Craig Interpolation for a Semi-Substructural Logic

Niccolò Veltri and Cheng-Syuan Wan

Department of Software Science, Tallinn University of Technology, Estonia.

Contributing authors: niccolo@cs.ioc.ee; cswan@cs.ioc.ee;

Abstract

This work studies Craig interpolation for the logic `SkNMILL`, a substructural logic supporting only directed versions of the structural rules of associativity and unitality. In this setting, Craig interpolation cannot be proved by directly employing standard proof-theoretic methods, such as Maehara's method, a situation that `SkNMILL` shares with other logical systems such as the product-free Lambek calculus and the implicational fragment of intuitionistic logic. We show how to overcome this issue and appropriately modify Maehara's method for recovering Craig interpolation. We take one step further and, following the category-theoretic perspective of Čubrić, we produce a proof-relevant version of the interpolation theorem, in which we show that our interpolation procedures are right inverses of the admissible cut rules. All our results have been formalized in the proof assistant Agda.

Keywords: Craig interpolation, semi-substructural logic, Maehara's method, proof-relevant interpolation, Agda

1 Introduction

Craig interpolation is a fundamental result in first-order logic, named after the logician William Craig [1]. A logic $\mathcal{L}$ has the *Craig interpolation property* if, for any formula $A \to C$ provable in $\mathcal{L}$ (where $\to$ is the implication connective in $\mathcal{L}$), there exists a formula D such that $A \to D$ and $D \to C$ are provable in $\mathcal{L}$, satisfying the variable condition: $\mathsf{var}(D) \subseteq \mathsf{var}(A) \cap \mathsf{var}(C)$, where $\mathsf{var}(A)$ is the set of atomic formulae appearing in A. Craig interpolation has been mostly employed to prove model-theoretical results, including Beth's definability theorem [2], but more recently it has found applications in other areas, e.g. in model checking [3].

From the viewpoint of sequent calculus, substructural logics are defined by the absence of at least one structural rule. A notable instance is Lambek's syntactic calculus [4], which forbids weakening, contraction and exchange. Its non-associative variant, also introduced by Lambek [5], disallows associativity as well. Another significant example is linear logic, introduced by Girard [6], where weakening and contraction are disallowed but can be recovered for specific formulae through modalities. Substructural logics have been proved useful for modelling various phenomena in different research areas, from the computational analysis of natural language syntax to the development of programming languages sensitive to resource management.

Craig interpolation for substructural logics has been extensively studied, using either algebraic or proof-theoretic techniques.

For substructural logics that lack a cut-free sequent calculus, such as arbitrary extensions of the full Lambek calculus with exchange ($\mathtt{FL_e}$), Craig interpolation is established using algebraic methods such as amalgamation. For further details on this approach, we refer the reader to a recent paper by Fussner and Santschi [7]. For the relationship between amalgamation and interpolation properties in substructural logics we refer to a paper by Kihara and Ono [8].

For substructural logics that admit a cut-free sequent calculus, Craig interpolation is typically proven using an adaptation of Maehara's method [9], which Maehara originally applied to the sequent calculus LK for classical logic. This class of substructural logics includes the full Lambek calculus (FL) and its extensions that incorporate various combinations of weakening, exchange, and contraction. In the case of FL, for instance, the proof starts by establishing a stronger form of interpolation which we call *Maehara interpolation property* (MIP) [10]. The property states:

(MIP for FL) Given $f : \Gamma \vdash C$ and a partition $\langle \Gamma_0, \Gamma_1, \Gamma_2 \rangle$ of Γ, there exist a formula D and two derivations $g : \Gamma_1 \vdash D$ and $h : \Gamma_0, D, \Gamma_2 \vdash C$ such that $\mathsf{var}(D) \subseteq \mathsf{var}(\Gamma_0) \cap \mathsf{var}(\Gamma_0, \Gamma_1, C)$

Being a partition simply means that the ordered list of formulae Γ is equal to the concatenation of Γ_0, Γ_1 and Γ_2, i.e. $\Gamma = \Gamma_0, \Gamma_1, \Gamma_2$. Maehara interpolation also holds for the commutative variant of FL, called $\mathtt{FL_e}$. In the commutative case, Γ is an unordered list of formulae, i.e. a finite multiset, and it is partitioned as a pair of multisets instead of a triple of lists. This simplification is allowed by the fact that the order of formulae in the antecedent is irrelevant, and therefore Γ_0 and Γ_2 can be combined together into a single multiset.

FL without additive connectives enjoys a stronger variant of Maehara interpolation where the variable condition is replaced by a variable *multiplicity* condition [11]. Let $\sigma_X(A)$ be the number of occurrences of the atomic formula X in the formula A, and $\sigma_X(\Gamma)$ be the number of occurrences of X in the list of formulae Γ. The stronger variant of Maehara interpolation states:

(MIP for FL with variable multiplicity condition) Given $f : \Gamma \vdash C$ and a partition $\langle \Gamma_0, \Gamma_1, \Gamma_2 \rangle$ of Γ, there exist a formula D and two derivations $g : \Gamma_1 \vdash D$ and $h : \Gamma_0, D, \Gamma_2 \vdash C$ such that $\sigma_X(D) \leq \sigma_X(\Gamma_1)$ and $\sigma_X(D) \leq \sigma_X(\Gamma_0, \Gamma_2, C)$ for all atomic X.

Notice that Craig interpolation is a property of a logic (with a notion of implication), while Maehara interpolation is a property of a deductive system in which it is possible to appropriately partition antecedents.

Maehara interpolation is a stronger form of the so-called *deductive interpolation property*. A logic $\mathcal{L}$ has the deductive interpolation property if, for any formulae A and C, whenever $A \vdash C$ (where $\vdash$ is the consequence relation of $\mathcal{L}$), then there exists a formula B such that $A \vdash B$ and $B \vdash C$ while also satisfying the usual variable condition. Furthermore, if the sequent calculus of $\mathcal{L}$ admits the invertibility of implication-right rules (as is the case in `FL` for both left and right implication), Craig interpolation follows immediately as a consequence of deductive interpolation.

While Maehara's method is often applicable to extensions of `FL`, it does not work for some of its fragments, which therefore do not enjoy Maehara interpolation. This is the case for fragments lacking multiplicative and/or additive conjunction, such as the product-free Lambek calculus [12] (with only left and right implications as connectives) and the implicational fragment of intuitionistic logic [13]. The variant of Maehara interpolation satisfied by the product-free Lambek calculus, which we dub *Maehara multi-interpolation* (MMIP), is particularly relevant for our work. Here is its statement, which we have slightly modified to better align with our forthcoming discussion:

(MMIP for product-free Lambek calculus) Given $f : \Gamma \vdash C$ and a partition $\langle \Gamma_0, \Gamma_1, \Gamma_2 \rangle$ of Γ, there exist
- a partition $\langle \Delta_1, \dots, \Delta_n \rangle$ of Γ_1,
- a list of interpolant formulae $D_1, \dots, D_n$,
- derivations $g_i : \Delta_i \vdash D_i$ for all $i \in [1, \dots, n]$,
- a derivation $h : \Gamma_0, D_1, \dots, D_n, \Gamma_2 \vdash C$, such that
- $\sigma_X(D_1, \dots, D_n) \leq \sigma_X(\Gamma_1)$ and $\sigma_X(D_1, \dots, D_n) \leq \sigma_X(\Gamma_0, \Gamma_2, C)$ for all atomic formulae X.

Differently from Maehara interpolation, in the above property we look for a list of interpolants instead of a single formula. This adjustment allows to overcome the difficulty caused by the absence of multiplicative conjunction.

In this paper, we aim at proving Craig interpolation for the *semi-substructural* logic `SkNMILL` which we recently introduced in collaboration with Tarmo Uustalu [14]. In our terminology, a logic is semi-substructural if it is an intermediate logic in between (certain fragments of) non-associative and associative intuitionistic linear logic (or the Lambek calculus). Sequents in `SkNMILL` are in the form $S \mid \Gamma \vdash A$, where the antecedent consists of an optional formula S, called stoup, adapted from Girard [15], and an ordered list of formulae Γ. The succedent is a single formula A. We restrict the application of introduction rules in an appropriate way to allow only one of the directions of associativity and unitality, i.e. only the sequents $(A \otimes B) \otimes C \mid \vdash A \otimes (B \otimes C)$, $I \otimes A \mid \vdash A$, and $A \mid \vdash A \otimes I$ are provable in `SkNMILL`, while their inverses $A \otimes (B \otimes C) \mid \vdash (A \otimes B) \otimes C$, $A \mid \vdash I \otimes A$, and $A \otimes I \mid \vdash A$ are not generally provable. In other words, only directed variants of the structural rules of associativity and unitality are included, while their inverses are generally disallowed.

The introduction of semi-substructural logics was originally motivated by the study of combinatorial properties of certain categorical structures, called *left skew monoidal*

categories [16]. These categories are a weaker variant of MacLane's monoidal categories. In left skew monoidal categories, the structural morphisms of associativity and unitality (which are natural transformations typically called 'associator' and 'unitors') are not required to have an inverse. Instead, they are natural family of morphisms with a specific orientation. For this reason, left skew monoidal categories can be seen as *semi-associative* and *semi-unital* variants of monoidal categories.

Different variants of left skew monoidal categories have led to the development of their corresponding semi-substructural logic. These include (i) left skew semigroup [17], (ii) left skew monoidal [18], (iii) left skew (prounital) closed [19], (iv) left skew monoidal closed categories [14, 20, 21], and (v) left distributive skew monoidal categories with binary products and coproducts [22]. Each of these logics admits a cut-free sequent calculus. Moreover, they admit a subcalculus of "proofs in normal form", which is inspired by Andreoli's focusing method [23] and as such provides a way to steer root-first proof search and make it more deterministic. Practically, the focusing method is employed for solving the coherence problem for the corresponding variants of left skew monoidal categories. In the case of left skew monoidal closed categories, a solution to the coherence (or word) problem consists of a procedure for deciding equality of parallel morphisms in the free left skew monoidal closed category on a given set At. In previous work, we showed that the focused subcalculus for `SkNMILL` is a concrete presentation of such free category, so a solution to the coherence problem is obtained by checking whether two morphisms are represented by the same derivation in the focused subcalculus.

