On the Completeness of Interpolation Algorithms

Stefan Hetzl
TU Wien
Vienna, Austria

Raheleh Jalali
Czech Academy of Sciences
Prague, Czechia

ABSTRACT

Craig interpolation is a fundamental property of logics with a plethora of applications from philosophical logic to computer-aided verification. The question of which interpolants can be obtained from an interpolation algorithm is of profound importance. Motivated by this question, we initiate the study of completeness properties of interpolation algorithms. An interpolation algorithm $\mathcal{I}$ is *complete* if, for every interpolant C of an implication $A \to B$, there is a proof P of $A \to B$ such that C is logically equivalent to $\mathcal{I}(P)$. We establish incompleteness and different kinds of completeness results for several standard algorithms for resolution and the sequent calculus for propositional, modal, and first-order logic.

KEYWORDS

(Craig) interpolation, proof theory, sequent calculus, resolution

ACM Reference Format:
Stefan Hetzl and Raheleh Jalali. 2024. On the Completeness of Interpolation Algorithms. In *39th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '24), July 8–11, 2024, Tallinn, Estonia.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3661814.3662112

1 INTRODUCTION

Interpolation is one of the most fundamental properties of classical and many non-classical logics. Proved first by Craig in the 1950s [8], the interpolation theorem has remained central to logic ever since and continues to find new applications, see, e.g., [20] to catch a glimpse of the remarkable breadth of this seminal result. It has widespread applications throughout all areas of logic in philosophy, mathematics, and computer science. For example, the interpolation theorem has strong connections to the foundations of mathematics through its use for proving Beth's definability theorem [5]. It underlies the technique of feasible interpolation for proving lower bounds in proof complexity [17]. It has also become a versatile and indispensable tool in theory reasoning [23, 26], model checking [21], and knowledge representation [1].

A question of profound importance, from a theoretical as well as a practical point of view, is that of the expressive power of interpolation algorithms: Given a particular interpolation algorithm, which interpolants can it compute? This question is relevant for all applications of interpolation and it is gaining importance with the increasing number of applications. In particular, it has been the focus of attention in the Computer-Aided Verification (CAV) community where a lot of work, see, e.g. [9, 11, 13–16, 28], has been devoted to developing algorithms that compute "good" interpolants, i.e., such interpolants that have properties, e.g., in terms of logical strength, that make them well suited for the application at hand. However, from a theoretical point of view, this question has not been investigated to a satisfactory degree yet.

In this paper, we set out to gauge the expressive power of different interpolation algorithms by carrying out a thorough theoretical analysis. To this aim, we initiate the study of completeness properties of interpolation algorithms. An interpolation algorithm outputs an interpolant of an implication $A \to B$ when given a proof of $A \to B$ (in a suitable proof system). We say that an interpolation algorithm $\mathcal{I}$ is *complete* if, for every interpolant C of an implication $A \to B$, there is a proof P of $A \to B$ such that $\mathcal{I}(P)$ is logically equivalent to C. The practical relevance of a completeness result is that it provides a guarantee that, at least in principle, the algorithm allows to find the "good" interpolants, whatever that may mean in the concrete application under consideration.

In this paper, we study the completeness of the standard interpolation algorithms for the sequent calculus and resolution, two of the most important proof systems in mathematical logic and computer science. We establish the following results: after introducing the necessary notions and definitions in Section 2 we will show in Section 3 that the standard interpolation algorithms for resolution and cut-free sequent calculus for classical propositional logic are incomplete. On the other hand, if we weaken the completeness property, it is possible to obtain a positive result for the cut-free sequent calculus: in Section 5 we will show that the standard interpolation algorithm for cut-free sequent calculus is complete up to subsumption for pruned interpolants. This result requires a subtle argument that traces the development of interpolants during cut-elimination. Furthermore, in Section 6 we show that in the sequent calculus with cut, already with atomic cut, the standard interpolation algorithm is complete. We then leave the realm of propositional logic to extend our results to normal modal logics in Section 7 and to classical first-order logic in Section 8. While the results for propositional logic carry over to first-order logic, we also find a new source of incompleteness on the first-order level which implies that standard algorithms for first-order interpolation are incomplete, even in the sequent calculus with atomic cuts. Last, but not least, in Section 9 we show that completeness properties of interpolation algorithms translate directly to completeness properties of Beth's definability theorem.

2 PRELIMINARIES

2.1 Formulas

For the preliminaries, the reader may consult [6, 22, 27]. Let the propositional language $\mathcal{L}_p = \{\bot, \wedge, \vee, \neg\}$. *Propositional variables* or *atoms* are denoted by $p, q, \ldots$, possibly with subscripts. *Formulas* are denoted by $A, B, \ldots$ and defined as usual. Define $A \to B := \neg A \vee B$

LICS '24, July 8–11, 2024, Tallinn, Estonia

ACM ISBN 979-8-4007-0660-8/24/07
https://doi.org/10.1145/3661814.3662112

and $\top := \bot \to \bot$ for any formulas A and B in the language $\mathcal{L}_p$. A *literal* ℓ is either an atom or a negation of an atom. We consider $\bot$ as an atom and $\top$ as a literal. If $\ell = p$, by $\bar{\ell}$ we mean $\neg p$ and if $\ell = \neg p$, by $\bar{\ell}$ we mean p. A *clause* C is a finite disjunction of literals $C = \ell_1 \vee \cdots \vee \ell_n$, sometimes simply written as $C = \{\ell_1, \ldots, \ell_n\}$. The clause $C \cup \{\ell\}$ is often abbreviated by C, ℓ. We assume two clauses are equal if they have the same set of literals. A *positive* literal is an unnegated atom and a *negative* literal is a negated atom. By convention, $\wedge\emptyset := \top$ and $\vee\emptyset := \bot$. For a formula A, the set of its *variables*, denoted by $V(A)$, is defined recursively; $V(\bot) = \emptyset$; $V(p) = \{p\}$ for any atom p; $V(\neg A) = V(A)$, and $V(A \circ B) = V(A) \cup V(B)$ for $\circ \in \{\wedge, \vee\}$. For a multiset Γ define $V(\Gamma) = \bigcup_{\gamma \in \Gamma} V(\gamma)$.

Definition 1. A logic L has the *Craig Interpolation Property* (*CIP*) if for any formulas A and B if $A \to B \in L$ then there exists a formula C such that $V(C) \subseteq V(A) \cap V(B)$ and $A \to C \in L$ and $C \to B \in L$.

2.2 Resolution

Propositional *resolution*, R, is one of the weakest proof systems. However, it is very important in artificial intelligence as it is useful in automated theorem proving and SAT solving. Resolution operates on clauses.

By a *clause set* we mean a set $\mathcal{C} = \{C_1, \ldots, C_n\}$ of clauses $C_i = \{\ell_{i1}, \ldots, \ell_{ik_i}\}$ and the *formula interpretation* of $\mathcal{C}$ is $\bigwedge_{i=1}^{n} \bigvee_{j=1}^{k_i} \ell_{ij}$. We say a clause set is *logically equivalent* to a formula φ when its formula interpretation is logically equivalent to φ. A formula is in *conjunctive normal form* if it is a conjunction of disjunctions of literals, i.e., has the form $\bigwedge_{i=1}^{n} \bigvee_{j=1}^{k_i} \ell_{ij}$ for some $1 \leq n$ and $1 \leq k_i$ for each $1 \leq i \leq n$, where ℓ_{ij}'s are literals. Equivalently, we may use the clause set $\mathcal{C} = \{C_i \mid 1 \leq i \leq n\}$, where each $C_i = \ell_{i1} \vee \cdots \vee \ell_{ik_i}$. We use these two formats interchangeably for the conjunctive normal form of a formula. By convention, the empty clause set is logically equivalent to $\top$ and the empty clause is logically equivalent to $\bot$. Similarly, $\top$ and $\bot$ are considered as formulas in conjunctive normal form.

A *resolution proof*, also called a *resolution refutation*, shows the unsatisfiability of a set of initial clauses by starting with these clauses and deriving new clauses by the *resolution rule*

$$\frac{C \cup \{p\} \qquad D \cup \{\neg p\}}{C \cup D}$$

until the empty clause is derived, where C and D are clauses. This rule is obviously sound, i.e., if a truth assignment satisfies both the premises, then the conclusion is also satisfied by the same assignment. Given a set of clauses, it is possible to use the resolution rule and derive new clauses. Specifically, we may derive the empty clause, denoted by $\bot$. Thus, we can interpret resolution as a refutation system; instead of proving a formula A is true we prove that $\neg A$ is unsatisfiable. We write $\neg A$ as a conjunction of disjunctions of literals and take each conjunct as a clause. Then, from these clauses, we derive the empty clause using the resolution rule only. It is also possible to add the *weakening rule* to the resolution system:

$$\frac{C}{C \cup D}$$

for arbitrary clauses C and D. The new system is called *resolution with weakening*.

Interpolation algorithm for resolution [18, 24]: Suppose a resolution proof P of the empty clause from the clauses $A_i(\bar{p}, \bar{q})$ and $B_j(\bar{p}, \bar{r})$ is given, where $i \in I$, $j \in J$, and $\bar{p}, \bar{q}, \bar{r}$ are disjoint sets of atoms. The only atoms common between the two sets of clauses are $\bar{p}$. Define a ternary connective sel as $sel(A, x, y) = (\neg A \to x) \wedge (A \to y) = (A \vee x) \wedge (\neg A \vee y)$, for formulas A, x, and y. For instance, by definition, we have $sel(\bot, x, y) = x$, $sel(\top, x, y) = y$, $sel(A, \bot, \top) = A$, and $sel(A, \top, \bot) = \neg A$. The interpolation algorithm operates as follows. Assign the constant $\bot$ to clauses A_i for each $i \in I$ and assign $\top$ to clauses B_j for $j \in J$. Then:

(1) Suppose the resolution rule is of the form

$$\frac{\Gamma, p_k \qquad \Delta, \neg p_k}{\Gamma, \Delta}$$

where $p_k \in \bar{p}$. If we have assigned x to the premise Γ, p_k and y to $\Delta, \neg p_k$ then we assign $z = sel(p_k, x, y)$ to the conclusion Γ, Δ.

(2) Suppose the resolution rule is of the form

$$\frac{\Gamma, q_k \qquad \Delta, \neg q_k}{\Gamma, \Delta}$$

where $q_k \in \bar{q}$. If we have assigned x to the premise Γ, q_k and y to $\Delta, \neg q_k$, then we will assign $x \vee y$ to the conclusion Γ, Δ.

(3) Suppose the resolution rule is of the form

$$\frac{\Gamma, r_k \qquad \Delta, \neg r_k}{\Gamma, \Delta}$$

where $r_k \in \bar{r}$. If we have assigned x to the premise Γ, r_k and y to $\Delta, \neg r_k$, then we will assign $x \wedge y$ to the conclusion Γ, Δ.

This algorithm outputs an interpolant of the valid formula $A \to B$, given a resolution refutation of the unsatisfiable formula $A \wedge \neg B$.

The interpolation algorithm for resolution with weakening is defined as above except that when the weakening rule is used

$$\frac{\Gamma}{\Gamma, \Delta}$$

and we have assigned x to the premise Γ, we will also assign x to the conclusion Γ, Δ.

Theorem 2. *[18, 24] Let π be a resolution refutation of the set of clauses $\{A_i(\bar{p}, \bar{q}) \mid i \in I\}$ and $\{B_j(\bar{p}, \bar{r}) \mid j \in J\}$. Then, the interpolation algorithm outputs an interpolant for the valid formula $\bigwedge_{i \in I} A_i(\bar{p}, \bar{q}) \to \bigvee_{j \in J} \neg B_j(\bar{p}, \bar{r})$.*

In refutational theorem provers, interpolation is also often formulated in terms of reverse interpolants. A *reverse interpolant* of an unsatisfiable formula $A \wedge B$ is a formula C with $V(C) \subseteq V(A) \cap V(B)$ such that $A \models C$ and $B \models \neg C$. Note that C is a reverse interpolant of $A \wedge B$ iff it is an interpolant of $A \to \neg B$.

