_{i<\omega}$ is an infinite $\lhd^\alpha$-path in (X, α), we know by Lemma 4.15 that there are infinitely many i such that $r^\alpha(x_i)$ is a box node. Therefore, for all of these i such that $i \neq 0$, by condition (1) there will be $j \leq m_i$ such that $r^{\beta_{x_i}}(z_{i,j}) = \mathsf{box}$, giving us our desired result. □

Our Lyndon interpolation result follows. We first state the syntactic result for finite $\mathsf{G}^{\text{split}}$-proofs, and then state the main result.

Theorem 4.28 (`syntactic_interpolation` ✓). *Given a finite $\mathsf{G}^{\text{split}}$-proof (X, α), there are $\mathsf{G}^{\text{ext}}_{\text{skip}}$-proofs (Y, β) and (Z, γ) which prove $f(x)^l \mid \iota_x$ and $\overline{\iota_x} \mid f(x)^r$ for each $x \in X$.*

Proof. We will prove there is a $\mathsf{G}^{\text{ext}}_{\text{skip}}$-proofs (Y, β) of $f(x)^l \mid \iota_x$ for each $x \in X$. The right interpolation correctness proof follows analogously. By Lemma 4.25 there is a structured collection of $\mathsf{G}^{\text{ext}}_{\text{par}}$-proofs $\{(L_x, \beta_x)\}_{x\in X}$. By Theorem 4.27 the proof transformation (Y, β) of this collection is a $\mathsf{G}^{\text{ext}}_{\text{skip}}$-proof. Take $x \in X$, it remains to show $(Y, \beta) \vdash_{\mathsf{G}^{\text{ext}}_{\text{skip}}} f(x)^l \mid \iota_x$. This is witnessed by $(x, l_x) \in Y$ since by Lemma 4.24, $f^{\beta_x}(l_x) = f(x)^l \mid \iota_x$, so $f^\beta(x, l_x) = f(x)^l \mid \iota_x$. □

Theorem 4.29 (`interpolation` ✓[1]). *Given $\varphi, \psi \in \mathcal{L}_\Box$ such that $\vDash \overline{\varphi} \vee \psi$, there is a formula χ such that the following two properties hold:*

1. $\mathsf{Lit}(\chi) \subseteq \mathsf{Lit}(\varphi) \cap \mathsf{Lit}(\psi)$,
2. $\vDash \overline{\varphi} \vee \chi$ *and* $\vDash \overline{\chi} \vee \psi$.

[1] In our Lean formalization we prove Craig interpolation not Lyndon interpolation. See Chapter 6 for details.

Proof. Take φ, ψ such that $\vDash \overline{\varphi} \vee \psi$. By Theorem 4.3 we know $\vdash_{\mathsf{G}^{\mathrm{split}}} \overline{\varphi} \mid \psi$, i.e. there is (X, α) and $x \in X$ such that $f(x) = \overline{\varphi} \mid \psi$. By Theorem 4.13 we can assume that X is finite. By Theorem 4.28 we know there exists χ such that

1. $\mathsf{Lit}(\chi) \subseteq \mathsf{Lit}(\varphi) \cap \mathsf{Lit}(\psi)$,
2. $\vdash_{\mathsf{G}^{\mathrm{ext}}_{\mathrm{skip}}} \overline{\varphi} \mid \chi$ and $\vdash_{\mathsf{G}^{\mathrm{ext}}_{\mathrm{skip}}} \overline{\chi} \mid \psi$.

If $\mathsf{G}^{\mathrm{ext}}_{\mathrm{skip}}$ is sound, we have $\vDash \overline{\varphi} \vee \chi$ and $\vDash \overline{\chi} \vee \psi$. The fact that this system is sound follows by the same argumentation in [Men24, Prop. 2.2.8] that Shamkanov's system is sound and the fact that the rule skip is not used cofinitely often on any infinite path. □

Chapter 5

The Alternation-Free μ-Calculus

We now move to a more complex example, a proof system for the alternation-free μ-calculus (AFMC). The systems we use as a basis for our coalgebraic notion of proof are the cyclic and infinite tree proof systems created by J. Marti and Y. Venema called Focus and Focus_∞ respectively [MV21]. This system restricts to guarded formulas, for which it is known that all alternation-free formulas are semantically equivalent to guarded alternation-free formulas. Let $\mathcal{L}^{\mathrm{af}}_{\Box}$ denote the set of all guarded formulas of the alternation-free μ-calculus.

Cyclic proof systems for fixpoint logics with fixpoint operators have appeared in multiple works. Key systems for the modal μ-calculus were introduced by C. Stirling in [Sti14] and N. Jungteerapanich in [Jun10], and relied on annotating formulas within the system with added strings of information used to show soundness and completeness of the systems. A natural starting point to discuss these systems is how they handle fixpoint formulas. Since $\eta x.\varphi \equiv \varphi[\eta x.\varphi/x]$, fixpoints could be handled as follows.

$$\frac{\Gamma, \varphi[\mu x.\varphi/x]}{\Gamma, \mu x.\varphi} \qquad \frac{\Gamma, \varphi[\nu x.\varphi/x]}{\Gamma, \nu x.\varphi}$$

Of course, adding these rules makes the proof system ill-founded because we can indefinitely unfold formulas. The flaw of such a system becomes clear when considering the following proof of $\mu x.\Box x$, which is well-known to be semantically equivalent to $\bot$.

$$\cfrac{\Box\mu x.\Box x}{\mu x.\Box x}$$

To remedy this, added pieces of information, called annotations, are added to the proof. In the alternation-free case, the work in [MV21] posits that we need only a single bit of data to be added to each formula in the proof. This bit will be an element of the set $\{u, f\}$, and we will refer to the set of annotated formulas as $a\mathcal{L}^{\mathrm{af}}_{\Box} := \mathcal{L}^{\mathrm{af}}_{\Box} \times \{u, f\}$. Given $\varphi \in \mathcal{L}^{\mathrm{af}}_{\Box}$, we let $\varphi^i = (\varphi, i)$ and we call φ^i *unfocused* if $i = u$ and *focused* if $i = f$. The rules of the system are given in Figure 5.1.

$$\frac{}{\top^a}\ \mathsf{top} \qquad \frac{}{p^a, \overline{p}^b}\ \mathsf{ax} \qquad \frac{\varphi^a, \psi^a, \Gamma}{(\varphi \vee \psi)^a, \Gamma}\ \mathsf{or} \qquad \frac{\varphi^a, \Gamma \qquad \psi^a, \Gamma}{(\varphi \wedge \psi)^a, \Gamma}\ \mathsf{and} \qquad \frac{\varphi^a, \Gamma}{\Box\varphi^a, \Diamond\Gamma}\ \mathsf{box}$$

$$\frac{\varphi[\mu x.\varphi/x]^u, \Gamma}{\mu x.\varphi^a, \Gamma}\ \mathsf{mu} \qquad \frac{\varphi[\nu x.\varphi/x]^a, \Gamma}{\nu x.\varphi^a, \Gamma}\ \mathsf{nu} \qquad \frac{\Gamma}{\varphi^a, \Gamma}\ \mathsf{wk} \qquad \frac{\varphi^f, \Gamma}{\varphi^u, \Gamma}\ \mathsf{f} \qquad \frac{\varphi^u, \Gamma}{\varphi^f, \Gamma}\ \mathsf{u}$$

Figure 5.1: Rule applications of the systems Focus and Focus_∞ [MV21].

Focus and Focus_∞-proofs have two added conditions about infinite paths that are needed to prove the system is sound and complete. The first (much like GL's implicit path condition) states that every infinite path must have infinitely many box rule applications, and the second states that every infinite path must have cofinitely many nodes which contain a formula in focus, and are not u or f rule applications. With these added path conditions, one can check that proving $\mu x.\Box x^a$ for any $a \in \{u, f\}$ becomes impossible. We now unify these two systems using our coalgebraic approach.

Definition 5.1. We define the set of rules $\mathcal{R}_{\mathsf{AFMC}}$ to be the set $\{\mathsf{top}, \mathsf{ax}, \mathsf{or}, \mathsf{and}, \mathsf{box}, \mathsf{mu}, \mathsf{nu}, \mathsf{wk}, \mathsf{f}, \mathsf{u}\}$.

Definition 5.2. Given a $\mathcal{T}^{a\mathcal{L}^{\mathrm{af}}_{\Box}}_{\mathcal{R}_{\mathsf{AFMC}}}$-coalgebra (X, α) and $x \in X$, we say that x is *annotated* if $r(x) \neq \mathsf{u}$, $r(x) \neq \mathsf{f}$, and $f(x)$ contains a formula in focus, i.e. there is φ such that $\varphi^f \in f(x)$.

Definition 5.3. A $\mathcal{T}^{a\mathcal{L}^{\mathrm{af}}_{\Box}}_{\mathcal{R}_{\mathsf{AFMC}}}$-coalgebra (X, α) is called an $\mathsf{A}^{\mathrm{gen}}$-proof if the following node, step, and path conditions are met:

(node) For all $x \in X$, $(r \times f_p)(x)$ has one of the following forms:

1. $(\mathsf{top}, \{\top^a\})$
2. $(\mathsf{ax}, \{q^a, \overline{q}^b\})$
3. $(\mathsf{or}, \{(\varphi \vee \psi)^a\})$
4. $(\mathsf{and}, \{(\varphi \wedge \psi)^a\})$
5. $(\mathsf{box}, \{\Box\varphi^a\})$
6. $(\mathsf{mu}, \{\mu x.\varphi^a\})$
7. $(\mathsf{nu}, \{\nu x.\varphi^a\})$
8. $(\mathsf{wk}, \{\varphi^a\})$
9. $(\mathsf{f}, \{\varphi^u\})$
10. $(\mathsf{u}, \{\varphi^f\})$

where $q \in \mathsf{Prop}$, $a, b \in \{u, f\}$, $\varphi, \psi \in \mathcal{L}^{\mathrm{af}}_{\Box}$.

(step) The step conditions represented in Figure 5.1 are satisfied.

(path) The following path conditions are satisfied:

1. On every infinite $\lhd$-path $(x_i)_{i<\omega}$ there are cofinitely many i such that x_i is annotated.
2. On every infinite $\lhd$-path $(x_i)_{i<\omega}$ there are infinitely many i such that $r(x_i) = \mathsf{box}$.

Remark 5.4. From now on, we will not write out each step condition using our structure morphism α for a given coalgebra (X, α). Instead we will refer to a figure of the rule applications. The definitions of the $\mathsf{G}^{\mathrm{gen}}$, $\mathsf{G}^{\mathrm{split}}$, and $\mathsf{G}^{\mathrm{ext}}$-proof systems in Chapter 4 (Definition 4.2, Definition 4.12, and Definition 4.23 respectively) should sufficiently demonstrate how to codify step conditions.

Theorem 5.5 (Soundness and Completeness)**.** *Given a sequent $\Delta \in \mathcal{P}_\omega(\mathcal{L}^{\mathrm{af}}_{\Box})$,*

$$\vdash_{\mathsf{A}^{\mathrm{gen}}} \{\varphi^f \mid \varphi \in \Delta\} \text{ if and only if } \vDash \bigvee \Delta.$$

where $\vdash_{\mathsf{A}^{\mathrm{gen}}}$ is as defined in Definition 3.10 and $\vDash$ denotes semantic validity on Kripke models.

Proof. The work in [MV21] establishes soundness and completeness for the ill-founded proof system Focus_∞. Observing that every ill-founded proof in Focus_∞ is an $\mathsf{A}^{\mathrm{gen}}$-proof, and that every $\mathsf{A}^{\mathrm{gen}}$-proof can be turned into an ill-founded Focus_∞-proof via unraveling, soundness and completeness follows. □

5.1 Finitization

As in Chapter 4, we need to finitize our proofs. We will begin by showing that point generation yields an f-finite proof. We get this result using the same argumentation as Section 4.1.

Lemma 5.6. *Given an* $\mathsf{A}^{\mathrm{gen}}$*-proof* (X, α) *and* $x, y \in X$*. If* $x \lhd^* y$*, then* $f(y) \subseteq \mathrm{Clos}(f(x)) \times \{u, f\}$*.*

Proof. Analogous to Lemma 4.5. □

Lemma 5.7. *Given an* $\mathsf{A}^{\mathrm{gen}}$*-proof* (X, α) *and some node* $x \in X$*, the point-generated* $\mathcal{T}^{a\mathcal{L}^{\mathrm{af}}_{\Box}}_{\mathcal{R}_{\mathsf{AFMC}}}$*-coalgebra around* x *(see Definition 3.13) given by* $(\uparrow x, \alpha_x)$ *is an* $\mathsf{A}^{\mathrm{gen}}$*-proof.*

Proof. The fact that the (node) and (step) conditions are satisfied is analogous to Lemma 4.6.

(path) Take an infinite $\lhd^{\alpha_x}$-path $(x_i)_{i<\omega}$ in $(\uparrow x, \alpha_x)$. This is also an infinite $\lhd^{\alpha}$-path $(x_i)_{i<\omega}$ in (X, α), which satisfies the (path) condition. Thus, since $r^{\alpha_x}(y) = r^{\alpha}(y)$ and $f^{\alpha_x}(y) = f^{\alpha}(y)$ for any $y \in \uparrow x$, we have our desired result. □

Theorem 5.8. *For any annotated sequent* Δ*, if there is an* $\mathsf{A}^{\mathrm{gen}}$*-proof* (X, α) *of* Δ *then there is an* f*-finite proof of* Δ *where* $|f[X]|$ *is bounded by* $2^{|\Delta|_{\mathrm{Cl}}+1}$*.*

Proof. Let (X, α) be an $\mathsf{A}^{\mathrm{gen}}$-proof of Δ witnessed by x. Take the point-generated coalgebra around x given by $(\uparrow x, \alpha_x)$. By Lemma 5.7 this is an $\mathsf{A}^{\mathrm{gen}}$-proof. Moreover, since $f^{\alpha}(x) = f^{\alpha_x}(x)$, we know that $(\uparrow x, \alpha_x)$ is an $\mathsf{A}^{\mathrm{gen}}$-proof of Δ. Finally, for all $y \in \uparrow x$, by Lemma 5.6 we know that $f(y) \subseteq \mathrm{Clos}(\Delta) \times \{u, f\}$, so $|f[\uparrow x]| \leq 2^{|\Delta|_{\mathrm{Cl}}+1}$. □

Thus our system $\mathsf{A}^{\mathrm{gen}}$ has f-finite proofs. If we were to continue following the methodology of Section 4.1, we would take an arbitrary filtration for the equivalence relation given by

$$x \sim^{\alpha} y \text{ if and only if } f(x) = f(y).$$

Figure 5.2 shows why taking arbitrary filtrations is not feasible. In short, filtrations preserve the node and step conditions, but may fail to preserve the path conditions.

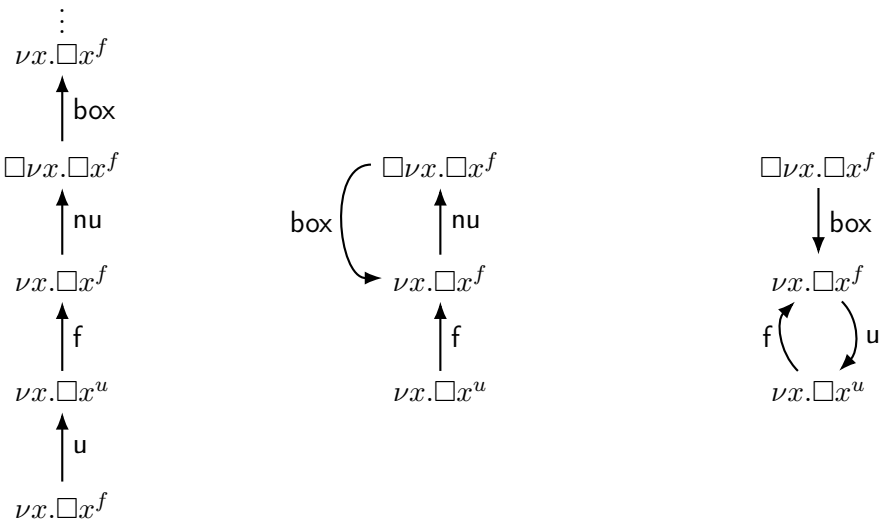

Figure 5.2: An $\mathsf{A}^{\mathrm{gen}}$-proof of $\nu x.\Box x^f$ (left), a filtration of the left proof which is an $\mathsf{A}^{\mathrm{gen}}$-proof (middle), and a filtration of the left proof which is not an $\mathsf{A}^{\mathrm{gen}}$-proof (right).

Our path conditions are recurrence and persistence statements expressible in the modal μ-calculus. It is a well-known property of the modal μ-calculus that filtrations do not preserve satisfaction [BS07]. In Figure 5.2, we give an example specific to our proof system. On the left we present a proof of

$\nu x.\Box x^f$, and two filtrations of this proof. The filtration shown in the middle is a valid proof, since one can check via direct inspection that all infinite paths use the box-rule application infinitely many times, and the u and f rules cofinitely many times as well as having cofinitely many sequents with a formula in focus. The filtration on the right does not satisfy either of these path conditions and is not a valid proof. Thus we must proceed via a different method.

We fix a f-finite proof (X, α) of Δ for the rest of this section, and give our construction of a finite proof proving the same sequents shown in (X, α). We are inspired by the construction of D. Kozen in [Koz88], but heavily simplify things. The contents of [Koz88] prove the finite model property of the modal μ-calculus by using a well-quasi-ordering on sets of formulas of the transfinite modal μ-calculus.

The key observation is that the (node), (step), and (path) conditions in Definition 5.3 can to some degree[1] be rewritten in the modal μ-calculus. Therefore one approach towards proving finitization could be to use the modal μ-calculus as a meta-language, write all of our (node), (step), and (path) conditions in the modal μ-calculus, view our $\mathsf{A}^{\mathrm{gen}}$-proof (X, α) as a Kripke frame $(X, \lhd^\alpha)$, give the necessary interpretation, finitize it using Kozen's well-quasi-order construction, turn the finite model into a finite $\mathcal{T}^{a\mathcal{L}^{\mathrm{af}}_{\Box}}_{\mathcal{R}_{\mathsf{AFMC}}}$-coalgebra, and prove that said $\mathcal{T}^{a\mathcal{L}^{\mathrm{af}}_{\Box}}_{\mathcal{R}_{\mathsf{AFMC}}}$-coalgebra is still an $\mathsf{A}^{\mathrm{gen}}$-proof which in fact proves Δ.

However, inspection of Kozen's method grants us a simpler and clearer construction. This construction will focus solely on the two path conditions as modal μ-calculus formulas and assign weights (ordinals) to each node in X that quantify how efficient the node is at proving the two path conditions in the scope of the entire proof. The main idea will be to use these weights to create an intuitive order $\prec$ on X, from which we will show there is a finite set $Y \subseteq X$ of nodes of minimal weights which we will then use as the base set of our finite $\mathsf{A}^{\mathrm{gen}}$-proof. To begin with, we define two weight functions we use to construct this ordering.

5.1.1 Weighing Nodes

Recall that we call x an *annotated* node if $r(x)$ is not u or f and $f(x)$ contains a formula in focus. We assign a valuation V to X given by $p_{\mathsf{ann}} \mapsto \{x \mid x \text{ annotated}\}$ and $p_{\mathsf{box}} \mapsto \{x \mid r(x) = \mathsf{box}\}$. By the path condition, we know that for every node $x \in X$, the following formulas are satisfied:

$\mu z.p_{\mathsf{box}} \vee (\overline{p_{\mathsf{box}}} \wedge \Box z)$ *"every infinite path has a box node"*

$\mu z_1.\nu z_2.(\overline{p_{\mathsf{ann}}} \wedge \Box z_1) \vee (p_{\mathsf{ann}} \wedge \Box z_2)$ *"every infinite path has cofinitely many annotated nodes"*

Thus every node x has a least ordinal for both of these formulas.

Definition 5.9. We define the *box weight* of a node $x \in X$, denoted $w_{\mathsf{box}}(x)$, to be the least ordinal of $\mu z.p_{\mathsf{box}} \vee (\overline{p_{\mathsf{box}}} \wedge \Box z)$ at x.

We state some properties about w_{box} that will be useful later.

Lemma 5.10. *For all $x \in X$, if $r(x) = \mathsf{box}$ then $w_{\mathsf{box}}(x) = 1$.*

Proof. This follows from the semantics of the transfinite μ-calculus. Since $x \vDash p_{\mathsf{box}}$ it follows that $x \vDash \mu^1 z.p_{\mathsf{box}} \vee (\overline{p_{\mathsf{box}}} \wedge \Box z)$. Thus $w_{\mathsf{box}}(x) \leq 1$ if $r(x) = \mathsf{box}$. To conclude the proof, we can observe that $w_{\mathsf{box}}(x) \neq 0$ since $x \nvDash \mu^0 z.p_{\mathsf{box}} \vee (\overline{p_{\mathsf{box}}} \wedge \Box z)$. □

[1] To properly express the conditions of being an $\mathsf{A}^{\mathrm{gen}}$-proof, the restriction on the number of premises each rule application has needs a graded modal logic to express, although for finitization via Kozen's method, this is not necessary.

Lemma 5.11. *If $x \lhd y$ and $r(x) \neq \mathsf{box}$ then $w_{\mathsf{box}}(y) < w_{\mathsf{box}}(x)$.*

Proof. Let $\beta = w_{\mathsf{box}}(x)$ and $\theta = \mu z.p_{\mathsf{box}} \vee (\overline{p_{\mathsf{box}}} \wedge \Box z)$. Observe the following sequence of deductions.

$x \vDash \mu^{\beta} z.p_{\mathsf{box}} \vee (\overline{p_{\mathsf{box}}} \wedge \Box z)$

$x \vDash p_{\mathsf{box}} \vee (\overline{p_{\mathsf{box}}} \wedge \Box \theta^{\gamma})$ where $\gamma < \beta$

$x \vDash \overline{p_{\mathsf{box}}} \wedge \Box \theta^{\gamma}$ since $r(x) \neq \mathsf{box}$

$y \vDash \mu^{\gamma} z.p_{\mathsf{box}} \vee (\overline{p_{\mathsf{box}}} \wedge \Box z)$

Thus $w_{\mathsf{box}}(y) \leq \gamma < w_{\mathsf{box}}(x)$. □

Definition 5.12. We define the *annotation weight* of a node x, denoted $w_{\mathsf{ann}}(x)$, to be the least ordinal of $\mu z_1.\nu z_2.(\overline{p_{\mathsf{ann}}} \wedge \Box z_1) \vee (p_{\mathsf{ann}} \wedge \Box z_2)$.

We observe that w_{ann} behaves differently from w_{box} with respect to $\lhd$. The difference will be made clear by the following lemmas.

Lemma 5.13. *For all $x, y \in X$ such that $x \lhd y$, if x is not annotated then $w_{\mathsf{ann}}(y) < w_{\mathsf{ann}}(x)$.*

Proof. Let $\beta = w_{\mathsf{ann}}(x)$ and $\theta = \mu z_1.\nu z_2.(\overline{p_{\mathsf{ann}}} \wedge \Box z_1) \vee (p_{\mathsf{ann}} \wedge \Box z_2)$. Observe the following sequence of deductions.

$x \vDash \mu^{\beta} z_1.\nu z_2.(\overline{p_{\mathsf{ann}}} \wedge \Box z_1) \vee (p_{\mathsf{ann}} \wedge \Box z_2)$

$x \vDash \nu z_2.(\overline{p_{\mathsf{ann}}} \wedge \Box \theta^{\gamma}) \vee (p_{\mathsf{ann}} \wedge \Box z_2)$ where $\gamma < \beta$

$x \vDash (\overline{p_{\mathsf{ann}}} \wedge \Box \theta^{\gamma}) \vee (p_{\mathsf{ann}} \wedge \Box \operatorname{unf}(\theta^{\gamma}))$

$x \vDash \overline{p_{\mathsf{ann}}} \wedge \Box \theta^{\gamma}$ since $x \nvDash p_{\mathsf{ann}}$

$y \vDash \mu^{\gamma} z_1.\nu z_2.(\overline{p_{\mathsf{ann}}} \wedge \Box z_1) \vee (p_{\mathsf{ann}} \wedge \Box z_2)$

So $w_{\mathsf{ann}}(y) \leq \gamma < w_{\mathsf{ann}}(x)$. □

Lemma 5.14. *For all $x, y \in X$ such that $x \lhd y$, if x is annotated then $w_{\mathsf{ann}}(y) \leq w_{\mathsf{ann}}(x)$.*

Proof. Let $\beta = w_{\mathsf{ann}}(x)$ and $\theta = \mu z_1.\nu z_2.(\overline{p_{\mathsf{ann}}} \wedge \Box z_1) \vee (p_{\mathsf{ann}} \wedge \Box z_2)$. Observe the following sequence of deductions.

$x \vDash \mu^{\beta} z_1.\nu z_2.(\overline{p_{\mathsf{ann}}} \wedge \Box z_1) \vee (p_{\mathsf{ann}} \wedge \Box z_2)$

$x \vDash \nu z_2.(\overline{p_{\mathsf{ann}}} \wedge \Box \theta^{\gamma}) \vee (p_{\mathsf{ann}} \wedge \Box z_2)$ where $\gamma < \beta$

$x \vDash (\overline{p_{\mathsf{ann}}} \wedge \Box \theta^{\gamma}) \vee (p_{\mathsf{ann}} \wedge \Box \operatorname{unf}(\theta^{\gamma}))$

$x \vDash p_{\mathsf{ann}} \wedge \Box \operatorname{unf}(\theta^{\gamma})$ since $x \vDash p_{\mathsf{ann}}$

$y \vDash \operatorname{unf}(\theta^{\gamma})$

$y \vDash \theta^{\gamma+1}$

So $w_{\mathsf{ann}}(y) \leq \gamma + 1 \leq w_{\mathsf{ann}}(x)$. □

Definition 5.15. Let $w : X \to \mathsf{Ord}^2$ be given by $w_{\mathsf{ann}} \times w_{\mathsf{box}}$. We will refer to $w(x)$ as the *weight* of a node x.