To prove Craig interpolation for `SkNMILL`, we need to modify the statement of Maehara interpolation. This modification is required due to issues similar to those encountered in the product-free Lambek calculus [12] and the implicational fragment of intuitionistic logic [13], where Maehara interpolation fails. The main result of the paper is the following:

Theorem 5. *In the sequent calculus for* `SkNMILL`*, the following two interpolation properties hold:*

(sMIP) *Given a derivation* $f : S \mid \Gamma \vdash C$ *and a partition* $\langle \Gamma_0, \Gamma_1 \rangle$ *of* Γ*, there exist*
- *an interpolant formula* D*,*
- *a derivation* $g : S \mid \Gamma_0 \vdash D$*,*
- *a derivation* $h : D \mid \Gamma_1 \vdash C$*, such that*
- $\sigma_X(D) \leq \sigma_X(S, \Gamma_0)$ *and* $\sigma_X(D) \leq \sigma_X(\Gamma_1, C)$ *for all atomic formulae* X*.*

(cMMIP) *Given a derivation* $f : S \mid \Gamma \vdash C$ *and a partition* $\langle \Gamma_0, \Gamma_1, \Gamma_2 \rangle$ *of* Γ*, there exist*
- *a partition* $\langle \Delta_1, \ldots, \Delta_n \rangle$ *of* Γ_1*,*
- *a list of interpolant formulae* $D_1, \ldots, D_n$*,*
- $g_i : - \mid \Delta_i \vdash D_i$ *for all* $i \in [1, \ldots, n]$*,*
- $h : S \mid \Gamma_0, D_1, \ldots, D_n, \Gamma_2 \vdash C$*, such that*
- $\sigma_X(D_1, \ldots, D_n) \leq \sigma_X(\Gamma_1)$ *and* $\sigma_X(D_1, \ldots, D_n) \leq \sigma_X(S, \Gamma_0, \Gamma_2, C)$ *for all atomic formulae* X*.*

In `SkNMILL`, the first property sMIP, which stands for *stoup Maehara interpolation*, resembles Maehara interpolation for the `FL`. Whereas cMMIP, which stands for *context Maehara multi-interpolation*, is similar to Maehara multi-interpolation for the product-free Lambek calculus.

Motivated by the categorical interpretation of SkNMILL, we take one more step and investigate the interplay between the admissible cut rules (called scut and ccut) and the derivations produced by the interpolation algorithm of Theorem 5. In previous work [14], we introduced an equivalence relation on derivations ($\mathrel{\mathring{=}}$) that captures η-conversions and permutative conversions, and is both sound and complete with respect to the categorical semantics. We show that the sMIP and cMMIP procedures of Theorem 5 are right inverses of the admissible rules scut and ccut, respectively. Formally, we prove the following theorem:

Theorem 9.

(i) *Let* $g : S \mid \Gamma_0 \vdash D$ *and* $h : D \mid \Gamma_1 \vdash C$ *be the derivations obtained by applying the sMIP procedure on a derivation* $f : S \mid \Gamma \vdash C$ *with the partition* $\langle \Gamma_0, \Gamma_1 \rangle$. *Then* $\mathsf{scut}(g, h) \mathrel{\mathring{=}} f$.

(ii) *Let* $g_i : - \mid \Delta_i \vdash D_i$ *for* $i \in [1, \dots, n]$ *and* $h : S \mid \Gamma_0, D_1, \dots, D_n, \Gamma_2 \vdash C$ *be derivations obtained by applying the cMMIP procedure on a derivation* $f : S \mid \Gamma \vdash C$ *with the partition* $\langle \Gamma_0, \Gamma_1, \Gamma_2 \rangle$. *Then* $\mathsf{ccut}^*([g_i], h) \mathrel{\mathring{=}} f$.

In the above statement, ccut^* denotes multiple applications of the admissible ccut rule, one for each derivation h_i. Theorems 5 and 9 together show that SkNMILL satisfies a *proof-relevant* form of Craig interpolation, in the sense formulated in the early 90s by Čubrić [24] in the setting of intuitionistic propositional logic and recently discussed also by Saurin [25] for (extensions of) classical linear logic.

All the proofs presented in this paper have been formalized in the proof assistant Agda. In Section 7 we showcase some aspects of the formalization. We discuss the datatypes employed for the representation of formulae, sequents and equivalence of derivations, and explain how the results presented in the paper translate to Agda types and terms. The full formalization is freely available at the following website:

https://github.com/niccoloveltri/code-skewmonclosed/tree/interpolation.

2 A Sequent Calculus for SkNMILL

We start by recalling the sequent calculus for left skew monoidal closed categories that was introduced in [14], which we name SkNMILL. This is a "skew variant" of non-commutative multiplicative intuitionistic linear logic, in a sense that will be made precise later in this section.

Formulae are inductively generated by the grammar $A, B ::= X \mid \mathsf{I} \mid A \otimes B \mid A \multimap B$, where X comes from a set At of atoms, I is a multiplicative unit, $\otimes$ is multiplicative conjunction and $\multimap$ is a linear implication. The set of formulae is denoted Fma.

A sequent is a triple of the form $S \mid \Gamma \vdash A$. The antecedent consists of two parts: an optional formula S, called the *stoup* (a terminology that comes from Girard [15]), and an ordered list of formulae Γ, that we call the *context*. The succedent A is a single formula. The symbol S consistently denotes a stoup, meaning S can either be a single formula or empty, indicated as $S = -$. Furthermore, letters X, Y, Z and W always denote atomic formulae.

Derivations are generated recursively by the following rules:

$$\frac{}{A \mid \ \vdash A}\ \mathsf{ax} \qquad \frac{A \mid \Gamma \vdash C}{- \mid A, \Gamma \vdash C}\ \mathsf{pass}$$

$$\frac{- \mid \Gamma \vdash A \quad B \mid \Delta \vdash C}{A \multimap B \mid \Gamma, \Delta \vdash C}\ \multimap\mathsf{L} \qquad \frac{S \mid \Gamma, A \vdash B}{S \mid \Gamma \vdash A \multimap B}\ \multimap\mathsf{R} \tag{1}$$

$$\frac{- \mid \Gamma \vdash C}{\mathsf{I} \mid \Gamma \vdash C}\ \mathsf{IL} \qquad \frac{}{- \mid \ \vdash \mathsf{I}}\ \mathsf{IR}$$

$$\frac{A \mid B, \Gamma \vdash C}{A \otimes B \mid \Gamma \vdash C}\ \otimes\mathsf{L} \qquad \frac{S \mid \Gamma \vdash A \quad - \mid \Delta \vdash B}{S \mid \Gamma, \Delta \vdash A \otimes B}\ \otimes\mathsf{R}$$

The inference rules in (1) are similar to the ones in the sequent calculus for non-commutative multiplicative intuitionistic linear logic [26], but with some crucial differences:

1. The left logical rules IL, $\otimes\mathsf{L}$ and $\multimap\mathsf{L}$, read bottom-up, are only allowed to be applied on the formula in the stoup position.
2. The right tensor rule $\otimes\mathsf{R}$, read bottom-up, splits the antecedent of a sequent $S \mid \Gamma, \Delta \vdash A \otimes B$ and in the case where S is a formula, S is always moved to the stoup of the left premise, even if Γ is empty.
3. The presence of the stoup distinguishes two types of antecedents, $A \mid \Gamma$ and $- \mid A, \Gamma$. The structural rule pass (for 'passivation'), read bottom-up, allows the moving of the leftmost formula in the context to the stoup position whenever the stoup is empty.
4. The logical connectives of non-commutative multiplicative intuitionistic linear logic typically include two ordered implications (often also called residuals) $\multimap$ and $\circ\!\!-$ (or $\backslash$ and $/$ in the notation of the Lambek calculus), which are two variants of linear implication arising from the removal of the exchange rule from intuitionistic linear logic. In our logic `SkNMILL`, only implication is present. In our notation $\multimap$ denotes a *right* implication, although in the Lambek calculus literature the symbol usually denotes a left implication.

When we call `SkNMILL` a *semi-associative and semi-unital* logic, we refer to the fact that only directed versions of associativity and unitality of $\otimes$ and I are derivable

in the sequent calculus. More precisely, the following derivations are valid:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \mid \ \vdash A}\,\mathsf{ax}}{- \mid A \vdash A}\,\mathsf{pass}}{I \mid A \vdash A}\,\mathsf{IL}}{I \otimes A \mid \ \vdash A}\,\otimes\mathsf{L} \qquad \dfrac{\dfrac{}{A \mid \ \vdash A}\,\mathsf{ax} \quad \dfrac{}{- \mid \ \vdash I}\,\mathsf{IR}}{A \mid \ \vdash A \otimes I}\,\otimes\mathsf{R}$$

$$\dfrac{\dfrac{}{A \mid \ \vdash A}\,\mathsf{ax} \quad \dfrac{\dfrac{\dfrac{}{B \mid \ \vdash B}\,\mathsf{ax} \quad \dfrac{\dfrac{}{C \mid \ \vdash C}\,\mathsf{ax}}{- \mid C \vdash C}\,\mathsf{pass}}{B \mid C \vdash B \otimes C}\,\otimes\mathsf{R}}{- \mid B, C \vdash B \otimes C}\,\mathsf{pass}}{\dfrac{A \mid B, C \vdash A \otimes (B \otimes C)}{A \otimes B \mid C \vdash A \otimes (B \otimes C)}\,\otimes\mathsf{L}}\,\otimes\mathsf{R}}{(A \otimes B) \otimes C \mid \ \vdash A \otimes (B \otimes C)}\,\otimes\mathsf{L} \tag{2}$$