2.3 Sequent calculus

A *sequent* is an expression of the form $\Gamma \Rightarrow \Delta$ where Γ, called the *antecedent*, and Δ, called the *succedent*, are multisets of formulas and the *formula interpretation* of the sequent is $\bigwedge \Gamma \to \bigvee \Delta$. In the propositional sequent calculus **LK** each proof is represented as a tree. The nodes correspond to sequents and the root of the proof tree, at the bottom, is the *end-sequent* and it is the sequent proved by the proof. The topmost nodes of the tree, the leaves, are called the *initial sequents* or *axioms*. Apart from the axioms, each sequent in an **LK** proof must be inferred from one of the inference rules provided in Table 1. An *inference rule* is an expression of

$$p \Rightarrow p \ \text{(Ax)} \qquad \bot \Rightarrow \ (\bot)$$

$$\frac{\Gamma \Rightarrow \Delta}{A, \Gamma \Rightarrow \Delta}\ (Lw) \qquad \frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, A}\ (Rw)$$

$$\frac{A, A, \Gamma \Rightarrow \Delta}{A, \Gamma \Rightarrow \Delta}\ (Lc) \qquad \frac{\Gamma \Rightarrow \Delta, A, A}{\Gamma \Rightarrow \Delta, A}\ (Rc)$$

$$\frac{A, \Gamma \Rightarrow \Delta}{A \wedge B, \Gamma \Rightarrow \Delta}\ (L\wedge_1) \qquad \frac{B, \Gamma \Rightarrow \Delta}{A \wedge B, \Gamma \Rightarrow \Delta}\ (L\wedge_2)$$

$$\frac{\Gamma \Rightarrow \Delta, A \qquad \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \wedge B}\ (R\wedge) \qquad \frac{\Gamma \Rightarrow \Delta, A}{\Gamma \Rightarrow \Delta, A \vee B}\ (R\vee_1)$$

$$\frac{\Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \vee B}\ (R\vee_2) \qquad \frac{A, \Gamma \Rightarrow \Delta \qquad B, \Gamma \Rightarrow \Delta}{A \vee B, \Gamma \Rightarrow \Delta}\ (L\vee)$$

$$\frac{\Gamma \Rightarrow \Delta, A}{\neg A, \Gamma \Rightarrow \Delta}\ (L\neg) \qquad \frac{\Gamma, A \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \neg A}\ (R\neg)$$

$$\frac{\Gamma \Rightarrow \Delta, A \qquad A, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}\ (cut)$$

Table 1: Propositional LK. In (Ax), p is an atom. The formula A in the cut rule is called the *cut formula*.

the form $\frac{S_1}{S}$ or $\frac{S_1\ S_2}{S}$ indicating that the conclusion S is inferred from the premise S_1 or the premises S_1 and S_2. In Table 1, rules are schematic, A and B denote arbitrary formulas, and Γ and Δ denote arbitrary multisets of formulas. The first row in Table 1 are axioms. The weakening and contraction rules and the cut rule are *structural* inference rules and the rest *logical*. By *length* of a formula A, denoted by $|A|$, we mean the number of symbols in it. Similarly, we define the length of a multiset, sequent, and a proof. The *size* of a tree is the number of nodes in it. The *depth* or *height* of a proof tree is the maximum length of the branches in the tree, where the *length* of a branch is the number of nodes in the branch minus one.

We will consider the $G1$ systems for our logics, as presented in [27, Section 3.1]. However, the results of this paper do not depend significantly on the concrete version of the sequent calculus. The system presented in Table 1 is the propositional **LK**. In each rule, the upper sequent(s) is (are) *premise(s)*, and the lower sequent is the *conclusion*. Γ and Δ are called the *context*. The formula in the conclusion of the rule not belonging to the contexts is called the *main* formula. The formulas in the premises not in the contexts are called the *active* or *auxiliary* formulas. The cut rule is called *atomic* when the cut formula is an atom or $\bot$ or $\top$. Denote **LK** with cuts only on atoms (resp. literals) by $\mathbf{LK}^{\mathrm{at}}$ (resp. $\mathbf{LK}^{\mathrm{lit}}$) and denote cut-free **LK** by $\mathbf{LK}^{-}$.

Maehara interpolation algorithm for $\mathbf{LK}^{\mathrm{at}}$, $\mathbf{LK}^{\mathrm{lit}}$ and $\mathbf{LK}^{\mathrm{m}}$: A *split sequent* is an expression of the from $\Gamma_1; \Gamma_2 \Rightarrow \Delta_1; \Delta_2$ such that $\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2$ is a sequent. The partition $\Gamma_1, \Gamma_2 = \Gamma$ and $\Delta_1, \Delta_2 = \Delta$ is called the *Maehara partition* of the sequent $\Gamma \Rightarrow \Delta$. A formula A is on the *left-hand side* (resp. *right-hand side*) of the Maehara partition if $A \in \Gamma_1 \cup \Delta_1$ (resp. $A \in \Gamma_2 \cup \Delta_2$). A proof π of a split sequent $\Gamma_1; \Gamma_2 \Rightarrow \Delta_1; \Delta_2$ is called *monochromatic* when for each cut rule used in π, the cut formula A has the following condition: either $V(A) \subseteq V(\Gamma_1) \cup V(\Delta_1)$ or $V(A) \subseteq V(\Gamma_2) \cup V(\Delta_2)$. Let $\mathbf{LK}^{\mathrm{m}}$ be the set of all monochromatic proofs. Let $G \in \{\mathbf{LK}^{\mathrm{at}}, \mathbf{LK}^{\mathrm{lit}}, \mathbf{LK}^{\mathrm{m}}\}$. We define the Maehara algorithm for G. Suppose a valid sequent $\Gamma \Rightarrow \Delta$ is given. Let π be a proof of the split sequent $\Gamma_1; \Gamma_2 \Rightarrow \Delta_1; \Delta_2$ in G. In the following, we will define the Maehara interpolation algorithm, denoted by $\mathcal{M}$. More concretely, we define the formula $\mathcal{M}(\pi) = C$, called the *interpolant*, recursively, such that $V(C) \subseteq V(\Gamma_1 \cup \Delta_1) \cap V(\Gamma_2 \cup \Delta_2)$ and $G \vdash \Gamma_1 \Rightarrow \Delta_1, C$ and $G \vdash C, \Gamma_2 \Rightarrow \Delta_2$. Let us denote the interpolant C for $\Gamma_1; \Gamma_2 \Rightarrow \Delta_1; \Delta_2$ by $\Gamma_1; \Gamma_2 \overset{C}{\Longrightarrow} \Delta_1; \Delta_2$. An interpolant for the provable formula $A \to B$ is defined as $\mathcal{M}(\pi)$, where π is the proof of the split sequent $A; \Rightarrow ; B$.

• If π is an axiom, it is either (Ax) or ($\bot$). Then, $\mathcal{M}(\pi)$ is defined as follows based on the occurrence of the atom p or $\bot$ in the partitions:

$$p; \overset{\bot}{\Longrightarrow} p; \quad ; p \overset{\top}{\Longrightarrow}; p \quad \bot; \overset{\bot}{\Longrightarrow}; \quad p; \overset{p}{\Longrightarrow}; p \quad ; p \overset{\neg p}{\Longrightarrow} p; \quad ; \bot \overset{\top}{\Longrightarrow};$$

• If the last rule $\mathcal{R}$ used in π is one of the one-premise rules, i.e., $(R\vee_1)$, $(R\vee_2)$, $(L\wedge_1)$, $(L\wedge_2)$, (Lw), (Rw), (Lc), (Rc), $(L\neg)$, or $(R\neg)$, then the interpolant of the premise also works as the interpolant for the conclusion, i.e., define $\mathcal{M}(\pi) = \mathcal{M}(\pi')$ where π' is the proof of the premise of $\mathcal{R}$.

• If the last rule in π is $(R\wedge)$, there are two cases based on the occurrence of the main formula in the conclusion:

$$\frac{\Gamma_1; \Gamma_2 \overset{C}{\Longrightarrow} \Delta_1, A; \Delta_2 \qquad \Gamma_1; \Gamma_2 \overset{D}{\Longrightarrow} \Delta_1, B; \Delta_2}{\Gamma_1; \Gamma_2 \overset{C\vee D}{\Longrightarrow} \Delta_1, A \wedge B; \Delta_2}$$

or

$$\frac{\Gamma_1; \Gamma_2 \overset{C}{\Longrightarrow} \Delta_1; A, \Delta_2 \qquad \Gamma_1; \Gamma_2 \overset{D}{\Longrightarrow} \Delta_1; B, \Delta_2}{\Gamma_1; \Gamma_2 \overset{C\wedge D}{\Longrightarrow} \Delta_1; A \wedge B, \Delta_2}$$

• If the last rule in π is $(L\vee)$, then:

$$\frac{\Gamma_1, A; \Gamma_2 \overset{C}{\Longrightarrow} \Delta_1; \Delta_2 \qquad \Gamma_1, B; \Gamma_2 \overset{D}{\Longrightarrow} \Delta_1; \Delta_2}{\Gamma_1, A \vee B; \Gamma_2 \overset{C\vee D}{\Longrightarrow} \Delta_1; \Delta_2}$$

or

$$\frac{\Gamma_1; A, \Gamma_2 \overset{C}{\Longrightarrow} \Delta_1; \Delta_2 \qquad \Gamma_1; B, \Gamma_2 \overset{D}{\Longrightarrow} \Delta_1; \Delta_2}{\Gamma_1; A \vee B, \Gamma_2 \overset{C\wedge D}{\Longrightarrow} \Delta_1; \Delta_2}$$

• Suppose the last rule in π is an instance of a cut rule in G. Let A be the cut formula. Denote $V_1 = V(\Gamma_1 \cup \Delta_1)$ and $V_2 = V(\Gamma_2 \cup \Delta_2)$. Then, either $V(A) \subseteq V_1$ or $V(A) \subseteq V_2$ or $V(A) \subseteq V_1 \cap V_2$. If $V(A) \subseteq V_1$, define

$$\frac{\Gamma_1; \Gamma_2 \overset{C}{\Longrightarrow} \Delta_1, A; \Delta_2 \qquad \Gamma_1, A; \Gamma_2 \overset{D}{\Longrightarrow} \Delta_1; \Delta_2}{\Gamma_1; \Gamma_2 \overset{C\vee D}{\Longrightarrow} \Delta_1; \Delta_2}$$

If $V(A) \subseteq V_2$, define

$$\frac{\Gamma_1; \Gamma_2 \overset{E}{\Longrightarrow} \Delta_1; A, \Delta_2 \qquad \Gamma_1; \Gamma_2, A \overset{F}{\Longrightarrow} \Delta_1; \Delta_2}{\Gamma_1; \Gamma_2 \overset{E\wedge F}{\Longrightarrow} \Delta_1; \Delta_2}$$

And if $V(A) \subseteq V_1 \cap V_2$, choose either case.

Theorem 3. *Let π be a proof of $A \to B$ in $G \in \{\mathbf{LK}^{\mathrm{at}}, \mathbf{LK}^{\mathrm{lit}}, \mathbf{LK}^{\mathrm{m}}\}$. Then, $\mathcal{M}(\pi)$ outputs an interpolant of the formula $A \to B$ with $|\mathcal{M}(\pi)| \leq |\pi|$.*

Proof in the appendix. □

A formula is in *negation normal form (NNF)* when the negation is only allowed on atoms and the other connectives in the formula are disjunctions and conjunctions. By an easy investigation of the form of the interpolants constructed in the Maehara algorithm, we

see that in binary rules, based on the position of the active formulas in the premises, the interpolant of the conclusion will be either the conjunction or disjunction of the interpolants of the premises.

Corollary 4. *The interpolants constructed via the Maehara algorithm are in NNF.*

Remark 5. By carefully investigating the definition of the Maehara algorithm for $G \in \{\mathbf{LK}^{\mathrm{at}}, \mathbf{LK}^{\mathrm{lit}}, \mathbf{LK}^{\mathrm{m}}\}$, we see that the following happens. If the cut formula is on the left-hand side (resp. right-hand side) of the Maehara partition, then the interpolant of the conclusion of the cut rule is the disjunction (resp. conjunction) of the interpolants of the premises of the cut rule. We will use this observation in the following results.

3 SIMPLE INCOMPLETENESS RESULTS

We start with a simple observation. Some interpolation algorithms return only formulas of a particular shape. For example, as observed in Corollary 4, Maehara algorithm $\mathcal{M}$ only returns formulas in NNF. This immediately yields a first incompleteness result as follows.

Definition 6. An interpolation algorithm $\mathcal{I}$ is called *syntactically complete* if for every valid implication $A \to B$ and every interpolant C of $A \to B$ there is a proof π such that $C = \mathcal{I}(\pi)$.

Observation 7. *$\mathcal{M}$ is syntactically incomplete.*

Proof. $\neg\neg p$ is an interpolant of $p \to p$ but, since it is not in NNF, there is no π such that $\mathcal{M}(\pi) = \neg\neg p$. □

This simple observation makes clear that in many cases we will only obtain an interesting question if we ask for completeness up to some equivalence relation coarser than syntactic equality. The first candidate that comes to mind, and the most important case we will deal with in this paper, is logical equivalence.

Definition 8. An interpolation algorithm $\mathcal{I}$ is called *complete up to logical equivalence* if for every valid implication $A \to B$ and every interpolant C of $A \to B$ there is a proof π s.t. C is logically equivalent to $\mathcal{I}(\pi)$.

Since, in this paper, we will mostly deal with completeness up to logical equivalence we will usually simply say *completeness* instead.

A useful basis for negative results is the implication $p \wedge q \to p \vee q$. It has the four interpolants $p \wedge q, p, q, p \vee q$ but, in proof systems that do not allow overly redundant proofs there are essentially only two different proofs: one that proceeds via p (and has p as interpolant) and one that proceeds via q (and has q as interpolant). This can be used to obtain incompleteness results for interpolation in the cut-free sequent calculus and resolution in a strong sense.

Definition 9. The notions of *positive* and *negative* subformulas of a formula A are defined as follows. A is a positive subformula of itself. If $B \circ C$ is a positive (resp. negative) subformula of A, so are B and C, where $\circ \in \{\wedge, \vee\}$. If $\neg B$ is a positive (resp. negative) subformula of A, then B is a negative (resp. positive) subformula of A. A formula B is a *subformula* of the formula A if it is a positive or negative subformula of A. Positive and negative occurrences of formulas in $\Gamma \Rightarrow \Delta$ are defined as positive and negative occurrences of formulas in $\neg \bigwedge \Gamma \vee \bigvee \Delta$.