Let $<_l$ denote the lexicographical order on Ord^2 with preference given to the first projection, in other words

$$w(x) <_l w(y) \text{ if and only if } w_{\mathsf{ann}}(x) < w_{\mathsf{ann}}(y), \text{ or } w_{\mathsf{ann}}(x) = w_{\mathsf{ann}}(y) \text{ and } w_{\mathsf{box}}(x) < w_{\mathsf{box}}(y).$$

We now introduce the following order $\prec \subseteq X \times X$ on nodes as follows.

$$x \prec y \iff f(x) = f(y) \text{ and } w(x) <_l w(y).$$

If $x \prec y$ we say x is *better* than y. The reason for calling one node better than another is that if $x \prec y$, we know that $f(x) = f(y)$, so the nodes are proving the same sequent. However $w_{\mathsf{ann}}(x)$ or $w_{\mathsf{box}}(x)$ are equal to or smaller than their counterparts in y, so x is essentially capable of proving the path conditions with less μ-unfoldings than y in the game-theoretic semantics of the modal μ-calculus. This comes with the stipulation that we give preference to the condition (path) 2. over (path) 1. as we will later use (path) 2. to prove (path) 1.

Lemma 5.16. *There is a set $Y \subseteq X$ such that for all $x \in X$, there exists $y \in Y$, such that $y \preceq x$ and $|Y| = |f[X]|$.*

Proof. Take the equivalence relation $\sim^\alpha$ given by

$$x \sim^\alpha y \text{ if and only if } f(x) = f(y).$$

Given an equivalence class $[x]$, take $y \in [x]$ such that $w(y) \preceq w(z)$ for all $z \in [x]$. Since $(\mathsf{Ord}^2, <_l)$ is well-ordered, there will always be such a y. Let $q : X \to X$ be a choice function which chooses for each $x \in X$, the choice of node $y \in [x]$ such that $w(y) \preceq w(z)$ for all $z \in [x]$. Let $Y := q[X]$, it remains to show $|Y| = |f[X]|$. This can be seen by observing that there is a bijection $Y \to f[X]$ by $y \mapsto f(y)$. □

Let q and Y be as defined above. Thus $q(x) \preceq x$ for all x. Moreover, let $i : Y \hookrightarrow X$ be the inclusion map. By definition of q it is a choice function for the equivalence relation given by $\sim^\alpha$ defined above. Let $\beta := (\mathcal{T}^{a\mathcal{L}^{\mathrm{af}}_{\Box}}_{\mathcal{R}_{\mathsf{AFMC}}} q) \circ \alpha \circ i$.

Remark 5.17. Recalling Figure 5.2, we saw that for the example proof we gave, there was a filtration through $\lhd^\alpha$ that is a proof, shown in the middle of the figure. It turns out this was not a coincidence, and we can now show that for every proof there always is a filtration that is also a proof.

5.1.2 Filtration

Lemma 5.18. *If $y \lhd^\beta z$ and y is not annotated then $w_{ann}(z) < w_{ann}(y)$.*

Proof. Suppose $y \lhd^\beta z$ and y is not annotated, then $z \in q[p^\alpha(y)]$ so there exists $x \in p^\alpha(y)$ such that $q(x) = z$. Since $x \in p^\alpha(y)$ we know $y \lhd^\alpha x$ and since y is not annotated we know $w_{\mathsf{ann}}(x) < w_{\mathsf{ann}}(y)$. Moreover, since $q(x) = z$ we know that $z \preceq x$, so $w_{\mathsf{ann}}(z) \leq w_{\mathsf{ann}}(x)$. Putting this together, we get $w_{\mathsf{ann}}(z) < w_{\mathsf{ann}}(y)$. □

Lemma 5.19. *If $y \lhd^\beta z$ and y is annotated then $w_{ann}(z) \leq w_{ann}(y)$.*

Proof. Suppose $y \lhd^\beta z$ and y is annotated, then $z \in q[p^\alpha(y)]$ so there exists $x \in p^\alpha(y)$ such that $q(x) = z$. Since $x \in p^\alpha(y)$ we know $y \lhd^\alpha x$ and since y is annotated we know $w_{\mathsf{ann}}(x) \leq w_{\mathsf{ann}}(y)$. Moreover, since $q(x) = z$ we know that $z \preceq x$, so $w_{\mathsf{ann}}(z) \leq w_{\mathsf{ann}}(x)$. Putting this together, we get $w_{\mathsf{ann}}(z) \leq w_{\mathsf{ann}}(y)$. □

Lemma 5.20. *If $y \lhd^\beta z$ and y is not a box node then $w(z) <_l w(y)$.*

Proof. Suppose $y \lhd^\beta z$ and y is not a box node. If y is not annotated then by Lemma 5.18 we have $w_{\mathsf{ann}}(z) < w_{\mathsf{ann}}(y)$ so $w(z) <_l w(y)$ so we are done. Otherwise, suppose y is annotated. Since $z \in q[p^\alpha(y)]$ there is $x \in p^\alpha(y)$ such that $y \lhd^\alpha x$ and $z = q(x)$. Since y is annotated we know $w_{\mathsf{ann}}(x) \leq w_{\mathsf{ann}}(y)$ and since it is not a box node we know $w_{\mathsf{box}}(x) < w_{\mathsf{box}}(y)$ so $w(x) < w(y)$. Moreover, since $q(x) = z$ we know that $z \preceq x$ so $w(z) \leq w(x) < w(y)$, giving our desired result. □

Theorem 5.21. *The coalgebra (Y, β) as defined above is an $\mathsf{A}^{\mathrm{gen}}$-proof.*

Proof. We show that the (node), (step), and (path) conditions are met. To begin with, observe that for all $y \in Y$,

$$(\mathcal{T}^{a\mathcal{L}^{\mathrm{af}}_{\Box}}_{\mathcal{R}_{\mathsf{AFMC}}} q) \circ \alpha \circ i(y) = (r^\alpha(y), f^\alpha_p(y), f^\alpha_n(y), q[p^\alpha(y)])$$

(node) Take $y \in Y$. Since $(r^\beta \times f^\beta_p(y)) = (r^\alpha \times f^\alpha_p(y))$, and $(r^\alpha \times f^\alpha_p(y))$ has one of the forms listed in the node condition, we know that $(r^\beta \times f^\beta_p(y))$ will have one of the necessary forms.

(step) Take $y \in Y$. Similar to above, since $(r^\beta \times f^\beta_p \times f^\beta_n)(y) = (r^\alpha \times f^\alpha_p \times f^\alpha_n)(y)$ all necessary conditions will clearly be met besides the ones about $|p^\beta(y)|$. Moreover, observe that $p^\beta(y) = q[p^\alpha(y)]$ so $|p^\beta(y)| = |p^\alpha(y)|$ if $|p^\alpha(y)|$ is 0 or 1, and $|p^\beta(y)| \leq |p^\alpha(y)|$ if $|p^\alpha(y)| = 2$. It can be checked that this meets the necessary condition by checking each case individually.

(path) We now prove the (path) conditions.

1. Let $(y_i)_{i<\omega}$ be an infinite $\lhd^\beta$-path. To begin with, observe that by Lemma 5.18 and Lemma 5.19 that $w_{\mathsf{ann}}(y_{i+1}) \leq w_{\mathsf{ann}}(y_i)$ so we have an infinite non-increasing chain of ordinals. Suppose for every $i < \omega$ we can find $j \geq i$ such that the node y_j is not annotated. Thus by Lemma 5.18 for each of these y_j we have $w_{\mathsf{ann}}(y_{j+1}) < w_{\mathsf{ann}}(y_j)$. Thus we have an infinite non-increasing chain with an infinitely many decreasing steps, a contradiction.
2. Let $(y_i)_{i<\omega}$ be an infinite $\lhd^\beta$-path. By (1) we know there must be some j such that for all $i \geq j$, the node y_i is annotated. We must show that infinitely many of these y_i are box nodes. Suppose only finitely many of them are, and let $n \geq j$ be the maximal index such that y_n is a box node (if there are none, let $n = j$). Then for all $k < \omega$, it follows from Lemma 5.20 that $w(y_{n+k+1}) <_l w(y_{n+k})$ so we have found an infinite decreasing chain, a contradiction. □

Recall for the entirety of this section we fixed an $\mathsf{A}^{\mathrm{gen}}$-proof (X, α). We unfix this and combine the work of this section to get our bounded finitization result.

Theorem 5.22. *Given an f-finite $\mathsf{A}^{\mathrm{gen}}$-proof (X, α) of Δ, there exists a finite $\mathsf{A}^{\mathrm{gen}}$-proof (Y, β) of Δ whose size is $|f^\alpha[X]|$.*

Proof. Let (X, α) be a f-finite proof of Δ, i.e. $|f[X]|$ is finite. Let $(\hat{X}, \hat{\alpha})$ be the filtration of (X, α) through the equivalence relation $\sim^\alpha$ for the choice function q which chooses for every $x \in X$ an element of the equivalence class which has minimal weight. By Theorem 5.21, this is an $\mathsf{A}^{\mathrm{gen}}$-proof. Observe there is a bijection $\hat{X} \to f^{\hat{\alpha}}[\hat{X}]$ given by $y \mapsto f^{\hat{\alpha}}(y)$. This map is injective since if two representatives y_1 and y_2 of different equivalence classes satisfy $f^{\hat{\alpha}}(y_1) = f^{\hat{\alpha}}(y_2)$, then $f^\alpha(y_1) = f^\alpha(y_2)$ so $y_1 \sim^\alpha y_2$, a contradiction. Thus $|\hat{X}| = |f^{\hat{\alpha}}[\hat{X}]|$. Finally, let x witness the proof of Δ in (X, α). Let $\hat{x}$ denote the representative of the equivalence class x is in. Then $f^{\hat{\alpha}}(\hat{x}) = f^\alpha(x)$ and $f^\alpha(x) = \Delta$ so $(\hat{X}, \hat{\alpha})$ proves Δ, witnessed by $\hat{x}$. □

We combine Theorem 5.8 and Theorem 5.22 to prove the main result of this section.

Theorem 5.23. *For all annotated sequents Δ, if there is an $\mathsf{A}^{\mathrm{gen}}$-proof of Δ then there is a finite $\mathsf{A}^{\mathrm{gen}}$-proof of Δ whose size is bounded by $2^{|\Delta|_{\mathrm{Cl}}+1}$.*

Proof. This follows from Theorem 5.8 and Theorem 5.22. □

5.2 Interpolation

We follow the same approach toward interpolation as we did in Section 4.2. In this case, we will not need to extend our system with a cut rule, making our result more straightforward. We begin by introducing a split version of the $\mathsf{A}^{\mathrm{gen}}$-proof system.

5.2.1 The Split Proof System

Let l and r be two more annotations denoting *left* and *right* respectively. Let $as\mathcal{L}^{\mathrm{af}}_{\Box}$ be the set $a\mathcal{L}^{\mathrm{af}}_{\Box} \times \{l, r\}$. We refer to formulas in this set as annotated formulas. We call $(\varphi, a, i) \in as\mathcal{L}^{\mathrm{af}}_{\Box}$ a formula in *focus* if $a = f$ and a formula *unfocused* if $a = u$. Moreover, given $a \in \{u, f\}$ and $i \in \{l, r\}$, let φ^a_i denote $(\varphi, a, i) \in as\mathcal{L}^{\mathrm{af}}_{\Box}$. Also, for $i \in \{l, r\}$ and $\Delta \in \mathcal{P}_\omega(as\mathcal{L}^{\mathrm{af}}_{\Box})$, define $\Delta^i := \{(\varphi, a) \mid (\varphi, a, i) \in \Delta\}$. We may denote a split annotated sequent Δ using the notation $\Delta^l \mid \Delta^r$.

$$\frac{}{\top^a_i}\,\mathsf{top} \qquad \frac{}{p^a_i, \overline{p}^b_j}\,\mathsf{ax} \qquad \frac{\varphi^a_i, \psi^a_i, \Gamma}{(\varphi \vee \psi)^a_i, \Gamma}\,\mathsf{or} \qquad \frac{\varphi^a_i, \Gamma \qquad \psi^a_i, \Gamma}{(\varphi \wedge \psi)^a_i, \Gamma}\,\mathsf{and} \qquad \frac{\varphi^a_i, \Gamma}{\Box\varphi^a_i, \Diamond\Gamma}\,\mathsf{box}$$

$$\frac{\varphi[\mu x.\varphi/x]^u_i, \Gamma}{\mu x.\varphi^a_i, \Gamma}\,\mathsf{mu} \qquad \frac{\varphi[\nu x.\varphi/x]^a_i, \Gamma}{\nu x.\varphi^a_i, \Gamma}\,\mathsf{nu} \qquad \frac{\Gamma}{\varphi^a_i, \Gamma}\,\mathsf{wk} \qquad \frac{\varphi^f_i, \Gamma}{\varphi^u_i, \Gamma}\,\mathsf{f} \qquad \frac{\varphi^u_i, \Gamma}{\varphi^f_i, \Gamma}\,\mathsf{u}$$

Figure 5.3: Rule applications for the $\mathsf{A}^{\mathrm{split}}$ system.

Definition 5.24. A $\mathcal{T}^{a\mathcal{L}^{\mathrm{af}}_{\Box}}_{\mathcal{R}_{\mathsf{AFMC}}}$-coalgebra (X, α) is called an $\mathsf{A}^{\mathrm{split}}$-proof if for all $x \in X$ the following node, step, and path conditions are met.

(node) $(r^\alpha \times f^\alpha_p)(x)$ has one of the following forms:

1. $(\mathsf{top}, \{\top^a_i\})$
2. $(\mathsf{ax}, \{q^a_i, \overline{q}^b_j\})$
3. $(\mathsf{or}, \{(\varphi \vee \psi)^a_i\})$
4. $(\mathsf{and}, \{(\varphi \wedge \psi)^a_i\})$
5. $(\mathsf{box}, \{\Box\varphi^a_i\})$
6. $(\mathsf{mu}, \{\mu x.\varphi^a_i\})$
7. $(\mathsf{nu}, \{\nu x.\varphi^a_i\})$
8. $(\mathsf{wk}, \{\varphi^a_i\})$
9. $(\mathsf{f}, \{\varphi^u_i\})$
10. $(\mathsf{u}, \{\varphi^f_i\})$

for $q \in \mathsf{Prop}$, $\varphi, \psi \in \mathcal{L}^{\mathrm{af}}_{\Box}$, $a, b \in \{u, f\}$, and $i, j \in \{l, r\}$.

(step) The step conditions represented in Figure 5.3 are satisfied.

(path) The following path conditions are satisfied:

1. On every infinite $\lhd$-path $(x_i)_{i<\omega}$ from x there are infinitely many i such that $r(x_i) = \mathsf{box}$,
2. On every infinite $\lhd$-path $(x_i)_{i<\omega}$ from x there are cofinitely many i such that $r(x_i)$ is not u or f and $f(x_i)$ contains a formula in focus.

Rather than proving soundness and completeness of this system, we will assume it. Since this is a thesis focused on syntactical proofs, we did not verify this result. We are certain that the proof method by J. Marti and Y. Venema in [MV21] can be applied to this system to achieve soundness and

completeness. This means that the interpolation results we achieve will be a *syntactic* interpolation result rather than a semantic one.

Assumption 5.25 (Soundness and Completeness)**.** *Given sequents* $\Gamma, \Delta \in \mathcal{P}_\omega(\mathcal{L}^{\mathrm{af}}_{\Box})$*,*

$$\vdash_{\mathsf{A}^{\mathrm{split}}} \{\varphi_l^f \mid \varphi \in \Gamma\} \cup \{\varphi_r^f \mid \varphi \in \Delta\} \text{ if and only if } \vDash \bigvee(\Gamma \cup \Delta)$$

where $\vdash_{\mathsf{A}^{\mathrm{split}}}$ *is as defined in Definition 3.10 and* $\vDash$ *denotes semantic validity on Kripke models.*

Theorem 5.26. *Given an* $\mathsf{A}^{\mathrm{split}}$*-proof* (X, α) *of* Δ*, there exists an* $\mathsf{A}^{\mathrm{split}}$*-proof* (Y, β) *of* Δ *such that* $|Y| \leq 2^{|\Delta|_{\mathrm{Cl}}+2}$*.*

Proof. Following the same method of finitization for $\mathsf{A}^{\mathrm{gen}}$-proofs in Section 5.1, we get our result. We get a bound of $2^{|\Delta|_{\mathrm{Cl}}+2}$ rather than $2^{|\Delta|_{\mathrm{Cl}}+1}$ since we have to account for formulas appearing on both the left and right side of a node simultaneously. □

We end this section with the following theorem about split proofs, which will help us when constructing interpolants.

Lemma 5.27. *Given an* $\mathsf{A}^{\mathrm{split}}$*-proof* (X, α) *and* $x, y \in X$ *such that* $x \lhd^* y$*,*

$$\mathsf{Lit}(\overline{f(y)^l}) \cap \mathsf{Lit}(f(y)^r) \subseteq \mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)$$

Proof. Let (X, α) and x, y be as specified. Suppose $q \in \mathsf{Lit}(\overline{f(y)^l}) \cap \mathsf{Lit}(f(y)^r)$ and proceed by induction on $x \lhd^* y$. If $x = y$ we immediately have our result. Suppose $x \lhd z$ and $z \lhd^* y$. By the induction hypothesis we know $q \in \mathsf{Lit}(\overline{f(z)^l}) \cap \mathsf{Lit}(f(z)^r)$ so it remains to show that $q \in \mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)$. This can be accomplished by checking cases on $(r \times f_p)(x)$ as this determines $f(z)$. From the different step conditions it can be seen that no new free variables are introduced when a forward step is made in the proof and that the rule applications of the $\mathsf{A}^{\mathrm{split}}$ system do not negate any formulas. □

5.2.2 Coloring Clusters

Recall Chapter 2 for preliminaries on clusters.

Definition 5.28. Given a proof (X, α) let the clusters of (X, α) refer to the clusters of $(X, \lhd)$. In other words x is in the same cluster as y if $x \lhd^* y$ and $y \lhd^* x$.

Lemma 5.29. *Given a cluster* $C \subseteq X$*,* $x, y \in C$ *and* $j \in \{l, r\}$*, if* $f(x)^j$ *has no formula in focus then* $f(y)^j$ *has no formula in focus.*

Proof. Fix a cluster C, and take $x, y \in C$ and $j \in \{l, r\}$. Suppose $f(x)^j$ has no formula in focus. We proceed by induction on the n such that $x \lhd^n y$. If $n = 0$ we are immediately done as $y = x$. Suppose $n = k + 1$ and $x \lhd^k z \lhd y$. Then by the induction hypothesis $f(z)^j$ has no formula in focus. Checking node and step conditions it is clear that the only case of interest (the only way a formula could be in focus in $f(y)^j$) is if $r(z) = \mathsf{f}$. However, this is not the case since z is on the loop $x \lhd^k z \lhd y \lhd^* x$ so by path (b) it cannot be that $r(z) = \mathsf{f}$. □

Lemma 5.30. *Given a non-trivial cluster* $C \subseteq X$*, there is a formula in focus in* $f(x)^l$ *for all* $x \in C$ *or there is a formula in focus in* $f(x)^r$ *for all* $x \in C$*.*

Proof. To begin with, observe that every element in C must have a formula in focus. This follows since if there were $x \in C$ without a formula in focus, then since $x \lhd^+ x$ there is an infinite path with infinitely many points with a sequent with no formulas in focus, a contradiction to path (b).

Now, if every $x \in C$ has a formula in focus in $f(x)^l$ and in $f(x)^r$ we are done. So suppose the opposite, without loss of generality this means there is some $x \in C$ with no formula in focus in $f(x)^l$. Thus it follows by Lemma 5.29 that for every $y \in C$, $f(y)^l$ has no formulas in focus, so there must be a formula in focus in $f(y)^r$ for every $y \in C$. □

Definition 5.31. Given a $\lhd$-cluster C, we say that C is *magenta* if there is a formula in focus in $f(x)^l$ for every x in C and we say that C is *navy* otherwise. We refer to this as the color of the cluster. Furthermore, let η_C be μ if C is magenta, and ν if C is navy. Likewise given any $x \in X$, let the color of x refer to the color of $C(x)$, and let η_x be $\eta_{C(x)}$.

5.2.3 Finding Interpolants

Definition 5.32 (Free Variables)**.** Let (X, α) be a finite $\mathsf{A}^{\text{split}}$-proof. For every $x \in X$ let $q_x \in \mathsf{Prop}$ be a unique atom such that q_x does not appear in $FV(\bigcup_{x \in X} f(x)) \cup BV(\bigcup_{x \in X} f(x))$ (i.e. q_x does not appear in any of the sequents of (X, α)). Moreover, for every $Y \subseteq X$, let Λ_Y denote the set $\{q_x \mid x \in Y\}$.

Note that since (X, α) is finite, we know that (X, α) only has finitely many propositional variables in its sequents, so we can always find countably many atoms not appearing in (X, α).

Definition 5.33. Given an $\mathsf{A}^{\text{split}}$-proof (X, α) we define a set of equations for interpolation (see Definition 3.14) $\chi : X \to \mathcal{L}^{\text{af}}_{\Box}$ such that χ is defined as follows

1. if $(r \times f_p)(x) = (\mathsf{top}, \{\top^a_l\})$ then $\chi_x := \bot$;
2. if $(r \times f_p)(x) = (\mathsf{top}, \{\top^a_r\})$ then $\chi_x := \top$;
3. if $(r \times f_p)(x) = (\mathsf{ax}, \{q^a_l, \overline{q}^b_l\})$ then $\chi_x := \bot$;
4. if $(r \times f_p)(x) = (\mathsf{ax}, \{q^a_l, \overline{q}^b_r\})$ then $\chi_x := \overline{q}$;
5. if $(r \times f_p)(x) = (\mathsf{ax}, \{q^a_r, \overline{q}^b_l\})$ then $\chi_x := q$;
6. if $(r \times f_p)(x) = (\mathsf{ax}, \{q^a_r, \overline{q}^b_r\})$ then $\chi_x := \top$;
7. if $(r \times f_p)(x) = (\mathsf{and}, \{(\varphi \wedge \psi)^a_l\})$ then $\chi_x := \bigvee_{y \in p^\alpha(x)} q_y$;
8. if $(r \times f_p)(x) = (\mathsf{and}, \{(\varphi \wedge \psi)^a_r\})$ then $\chi_x := \bigwedge_{y \in p^\alpha(x)} q_y$;
9. if $(r \times f_p)(x) = (\mathsf{box}, \{(\Box\varphi)^a_l\})$ then $\chi_x := \Diamond q_y$ where y is the single element in $p^\alpha(x)$;
10. if $(r \times f_p)(x) = (\mathsf{box}, \{(\Box\varphi)^a_r\})$ then $\chi_x := \Box q_y$ where y is the single element in $p^\alpha(x)$;
11. for all other cases, $\chi_x := q_y$ where y is the single element in $p^\alpha(x)$.

We note that χ is well-defined since each of the aforementioned cases on $(r \times f_p)(x)$ are unique and covers all possible cases allowed in an $\mathsf{A}^{\text{split}}$-proof.

The following lemmas will be useful later to prove that our interpolants have the properties we desire.

Lemma 5.34. *Let (X, α) be an $\mathsf{A}^{\mathrm{split}}$-proof. Given any $Y \subseteq X$ and any $x \in X$ and $\eta \in \{\mu, \nu\}$, it follows that $\chi_x \in \mathcal{N}^{\eta}_{\Lambda_Y}$.*

Proof. This follows from Lemma 2.28 since χ_x does not contain any fixpoint formulas for any $x \in X$, and q_y does not occur negatively in χ_x for any $y \in X$. □

Lemma 5.35. *Let (X, α) be an $\mathsf{A}^{\mathrm{split}}$-proof and take $x \in X$ and $q \in \mathsf{Lit}$. If $q \in \mathsf{Lit}(\chi_x)$ then one of the following holds*

1. *$q \in \mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)$,*
2. *$q = q_y$ for some $y \in X$ such that $x \lhd y$.*

Proof. Let (X, α) be an $\mathsf{A}^{\mathrm{split}}$-proof and let $x \in X$ and $q \in \mathsf{Prop}$ be as specified. By taking cases on $(r \times f_p)(x)$ we get our desired result from the definition of χ and f. □

Lemma 5.36. *Let (X, α) be an $\mathsf{A}^{\mathrm{split}}$-proof, given $x, y \in X$, $q_y \in FV(\chi_x)$ if and only if $x \lhd y$.*

Proof. Let (X, α) be an $\mathsf{A}^{\mathrm{split}}$-proof and let $x \in X$ and $q \in \mathsf{Prop}$ be as specified. By taking cases on $(r \times f_p)(x)$ we get our desired result from the definition of χ. □

Recall that our approach to interpolation is to view the interpolant for a node x in an $\mathsf{A}^{\mathrm{split}}$-proof (X, α) as $\sigma(q_x)$ where σ is a solution to the aforementioned set of equations. In fact, interpolants require a stronger vocabulary condition than the one given in Definition 3.15, which says

$$\mathsf{Lit}(\sigma(q_x)) \subseteq \mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)$$

We begin by showing in Lemma 5.37 that for each cluster of a finite $\mathsf{A}^{\mathrm{split}}$-proof, there is something close to a solution for said cluster, and then in Theorem 5.38 we will show how to combine the solutions for each cluster into an overall solution.