On the other hand, the "inverses" of the conclusions in (2), obtained by swapping the stoup formula with the succedent formula, do not have any derivation. All possible attempts of constructing a valid derivation for each of them end in failure.

$$\dfrac{X \mid \overset{??}{\vdash} I \quad - \mid \overset{??}{\vdash} X}{X \mid \ \vdash I \otimes X}\,\otimes\mathsf{R} \qquad \dfrac{X \mid \overset{??}{I \vdash} X}{X \otimes I \mid \ \vdash X}\,\otimes\mathsf{L}$$

($\otimes\mathsf{R}$ sends X to 1st premise) (IL does not act on I in context)

$$\dfrac{X \mid Y \otimes Z \overset{??}{\vdash} (X \otimes Y) \otimes Z}{X \otimes (Y \otimes Z) \mid \ \vdash (X \otimes Y) \otimes Z}\,\otimes\mathsf{L}$$

($\otimes\mathsf{L}$ does not act on $\otimes$ in context)

`SkNMILL`, as a semi-substructural logic, can be seen as an intermediate logic between: (*i*) the $(\mathsf{I}, \otimes, \multimap)$-fragment of non-commutative intuitionistic linear logic (where $\multimap$ is right residual $/$), i.e. the associative Lambek calculus with multiplicative unit, and (*ii*) the non-associative variant of this fragment. In fact, every derivation in `SkNMILL` can be replicated in the associative Lambek calculus with multiplicative unit, and every derivation in the non-associative Lambek calculus without left residual but with multiplicative unit can be replicated in `SkNMILL`.

This calculus is cut-free, in the sense that the following two rules are admissible:

$$\dfrac{S \mid \Gamma \vdash A \quad A \mid \Delta \vdash C}{S \mid \Gamma, \Delta \vdash C}\,\mathsf{scut} \qquad \dfrac{- \mid \Gamma \vdash A \quad S \mid \Delta_0, A, \Delta_1 \vdash C}{S \mid \Delta_0, \Gamma, \Delta_1 \vdash C}\,\mathsf{ccut} \tag{3}$$

The presence of two cut rules comes from the fact that the cut formula can appear either in the stoup or the context of the second premise.

We introduce a few admissible rules that will be employed later in the paper. First, given a list of formulae $\Delta = A_1, \ldots, A_n$, we define an iterated version of the rule $\multimap\mathsf{R}$, consisting of n applications of $\multimap\mathsf{R}$. Below and in the future we write $\Delta \multimap^* B$ for the formula $A_1 \multimap (A_2 \multimap (\ldots (A_n \multimap B)\ldots))$, which is simply B when Δ is empty. The double-line inference rule denotes an equality of sequents.

$$\frac{\overset{f}{S \mid \Gamma, \Delta \vdash B}}{S \mid \Gamma \vdash \Delta \multimap^* B}\ \multimap\mathsf{R}^* \quad = \quad \cfrac{\cfrac{\cfrac{\overset{f}{S \mid \Gamma, \Delta \vdash B}}{S \mid \Gamma, A_1, A_2, \ldots, A_n \vdash B}}{S \mid \Gamma, A_1, A_2, \ldots, A_{n-1} \vdash A_n \multimap B}\ \multimap\mathsf{R} \\ \vdots \\ \cfrac{S \mid \Gamma, A_1 \vdash A_2 \multimap (\ldots (A_n \multimap B)\ldots)}{S \mid \Gamma \vdash A_1 \multimap (A_2 \multimap (\ldots (A_n \multimap B)\ldots))}\ \multimap\mathsf{R}}{S \mid \Gamma \vdash \Delta \multimap^* B} \tag{4}$$

If $n = 0$, then $\multimap\mathsf{R}^* f = f$.

Second, given a list of formulae $\Delta = A_1, \ldots, A_n$ and a list of derivations $f_i : - \mid \Gamma_i \vdash A_i$ for $i \in [1, \ldots, n]$, we define an iterated version of $\multimap\mathsf{L}$, consisting of n applications of $\multimap\mathsf{L}$, one for each derivation f_i:

$$\frac{\overset{[f_i]}{[- \mid \Gamma_i \vdash A_i]_i} \quad \overset{g}{B \mid \Lambda \vdash C}}{\Delta \multimap^* B \mid \Gamma_1, \ldots, \Gamma_n, \Lambda \vdash C}\ \multimap\mathsf{L}^* \qquad\qquad \frac{\overset{f_n}{- \mid \Gamma_n \vdash A_n} \quad \overset{g}{B \mid \Lambda \vdash C}}{A_n \multimap B \mid \Gamma_n, \Lambda \vdash C}\ \multimap\mathsf{L}$$

$$= \quad \cfrac{\overset{f_1}{- \mid \Gamma_1 \vdash A_1} \quad \overset{\vdots}{A_2 \multimap (\ldots (A_n \multimap B)\ldots) \mid \Gamma_2, \ldots, \Gamma_n, \Lambda \vdash C}}{\cfrac{A_1 \multimap (A_2 \multimap (\ldots (A_n \multimap B)\ldots)) \mid \Gamma_1, \Gamma_2, \ldots, \Gamma_n, \Lambda \vdash C}{\Delta \multimap^* B \mid \Gamma_1, \ldots, \Gamma_n, \Lambda \vdash C}}\ \multimap\mathsf{L} \tag{5}$$

The rule $\multimap\mathsf{L}^*$ has $n + 1$ premises, the first n are collected in the list of sequents $[- \mid \Gamma_i \vdash A_i]_i$. If $n = 0$, then $\multimap\mathsf{L}^*([\,], g) = g$.

Finally, given a list of derivations $f_i : - \mid \Delta_i \vdash A_i$ for $i \in [1, \ldots, n]$, we define an iterated version of ccut, consisting on n applications of ccut, one for each derivation f_i:

$$\dfrac{\begin{matrix}[f_i] & g \\ [- \mid \Delta_i \vdash A_i]_i & S \mid \Gamma_0, A_1, \ldots, A_n, \Gamma_1 \vdash C\end{matrix}}{S \mid \Gamma_0, \Delta_1, \Delta_2, \ldots, \Delta_n, \Gamma_1 \vdash C}\ \mathsf{ccut}^*$$

$$= \dfrac{\begin{matrix}f_n & g \\ - \mid \Delta_n \vdash A_n & S \mid \Gamma_0, A_1, A_2, \ldots, A_n, \Gamma_1 \vdash C\end{matrix}}{S \mid \Gamma_0, A_1, A_2, \ldots, \Delta_n, \Gamma_1 \vdash C}\ \mathsf{ccut} \qquad (6)$$

$$= \dfrac{\begin{matrix}f_1 & \vdots \\ - \mid \Delta_1 \vdash A_1 & S \mid \Gamma_0, A_1, \Delta_2, \ldots, \Delta_n, \Gamma_1 \vdash C\end{matrix}}{S \mid \Gamma_0, \Delta_1, \Delta_2, \ldots, \Delta_n, \Gamma_1 \vdash C}\ \mathsf{ccut}$$

If $n = 0$, then $\mathsf{ccut}^*([\,], g) = g$.

3 Equivalence of Derivations

Sets of derivations are quotiented by a congruence relation $\doteq$, generated by the pairs of derivations in Figure 1 and 2. The eight equations in Figure 1 are permutative conversions. The three equations in Figure 2 are η-conversions, completely characterizing the ax rule on non-atomic formulae.

The generating equations of $\doteq$ have been carefully selected to appropriately match the equational theory of left skew monoidal closed categories. More information about this relationship in terms of categorical semantics can be found in [14], where a precise correspondence between sequent calculus derivations, the congruence relation $\doteq$ and left skew monoidal closed categories is described. The latter paper also contains an interpretation of the sequent calculus as a logic of resources, as well as a calculus of derivations in normal form, which completely characterizes proofs modulo the congruence relation $\doteq$.

More equations on derivations hold in `SkNMILL` due to the cut-elimination procedures defined in [14, 21], which fully describes the possible interaction between different applications of the cut rules. The first set of equations in Figure 4 shows that parallel composition of cut rules is commutative. The second set of equations in Figure 3 shows that sequential composition of cut rules is associative. Analogous equations have been proved in [18] for the fragment of `SkNMILL` without linear implication. Notice that pairs of derivations in these equations are *strictly* equal, not merely $\doteq$-related.

Proposition 1. *The commutativity equations in Figure 4 and the associativity equations in Figure 3 are admissible.*

The proof of Proposition 1 proceeds by mutual induction on the structure of derivations. There are many cases to consider. We do not include the long proof here and refer the interested reader to consult our Agda formalization. Heavy proofs by pattern matching like this one is where the employment of a proof assistant becomes very helpful, in our experience.

We conclude this section by introducing a final equation and two equivalences, that will be employed later in Section 6. In the construction of the scut admissibility procedure [14, 21], the case when the first premise is of the form $\multimap\mathsf{R}\ f$ and the second premise of the form $\multimap\mathsf{L}\ (g, h)$ (i.e. a principal cut when the cut formula is an implication) is defined as follows:

$$\dfrac{\dfrac{\overset{f}{S \mid \Gamma, A \vdash B}}{S \mid \Gamma \vdash A \multimap B}\ \multimap\mathsf{R} \quad \dfrac{\overset{g}{- \mid \Delta \vdash A} \quad \overset{h}{B \mid \Lambda \vdash C}}{A \multimap B \mid \Delta, \Lambda \vdash C}\ \multimap\mathsf{L}}{S \mid \Gamma, \Delta, \Lambda \vdash C}\ \mathsf{scut}$$

$$= \dfrac{\overset{g}{- \mid \Delta \vdash A} \quad \overset{f}{S \mid \Gamma, A \vdash B}}{S \mid \Gamma, \Delta \vdash B}\ \mathsf{ccut} \quad \overset{h}{B \mid \Lambda \vdash C}}{S \mid \Gamma, \Delta, \Lambda \vdash C}\ \mathsf{scut}$$

This equation can be generalized to one where $\multimap\mathsf{R}$, $\multimap\mathsf{L}$ and ccut are replaced by their iterated versions $\multimap\mathsf{R}^*$, $\multimap\mathsf{L}^*$ and ccut^* introduced in (4), (5) and (6).