Proposition 10. *Maehara interpolation in $\mathbf{LK}^{-}$ is not complete.*

Proof. We use the subformula property of cut-free proofs (see Proposition 4.2.1 and its Corollary in [27]). Any cut-free proof of $\pi : p \wedge q \Rightarrow p \vee q$ has the following properties:

- For any sequent $\Gamma \Rightarrow \Delta$ in π, the formulas of Γ occur positively in $p \wedge q$ and the formulas of Δ occur positively in $p \vee q$.
- The rules $(L\neg)$ and $(R\neg)$ are not used in π.
- As the formula $p \wedge q$ occurs negatively in $p \wedge q \Rightarrow p \vee q$ then $p \wedge q$ is only obtained by using left rules, i.e., either $(L\wedge)$, (Lc), or (Lw). And the rule $(L\vee)$ cannot be used.
- Similarly, for $p \vee q$, it can be only obtained using the right rules $(R\vee)$, (Rc), or (Rw). And the rule $(R\wedge)$ cannot be used.
- Axioms in π have one of the forms: $p \Rightarrow p$ or $q \Rightarrow q$. The axioms $\bot \Rightarrow$, $p \wedge q \Rightarrow p \wedge q$ or $p \vee q \Rightarrow p \vee q$ cannot appear in π, because of the subformula property and the positive or negative occurrences.

Thus, the axioms of π are either $p \Rightarrow p$ or $q \Rightarrow q$. And, as all the rules in π are one-premise rules, the interpolants $p \wedge q$ or $p \vee q$ can never be obtained. □

Proposition 11. *Standard interpolation in propositional resolution is not complete.*

Proof. The formula $p \wedge q \to p \vee q$ has the interpolants p, q, $p \wedge q$, or $p \vee q$. However, neither $p \wedge q$ nor $p \vee q$ can be read off from a resolution proof. To form a resolution refutation, as $p \wedge q \to p \vee q$ is classically valid, the set of clauses $A_1 = \{p\}$, $A_2 = \{q\}$, $B_1 = \{\neg p\}$, and $B_2 = \{\neg q\}$ is unsatisfiable. The set of atoms common in both A_1, A_2 and B_1, B_2 is $\{p, q\}$. A resolution refutation of the mentioned set of clauses has either of the following forms:

$$\frac{p \qquad \neg p}{\emptyset} \qquad \frac{q \qquad \neg q}{\emptyset}$$

Then, based on the algorithm we assign the constant $\bot$ on clauses A_1 and A_2, and assign $\top$ on clauses B_1 and B_2. Then for the left refutation we obtain $sel(p, \bot, \top)$ which is logically equivalent to p and for the right refutation we obtain $sel(q, \bot, \top)$ which is logically equivalent to q on the conclusion as the interpolant in each case. □

Note that the above proofs are quite general. In the case of resolution, it shows the incompleteness of any interpolation algorithm which, when given a resolution refutation in some set of propositional variables V will output a formula in V. This is arguably the case for any reasonable interpolation algorithm. In the case of sequent calculus, it shows the incompleteness of any interpolation algorithm that produces only such interpolants that contain only atoms that occur in axioms of the input proof.

Remark 12. It is worth mentioning that the above example does not prove the incompleteness of the system resolution with weakening. Because for the same set of clauses $A_1 = \{p\}$, $A_2 = \{q\}$, $B_1 = \{\neg p\}$, and $B_2 = \{\neg q\}$, to form the interpolant $p \vee q$ we can take the following resolution refutation:

$$\dfrac{\dfrac{\dfrac{p}{p, q}\,(w) \qquad \neg p}{q} \qquad \neg q}{\emptyset} \quad \overset{\text{interpolant}}{\longrightarrow} \quad \dfrac{\dfrac{\dfrac{\bot}{\bot} \qquad \top}{sel(p, \bot, \top) = p} \qquad \top}{sel(q, p, \top) = p \vee q}$$

Question 13. *Is standard interpolation in resolution with weakening complete?*

Question 14. *Are the standard interpolation algorithms in algebraic proof systems, such as the cutting planes system [24], complete?*

4 INTERPOLANTS IN THE SEQUENT CALCULUS

In this section, we prove some general results about interpolants in the sequent calculus in preparation for our main results in Sections 5 and 6.

For some aspects of the following proofs, it will be useful to distinguish between different occurrences of a formula in an **LK** proof. We use lowercase Greek letters like $\mu, \nu, \ldots$ to denote *formula occurrences*. There is a natural ancestor relation on the set of formula occurrences in a proof: if a formula occurrence μ is the main occurrence of a logical or structural inference rule and ν is an auxiliary occurrence then ν is a *direct ancestor* of μ. Moreover, if μ is a formula occurrence in the context of the conclusion sequent of an inference rule and ν is a corresponding formula occurrence in the context of a premise sequent, then ν is a *direct ancestor* of μ. The *ancestor* relation is then the reflexive and transitive closure of the direct ancestor relation. We write A_μ in a proof to denote an occurrence μ of a formula A. We also write L_i for a label L of an inference rule to give this inference the name i.

Example 15. In the proof

$$\dfrac{\dfrac{\dfrac{A \Rightarrow A_{\nu_1}}{A, \neg A \Rightarrow}\,(L\neg)}{A_{\mu_1}, \neg A \wedge \neg B \Rightarrow}\,(L\wedge_1) \quad \dfrac{\dfrac{B \Rightarrow B_{\nu_2}}{B, \neg B \Rightarrow}\,(L\neg)}{B_{\mu_2}, \neg A \wedge \neg B \Rightarrow}\,(L\wedge_2)}{\dfrac{\neg A \wedge \neg B, (A \vee B)_\mu \Rightarrow}{A \vee B \Rightarrow (\neg(\neg A \wedge \neg B))_\nu}\,(R\neg)}\,(L\vee)_i$$

the formula occurrence μ has two direct ancestors: μ_1 and μ_2. The occurrences μ, μ_1, and μ_2 are the active formulas of the inference i. The formula occurrence ν has eight ancestors, including itself, and the formula occurrences ν_1 and ν_2.

We will often work with formulas in conjunctive normal form as a convenient representative for a class of formulas up to logical equivalence. Many of the following results do not depend strongly on the shape and could be adapted to other normal forms. Let L be a set of atoms. By $\mathcal{A}$ *is a clause set in the language* L we mean every literal in $\mathcal{A}$ is either an atom or negation of an atom in L. If $\mathcal{C}$ and $\mathcal{D}$ are clause sets, define $\mathcal{C} \times \mathcal{D} := \{C \cup D \mid C \in \mathcal{C} \text{ and } D \in \mathcal{D}\}$.

Definition 16. We define the function CNF, which maps formulas in negation normal form to clause sets recursively: $\mathrm{CNF}(\top) = \emptyset$, $\mathrm{CNF}(\bot) = \{\emptyset\}$, $\mathrm{CNF}(\ell) = \{\ell\}$, $\mathrm{CNF}(A \wedge B) = \mathrm{CNF}(A) \cup \mathrm{CNF}(B)$, and $\mathrm{CNF}(A \vee B) = \mathrm{CNF}(A) \times \mathrm{CNF}(B)$, where ℓ is a literal and A and B are formulas.

Observation 17. *Let A, B, C be formulas, let ℓ be a literal, and let $\circ \in \{\wedge, \vee\}$. Then:*

(1) $\mathrm{CNF}(A)$ *is logically equivalent to* A
(2) $\mathrm{CNF}(A \circ B) = \mathrm{CNF}(B \circ A)$
(3) $\mathrm{CNF}((A \circ B) \circ C) = \mathrm{CNF}(A \circ (B \circ C))$
(4) $\mathrm{CNF}(A \wedge A) = \mathrm{CNF}(A)$
(5) $\mathrm{CNF}(\ell \vee \ell) = \mathrm{CNF}(\ell)$
(6) $\mathrm{CNF}(A \wedge \top) = \mathrm{CNF}(A)$
(7) $\mathrm{CNF}(A \vee \bot) = \mathrm{CNF}(A)$

Proof in the appendix. □

In the upcoming Obervation 18 and Lemma 19 we prove two results about partitions of the end-sequent which determine the interpolant independently of the proof.

Observation 18. *Let Γ and Δ be multisets of formulas.*

(1) If π is an $\mathbf{LK}^-$ *proof of* $\Gamma; \Rightarrow \Delta$; *then* $\mathrm{CNF}(\mathcal{M}(\pi)) = \{\emptyset\}$.
(2) If σ is an $\mathbf{LK}^-$ *proof of* $; \Gamma \Rightarrow ; \Delta$ *then* $\mathrm{CNF}(\mathcal{M}(\pi)) = \emptyset$.

Proof. In case (1) all inferences work on the left-hand side of the Maehara partition. Therefore all axioms have interpolant $\bot$ and binary inferences induce disjunctions. Unary inferences do not modify the interpolant. Therefore $\mathrm{CNF}(\mathcal{M}(\pi)) = \mathrm{CNF}(\bot \vee \cdots \bot) = \{\emptyset\}$. Case (2) is symmetric. □

Lemma 19. *Let A be a formula, let $\{\ell_1, \ldots, \ell_n\}$ be a non-tautological clause, and let π be an* $\mathbf{LK}^-$ *proof of $A; \Rightarrow; \ell_1, \ldots, \ell_n$. Then we have $\mathrm{CNF}(\mathcal{M}(\pi)) = \{M\}$ for some clause M with $M \subseteq \{\ell_1, \ldots, \ell_n\}$.*

Proof. First note that a formula occurrence in π is an ancestor of A (resp. of the ℓ_i's) iff it is on the left-hand side (resp. right-hand side) of the Maehara partition. All binary inferences in π operate on ancestors of A and thus induce disjunctions in the computation of $\mathcal{M}(\pi)$. Therefore, $\mathcal{M}(\pi) = \bigvee_{S \in \mathcal{S}} \mathcal{M}(S)$ where $\mathcal{S}$ is the set of initial split sequents in π. We make a case distinction on the form of an $S \in \mathcal{S}$.

(1) If S is $B; \overset{\bot}{\Longrightarrow} B$; then the interpolant is $\bot$.
(2) The case of S being $; B \overset{\top}{\Longrightarrow}; B$ is impossible because it would entail that both occurrences of B are ancestors of the ℓ_i's and hence $\{\ell_1, \ldots, \ell_n\}$ would be tautological.
(3) If S is $B; \overset{B}{\Longrightarrow}; B$ then the occurrence of B on the succedent of the sequent is ancestor of ℓ_j for some $j \in \{1, \ldots, n\}$ and thus the interpolant is $B = \ell_j$.
(4) If S is $; B \overset{\neg B}{\Longrightarrow} B$; then the occurrence of B on the antecedent of the sequent is ancestor of ℓ_j for some $j \in \{1, \ldots, n\}$ and thus the interpolant is $\neg B = \ell_j$.
(5) If S is $\bot; \overset{\bot}{\Longrightarrow}$ then the interpolant is $\bot$.
(6) The case of S being $; \bot \overset{\top}{\Longrightarrow}$ is impossible because it would entail that $\ell_j = \neg\bot$ for some $j \in \{1, \ldots, n\}$ and hence $\{\ell_1, \ldots, \ell_n\}$ would be tautological.

We have thus shown that $\mathcal{M}(S) \in \{\bot, \ell_1, \ldots, \ell_n\}$ for all $S \in \mathcal{S}$. So, by Observation 17/(2),(3),(5),(7), we have $\mathrm{CNF}(\mathcal{M}(\pi)) = \{M\}$ for some $M \subseteq \{\ell_1, \ldots, \ell_n\}$. □

The next useful result is that $\mathcal{M}$ is just as complete in $\mathbf{LK}^{\mathrm{at}}$ as it is in $\mathbf{LK}^{\mathrm{lit}}$. To show this, we first need a version of the inversion lemma for negation that preserves the interpolant.

Lemma 20. *If π is an* $\mathbf{LK}^{\mathrm{m}}$ *proof with monochrome cuts of*

(1) $\Gamma_1; \Gamma_2 \Rightarrow \Delta_1; \Delta_2, \neg A$
(2) $\Gamma_1; \Gamma_2 \Rightarrow \Delta_1, \neg A; \Delta_2$
(3) $\Gamma_1; \Gamma_2, \neg A \Rightarrow \Delta_1; \Delta_2$
(4) $\Gamma_1, \neg A; \Gamma_2 \Rightarrow \Delta_1; \Delta_2$

then there is an $\mathbf{LK}^{\mathrm{m}}$ *proof with monochrome cuts* π' *of*

(1) $\Gamma_1;\Gamma_2, A \Rightarrow \Delta_1;\Delta_2$
(2) $\Gamma_1, A;\Gamma_2 \Rightarrow \Delta_1;\Delta_2$
(3) $\Gamma_1;\Gamma_2 \Rightarrow \Delta_1;\Delta_2, A$
(4) $\Gamma_1;\Gamma_2 \Rightarrow \Delta_1, A;\Delta_2$

with $\mathcal{M}(\pi') = \mathcal{M}(\pi)$ *and* $|\pi'| \leq 2|\pi|$.

Proof in the appendix. □

Lemma 21. *If* π *is an* $\mathbf{LK}^{\mathrm{lit}}$ *proof of* $\Gamma_1;\Gamma_2 \Rightarrow \Delta_1;\Delta_2$ *then there is an* $\mathbf{LK}^{\mathrm{at}}$ *proof* π' *of* $\Gamma_1;\Gamma_2 \Rightarrow \Delta_1;\Delta_2$ *with* $\mathrm{CNF}(\mathcal{M}(\pi')) = \mathrm{CNF}(\mathcal{M}(\pi))$ *and* $|\pi'| \leq 2|\pi|$.