Lemma 5.37. *Given a proof (X, α) and a set of equations for interpolation χ as defined above and a cluster $C \subseteq X$, for every $Y \subseteq C$ there is a partial map $\sigma_Y : \Lambda_Y \to \mathcal{L}^{\mathrm{af}}_{\Box}$ such that for all $x \in Y$ the following six statements hold:*

1. *Exactly one of the following hold:*
   (a) *$\sigma_Y(q_x) = \sigma_Y(\chi_x)$,*
   (b) *A loop occurs in $(Y, \lhd_Y)$ and there is a formula φ such that*
   $$\sigma_Y(q_x) = \eta_C q_x.\varphi \quad \text{and} \quad \sigma_Y(\chi_x) = \mathrm{unf}(\eta_C q_x.\varphi).$$
2. *$\mathsf{Lit}(\sigma_Y(q_x)) \subseteq (\mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)) \cup \Lambda_{\lhd[C]\setminus Y}$.*
3. *$\sigma_Y(q_x) \in \mathcal{N}^{\eta_C}_{\Lambda_{C\setminus Y}}$.*
4. *For all $z \in X$, $q_z \in FV(\sigma_Y(q_x))$ iff $z \notin Y$ and there exists $u \in Y$ such that $x \lhd^*_Y u$ and $u \lhd z$.*

5. *For all* $z \in X$*, if* $x \lhd_Y^* z$ *then* $\mathrm{Clos}(\sigma_Y(q_z)) \subseteq \mathrm{Clos}(\sigma_Y(q_x))$.

6. $|\sigma_Y(q_x)|_{\mathrm{Cl}} \leq 3|Y|$.

Proof. We will prove the existence of σ_Y by induction on the size of Y. Proving that this substitution satisfies conditions (1) through (6) requires a lot of small checks, and we provide the proof in the appendix in Lemma A.2.

The base case is immediate. Suppose $Y \neq \emptyset$, we inspect the structure of $(Y, \lhd_Y)$ where $\lhd_Y$ is the restriction of $\lhd$ to Y. We distinguish cases depending on whether there is a node $y \in Y$ such that $y \lhd_Y^+ y$, i.e. whether there is a cycle within Y.

- Suppose that no loops occur in Y. Then since the graph $(Y, \lhd_Y)$ is finite there must exist a leaf y with respect to $\lhd_Y$. Let $Z = Y \setminus \{y\}$. Then, by the induction hypothesis we know that there must be $\sigma_Z : \Lambda_Z \to \mathcal{L}_\Box^{\mathrm{af}}$ with the properties listed above. We define $\sigma_Y : \Lambda_Y \to \mathcal{L}_\Box^{\mathrm{af}}$ by

$$\sigma_Y(q_x) = \sigma_Z(q_x)[\chi_y/q_y].$$

- Suppose now that a loop occurs. By our path condition, there must be a box node on this cycle; let it be y. We define $\sigma_Y : \Lambda_Y \to \mathcal{L}_\Box^{\mathrm{af}}$ by

$$\sigma_Y(q_x) = \sigma_Z(q_x)[\eta_C q_y.\sigma_Z(\chi_y)/q_y]. \qquad \Box$$

Knowing that we can find something close to a solution to the set of equations for each cluster, we can use this to find the overall solution to the set of equations as follows.

Theorem 5.38. *Given a finite* $\mathsf{A}^{\mathrm{split}}$*-proof* (X, α) *and* $x \in X$*, there is* $\sigma : \Lambda_X \to \mathcal{L}_\Box^{\mathrm{af}}$ *such that for all* $x \in X$*, the following hold.*

1. *Exactly one of the following hold:*

   (a) $\sigma(q_x) = \sigma(\chi_x)$,

   (b) *There is a formula* φ *such that* $\sigma(q_x) = \eta_x q_x.\varphi$ *and* $\sigma(\chi_x) = \mathrm{unf}(\eta_x q_x.\varphi)$.

2. $\mathsf{Lit}(\sigma(q_x)) \subseteq \mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)$.

3. $\sigma(q_x) \in \mathcal{L}_\Box^{\mathrm{af}}$.

4. $|\sigma(q_x)|_{\mathrm{Cl}} \leq 3|X|$.

Proof. Take $x \in X$, we want to define $\sigma(q_x)$, we prove this map satisfies the above conditions in Theorem A.3. We know $x \in C$ for some cluster C. We proceed by well-founded induction on $(\mathrm{Clus}(X), \prec)$ where $\prec$ denotes the ordering on clusters defined in Section 2.1

- If C is a leaf in $(\mathrm{Clus}(X), \prec)$, we define

$$\sigma(q_x) := \sigma_C(q_x)$$

  where $\sigma_C(q_x)$ is guaranteed by Lemma 5.37.

- If C is not a leaf then inductively we can assume that $\sigma(q_y)$ is defined and meets the above properties for all y in a higher cluster than x. We define

$$\sigma(q_x) := \sigma_C(q_x)[\sigma(q_y)/q_y \mid y \in \lhd[C] \setminus C]$$

where $\sigma_C(q_x)$ is guaranteed by Lemma 5.37. □

Definition 5.39 (Interpolant)**.** Given a finite $\mathsf{A}^{\mathrm{split}}$-proof (X, α), choose a solution σ to a set of equations for interpolation as guaranteed by Theorem 5.38. We define the *interpolant* for a node $x \in X$, denoted ι_x, to be $\sigma(q_x)$.

5.2.4 Partial Interpolation Proofs

We will now show syntactic interpolation in the $\mathsf{A}^{\mathrm{split}}_{\mathrm{skip}}$ system. We start by determining how we will annotate our interpolants.

Definition 5.40. Given a coalgebra (X, α), let $a : X \to \{u, f\}$ be such that $a(x) = u$ if x is magenta and $a(x) = f$ otherwise. Moreover, let $\overline{a} : X \to \{u, f\}$ be such that $\overline{a}(x) = f$ if x is magenta and $\overline{a}(x) = u$ otherwise.

In the last section we showed how to take a finite $\mathsf{A}^{\mathrm{split}}$-proof (X, α) and find formulas ι_x for each $x \in X$ which we referred to as the *interpolants*. We want to show that we are justified in calling these formulas interpolants by constructing left and right interpolation proofs of $f(x)^l \mid \iota_x^{a(x)}$ and $\overline{\iota_x}^{\overline{a}(x)} \mid f(x)^r$ respectively.

Lemma 5.41. *Given a finite $\mathsf{A}^{\mathrm{split}}$-proof, it holds that for every $x \in X$ there is an $\mathsf{A}^{\mathrm{split}}_{\mathrm{par}}$-proof (L_x, β_x) such that the following conditions hold*

1. *There is a node l_x in L_x such that $f^{\beta_x}(l_x) = f(x)^l \mid \iota_x^{a(x)}$.*
2. *Every non-axiomatic leaf node in L_x is annotated with a sequent of the form $f(y)^l \mid \iota_y^{a(y)}$ for some $y \in p^\alpha(x)$.*
3. *If $r^\alpha(x) = \mathsf{box}$ then L_x has a box rule application on every path from l_x to a non-axiomatic leaf.*
4. *If x is an annotated node then for all $y \in p^\alpha(x) \cap C(x)$ and all nodes $z \in L_x$ such that $r^{\beta_x}(z) = \mathsf{as}$, on every path from l_x to z every node on said path is annotated.*
5. *The relation $\lhd^{\beta_x}$ is well-founded.*

Proof. Take $x \in X$. We will only write out the cases where $(r^\alpha \times f_p^\alpha)(x) = (\mathsf{box}, \Box\varphi_l^a)$ for $a \in \{u, f\}$. These partial proofs will be trees and the root will be the root of the tree. Since x is a box node, x will have one unique premise node y such that $\chi_x = \Diamond q_y$. Theorem 5.38 (1) tells us that exactly one of the following will hold: (1) $\iota_x = \Diamond\iota_y$, (2) x is a magenta node in a loop and there is a formula ψ such that: $\iota_x = \mu q_x.\psi$ and $\Diamond\iota_y = \mathrm{unf}(\mu q_x.\psi)$, or (3) x is a navy node in a loop and there is a formula ψ such that: $\iota_x = \nu q_x.\psi$ and $\Diamond\iota_y = \mathrm{unf}(\nu q_x.\psi)$. We split cases on this.

1. If $\iota_x = \Diamond\iota_y$ then we now consider cases on $a(x)$ and $a(y)$.
   (a) If $a(x) = u$ and $a(y) = u$ then let the proof be given by

$$\frac{\Box^{-1}(f_n(x)^l), \varphi^a \mid \iota_y^u}{f_n(x)^l, \underline{\Box\varphi^a} | (\Diamond\iota_y)^u}\,\mathsf{box}$$

Condition 1 and 3 are clearly met. For Condition 2 suppose x is an annotated node and $y \in C(x)$. Since $a(x) = u$ and $a(y) = u$ we know that both $f(x)$ and $f(y)$ have a formula in focus on the left and no focus or un-focus rules are applied so we have our desired result.

(b) If $a(x) = u$ and $a(y) = f$ then let the proof be given by

$$\frac{\dfrac{\Box^{-1}(f_n(x)^l), \varphi^a \mid \iota_y^f}{\Box^{-1}(f_n(x)^l), \varphi^a \mid \iota_y^u}\,\mathsf{f}}{f_n(x)^l, \underline{\Box\varphi^a} \mid (\Diamond\iota_y)^u}\,\mathsf{box}$$

Condition 1 and 3 are clearly met. Condition 2 holds vacuously since $p^\alpha(x) \cap C(x) = \emptyset$ as $a(x) \neq a(y)$.

(c) If $a(x) = f$ and $a(y) = u$ then let the proof be given by

$$\frac{\dfrac{\Box^{-1}(f_n(x)^l), \varphi^a \mid \iota_y^u}{\Box^{-1}(f_n(x)^l), \varphi^a \mid \iota_y^f}\,\mathsf{u}}{f_n(x)^l, \underline{\Box\varphi^a} \mid (\Diamond\iota_y)^f}\,\mathsf{box}$$

Condition 1 and 3 are clearly met. Condition 2 holds vacuously since $p^\alpha(x) \cap C(x) = \emptyset$ as $a(x) \neq a(y)$.

(d) If $a(x) = f$ and $a(y) = f$ then let the proof be given by

$$\frac{\Box^{-1}(f_n(x)^l), \varphi^a \mid \iota_y^f}{f_n(x)^l, \underline{\Box\varphi^a} \mid (\Diamond\iota_y)^f}\,\mathsf{box}$$

Condition 1 and 3 are clearly met. Condition 2 is also vacuously met as the antecedent is always true; there is a formula in focus at every node in the partial proof, and the rule u and f are never applied.

2. If x is magenta and in a loop and there is a formula ψ such that $\iota_x = \mu q_x.\psi$ and $\Diamond\iota_y = \mathrm{unf}(\mu q_x.\psi)$. Then $a(x) = u$ and also $a(y) = u$ since y must also occur in this loop as it is the unique premise of x and therefore also be in the same cluster as x. The proof will look as follows:

$$\frac{\dfrac{\Box^{-1}(f_n(x)^l), \varphi^a \mid \iota_y^u}{f_n(x)^l, \underline{\Box\varphi^a} \mid (\Diamond\iota_y)^u}\,\mathsf{box}}{f_n(x)^l, \Box\varphi^a \mid \underline{\iota_x^u}}\,\mathsf{mu}$$

Condition 1 and 3 are clearly met, and Condition 2 will also be met since by assumption there will always be a formula in focus on the left side of $f(x)$ and $f(y)$.

3. If x is navy and in a loop and there is a formula ψ such that: $\iota_x = \nu q_x.\psi$ and $\Diamond\iota_y = \mathrm{unf}(\nu q_x.\psi)$ then $a(x) = f$ and also $a(y) = f$ since y must also occur in this loop and therefore also be in the same cluster as x. The proof will look as follows:

$$\frac{\dfrac{\Box^{-1}(f_n(x)^l), \varphi^a \mid \iota_y^f}{f_n(x)^l, \underline{\Box\varphi^a} \mid (\Diamond\iota_y)^f}\,\mathsf{box}}{f_n(x)^l, \Box\varphi^a \mid \underline{\iota_x^f}}\,\mathsf{nu}$$

Condition 1 and 3 are clearly met, and Condition 2 will also be met since there is always a formula in focus on the right side of the sequent. □

Lemma 5.42. *Given a finite* $\mathsf{A}^{\text{split}}$*-proof* (X, α)*, it holds that for every* $x \in X$ *there is an* $\mathsf{A}^{\text{split}}_{\text{par}}$*-proof* (R_x, γ_x) *such that the following conditions hold*

1. *There is a node* r_x *in* R_x *such that* $f^{\gamma_x}(r_x) = \overline{\iota_x}^{\overline{a}(x)} \mid f(x)^r$.
2. *Every non-axiomatic leaf node in* (R_x, γ_x) *is annotated with a sequent of the form* $\overline{\iota_y}^{\overline{a}(y)} \mid f(y)^r$ *for some* $y \in p^\alpha(x)$.
3. *If* $r^\alpha(x) = \mathsf{box}$ *then* (R_x, γ_x) *has a* box *rule application on every path from* r_x *to a non-axiomatic leaf.*
4. *If* x *is an annotated node then for all* $y \in p^\alpha(x) \cap C(x)$ *and all nodes* $z \in R_x$ *such that* $r^{\gamma_x}(z) = \mathsf{as}$*, on every path from* r_x *to* z *every node on said path is annotated.*
5. *The relation* $\lhd^{\gamma_x}$ *is well-founded.*

Proof. Dual to Lemma 5.41. □

5.2.5 Proof Transformations

Given a split coalgebra (X, α) it is easy to check that proof transformation of the aforementioned left and right partial interpolation proofs for x satisfy the node and step conditions for $\mathsf{A}^{\text{split}}_{\text{skip}}$-proofs (see Theorem 4.27 for the GL variant of this claim). We still need to show that the path condition is satisfied. To do this, we need more requirements on our partial proofs. We start with the following lemma about infinite paths in proof transformations which will be useful later.

Lemma 5.43. *Given a finite* $\mathsf{A}^{\text{split}}$*-proof* (X, α)*, and a structured collection of partial* $\mathsf{A}^{\text{split}}$*-proofs* $\{(P_x, \beta_x)_{x \in X}\}$ *then an infinite path in the proof transformation* (Y, β) *(see Definition 3.22) induces an infinite path in* (X, α) *if for every* $x \in X$*, the following conditions are met:*

1. *The relation* $\lhd^{\beta_x}$ *is well-founded,*
2. *For all* $z \in P_x$ *such that* $r^{\beta_x}(z) = \mathsf{as}$*, the chosen* $y \in X$ *such that* $f^{\beta_x}(z) = f^{\beta_y}(p_y)$ *is in* $p^\alpha(x)$.

Proof. Suppose there is an infinite path in (Y, β). This will be a path $(x_1, z_1), (x_2, z_2), \dots$ such that either (1) $x_i \lhd^\alpha x_{i+1}$ or (2) $x_i = x_{i+1}$ and $z_i \lhd^{\beta_x} z_{i+1}$. If there are infinitely many i such that $x_i \lhd^\alpha x_{i+1}$ we are done. So suppose not, then there is some n such that for all $m \geq n$, $x_m = x_n$ and $z_m \lhd^{\beta_{x_n}} z_{m+1}$. Thus we have an infinite increasing sequence in $(P_{x_n}, \lhd_{\beta_{x_n}})$, which is not possible since $\lhd^{\beta_{x_n}}$ is well-founded. □

As a result, we will write infinite paths in a partial proof (Y, β) as

$$(x_0, z_{0,0}), \dots, (x_0, z_{0,m_0}), (x_1, z_{1,0}), \dots, (x_1, z_{1,m_1}), \dots$$

where the induced infinite path in (X, α) is given by $(x_i)_{i<\omega}$. Moreover, note that $z_{i,0}$ for all $i \neq 0$ must be p_{x_i} since partial proofs are specifically connected through the nodes p_x for $x \in X$.

Theorem 5.44. *Given a finite* $\mathsf{A}^{\text{split}}$*-proof* (X, α)*, and a structured collection of partial* $\mathsf{A}^{\text{split}}$*-proof* (P_x, β_x)*, let the root of* P_x *be denoted by* p_x*; the proof transformation* (Y, β) *(see Definition 3.22) is an* $\mathsf{A}^{\text{split}}_{\text{skip}}$*-proof if the following four conditions hold for each* $x \in X$*:*

1. *If* $r^\alpha(x) = \mathsf{box}$ *then* (P_x, β_x) *has a* box *rule application on every path from* p_x *to the non-axiomatic leaves,*
2. *If* x *is an annotated node then for all* $y \in p^\alpha(x) \cap C(x)$ *and all nodes* $z \in P_x$ *such that* $r^{\beta_x}(z) = \mathsf{as}$*, on every path from* p_x *to* z *every node on said path is annotated,*
3. *The relation* $\lhd^{\beta_x}$ *is well-founded,*
4. *For all* $z \in P_x$ *such that* $r^{\beta_x}(z) = \mathsf{as}$*, the chosen* $y \in X$ *such that* $f^{\beta_x}(z) = f^{\beta_y}(p_y)$ *is in* $p^\alpha(x)$*.*

Proof. By the same argumentation as Theorem 4.27 we know that (node) and (step) conditions are met. Take $(x, z) \in Y$, we will show that the (path) conditions are met.

1. Take an infinite path

$$(x_0, z_{0,0}), \ldots, (x_0, z_{0,m_0}), (x_1, z_{1,0}), \ldots, (x_1, z_{1,m_1}), \ldots$$

   in (Y, β) which induces an infinite path $x_1 \lhd^\alpha x_2 \lhd^\alpha \ldots$ in (X, α). We want to show there are infinitely many box nodes. This infinite path in (X, α) will have infinitely many i such that x_i is a box node, so by condition (1) each of these P_{x_i} will have a box rule application somewhere on every path from p_x to the non-axiomatic leaves. Recall that $z_{i,0}$ for $i \neq 0$ must be p_{x_i} meaning by condition (1) that there must exist $j \leq m_i$ such that $r^\beta(x_i, z_{i,j}) = \mathsf{box}$. Thus we can conclude that there are infinitely many box rule applications on this path, giving us our desired result.

2. Take an infinite path

$$(x_0, z_{0,0}), \ldots, (x_0, z_{0,m_0}), (x_1, z_{1,0}), \ldots, (x_1, z_{1,m_1}), \ldots$$

   in (Y, β) which induces an infinite path $x_1 \lhd^\alpha x_2 \lhd^\alpha \ldots$ in (X, α). We want to show there are cofinitely many annotated nodes. Since Y is finite there must be some minimal (i, j) such that $(x_i, z_{i,j})$ repeats infinitely many times along the aforementioned path. This means in particular that x_i repeats infinitely many times on the path in (X, α) so x_n for all $n > i$ (we use $>$ so $n \neq 0$) must be annotated and $x_n \in C(x_i)$. Now we claim that $(x_n, z_{n,k})$ for all $n > i$ and $k \leq m_n$ must be annotated. This will give us our desired result. This follows from Condition (2) by showing that $x_{n+1} \in p^\alpha(x_n) \cap C(x_n)$ which follows since $x_n \lhd x_{n+1}$ and as previously mentioned $x_n, x_{n+1} \in C(x_i)$ so $x_{n+1} \in C(x_n)$.

3. Lastly, we want to show that on every infinite path, the rule skip is not used cofinitely many times. This holds since the rule box is used infinitely many times as shown in (1). □

Our syntactic interpolation result now comes as a corollary.

Theorem 5.45 (Syntactic Interpolation)**.** *Given a finite* $\mathsf{A}^{\text{split}}$*-proof* (X, α)*, there are* $\mathsf{A}^{\text{split}}_{\text{skip}}$*-proofs* (Y, β) *and* (Z, γ) *which prove* $f(x)^l \mid \iota_x^{a(x)}$ *and* $\overline{\iota_x}^{\overline{a}(x)} \mid f(x)^r$ *for each* $x \in X$*.*

Proof. We will prove there is a $\mathsf{A}^{\mathrm{split}}_{\mathrm{skip}}$-proof (Y, β) of $f(x)^l \mid \iota_x^{a(x)}$ for each $x \in X$. The right syntactic interpolation proof follows analogously. By Lemma 5.41 there is a structured collection of $\mathsf{A}^{\mathrm{split}}_{\mathrm{par}}$-proofs $\{(L_x, \beta_x)\}_{x \in X}$. By Theorem 5.44 there is a proof transformation (Y, β) which is an $\mathsf{A}^{\mathrm{split}}_{\mathrm{skip}}$-proof. Take $x \in X$, it remains to show $(Y, \beta) \vdash_{\mathsf{A}^{\mathrm{split}}_{\mathrm{skip}}} f(x)^l \mid \iota_x$. This is witnessed by $(x, l_x) \in Y$ since by Lemma 5.41, $f^{\beta_x}(l_x) = f(x)^l \mid \iota_x^{a(x)}$, so $f^{\beta}(x, l_x) = f(x)^l \mid \iota_x^{a(x)}$. □

Combining Theorem 5.45 with a proof of Assumption 5.25 results in Lyndon interpolation for the entire alternation-free μ-calculus. This follows by observing that every alternation-free formula is semantically equivalent to a guarded alternation-free formula [MV21]. In particular, by Theorem 5.38 the guarded fragment would have interpolants whose size is bounded exponentially.

Chapter 6

Formalization in Lean

This chapter presents our formalization of the theoretical results of Chapter 4 in the functional programming language and interactive theorem prover Lean 4.

- The code is available on GitHub at: `github.com/mgignoux/lean4-gl-coalgebras`
- The documentation is available at: `mgignoux.github.io/lean4-gl-coalgebras/docs`

As mathematical proofs grow in complexity, interactive theorem provers have grown in popularity as a tool for ensuring correctness. Lean is particularly well-suited to proof theory: as a functional programming language it provides a natural means for defining proof systems and their properties; and as a theorem prover it enables fully machine-verified reasoning about them.

Formalizing non-cyclic proof systems in functional programming languages has been explored in [Li20; MP21; BMP26]. Such well-founded systems are naturally expressible using Lean's inductive definitions. However, our focus is on ill-founded proof systems, for which, as previously mentioned, the literature normally distinguishes two kinds of ill-founded proofs: cyclic proofs and infinite tree proofs. To our knowledge, the only existing formalization of an ill-founded proof system in Lean is by M. Gattinger *et al.* (in forthcoming work) for a cyclic proof system for PDL, which follows the work in [Bor+25]. If we expand our scope outside of Lean, the work by H. Férée *et al.* in [Fér+24] formalizes a cyclic proof system in Rocq for GL (amongst other modal logics) to compute uniform interpolants as well as their syntactic correctness proofs. This work does not formalize soundness and completeness for the proof systems, but rather syntactic interpolation. In our literature review, we found no formalization of infinite tree proof systems.

Our work brings the coalgebraic approach to proofs into Lean, replicating the results of Chapter 4 within the theorem prover. This is interesting for two reasons. First, the coalgebraic framework unifies cyclic and infinite tree proofs into a single notion. Second, there is no standardized methodology for representing ill-founded proofs in Lean, making this an open design problem.

We formalize the interpolation results for GL developed in Chapter 4 using this coalgebraic approach. This encompasses soundness and completeness of the GL-proof systems $\mathsf{G}^{\mathrm{gen}}$, $\mathsf{G}^{\mathrm{split}}$, and $\mathsf{G}^{\mathrm{ext}}_{\mathrm{skip}}$, finitization for proofs in $\mathsf{G}^{\mathrm{gen}}$ and $\mathsf{G}^{\mathrm{split}}$, the set of equations for interpolation, the proof that a solution exists, and lastly partial proofs for interpolation and combining them into $\mathsf{G}^{\mathrm{ext}}_{\mathrm{skip}}$-proofs of interpolant correctness. The import tree of our work in Lean is shown in Figure 6.1.

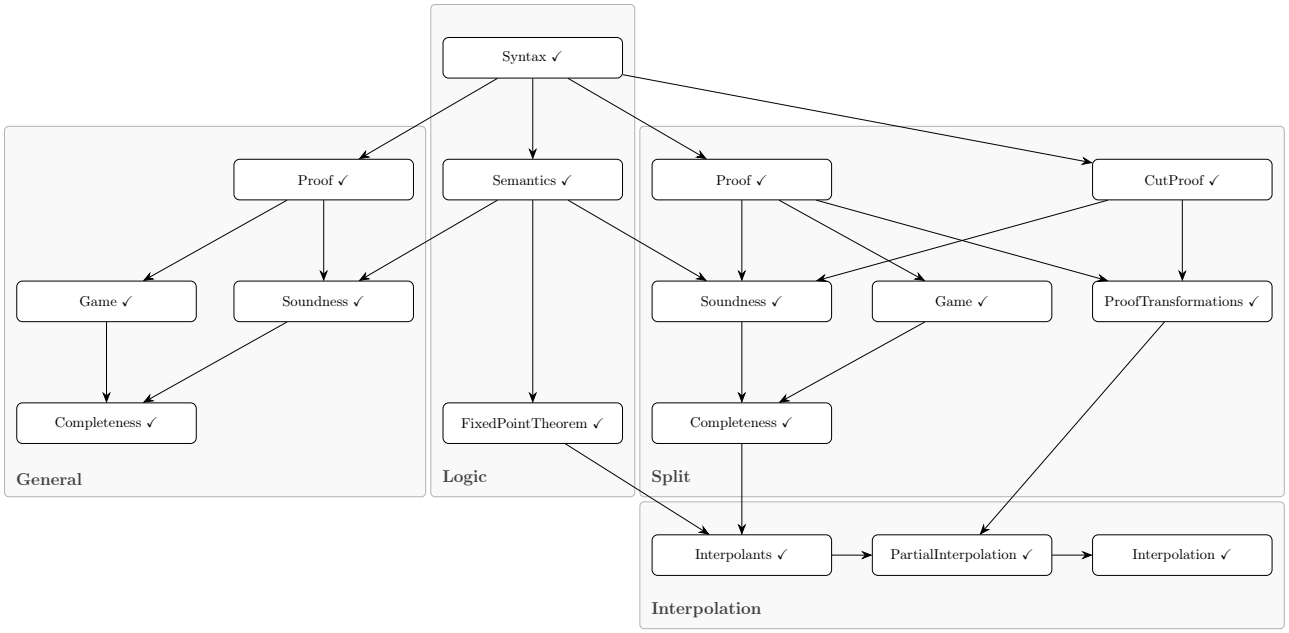

Figure 6.1: Structure and module dependency of our formalization work.