Proposition 2. *Given a list of formulae $\Lambda = A_1, \ldots, A_n$, a derivation $f : S \mid \Gamma_0, \Lambda \vdash B$ and a list of derivations $g_i : - \mid \Delta_i \vdash A_i$ for $i \in [1, \ldots, n]$, the following equation is derivable:*

$$\dfrac{\dfrac{\overset{f}{S \mid \Gamma_0, \Lambda \vdash B}}{S \mid \Gamma_0 \vdash \Lambda \multimap^* B}\ \multimap\mathsf{R}^* \quad \dfrac{\overset{[g_i]}{[- \mid \Delta_i \vdash A_i]_i} \quad \overset{h}{B \mid \Gamma_1 \vdash C}}{\Lambda \multimap^* B \mid \Delta_1, \ldots, \Delta_n, \Gamma_1 \vdash C}\ \multimap\mathsf{L}^*}{S \mid \Gamma_0, \Delta_1, \ldots, \Delta_n, \Gamma_1 \vdash C}\ \mathsf{scut}$$

$$= \dfrac{\dfrac{\overset{[g_i]}{[- \mid \Delta_i \vdash A_i]_i} \quad \overset{f}{S \mid \Gamma_0, \Lambda \vdash B}}{S \mid \Gamma_0, \Delta_1, \ldots, \Delta_n \vdash B}\ \mathsf{ccut}^* \quad \overset{h}{B \mid \Gamma_1 \vdash C}}{S \mid \Gamma_0, \Delta_1, \ldots, \Delta_n, \Gamma_1 \vdash C}\ \mathsf{scut}$$

Proof. Proving the validity of the equation requires various applications of the associativity equations in Proposition 1. □

The admissibility of rule scut is proved in [14, 21] by structural recursion on the derivation of the left premise. This implies that "scut commutes with left rules in first premise", i.e. that $\mathsf{scut}(\odot\mathsf{L}\ f, g) = \odot\mathsf{L}(\mathsf{scut}(f, g))$ for any one-premise left rule $\odot\mathsf{L}$ among IL, $\otimes\mathsf{L}$ and pass, and also $\mathsf{scut}(\multimap\mathsf{L}(f, f'), g) = \multimap\mathsf{L}(f, \mathsf{scut}(f', g))$. It is possible to also show that "scut commutes with right rules in second premise", but only up to equivalence $\mathrel{\mathring{=}}$.

$$\dfrac{\dfrac{\overset{f}{A' \mid \Gamma \vdash A}}{- \mid A', \Gamma \vdash A}\,\mathsf{pass} \qquad \overset{g}{- \mid \Delta \vdash B}}{- \mid A', \Gamma, \Delta \vdash A \otimes B}\,\otimes\mathsf{R} \;\doteq\; \dfrac{\dfrac{\overset{f}{A' \mid \Gamma \vdash A} \quad \overset{g}{- \mid \Delta \vdash B}}{A' \mid \Gamma, \Delta \vdash A \otimes B}\,\otimes\mathsf{R}}{- \mid A', \Gamma, \Delta \vdash A \otimes B}\,\mathsf{pass}$$

$$\dfrac{\dfrac{\overset{f}{- \mid \Gamma \vdash A}}{\mathsf{I} \mid \Gamma \vdash A}\,\mathsf{IL} \qquad \overset{g}{- \mid \Delta \vdash B}}{\mathsf{I} \mid \Gamma, \Delta \vdash A \otimes B}\,\otimes\mathsf{R} \;\doteq\; \dfrac{\dfrac{\overset{f}{- \mid \Gamma \vdash A} \quad \overset{g}{- \mid \Delta \vdash B}}{- \mid \Gamma, \Delta \vdash A \otimes B}\,\otimes\mathsf{R}}{\mathsf{I} \mid \Gamma, \Delta \vdash A \otimes B}\,\mathsf{IL}$$

$$\dfrac{\dfrac{\overset{f}{A' \mid B', \Gamma \vdash A}}{A' \otimes B' \mid \Gamma \vdash A}\,\otimes\mathsf{L} \qquad \overset{g}{- \mid \Delta \vdash B}}{A' \otimes B' \mid \Gamma, \Delta \vdash A \otimes B}\,\otimes\mathsf{R} \;\doteq\; \dfrac{\dfrac{\overset{f}{A' \mid B', \Gamma \vdash A} \quad \overset{g}{- \mid \Delta \vdash B}}{A' \mid B', \Gamma, \Delta \vdash A \otimes B}\,\otimes\mathsf{R}}{A' \otimes B' \mid \Gamma, \Delta \vdash A \otimes B}\,\otimes\mathsf{L}$$

$$\dfrac{\overset{f}{- \mid \Gamma \vdash C} \quad \overset{g}{D \mid \Delta \vdash A}}{C \multimap D \mid \Gamma, \Delta \vdash A}\,\multimap\mathsf{L} \qquad \overset{h}{- \mid \Lambda \vdash B}}{C \multimap D \mid \Gamma, \Delta, \Lambda \vdash A \otimes B}\,\otimes\mathsf{R} \;\doteq\; \dfrac{\overset{f}{- \mid \Gamma \vdash C} \quad \dfrac{\overset{g}{D \mid \Delta \vdash A} \quad \overset{h}{- \mid \Lambda \vdash B}}{D \mid \Delta, \Lambda \vdash A \otimes B}\,\otimes\mathsf{R}}{C \multimap D \mid \Gamma, \Delta, \Lambda \vdash A \otimes B}\,\multimap\mathsf{L}$$

$$\dfrac{\dfrac{\overset{f}{A' \mid \Gamma, A \vdash B}}{A' \mid \Gamma \vdash A \multimap B}\,\multimap\mathsf{R}}{- \mid A', \Gamma \vdash A \multimap B}\,\mathsf{pass} \;\doteq\; \dfrac{\dfrac{\overset{f}{A' \mid \Gamma, A \vdash B}}{- \mid A', \Gamma, A \vdash B}\,\mathsf{pass}}{- \mid A', \Gamma \vdash A \multimap B}\,\multimap\mathsf{R}$$

$$\dfrac{\dfrac{\overset{f}{- \mid \Gamma, A \vdash B}}{- \mid \Gamma \vdash A \multimap B}\,\multimap\mathsf{R}}{\mathsf{I} \mid \Gamma \vdash A \multimap B}\,\mathsf{IL} \;\doteq\; \dfrac{\dfrac{\overset{f}{- \mid \Gamma, A \vdash B}}{\mathsf{I} \mid \Gamma, A \vdash B}\,\mathsf{IL}}{\mathsf{I} \mid \Gamma \vdash A \multimap B}\,\multimap\mathsf{R}$$

$$\dfrac{\dfrac{\overset{f}{A' \mid B', \Gamma, A \vdash B}}{A' \mid B', \Gamma \vdash A \multimap B}\,\multimap\mathsf{R}}{A' \otimes B' \mid \Gamma \vdash A \multimap B}\,\otimes\mathsf{L} \;\doteq\; \dfrac{\dfrac{\overset{f}{A' \mid B', \Gamma, A \vdash B}}{A' \otimes B' \mid \Gamma, A \vdash B}\,\otimes\mathsf{L}}{A' \otimes B' \mid \Gamma \vdash A \multimap B}\,\multimap\mathsf{R}$$

$$\dfrac{\overset{f}{- \mid \Gamma \vdash A'} \quad \dfrac{\overset{g}{B' \mid \Delta, A \vdash B}}{B' \mid \Delta \vdash A \multimap B}\,\multimap\mathsf{R}}{A' \multimap B' \mid \Gamma, \Delta \vdash A \multimap B}\,\multimap\mathsf{L} \;\doteq\; \dfrac{\overset{f}{- \mid \Gamma \vdash A'} \quad \overset{g}{B' \mid \Delta, A \vdash B}}{A' \multimap B' \mid \Gamma, A \vdash B}\,\multimap\mathsf{L}}{A' \multimap B' \mid \Gamma, \Delta \vdash A \multimap B}\,\multimap\mathsf{R}$$

Fig. 1 Equivalence of derivations: permutative conversions

$$\frac{}{\mathsf{I} \mid \quad \vdash \mathsf{I}}\ \mathsf{ax} \quad \triangleq \quad \frac{\dfrac{}{- \mid \quad \vdash \mathsf{I}}\ \mathsf{IR}}{\mathsf{I} \mid \quad \vdash \mathsf{I}}\ \mathsf{IL}$$

$$\frac{}{A \otimes B \mid \quad \vdash A \otimes B}\ \mathsf{ax} \quad \triangleq \quad \frac{\dfrac{}{A \mid \quad \vdash A}\ \mathsf{ax} \quad \dfrac{\dfrac{}{B \mid \quad \vdash B}\ \mathsf{ax}}{- \mid B \vdash B}\ \mathsf{pass}}{A \mid B \vdash A \otimes B}\ \otimes\mathsf{R}}{A \otimes B \mid \quad \vdash A \otimes B}\ \otimes\mathsf{L}$$

$$\frac{}{A \multimap B \mid \quad \vdash A \multimap B}\ \mathsf{ax} \quad \triangleq \quad \frac{\dfrac{\dfrac{}{A \mid \quad \vdash A}\ \mathsf{ax}}{- \mid A \vdash A}\ \mathsf{pass} \quad \dfrac{}{B \mid \quad \vdash B}\ \mathsf{ax}}{A \multimap B \mid A \vdash B}\ \multimap\mathsf{L}}{A \multimap B \mid \quad \vdash A \multimap B}\ \multimap\mathsf{R}$$