Proof. As all the cuts in π are on literals, we apply Lemma 20 to each instance of a cut rule on negative literals in the proof to obtain a proof with only atomic cuts. Take a topmost instance of a cut rule where the cut formula is a negative literal. Call this subproof σ. Based on the position of the cut formula in the Maehara partition, σ either looks like

$$\dfrac{\begin{array}{cc}\sigma_1 & \sigma_2\\ \Gamma_1;\Gamma_2 \overset{I}{\Longrightarrow} \Delta_1, \neg p;\Delta_2 & \Gamma_1, \neg p;\Gamma_2 \overset{J}{\Longrightarrow} \Delta_1;\Delta_2\end{array}}{\Gamma_1;\Gamma_2 \overset{I\vee J}{\Longrightarrow} \Delta_1;\Delta_2}$$

or

$$\dfrac{\begin{array}{cc}\sigma_1 & \sigma_2\\ \Gamma_1;\Gamma_2 \overset{I}{\Longrightarrow} \Delta_1;\neg p, \Delta_2 & \Gamma_1;\Gamma_2, \neg p \overset{J}{\Longrightarrow} \Delta_1;\Delta_2\end{array}}{\Gamma_1;\Gamma_2 \overset{I\wedge J}{\Longrightarrow} \Delta_1;\Delta_2}$$

where p is an atom and the cut formula is $\neg p$. In the first case, we apply Lemma 20 to σ_1 and σ_2 to get σ_1' and σ_2' and use the cut to get the end sequent of the proof σ:

$$\dfrac{\begin{array}{cc}\sigma_1' & \sigma_2'\\ \Gamma_1, p;\Gamma_2 \overset{I}{\Longrightarrow} \Delta_1;\Delta_2 & \Gamma_1;\Gamma_2 \overset{J}{\Longrightarrow} \Delta_1, p;\Delta_2\end{array}}{\Gamma_1;\Gamma_2 \overset{I\vee J}{\Longrightarrow} \Delta_1;\Delta_2}$$

Now, this cut is atomic, the cut formula is p, and the interpolant remains the same up to commutativity of conjunction. The other case is similar. We apply the same process to every cut on a negated literal in π, resulting in a proof with only atomic cuts. □

The main technical lemma of Section 5, Lemma 38, will be shown by carrying out a cut-elimination argument on a carefully chosen class of proofs. This class on the one hand is large enough to permit an embedding of all pruned interpolants, but on the other hand small enough to exhibit a very nice behaviour during cut-elimination: the interpolant of the reduced proof is subsumed by the interpolant of the original proof. We now proceed to introduce this class of proofs, called "tame" proofs, which is a new invariant for cut-elimination.

Definition 22. We say that a cut is of type R if it is of the form

$$\dfrac{\Gamma_1;\Gamma_2 \Rightarrow \Delta_1;\Delta_2, C \quad \Gamma_1;\Gamma_2, C \Rightarrow \Delta_1;\Delta_2}{\Gamma_1;\Gamma_2 \Rightarrow \Delta_1;\Delta_2}\ \text{cut}$$

and of type L if it is of the form

$$\dfrac{\Gamma_1;\Gamma_2 \Rightarrow \Delta_1, C;\Delta_2 \quad \Gamma_1, C;\Gamma_2 \Rightarrow \Delta_1;\Delta_2}{\Gamma_1;\Gamma_2 \Rightarrow \Delta_1;\Delta_2}\ \text{cut}$$

Our cut-elimination argument will work with proofs all of whose cuts are of type *R*.

Definition 23. We say that an axiom is of type L/L if it is of the form $A;\ \Rightarrow A;$, of type L/R if it is of the form $A;\ \Rightarrow\ ;A$, of type R/L if it is of the form $;A \Rightarrow A;$, and of type R/R if it is of the form $;A \Rightarrow\ ;A$.

Definition 24. We say that an axiom occurrence $A \Rightarrow A$ in a proof π is of type Ω if both occurrences of A are ancestors of cut formulas in π.

Every cut c in a proof has a *left subproof* and a *right subproof*: the subproof whose end-sequent is the left, or respectively: right, premise sequent of c.

Definition 25. An $\mathbf{LK}^{\mathrm{m}}$ proof π is called *tame* if

(1) π does not contain axioms of type Ω and
(2) every cut in π has a subproof in which all axioms in which an ancestor of the cut formula is active are of type R/R.

Definition 26. A clause set $\mathcal{A}$ is called *pruned* if no atom occurs both positively and negatively in $\mathcal{A}$ and $\mathcal{A}$ does not contain the literal $\top$.

For instance, none of the following clause sets are pruned:

$$\{\{p\}, \{r, \neg p\}\} \qquad \{\{\top, p\}\} \qquad \{\{p, \neg p\}, \{r\}\}$$

Lemma 27. *Let* $\mathcal{A}$ *be a clause set in some language* L. *Let* $L_{\mathrm{D}} = \{p \in L \mid p \text{ occurs both positively and negatively in } \mathcal{A}\}$. *Then, there is a pruned clause set* $\mathcal{A}^*$ *in the language* $L \setminus L_{\mathrm{D}}$ *such that*

(1) $I \models \mathcal{A}$ *implies* $I \models \mathcal{A}^*$.
(2) $I' \models \mathcal{A}^*$ *implies that there is an extension of* I' *to an* L *interpretation* I *such that* $I \models \mathcal{A}$.

Proof in the appendix. □

The above lemma is shown essentially by computing the closure $\mathcal{A}'$ of $\mathcal{A}$ under resolution and then obtaining $\mathcal{A}^*$ from $\mathcal{A}'$ by deleting all clauses that contain an atom in L_{D}, including all tautologies. Thus, pruned clause sets allow for a simple standardised representation of a formula or a clause set.

The following useful Lemma essentially combines existing interpolants with a conjunction.

Lemma 28. *Let* $A \to B$ *be a valid formula, and let the clause set* $C = \{C_i \mid 1 \leq i \leq n\}$ *be an interpolant of* $A \to B$. *For* $i = 1, \ldots, n$ *let* $\pi_i : A;\ \Rightarrow\ ;C_i$ *be an* $\mathbf{LK}^{\mathrm{at}}$ *proof. Then, there is an* $\mathbf{LK}^{\mathrm{at}}$ *proof* ψ *of* $A;\ \Rightarrow\ ;B$ *all of whose cuts are of type R with* $\mathrm{CNF}(\mathcal{M}(\psi)) = \mathrm{CNF}(\bigwedge_{i=1}^{n} \mathcal{M}(\pi_i))$. *Moreover, if* C *is a pruned clause set, then* ψ *is tame.*

Proof. As $C = \bigwedge_{i=1}^{n} \bigvee_{j=1}^{k_i} \ell_{ij}$ is an interpolant of $A \to B$, both $A \Rightarrow C$ and $C \Rightarrow B$ are valid sequents. Take $\neg C = \neg \bigwedge_{i=1}^{n} \bigvee_{j=1}^{k_i} \ell_{ij}$ which is logically equivalent to $\bigvee_{i=1}^{n} \bigwedge_{j=1}^{k_i} \overline{\ell_{ij}}$ and to the clause set $\mathrm{CNF}(\bigvee_{i=1}^{n} \bigwedge_{j=1}^{k_i} \overline{\ell_{ij}})$. Define

$$\begin{cases} C^+ = \{\{\ell_{i1}, \ldots, \ell_{ik_i}\} \mid 1 \le i \le n\} \\ C^- = \{\{\overline{\ell_{1j_1}}, \ldots, \overline{\ell_{nj_n}}\} \mid 1 \le j_i \le k_i\} \end{cases}$$

The set of clauses $C^+ \cup C^-$ is unsatisfiable. Let F be a resolution refutation with these clauses as the initial clauses and $\emptyset$ as the conclusion of the refutation. Transform F to a split proof in $\mathbf{LK}^{\mathrm{lit}}$ as follows. Take the initial clauses; if a clause is in C^+ then it is of the form $C_i = \{\ell_{i1}, \ldots, \ell_{ik_i}\}$ for some $1 \le i \le n$. And if an initial clause is in C^-, then it is of the form $D_{j_i} = \{\overline{\ell_{1j_1}}, \ldots, \overline{\ell_{nj_n}}\}$ for some $1 \le j_i \le k_i$. Replace these clauses with the following split sequents:

$$C_i \text{ with } A; \Rightarrow; \ell_{i1}, \ldots, \ell_{ik_i} \qquad D_{j_i} \text{ with } ; \ell_{1j_1}, \ldots, \ell_{nj_n} \Rightarrow; B$$

Now, suppose a resolution rule is applied on a literal ℓ in F

$$\frac{M, \ell \qquad \overline{\ell}, N}{M, N}$$

We replace this rule with a cut on the literal ℓ where the premises of the cut rule are the corresponding split sequents of M, ℓ and ℓ, N. After making all these replacements, we obtain a derivation ψ_1 of $A; \Rightarrow; B$, where the split sequents $A; \Rightarrow; \ell_{i1}, \ldots, \ell_{ik_i}$ and $; \ell_{1j_1}, \ldots, \ell_{nj_n} \Rightarrow; B$, for all $1 \le i \le n$ and $1 \le j_i \le k_i$ appear as leaves of ψ_1. Note that in ψ_1 inferences of the following form appear:

$$\frac{\begin{array}{cc} \pi_1 & \pi_2 \\ \Gamma_1; \Gamma_2 \Rightarrow \Delta_1; \ell, \Delta_2 & \Sigma_1; \ell, \Sigma_2 \Rightarrow \Lambda_1; \Lambda_2 \end{array}}{\Gamma_1, \Sigma_1; \Gamma_2, \Sigma_2 \Rightarrow \Delta_1, \Lambda_1; \Delta_2, \Lambda_2}$$

for some literal ℓ. We use these inferences as abbreviations for

$$\dfrac{\dfrac{\begin{array}{c}\pi_1 \\ \Gamma_1; \Gamma_2 \Rightarrow \Delta_1; \ell, \Delta_2\end{array}}{\Gamma_1, \Sigma_1; \Gamma_2, \Sigma_2 \Rightarrow \Delta_1, \Lambda_1; \ell, \Delta_2, \Lambda_2} \qquad \dfrac{\begin{array}{c}\pi_2 \\ \Sigma_1; \ell, \Sigma_2 \Rightarrow \Lambda_1; \Lambda_2\end{array}}{\Gamma_1, \Sigma_1; \ell, \Gamma_2, \Sigma_2 \Rightarrow \Delta_1, \Lambda_1; \Delta_2, \Lambda_2}}{\Gamma_1, \Sigma_1; \Gamma_2, \Sigma_2 \Rightarrow \Delta_1, \Lambda_1; \Delta_2, \Lambda_2}$$

where the double lines mean applying the left and right weakening rules as often as needed. For the curious reader, this means that we can replace each *context-splitting* cut (a cut rule of the first form) with a combination of weakening rules and *context-sharing* cut rules, to get a derivation in $\mathbf{LK}^{\mathrm{lit}}$. Now, replace each leaf of ψ_1 of the form $A; \Rightarrow; \ell_{i1}, \ldots, \ell_{ik_i}$ by the proof π_i. Moreover, replace each leaf of ψ_1 of the form $; \ell_{1j_1}, \ldots, \ell_{nj_n} \Rightarrow; B$ by a cut-free proof $\sigma_{j_1,\ldots,j_n}$ of it which exists since it is a valid sequent. The result of making all these replacements yields an $\mathbf{LK}^{\mathrm{lit}}$ proof $\psi_2 : A; \Rightarrow; B$. Now, we investigate the Maehara interpolant of ψ_2. Note that in each cut rule, the cut formula is on the right-hand side of the Maehara partition. The initial sequents of ψ_2 are either the initial sequents of π_i's or initial sequents of the cut-free proofs of the sequents $; \ell_{1j_1}, \ldots, \ell_{nj_n} \Rightarrow; B$. For the latter, by Observation 18, the interpolant is $\top$. On the other hand, the proof π_i has the interpolant $\mathcal{M}(\pi_i)$. Hence, $\mathcal{M}(\psi_2)$ is the conjunction of $\bigwedge_{i=1}^n \mathcal{M}(\pi_i)$ with several $\top$'s. The final step is applying Lemma 21 to obtain a proof $\psi : A; \Rightarrow; B$ in $\mathbf{LK}^{\mathrm{at}}$. We have

$$\mathcal{M}(\psi_1) = \mathcal{M}(\psi_2) = \mathcal{M}(\psi)$$

and we have $\mathrm{CNF}(\mathcal{M}(\psi)) = \mathrm{CNF}(\bigwedge_{i=1}^n \mathcal{M}(\pi_i))$. All cuts in the proofs ψ_1, ψ_2, and ψ are of type R. Since C is a pruned clause set, no C_i is a tautology, so no π_i contains an axiom of type Ω. Similarly, since C is a pruned clause set, no $\{\overline{\ell_{1,j_1}}, \ldots, \overline{\ell_{n,j_n}}\}$ contains two dual literals, so no $\sigma_{j_1,\ldots,j_n}$ contains an axiom of type Ω. Therefore tameness condition (1) is satisfied. Since all formula occurrences in $\sigma_{j_1,\ldots,j_n}$ are on the right-hand side of the split sequents, all axioms on the right-hand side of a cut on ℓ_{i,j_i} in which an ancestor of the cut formula is active are of type R/R. Therefore tameness condition (2) is satisfied.