The import tree in Figure 6.1 reflects the logical dependency structure of our formalization. The foundation is the group **Logic**, which defines the syntax and semantics of GL together with the fixpoint theorem. On top of this base, the group **General** establishes the core results for $\mathsf{G}^{\mathrm{gen}}$: finitization, soundness, and completeness. The module group **Split** then introduces the split sequent systems $\mathsf{G}^{\mathrm{split}}$ and $\mathsf{G}^{\mathrm{ext}}$, along with their partial and skip proof variants $\mathsf{G}^{\mathrm{ext}}_{\mathrm{skip}}$ and $\mathsf{G}^{\mathrm{ext}}_{\mathrm{par}}$. Finally, the group **Interpolation** assembles these components to define interpolants, construct partial interpolation proofs, and prove the interpolation theorem.

The structure of this chapter is as follows. In Section 6.1 we give an introduction to Lean with examples from our formalization. In the next three sections we discuss key design choices we made in Lean, which differ from or are not discussed in the mathematical contents of this thesis. In Section 6.2 we discuss how we defined proofs as coalgebras in Lean, in Section 6.3 we discuss the completeness proof, and in Section 6.4 we discuss partial proofs. Finally, in Section 6.5 we evaluate our formalization and in Section 6.6 we discuss future work.

6.1 Introduction to Lean

We give a brief introduction to Lean 4 below; for a thorough treatment, see [Mac24; Baa+25]. Lean 4 is an interactive theorem prover and functional programming language developed by Leonardo de Moura and others, with its internal logic grounded in the calculus of constructions.

Lean 4 uses several keywords to construct types, define functions, and state theorems. We briefly discuss the most important ones. Lean comes with a collection of built-in libraries defining basic notions such as natural numbers, lists, and strings. It also has a large community library, mathlib, which contains a broad base of formalized mathematics including the theory of functors and coalgebras that we need [Mat20].

In a fresh project, we import mathlib and begin by defining the objects we need, which in Lean's type-based system means defining new types. For instance, we need to define the type of BML formulas

to be able to reason about formulas. To define new types, the keywords `inductive` and `structure` are used. The keyword `inductive` allows new types to be defined via inductive clauses. An appropriate first example is defining $\mathcal{L}_{\Box}$ from Definition 2.1.

```
inductive Formula : Type
  | bottom : Formula
  | top : Formula
  | atom : Nat → Formula
  | neg_atom : Nat → Formula
  | and : Formula → Formula → Formula
  | or : Formula → Formula → Formula
  | box : Formula → Formula
  | diamond : Formula → Formula
```

We also add some notation to make our constructors easier to read.

```
prefix:70 "at" => atom
prefix:70 "na" => neg_atom
prefix:70 "□" => box
prefix:70 "◇" => diamond
infixr:6 "&" => and
infixr:6 "v" => or
@[simp] instance instBot : Bot Formula where bot := Formula.bottom
@[simp] instance instTop : Top Formula where top := Formula.top
```

The keywords `prefix`, `infixr`, `infixl` allow us to define custom syntax for functions, which in this case we use for the different constructors for `Formula`. This allows us to write φ `v` ψ in our code rather than `Formula.or` φ ψ, improving readability.

The keyword `instance` allows us to register type class instances. Specifically, `Bot Formula` and `Top Formula` allow us to use the unicode characters $\bot$ and $\top$ in place of writing `Formula.bottom` and `Formula.top` respectively. The marker `@[simp]` is referred to as an attribute, and will be explained in more detail later.

For types with a single constructor bundling multiple fields, the keyword `structure` is used instead. For instance, consider defining a endofunctor `F` on a category `C`. Those familiar with category theory know this is a four-part definition requiring the object map, the morphism map, and proofs that the endofunctor preserves the structure of the identity map and composition. A possible definition using `structure` looks as follows.

```
structure Endofunctor (C : Type) [Category C] where
  obj : C → C
  map : ∀ {X Y : C}, (X ⟶ Y) → ((obj X) ⟶ (obj Y))
  map_id : ∀ X : C, map (𝟙 X) = 𝟙 (obj X)
  map_comp : ∀ {X Y Z : C} (f : X ⟶ Y) (g : Y ⟶ Z), map (f ≫ g) = map f ≫ map g
```

Remark 6.1. This example is a rewrite of Mathlib's definition of a functor. It uses some notation we will not discuss or use, but those familiar with category theory may see a relation to the mathematical definition of an endofunctor. This definition also ignores type universes: in Lean 4, types are partitioned into universes `Type 0`, `Type 1`, and so on to avoid paradoxes, and Mathlib's definition functor works at any universe level. Since all types we use in our thesis live in `Type 0` (also written as `Type`), we simply write `Type` throughout.

Functions are defined using the keywords `def` or `abbrev`: `def` is the standard keyword, whereas `abbrev` is used for definitions that should unfold automatically. For instance, we use the abbreviation `abbrev Sequent := Finset Formula`. An important function that we will need to define for our type `Formula` is its negation `neg`:

```
@[simp] def neg : Formula → Formula
  | ⊥ => ⊤
  | ⊤ => ⊥
  | at n => na n
  | na n => at n
  | φ & ψ => (neg φ) v (neg ψ)
  | φ v ψ => (neg φ) & (neg ψ)
  | □ φ => ◇ (neg φ)
  | ◇ φ => □ (neg φ)

prefix:50 "~" => Formula.neg
```

So far we have looked at the functional programming capabilities of Lean, but now, we combine this with Lean's theorem proving capabilities using the keywords `theorem` and `lemma`. An example proof of the statement $\overline{\overline{\varphi}} = \varphi$ for all $\varphi \in \mathcal{L}_{\Box}$ follows.

```
lemma neg_neg_eq (φ : Formula) : (~~φ) = φ := by
    induction φ
    all_goals
      simp
      try grind
```

In Lean 4, proofs may be constructed in two distinct modes: term mode and tactic mode. In both modes, Lean constructs proofs as functions. Term mode precisely specifies the function constituting the proof, which is more direct, but can be harder to work with in practice. Tactic mode on the other hand, invoked by the keyword `by` as seen above, provides access to Lean's extensive collection of tactics for simplifying and closing goals.

The proof above proceeds in tactic mode as follows: the tactic `induction` splits φ into its different constructors and supplies the standard induction hypotheses, with each case becoming a new goal; `all_goals` instructs Lean to handle all open goals at once; `simp` simplifies each goal using a collection of lemmas marked `@[simp]` appearing previously in the file and from the libraries imported into the project; and finally the `grind` tactic successfully closes the remaining goals using a multitude of techniques found in SMT solvers [Chr+26, 16. The `grind` tactic].

6.2 Proofs as Coalgebras in Lean

Having defined formulas, our next step is to give a coalgebraic definition of proof in Lean. In Chapter 4 we encountered four different proof systems for GL. For now, we focus on the most basic system $\mathsf{G}^{\mathrm{gen}}$; the others follow the same pattern, with the exception of $\mathsf{G}^{\mathrm{ext}}_{\mathrm{par}}$, which we discuss later. Here we had to make a key design choice to define our endofunctor $\mathcal{T}^{\mathcal{L}_{\Box}}_{\mathcal{R}_{\mathsf{GL}}}$. Recall the definition of the endofunctor used

for G^{gen}-proofs was given by

$$\mathrm{ob}(\mathsf{Set}) \ni X \mapsto \mathcal{R}_{\mathsf{GL}} \times \mathcal{P}_\omega(\mathcal{L}_\Box) \times \mathcal{P}_\omega(\mathcal{L}_\Box) \times \mathcal{P}_\omega(X)$$
$$\mathrm{mor}(\mathsf{Set}) \ni f \mapsto I_{\mathcal{R}_{\mathsf{GL}}} \times I_{\mathcal{P}_\omega(\mathcal{L}_\Box)} \times I_{\mathcal{P}_\omega(\mathcal{L}_\Box)} \times \mathcal{P}_\omega(f)$$

In Lean, we could recreate this endofunctor directly using Lean's built-in notion of finite sets `Finset` α (finite sets with elements of type α) as follows:

```
import Mathlib.CategoryTheory.Endofunctor.Algebra

@[simp] def T : CategoryTheory.Functor Type Type where
  obj := fun X ↦ String × Finset Formula × Finset Formula × Finset X
  map := fun f ⟨r, f_prin, f_non, A⟩ ↦ ⟨r, f_prin, f_non, A.map f⟩
  map_id := by aesop_cat
  map_comp := by aesop_cat
```

This is a direct parallel of our definition of the endofunctor given in Definition 3.4, albeit applied directly to $\mathcal{L}_\Box$ rather than arbitrary formula types. This approach uses `String` to identify the rule labels "and", "or", etc. This definition is not as easy to work with in Lean as it is to reason about on paper. For example, suppose that on paper we know that the rule label for a node x in some proof is and, i.e. $r(x) = \mathsf{and}$. On paper we know from our node conditions that the principal formula will have the form $\varphi \wedge \psi$ for some unique φ and ψ and we can immediately begin reasoning using these formulas. In Lean this reasoning is cumbersome although still valid. To demonstrate this, consider formalizing the condition for when the rule application is and. The node condition in Definition 4.2 would read something like the following.

(node) If $r(x) = \mathsf{and}$ then $\exists \varphi, \psi$ s.t. $f_p(x) = \{\varphi \wedge \psi\}$.

Suppose one knows $r(x) = \mathsf{and}$; then all one knows is the existential statement $\exists \varphi, \psi$ s.t. $f_p(x) = \{\varphi \wedge \psi\}$, but, without choice, this statement does not contain the data of said φ and ψ, nor does it tell us that φ and ψ are the unique formulas such that $f_p(x) = \{\varphi \wedge \psi\}$ (even though they provably are). Thus working with this definition in practice would mean using choice whenever we want to extract the principal formula and it would mean dealing with issues of uniqueness. Overall, this undermines the simplicity that our notion of a rule application should have. Similarly, defining rules as strings or other tokens is unnatural in Lean, since it makes case-splitting on the rule label inconvenient.

Fortunately, Lean's inductive definitions resolve this issue, making rule applications extremely natural to work with. These allow us to encode all node conditions directly into an inductive type, which we call `RuleApp`:

```
inductive RuleApp
  | top : (Δ : Sequent) → ⊤ ∈ Δ → RuleApp
  | ax  : (Δ : Sequent) → (n : Nat) → (at n ∈ Δ ∧ na n ∈ Δ) → RuleApp
  | and : (Δ : Sequent) → (φ : Formula) → (ψ : Formula) → (φ & ψ) ∈ Δ → RuleApp
  | or  : (Δ : Sequent) → (φ : Formula) → (ψ : Formula) → (φ v ψ) ∈ Δ → RuleApp
  | box : (Δ : Sequent) → (φ : Formula) → (□ φ) ∈ Δ → RuleApp
```

`RuleApp` is a new type with five constructors, each corresponding to a rule application shown in Figure 4.2. These are very straightforward to work with, as given `r : RuleApp` we can use `cases r` in tactic mode (or `match r` in term mode) to distinguish rule applications and get the data of the principal formulas.

As a slight adaptation, we now must define our maps f, f_p, and f_n as maps from `RuleApp` $\to$ `Sequent`, rather than projections of the structure morphism of a given coalgebra.

Similarly, for the on-paper work in this thesis we viewed the premises of a node x as a finite subset of the base set X of a coalgebra. For the same reason that we want to computably extract the data of the premise nodes, we here use `List X` instead of `Finset X` for the type of our collection of premise nodes. This allows for operations such as the set of equations in Definition 4.18 to remain computable in Lean. Thus we define our endofunctor `T` as follows:

```
@[simp] def T : CategoryTheory.Functor Type Type where
  obj := fun X ↦ (RuleApp × List X)
  map := fun f ⟨r, A⟩ ↦ ⟨r, A.map f⟩
  map_id := by aesop_cat
  map_comp := by aesop_cat
```

This endofunctor behaves exactly as in Definition 3.4, except that the principal and non-principal formulas are now contained in the first projection (the `RuleApp`). Now, given a map α `: X →` `T.obj X`, we can define projections `r : X → RuleApp := fun x ↦ (`α `x).1` and `p : X → RuleApp := fun x ↦` $(\alpha$ `x).2`, telling us how to label the rule application of a node using `r` and the premises of the node using `p`. We also define our $\Box_4^{-1}$ operator in Lean and call it `D`. Using these, we define our Lean version of G^{gen}-proofs given in Definition 4.2 as follows:

```
structure Proof where
  X : Type
  α : X → T.obj X
  step : ∀ (x : X), match r α x with
    | RuleApp.top _ _ => p α x = []
    | RuleApp.ax _ _ _ => p α x = []
    | RuleApp.and _ φ ψ _
        => (p α x).map (fun x ↦ f (r α x)) = [(f_n (r α x)) ∪ {φ}, (f_n (r α x)) ∪ {ψ}]
    | RuleApp.or _ φ ψ _ => (p α x).map (fun x ↦ f (r α x)) = [(f_n (r α x)) ∪ {φ, ψ}]
    | RuleApp.box _ φ _ => (p α x).map (fun x ↦ f (r α x)) = [D (f_n (r α x)) ∪ {φ}]
```

This definition of `Proof` is convenient to work with: one can split cases on the rule application to extract all necessary information about which rule was applied and the data of the principal formulas. Moreover, the node condition of Definition 4.2 becomes unnecessary, as it is built into the type `RuleApp`. What remains are the step and path conditions. The step conditions are encoded in `step` and mirror those given in Definition 4.2. Since the system G^{gen} carries no path conditions, none appear here, although for systems that do, such as $\mathsf{G}^{\text{ext}}_{\text{par}}$ and $\mathsf{G}^{\text{ext}}_{\text{skip}}$, we add these conditions as a separate field in the definition of those systems.

Furthermore, as noted in Lemma 4.4, the system G^{gen} implicitly enforces the infinite path condition that the box rule is applied infinitely often on every infinite path. This condition is embedded in the proof system itself, and establishing it is needed for soundness. This is an example of one of many theorems we can and need to prove in Lean. Formalized as a statement in Lean, this condition reads as follows:

```
theorem inf_path_has_inf_boxes {𝕏 : Proof} (g : ℕ → 𝕏.X)
  (h : ∀ n, edge 𝕏.α (g n) (g (n + 1))) : ∀ n, ∃ m, (r 𝕏.α (g (n + m))).isBox := ...
```

Translating this back into mathematical language, this condition says: For every G^{gen}-proof $\mathbb{X} = (X, \alpha)$ and function $g : \mathbb{N} \to X$, given the hypothesis h which says "for all n, $g(n) \lhd g(n+1)$", then for all n

there exists m such that $r^\alpha(g(n+m)) = \mathsf{box}$.

This is exactly what we mean when we say "on every infinite $\lhd$-path $(x_i)_{i<\omega}$ there are infinitely many i such that $r(x_i) = \mathsf{box}$". In Lean's formalism, we use functions $g : \mathbb{N} \to X$ such that for all n, $R(g(n), g(n+1))$ to represent infinite R-chains. There are also multiple ways to express that there are infinitely many $i < \omega$ such that $r(x_i) = \mathsf{box}$. For instance, we could instead say there is $A \subseteq \mathbb{N}$ such that A is infinite and for all $n \in A$, we have $r^\alpha(g(n)) = \mathsf{box}$. Since Lean is built on inductively defined types such as $\mathbb{N}$, it is easier to use the property "for all n, there exists $m \geq n$ such that $r^\alpha(g(n+m)) = \mathsf{box}$", as this is a statement about $\mathbb{N}$ rather than about an infinite set A.

6.3 Completeness

Beyond formalizing the GL material discussed in this thesis, several results from the literature required formalization, as no existing Lean formalizations were available. Notable examples include the special cases of the fixpoint theorem stated in Lemma 2.11, as well as soundness and completeness for the various coalgebraic proof systems for GL that we formalized. To prove the modal cases of the fixpoint theorem, we followed the construction in [Rei89] and to achieve soundness results we formalized the proof of the soundness of D. Shamkanov's system by G. Menéndez Turata in [Men24, Prop. 2.2.8]. For these two theorems, the main ideas of the existing proofs translated cleanly to our coalgebraic setting in Lean. The completeness theorem, however, required a different approach.

D. Shamkanov's completeness proof for the cyclic proof system for $\mathsf{GL}_{\mathrm{cyc}}$ employs ultrametric spaces as a means to translate proofs in $\mathsf{GL}_{\mathrm{cyc}}$ into proofs in the finite-tree proof system GLSC shown in Figure 4.1. Although we believe the coalgebraic notion of proof may work well with the ultrametric space transformation, as ultrametric spaces are themselves a coalgebraic concept, this still would require defining the finite system GLSC in Lean and proving its completeness. This runs counter to the goals of this thesis, which is to explore what is directly achievable with our coalgebraic notion of proof. Therefore, rather than following this approach, we adopt a game-theoretic argument for completeness similar to that for PDL in [Bor+25], and alternation-Free μ-calculus in [MV21]. This establishes completeness in a direct manner, without requiring an intermediate step of translating the proof into a finite system.

The game-theoretic approach explored in [Bor+25; MV21] uses builder-prover games to show completeness. Game theory is not formalized in Lean's community library mathlib, but multiple attempts at formalizing game-theoretic results in Lean 4 can be found on GitHub. Specifically, N. Cohen formalized terminating two-player games and proved Zermelo's theorem. This formalization was developed for the PDL interpolation project by M. Gattinger *et al.* [Bor+25]. Since it is sufficiently general, we reuse it for our own work.

The game works as follows: two players Prover (she/her) and Builder (he/him) take turns playing moves. Prover plays a move of `RuleApp` and Builder plays a move of type `Sequent`. Specifically, Prover receives a sequent and returns a valid rule application in her attempt to construct the proof, and Builder responds by choosing a sequent matching the premises of that rule application in his attempt to build a countermodel. The game $\mathbb{G}$ is initialized at a sequent Δ, denoted $\mathbb{G}@\Delta$. The available moves of the game look as follows:

Position	Player	Available Moves
Γ with $\top \in \Gamma$	Prover	$(\mathsf{top}, \{\top\}, \Gamma)$
Γ with $q, \overline{q} \in \Gamma$	Prover	$(\mathsf{ax}, \{q, \overline{q}\}, \Gamma)$
Γ with $\varphi \vee \psi \in \Gamma$	Prover	$(\mathsf{or}, \{\varphi \vee \psi\}, \Gamma)$
Γ with $\varphi \wedge \psi \in \Gamma$	Prover	$(\mathsf{and}, \{\varphi \wedge \psi\}, \Gamma)$
Γ with $\Box\varphi \in \Gamma$	Prover	$(\mathsf{box}, \{\Box\varphi\}, \Gamma)$
$(\mathsf{top}, \{\top\}, \Gamma)$	Builder	$\emptyset$
$(\mathsf{ax}, \{q, \overline{q}\}, \Gamma)$	Builder	$\emptyset$
$(\mathsf{or}, \{\varphi \vee \psi\}, \Gamma)$	Builder	$(\Gamma \setminus \{\varphi \vee \psi\}) \cup \{\varphi, \psi\}$
$(\mathsf{and}, \{\varphi \wedge \psi\}, \Gamma)$	Builder	$(\Gamma \setminus \{\varphi \wedge \psi\}) \cup \{\varphi\}, (\Gamma \setminus \{\varphi \wedge \psi\}) \cup \{\psi\}$
$(\mathsf{box}, \{\Box\varphi\}, \Gamma)$	Builder	$\Box_4^{-1}(\Gamma) \cup \{\varphi\}$

We present these results in mathematical notation rather than Lean's syntax. The underlying approach is identical but the latter may be harder to parse for those unfamiliar with Lean. The main difference, as discussed in the previous section, is that labels of a proof are denoted `RuleApp.and` φ ψ Γ `in_`Γ rather than $(\mathsf{and}, \varphi \wedge \psi, \Gamma)$. Additionally, since the formalization by N. Cohen does not include history, we augment each move with a list of all prior moves. These are the only changes we make.

There are three conditions for the game ending:

- Prover has no available moves: Builder wins,
- Builder has no available moves: Prover wins,
- A sequent repeats itself: Prover wins.

Zermelo's theorem tells us that if the move relation, denoted $\lessdot$ is well-founded, the game is determined, i.e. from any game position, one player will always have a winning strategy. Thus we show (1) the game is well-founded, (2) if Prover has a winning strategy then there exists a proof of Γ, and (3) if Builder has a winning strategy then there exists a model refuting Γ. Completeness then comes as a corollary.

The first result follows from the observation that any infinite run of the game would contain a sequent repeating infinitely often, which is impossible since Prover wins immediately upon any repeat. We will provide a sketch of the results (2) and (3).

Theorem 6.2 (`prover_win_builds_proof` ✓)**.** *If Prover has a winning strategy $\mathcal{S}$ in the game $\mathbb{G}@\Delta$, then there exists a proof of Δ.*

Proof. (Sketch.) We follow a four-step process: (1) define the set of nodes of the proof (in Lean, the type of nodes), (2) label each node with a rule and principal/non-principal formulas (i.e. define $(r \times f_p \times f_n)(x)$), (3) specify the premises of each node (i.e. define $p(x)$), and (4) verify that these connections satisfy the step condition. To begin with, for (1) let the set X of nodes of the proof be the set of all moves made by Prover in the cone $\mathcal{S}$. These nodes will be rule applications that tell us exactly how to label the node, answering (2). For (3), we define the set of premise nodes to be as follows. Take $x \in X$, this will be a triple $(\mathsf{r}, \Gamma, \Delta)$. We check cases on r, and give two demonstrative cases:

case If $\mathsf{r} = \mathsf{top}$, then x should be an axiomatic node in the proof, so define $p(x) = \emptyset$.

case If $\mathsf{r} = \mathsf{or}$, then Γ will be of the form $\varphi \vee \psi$. Check if the move $(\Gamma \setminus \{\varphi \vee \psi\}) \cup \{\varphi, \psi\}$ has been played before in the run of the game. If it has, then since Prover has a winning strategy, we know

Prover will have responded to this move with a rule application y, so we set $p(x) = \{y\}$. If it has not, then Builder will have played $(\Gamma \setminus \{\varphi \vee \psi\}) \cup \{\varphi, \psi\}$ in response since it is the only available move, and since Prover has a winning strategy, it will have responded with a rule application y, so set $p(x) = \{y\}$.

Finally, for (4) it remains to show that these connections satisfy the step condition, which is inherent in the design of the game. □

Theorem 6.3 (`builder_win_builds_model` ✓)**.** *If Builder has a winning strategy $\mathcal{S}$ in the game $\mathbb{G}@\Delta$, then there exists a model which falsifies Δ.*

Proof. (Sketch.) We say a path π in $\mathcal{S}$ is maximal if the first node in the path is the starting move Δ or a sequent Γ such that the prior move before Γ was a box rule application, and if any other move were to be made, then that move would be a box rule application. Since the game is well-founded these paths will always be finite.

Define the model $\mathcal{M} = (M, R, V)$ as follows: let M be the set of maximal paths in the game, let R be the transitive closure of the relation given by

$$\pi R \tau \text{ if and only if } \text{last}(\pi) \lessdot^2 \text{first}(\tau)$$

and let V be the valuation given by $V(p) = \{\pi \mid p \notin \mathsf{Voc}(\text{last}(\pi))\}$.

To prove that the model falsifies Δ we show the following statement: If $\varphi \in \pi_i$ then $M, \pi \nVdash \varphi$. The proof requires an outer induction on φ and an inner induction on $\text{length}(\pi) - i$. We will cover some interesting cases:

case If $p \in \pi_i$, then $\overline{p} \notin \pi_i$. If $i < \text{length}(\pi)$ then $p \in \pi_{i+1}$ and we can apply our inner induction hypothesis. If $i = \text{length}(\pi)$ then $p \in \mathsf{Voc}(\text{last}(\pi))$, our desired result.

case If $\varphi \vee \psi \in \pi_i$ then if $i < \text{length}(\pi)$ then $\varphi, \psi \in \pi_{i+1}$ or $\varphi \vee \psi \in \pi_{i+1}$, either way, we apply our inner induction hypothesis. If $i = \text{length}(\pi)$ then we have a contradiction, as π is not maximal.

case If $\Box\varphi \in \pi_i$ then if $i < \text{length}(\pi)$ then $\Box\varphi \in \pi_{i+1}$ so we proceed by our inner induction hypothesis. If $i = \text{length}(\pi)$ then define τ to be a maximal path from $\Box_4^{-1}(\pi_i) \cup \{\varphi\}$. (The fact that such an R successor always exists is a corollary of the fact that Builder has a winning strategy). Then $\pi R \tau$ and since $\varphi \in \tau_0$ by our outer inductive hypothesis we know $M, \tau \nVdash \varphi$, our desired result.

case If $\Diamond\varphi \in \pi_i$ then $\Diamond\varphi \in \text{last}(\pi)$. Moreover, given any τ such that $\pi R \tau$, one can verify that $\varphi \in \tau_0$ so $M, \tau \nVdash \varphi$, our desired result. □

6.4 Partial Proofs

Another design choice we made that diverged from our mathematical approach was how to handle partial proofs. In our mathematical setting, partial proofs (see Definition 3.17) are open-ended coalgebras in which any node may be declared an assumption, leaving it as a non-axiomatic leaf. Then, in Definition 3.21, given a $\mathsf{G}^{\text{split}}$-proof (X, α) we added the requirement that for each $x \in X$ there is a partial proof P_x and that the collection of these are structured in a way that allows us to connect them when we eventually do the proof transformation as defined in Definition 3.22. This condition states the following:

- For all $x \in X$ there is a node $p_x \in P_x$ which we refer to as the root node,
- For all $x \in X$ and $z \in P_x$, if $r^{\beta_x}(z) = \mathsf{as}$ then there is $y \in p^\alpha(x)$ such that $f^{\beta_x}(z) = f^{\beta_y}(p_y)$.