Fig. 2 Equivalence of derivations: η-conversions

$$\frac{\overset{f}{S \mid \Gamma_0 \vdash A} \qquad \dfrac{\overset{g}{A \mid \Gamma_1 \vdash B} \quad \overset{h}{B \mid \Gamma_2 \vdash C}}{A \mid \Gamma_1, \Gamma_2 \vdash C}\ \mathsf{scut}}{S \mid \Gamma_0, \Gamma_1, \Gamma_2 \vdash C}\ \mathsf{scut}$$

$$= \frac{\dfrac{\overset{f}{S \mid \Gamma_0 \vdash A} \quad \overset{g}{A \mid \Gamma_1 \vdash B}}{S \mid \Gamma_0, \Gamma_1 \vdash B}\ \mathsf{scut} \qquad \overset{h}{B \mid \Gamma_2 \vdash C}}{S \mid \Gamma_0, \Gamma_1, \Gamma_2 \vdash C}\ \mathsf{scut}$$

$$\frac{\overset{f}{- \mid \Gamma_1 \vdash A} \qquad \dfrac{\overset{g}{S \mid \Gamma_0, A, \Gamma_2 \vdash B} \quad \overset{h}{B \mid \Gamma_3 \vdash C}}{S \mid \Gamma_0, A, \Gamma_2, \Gamma_3 \vdash C}\ \mathsf{scut}}{S \mid \Gamma_0, \Gamma_1, \Gamma_2, \Gamma_3 \vdash C}\ \mathsf{ccut}$$

$$= \frac{\dfrac{\overset{f}{- \mid \Gamma_1 \vdash A} \quad \overset{g}{S \mid \Gamma_0, A, \Gamma_2 \vdash B}}{S \mid \Gamma_0, \Gamma_1, \Gamma_2 \vdash B}\ \mathsf{ccut} \qquad \overset{h}{B \mid \Gamma_3 \vdash C}}{S \mid \Gamma_0, \Gamma_1, \Gamma_2, \Gamma_3 \vdash C}\ \mathsf{scut}$$

$$\frac{\overset{f}{- \mid \Gamma_2 \vdash A} \qquad \dfrac{\overset{g}{- \mid \Gamma_1, A, \Gamma_3 \vdash B} \quad \overset{h}{S \mid \Gamma_0, B, \Gamma_4 \vdash C}}{S \mid \Gamma_0, \Gamma_1, A, \Gamma_3, \Gamma_4 \vdash C}\ \mathsf{ccut}}{S \mid \Gamma_0, \Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4 \vdash C}\ \mathsf{ccut}$$

$$= \frac{\dfrac{\overset{f}{- \mid \Gamma_2 \vdash A} \quad \overset{g}{- \mid \Gamma_1, A, \Gamma_3 \vdash B}}{S \mid \Gamma_1, \Gamma_2, \Gamma_3 \vdash B}\ \mathsf{ccut} \qquad \overset{h}{S \mid \Gamma_0, B, \Gamma_4 \vdash C}}{S \mid \Gamma_0, \Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4 \vdash C}\ \mathsf{ccut}$$

Fig. 3 Associativity of cut

$$\dfrac{\overset{f}{S \mid \Gamma_0 \vdash A} \quad \dfrac{\overset{g}{- \mid \Gamma_2 \vdash B} \quad \overset{h}{A \mid \Gamma_1, B, \Gamma_3 \vdash C}}{A \mid \Gamma_1, \Gamma_2, \Gamma_3 \vdash C}\ \mathsf{scut}}{S \mid \Gamma_0, \Gamma_1, \Gamma_2, \Gamma_3 \vdash C}\ \mathsf{ccut}$$

$$= \dfrac{\overset{g}{- \mid \Gamma_2 \vdash B} \quad \dfrac{\overset{f}{S \mid \Gamma_0 \vdash A} \quad \overset{h}{A \mid \Gamma_1, B, \Gamma_3 \vdash C}}{S \mid \Gamma_0, \Gamma_1, B, \Gamma_3 \vdash C}\ \mathsf{scut}}{S \mid \Gamma_0, \Gamma_1, \Gamma_2, \Gamma_3 \vdash C}\ \mathsf{ccut}$$

$$\dfrac{\overset{f}{- \mid \Gamma_1 \vdash A} \quad \dfrac{\overset{g}{- \mid \Gamma_3 \vdash B} \quad \overset{h}{S \mid \Gamma_0, A, \Gamma_2, B, \Gamma_4 \vdash C}}{S \mid \Gamma_0, A, \Gamma_2, \Gamma_3, \Gamma_4 \vdash C}\ \mathsf{ccut}}{S \mid \Gamma_0, \Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4 \vdash C}\ \mathsf{ccut}$$

$$= \dfrac{\overset{g}{- \mid \Gamma_3 \vdash B} \quad \dfrac{\overset{f}{- \mid \Gamma_1 \vdash A} \quad \overset{h}{S \mid \Gamma_0, A, \Gamma_2, B, \Gamma_4 \vdash C}}{S \mid \Gamma_0, \Gamma_1, \Gamma_2, B, \Gamma_4 \vdash C}\ \mathsf{ccut}}{S \mid \Gamma_0, \Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4 \vdash C}\ \mathsf{ccut}$$

Fig. 4 Commutativity of cut

Proposition 3. *The following equivalences of derivations involving* scut*,* $\otimes\mathsf{R}$*, and* $\multimap\mathsf{R}$ *are admissible in* `SkNMILL`*:*

$$\dfrac{\overset{f}{S \mid \Gamma \vdash A} \quad \dfrac{\overset{g}{A \mid \Delta \vdash B} \quad \overset{h}{- \mid \Lambda \vdash C}}{A \mid \Delta, \Lambda \vdash B \otimes C}\ \otimes\mathsf{R}}{S \mid \Gamma, \Delta, \Lambda \vdash B \otimes C}\ \mathsf{scut} \quad \triangleq \quad \dfrac{\dfrac{\overset{f}{S \mid \Gamma \vdash A} \quad \overset{g}{A \mid \Delta \vdash B}}{S \mid \Gamma, \Delta \vdash B}\ \mathsf{scut} \quad \overset{h}{- \mid \Lambda \vdash C}}{S \mid \Gamma, \Delta, \Lambda \vdash B \otimes C}\ \otimes\mathsf{R}$$

$$\dfrac{\overset{f}{S \mid \Gamma \vdash B} \quad \dfrac{\overset{g}{A \mid \Delta, B \vdash C}}{A \mid \Delta \vdash B \multimap C}\ \multimap\mathsf{R}}{S \mid \Gamma, \Delta \vdash B \multimap C}\ \mathsf{scut} \quad \triangleq \quad \dfrac{\overset{f}{S \mid \Gamma \vdash A} \quad \overset{g}{A \mid \Delta, B \vdash C}}{\dfrac{S \mid \Gamma, \Delta, B \vdash C}{S \mid \Gamma, \Delta \vdash B \multimap C}\ \multimap\mathsf{R}}\ \mathsf{scut}$$

Proof. Both equivalences are proved by structural induction on the derivation f. □

4 Failure of Maehara Interpolation

The goal of this paper is proving that the logic `SkNMILL` satisfies the Craig interpolation property. But, as already mentioned in the introductive section, we cannot follow the same proof strategy used in the associative Lambek calculus, where Craig interpolation follows as a corollary of Maehara interpolation. In fact, the sequent calculus of `SkNMILL` does not satisfy Maehara interpolation. Let us see why.

First, in analogy with the presence of two admissible cut rules (3), there are also two different forms of interpolation. This is because the subsequence of the antecedents

for which we wish to find an interpolant can either contain the stoup or it can be fully included in the context. More explicitly, given an antecedent $S \mid \Gamma$, we can either: (i) split the context $\Gamma = \Gamma_1, \Gamma_2$ in two parts and look for an interpolant of the sub-antecedent $S \mid \Gamma_1$, or (ii) split the context $\Gamma = \Gamma_0, \Gamma_1, \Gamma_2$ in three parts and look for an interpolant of the sub-context Γ_1. The Maehara interpolation property in `SkNMILL` would then consist of two statements, a *stoup Maehara interpolation* (sMIP) and a *context Maehara interpolation* (cMIP):

(sMIP) Given $f : S \mid \Gamma \vdash C$ and a partition $\langle \Gamma_0, \Gamma_1 \rangle$ of Γ, there exists
- an interpolant formula D,
- a derivation $g : S \mid \Gamma_0 \vdash D$,
- a derivation $h : D \mid \Gamma_1 \vdash C$, such that
- $\sigma_X(D) \leq \sigma_X(S, \Gamma_0)$ and $\sigma_X(D) \leq \sigma_X(\Gamma_1, C)$ for all atomic formulae X.

(cMIP) Given $f : S \mid \Gamma \vdash C$ and a partition $\langle \Gamma_0, \Gamma_1, \Gamma_2 \rangle$ of Γ, there exists
- an interpolant formula D,
- a derivation $g : - \mid \Gamma_1 \vdash D$,
- a derivation $h : S \mid \Gamma_0, D, \Gamma_2 \vdash C$, such that
- $\sigma_X(D) \leq \sigma_X(\Gamma_1)$ and $\sigma_X(D) \leq \sigma_X(S, \Gamma_0, \Gamma_2, C)$.

However, the second property cMIP is not provable. An attempt to prove it would proceed by induction on the height of the derivation $f : S \mid \Gamma \vdash C$ and then inspecting what is the last rule applied in f. The rules $\otimes\mathsf{R}$ and $\multimap\mathsf{L}$ split the context, so one should be careful to consider all possible ways in which these splittings relate to the given partition $\langle \Gamma_0, \Gamma_1, \Gamma_2 \rangle$ of Γ.