□

Remark 29. Up to associativity, commutativity, idempotence, and unit elimination of $\wedge$ we have $\mathcal{M}(\psi) = \bigwedge_{i=1}^n \mathcal{M}(\pi_i)$ in Lemma 28.

5 COMPLETENESS UP TO PRUNING AND SUBSUMPTION

Even though, as we have seen in Proposition 10, Maehara interpolation in $\mathbf{LK}^-$ is incomplete it is possible to obtain also positive results for $\mathbf{LK}^-$. In this section we will prove such a positive result: we will show that if we restrict our attention to *pruned interpolants*, then Maehara interpolation is complete up to subsumption. The proof strategy consists of carrying out a cut-elimination procedure on tame proofs with monochromatic cuts and showing that, in this setting, the interpolants have a very nice behaviour: the interpolant of the reduced proof is subsumed by the interpolant of the original proof. Applying this cut-elimination on a suitable chosen proof with atomic cuts will yield the desired result. This strategy is reminiscent of that used in [3] to compare cut-elimination by resolution (CERES) with traditional methods for cut-elimination.

Definition 30. A clause set $\mathcal{A}$ *subsumes* a clause set $\mathcal{B}$, in symbols $\mathcal{A} \le_{\mathrm{ss}} \mathcal{B}$, if for all $B \in \mathcal{B}$ there is an $A \in \mathcal{A}$ s.t. $A \subseteq B$.

For instance, $\{\{p\}\}$ subsumes $\{\{p, q\}, \{p\}\}$. Subsumption is one of the most useful and one of the most thoroughly studied mechanisms for the detection and elimination of redundancy in automated deduction. Note that, if $\mathcal{A} \le_{\mathrm{ss}} \mathcal{B}$ then $\mathcal{A} \models \mathcal{B}$. In this sense, subsumption is a restricted form of implication.

Observation 31. *Let $\mathcal{A}, \mathcal{B}, C$ be clause sets.*

(1) If $\mathcal{A} \supseteq \mathcal{B}$ then $\mathcal{A} \le_{\mathrm{ss}} \mathcal{B}$.
(2) If $\mathcal{A} \le_{\mathrm{ss}} \mathcal{B}$ and $\mathcal{B} \le_{\mathrm{ss}} C$ then $\mathcal{A} \le_{\mathrm{ss}} C$.
(3) If $\mathcal{A} \le_{\mathrm{ss}} \mathcal{B}$ then $\mathcal{A} \cup C \le_{\mathrm{ss}} \mathcal{B} \cup C$.
(4) If $\mathcal{A} \le_{\mathrm{ss}} \mathcal{B}$ then $\mathcal{A} \times C \le_{\mathrm{ss}} \mathcal{B} \times C$.
(5) $(\mathcal{A} \times \mathcal{B}) \cup C \le_{\mathrm{ss}} (\mathcal{A} \cup C) \times (\mathcal{B} \cup C)$

Proof in the appendix. □

We proceed to set up the initial $\mathbf{LK}^{\mathrm{at}}$ proof for our cut-elimination argument.

Definition 32. A pruned clause set C is called *pruned interpolant* of a formula $A \to B$ if it is an interpolant and there are no $C' \subset C \in C$ with $A \models C'$.

So a pruned interpolant, in addition to being a pruned clause set, must not contain redundant literals in the sense of the above definition.

Lemma 33. *Let C be a pruned interpolant of an implication $A \to B$. Then there is a tame $\mathbf{LK}^{\mathrm{at}}$ proof π of $A; \Rightarrow; B$ all of whose cuts are of type R with $\mathrm{CNF}(\mathcal{M}(\pi)) = C$.*

Proof. Let $C = \{C_i \mid 1 \le i \le n\}$. Since C is an interpolant, $\models A \to C_i$ for all $i = 1, \ldots, n$. Since C is pruned, the C_i are non-tautological, so Lemma 19 yields an $\mathbf{LK}^-$ proof π_i of $A; \Rightarrow; C_i$ with

$\mathrm{CNF}(\mathcal{M}(\pi_i)) = \{M_i\}$ for some $M_i \subseteq C_i$. Since $\mathcal{M}(\pi_i)$ is an interpolant of $A \to C_i$, we have $\models A \to \mathcal{M}(\pi_i)$. By Observation 17/(1), we have $\models A \to \mathrm{CNF}(\mathcal{M}(\pi_i))$, i.e., $\models A \to M_i$. Then $M_i = C_i$ because $M_i \subset C_i$ would contradict prunedness of C. Then, by applying Lemma 28, we obtain a tame $\mathbf{LK}^{\mathrm{at}}$ proof π all of whose cuts are of type R with $\mathrm{CNF}(\mathcal{M}(\pi)) = \mathrm{CNF}(\bigwedge_{i=1}^{n} \mathcal{M}(\pi_i)) = C$. □

Definition 34. We call a proof π *w-reduced* if every weakening inference in π occurs immediately below an axiom or another weakening inference.

The point of the notion of w-reduced proofs is to facilitate the technical matters of the cut-elimination argument in our variant of **LK**. We now proceed to set up the termination measure for the cut-elimination procedure.

Definition 35. A formula occurrence in a proof π is called *weak* if all its ancestors are introduced by weakening inferences.

Let μ be an occurrence of a formula A in a proof π. An occurrence μ' of A is called *relevant* for μ if μ' is an ancestor of μ, μ' is not weak, and μ' is not in the conclusion sequent of a weakening inference.

The *weight of a formula occurrence* μ in a proof π, written as $\mathrm{w}(\mu)$, is the number of formula occurrences which are relevant for μ.

The *weight of a cut* c in a proof π is defined as $\mathrm{w}(c) = \mathrm{w}(\mu_\mathrm{l}) + \mathrm{w}(\mu_\mathrm{r})$ where μ_l (resp. μ_r) is the occurrence of the cut formula in the left (resp. right) premise of c.

Definition 36. The *degree of a cut* c, written as $\deg(c)$, is the logical complexity, i.e., the number of propositional connectives, of the cut formula of c.

Lemma 37. *Let $G \in \{\mathbf{LK}^{\mathrm{m}}, \mathbf{LK}^{-}\}$. For every G proof π of a sequent $\Gamma_1;\Gamma_2 \Rightarrow \Delta_1;\Delta_2$ there is a w-reduced G proof π' of $\Gamma_1;\Gamma_2 \Rightarrow \Delta_1;\Delta_2$ such that*

(1) $\mathcal{M}(\pi') = \mathcal{M}(\pi)$,
(2) if π is tame then so is π'
(3) if all cuts in π are of type R then all cuts in π' are of type R
(4) for any formula occurrence μ in the end-sequent of π and its corresponding formula occurrence μ' in the end-sequent of π':
  (a) $\mathrm{w}(\mu') = \mathrm{w}(\mu)$ and
  (b) every axiom of π' in which an ancestor of μ' is active is, up to weak formulas, also an axiom of π in which an ancestor of μ is active.

Proof Sketch. Shifting up weakenings until they are in a permitted position satisfies the mentioned properties. For a more detailed proof, please see the appendix. □

Lemma 38. *For every tame $\mathbf{LK}^{\mathrm{m}}$ proof π all of whose cuts are of type R, there is an $\mathbf{LK}^{-}$ proof π' with $\mathrm{CNF}(\mathcal{M}(\pi)) \leq_{\mathrm{ss}} \mathrm{CNF}(\mathcal{M}(\pi'))$.*

Proof. By Lemma 37 we can assume that π is tame and w-reduced. In this proof we will write

$$\frac{}{\Gamma, A \Rightarrow \Delta, A}\ \mathrm{wax}$$

as an abbreviation for the axiom $A \Rightarrow A$ followed by weakening inferences to derive $\Gamma, A \Rightarrow \Delta, A$.

We write π as $\pi[\chi]$ where χ is a subproof of π that ends with an uppermost cut. Based on a case distinction on the form of χ we will define a proof χ^* which, by replacing χ, yields a proof $\pi[\chi^*]$. We will show that

(i) $\pi[\chi^*]$ is tame and w-reduced,
(ii) the cut c in χ is replaced by cuts c' in χ^* with $\mathrm{d}(c) > \mathrm{d}(c')$ or ($\mathrm{d}(c) = \mathrm{d}(c')$ and $\mathrm{w}(c) > \mathrm{w}(c')$), and
(iii) $\mathcal{M}(\chi) \leq_{\mathrm{ss}} \mathcal{M}(\chi^*)$.

Points (i) and (ii) ensure correctness and termination of the cut-elimination sequence while point (iii) shows $\mathrm{CNF}(\mathcal{M}(\pi[\chi])) \leq_{\mathrm{ss}} \mathrm{CNF}(\mathcal{M}(\pi[\chi^*]))$ by Observation 31/(3) and (4). This suffices to prove the result by transitivity of $\leq_{\mathrm{ss}}$. In order to show that $\pi[\chi^*]$ is tame we will show the following condition (*) in most cases:

Every axiom of χ^* is, up to weak formulas, an axiom of χ.

Hence tameness condition (1) is preserved, because χ is tame. Moreover, if μ is a formula occurrence in the end-sequent of χ and μ^* is the corresponding formula occurrence in the end-sequent of χ^*, then every axiom of χ^* in which an ancestor of μ^* is active is, up to weak formulas, also an axiom of χ in which μ is active. Therefore, also tameness condition (2) is preserved.

Exclusion of weak cut formulas: If $\chi =$

$$\frac{\begin{array}{cc}(\chi_1) & (\chi_2)\\ \Gamma_1;\Gamma_2 \Rightarrow \Delta_1;\Delta_2, C_{\mu_1} & \Gamma_1;C_{\mu_2},\Gamma_2 \Rightarrow \Delta_1;\Delta_2\end{array}}{\Gamma_1;\Gamma_2 \Rightarrow \Delta_1;\Delta_2}\ \mathrm{cut}$$

where μ_1 is weak, define $\chi^* : \Gamma_1;\Gamma_2 \Rightarrow \Delta_1;\Delta_2$ from χ_1 by removing μ_1, all its ancestors, and all weakening inferences that introduce these ancestors. Then $\pi[\chi^*]$ is w-reduced and, since χ_1, χ^* satisfy (*), $\pi[\chi^*]$ is also tame which shows (i). (ii) holds vacuously. Furthermore, $\mathrm{CNF}(\mathcal{M}(\chi)) = \mathrm{CNF}(\mathcal{M}(\chi_1) \wedge \mathcal{M}(\chi_2)) \supseteq \mathrm{CNF}(\mathcal{M}(\chi_1)) = \mathrm{CNF}(\mathcal{M}(\chi^*))$. So (iii) follows from Observation 31/(1). In case μ_2 is weak we proceed analogously. So, in the remaining cases, we assume that none of the two cut formulas are weak.

Permutation of a binary inference over a cut: If χ is of the form

$$\frac{\dfrac{\begin{array}{cc}(\chi_1) & (\chi_2)\\ \Gamma_1;\Gamma_2 \Rightarrow \Delta_1, A;\Delta_2, C_{\mu_1} & \Gamma_1;\Gamma_2 \Rightarrow \Delta_1, B;\Delta_2, C_{\mu_2}\end{array}}{\Gamma_1;\Gamma_2 \Rightarrow \Delta_1, A \wedge B;\Delta_2, C}\ \begin{array}{c}(\chi_3)\\ \Gamma_1;\Gamma_2, C_{\mu_3} \Rightarrow \Delta_1, A \wedge B;\Delta_2\end{array}}{\Gamma_1;\Gamma_2 \Rightarrow \Delta_1, A \wedge B;\Delta_2}\ \mathrm{cut}_c$$

we define the proofs

$$\chi_1' = \frac{\begin{array}{c}(\chi_1)\\ \Gamma_1;\Gamma_2 \Rightarrow \Delta_1, A;\Delta_2, C\end{array}}{\Gamma_1;\Gamma_2 \Rightarrow \Delta_1, A, A \wedge B;\Delta_2, C}$$

$$\chi_2' = \frac{\begin{array}{c}(\chi_2)\\ \Gamma_1;\Gamma_2 \Rightarrow \Delta_1, B;\Delta_2, C\end{array}}{\Gamma_1;\Gamma_2 \Rightarrow \Delta_1, B, A \wedge B;\Delta_2, C}$$

$$\chi_{3,1}' = \frac{\begin{array}{c}(\chi_3)\\ \Gamma_1;\Gamma_2, C \Rightarrow \Delta_1, A \wedge B;\Delta_2\end{array}}{\Gamma_1;\Gamma_2, C \Rightarrow \Delta_1, A, A \wedge B;\Delta_2}$$

$$\chi_{3,2}' = \frac{\begin{array}{c}(\chi_3)\\ \Gamma_1;\Gamma_2, C \Rightarrow \Delta_1, A \wedge B;\Delta_2\end{array}}{\Gamma_1;\Gamma_2, C \Rightarrow \Delta_1, B, A \wedge B;\Delta_2}$$

By applying Lemma 37 to these proofs individually, we obtain proofs χ_1^*, χ_2^*, $\chi_{3,1}^*$, and $\chi_{3,2}^*$ that satisfy (1) and (4). We finally define $\chi^* =$

$$\dfrac{\dfrac{(\chi_1^*) \qquad (\chi_{3,1}^*)}{\Gamma_1;\Gamma_2 \Rightarrow \Delta_1, A, A \wedge B; \Delta_2}\,\mathrm{cut}_{c_1} \qquad \dfrac{(\chi_2^*) \qquad (\chi_{3,2}^*)}{\Gamma_1;\Gamma_2 \Rightarrow \Delta_1, B, A \wedge B; \Delta_2}\,\mathrm{cut}_{c_2}}{\dfrac{\Gamma_1;\Gamma_2 \Rightarrow \Delta_1, A \wedge B, A \wedge B; \Delta_2}{\Gamma_1;\Gamma_2 \Rightarrow \Delta_1, A \wedge B; \Delta_2}}\,(R\wedge)$$

For (i) we observe that $\pi[\chi^*]$ is w-reduced and, since χ, χ^* satisfy (*), $\pi[\chi^*]$ is tame. For (ii) we observe that $\mathrm{w}(c) = \mathrm{w}(\mu_1) + w(\mu_2) + 1 + \mathrm{w}(\mu_3)$, $\mathrm{w}(c_1) = \mathrm{w}(\mu_1) + \mathrm{w}(\mu_3)$, and $\mathrm{w}(c_2) = \mathrm{w}(\mu_2) + \mathrm{w}(\mu_3)$. For (iii) we have $\mathrm{CNF}(\mathcal{M}(\chi)) = \mathrm{CNF}((\mathcal{M}(\chi_1) \vee \mathcal{M}(\chi_2)) \wedge \mathcal{M}(\chi_3)) \leq_{\mathrm{ss}}^{\text{Obs. 31/(5)}} \mathrm{CNF}((\mathcal{M}(\chi_1) \wedge \mathcal{M}(\chi_3)) \vee (\mathcal{M}(\chi_2) \wedge \mathcal{M}(\chi_3))) = \mathrm{CNF}(\mathcal{M}(\chi^*))$.