However, this condition only tells us that such a node exists but does not carry the data of which node to use for the attachment. Moreover, it misses a key property which says that not only is there another partial proof we can attach to, but we actually know what the formulas should be; precisely the fact that all root nodes have the form $\tau(x)$ for some $\tau : X \to \mathcal{P}_\omega(\mathcal{L}_\Box)$ and all non-axiomatic leaf nodes have the form $\tau(y)$ for some $y \in p^\alpha(x)$. As an example, in the case of the left interpolation correctness proof, the function τ is given by $x \mapsto f(x)^l \mid \iota_x$.

We employ two techniques: first, we combine the data of the aforementioned node y into the rule application itself, much like we do for the principal formulas in the or, and and box rule applications. Additionally, instead of creating the rule application by providing the sequent Δ, we use a function τ : X → SplitSequent which we use to define what the sequent at a as rule application should be (since as is a reserved keyword in Lean, we call the constructor for the as rule application pre in Lean). In Lean, this is expressed as follows.

```
inductive RuleApp {𝕏 : Split.Proof} (x : 𝕏.X) (τ : 𝕏.X → SplitSequent)
  | pre : (y : 𝕏.X) → (y ∈ Split.p 𝕏.α x) → RuleApp x τ
  | ...  (other cases)

def f {𝕏 : Split.Proof} {x : 𝕏.X} {τ : 𝕏.X → SplitSequent} : RuleApp x τ → SplitSequent
  | RuleApp.pre y _ => τ y
  | ...  (other cases)
```

This allows us to define the proof transformation directly: the information about how to connect the partial proof is stored directly inside `RuleApp.pre` rather than supplied via added structural conditions. We also need not store a sequent inside `RuleApp.pre`, since the principal and non-principal formulas are already determined. In other words, all structure needed for the proof transformation is baked directly into the definition of partial proofs, making the transformation computable.

This allows us to define the type of `X` in our proof transformation as `(y : 𝕏.X) × (PartialProof y).X`. A key difficulty arises here: definitional equality of dependent sum types. Two terms ⟨`x_1, z_1`⟩ and ⟨`x_2, z_2`⟩ of type `(y : 𝕏.X) × (PartialProof y).X` are equal if and only if `x_1 = x_2` and `z_1` and `z_2` are heterogeneously equal. Heterogeneous equality (`HEq`) is a weaker form of equality that allows comparing terms of different types; in our case, `z_1 : PartialProof x_1` and `z_2 : PartialProof x_2` have different types. Thus instead of being able to state `z_1 = z_2`, we must use heterogenous equality `z_1 HEq z_2`.

Mathlib's documentation of `HEq` explicitly warns us not to use heterogeneous equality, "*you should avoid using this type if you can*" [Chr+26, 19.4.2. Heterogeneous Equality]. The impact becomes clear when checking the `step` and `path` conditions: whenever a step to a premise is made, the heterogeneous equality must be supplied with an explicit proof. Heterogeneous equality proofs are individually simple but verbose, often spanning multiple lines, and difficult to interpret. They ultimately constitute the majority of the code in the proof of `proofTransformation` and its helper lemmas. In contrast to pen-and-paper mathematics, where this reasoning is immediate, a significant amount of effort is spent solely to prove these equalities.

6.5 Evaluation

We now evaluate our Lean formalization and identify directions for future improvement. The main points of critique fall into two categories: critique of the methodology and critique of the results themselves, with the former impacting the latter.

6.5.1 Evaluation of the Method

The guiding methodology of this thesis is to treat proofs as general, not necessarily well-founded, structures. In the literature, however, proof systems commonly carry some structure, be it an underlying finite tree structure in the case of cyclic proofs, or a tree structure in the case of infinite-tree proofs. As previously mentioned, approaches toward cyclic proofs in Lean have been explored by M. Gattinger *et al.* for [Bor+25]. The project for PDL is considerably larger and concerns a more complex logic and system, but shares several similarities with ours. For instance, the core techniques — the soundness proof, builder-prover games, and using fixpoints to define interpolants — appear in both projects. However, the key difference is that the PDL project often uses well-founded induction on the proof structure to prove these properties, something which is unavailable to us. Another difference is that interpolant correctness is proved semantically rather than syntactically.

Working on this project has highlighted one significant advantage of inductive proof structures: they come equipped with induction hypotheses which are extremely useful. The calculus of constructions is central to Lean's type theory, making Lean a powerful tool for managing the complex inductive hypotheses that arise from cyclic proof structures. This is also a significant difference from on-paper work: in a complex proof system with many rules and annotations it can be hard to keep track of what the inductive hypotheses are. This is a major benefit of Lean, where we can use the `induction` tactic to directly get our hypothesis, and can change a statement about a proof x into one with supplied hypothesis, something which our coalgebraic methodology lacks.

In short, Lean is the perfect medium for presenting the user with a transparent set of assumptions and a clear goal. Suppose we need to prove a statement about all nodes within a proof, such as the statement that we can find interpolants for all nodes. Using induction on the proof structure, the proof obligation reduces to a step-by-step analysis of each rule application together with the corresponding induction hypotheses, which Lean tracks and presents to the user automatically. Our method affords no such luxury. While we can look at which rule was applied, we have no induction hypothesis and often must figure out how to define everything at once.

6.5.2 Evaluation of the Results

We now summarize the significant results achieved in the formalization.

- Modal soundness of our systems $\mathsf{G}^{\mathrm{gen}}$, $\mathsf{G}^{\mathrm{split}}$, and $\mathsf{G}^{\mathrm{ext}}_{\mathrm{skip}}$ (`soundness` ✓, `Split.soundness` ✓, and `ExtSkip.soundness` ✓)
- Modal completeness of our systems GL and $\mathsf{G}^{\mathrm{split}}$ (`completeness` ✓ and `Split.completeness` ✓)
- Given a finite split-proof
  - (Noncomputable) function defining the interpolant of each node (`interpolant`)
  - Syntactic interpolation (`syntactic_interpolation` ✓)

- Craig Interpolation (`interpolation`)

Further supporting results, more specific to our system but potentially valuable for future work, include:

- The definition of our proof systems G^{gen}, $\mathsf{G}^{\text{split}}$, and $\mathsf{G}^{\text{ext}}_{\text{skip}}$ (`Proof`, `Split.Proof`, and `ExtSkip.Proof`)
- The GL fixpoint theorem for modal formulas (`fixed_point_theorem_modal` ✓)
- Finitization for G^{gen} and $\mathsf{G}^{\text{split}}$ (`finite_proof_of_proof` ✓ and `Split.finite_proof_of_proof` ✓)
- Proof transformations (`proofTransformation`)

All of our work is sorry-free (fully verified) and culminates in the final theorem `Interpolation`, which states

```
def isInterpolant (φ ψ χ : Formula) :=
  χ.vocab ⊆ φ.vocab ∩ ψ.vocab ∧ ⊨ (φ ⟶ χ) ∧ ⊨ (χ ⟶ ψ)

theorem interpolation (φ ψ : Formula) : ⊨ (φ ⟶ ψ) → ∃ χ, isInterpolant φ ψ χ
```

This proves the correctness of our interpolant via our (coalgebraic) proof theoretic method and transfers back and forth between validity and provability using soundness and completeness. Our work builds on Lean's built-in libraries and the community library mathlib, as well as the finite game formalization by N. Cohen imported from the PDL project.

We now turn to the shortcomings of the project. We begin with those specific to the GL results. First, although Lean is a programming language, our function for defining the interpolant is noncomputable. This result should not surprise us as it is also noncomputable in our on-paper work; even the set of formulas for interpolation is noncomputable. In Lean, the set of equations itself is computable because we made the to rule applications and premises discussed in 6.2. However, to define the interpolant, we proceed by induction on the subsets of nodes of the proof Y, and without a linear ordering on the type of our nodes to pick a minimal node z to work with, or decidable equality for nodes to take $Y \setminus \{z\}$, we cannot achieve computability. Decidable equality for nodes of a proof is a reasonable assumption to add, but stipulating that the nodes of a proof are linearly ordered seems too arbitrary an addition.

By contrast, in a system that defines proof inductively using history like PDL does, as discussed above, we can expect that the interpolant could be defined recursively in a computable manner. This idea has not yet been explored in Lean since the PDL interpolant definition is also noncomputable. In [Fér+24], computable uniform interpolation for several modal logics including GL was proven in the proof assistant Rocq using a similar approach to PDL that recursively defines proofs using history.

A further shortcoming is that the general notion of proof introduced in Definition 3.8 is not present in our Lean development: each type of proof system, when defined in Lean, constitutes its own structure. This arose for two reasons. First, the general notion of proof was not fully crystallized until late in the thesis, whereas the GL-specific definition of proof was established early on. As a result, the methods used in Lean could in principle be adapted to other proof systems, but no general framework for doing so exists within the codebase.

With these limitations in mind, this project stands as a self-contained formalization of interpolation for GL. Any logic project requiring this result could import it to obtain the theorem per its licensing. However, the result is unlikely to be incorporated into an existing Lean library, as Lean currently has few logic libraries, and those that exist impose standardized methodologies and code conventions.

6.6 Future Work

There are three suggestions for future work that we consider feasible, listed in increasing order of difficulty, followed by a discussion of more ambitious directions.

The two results not formalized in the Lean work were the size bound and the Lyndon interpolation result. Recall that given a $\mathsf{G}^{\text{split}}$-proof (X, α) and $x \in X$, Theorem 4.21 establishes the following properties about an interpolant ι_x:

1. $\mathsf{Lit}(\iota_x) \subseteq \mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)$,
2. $|\iota_x| \leq 3 \times 2^{|X|}$.

In Lean, we only prove Craig interpolation and do not prove the size result, meaning we instead show:

1. $\mathsf{Voc}(\iota_x) \subseteq \mathsf{Voc}(f(x)^l) \cap \mathsf{Voc}(f(x)^r)$.

The reason for this discrepancy is that we looked into the size and Lyndon interpolation results relatively late in the thesis and did not have time to formalize them within Lean. The significance of the size result is limited, since it states the size of an interpolant ι_x is bounded by a double exponential (as $|X| \leq 2^{|\Delta|_{\mathrm{Cl}}+1}$), whereas in the literature there are (non-formalized) results proving interpolants which have single exponential size bounds exist for GL [CKW25]. We believe both the Lyndon interpolation result and the size result could easily be formalized in Lean, with the Lyndon interpolation result being the more interesting result to add.

A second direction, particularly relevant if formalization via the coalgebraic method is to be developed further, would be to formulate a carefully designed general definition of proof along with generalized notions of logics. As noted, our focus was proof-of-concept of the coalgebraic method in formalizing interpolation for GL rather than developing generalized notions of proof applicable to other proof systems. Indeed, the systems G^{gen}, $\mathsf{G}^{\text{split}}$, $\mathsf{G}^{\text{ext}}_{\text{par}}$, and $\mathsf{G}^{\text{ext}}_{\text{skip}}$ were defined separately, rather than using type class instances for more generic notions of proof. The same goes for the endofunctors of the aforementioned proof systems.

Third, other proof systems could be formalized. Natural candidates include simple fixpoint modal logics with similar proof systems whose finitization and interpolation could be handled via the methods discussed in this thesis, such as $\mathsf{S4Grz}$ and $\mathsf{K4Grz}$, along with intuitionistic variants of these systems and of GL. We believe many of these could be formalized in a fairly straightforward manner, given that, via our work for GL, we have now established how to formalize the necessary proof techniques in Lean. Moreover, if this were combined with generalizing the notion of (coalgebraic) proof, large portions of the work could be automated, such as notions of filtration or proof transformation.

Finally, a more ambitious goal would be the formalization of the results of Chapter 5. Formalization of this system presents additional challenges. The first one would be determining the best definition of an alternation-free formula. The mutually inductive definition of AFMC formulas given in Definition 2.25 is self-referential, and it is not obvious how to encode it in Lean. Defining AFMC formulas as a subtype of MC formulas is one option, but this is also not easy to work with when it comes to showing that substitutions of formulas stay alternation-free. Second, μ-calculus semantics must be formalized in Lean to address soundness and completeness, or equivalent game-theoretic semantics must be used. This would require formalizing the necessary infinite game theory or locating an existing formalization. Additionally, μ- and ν-traces, which we have avoided entirely in this thesis, would need to be formalized

for the game-theoretic semantics. Finally, the finitization methodology uses least and greatest fixpoint ordinals. The community library mathlib does appear to contain the relevant formalization of ordinals and the Knaster-Tarski theorem, making us believe this approach to finitization may be feasible in Lean.

Chapter 7

Conclusion

Infinite proofs are often necessary to handle fixpoint logics. In this thesis, we developed a coalgebraic framework for ill-founded proof systems and applied it to the Gödel-Löb provability logic (GL) and the alternation-free μ-calculus (AFMC), pursuing three lines of investigation: *finitization*, *interpolation*, and *formalization*.

7.1 Evaluation

Finitization. Finite objects are necessary for computability, such as computing interpolants. To finitize proofs in our coalgebraic proof systems, we used the well-known finitization method of filtration and took inspiration from D. Kozen's proof of the finite model property for the modal μ-calculus. This approach integrated well with our coalgebraic framework as filtration is a coalgebraic concept. Specifically, the approach gave us an upper bound on the size of proofs for GL and AFMC, where we showed that given a proof of Δ, there exists a proof of Δ whose size is bounded exponentially by $|\Delta|_{\mathrm{Cl}}$. In contrast, finitization bounds for cyclic proof systems are often bounded by formula length, which can be exponentially larger than closure size in the case of the μ-calculus [KMV20] and basic modal logic.

Interpolation. We found interpolants for the logics GL and AFMC by solving a system of equations and proved interpolation syntactically within a proof system with an added skip rule with the stipulation that skip is not used cofinitely often along infinite paths. We saw that an advantage this approach has over the standard approach to cyclic proofs was that it gave us better control over where in the proof our interpolant was defined, without first normalizing our proof; while still being able to relate our interpolant to closure size. Specifically, in the case of guarded AFMC, we got an exponential upper bound on the size of our Lyndon interpolant. While exponential size bounds for Lyndon interpolants are common in the literature, to the best of our knowledge, neither Lyndon interpolation nor the upper bound have previously appeared in the literature for the guarded alternation-free μ-calculus.

Formalization. We were able to formalize the majority of our results about the logic GL in the functional programming language and theorem prover Lean 4, resulting in a formalized proof of Craig interpolation for GL. We provided an analysis of this work in Section 6.5.

As part of this thesis we also made an attempt to apply our coalgebraic approach to a proof system for the modal μ-calculus. To this end, we examined a coalgebraic representation of the proof system Stir, introduced by B. Afshari and G. E. Leigh in [AL16]. This system annotates formulas with control words taken from an infinite set, and uses a complex path condition relating to these control words. The resulting complexity reveals two weaknesses of our methodologies for finitization and interpolation laid out in this thesis when applied to proof systems with more complex path conditions. First, point-generated proofs for Stir do not have a finite number of distinct annotated sequents, so an additional proof step, perhaps using proof transformations to restrict which annotations appear, would be required before filtration could be applied. Second, the complex path condition of Stir prevents filtration as a means for achieving finitization. However, the path condition for this proof system is expressible in the modal μ-calculus meaning D. Kozen's finitization result could be used to achieve finitization (without an upper bound). The path condition also creates difficulties when it comes to interpolation: to this end, B. Afshari and G. E. Leigh in [AL19] do in fact normalize the Stir system first to approach interpolation.

These observations identify where our methodology falls short, and also motivate future work. To obtain proof-size bounds, our approach favors "simpler" path conditions, although we did not look into qualifying what "simpler" by identifying a class of path conditions which do yield finite proofs. In our work for AFMC our method shows how to finitize systems whose path condition requires a designated property to hold infinitely often on every infinite path (recurrence), or cofinitely often on every infinite path (persistence), or both. These are common conditions on proofs in the literature, and we therefore believe the method we supplied could be used in those cases to give size-bounded results for other proof systems. Of course, our finitization result is always limited to systems that have path conditions expressible in the modal μ-calculus. At the far end, systems such as $\mu\mathsf{MALL}^\infty$ in [Bae+22], whose path condition involves backtracking and "bouncing" between premises, would likely require a logic with a backward-looking modality to express, which lies further beyond the reach of the method pursued in this thesis.

7.2 Future Work

Alternation-free μ-calculus. The most immediate open problem is extending the interpolation results of Chapter 5 to the full alternation-free μ-calculus. The focus-style proof system of J. Marti and Y. Venema covers only the guarded fragment, so a natural next step would be to identify an ill-founded proof system for the full AFMC with a similar focus and unfocus mechanism. We believe the finitization and equation-solving approach developed in this thesis would then yield Lyndon interpolation with an exponential bound in interpolant size for the full AFMC — a result that has not yet appeared in the literature.

Modal μ-calculus. A further development of the work would be to apply the methods of this thesis to the full modal μ-calculus. As we discussed previously, we would need to work through the difficulties imposed by the Stir system, which we believe to be possible, or another option would be to base our notion of proof on another system for the modal μ-calculus, for which there are several candidates [AL16].

Uniform interpolation. A direction we did not explore in this thesis but nonetheless believe is applicable to our coalgebraic methodology is *uniform interpolation*. Cyclic proof systems have been used to

establish uniform interpolation for several modal logics [ALM21; HSS25], and it remains an open question whether the coalgebraic methodology could be applied in this setting, and whether it could be used to give bounds on interpolant size.

Cut-elimination. Finally, we consider cut-elimination. In [SSZ24], B. Sierra Miranda *et al.* achieve cut-elimination for an ill-founded proof system for Grz co-recursively by forcing the cut upward in infinite-tree proofs, using a different underlying endofunctor from the one employed here. In [AK25], B. Afshari and J. Kloibhofer establish cut-elimination in a cyclic variant of the focus proof system for AFMC. These results raise the question of whether either approach could be generalized to yield a coalgebraic approach toward cut-elimination, and in particular whether a co-inductive strategy inspired by [SSZ24] can be developed within our general framework.

Bibliography

[AK25] Bahareh Afshari and Johannes Kloibhofer. *Cut-elimination for the alternation-free modal mu-calculus*. Technical report. Oct. 13, 2025. arxiv.org/abs/2510.11293 (cited on page 67).

[AL16] Bahareh Afshari and Graham E. Leigh. *Finitary Proof Systems for Kozen's μ*. Preprint. Dec. 30, 2016. doi.org/10.14760/OWP-2016-26 (cited on page 66).

[AL19] Bahareh Afshari and Graham E. Leigh. "Lyndon Interpolation for Modal μ-Calculus". In: *Language, Logic, and Computation: 13th International Tbilisi Symposium, TbiLLC 2019, Batumi, Georgia, September 16-20, 2019, Revised Selected Papers*. Edited by Aybüke Özgün and Yulia Zinova. Springer International Publishing, 2019, pages 197–213. ISBN: 978-3-030-98479-3 (cited on pages 4, 66).

[ALM21] Bahareh Afshari, Graham E. Leigh, and Guillermo Menéndez Turata. "Uniform Interpolation from Cyclic Proofs: The Case of Modal Mu-Calculus". In: *Automated Reasoning with Analytic Tableaux and Related Methods*. Edited by Anupam Das and Sara Negri. Springer International Publishing, 2021, pages 335–353 (cited on page 67).

[Baa+25] Anne Baanen, Alexander Bentkamp, Jasmin Blanchette, Johannes Hölzl, and Jannis Limperg. *The Hitchhiker's Guide to Logical Verification*. Online textbook. 2025. github.com/lean-forward/logical_verification_2025 (visited on 03/11/2026) (cited on page 52).

[Bae+22] David Baelde, Amina Doumane, Denis Kuperberg, and Alexis Saurin. "Bouncing Threads for Circular and ill-founded Proofs: Towards Compositionality with Circular Proofs". In: *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science*. ACM, Aug. 2, 2022, pages 1–13. ISBN: 978-1-4503-9351-5 (cited on page 66).

[BMP26] Antonella Bilotta, Marco Maggesi, and Cosimo Perini Brogi. "A Modular Framework for Proof-Search via Formalised Modal Completeness in HOL Light". In: *34th EACSL Annual Conference on Computer Science Logic (CSL 2026)*. Edited by Stefano Guerrini and Barbara König. Volume 363. Leibniz International Proceedings in Informatics. Schloss Dagstuhl (Leibniz-Zentrum für Informatik), 2026, 18:1–18:29. ISBN: 978-3-95977-411-6 (cited on page 51).

[Boo94] George S. Boolos. *The Logic of Provability*. Cambridge University Press, 1994 (cited on pages 6, 19).

[Bor+25] Manfred Borzechowski, Malvin Gattinger, Helle Hvid Hansen, Revantha Ramanayake, Valentina Trucco Dalmas, and Yde Venema. *Propositional Dynamic Logic has Craig Interpolation: a tableau-based proof*. Preprint. Mar. 2025. arxiv.org/abs/2503.13276 (cited on pages 4, 51, 57, 61).

[Bor88] Manfred Borzechowski. "Tableau–Kalkül für PDL und Interpolation". German. English translation by Malvin Gattinger, (2020), available at malv.in/2020/borzechowski-pdl. Master's thesis. Freie Universität Berlin, 1988 (cited on page 4).

[Bro06] James Brotherston. "Sequent Calculus Proof Systems for Inductive Definitions". PhD thesis. University of Edinburgh, Nov. 2006 (cited on page 3).

[BRV01] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2001. ISBN: 978-0-521-52714-9 (cited on page 19).

[BS07] Julian Bradfield and Colin Stirling. "Modal mu-calculi". In: *Handbook of Modal Logic*. Edited by Patrick Blackburn, Johan van Benthem, and Frank Wolter. Volume 3. Studies in Logic and Practical Reasoning. Elsevier, 2007. Chapter 12, pages 721–756 (cited on pages 8, 36).

[Chr+26] David Thrane Christiansen et al. *The Lean Language Reference*. for Lean version 4.28.0. 2026. https://lean-lang.org/doc/reference/4.28.0/ (visited on 03/13/2026) (cited on pages 54, 60).

[CKW25] Balder ten Cate, Louwe Kuijer, and Frank Wolter. *The Size of Interpolants in Modal Logics*. Nov. 6, 2025. arxiv.org/abs/2511.04577 (cited on page 63).

[Fér+24] Hugo Férée, Iris van der Giessen, Sam van Gool, and Ian Shillito. "Mechanised Uniform Interpolation for Modal Logics K, GL, and iSL". In: *Automated Reasoning - 12th International Joint Conference, IJCAR 2024, Nancy, France, July 3-6, 2024, Proceedings, Part II*. Edited by Christoph Benzmüller, Marijn J. H. Heule, and Renate A. Schmidt. Lecture Notes in Computer Science. Springer, 2024, pages 43–60 (cited on pages 51, 62).

[HSS25] Sebastijan Horvat, Borja Sierra Miranda, and Thomas Studer. *Uniform interpolation for interpretability logic*. Nov. 3, 2025. arxiv.org/abs/2511.01428 (cited on page 67).

[Jac16] Bart Jacobs. *Introduction to Coalgebra: Towards Mathematics of States and Observation*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2016 (cited on page 13).

[Jun10] Natthapong Jungteerapanich. "Tableau Systems for the Modal μ-calculus". PhD thesis. University of Edinburgh, Nov. 2010 (cited on pages 3, 34).

[Klo26] Johannes Kloibhofer. "Cycles with Annotations: ill-founded Proof Theory of Modal Fixpoint Logics". Ph.D. thesis. Universiteit van Amsterdam, Mar. 13, 2026 (cited on page 3).

[KMV20] Clemens Kupke, Johannes Marti, and Yde Venema. *Size matters in the modal μ-calculus*. Oct. 27, 2020. arxiv.org/abs/2010.14430 (cited on pages 10, 11, 65).

[Koz88] Dexter Kozen. "A Finite Model Theorem for the Propositional μ-Calculus". In: *Studia Logica* 47.3 (1988), pages 233–241 (cited on pages 4, 37).

[Li20] Jiatu Li. *Formalization of PAL·S5 in Proof Assistant*. Dec. 7, 2020. arxiv.org/abs/2012.09388 (cited on page 51).

[Löb55] Martin H. Löb. "Solution of a problem of Leon Henkin". In: *Journal of Symbolic Logic* 20.2 (Mar. 1955), pages 115–118. ISSN: 0022-4812, 1943-5886 (cited on page 5).

[Mac24] Heather Macbeth. *The Mechanics of Proof*. Online textbook. 2024. hrmacbeth.github.io/math2001 (visited on 03/11/2026) (cited on page 52).

[Mae61] Shoji Maehara. “Craig no interpolation theorem”. Japanese. In: *Sugaku* 12 (1961), pages 235–237 (cited on page 16).

[Mat20] The mathlib Community. “The Lean Mathematical Library”. In: *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*. CPP 2020. ACM, Jan. 2020 (cited on page 52).

[Men24] Guillermo Menéndez Turata. “Cyclic Proof Systems for Modal Fixpoint Logics”. Ph.D. thesis. Universiteit van Amsterdam, Jan. 30, 2024. ISBN: 978-94-6473-347-1 (cited on pages 20, 25, 33, 57).