The critical case is $f = \otimes\mathsf{R}(f', f'')$ with the partition $\langle \Gamma_0, (\Gamma_1', \Gamma_1''), \Gamma_2 \rangle$ and two derivations $f' : S \mid \Gamma_0, \Gamma_1' \vdash A$ and $f'' : - \mid \Gamma_1'', \Gamma_2 \vdash B$. So this is the case when the $\otimes\mathsf{R}$ rule splits Γ_1 in two parts Γ_1', Γ_1''. By inductive hypothesis on f' and the partition $\langle \Gamma_0, \Gamma_1', [\,] \rangle$, we would be given a formula D and derivations $g' : - \mid \Gamma_1' \vdash D$ and $h' : S \mid \Gamma_0, D \vdash A$. By inductive hypothesis on f'' and the partition $\langle [\,], \Gamma_1'', \Gamma_2 \rangle$, we would be given a formula E and derivations $g'' : - \mid \Gamma_1'' \vdash E$ and $h'' : - \mid E, \Gamma_2 \vdash B$. We obtain $\otimes\mathsf{R}(g', g'') : - \mid \Gamma_1', \Gamma_1'' \vdash D \otimes E$, but we are unable to construct the other desired proof of sequent $S \mid \Gamma_0, D \otimes E, \Gamma_1 \vdash A \otimes B$. We get very close via $\otimes\mathsf{R}(h', h'') : S \mid \Gamma_0, D, E, \Gamma_1 \vdash A \otimes B$, but we are unable to merge D and E into $D \otimes E$, since in our calculus the $\otimes\mathsf{L}$ cannot be applied on formulae in context.

Counterexample 4. *For a simple concrete counterexample, consider the derivation*

$$\dfrac{\dfrac{\dfrac{}{X \mid \vdash X}\,\mathsf{ax} \qquad \dfrac{\dfrac{}{Y \mid \vdash Y}\,\mathsf{ax}}{- \mid Y \vdash Y}\,\mathsf{pass}}{X \mid Y \vdash X \otimes Y}\,\otimes\mathsf{R} \qquad \dfrac{\dfrac{}{Z \mid \vdash Z}\,\mathsf{ax}}{- \mid Z \vdash Z}\,\mathsf{pass}}{X \mid Y, Z \vdash (X \otimes Y) \otimes Z}\,\otimes\mathsf{R}$$

and the partition $\langle [\,], [Y, Z], [\,] \rangle$. Suppose by contradiction that Maehara interpolation holds, so we would have a formula D and two derivations $g : - \mid Y, Z \vdash D$ and $h : X \mid D \vdash (X \otimes Y) \otimes Z$. The variable multiplicity condition of Maehara interpolation and the existence of the derivation g ensure that D does not contain atomic formulae other than Y and Z, and the latter must have a unique occurrence in D. Nevertheless, the existence of derivation h is absurd. Since X is atomic, h can only be of the form:

(i) $f = \otimes\mathsf{R}(f_1, f_2)$ *for some derivations* $f_1 : X \mid D \vdash X \otimes Y$ *and* $f_2 : - \mid \ \vdash Z$*, or (ii)* $f = \otimes\mathsf{R}(f_1', f_2')$ *for some derivations* $f_1' : X \mid \ \vdash X \otimes Y$ *and* $f_2' : - \mid D \vdash Z$*. Case (i) is impossible since there is no such* f_2*, while case (ii) is impossible since there is no such* f_1'*.*

This situation is reminiscent of the failure of interpolation in the product-free Lambek calculus [12] and in the implicational fragment of intuitionistic logic [13]. In both cases, Maehara interpolation fails because neither the additive ($\wedge$) nor the multiplicative ($\otimes$) conjunction is present. A concrete counterexample in the product-free Lambek calculus (adapted from Kanazawa [13]) is given by the derivable sequent $W, W\backslash Y, W, W\backslash X, X\backslash(Y\backslash Z) \ \vdash \ Z$ with the partition $\langle[\,], [W, W\backslash Y, W, W\backslash X], [X\backslash(Y\backslash Z)]\rangle$. This can be shown to not satisfy the Maehara interpolation property. In the presence of $\otimes$, Maehara's method would produce the interpolant formula $X \otimes Y$. The situation of `SkNMILL` is somewhere in-between: we have a multiplicative conjunction $\otimes$ but we cannot do much with it if a formula $A \otimes B$ is in context instead of the stoup position, since the rule $\otimes\mathsf{L}$ cannot be applied arbitrarily in the antecedent. The counterexample to MIP in product-free Lambek calculus, when appropriately modified, also works as a counterexample to cMIP in `SkNMILL`: consider the derivable sequent $X \multimap (Y \multimap Z) \mid W \multimap X, W, W \multimap Y, W \vdash Z$ with the partition $\langle[\,], [W \multimap X, W, W \multimap Y, W], [\,]\rangle$.

5 Craig Interpolation for `SkNMILL`

In this section, we show that `SkNMILL` enjoys Craig interpolation, even though it does not generally enjoy Maehara interpolation. This is again in analogy with the product-free Lambek calculus. As mentioned in the introductive section, Pentus [12] proved that the latter satisfies a relaxation of Maehara interpolation, that we dubbed Maehara multi-interpolation (MMIP), which is sufficient to show Craig interpolation. Here is a brief description of how this works. Suppose the formula $A\backslash B$ is provable in product-free Lambek calculus (with non-empty antecedents, as in Pentus' case). This implies that there exists a derivation $f : A \vdash B$. Apply the Maehara multi-interpolation procedure to f and the partition $\langle[\,], [A], [\,]\rangle$. This produces a partition $\langle\Delta_1, \ldots, \Delta_n\rangle$ of $[A]$. Since $[A]$ is a singleton list and antecedents cannot be empty, it follows that $n = 1$ and $\Delta_1 = [A]$. Maehara multi-interpolation then produces a formula D_1 and two derivations $h : A \vdash D_1$ and $g : D_1 \vdash B$ with D_1 satisfying the appropriate variable condition, i.e. D_1 is a Craig interpolant of A and B.

We showed in the previous section that `SkNMILL` does not satisfy the context Maehara interpolation property (cMIP). We prove now that instead it satisfies a *context Maehara multi-interpolation property* (cMMIP). And the stoup Maehara interpolation property (sMIP) also holds.

Theorem 5. *In the sequent calculus for* `SkNMILL`*, the following two interpolation properties hold:*

(sMIP) *Given a derivation* $f : S \mid \Gamma \vdash C$ *and a partition* $\langle\Gamma_0, \Gamma_1\rangle$ *of* Γ*, there exist*
- *an interpolant formula* D*,*
- *a derivation* $g : S \mid \Gamma_0 \vdash D$*,*
- *a derivation* $h : D \mid \Gamma_1 \vdash C$*, such that*

– $\sigma_X(D) \leq \sigma_X(S, \Gamma_0)$ *and* $\sigma_X(D) \leq \sigma_X(\Gamma_1, C)$ *for all atomic formulae* X.

(cMMIP) *Given a derivation* $f : S \mid \Gamma \vdash C$ *and a partition* $\langle \Gamma_0, \Gamma_1, \Gamma_2 \rangle$ *of* Γ*, there exist*
– *a partition* $\langle \Delta_1, \ldots, \Delta_n \rangle$ *of* Γ_1*,*
– *a list of interpolant formulae* $D_1, \ldots, D_n$*,*
– $g_i : - \mid \Delta_i \vdash D_i$ *for all* $i \in [1, \ldots, n]$*,*
– $h : S \mid \Gamma_0, D_1, \ldots, D_n, \Gamma_2 \vdash C$*, such that*
– $\sigma_X(D_1, \ldots, D_n) \leq \sigma_X(\Gamma_1)$ *and* $\sigma_X(D_1, \ldots, D_n) \leq \sigma_X(S, \Gamma_0, \Gamma_2, C)$ *for all atomic formulae* X*.*

These two statements of the theorem are proved mutually by structural induction on derivations. We separate the proofs for readability.

Proof of sMIP. We proceed by induction on the structure of f.

Case $f = \mathsf{ax}$. Suppose $f = \mathsf{ax} : A \mid \ \vdash A$, which forces $\Gamma_0 = \Gamma_1 = [\,]$. In this case, the interpolant formula is A and $g = h = \mathsf{ax} : A \mid \ \vdash A$, where the variable multiplicity condition is automatically satisfied.

Case $f = \mathsf{IR}$. Since $f : - \mid \ \vdash \mathsf{I}$, this again forces Γ_0 and Γ_1 to be empty lists. In this case, the interpolant formula is I and $g = \mathsf{IR} : - \mid \ \vdash \mathsf{I}$ and $h = \mathsf{IL}(\mathsf{IR}) : \mathsf{I} \mid \ \vdash \mathsf{I}$, where the variable multiplicity condition is vacuously satisfied.

Case $f = \mathsf{IL}\ f'$. Given a derivation $f' : - \mid \Gamma \vdash C$, by inductive hypothesis on f' with the same partition $\langle \Gamma_0, \Gamma_1 \rangle$ of Γ we obtain
– a formula D,
– a derivation $g' : - \mid \Gamma_0 \vdash D$,
– a derivation $h' : D \mid \Gamma_1 \vdash C$, such that
– $\sigma_X(D) \leq \sigma_X(\Gamma_0)$ and $\sigma_X(\Gamma_1, C)$ for all atomic formulae X.

In this case, the interpolant formula for f is D and the two desired derivations are $g = \mathsf{IL}\ g'$ and $h = h'$. The variable multiplicity condition is automatically satisfied.

Cases $f = \otimes\mathsf{L}\ f'$ and $f = \multimap\mathsf{R}\ f'$. Analogous to the previous case.

Case $f = \mathsf{pass}\ f'$. Let $f' : A \mid \Gamma' \vdash C$ and $\Gamma = A, \Gamma'$. There are two subcases determined by the partition $\langle \Gamma_0, \Gamma_1 \rangle$ of Γ. Specifically, either Γ_0 is empty or not.