We proceed analogously for the other binary inferences. If the binary inference above the cut works on the right-hand side of the Maehara partition, we define χ^* analogously and obtain the calculation $\mathrm{CNF}(\mathcal{M}(\chi)) = \mathrm{CNF}((\mathcal{M}(\chi_1) \wedge \mathcal{M}(\chi_2)) \wedge \mathcal{M}(\chi_3)) =^{\text{Obs. 17}} \mathrm{CNF}((\mathcal{M}(\chi_1) \wedge \mathcal{M}(\chi_3)) \wedge (\mathcal{M}(\chi_2) \wedge \mathcal{M}(\chi_3))) = \mathrm{CNF}(\mathcal{M}(\chi^*))$.

Reduction of axioms: Since all cuts are of type R, we have to consider the four cases of axioms of type T_1/R and R/T_2 for $T_1, T_2 \in \{\mathrm{L}, \mathrm{R}\}$.

If χ is of the form R/R and R/R as in

$$\dfrac{\dfrac{}{\Gamma_1;\Gamma_2, A \Rightarrow \Delta_1;\Delta_2, A, A}\,\mathrm{wax} \qquad \dfrac{}{\Gamma_1;\Gamma_2, A, A \Rightarrow \Delta_1;\Delta_2, A}\,\mathrm{wax}}{\Gamma_1;\Gamma_2, A_{\mu_1} \Rightarrow \Delta_1;\Delta_2, A_{\mu_2}}\,\mathrm{cut}$$

we reduce χ to $\chi^* =$

$$\dfrac{}{\Gamma_1;\Gamma_2, A \Rightarrow \Delta_1;\Delta_2, A}\,\mathrm{wax}$$

For (i) we observe that $\pi[\chi^*]$ is w-reduced. Moreover, the axiom in χ^* is not of type Ω for if it were, then both μ_1 and μ_2 would be ancestors of cuts in π and hence χ would contain two axioms of type Ω. (ii) is trivial. For (iii) we have $\mathrm{CNF}(\mathcal{M}(\chi)) = \mathrm{CNF}(\top \wedge \top) =^{\text{Obs. 17/(6)}} \mathrm{CNF}(\top) = \mathrm{CNF}(\mathcal{M}(\chi^*))$.

If χ is of the form R/R and R/L as in

$$\dfrac{\dfrac{}{\Gamma_1;\Gamma_2, A \Rightarrow \Delta_1, A;\Delta_2, A}\,\mathrm{wax} \qquad \dfrac{}{\Gamma_1;\Gamma_2, A, A \Rightarrow \Delta_1, A;\Delta_2}\,\mathrm{wax}}{\Gamma_1;\Gamma_2, A \Rightarrow \Delta_1, A;\Delta_2}\,\mathrm{cut}$$

we reduce χ to $\chi^* =$

$$\dfrac{}{\Gamma_1;\Gamma_2, A \Rightarrow \Delta_1, A;\Delta_2}\,\mathrm{wax}$$

(i) is shown as above. (ii) is trivial. For (iii) we have $\mathrm{CNF}(\mathcal{M}(\chi)) = \mathrm{CNF}(\top \wedge \neg A) =^{\text{Obs. 17/(6)}} \mathrm{CNF}(\neg A) = \mathrm{CNF}(\mathcal{M}(\chi^*))$.

If χ is of the form L/R and R/R as in

$$\dfrac{\dfrac{}{\Gamma_1, A;\Gamma_2 \Rightarrow \Delta_1;\Delta_2, A, A}\,\mathrm{wax} \qquad \dfrac{}{\Gamma_1, A;\Gamma_2, A \Rightarrow \Delta_1;\Delta_2, A}\,\mathrm{wax}}{\Gamma_1, A;\Gamma_2 \Rightarrow \Delta_1;\Delta_2, A}\,\mathrm{cut}$$

we reduce χ to $\chi^* =$

$$\dfrac{}{\Gamma_1, A;\Gamma_2 \Rightarrow \Delta_1;\Delta_2, A}\,\mathrm{wax}$$

(i) is shown as above. (ii) is trivial. For (iii) we have $\mathrm{CNF}(\mathcal{M}(\chi)) = \mathrm{CNF}(A \wedge \top) =^{\text{Obs. 17/(6)}} \mathrm{CNF}(A) = \mathrm{CNF}(\mathcal{M}(\chi^*))$.

The case of χ being of the form L/R and R/L as in

$$\dfrac{\dfrac{}{\Gamma_1, A;\Gamma_2 \Rightarrow \Delta_1, A;\Delta_2, A}\,\mathrm{wax} \qquad \dfrac{}{\Gamma_1, A;\Gamma_2, A \Rightarrow \Delta_1, A;\Delta_2}\,\mathrm{wax}}{\Gamma_1, A;\Gamma_2 \Rightarrow \Delta_1, A;\Delta_2}\,\mathrm{cut}$$

is ruled out by χ being tame.

The remaining cases of this cut-elimination argument can be found in the appendix. □

Theorem 39. *Let C be a pruned interpolant of an implication $A \to B$. Then there is an $\mathbf{LK}^-$ proof π of $A; \Rightarrow ; B$ with $C \leq_{\mathrm{ss}} \mathrm{CNF}(\mathcal{M}(\pi))$.*

Proof. By Lemma 33 there is a tame $\mathbf{LK}^{\mathrm{at}}$ proof π of $A; \Rightarrow ; B$ all of whose cuts are of type R with $\mathrm{CNF}(\mathcal{M}(\pi)) = C$. Then, by applying Lemma 38, we obtain an $\mathbf{LK}^-$ proof π' with $C = \mathrm{CNF}(\mathcal{M}(\pi)) \leq_{\mathrm{ss}} \mathrm{CNF}(\mathcal{M}(\pi'))$. □

So even though interpolation in $\mathbf{LK}^-$ is not complete as shown in Proposition 10, we can still recover a desired interpolant I in a restricted sense: after transforming I into a pruned interpolant C we can obtain a proof whose interpolant is subsumed by C.

Example 40. The formula $p \wedge q \to p \vee q$ has the four interpolants $p \wedge q, p, q, p \vee q$. We know from the proof of Proposition 10 that the only interpolants obtainable from $\mathbf{LK}^-$ proofs are p and q. The clause set $\{\{p, q\}\}$, representing the formula $p \vee q$, is not a pruned interpolant. The clause set $\{\{p\}, \{q\}\}$, representing the formula $p \wedge q$, subsumes both $\{\{p\}\}$ and $\{\{q\}\}$.

Question 41. *Is standard interpolation in resolution complete up to subsumption for pruned interpolants?*

Question 42. *Can we extend these results to the sequent calculus* **LJ** *for the intuitionistic logic? How about other super intuitionistic or substructural logics?*

6 SEQUENT CALCULUS WITH ATOMIC CUTS

If we move from $\mathbf{LK}^-$ to the slightly stronger $\mathbf{LK}^{\mathrm{at}}$ we can even obtain a full completeness result. This is achieved by a variant of the construction used in the proof of Lemma 33.

Theorem 43. *Maehara interpolation in $\mathbf{LK}^{\mathrm{at}}$ is complete.*

Proof. Let I be an interpolant of an implication $A \to B$ and let the clause set $C = \{C_1, \ldots, C_n\}$ be logically equivalent to I. For $i = 1, \ldots, n$ let $C_i = \{\ell_{i,1}, \ldots, \ell_{i,k_i}\}$. We start by constructing proofs $\pi_i : A; \Rightarrow ; \ell_{i,1}, \ldots, \ell_{i,k_i}$ such that $\mathcal{M}(\pi_i)$ is logically equivalent to C_i. As C is an interpolant of $A \to B$, we have $\mathbf{LK} \vdash A \Rightarrow \bigwedge_{i=1}^n C_i$. Thus, for each $1 \leq i \leq n$, $\mathbf{LK} \vdash A \Rightarrow C_i$ and hence $\mathbf{LK} \vdash A \Rightarrow \ell_{i1}, \ldots, \ell_{ik_i}$. Let α_i be a cut-free proof of

$$\alpha_i : \qquad A; \Rightarrow \ell_{i1}, \ldots, \ell_{ik_i};$$

By Observation 18, $\mathcal{M}(\alpha_i)$ is logically equivalent to $\bot$. Take the following proof tree as π_i, which moves all the literals in the succedent of α_i to the right-hand side of the Maehara partition only by cuts on literals and weakening rules:

$$\dfrac{\dfrac{\dfrac{\dfrac{\alpha_i}{A; \Rightarrow \ell_{i1}, \ell_{i2}, \ldots, \ell_{ik_i};} \qquad \dfrac{}{\ell_{i1}; \Rightarrow ; \ell_{i1}}}{A; \Rightarrow \ell_{i1}, \ell_{i2}, \ldots, \ell_{ik_i}; \ell_{i1}} \qquad \dfrac{}{A, \ell_{i1}; \Rightarrow \ell_{i2}, \ldots, \ell_{ik_i}; \ell_{i1}}}{A; \Rightarrow \ell_{i2}, \ldots, \ell_{ik_i}; \ell_{i1}}}{\vdots}}{\dfrac{\dfrac{A; \Rightarrow \ell_{ik_i}; \ell_{i1}, \ldots, \ell_{ik_i-1}}{A; \Rightarrow \ell_{ik_i}; \ell_{i1}, \ldots, \ell_{ik_i-1}, \ell_{ik_i}} \qquad \dfrac{}{\ell_{ik_i}; \Rightarrow ; \ell_{ik_i}}}{A, \ell_{ik_i}; \Rightarrow ; \ell_{i1}, \ldots \ell_{ik_i}}}{A; \Rightarrow ; \ell_{i1}, \ldots \ell_{ik_i}}}$$

The double lines in the proof mean using the rules (Lw) and (Rw) as often as needed. To construct the proof π_i, we start with α_i and use the rule (Rw) on its end-sequent. Then, we take the valid sequent $\ell_{i1}; \Rightarrow; \ell_{i1}$ and apply (Lw) and (Rw) as needed. Then we can use the cut rule to move ℓ_{i1} to the right-hand side of the Maehara partition. We repeat this procedure for the rest of the literals to get π_i. To see that $\mathcal{M}(\pi_i)$ is logically equivalent to C_i, note that each cut in the proof π_i is on a literal and the cut formula is always on the left-hand side of the Maehara partition. Moreover, $V(C_i) \subseteq V(C) \subseteq V(A) \cap V(B) \subseteq V(A)$. Hence, in each cut, the cut formula is in $V(A)$, and by the Maehara interpolation algorithm for $\mathbf{LK}^{\mathrm{lit}}$, the interpolant of the conclusion of the cut rule is the disjunction of the interpolants of the premises. Since $\mathcal{M}(\alpha_1)$ is logically equivalent to $\bot$ and $\mathcal{M}(\ell_{ij}; \Rightarrow; \ell_{ij}) = \ell_{ij}$ for $1 \leq j \leq k_i$, we get $\mathcal{M}(\pi_i)$ is logically equivalent to $\bot \vee \ell_{i1} \vee \cdots \vee \ell_{ik_i}$, which is logically equivalent to $\ell_{i1} \vee \cdots \vee \ell_{ik_i} = \bigvee_{j=1}^{k_i} \ell_{ij} = C_i$. Now apply Lemmas 21 and 28 to obtain an $\mathbf{LK}^{\mathrm{at}}$ proof π of $A; \Rightarrow; B$ with $\mathcal{M}(\pi)$ logically equivalent to C. □

A trivial consequence of Theorem 43 is that Maehara interpolation in $\mathbf{LK}^{\mathrm{lit}}$ and $\mathbf{LK}^{\mathrm{m}}$ is also complete.