[MP21] Marco Maggesi and Cosimo Perini Brogi. “A Formal Proof of Modal Completeness for Provability Logic”. In: *12th International Conference on Interactive Theorem Proving (ITP 2021)*. Edited by Liron Cohen and Cezary Kaliszyk. Volume 193. Leibniz International Proceedings in Informatics. Schloss Dagstuhl (Leibniz-Zentrum für Informatik), 2021, 26:1–26:18. ISBN: 978-3-95977-188-7 (cited on page 51).

[MV21] Johannes Marti and Yde Venema. *Focus-style proof systems and interpolation for the alternation-free μ-calculus*. Apr. 2021. `arxiv.org/abs/2103.01671` (cited on pages 4, 12, 34, 35, 41, 50, 57).

[NW96] Damian Niwiński and Igor Wałukiewicz. “Games for the μ-calculus”. In: *Theoretical Computer Science* 163.1–2 (Aug. 1996), pages 99–116. ISSN: 0304-3975. `https://doi.org/10.1016/0304-3975(95)00136-0`. `https://doi.org/10.1016/0304-3975(95)00136-0` (cited on page 3).

[Rei89] Lisa Reidhaar-Olson. “A New Proof of the Fixed-Point Theorem of Provability Logic”. In: *Notre Dame Journal of Formal Logic* 31.1 (1989), pages 37–43 (cited on pages 8, 57).

[Sam76] Giovanni Sambin. “An effective fixed-point theorem in intuitionistic diagonalizable algebras”. In: *Studia Logica* 35.4 (Dec. 1976), pages 345–361. ISSN: 1572-8730 (cited on page 8).

[Sha11] Daniyar S. Shamkanov. “Interpolation properties for provability logics GL and GLP”. In: *Proceedings of the Steklov Institute of Mathematics*. Volume 274. 1. Oct. 1, 2011, pages 303–316 (cited on page 19).

[Sha14] Daniyar S. Shamkanov. “Circular Proofs for Gödel-Löb Logic”. In: *Mathematical Notes* 96.3 (Sept. 24, 2014), pages 575–585. ISSN: 0001-4346, 1573-8876 (cited on pages 4, 19, 20, 24, 28).

[SSZ24] Borja Sierra-Miranda, Thomas Studer, and Lukas Zenger. “Coalgebraic Proof Translations for ill-founded Proofs”. In: *Advances in Modal Logic, AiML 2024, Prague, Czech Republic, August 19-23, 2024*. Edited by Agata Ciabattoni, David Gabelaia, and Igor Sedlár. College Publications, 2024, pages 527–548 (cited on pages 3, 67).

[Sti14] Colin Stirling. “A Tableau Proof System with Names for Modal μ-calculus”. In: *HOWARD-60. A Festschrift on the Occasion of Howard Barringer’s 60th Birthday*. Edited by Andrei Voronkov and Margarita Korovina. Volume 42. EPiC Series in Computing. Festschrift. 2014, pages 306–318 (cited on pages 3, 34).

[SV80] Giovanni Sambin and Silvio Valentini. “A Modal Sequent Calculus for a Fragment of Arithmetic”. In: *Studia Logica* 39.2 (June 1980), pages 245–256. ISSN: 0039-3215, 1572-8730 (cited on page 19).

[vBe24] Johan van Benthem. "An Abstract Look at the Fixed-Point Theorem for Provability Logic". In: *Dick de Jongh on Intuitionistic and Provability Logics*. Edited by Nick Bezhanishvili, Rosalie Iemhoff, and Fan Yang. Volume 28. Outstanding Contributions to Logic. Springer International Publishing, 2024, pages 75–88. ISBN: 978-3-031-47921-2 (cited on page 8).

[Ven19] Yde Venema. *Coalgebra and Modal Logic: an introduction*. Unpublished notes. 2019. `staff.science.uva.nl/y.venema/teaching/ml/notes/20191215-cml.pdf` (visited on 02/13/2026) (cited on page 13).

Appendix A

Interpolant Correctness Proofs

We provide the proofs that our interpolants which we define in Section 4.2 and Section 5.2 meet the necessary properties which we claim. We provide proofs of Lemma 4.20, Lemma 5.37, and Theorem 5.38. So that this appendix remains self-contained and does not refer back to previous chapters for the definition of the interpolant, we restate the definition of the interpolant as well as supply the proofs of the claims.

A.1 The Gödel-Löb Provability Logic

Lemma A.1. *Given a finite* $\mathsf{G}^{\mathrm{split}}$*-proof* (X, α) *and a set of equations for interpolation* χ *as defined above, for every* $Y \subseteq X$ *there is a map* $\sigma_Y : \Lambda_Y \to \mathcal{L}_\Box$ *such that for all* $x \in Y$*, the following statements hold:*

1. *One of the following hold:*
   (a) $\sigma_Y(q_x) = \sigma_Y(\chi_x)$,
   (b) $\sigma_Y(q_x) \equiv \sigma_Y(\chi_x)$, $r(x) = \mathsf{box}$, *and* $x \lhd_Y^+ x$.
2. *For all* $z \in X$*, if* $q_z \in \mathsf{Voc}(\sigma_Y(q_x))$ *then* $x \lhd^+ z$.
3. $\mathsf{Lit}(\sigma_Y(q_x)) \subseteq (\mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)) \cup \Lambda_{X \setminus Y}$.
4. $|\sigma_Y(q_x)|_{\mathrm{Cl}} \leq 3 \times 2^{|Y|}$.

Proof. We proceed by induction on the size of Y. The case where $Y = \emptyset$ is vacuously true. Suppose $Y \neq \emptyset$. We now inspect the structure of $(Y, \lhd_Y)$ where $\lhd_Y$ is the restriction of $\lhd$ to Y. We distinguish cases depending on whether there is a node $y \in Y$ such that $y \lhd_Y^+ y$.

Suppose that no loops occur in Y. Then since the graph $(Y, \lhd_Y)$ is finite there must exist a leaf y with respect to $\lhd_Y$. Let $Z = Y \setminus \{y\}$. Then, by the induction hypothesis we know that there must be $\sigma_Z : \Lambda_Z \to \mathcal{L}_\Box$ with the properties listed above. Let $\sigma_Y : \Lambda_Y \to \mathcal{L}_\Box$ be the map given by

$$\sigma_Y(q_x) = \sigma_Z(q_x)[\chi_y / q_y]$$

for all $x \in Y$. Take an arbitrary $x \in Y$, we will prove this map satisfies condition (1) through (4).

1. We want to show that (1a) or (1b) holds. We split cases depending on whether $x = y$.

(a) If $x = y$ then observe the following equalities

$$\begin{aligned}
\sigma_Y(q_y) &= \sigma_Z(q_y)[\chi_y/q_y] && \text{by definition of } \sigma_Y \\
&= q_y[\chi_y/q_y] && \text{since } y \notin Z \\
&= \chi_y && \text{Definition 2.3}
\end{aligned}$$

Moreover

$$\begin{aligned}
\sigma_Y(\chi_y) &= \sigma_Z(\chi_y)[\chi_y/q_y] && \text{by definition of } \sigma_Y \\
&= \chi_y[\chi_y/q_y] && \text{since } \mathsf{Voc}(\chi_y) \cap \Lambda_Y = \emptyset \\
&= \chi_y && \text{since } q_y \notin \mathsf{Voc}(\chi_y)
\end{aligned}$$

So we have our desired result.

(b) If $x \neq y$ then by the induction hypothesis, we know that $\sigma_Z(q_x)$ is $\sigma_Z(\chi_x)$ as there are no loops within $(Y, \lhd_Y)$ so there are no loops within $(Z, \lhd_Z)$. Observe the following equalities

$$\begin{aligned}
\sigma_Y(q_x) &= \sigma_Z(q_x)[\chi_y/q_y] && \text{by definition of } \sigma_Y \\
&= \sigma_Z(\chi_x)[\chi_y/q_y] && \sigma_Z(q_x) = \sigma_Z(\chi_x) \\
&= \sigma_Y(\chi_x) && \text{by definition of } \sigma_Y
\end{aligned}$$

2. Take an arbitrary $z \in X$ and suppose that $q_z \in \mathsf{Voc}(\sigma_Y(q_x))$. Again, we split cases depending on whether $x = y$.

(a) If $x = y$ then $\sigma_Y(q_x) = \chi_y$ so $q_z \in \mathsf{Voc}(\chi_y)$. By Lemma 4.19 we know $y \lhd z$.

(b) If $x \neq y$ then $\sigma_Y(q_x) = \sigma_Z(q_x)[\chi_y/q_y]$ so either (i) $q_z \in \mathsf{Voc}(\sigma_Z(q_x))$ and $q_z \neq q_y$ or (ii) $q_y \in \mathsf{Voc}(\sigma_Z(q_x))$ and $q_z \in \mathsf{Voc}(\chi_y)$. In (i) by the induction hypothesis we know $x \lhd^+ z$ so we are done. In (ii) since $q_z \in \mathsf{Voc}(\chi_y)$ we know $y \lhd z$ by Lemma 4.19 and since $q_y \in \mathsf{Voc}(\sigma_Z(q_x))$ we know by the induction hypothesis that $x \lhd^+ y$ so combining these results we get $x \lhd^+ z$, our desired result.

3. We want to show that

$$\mathsf{Lit}(\sigma_Y(q_x)) \subseteq (\mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)) \cup \Lambda_{X \setminus Y}.$$

We proceed by splitting cases depending on whether $x = y$.

(a) If $x = y$ then $\sigma_Y(q_x) = \chi_y$. Take $q \in \mathsf{Lit}(\chi_y)$, by Lemma 4.19 we know that either (i) $q \in \mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)$ or (ii) $q = q_z$ for some z such that $y \lhd z$. In (i) we immediately have our result. In (ii), we can conclude that $z \notin Y$ as y was chosen to be a leaf in $(Y, \lhd_Y)$, so $q \in \Lambda_{X \setminus Y}$.

(b) If $x \neq y$ then $\sigma_Y(q_x) = \sigma_Z(q_x)[\chi_y/q_y]$. By the induction hypothesis we know that

$$\mathsf{Lit}(\sigma_Z(q_x)) \subseteq (\mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)) \cup \Lambda_{X \setminus Z}, \tag{A.1}$$

since $x \in Z$. Take $q \in \mathsf{Lit}(\sigma_Z(q_x)[\chi_y/q_y])$. Then (i) $q \in \mathsf{Lit}(\sigma_Z(q_x))$ and $q \neq q_y$ or (ii)

$q_y \in \mathsf{Lit}(\sigma_Z(q_x))$ and $q \in \mathsf{Lit}(\chi_y)$. In (i), we get our desired result via (A.1) and basic set theory. In (ii) since $q_y \in \mathsf{Lit}(\sigma_Z(q_x))$ we get $x \lhd^+ y$ and since $q \in \mathsf{Lit}(\chi_y)$ by Lemma 4.19 we get (A) $q \in \mathsf{Lit}(\overline{f(y)^l}) \cap \mathsf{Lit}(f(y)^r)$ or (B) $q = q_z$ for some z such that $y \lhd z$. In (A) we get our desired result since $q \in \mathsf{Lit}(\overline{f(y)^l}) \cap \mathsf{Lit}(f(y)^r)$ and $x \lhd^+ y$ so by Lemma 4.16 we have $q \in \mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)$. In (B) we can conclude that $z \notin Y$ since y was chosen to be a leaf in $(Y, \lhd_Y)$ so $q_z \in \Lambda_{X \setminus Y}$.

4. We want to show that $|\sigma_Y(q_x)|_{\mathrm{Cl}} \leq 3 \times 2^{|Y|}$. We split cases depending on whether $x = y$.

   (a) If $x = y$ then $\sigma_Y(q_y) = \chi_y$, and by the definition of χ_y given in Definition 4.18 we know $|\chi_y|_{\mathrm{Cl}} \leq 3$ so clearly $|\chi_y|_{\mathrm{Cl}} \leq 3 \times 2^{|Y|}$ for $|Y| \geq 1$.

   (b) If $x \neq y$ then $\sigma_Y(q_x) = \sigma_Z(q_x)[\chi_y/q_y]$, thus

$$\begin{aligned}
|\sigma_Y(q_x)|_{\mathrm{Cl}} &= |\sigma_Z(q_x)[\chi_y/q_y]|_{\mathrm{Cl}} \\
&\leq |\sigma_Z(q_x)|_{\mathrm{Cl}} + |\chi_y|_{\mathrm{Cl}} && \text{Lemma 2.9} \\
&\leq 3 \times 2^{|Z|} + 3 && \text{(IH), } |\chi_y|_{\mathrm{Cl}} \leq 3 \\
&\leq 3 \times 2^{|Y|} && |Y| \geq 1,\ |Y| = |Z| + 1
\end{aligned}$$

Suppose now that a loop occurs. By Lemma 4.15 there must be a box node on this cycle, let it be y. Then $\chi_y = \triangle q_{y'}$ where y' is the unique premise of y and $\triangle \in \{\Box, \Diamond\}$. We note for later that y' must be in Y since it is the unique premise of y and we know $y \lhd^+ y$. Let $Z = Y \setminus \{y\}$, by the induction hypothesis we know that there must be $\sigma_Z : \Lambda_Z \to \mathcal{L}_\Box$ that satisfies all properties listed above. Let $\delta \in \{\top, \bot\}$ be $\top$ if $\triangle = \Box$ and $\bot$ if $\triangle = \Diamond$. Recall from Lemma 2.11 that we know $\sigma_Z(\chi_y)[(\sigma_Z(\chi_y)[\delta/q_y])/q_y] \equiv \sigma_Z(\chi_y)[\delta/q_y]$. Let $\sigma_Y : \Lambda_Y \to \mathcal{L}_\Box$ be the map given by

$$\sigma_Y(q_x) = \sigma_Z(q_x)[(\sigma_Z(\chi_y)[\delta/q_y])/q_y].$$

for all $x \in Y$. Take an arbitrary $x \in Y$, we will prove this map satisfies all of the listed conditions.

1. We want to show that either (1a) or (1b) holds. We split cases depending on whether $x = y$.

   (a) If $x = y$ then observe the following equalities

$$\begin{aligned}
\sigma_Y(q_y) &= \sigma_Z(q_y)[(\sigma_Z(\chi_y)[\delta/q_y])/q_y] && \text{definition of } \sigma_Y \\
&= q_y[(\sigma_Z(\chi_y)[\delta/q_y])/q_y] && \text{since } y \notin Z \\
&= \sigma_Z(\chi_y)[\delta/q_y] && \text{Definition 2.3} \\
&\equiv \sigma_Z(\chi_y)[(\sigma_Z(\chi_y)[\delta/q_y])/q_y] && \text{Lemma 2.11} \\
&= \sigma_Y(\chi_y) && \text{definition of } \sigma_Y
\end{aligned}$$

   (b) If $x \neq y$ then $\sigma_Y(q_x) = \sigma_Z(q_x)[(\sigma_Z(\chi_y)[\delta/q_y])/q_y]$. By the induction hypothesis, we know that (i) $\sigma_Z(q_x) = \sigma_Z(\chi_x)$ or (ii) x is a box node on a loop in $(Y, \lhd_Y)$ and $\sigma_Z(q_x) \equiv \sigma_Z(\chi_x)$.

In (i) we get our desired result since

$$\begin{aligned}
\sigma_Y(q_x) &= \sigma_Z(q_x)[(\sigma_Z(\chi_y)[\delta/q_y])/q_y] && \text{definition of } \sigma_Y \\
&= \sigma_Z(\chi_x)[(\sigma_Z(\chi_y)[\delta/q_y])/q_y] && \sigma_Z(q_x) = \sigma_Z(\chi_x) \\
&= \sigma_Y(\chi_x) && \text{definition of } \sigma_Y
\end{aligned}$$

In (ii), we use the fact that substitution preserves equivalence to get our desired result.

$$\begin{aligned}
\sigma_Y(q_x) &= \sigma_Z(q_x)[(\sigma_Z(\chi_y)[\delta/q_y])/q_y] && \text{definition of } \sigma_Y \\
&\equiv \sigma_Z(\chi_x)[(\sigma_Z(\chi_y)[\delta/q_y])/q_y] && \sigma_Z(q_x) \equiv \sigma_Z(\chi_x) \\
&= \sigma_Y(\chi_x) && \text{definition of } \sigma_Y
\end{aligned}$$

2. Take an arbitrary $z \in X$ and suppose that $q_z \in \mathsf{Lit}(\sigma_Y(q_x))$. We split cases depending on whether $x = y$.

   (a) If $x = y$ then $\sigma_Y(q_x) = \sigma_Z(\chi_y)[\delta/q_y]$, so $q_z \in \mathsf{Lit}(\sigma_Z(\chi_y))$ and $z \neq y$. Recall that $\chi_y = \triangle q_{y'}$ where y' is the unique premise of y. Therefore $q_z \in \mathsf{Lit}(\sigma_Z(q_{y'}))$. By the induction hypothesis we know that $y' \lhd^+ z$ as $y' \in Z$ and since $y \lhd y'$ we have our desired result that $y \lhd^+ z$.

   (b) If $x \neq y$ then $\sigma_Y(q_x) = \sigma_Z(q_x)[(\sigma_Z(\chi_y)[\delta/q_y])/q_y]$, thus if $q_z \in \mathsf{Lit}(\sigma_Z(q_x)[(\sigma_Z(\chi_y)[\delta/q_y])/q_y])$ then (i) $q_z \in \mathsf{Lit}(\sigma_Z(q_x))$ and $z \neq y$ or (ii) $q_y \in \mathsf{Lit}(\sigma_Z(q_x))$ and $q_z \in \mathsf{Lit}(\sigma_Z(\chi_y))$ and $z \neq y$. In (i) since $z \neq y$ we can apply the induction hypothesis to $q_z \in \mathsf{Lit}(\sigma_Z(q_x))$ to get $x \lhd^+ z$, our desired result. In (ii) we know by applying the induction hypothesis to $q_y \in \mathsf{Lit}(\sigma_Z(q_x))$ that $x \lhd^+ y$ and since $q_z \in \mathsf{Lit}(\sigma_Z(\chi_y))$ we know by the reasoning in (2a) that $y \lhd^+ z$, so we have $x \lhd^+ z$.

3. We want to show that

$$\mathsf{Lit}(\sigma_Y(q_x)) \subseteq (\mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)) \cup \Lambda_{X \setminus Y}.$$

   We begin by splitting cases depending on whether $x = y$.

   (a) If $x = y$ then $\sigma_Y(q_y) = \sigma_Z(\chi_y)[\delta/q_y]$. Take $q \in \mathsf{Lit}(\sigma_Z(\chi_y)[\delta/q_y])$. As previously mentioned this means $q \in \mathsf{Lit}(\sigma_Z(q_{y'}))$ and $q \neq q_y$. By the induction hypothesis we have

$$\mathsf{Lit}(\sigma_Z(q_{y'})) \subseteq (\mathsf{Lit}(\overline{f(y')^l}) \cap \mathsf{Lit}(f(y')^r)) \cup \Lambda_{X \setminus Z}.$$

   so $q \in (\mathsf{Lit}(\overline{f(y')^l}) \cap \mathsf{Lit}(f(y')^r)) \cup \Lambda_{X \setminus Z}$. We get two cases: (i) $q \in \mathsf{Lit}(\overline{f(y')^l}) \cap \mathsf{Lit}(f(y')^r)$ or (ii) $q \in \Lambda_{X \setminus Z}$. In (i) $q \in \mathsf{Lit}(\overline{f(y')^l}) \cap \mathsf{Lit}(f(y')^r)$ and $y \lhd y'$ so $q \in \mathsf{Lit}(\overline{f(y)^l}) \cap \mathsf{Lit}(f(y)^r)$, our desired result. In (ii) since $q \in \Lambda_{X \setminus Z}$ and $q \neq q_y$ we get our desired result.

   (b) If $x \neq y$ then $\sigma_Y(q_x) = \sigma_Z(q_x)[(\sigma_Z(\chi_y)[\delta/q_y])/q_y]$. By the induction hypothesis we know that

$$\mathsf{Lit}(\sigma_Z(q_x)) \subseteq (\mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)) \cup \Lambda_{X \setminus Z},$$

   since $x \in Z$. Suppose $q \in \mathsf{Lit}(\sigma_Z(q_x)[(\sigma_Z(\chi_y)[\delta/q_y])/q_y])$, as usual, either (i) $q \in \mathsf{Lit}(\sigma_Z(q_x))$ and $q \neq q_y$ or (ii) $q_y \in \mathsf{Lit}(\sigma_Z(q_x))$ and $q \in \mathsf{Lit}(\sigma_Z(\chi_y))$ and $q \neq q_y$. In (i) using the induction

hypothesis we get $q \in (\mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)) \cup \Lambda_{X \setminus Z}$ and since $q \neq q_y$ we get our desired result from basic set theory manipulations. In (ii) since $q_y \in \mathsf{Lit}(\sigma_Z(q_x))$ we get $x \lhd^+ y$ and since $q \in \mathsf{Lit}(\sigma_Z(\chi_y))$ we get by the same reasoning as (3a) that

$$\mathsf{Lit}(\sigma_Z(q_{y'})) \subseteq (\mathsf{Lit}(\overline{f(y')^l}) \cap \mathsf{Lit}(f(y')^r)) \cup \Lambda_{X \setminus Z}$$

so $q \in (\mathsf{Lit}(\overline{f(y')^l}) \cap \mathsf{Lit}(f(y')^r)) \cup \Lambda_{X \setminus Z}$. Therefore (A) $q \in \mathsf{Lit}(\overline{f(y')^l}) \cap \mathsf{Lit}(f(y')^r)$ or (B) $q \in \Lambda_{X \setminus Z}$. In (A) we get our desired result since $x \lhd^+ y$ and $y \lhd y'$ so $q \in \mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)$. In (B) we get our desired result since $q \in \Lambda_{X \setminus Z}$ and $q \neq q_y$.

4. We want to show that $|\sigma_Y(q_x)|_{\mathrm{Cl}} \leq 3 \times 2^{|Y|}$. We split cases depending on whether $x = y$.

   (a) If $x = y$ then $\sigma_Y(q_y) = \sigma_Z(\chi_y)[\delta/q_y]$, and we know $\chi_y = \triangle q_{y'}$ where y' is the unique premise of y and $\triangle \in \{\Box, \Diamond\}$ as well as $y' \in Z$. Putting this all together we have

$$\begin{aligned}
|\sigma_Y(q_x)|_{\mathrm{Cl}} &= |\sigma_Z(\chi_y)[\delta/q_y]|_{\mathrm{Cl}} && \text{definition of } \sigma_Y \\
&\leq |\sigma_Z(q_{y'})|_{\mathrm{Cl}} \\
&\leq 3 \times 2^{|Z|} && \text{(IH)} \\
&\leq 3 \times 2^{|Y|} && |Y| = |Z| + 1
\end{aligned}$$

   (b) If $x \neq y$ then $\sigma_Y(q_x) = \sigma_Z(q_x)[(\sigma_Z(\chi_y)[\delta/q_y])/q_y]$

$$\begin{aligned}
|\sigma_Y(q_x)|_{\mathrm{Cl}} &= |\sigma_Z(q_x)[(\sigma_Z(\chi_y)[\delta/q_y])/q_y]|_{\mathrm{Cl}} && \text{definition of } \sigma_Y \\
&\leq |\sigma_Z(q_x)|_{\mathrm{Cl}} + |\sigma_Z(\chi_y)[\delta/q_y]|_{\mathrm{Cl}} && \text{Lemma 2.9} \\
&\leq 3 \times 2^{|Z|} + 3 \times 2^{|Z|} && \text{(IH)} \\
&= 3 \times 2^{|Y|} && |Y| = |Z| + 1
\end{aligned}$$

□

A.2 The Alternation-Free μ-Calculus

Lemma A.2. *Given a proof (X, α) and a set of equations for interpolation χ as defined above and a cluster $C \subseteq X$, for every $Y \subseteq C$ there is a partial map $\sigma_Y : \Lambda_Y \to \mathcal{L}^{\mathrm{af}}_{\Box}$ such that for all $x \in Y$ the following six statements hold:*

1. *Exactly one of the following hold:*

   (a) $\sigma_Y(q_x) = \sigma_Y(\chi_x)$,

   (b) A loop occurs in $(Y, \lhd_Y)$ and there is a formula φ such that

$$\sigma_Y(q_x) = \eta_C q_x.\varphi \quad \textit{and} \quad \sigma_Y(\chi_x) = \mathrm{unf}(\eta_C q_x.\varphi).$$

2. $\mathsf{Lit}(\sigma_Y(q_x)) \subseteq (\mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)) \cup \Lambda_{\lhd[C] \setminus Y}$.

3. $\sigma_Y(q_x) \in \mathcal{N}^{\eta_C}_{\Lambda_{C \setminus Y}}$.

4. *For all $z \in X$, $q_z \in FV(\sigma_Y(q_x))$ iff $z \notin Y$ and there exists $u \in Y$ such that $x \lhd^*_Y u$ and $u \lhd z$.*

5. *For all* $z \in X$*, if* $x \lhd_Y^* z$ *then* $\mathrm{Clos}(\sigma_Y(q_z)) \subseteq \mathrm{Clos}(\sigma_Y(q_x))$.

6. $|\sigma_Y(q_x)|_{\mathrm{Cl}} \leq 3|Y|$.

Proof. We proceed by induction on the size of Y. The case where $Y = \emptyset$ is vacuously true. Suppose $Y \neq \emptyset$. We now inspect the structure of $(Y, \lhd_Y)$ where $\lhd_Y$ is the restriction of $\lhd$ to Y. We distinguish cases depending on whether there is a node $y \in Y$ such that $y \lhd_Y^+ y$, i.e. whether there is a loop within Y.