- If $\Gamma_0 = [\,]$, then the interpolant is I and the two desired derivations are IR and $\mathsf{IL}(\mathsf{pass}\ f')$. The variable multiplicity condition is satisfied because $\sigma_X(\mathsf{I}) = 0$.
- If $\Gamma_0 = A, \Gamma_0'$, then by inductive hypothesis on f' with the partition $\langle \Gamma_0', \Gamma_1 \rangle$ we obtain
  – a formula D,
  – a derivation $g' : A \mid \Gamma_0' \vdash D$,
  – a derivation $h' : D \mid \Gamma_1 \vdash C$, such that
  – $\sigma_X(D) \leq \sigma_X(A, \Gamma_0')$ and $\sigma_X(D) \leq \sigma_X(\Gamma_1, C)$ for all atomic formulae X.

  In this case, the interpolant formula for f is D, and the two desired derivations are $g = \mathsf{pass}\ g'$ and $h = h'$. The variable multiplicity condition follows directly from the inductive hypothesis.

Case $f = \otimes\mathsf{R}(f', f'')$. Let $f' : S \mid \Lambda \vdash A$ and $f'' : - \mid \Omega \vdash B$, so that $\Gamma = \Lambda, \Omega$. We need to check how the latter splitting of Γ compares to the given partition $\langle \Gamma_0, \Gamma_1 \rangle$. There are two possibilities:

- Γ_0 is fully contained in Λ. This means that $\Lambda = \Gamma_0, \Gamma_1'$ and $\Gamma_1 = \Gamma_1', \Omega$. Then $f' : S \mid \Gamma_0, \Gamma_1' \vdash A$ and $f'' : - \mid \Omega \vdash B$. In this case, by inductive hypothesis on f' with the partition $\langle \Gamma_0, \Gamma_1' \rangle$ we obtain

– a formula D,
– a derivation $g' : S \mid \Gamma_0 \vdash D$,
– a derivation $h' : D \mid \Gamma_1' \vdash A$ such that
– $\sigma_X(D) \leq \sigma_X(S, \Gamma_0)$ and $\sigma_X(D) \leq \sigma_X(\Gamma_1', A)$ for all atomic formulae X.

The desired interpolant formula is D and the desired derivations are $g = g'$ and $h = \otimes\mathsf{R}(h', f'') : D \mid \Gamma_1', \Omega \vdash A \otimes B$. The variable multiplicity condition is satisfied because $\sigma_X(D) \leq \sigma_X(\Gamma_1', A) \leq \sigma_X(\Gamma_1', \Omega, A \otimes B)$ for all atomic formulae X.

- Γ_0 splits between Λ and Ω. This means that $\Gamma_0 = \Lambda, \Gamma_0'$ and $\Omega = \Gamma_0', \Gamma_1$, and Γ_0' is non-empty. Then $f' : S \mid \Lambda \vdash A$ and $f'' : - \mid \Gamma_0', \Gamma_1 \vdash B$. In this case, by inductive hypothesis on f' with the partition $\langle \Lambda, [\,] \rangle$ and on f'' with the partition $\langle \Gamma_0', \Gamma_1 \rangle$, respectively, we obtain
  – formulae E and F,
  – derivations $g' : S \mid \Lambda \vdash E$ and $g'' : - \mid \Gamma_0' \vdash F$,
  – derivations $h' : E \mid \ \vdash A$ and $h'' : F \mid \Gamma_1 \vdash B$, such that
  – $\sigma_X(E) \leq \sigma_X(S, \Lambda)$ and $\sigma_X(E) \leq \sigma_X(A)$, and
  – $\sigma_X(F) \leq \sigma_X(\Gamma_0')$ and $\sigma_X(F) \leq \sigma_X(\Gamma_1, B)$ for all atomic formulae X.

  The desired interpolant formula is $D = E \otimes F$ and the desired derivations are

$$g = \dfrac{\overset{g'}{S \mid \Lambda \vdash E} \quad \overset{g''}{- \mid \Gamma_0' \vdash F}}{S \mid \Lambda, \Gamma_0' \vdash E \otimes F}\otimes\mathsf{R} \qquad h = \dfrac{\dfrac{\overset{h'}{E \mid \ \vdash A} \quad \dfrac{\overset{h''}{F \mid \Gamma_1 \vdash B}}{- \mid F, \Gamma_1 \vdash B}\,\mathsf{pass}}{E \mid F, \Gamma_1 \vdash A \otimes B}\otimes\mathsf{R}}{E \otimes F \mid \Gamma_1 \vdash A \otimes B}\otimes\mathsf{L}$$

  The variable multiplicity condition is satisfied because $\sigma_X(E \otimes F) = \sigma_X(E) + \sigma_X(F) \leq \sigma_X(S, \Lambda) + \sigma_X(\Gamma_0') = \sigma_X(S, \Lambda, \Gamma_0')$ and $\sigma_X(E \otimes F) = \sigma_X(E) + \sigma_X(F) \leq \sigma_X(A) + \sigma_X(\Gamma_1, B) = \sigma_X(A, \Gamma_1, B) = \sigma_X(\Gamma_1, A \otimes B)$.

Case $f = \multimap\mathsf{L}(f', f'')$. Let $f' : - \mid \Lambda \vdash A$ and $f'' : B \mid \Omega \vdash C$, so that $\Gamma = \Lambda, \Omega$. Again we check how the latter splitting of Γ compares to the given partition $\langle \Gamma_0, \Gamma_1 \rangle$. There are two possibilities:

- Γ_1 is fully contained in Ω. This means that $\Gamma_0 = \Lambda, \Gamma_0'$ and $\Omega = \Gamma_0', \Gamma_1$. Then $f' : - \mid \Lambda \vdash A$ and $f'' : B \mid \Gamma_0', \Gamma_1 \vdash C$. In this case, by inductive hypothesis on f'' with the partition $\langle \Gamma_0', \Gamma_1 \rangle$ we obtain
  – a formula D,
  – a derivation $g'' : B \mid \Gamma_0' \vdash D$,
  – a derivation $h'' : D \mid \Gamma_1 \vdash C$ such that
  – $\sigma_X(D) \leq \sigma_X(B, \Gamma_0')$ and $\sigma_X(D) \leq \sigma_X(\Gamma_1, C)$ for all atomic formulae X.

  The desired interpolant formula is D and the desired derivations are $g = \multimap\mathsf{L}(f', g'') : A \multimap B \mid \Lambda, \Gamma_0' \vdash D$ and $h = h''$. The variable multiplicity condition is satisfied because $\sigma_X(D) \leq \sigma_X(B, \Gamma_0') \leq \sigma_X(A \multimap B, \Lambda, \Gamma_0')$.
- Γ_1 splits between Λ and Ω. This means that $\Lambda = \Gamma_0, \Gamma_1'$ and $\Gamma_1 = \Gamma_1', \Omega$, and Γ_1' is non-empty. Then $f' : - \mid \Gamma_0, \Gamma_1' \vdash A$ and $f'' : B \mid \Omega \vdash C$. Our goal is to find a formula D and derivations $g : A \multimap B \mid \Gamma_0 \vdash D$ and $h : D \mid \Gamma_1', \Omega \vdash C$. by inductive hypothesis on f'' with the partition $\langle [\,], \Omega \rangle$ we obtain
  – a formula E,

– a derivation $g'' : B \mid \ \vdash E$,
– a derivation $h'' : E \mid \Omega \vdash C$ such that
– $\sigma_X(E) \leq \sigma_X(B)$ and $\sigma_X(E) \leq \sigma_X(\Omega, C)$ for all atomic formulae X.

We also apply the cMMIP procedure (which, remember, is proved by mutual induction with sMIP) on the derivation f' with the partition $\langle \Gamma_0, \Gamma_1', [\,] \rangle$ and obtain

– a partition $\langle \Delta_1, \ldots, \Delta_n \rangle$ of Γ_1',
– a list of formulae $D_1, \ldots, D_n$,
– a list of derivations $g_i' : - \mid \Delta_i \vdash D_i$, for $i \in [1, \ldots, n]$,
– a derivation $h' : - \mid \Gamma_0, D_1, \ldots, D_n \vdash A$, such that
– $\sigma_X(D_1, \ldots, D_n) \leq \sigma_X(\Gamma_1')$ and $\sigma_X(D_1, \ldots, D_n) \leq \sigma_X(\Gamma_0, A)$ for all atomic formulae X.

The desired interpolant formula is $D = D_1 \multimap (D_2 \multimap (\ldots (D_n \multimap E) \ldots))$. The desired derivations g and h are constructed as follows:

$$g = \dfrac{\dfrac{\overset{h'}{- \mid \Gamma_0, D_1, \ldots, D_n \vdash A} \quad \overset{g''}{B \mid \ \vdash E}}{A \multimap B \mid \Gamma_0, D_1, \ldots, D_n \vdash E}\ \multimap\mathsf{L}}{A \multimap B \mid \Gamma_0 \vdash D_1 \multimap (\ldots (D_n \multimap E) \ldots)}\ \multimap\mathsf{R}^*$$

$$h = \dfrac{\overset{[g_i']}{[- \mid \Delta_i \vdash D_i]_i} \quad \overset{h''}{E \mid \Omega \vdash C}}{D_1 \multimap (\ldots (D_n \multimap E) \ldots) \mid \Delta_1, \ldots, \Delta_n, \Omega \vdash C}\ \multimap\mathsf{L}^*$$

Notice that $\Gamma_1' = \Delta_1, \ldots, \Delta_n$, so the variable multiplicity condition is easy to check.