Remark 44. It is worth noting that the proof of Theorem 43 provides an interpolant logically equivalent to the given interpolant I in CNF, only by adding $\top$ as conjuncts and $\bot$ as disjuncts. Thus, we have even shown that Maehara interpolation is syntactically complete up to unit elimination of conjunction and disjunction for formulas in CNF.

Example 45. We find a proof $\pi : p \wedge q; \Rightarrow; p \vee q$ in $\mathbf{LK}^{\mathrm{at}}$ such that $\mathcal{M}(\pi) = p \wedge q$. Denote $I_1 = p$ and $I_2 = q$. We have

$$\pi_1 : \ \frac{p; \Rightarrow; p}{p \wedge q; \Rightarrow; p} \qquad \pi_2 : \ \frac{q; \Rightarrow; q}{p \wedge q; \Rightarrow; q}$$

and $\mathcal{M}(\pi_1) = p$ and $\mathcal{M}(\pi_2) = q$. Take the following proof tree $\sigma : p \wedge q; \Rightarrow; p \vee q$ in $\mathbf{LK}^{\mathrm{at}}$ where all the cuts are context-splitting:

$$\dfrac{\dfrac{\pi_2}{p \wedge q; \overset{q}{\Longrightarrow}; q} \qquad \dfrac{\dfrac{\pi_1}{p \wedge q; \overset{p}{\Longrightarrow}; p} \qquad \dfrac{\dfrac{}{; p \overset{\top}{\Longrightarrow}; p}}{; p, q \overset{\top}{\Longrightarrow}; p}}{; p, q \overset{\top}{\Longrightarrow}; p \vee q}}{p \wedge q; q \overset{p \wedge \top}{\Longrightarrow}; p \vee q}}{p \wedge q, p \wedge q; \overset{q \wedge p \wedge \top}{\Longrightarrow}; p \vee q}}{p \wedge q; \overset{q \wedge p \wedge \top}{\Longrightarrow}; p \vee q}$$

where the double lines mean applying the left and right weakening rules (to get the same contexts in the premises of the cut rule) and then the cut rule. We have $\mathcal{M}(\sigma)$ is logically equivalent to $p \wedge q$.

7 PROPOSITIONAL NORMAL MODAL LOGICS

We work with the language $\mathcal{L}_\Box = \{\bot, \wedge, \vee, \neg, \Box\}$. The modal rules we consider are:

$$\frac{\Gamma \Rightarrow A}{\Box\Gamma \Rightarrow \Box A}\ (K) \qquad \frac{\Gamma \Rightarrow}{\Box\Gamma \Rightarrow}\ (D) \qquad \frac{\Gamma, \Box\Gamma \Rightarrow A}{\Box\Gamma \Rightarrow \Box A}\ (4) \qquad \frac{\Gamma, A \Rightarrow \Delta}{\Gamma, \Box A \Rightarrow \Delta}\ (T)$$

Consider the normal modal logics K, D, T, K4, KD4, and S4. Take the usual sequent calculi for these logics by adding the corresponding modal rules to **LK**:

$$\mathbf{K} = \mathbf{LK} + (K) \quad \mathbf{KD} = \mathbf{LK} + (K) + (D) \quad \mathbf{KT} = \mathbf{LK} + (K) + (T)$$
$$\mathbf{K4} = \mathbf{LK} + (4) \quad \mathbf{KD4} = \mathbf{LK} + (4) + (D) \quad \mathbf{S4} = \mathbf{LK} + (T) + (4)$$

We extend the interpolation algorithm with the following rules:

$$\Box A; \overset{\bot}{\Longrightarrow} \Box A; \quad ; \Box A \overset{\top}{\Longrightarrow}; \Box A$$
$$\Box A; \overset{\Box A}{\Longrightarrow}; \Box A \quad ; \Box A \overset{\neg\Box A}{\Longrightarrow} \Box A;$$

If the last rule is (K):

$$\frac{\Gamma_1; \Gamma_2 \overset{C}{\Longrightarrow}; A}{\Box\Gamma_1; \Box\Gamma_2 \overset{\Box C}{\Longrightarrow}; \Box A} \qquad \frac{\Gamma_1; \Gamma_2 \overset{C}{\Longrightarrow} A;}{\Box\Gamma_1; \Box\Gamma_2 \overset{\neg\Box\neg C}{\Longrightarrow} \Box A;}$$

If the last rule is (D), then note that the rule (K) is also present in the calculus. Then:

$$\frac{\Gamma_1; \Gamma_2 \overset{C}{\Longrightarrow}}{\Box\Gamma_1; \Box\Gamma_2 \overset{\Box C}{\Longrightarrow}}$$

If the last rule is (4):

$$\frac{\Gamma_1, \Box\Gamma_1; \Gamma_2, \Box\Gamma_2 \overset{C}{\Longrightarrow}; A}{\Box\Gamma_1; \Box\Gamma_2 \overset{\Box C}{\Longrightarrow}; \Box A} \qquad \frac{\Gamma_1, \Box\Gamma_1; \Gamma_2, \Box\Gamma_2 \overset{C}{\Longrightarrow} A;}{\Box\Gamma_1; \Box\Gamma_2 \overset{\neg\Box\neg C}{\Longrightarrow} \Box A;}$$

If the last rule is (T):

$$\frac{\Gamma_1, A; \Gamma_2 \overset{C}{\Longrightarrow} \Delta_1; \Delta_2}{\Gamma_1, \Box A; \Gamma_2 \overset{C}{\Longrightarrow} \Delta_1; \Delta_2} \qquad \frac{\Gamma_1; A, \Gamma_2 \overset{C}{\Longrightarrow} \Delta_1; \Delta_2}{\Gamma_1; \Box A, \Gamma_2 \overset{C}{\Longrightarrow} \Delta_1; \Delta_2}$$

Recall the definition of a monochromatic proof from Subsection 2.3. It can be easily extended to the modal language and modal sequent calculi. Then, for any $X \in \{\mathbf{K}, \mathbf{KD}, \mathbf{KT}, \mathbf{K4}, \mathbf{KD4}, \mathbf{S4}\}$, we define X^m as the set of all monochromatic proofs in X. Define *modal literals*, denoted by $\ell_\Box$, as p, $\neg p$, $\Box A$, $\neg\Box A$, where p is a propositional atom and A is a modal formula. The *conjunctive normal form* of a modal formula, denoted by mCNF, is defined similarly to the propositional case: conjunctions of disjunctions of modal literals. A modal formula of the form $\Box B$ is an *immediate modal subformula* of a modal formula A if it is a subformula of A and there is no other subformula of A that contains $\Box B$. By an easy observation, we see that every modal formula has a logically equivalent modal conjunctive normal form. The reason is that we can see immediate modal subformulas of a formula as new atomic formulas (i.e., not occurring in A) and then this formula will be propositional and it has a CNF. Transforming the new atoms back to the immediate modal subformulas will provide an mCNF for A.

Proposition 46. *Maehara interpolation in cut-free propositional* **K** *is not complete.*

Proof. The formula $\Box(p \wedge q) \rightarrow \Box(p \vee q)$, which is valid in propositional **K**, has the interpolants $\Box(p \wedge q)$, $\Box(p \vee q)$, and $\Box p \wedge \Box q$ but neither of them can be read off of a cut-free proof. □

Theorem 47. *Let $G \in \{\mathbf{K}^m, \mathbf{KD}^m, \mathbf{KT}^m, \mathbf{K4}^m, \mathbf{KD4}^m, \mathbf{S4}^m\}$. Maehara interpolation $\mathcal{M}$ in G where cuts are only allowed on atomic formulas and boxed formulas is complete.*

Proof. Similar to the proof of Theorem 43. Suppose I is an interpolant of the G-provable split sequent $A; \Rightarrow; B$. Take the mCNF of interpolant $I = \bigwedge_{i=1}^{n} I_i = \bigwedge_{i=1}^{n} \bigvee_{j=1}^{k_i} \ell_{ij}$, where each ℓ_{ij} is a modal literal. We can construct the proofs π_i as in Theorem 43 and every step of the proof is similar to before. Note that the cut rule is always monochromatic. □

It is interesting to investigate these questions for non-normal modal logics, and other logics such as the Gödel-löb logic GL.

8 FIRST-ORDER LOGIC

We now move to the completeness of interpolation algorithms in first-order logic. There are different strategies for interpolation in first-order logic (with and without equality). The most prominent strategy consists of 1. computing a propositional interpolant and 2. introducing quantifiers into this propositional interpolant in order to convert it to a first-order interpolant. Such interpolation algorithms can be found, e.g., in [19, Theorem 13], [12], [10, Section 5.13], and [4, Section 8.2]. An alternative strategy consists of 1. replacing function symbols by relation symbols in the input formula, 2. applying an (almost) propositional interpolation algorithm, and 3. translating the interpolant thus obtained back into a language with function symbols. Such algorithms can be found, e.g., in [8] and [2, Section 7.3].

What kind of results can we expect for these two strategies? We can clearly obtain incompleteness results for the cut-free sequent calculus for first-order logic as a straightforward extension of Proposition 10. However, when we consider sequent calculus with atomic cuts, the answer is not clear at first sight and boils down to the question of whether there is some source of incompleteness on the first-order level.

In this section, we will prove a strong incompleteness result for the first strategy: we will show that it is incomplete, regardless of the concrete algorithms employed in the two phases and the concrete calculus used for obtaining the interpolant in the first phase. In order to do this we first have to make this strategy precise. To that aim, we consider the language $\mathcal{L} = \{\bot, \wedge, \vee, \neg, \exists, \forall\}$ and write $L_P(A)$ for the set of all predicate symbols occurring in the first-order formula A. If we add the following rules to propositional **LK**, we get first-order **LK**:

$$\frac{A[x/t], \Gamma \Rightarrow \Delta}{\forall x A, \Gamma \Rightarrow \Delta}\ (L\forall) \qquad \frac{\Gamma \Rightarrow \Delta, A[x/y]}{\Gamma \Rightarrow \Delta, \forall x A}\ (R\forall)$$

$$\frac{A[x/y], \Gamma \Rightarrow \Delta}{\exists x A, \Gamma \Rightarrow \Delta}\ (L\exists) \qquad \frac{\Gamma \Rightarrow \Delta, A[x/t]}{\Gamma \Rightarrow \Delta, \exists x A}\ (R\exists)$$

where t is an arbitrary term and in $(R\forall)$ and $(L\exists)$, the variable y is not free in the conclusion.

Definition 48. Let $A \to B$ be a valid first-order formula. A formula C is called a *weak interpolant* of $A \to B$ if $\models A \to C$, $\models C \to B$, and $L_P(C) \subseteq L_P(A) \cap L_P(B)$.

So, while a weak interpolant satisfies the usual language condition on the predicate symbols, it may contain constants and function symbols, which are not in the intersection of the languages of A and B.

Example 49. Define the formulas

$$A = \forall v\,(v < f(v)) \wedge \forall v \forall w\,(Z(v) \wedge v < w \to \neg Z(w))$$
$$B = Z(c) \to \exists u \exists v\,(Z(u) \wedge \neg Z(v))$$

Then $L(A) = \{Z, <, f\}$, $L(B) = \{Z, c\}$, thus $L(A) \cap L(B) = \{Z\}$. Then $C_0 = Z(c) \to \neg Z(f(c))$ is a weak interpolant.

An algorithm for the computation of interpolants in propositional logic can usually be easily adapted to compute weak interpolants in first-order logic. For example, we can add the rules

$$\frac{A[x/y], \Gamma_1; \Gamma_2 \overset{C}{\Rightarrow} \Delta_1; \Delta_2}{\exists x A(x), \Gamma_1; \Gamma_2 \overset{C}{\Rightarrow} \Delta_1; \Delta_2} \qquad \frac{\Gamma_1; A[x/y], \Gamma_2 \overset{C}{\Rightarrow} \Delta_1; \Delta_2}{\Gamma_1; \exists x A(x), \Gamma_2 \overset{C}{\Rightarrow} \Delta_1; \Delta_2}$$

$$\frac{\Gamma_1; \Gamma_2 \overset{C}{\Rightarrow} \Delta_1, A[x/t]; \Delta_2}{\Gamma_1; \Gamma_2 \overset{C}{\Rightarrow} \Delta_1, \exists x A(x); \Delta_2} \qquad \frac{\Gamma_1; \Gamma_2 \overset{C}{\Rightarrow} \Delta_1; \Delta_2, A[x/t]}{\Gamma_1; \Gamma_2 \overset{C}{\Rightarrow} \Delta_1; \Delta_2, \exists x A(x)}$$

and analogous rules for $\forall$ to the algorithm from Section 2.3 in order to obtain an algorithm that computes a weak interpolant of $A \to B$ from an $\mathbf{LK}^{\mathrm{m}}$ proof of $A;\ \Rightarrow\ ; B$. We can now make the first phase of the strategy precise: 1. we compute a weak interpolant C_0 for $A \to B$, e.g., as in the algorithm described above. To describe the second phase, we define:

Definition 50. We define the binary relation *A is an abstraction of B* on first-order formulas as the smallest reflexive and transitive relation that satisfies the following condition: $A = Qx\,A_0$ for $Q \in \{\forall, \exists\}$ and $B = A_0[x \backslash t]$ for some term t.