- Suppose that no loops occur. Since the graph is finite there must be a leaf y with respect to $\lhd_Y$. Let $Z = Y \setminus \{y\}$, then by the induction hypothesis we know that there must be $\sigma_Z : \Lambda_Z \to \mathcal{L}_\Box^{\mathrm{af}}$ with the properties listed above. Let $\sigma_Y : \Lambda_Y \to \mathcal{L}_\Box^{\mathrm{af}}$ be the map given by

$$\sigma_Y(q_x) = \sigma_Z(q_x)[\chi_y/q_y]$$

for all $x \in Y$. Take an arbitrary $x \in Y$, we will prove this map satisfies all of the desired conditions.

1. We want to show that either 1(a) or 1(b) holds. Observe that these conditions are unique. We split cases depending on whether $x = y$.

   (a) If $x = y$ then observe the following equalities

$$\begin{aligned} \sigma_Y(q_y) &= \sigma_Z(q_y)[\chi_y/q_y] && \text{by definition of } \sigma_Y \\ &= q_y[\chi_y/q_y] && \text{since } y \notin Z \\ &= \chi_y && \text{Definition 2.16} \end{aligned}$$

   Moreover

$$\begin{aligned} \sigma_Y(\chi_y) &= \sigma_Z(\chi_y)[\chi_y/q_y] && \text{by definition of } \sigma_Y \\ &= \chi_y[\chi_y/q_y] && \text{since } FV(\chi_y) \cap \Lambda_Y = \emptyset \\ &= \chi_y && \text{since } q_y \notin FV(\chi_y) \end{aligned}$$

   So we have our desired result.

   (b) If $x \neq y$ then by the induction hypothesis, we know that $\sigma_Z(q_x)$ is $\sigma_Z(\chi_x)$ as there are no loops within $(Y, \lhd_Y)$ so there are no loops within $(Z, \lhd_Z)$. Observe the following equalities

$$\begin{aligned} \sigma_Y(q_x) &= \sigma_Z(q_x)[\chi_y/q_y] && \text{by definition of } \sigma_Y \\ &= \sigma_Z(\chi_x)[\chi_y/q_y] && \sigma_Z(q_x) = \sigma_Z(\chi_x) \\ &= \sigma_Y(\chi_x) && \text{by definition of } \sigma_Y \end{aligned}$$

2. We want to show that

$$\mathsf{Lit}(\sigma_Y(q_x)) \subseteq (\mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)) \cup \Lambda_{\lhd[C]\setminus Y}.$$

   We begin by splitting cases depending on whether $x = y$.

(a) If $x = y$ then $\sigma_Y(q_x) = \chi_y$. Take $q \in \mathsf{Lit}(\chi_x)$, by Lemma 5.35 we know that (i) $q \in \mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)$ or (ii) $q = q_z$ for some z such that $y \lhd z$. In (i) we immediately have our result. In (ii), we can conclude that $z \in \lhd[C]$ since $y \in C$ and $y \lhd z$, and that $z \notin Y$ since y was chosen to be a leaf in $(Y, \lhd_Y)$.

(b) If $x \neq y$ then $\sigma_Y(q_x) = \sigma_Z(q_x)[\chi_y/q_y]$. By the induction hypothesis we know that

$$\mathsf{Lit}(\sigma_Z(q_x)) \subseteq (\mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)) \cup \Lambda_{\lhd[C]\setminus Z}, \tag{A.2}$$

since $x \in Z$. Take $q \in \mathsf{Lit}(\sigma_Z(q_x)[\chi_y/q_y])$. Then (i) $q \in \mathsf{Lit}(\sigma_Z(q_x))$ and $q \neq q_y$ or (ii) $q_y \in FV(\sigma_Z(q_x))$ and $q \in \mathsf{Lit}(\chi_y)$. In (i), we get our desired result via (A.2) and basic set theory manipulations. In (ii) since $q \in \mathsf{Lit}(\chi_y)$ by Lemma 5.35 we get (1) $q \in \mathsf{Lit}(\overline{f(y)^l}) \cap \mathsf{Lit}(f(y)^r)$ or (2) $q = q_z$ for some z such that $y \lhd z$. In (1) we get our desired result since $q \in \mathsf{Lit}(\overline{f(y)^l}) \cap \mathsf{Lit}(f(y)^r)$ and $x \lhd^+ y$ since $x, y \in C$ so by Lemma 5.27 we have $q \in \mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)$. In (2) we can conclude that $z \in \lhd[C] \setminus Y$ since $y \in C$ and $y \lhd z$, and that $z \notin Y$ since y was chosen to be a leaf in $(Y, \lhd_Y)$.

3. We want to show that $\sigma_Y(q_x) \in \mathcal{N}^{\eta_C}_{\Lambda_{C\setminus Y}}$. We split cases depending on whether $x = y$.

(a) If $x = y$ then $\sigma_Y(q_y) = \chi_y$. Since χ_y is a basic formula we know by Lemma 5.34 that $\chi_y \in \mathcal{N}^{\eta_C}_{C\setminus Y}$.

(b) If $x \neq y$ then $\sigma_Y(q_x) = \sigma_Z(q_x)[\chi_y/q_y]$. By the induction hypothesis we know $\sigma_Z(q_x) \in \mathcal{N}^{\eta_C}_{C\setminus Z}$ and as mentioned in (a) we know that $\chi_y \in \mathcal{N}^{\eta_C}_{C\setminus Y}$. Thus since $C\setminus Z = (C\setminus Y)\cup\{y\}$ we know $\sigma_Z(q_x)[\chi_y/q_y] \in \mathcal{N}^{\eta_C}_{C\setminus Y}$ by Lemma 2.27.

4. Take $z \in X$, we want to show that $q_z \in FV(\sigma_Y(q_x))$ if and only if there is u such that $x \lhd^*_Y u$ and $u \lhd z$. We split cases depending on whether $x = y$.

(a) If $x = y$ then $\sigma_Y(q_x) = \chi_y$. Suppose $q_z \in FV(\chi_y)$, then by Lemma 5.35 we know $y \lhd z$ and $z \notin Y$ since y is a leaf in $(Y, \lhd_Y)$. Thus we have $y \lhd^*_Y y$ and $y \lhd z$. Now suppose $z \notin Y$ and there is u such that $y \lhd^*_Y u$ and $u \lhd z$. Then since y is a leaf in Y it follows that $u = y$ so $y \lhd z$. Then by Lemma 5.36, $q_z \in FV(\chi_y)$, so $q_z \in FV(\sigma_Y(q_y))$, our desired result.

(b) If $x \neq y$ then $\sigma_Y(q_x) = \sigma_Z(q_x)[\chi_y/q_y]$. Suppose $q_z \in FV(\sigma_Z(q_x)[\chi_y/q_y])$, then (i) $q_z \in FV(\sigma_Z(q_x))$ and $z \neq y$ or (ii) $q_z \in FV(\chi_y)$ and $q_y \in FV(\sigma_Z(q_x))$. In (i) we know by the induction hypothesis that $z \notin Z$ and there is u such that $x \lhd^*_Z u$ and $u \lhd z$. Therefore $x \lhd^*_Y u$ and $u \lhd z$ and $z \notin Y$ since $z \notin Z$ and $z \neq y$. In (ii) we know by the induction hypothesis that there is u such that $x \lhd^*_Z u$ and $u \lhd y$ and by Lemma 5.36 that $y \lhd z$. It follows that $z \notin Y$ since y is a leaf in Y and $y \lhd z$. Moreover, $x \lhd^*_Y y$ and $y \lhd z$, so we have our desired result.

Now suppose in the opposite direction that $z \notin Y$ and there is u such that $x \lhd^*_Y u$ and $u \lhd z$. We check cases depending on whether $x \lhd^*_Z u$. If $x \lhd^*_Z u$ then since $z \notin Y$ we know $z \notin Z$ so by the induction hypothesis we have $q_z \in FV(\sigma_Z(q_x))$. Therefore $q_z \in FV(\sigma_Z(q_x)[\chi_y/q_y])$ since $z \neq y$ as $z \notin Y$. If it does not hold that $x \lhd^*_Z u$ then it must be that there exists v such that $x \lhd^*_Z v$ and $v \lhd_Y y$ and $y \lhd z$ since y is a leaf in Y. So by induction hypothesis we know $q_y \in FV(\sigma_Z(q_x))$ and since $y \lhd z$ we know by Lemma 5.36 that $q_z \in FV(\chi_y)$, so $q_z \in FV(\sigma_Z(q_x)[\chi_y/q_y])$, our desired result.

5. Suppose $x \lhd_Y^* z$, we want to show that $\text{Clos}(\sigma_Y(q_z)) \subseteq \text{Clos}(\sigma_Y(q_x))$. We split cases depending on whether $x = y$.
   (a) Suppose $x = y$, then since y is a leaf in $(Y, \lhd_Y)$ we know that $z = y$, so our result follows immediately.
   (b) Suppose $x \neq y$, since y is a leaf in $(Y, \lhd_Y)$ we can conclude that either (i) $z = y$ or (ii) $x \lhd_Z^* z$. In (i) we can conclude that since $z = y$ and y is a leaf in $(Y, \lhd_Y)$ that there exists u such that $x \lhd_Z^* u$ and $u \lhd y$, so by our induction hypothesis, we know that $q_y \in FV(\sigma_Y(q_x))$. Thus we have

$$\begin{aligned}
\text{Clos}(\sigma_Y(q_x)) &= \text{Clos}(\sigma_Z(q_x)[\chi_y/q_y]) && \text{by definition of } \sigma_Y \\
&= \text{Clos}(\sigma_Z(q_x)) \cup \text{Clos}(\chi_y) && \text{Lemma 2.21} \\
&\supseteq \text{Clos}(\chi_y) \\
&= \text{Clos}(\sigma_Y(q_y)) && \text{by definition of } \sigma_Y
\end{aligned}$$

   In (ii), we know by our induction hypothesis that $\text{Clos}(\sigma_Z(q_z)) \subseteq \text{Clos}(\sigma_Z(q_x))$. Using Lemma 2.22, we know that $\text{Clos}(\sigma_Z(q_z)[\chi_y/q_y]) \subseteq \text{Clos}(\sigma_Z(q_x)[\chi_y/q_y])$, our desired result.
6. We want to show that $|\sigma_Y(q_x)|_{\text{Cl}} \leq 3|Y|$. We split cases depending on whether $x = y$.
   (a) If $x = y$ then $\sigma_Y(q_y) = \chi_y$, and by inspection on the definition of χ_y given in Definition 5.33 we know $|\chi_y|_{\text{Cl}} \leq 3$ so $|\chi_y|_{\text{Cl}} \leq 3|Y|$ for $|Y| > 0$.
   (b) If $x \neq y$ then $\sigma_Y(q_x) = \sigma_Z(q_x)[\chi_y/q_y]$, thus

$$\begin{aligned}
|\sigma_Y(q_x)|_{\text{Cl}} &= |\sigma_Z(q_x)[\chi_y/q_y]|_{\text{Cl}} \\
&\leq |\sigma_Z(q_x)|_{\text{Cl}} + |\chi_y|_{\text{Cl}} && \text{Lemma 2.24} \\
&\leq 3|Z| + 3 && \text{(IH), } |\chi_y|_{\text{Cl}} \leq 3 \\
&= 3|Y| && \text{since } |Y| = |Z| + 1
\end{aligned}$$

- Suppose that a loop occurs. There must be a box node on this cycle; let it be y. Then $\chi_y = \triangle q_{y'}$ where y' is the unique premise of y and $\triangle \in \{\Box, \Diamond\}$. We note for later that y' will be in the same cluster as y, and must also be in Y since it is the unique premise of y and $y \lhd^+ y$. Let $Z = Y \setminus \{y\}$, by the induction hypothesis we know that there must be $\sigma_Z : \Lambda_Z \to \mathcal{L}_\Box^{\text{af}}$ that satisfies all properties listed above. Let $\sigma_Y : \Lambda_Y \to \mathcal{L}_\Box^{\text{af}}$ be the map given by

$$\sigma_Y(q_x) = \sigma_Z(q_x)[\eta_C q_y.\sigma_Z(\chi_y)/q_y].$$

  for all $x \in Y$. Take an arbitrary $x \in Y$, we will prove this map satisfies all of the listed conditions.

  1. We want to show that either 1(a) or 1(b) holds. We split cases depending on whether $x = y$.
     (a) If $x = y$ then observe the following equalities

$$\begin{aligned}
\sigma_Y(q_y) &= \sigma_Z(q_y)[\eta_C q_y.\sigma_Z(\chi_y)/q_y] && \text{by definition of } \sigma_Y \\
&= q_y[\eta_C q_y.\sigma_Z(\chi_y)/q_y] && \text{since } y \notin Z \\
&= \eta_C q_y.\sigma_Z(\chi_y) && \text{Definition 2.16}
\end{aligned}$$

Therefore $\sigma_Y(q_y) = \eta_C q_y.\varphi$ where $\varphi = \sigma_Z(\chi_y)$. It remains to show that $\sigma_Y(\chi_y) = \mathrm{unf}(\eta_C q_y.\sigma_Z(\chi_y))$. To see this, observe that

$$\begin{aligned}\mathrm{unf}(\eta q_y.\sigma_Z(\chi_y)) &= \sigma_Z(\chi_y)[\eta q_y.\sigma_Z(\chi_y)/q_y] && \text{Definition 2.16}\\ &= \sigma_Y(\chi_y) && \text{by definition of } \sigma_Y\end{aligned}$$

So we have our desired result.

(b) If $x \neq y$ then $\sigma_Y(q_x) = \sigma_Z(q_x)[\eta_C q_y.\sigma_Z(\chi_y)/q_y]$. By the induction hypothesis, we know that (1) $\sigma_Z(q_x) = \sigma_Z(\chi_x)$ or (2) there is a formula φ such that $\sigma_Z(q_x) = \eta_C q_x.\varphi$ and $\sigma_Z(\chi_x) = \mathrm{unf}(\eta_C q_x.\varphi)$. In (1) we get our desired result since

$$\begin{aligned}\sigma_Y(q_x) &= \sigma_Z(q_x)[\eta_C q_y.\sigma_Z(\chi_y)/q_y] && \text{by definition of } \sigma_Y\\ &= \sigma_Z(\chi_x)[\eta_C q_y.\sigma_Z(\chi_y)/q_y] && \text{since } \sigma_Z(q_x) = \sigma_Z(\chi_x)\\ &= \sigma_Y(\chi_x) && \text{by definition of } \sigma_Y\end{aligned}$$

Case (2) requires more work. Putting everything we know so far together, we get

$$\begin{aligned}\sigma_Y(q_x) &= \sigma_Z(q_x)[\eta_C q_y.\sigma_Z(\chi_y)/q_y] && \text{by definition of } \sigma_Y\\ &= (\eta_C q_x.\varphi)[\eta_C q_y.\sigma_Z(\chi_y)/q_y] && \text{since } \sigma_Z(q_x) = \eta_C q_x.\varphi\\ &= \eta_C q_x.\varphi[\eta_C q_y.\sigma_Z(\chi_y)/q_y] && \text{Definition 2.16}\end{aligned}$$

therefore $\sigma_Y(q_x)$ has the form $\eta_C q_x.\psi$ where ψ is $\varphi[\eta_C q_y.\sigma_Z(\chi_y)/q_y]$. It remains to show that $\mathrm{unf}(\eta_C q_x.\varphi[\eta_C q_y.\sigma_Z(\chi_y)/q_y]) = \sigma_Y(\chi_x)$. Taking the unfolding, we get

$$\begin{aligned}&\mathrm{unf}(\eta_C q_x.\varphi[\eta_C q_y.\sigma_Z(\chi_y)/q_y])\\ &= \varphi[\eta_C q_y.\sigma_Z(\chi_y)/q_y][(\eta_C q_x.\varphi[\eta_C q_y.\sigma_Z(\chi_y)/q_y])/q_x] && \text{Definition 2.17}\\ &= \varphi[\eta_C q_x.\varphi/q_x][\eta_C q_y.\sigma_Z(\chi_y)/q_y] && \text{Lemma 2.18}\\ &= \mathrm{unf}(\eta_C q_x.\varphi)[\eta_C q_y.\sigma_Z(\chi_y)/q_y] && \text{Definition 2.17}\\ &= \sigma_Z(\chi_x)[\eta_C q_y.\sigma_Z(\chi_y)/q_y] && \text{since } \sigma_Z(\chi_x) = \mathrm{unf}(\eta_C q_x.\varphi)\\ &= \sigma_Y(\chi_x) && \text{by definition of } \sigma_Y\end{aligned}$$

Thus we have our desired result.

2. We want to show that

$$\mathsf{Lit}(\sigma_Y(q_x)) \subseteq (\mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)) \cup \Lambda_{\lhd[C]\setminus Y}.$$

We begin by splitting cases depending on whether $x = y$.

(a) If $x = y$ then $\sigma_Y(q_x) = \eta_C q_y.\sigma_Z(\chi_y)$. Take $q \in \mathsf{Lit}(\eta_C q_y.\sigma_Z(\chi_y))$. As previously mentioned this means $q \in \mathsf{Lit}(\sigma_Z(q_{y'}))$ and $q \neq q_y$. By the induction hypothesis we have

$$\mathsf{Lit}(\sigma_Z(q_{y'})) \subseteq (\mathsf{Lit}(\overline{f(y')^l}) \cap \mathsf{Lit}(f(y')^r)) \cup \Lambda_{\lhd[C]\setminus Z}.$$

so $q \in (\mathsf{Lit}(\overline{f(y')^l}) \cap \mathsf{Lit}(f(y')^r)) \cup \Lambda_{\lhd[C]\setminus Z}$. We get two cases: (1) $q \in \mathsf{Lit}(\overline{f(y')^l}) \cap$

$\mathsf{Lit}(f(y')^r)$ or (2) $q \in \Lambda_{\lhd[C]\setminus Z}$. In (1) $q \in \mathsf{Lit}(\overline{f(y')^l}) \cap \mathsf{Lit}(f(y')^r)$ and $y \lhd y'$ so $q \in \mathsf{Lit}(\overline{f(y)^l}) \cap \mathsf{Lit}(f(y)^r)$ by Lemma 5.27. In (2) since $q \in \Lambda_{\lhd[C]\setminus Z}$ and $q \neq q_y$ we get our desired result.

(b) If $x \neq y$ then $\sigma_Y(q_x) = \sigma_Z(q_x)[\eta_C q_y.\sigma_Z(\chi_y)/q_y]$. By the induction hypothesis we know that

$$\mathsf{Lit}(\sigma_Z(q_x)) \subseteq (\mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)) \cup \Lambda_{\lhd[C]\setminus Z}, \tag{A.3}$$

since $x \in Z$. Suppose $q \in \mathsf{Lit}(\sigma_Z(q_x)[\eta_C q_y.\sigma_Z(\chi_y)/q_y])$, then either (1) $q \in \mathsf{Lit}(\sigma_Z(q_x))$ and $q \neq q_y$ or (2) $q_y \in FV(\sigma_Z(q_x))$ and $q \in \mathsf{Lit}(\eta_C q_y.\sigma_Z(\chi_y))$. In (1) by (A.3) we get $q \in (\mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)) \cup \Lambda_{\lhd[C]\setminus Z}$ and since $q \neq q_y$ we get our desired result from basic set theory manipulations. In (2) since $q_y \in FV(\sigma_Z(q_x))$ we get $x \lhd^+ y$ and since $q \in \mathsf{Lit}(\eta_C q_y.\sigma_Z(\chi_y))$ we get $q \in \mathsf{Lit}(\sigma_Z(q_{y'}))$ and $q \neq q_y$. By the induction hypothesis we know

$$\mathsf{Lit}(\sigma_Z(q_{y'})) \subseteq (\mathsf{Lit}(\overline{f(y')^l}) \cap \mathsf{Lit}(f(y')^r)) \cup \Lambda_{\lhd[C]\setminus Z}$$

so $q \in (\mathsf{Lit}(\overline{f(y')^l}) \cap \mathsf{Lit}(f(y')^r)) \cup \Lambda_{\lhd[C]\setminus Z}$. Therefore (i) $q \in \mathsf{Lit}(\overline{f(y')^l}) \cap \mathsf{Lit}(f(y')^r)$ or (ii) $q \in \Lambda_{\lhd[C]\setminus Z}$. In (i) we get our desired result since $x \lhd^+ y$ and $y \lhd y'$ so $q \in \mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)$ by Lemma 5.27. In (ii) we get our desired result since $q \in \Lambda_{\lhd[C]\setminus Z}$ and $q \neq q_y$.

3. We want to show that $\sigma_Y(q_x) \in \mathcal{N}^{\eta_C}_{\Lambda_{C\setminus Y}}$. We split cases depending on whether $x = y$.

(a) If $x = y$ then $\sigma_Y(q_x) = \eta_C q_y.\sigma_Z(\chi_y)$. Recall that $\sigma_Z(\chi_y) = \triangle\sigma_Z(q_{y'})$ where $y' \in Z$ is the unique premise of y. By the induction hypothesis we know $\sigma_Z(q_{y'}) \in \mathcal{N}^{\eta_C}_{\Lambda_{C\setminus Z}}$ and moreover we know $C \setminus Z = (C \setminus Y) \cup \{y\}$. Thus $\eta_C q_y.\sigma_Z(\chi_y) \in \mathcal{N}^{\eta_C}_{\Lambda_{C\setminus Y}}$ by Definition 2.25, our desired result.

(b) If $x \neq y$ then $\sigma_Y(q_x) = \sigma_Z(q_x)[\eta_C q_y.\sigma_Z(\chi_y)/q_y]$. By the induction hypothesis we know that $\sigma_Z(q_x) \in \mathcal{N}^{\eta_C}_{\Lambda_{C\setminus Z}}$ and as mentioned previously we know $\eta_C q_y.\sigma_Z(\chi_y) \in \mathcal{N}^{\eta_C}_{\Lambda_{C\setminus Y}}$. Thus by Lemma 2.26 we know $\sigma_Z(q_x)[\eta_C q_y.\sigma_Z(\chi_y)/q_y] \in \mathcal{N}^{\eta_C}_{\Lambda_{C\setminus Y}}$ since $C \setminus Z = (C \setminus Y) \cup \{y\}$.

4. Take $z \in X$, we want to show that $q_z \in FV(\sigma_Y(q_x))$ if and only if there is u such that $x \lhd_Y^* u$ and $u \lhd z$. We proceed by splitting cases depending on whether $x = y$.

(a) If $x = y$ then $\sigma_Y(q_x) = \eta_C q_y.\sigma_Z(\chi_y)$. Suppose $q_z \in FV(\eta_C q_y.\sigma_Z(\chi_y))$, then recall $\chi_y = \triangle q_{y'}$ where y' is the unique premise of y and $\triangle \in \{\Box, \Diamond\}$. Thus $q_z \in FV(\sigma_Z(q_{y'}))$ and $z \neq y$ by definition of FV. Thus by the induction hypothesis we know there exists u such that $y' \lhd_Z^* u$ and $u \lhd z$ and $z \notin Z$. Thus we know since $y \lhd_Y y'$ that $y \lhd_Y^* u$ and $u \lhd z$ as well as $z \notin Y$ since $z \notin Z$ and $z \neq y$, giving us our desired result.

In the opposite direction suppose $z \notin Y$ and there is u such that $y \lhd_Y^* u$ and $u \lhd z$. Then since $y \lhd_Y^* u$ and y' is the unique premise of y, we can conclude either $y = u$ or $y' \lhd_Z^* u$. If $y = u$ then $y \lhd z$ so $z = y'$ since y' is the unique premise of y and $q_{y'} \in FV(\sigma_Y(q_y))$ follows by the induction hypothesis since $y \notin Z$ and $y' \lhd_Z^* y'$ and $y' \lhd y$. Otherwise, if $y' \lhd_Z^* u$ then by the induction hypothesis we know $q_z \in FV(\sigma_Z(q_{y'}))$ and $z \neq y$ since $z \notin Y$. Thus $q_z \in FV(\eta_C q_y.\triangle\sigma_Z(q_{y'}))$, our desired result.

(b) If $x \neq y$ then $\sigma_Y(q_x) = \sigma_Z(q_x)[\eta_C q_y.\sigma_Z(\chi_y)/q_y]$. Let $q_z \in FV(\sigma_Z(q_x)[\eta_C q_y.\sigma_Z(\chi_y)/q_y])$, then (i) $q_z \in FV(\sigma_Z(q_x))$ and $z \neq y$ or (ii) $q_z \in FV(\eta_C q_y.\sigma_Z(\chi_y))$ and $q_y \in FV(\sigma_Z(q_x))$. In (i) we know by the induction hypothesis that there is u such that $x \lhd_Z^* u$ and $u \lhd z$ as well as $z \notin Z$. Since $z \neq y$ we can conclude $z \notin Y$ and $x \lhd_Y^* u$ and $u \lhd z$, our desired

result. In (ii) we know from $q_z \in FV(\eta_C q_y.\sigma_Z(\chi_y))$ that there is u such that $y \lhd_Y^* u$ and $u \lhd z$ and $z \notin Y$ by the same reasoning used in (a). Moreover, by the induction hypothesis and $q_y \in FV(\sigma_Z(q_x))$ we know that there is v such that $x \lhd_Z^* v$ and $v \lhd y$. Thus, combining everything we have $x \lhd_Y^* u$ and $u \lhd z$, our desired result.

In the opposite direction suppose that $z \notin Y$ and there is u such that $x \lhd_Y^* u$ and $u \lhd z$. Two cases follow, depending on whether $x \lhd_Z^* u$. If $x \lhd_Z^* u$ then by the induction hypothesis we know since $z \notin Z$ that $q_z \in FV(\sigma_Z(q_x))$. Since $z \notin Y$ we know $z \neq x$ so we can conclude that $q_z \in FV(\sigma_Z(q_x)[\eta_C q_y.\sigma_Z(\chi_y)/q_y])$. If it does not hold that $x \lhd_Z^* u$ then it follows from basic reasoning about graphs that either $u = y$ or there exists v such that $x \lhd_Z^* v$ and $v \lhd y$ and $y' \lhd_Z^* u$. Thus by the induction hypothesis we get $q_y \in FV(\sigma_Z(q_x))$ and $q_z \in FV(\sigma_Z(q_{y'}))$, which we can put together to get $q_z \in FV(\sigma_Z(q_x)[\eta_C q_y.\sigma_Z(\chi_y)/q_y])$, our desired result.

5. Take $z \in Y$ and suppose that $x \lhd_Y^* z$. We want to show that $\mathrm{Clos}(\sigma_Y(q_z)) \subseteq \mathrm{Clos}(\sigma_Y(q_x))$. We split cases depending on whether $x = y$.

   (a) If $x = y$ then $\sigma_Y(q_x) = \eta_C q_y.\sigma_Z(\chi_y)$. Since $y \lhd_Y^* z$ we know by some simple graph theory that either $y = z$ or $y' \lhd_Z^* z$ where we recall that y' is the unique premise of y. In the first case, we instantly get our result since $x = z$. So we suppose $y' \lhd_Z^* z$. Then by the induction hypothesis we know that $\mathrm{Clos}(\sigma_Z(q_z)) \subseteq \mathrm{Clos}(\sigma_Z(q'_y))$. From this, since $\chi_y = \triangle q_{y'}$ we can deduce that

   $$\mathrm{Clos}(\sigma_Z(q_z)) \subseteq \mathrm{Clos}(\sigma_Z(q'_y)) \subseteq \mathrm{Clos}(\triangle\sigma_Z(q'_y)) = \mathrm{Clos}(\sigma_Z(\chi_y)) \tag{A.4}$$

   Now, observe the following

   $$\begin{aligned}
   \mathrm{Clos}(\eta_C q_y.\sigma_Z(\chi_y)) &= \{\varphi[\eta_C q_y.\sigma_Z(\chi_y)/q_y] \mid \varphi \in \mathrm{Clos}(\sigma_Z(\chi_y))\} \\
   &\quad \cup \mathrm{Clos}(\eta_C q_y.\sigma_Z(\chi_y)) && \text{Lemma 2.21} \\
   &\supseteq \{\varphi[\eta_C q_y.\sigma_Z(\chi_y)/q_y] \mid \varphi \in \mathrm{Clos}(\sigma_Z(q_z))\} \\
   &\quad \cup \mathrm{Clos}(\eta_C q_y.\sigma_Z(\chi_y)) && \text{(A.4)} \\
   &\supseteq \mathrm{Clos}(\sigma_Y(q_z)[\eta_C q_y.\sigma_Z(\chi_y)/q_y]) && \text{Lemma 2.21} \\
   &= \mathrm{Clos}(\sigma_Y(q_z)) && \text{by definition of } \sigma_Y
   \end{aligned}$$

   (b) If $x \neq y$ then we know either (i) $x \lhd_Z^* z$ or (ii) $x \lhd_Y^+ y$ and $y \lhd_Y^* z$. In (i) we know by our induction hypothesis that $\mathrm{Clos}(\sigma_Z(q_z)) \subseteq \mathrm{Clos}(\sigma_Z(q_x))$. Using Lemma 2.22, we know that $\mathrm{Clos}(\sigma_Z(q_z)[\chi_y/q_y]) \subseteq \mathrm{Clos}(\sigma_Z(q_x)[\chi_y/q_y])$, our desired result. In (ii) we know from that work of (a) that if $y \lhd_Y^* z$ then $\mathrm{Clos}(\sigma_Y(q_z)) \subseteq \mathrm{Clos}(\sigma_Y(q_y))$. Thus it remains to show that $\mathrm{Clos}(\sigma_Y(q_y)) \subseteq \mathrm{Clos}(\sigma_Y(q_x))$. From $x \lhd^+ y$ we may assume by cutting out loops that there is u such that $x \lhd_Z^* u$ and $u \lhd y$. From this, it follows that

$q_y \in FV(\sigma_Z(q_x))$. Therefore we get our desired result as follows:

$$\begin{aligned}
\mathrm{Clos}(\sigma_Y(q_x)) &= \mathrm{Clos}(\sigma_Z(q_x)[\eta_C q_y.\sigma_Z(\chi_y)/q_y]) && \text{by definition of } \sigma_Y \\
&= \{\varphi[\eta_C q_y.\sigma_Z(\chi_y)/q_y] \mid \varphi \in \mathrm{Clos}(\sigma_Z(q_x))\} \\
&\quad \cup \mathrm{Clos}(\eta_C q_y.\sigma_Z(\chi_y)) && \text{Lemma 2.21} \\
&\supseteq \mathrm{Clos}(\eta_C q_y.\sigma_Z(\chi_y)) \\
&= \mathrm{Clos}(\sigma_Y(q_y)) && \text{by definition of } \sigma_Y
\end{aligned}$$

6. We want to show that $|\sigma_Y(q_x)|_{\mathrm{Cl}} \leq 3|Y|$. We split cases depending on whether $x = y$.

   (a) If $x = y$ then $\sigma_Y(q_x) = \eta_C q_y.\sigma_Z(\chi_y)$, and we know $\chi_y = \triangle q_{y'}$ where y' is the unique premise of y and $\triangle \in \{\Box, \Diamond\}$ as well as $y' \in Z$. Putting this all together we have

$$\begin{aligned}
|\sigma_Y(q_x)|_{\mathrm{Cl}} &= |\eta_C q_y.\triangle\sigma_Z(q_{y'})|_{\mathrm{Cl}} && \text{by definition of } \sigma_Y \\
&\leq 2 + |\sigma_Z(q_{y'})|_{\mathrm{Cl}} && \text{Definition 2.23} \\
&\leq 2 + 3|Z| && \text{(IH)} \\
&< 3|Y| && |Y| = |Z| + 1
\end{aligned}$$

   (b) If $x \neq y$ then $\sigma_Y(q_x) = \sigma_Z(q_x)[\eta_C q_y.\sigma_Z(\chi_y)/q_y]$. To begin with, if $q_y \notin FV(\sigma_Z(q_x))$ then $\sigma_Y(q_x) = \sigma_Z(q_x)$ so by the induction hypothesis we easily get

$$|\sigma_Y(q_x)|_{\mathrm{Cl}} = |\sigma_Z(q_x)|_{\mathrm{Cl}} \leq 3|Z| < 3|Y|.$$

   Otherwise, if $q_y \in FV(\sigma_Z(q_x))$ we will actually unfold the definition of closure, and use our results about closure. To begin with, note the following:

$$\begin{aligned}
\mathrm{Clos}(\sigma_Y(q_x)) &= \mathrm{Clos}(\sigma_Z(q_x)[\eta_C q_y.\sigma_Z(\chi_y)/q_y]) \\
&= \{\varphi[\eta_C q_y.\sigma_Z(\chi_y)/q_y] \mid \varphi \in \mathrm{Clos}(\sigma_Z(q_x))\} \cup \mathrm{Clos}(\eta_C q_y.\sigma_Z(\chi_y))
\end{aligned}$$

   This follows from Lemma 2.22. Now observe the following

$$\mathrm{Clos}(\eta_C q_y.\sigma_Z(\chi_y)) = \{\eta_C q_y.\sigma_Z(\chi_y)\} \cup \{\varphi[\eta_C q_y.\sigma_Z(\chi_y)/q_y] \mid \varphi \in \mathrm{Clos}(\sigma_Z(\chi_y))\} \quad \text{(A.5)}$$

   which follows from Lemma 2.21. Now, recall that $\chi_y = \triangle q_{y'}$ where y' is the unique premise of y. Moreover, since $q_y \in FV(\sigma_Z(q_x))$ we know that $x \lhd_Y^+ y'$ so by our induction hypothesis we know $\mathrm{Clos}(\sigma_Z(q_{y'})) \subseteq \mathrm{Clos}(\sigma_Z(q_x))$. Applying this to (A.5) we get

$$\begin{aligned}
&\mathrm{Clos}(\eta_C q_y.\sigma_Z(\chi_y)) \\
&= \{\eta_C q_y.\sigma_Z(\chi_y)\} \cup \{\varphi[\eta_C q_y.\sigma_Z(\chi_y)/q_y] \mid \varphi \in \mathrm{Clos}(\triangle\sigma_Z(q_{y'}))\} && \text{(A.5)} \\
&= \{\eta_C q_y.\sigma_Z(\chi_y), \sigma_Z(\chi_y)[\eta_C q_y.\sigma_Z(\chi_y)/q_y]\} \\
&\qquad \cup \{\varphi[\eta_C q_y.\sigma_Z(\chi_y)/q_y] \mid \varphi \in \mathrm{Clos}(\sigma_Z(q_{y'}))\} && \text{Lemma 2.21} \\
&\subseteq \{\eta_C q_y.\sigma_Z(\chi_y), \sigma_Z(\chi_y)[\eta_C q_y.\sigma_Z(\chi_y)/q_y]\} \\
&\qquad \cup \{\varphi[\eta_C q_y.\sigma_Z(\chi_y)/q_y] \mid \varphi \in \mathrm{Clos}(\sigma_Z(q_x))\} && \text{(IH)}
\end{aligned}$$

Combining everything we get

$$\begin{aligned}\mathrm{Clos}(\sigma_Y(q_x)) &= \{\varphi[\eta_C q_y.\sigma_Z(\chi_y)/q_y] \mid \varphi \in \mathrm{Clos}(\sigma_Z(q_x))\} \cup \mathrm{Clos}(\eta_C q_y.\sigma_Z(\chi_y)) \\ &\subseteq \{\varphi[\eta_C q_y.\sigma_Z(\chi_y)/q_y] \mid \varphi \in \mathrm{Clos}(\sigma_Z(q_x))\} \\ &\quad \cup \{\eta_C q_y.\sigma_Z(\chi_y), \sigma_Z(\chi_y)[\eta_C q_y.\sigma_Z(\chi_y)/q_y]\}\end{aligned}$$

Therefore $|\sigma_Y(q_x)|_{\mathrm{Cl}} \leq |\sigma_Z(q_x)|_{\mathrm{Cl}} + 2$, so by our induction hypothesis we get

$$|\sigma_Y(q_x)|_{\mathrm{Cl}} \leq |\sigma_Z(q_x)|_{\mathrm{Cl}} + 2 \leq 3|Z| < 3|Y|. \qquad \square$$

Theorem A.3. *Given a finite* $\mathsf{A}^{\mathrm{split}}$*-proof* (X, α) *and* $x \in X$*, there is* $\sigma : \Lambda_X \to \mathcal{L}^{\mathrm{af}}_{\Box}$ *such that for all* $x \in X$*, the following hold.*

1. *Exactly one of the following hold:*
   (a) $\sigma(q_x) = \sigma(\chi_x)$,
   (b) *There is a formula* φ *such that* $\sigma(q_x) = \eta_x q_x.\varphi$ *and* $\sigma(\chi_x) = \mathrm{unf}(\eta_x q_x.\varphi)$.
2. $\mathsf{Lit}(\sigma(q_x)) \subseteq \mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r)$.
3. $\sigma(q_x) \in \mathcal{L}^{\mathrm{af}}_{\Box}$.
4. $|\sigma(q_x)|_{\mathrm{Cl}} \leq 3|X|$.

Proof. Take $x \in X$, we want to define $\sigma(q_x)$. We know $x \in C$ for some cluster C. We proceed by well-founded induction on $(\mathrm{Clus}(X), \lhd)$.

- If C is a leaf in $(\mathrm{Clus}(X), \prec)$, we define

  $$\sigma(q_x) := \sigma_C(q_x)$$

  where $\sigma_C(q_x)$ is as defined in Lemma 5.37. We prove this map meets the above properties. For this case, these properties follow directly from Lemma 5.37.

- If C is not a leaf then inductively we can assume that $\sigma(q_y)$ is defined and meets the above properties for all y in a higher cluster than x. We define

  $$\sigma(q_x) := \sigma_C(q_x)[\sigma(q_y)/q_y \mid y \in \lhd[C] \setminus C]$$

  We prove this map satisfies the above conditions. To do this we will need the following observations which we will refer to.

  $$q_y \in FV(\sigma_C(q_x)) \text{ if and only if } y \in \lhd[C] \setminus C \tag{A.6}$$

  This follows since by Lemma 5.37 we know that $q_y \in FV(\sigma_C(q_x))$ if and only if there exists u such that $x \lhd_C^+ u$ and $u \lhd y$ and $y \notin C$, and it can be checked that the latter condition is equivalent to the condition $y \in \lhd[C] \setminus C$. Another important property is

  $$\sigma(\chi_x) = \sigma_C(\chi_x)[\sigma(q_y)/q_y \mid y \in \lhd[C] \setminus C] \tag{A.7}$$

This follows from induction on χ_x, recalling that it is a modal formula which only contains q_y such that y is in C or a higher cluster. We now show that x satisfies the necessary properties.

1. We want to show that 1(a) or 1(b) holds. We know from Lemma 5.37 that (a) $\sigma_C(q_x) = \sigma_C(\chi_x)$ or (b) $\sigma_C(q_x) = \eta_C q_x.\varphi$ and $\mathrm{unf}(\eta_C q_x.\varphi) = \sigma_C(\chi_x)$.

   (a) If $\sigma_C(q_x) = \sigma_C(\chi_x)$ then observe the following equalities.

$$\begin{aligned}
\sigma(q_x) &= \sigma_C(q_x)[\sigma(q_y)/q_y \mid y \in \lhd[C] \setminus C] && \text{by definition of } \sigma \\
&= \sigma_C(\chi_x)[\sigma(q_y)/q_y \mid y \in \lhd[C] \setminus C] && \text{since } \sigma_C(q_x) = \sigma_C(\chi_x) \\
&= \sigma(\chi_x) && \text{by (A.7)}
\end{aligned}$$

   Thus we have our desired result.

   (b) Now suppose $\sigma_C(q_x) = \eta_C q_x.\varphi$ and $\mathrm{unf}(\eta_C q_x.\varphi) = \sigma_C(\chi_x)$. Then since C is the cluster x resides in, we know $x \in C$ and $\eta_C = \eta_x$. Observe the following equalities.

$$\begin{aligned}
\sigma(q_x) &= \sigma_C(q_x)[\sigma(q_y)/q_y \mid y \in \lhd[C] \setminus C] && \text{by definition of } \sigma \\
&= (\eta_x q_x.\varphi)[\sigma(q_y)/q_y \mid y \in \lhd[C] \setminus C] && \text{since } \sigma_C(q_x) = \eta_C q_x.\varphi \\
&= \eta_x q_x.\varphi[\sigma(q_y)/q_y \mid y \in \lhd[C] \setminus C] && \text{Definition 2.16, } x \in C
\end{aligned}$$

   Thus $\sigma(q_x)$ is an η_x-formula. It remains to show that the unfolding is equal to $\sigma(\chi_x)$.

$$\begin{aligned}
&\mathrm{unf}(\eta_x q_x.\varphi[\sigma(q_y)/q_y \mid y \in \lhd[C] \setminus C]) \\
&= \varphi[\sigma(q_y)/q_y \mid y \in \lhd[C] \setminus C][(\eta_x q_x.\varphi[\sigma(q_y)/q_y \mid y \in \lhd[C] \setminus C])/q_x] && \text{Definition 2.16} \\
&= \varphi[\eta_x q_x.\varphi/q_x][\sigma(q_y)/q_y \mid y \in \lhd[C] \setminus C] && \text{Lemma 2.18} \\
&= \mathrm{unf}(\eta_x q_x.\varphi)[\sigma(q_y)/q_y \mid y \in \lhd[C] \setminus C] && \text{Definition 2.17} \\
&= \sigma_C(\chi_x)[\sigma(q_y)/q_y \mid y \in \lhd[C] \setminus C] && \text{since } \mathrm{unf}(\eta_C q_x.\varphi) = \sigma_C(\chi_x) \\
&= \sigma(\chi_x) && \text{by (A.7)}
\end{aligned}$$

   Thus we have our desired result.

2. We now want to show the vocabulary condition. By the induction hypothesis we know that for all $y \in \lhd[C] \setminus C$ we have $\mathsf{Lit}(\sigma(q_y)) \subseteq \mathsf{Lit}(\overline{f(y)^l}) \cap \mathsf{Lit}(f(y)^r)$ and since $y \in \lhd[C] \setminus C$ we know $x \lhd^+ y$ so by Lemma 5.27 we know

$$\mathsf{Lit}(\sigma(q_y)) \subseteq \mathsf{Lit}(\overline{f(x)^l}) \cap \mathsf{Lit}(f(x)^r) \tag{A.8}$$

Observe the following

$$
\begin{aligned}
&\mathsf{Lit}(\sigma(q_x)) \\
&= \mathsf{Lit}(\sigma_C(q_x)[\sigma(q_y)/q_y \mid y \in \lhd[C] \setminus C]) && \text{by definition of } \sigma \\
&\subseteq (\mathsf{Lit}(\sigma_C(q_x)) \setminus \Lambda_{\lhd[C]\setminus C}) \cup \bigcup_{y \in \lhd[C]\setminus C} \mathsf{Lit}(\sigma(q_y)) && \text{by (A.6), Lemma 2.19} \\
&\subseteq (((\mathsf{Lit}(f(x)^l) \cap \mathsf{Lit}(f(x)^r)) \cup \Lambda_{\lhd[C]\setminus C}) \setminus \Lambda_{\lhd[C]\setminus C}) \cup \bigcup_{y \in \lhd[C]\setminus C} \mathsf{Lit}(\sigma(q_y)) && \text{Lemma 5.37} \\
&\subseteq \mathsf{Lit}(f(x)^l) \cap \mathsf{Lit}(f(x)^r) && \text{by (A.8)}
\end{aligned}
$$

3. We want to show that $\sigma_C(q_x)[\sigma(q_y)/q_y \mid y \in \lhd[C] \setminus C] \in \mathcal{L}^{\text{af}}_{\Box}$. By Lemma 5.37 we know that $\sigma_C(q_x) \in \mathcal{L}^{\text{af}}_{\Box}$ and by our induction hypothesis we know that $\sigma(q_y) \in \mathcal{L}^{\text{af}}_{\Box}$ for each $y \in \lhd[C]\setminus C$. By Lemma 2.29, we know that since $\sigma(q_y) \in \mathcal{L}^{\text{af}}_{\Box}$ and $FV(\sigma_C(q_x)) \cap FV(\sigma_C(q_y)) = \emptyset$ for each $y \in \lhd[C] \setminus C$, we have our desired result.

4. To prove that $\sigma(q_x) \leq 3|X|$ we prove the following stronger claim. Let the *height* $h(D)$ of a cluster $D \in \text{Clus}(X)$ be defined by $h(D) = 0$ if D is a leaf in $(\text{Clus}(X), \prec)$, and $h(D) = 1 + \max\{h(D') \mid D \prec D'\}$ otherwise. Let $\text{Clus}_n := \{D \in \text{Clus}(X) \mid h(D) \leq n\}$. We claim that for all n,

$$\left| \bigcup_{\substack{D \in \text{Clus}_n \\ z \in D}} \text{Clos}(\sigma(q_z)) \right|_{\text{Cl}} \leq 3 \times \left| \bigcup_{D \in \text{Clus}_n} D \right|.$$

If we prove this claim, we have our desired result. To see so, observe the following:

$$|\sigma(q_x)|_{\text{Cl}} = |\,\text{Clos}(\sigma(q_x))|_{\text{Cl}} \leq \left| \bigcup_{\substack{D \in \text{Clus}_{h(C)} \\ z \in D}} \text{Clos}(\sigma(q_z)) \right| \leq 3 \times \left| \bigcup_{D \in \text{Clus}_{h(C)}} D \right|_{\text{Cl}} \leq 3|X|.$$

Thus it suffices to prove the claim. To do so, we proceed by induction on n, we refer to the induction hypothesis as the inner induction hypothesis. If $n = 0$ we want to show

$$\left| \bigcup_{\substack{D \in \text{Clus}_0 \\ z \in D}} \text{Clos}(\sigma(q_z)) \right|_{\text{Cl}} \leq 3 \times \left| \bigcup_{D \in \text{Clus}_0} D \right|.$$

Choose a representative for each cluster D, we will denote this by z_D. We know by Lemma 5.37 that for all $z \in D$ we have

$$\text{Clos}(\sigma_D(q_z)) = \text{Clos}(\sigma_D(q_{z_D})) \tag{A.9}$$

By definition of $\sigma(q_z)$ for $z \in D$ where $h(D) = 0$ (i.e. D is a leaf in $(\mathrm{Clus}(X), \prec)$) we know

$$\begin{aligned}
\left| \bigcup_{\substack{D \in \mathrm{Clus}_0 \\ z \in D}} \mathrm{Clos}(\sigma(q_z)) \right|_{\mathrm{Cl}} &= \left| \bigcup_{\substack{D \in \mathrm{Clus}_0 \\ z \in D}} \mathrm{Clos}(\sigma_D(q_z)) \right|_{\mathrm{Cl}} \\
&= \left| \bigcup_{D \in \mathrm{Clus}_0} \mathrm{Clos}(\sigma_D(q_{z_D})) \right|_{\mathrm{Cl}} && \text{by (A.9)} \\
&\leq \sum_{D \in \mathrm{Clus}_0} |\sigma_D(q_{z_D})|_{\mathrm{Cl}} \\
&\leq \sum_{D \in \mathrm{Clus}_0} 3|D| && \text{Lemma 5.37} \\
&= 3 \times \left| \bigcup_{D \in \mathrm{Clus}_0} D \right|
\end{aligned}$$

Now suppose $n = k + 1$, we want to show that

$$\left| \bigcup_{\substack{D \in \mathrm{Clus}_{k+1} \\ z \in D}} \mathrm{Clos}(\sigma(q_y)) \right|_{\mathrm{Cl}} \leq 3 \times \left| \bigcup_{D \in \mathrm{Clus}_{k+1}} D \right|$$

Observe:

$$\begin{aligned}
&\bigcup_{\substack{D \in \mathrm{Clus}_{k+1} \\ z \in D}} \mathrm{Clos}(\sigma(q_z)) \\
= &\bigcup_{\substack{D \in \mathrm{Clus}(X) \\ h(D) = k+1 \\ z \in D}} \mathrm{Clos}(\sigma(q_z)) \cup \bigcup_{\substack{D \in \mathrm{Clus}_k \\ z \in D}} \mathrm{Clos}(\sigma(q_z)) \\
= &\bigcup_{\substack{D \in \mathrm{Clus}(X) \\ h(D) = k+1 \\ z \in D}} \mathrm{Clos}(\sigma_D(q_z))[\sigma(q_y)/q_y \mid y \in \lhd[D] \setminus D] \cup \bigcup_{\substack{D \in \mathrm{Clus}_k \\ z \in D}} \mathrm{Clos}(\sigma(q_z)) \\
= &\bigcup_{\substack{D \in \mathrm{Clus}(X) \\ h(D) = k+1 \\ z \in D}} \left(\mathrm{Clos}(\sigma_D(q_z))[\sigma(q_y)/q_y \mid y \in \lhd[D] \setminus D] \cup \bigcup_{y \in \lhd D \setminus D} \mathrm{Clos}(\sigma(q_y)) \right) \\
&\quad \cup \bigcup_{\substack{D \in \mathrm{Clus}_k \\ z \in D}} \mathrm{Clos}(\sigma(q_z)) \\
= &\bigcup_{\substack{D \in \mathrm{Clus}(X) \\ h(D) = k+1 \\ z \in D}} \mathrm{Clos}(\sigma_D(q_z))[\sigma(q_y)/q_y \mid y \in \lhd[D] \setminus D] \cup \bigcup_{\substack{D \in \mathrm{Clus}_k \\ z \in D}} \mathrm{Clos}(\sigma(q_z)) \\
= &\bigcup_{\substack{D \in \mathrm{Clus}(X) \\ h(D) = k+1}} \mathrm{Clos}(\sigma_D(q_{z_D}))[\sigma(q_y)/q_y \mid y \in \lhd[D] \setminus D] \cup \bigcup_{\substack{D \in \mathrm{Clus}_k \\ z \in D}} \mathrm{Clos}(\sigma(q_z)) && \text{by (A.9)}
\end{aligned}$$

Taking the cardinality and applying some basic set theory we get

$$\begin{aligned}
\left|\bigcup_{\substack{D\in\mathrm{Clus}_{k+1}\\ z\in D}} \mathrm{Clos}(\sigma(q_z))\right|_{\mathrm{Cl}} &\leq \sum_{\substack{D\in\mathrm{Clus}(X)\\ h(D)=k+1}} |\sigma_D(q_{z_D})|_{\mathrm{Cl}} + \left|\bigcup_{\substack{D\in\mathrm{Clus}_k\\ z\in D}} \mathrm{Clos}(\sigma(q_y))\right|_{\mathrm{Cl}} \\
&\leq \sum_{\substack{D\in\mathrm{Clus}(X)\\ h(D)=k+1}} |\sigma_D(q_{z_D})|_{\mathrm{Cl}} + 3\times\left|\bigcup_{D\in\mathrm{Clus}_k} D\right| && \text{inner IH} \\
&\leq 3\sum_{\substack{D\in\mathrm{Clus}(X)\\ h(D)=k+1}} |D| + 3\times\left|\bigcup_{D\in\mathrm{Clus}_k} D\right| && \text{Lemma 5.37} \\
&= 3\times\left|\bigcup_{D\in\mathrm{Clus}_{k+1}} D\right|
\end{aligned}$$

Thus we have finished the proof of our claim. □