Lemma 51. *There is an algorithm $\mathcal{B}$ which, given a weak interpolant C_0 of $A \to B$ computes an interpolant C of $A \to B$ which is an abstraction of C_0.*

Proof Sketch. $\mathcal{B}$ replaces a maximal term t which is not in $L(A) \cap L(B)$ by a new bound variable x which is either existentially or universally quantified, depending on whether the leading function symbol of t is in $L(A) \setminus L(B)$ or $L(B) \setminus L(A)$. If $C[t]$ is a weak interpolant of $A \to B$, then $Qx\,C[x]$ is an abstraction of $C[t]$ and a weak interpolant of $A \to B$ which contains one less term violating the language condition. By repeating this step one obtains an interpolant in the usual sense. See, e.g., [4, Lemma 8.2.2] for a detailed exposition of this proof. □

We can now make the second phase precise: 2. we apply the algorithm $\mathcal{B}$ to the weak interpolant C_0 of $A \to B$ in order to obtain an interpolant $C := \mathcal{B}(C_0)$ of $A \to B$.

Example 52. Continuing Example 49 we obtain the interpolant $\mathcal{B}(C_0) = \forall x \exists y\,(Z(x) \to \neg Z(y))$ of $A \to B$.

The central observation is now the following: in first-order logic, there are interpolants which are not abstractions of weak interpolants. More precisely:

Lemma 53. *There are first-order sentences A, B, and C s.t. C is an interpolant of $A \to B$ but C is not an abstraction of a quantifier-free weak interpolant of $A \to B$.*

Proof. Let $A := \forall x\,(I(0) \wedge (I(x) \to I(s(x))))$, let $B := I(s(s(0)))$, and let $C := A$. Then $L(A) = L(B) = L(C) = \{0, s, I\}$, $A \to B$ is a valid formula, and C is an interpolant of $A \to B$.

Suppose that C is an abstraction of a quantifier-free weak interpolant C_0. Then C_0 is of the form $I(0) \wedge (I(t) \to I(s(t)))$ for some term t and we would have $\models C_0 \to B$. However,

$$I(0) \wedge (I(t) \to I(s(t))) \to I(s(s(0)))$$

is not a valid formula which can be shown easily by a countermodel $\mathcal{N}$ with domain $\mathbb{N}$ s.t. $0 \in I^{\mathcal{N}}$, $2 \notin I^{\mathcal{N}}$, and $1 \in I^{\mathcal{N}}$ iff $t^{\mathcal{N}} = 0$. □

Therefore, any algorithm that computes only abstractions of weak interpolants is incomplete. In particular: let $\mathcal{M}$ be the interpolation algorithm for first-order $\mathbf{LK}^{-}$ from Section 8.2 in [4]. Clearly,

$\mathcal{M}$ is not complete due to Proposition 10. Let $\mathcal{M}'$ be the (straightforward) extension of $\mathcal{M}$ to first-order $\mathbf{LK}^{\mathrm{at}}$. Then we obtain:

Theorem 54. *$\mathcal{M}'$ is not complete.*

Proof. Let $A \to B$ and C be as in Lemma 53. Then $\mathcal{M}(\pi')$ is an abstraction of a weak interpolant of $A \to B$ and hence different from C. □

Question 55. *Are interpolation algorithms following the second strategy incomplete?*

9 A REMARK ON BETH'S DEFINABILITY THEOREM

Beth's definability theorem is one of the most important applications of interpolation in mathematical logic. As we will briefly point out in this section, the completeness properties of the interpolation theorem apply directly to Beth's definability theorem. For the results in this section, it will be convenient to explicitly indicate all predicate symbols that occur in a first-order formula by writing $A(R_1, \dots, R_n)$.

Definition 56. Let $R, R_1, \dots, R_n$ be predicate symbols. A sentence $A(R, R_1, \dots, R_n)$ is an *implicit definition* of R if

$$A(R, R_1, \dots, R_n) \wedge A(R', R_1, \dots, R_n) \to \forall \vec{x}(R(\vec{x}) \leftrightarrow R'(\vec{x})) \quad (1)$$

is valid.

$A(R, R_1, \dots, R_n)$ is an *explicit definition* of R if there is a formula $F(\vec{x})$ s.t.

$$A(R, R_1, \dots, R_n) \to \forall \vec{x}(R(\vec{x}) \leftrightarrow F(\vec{x})) \quad (2)$$

is valid.

Beth's definability theorem states that whenever R is definable implicitly (in first-order logic), then R is also definable explicitly. We first observe that (1) is valid iff

$$(A(R, R_1, \dots, R_n) \wedge R(\overline{x})) \to (A(R', R_1, \dots, R_n) \to R'(\overline{x})) \quad (3)$$

is valid. Then, in analogy to Definition 8, we can say that an algorithm $\mathcal{D}$ which receives a proof π of (3) as input and returns an F s.t. (2) is valid is complete if for every F there is a π with $F = \mathcal{D}(\pi)$.

The standard proof of Beth's definability theorem from the interpolation theorem, see, e.g. [25], now proceeds as follows: Let $A(R, R_1, \dots, R_n)$ be an implicit definition of R and let π be a proof of (3). Then applying an interpolation algorithm $\mathcal{I}$ to π yields a formula $F = \mathcal{I}(\pi)$ with $L_P(F) \subseteq \{R_1, \dots, R_n\}$ such that both

$$A(R, R_1, \dots, R_n) \wedge R(\overline{x}) \to F(\overline{x})$$

and, by renaming R' to R,

$$F(\overline{x}) \to (A(R, R_1, \dots, R_n) \to R(\overline{x}))$$

are valid. Hence also

$$A(R, R_1, \dots, R_n) \to \forall \vec{x}(R(\vec{x}) \leftrightarrow F(\vec{x}))$$

is valid, so $F(\vec{x})$ is an explicit definition of R. Writing $\mathcal{D}_{\mathcal{I}}$ for the algorithm that takes a proof (3) and returns an explicit definition $F(\vec{x})$ of R we see that $\mathcal{D}_{\mathcal{I}}$ is the restriction of $\mathcal{I}$ to formulas of the form (3). In particular:

Observation 57. *$\mathcal{D}_{\mathcal{I}}$ is complete iff $\mathcal{I}$ is complete on formulas of the form (3).*

10 CONCLUSION

We have initiated the study of completeness properties of interpolation algorithms by proving several results about some of the most important interpolation algorithms: The standard algorithms for resolution and cut-free sequent calculus for propositional logic are incomplete. On the other hand, in the sequent calculus with atomic cuts, it is complete. Moreover, even in the cut-free sequent calculus, one can obtain a weaker completeness result: completeness of pruned interpolants up to subsumption. We have also extended our results to normal modal logics and to first-order logic and found a new source of incompleteness in first-order logic that applies to a wide variety of interpolation algorithms. We have also shown that the completeness properties of interpolation algorithms correspond directly tothe completeness properties of Beth's definability theorem.

These results show that the completeness of an interpolation algorithm is related to the amount of freedom, or redundancy, that is permitted by a proof system and its subtle interplay with the interpolation algorithm, as witnessed very clearly, e.g., by the proof of Theorem 43, the completeness of interpolation in the sequent calculus with atomic cut.

Moreover, our results show clearly that the completeness of a proof calculus (w.r.t. some semantics) is a different question from that of the completeness of an interpolation algorithm in this calculus. For example, both cut-free sequent calculus and sequent calculus with atomic cuts are complete w.r.t. Tarski semantics. However, the standard interpolation algorithm is complete in the latter but not in the former.

We believe that this work is merely a first step in a wider project of gauging the expressive power of interpolation algorithms based on their completeness properties. We have already mentioned many open questions in the paper. We consider the following open problems to be the most promising and relevant: We plan to investigate the completeness of interpolation algorithms for local proofs [14, 16] which are of particular relevance in the CAV community. In order to get a better picture of the situation in first-order logic, it would be useful to investigate the second interpolation strategy for first-order proofs as mentioned in Section 8. Moreover, we are intrigued by the question whether the cut-elimination argument underlying Theorem 39 can be extended to first-order logic. It would be interesting to investigate these questions also for intuitionistic logic where, due to the more restricted availability of normal forms, quite different techniques will presumably be needed. Resolution with weakening for classical propositional logic is interesting from a proof-theoretic point of view since, in contrast to ordinary resolution, it is complete (as a proof calculus, w.r.t. standard semantics). However, the completeness of its interpolation algorithm is unknown. Also, it would be interesting to relate these completeness and incompleteness results more directly to applications in verification, for example based on the relationship between interpolation and narrowing as described in [7]. We leave these, and the questions mentioned throughout the paper, to future work.

11 ACKNOWLEDGEMENTS

The authors would like to thank Iris van der Giessen as well as the anonymous reviewers for many comments that have helped to improve this paper.

REFERENCES

[1] E. Amir and S. McIlraith. Partition-based logical reasoning for first-order and propositional theories. *Artificial intelligence*, 162(1-2):49–88, 2005.
[2] J. Avigad. *Mathematical Logic and Computation*. Cambridge University Press, 2023.
[3] M. Baaz and A. Leitsch. Towards a clausal analysis of cut-elimination. *Journal of Symbolic Computation*, 41(3–4):381–410, 2006.
[4] M. Baaz and A. Leitsch. *Methods of Cut-Elimination*, volume 34 of *Trends in Logic*. Springer, 2011.
[5] E. Beth. On Padoa's Method in the Theory of Definition. *Indagationes Mathematicae (Proceedings)*, 56:330–339, 1953.
[6] S. R. Buss. *Handbook of proof theory*. Elsevier, 1998.
[7] P. Cousot. Abstracting induction by extrapolation and interpolation. In D. D'Souza, A. Lal, and K. G. Larsen, editors, *Verification, Model Checking, and Abstract Interpretation - 16th International Conference, VMCAI 2015, Mumbai, India, January 12-14, 2015. Proceedings*, volume 8931 of *Lecture Notes in Computer Science*, pages 19–42. Springer, 2015.
[8] W. Craig. Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *Journal of Symbolic Logic*, 22(3):269–285, 1957.
[9] V. V. D'Silva, D. Kroening, M. Purandare, and G. Weissenbacher. Interpolant strength. In G. Barthe and M. V. Hermenegildo, editors, *11th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI)*, volume 5944 of *Lecture Notes in Computer Science*, pages 129–145. Springer, 2010.
[10] J. Harrison. *Handbook of Practical Logic and Automated Reasoning*. Cambridge University Press, 2009.
[11] K. Hoder, L. Kovács, and A. Voronkov. Playing in the Grey Area of Proofs. In J. Field and M. Hicks, editors, *Symposium on Principles of Programming Languages (POPL) 2012*, pages 259–272. ACM, 2012.
[12] G. Huang. Constructing craig interpolation formulas. In D. Du and M. Li, editors, *First Annual International Conference on Computing and Combinatorics (COCOON)*, volume 959 of *Lecture Notes in Computer Science*, pages 181–190. Springer, 1995.
[13] R. Jhala and K. L. McMillan. Interpolant-based transition relation approximation. In K. Etessami and S. K. Rajamani, editors, *17th International Conference on Computer Aided Verification (CAV)*, volume 3576 of *Lecture Notes in Computer Science*, pages 39–51. Springer, 2005.
[14] R. Jhala and K. L. McMillan. A practical and complete approach to predicate refinement. In H. Hermanns and J. Palsberg, editors, *12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 3920 of *Lecture Notes in Computer Science*, pages 459–473. Springer, 2006.
[15] R. Jhala and K. L. McMillan. Interpolant-Based Transition Relation Approximation. *Logical Methods in Computer Science*, 3(4), 2007.
[16] L. Kovács and A. Voronkov. Interpolation and symbol elimination. In R. A. Schmidt, editor, *22nd International Conference on Automated Deduction (CADE-22)*, volume 5663 of *Lecture Notes in Computer Science*, pages 199–213. Springer, 2009.
[17] J. Krajíček. Lower bounds to the size of constant-depth propositional proofs. *Journal of Symbolic Logic*, 59(1), 1994.
[18] J. Krajíček. Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. *The Journal of Symbolic Logic*, 62(2):457–486, 1997.
[19] G. Kreisel and J.-L. Krivine. *Elements of Mathematical Logic (Model Theory)*. North-Holland, 1967.
[20] P. Mancosu. Introduction: Interpolations – Essays in Honor of William Craig. *Synthese*, 164(3):313–319, 2008.
[21] K. L. McMillan. Interpolation and model checking. In E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem, editors, *Handbook of Model Checking*, pages 421–446. Springer, 2018.
[22] S. Negri and J. Von Plato. *Structural proof theory*. Cambridge university press, 2008.
[23] G. Nelson and D. C. Oppen. Simplification by cooperating decision procedures. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 1(2):245–257, 1979.
[24] P. Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *The Journal of Symbolic Logic*, 62(3):981–998, 1997.
[25] G. Takeuti. *Proof Theory*. North-Holland, Amsterdam, 2nd edition, March 1987.
[26] C. Tinelli. Cooperation of background reasoners in theory reasoning by residue sharing. *Journal of Automated Reasoning*, 30:1–31, 2003.
[27] A. S. Troelstra and H. Schwichtenberg. *Basic proof theory*. Number 43. Cambridge University Press, 2000.
[28] G. Weissenbacher. Interpolant strength revisited. In A. Cimatti and R. Sebastiani, editors, *15th International Conference on Theory and Applications of Satisfiability Testing (SAT)*, volume 7317 of *Lecture Notes in Computer Science*, pages 312–326. Springer, 2012.