□

Proof of cMMIP. We proceed by induction on the structure of f.
Case $f = \mathsf{ax}$. Suppose $f = \mathsf{ax} : A \mid \ \vdash A$, which means that $\Gamma_0 = \Gamma_1 = \Gamma_2 = [\,]$. In this case, the desired partition of Γ_1 is the empty one, i.e. $n = 0$. The desired lists of formulae D_i and of derivations g_i are also empty. The desired derivation h is ax.
Case $f = \mathsf{IR}$. Similar to the previous one.
Case $f = \mathsf{IL}\ f'$. Given a derivation $f' : - \mid \Gamma \vdash C$, by inductive hypothesis on f' with the same partition $\langle \Gamma_0, \Gamma_1, \Gamma_2 \rangle$ of Γ we obtain

– a partition $\langle \Delta_0, \ldots, \Delta_n \rangle$ of Γ_1,
– a list of interpolant formulae $D_1, \ldots, D_n$,
– derivations $g_i' : - \mid \Delta_i \vdash D_i$, for $i \in [1, \ldots, n]$,
– a derivation $h' : - \mid \Gamma_0, D_1, \ldots, D_n, \Gamma_2 \vdash C$, such that
– $\sigma_X(D_1, \ldots, D_n) \leq \sigma_X(\Gamma_1)$ and $\sigma_X(D_1, \ldots, D_n) \leq \sigma_X(\Gamma_0, \Gamma_2, C)$ for all X.

The desired partition of Γ_1 is $\langle \Delta_0, \ldots, \Delta_n \rangle$, the desired list of interpolant formulae is $D_1, \ldots, D_n$. The desired derivations are $g_i = g_i'$ for $i \in [1, \ldots, n]$ and $h = \mathsf{IL}\ h'$. The variable multiplicity condition is automatically satisfied.
Cases $f = \otimes\mathsf{L}\ f'$ and $f = \multimap\mathsf{R}\ f'$. Analogous to the previous case.
Case $f = \mathsf{pass}\ f'$. Let $f' : A \mid \Gamma' \vdash C$ and $\Gamma = A, \Gamma'$. There are subcases determined by

the partition $\langle \Gamma_0, \Gamma_1, \Gamma_2 \rangle$ of Γ. The most interesting case is the one where $\Gamma_0 = [\,]$ and $\Gamma_1 = A, \Gamma_1'$, so that $\Gamma' = \Gamma_1', \Gamma_2$. The other possible cases are handled similarly to the IL case discussed above. We apply the sMIP procedure (which, remember, is proved by mutual induction with cMMIP) on the derivation f' and the partition $\langle \Gamma_1', \Gamma_2 \rangle$, which gives us

- a formula D,
- a derivation $g' : A \mid \Gamma_1' \vdash D$,
- a derivation $h' : D \mid \Gamma_2 \vdash C$, such that
- $\sigma_X(D) \leq \sigma_X(A, \Gamma_1')$ and $\sigma_X(D) \leq \sigma_X(\Gamma_2, C)$ for any X.

The desired partition of A, Γ_1' is the singleton context $[A, \Gamma_1']$, i.e. $n = 1$. The desired list of interpolant formulae is the singleton $[D]$. The desired list of derivations g_i is the singleton list consisting only of $\mathsf{pass}\ g' : - \mid A, \Gamma_1' \vdash D$ and the desired derivation h is $\mathsf{pass}\ h' : - \mid D, \Gamma_2 \vdash C$. The variable multiplicity condition follows from the inductive hypothesis.

Case $f = \otimes\mathsf{R}(f', f'')$. Let $f' : S \mid \Lambda \vdash A$ and $f'' : - \mid \Omega \vdash B$, so that $\Gamma = \Lambda, \Omega$. We need to check how the latter splitting of Γ compares to the given partition $\langle \Gamma_0, \Gamma_1, \Gamma_2 \rangle$. There are three possibilities:

- Γ_1 is fully contained in Ω. This means that $\Gamma_0 = \Lambda, \Gamma_0'$ and $\Omega = \Gamma_0', \Gamma_1, \Gamma_2$. Then $f' : S \mid \Lambda \vdash A$ and $f'' : - \mid \Gamma_0', \Gamma_1, \Gamma_2 \vdash B$. by inductive hypothesis on f'' with partition $\langle \Gamma_0', \Gamma_1, \Gamma_2 \rangle$ we obtain
  - a partition $\langle \Delta_0, \ldots, \Delta_n \rangle$ of Γ_1,
  - a list of interpolant formulae $D_1, \ldots, D_n$,
  - derivations $g_i'' : - \mid \Delta_i \vdash D_i$, for $i \in [1, \ldots, n]$,
  - a derivation $h'' : - \mid \Gamma_0', D_1, \ldots, D_n, \Gamma_2 \vdash B$, such that
  - $\sigma_X(D_1, \ldots, D_n) \leq \sigma_X(\Gamma_1)$ and $\sigma_X(D_1, \ldots, D_n) \leq \sigma_X(\Gamma_0', \Gamma_2, B)$ for any X.

  The desired partition of Γ_1 is $\langle \Delta_0, \ldots, \Delta_n \rangle$. The desired list of interpolant formulae is $D_1, \ldots, D_n$. The desired derivation g_i is g_i'' for $i \in [1, \ldots, n]$ and the desired derivation h is $\otimes\mathsf{R}(f', h'')$. The variable multiplicity condition is satisfied because $\sigma_X(D_1, \ldots, D_n) \leq \sigma_X(\Gamma_0', \Gamma_2, B) \leq \sigma_X(S, \Lambda, \Gamma_0', \Gamma_2, A \otimes B)$ for any X.
- Γ_1 is fully contained in Λ. This case is analogous to the one above, but now we have to use the inductive hypothesis on the derivation f' instead of f''.
- Γ_1 splits between Λ and Ω. This means that $\Gamma_1 = \Gamma_1', \Gamma_1''$ and $\Lambda = \Gamma_0, \Gamma_1'$ and $\Omega = \Gamma_1'', \Gamma_2$, and Γ_1'' is non-empty. Then $f' : S \mid \Gamma_0, \Gamma_1' \vdash A$ and $f'' : - \mid \Gamma_1'', \Gamma_2 \vdash B$. by inductive hypothesis on f' with the partition $\langle \Gamma_0, \Gamma_1', [\,] \rangle$ and on f'' with the partition $\langle [\,], \Gamma_1'', \Gamma_2 \rangle$, respectively, we obtain
  - a partition $\langle \Delta_0, \ldots, \Delta_n \rangle$ of Γ_1' and a partition $\langle \Delta_{n+1}, \ldots, \Delta_m \rangle$ of Γ_1'',
  - two lists of interpolant formulae $D_1, \ldots, D_n$ and $D_{n+1}, \ldots, D_m$,
  - derivations $g_i' : - \mid \Delta_i \vdash D_i$, for $i \in [1, \ldots, n]$, and derivations $g_j' : - \mid \Delta_j \vdash D_j$, for $j \in [n+1, \ldots, m]$,
  - derivations $h' : S \mid \Gamma_0, D_1, \ldots, D_n \vdash A$ and $h'' : - \mid D_{n+1}, \ldots, D_m, \Gamma_2 \vdash B$, such that
  - $\sigma_X(D_1, \ldots, D_n) \leq \sigma_X(\Gamma_1')$ and $\sigma_X(D_1, \ldots, D_n) \leq \sigma_X(S, \Gamma_0, A)$, and
  - $\sigma_X(D_{n+1}, \ldots, D_m) \leq \sigma_X(\Gamma_1'')$ and $\sigma_X(D_{n+1}, \ldots, D_m) \leq \sigma_X(\Gamma_2, B)$ for any X.

The desired partition of Γ_1', Γ_1'' is $\langle \Delta_0, \ldots, \Delta_n, \Delta_{n+1}, \ldots, \Delta_m \rangle$. The desired list of interpolant formulae is $D_1, \ldots, D_n, D_{n+1}, \ldots, D_m$. The desired derivation g is

$$\dfrac{S \mid \Gamma_0, D_1, \ldots, D_n \vdash A \quad (h') \qquad - \mid D_{n+1}, \ldots, D_m, \Gamma_2 \vdash B \quad (h'')}{S \mid \Gamma_0, D_1, \ldots, D_n, D_{n+1}, \ldots, D_m, \Gamma_2 \vdash A \otimes B}\ \otimes\mathsf{R}$$

while the desired derivation g_i is g_i' for $i \in [1, \ldots, m]$. For the variable multiplicity condition, we have $\sigma_X(D_1, \ldots, D_m) \leq \sigma_X(S, \Gamma_0, A, \Gamma_2, B) = \sigma_X(S, \Gamma_0, \Gamma_2, A \otimes B)$ for any X.

Case $f = \multimap\mathsf{L}(f', f'')$. Analogous to the case of $\otimes\mathsf{R}$ above. □

Notice that cMMIP is invoked in the proof of sMIP, in the case $f = \multimap\mathsf{L}(f', f'')$, and sMIP is invoked in the proof of cMMIP, in the case $f = \mathsf{pass}\ f'$. The proof of Theorem 5 describes an effective procedure for building interpolant formulae and derivations. This procedure is terminating, since each recursive call happens on a derivation with height strictly smaller than the one of the derivation in input. This behaviour is further confirmed in our Agda formalization, where the inductive proof of sMIP/cMMIP is accepted by the proof assistant as terminating.

In his PhD thesis [27], Roorda considered a stronger variable multiplicity condition, characterizing the *positive and negative* occurrences of atomic formulae in interpolants. Let $\mathsf{At}^+_X(A)$ be the multiset of positive occurrences of the atomic formula X in A, and let $\mathsf{At}^-_X(A)$ be the multiset of negative occurrences of the atomic formula X in A. These notions can be straightforwardly extended to stoups and contexts. We can prove that the sMIP procedure satisfies the following stronger variable condition: given $p \in \{+, -\}$ and any atomic formula X, there is an injective function from $\mathsf{At}^p_X(D)$ to $\mathsf{At}^p_X(S, \Gamma_0)$, and an injective function from $\mathsf{At}^p_X(D)$ to the multiset union of $\mathsf{At}^{\neg p}_X(\Gamma_1)$ and $\mathsf{At}^p_X(C)$. Here $\neg p$ denotes the opposite polarity of p, i.e. $\neg + = -$ and $\neg - = +$. We can also prove that the cMMIP procedure satisfies the following stronger variable condition: given $p \in \{+, -\}$ and any atomic formula X, there is an injective function from $\mathsf{At}^p_X(D_1, \ldots, D_n)$ to $\mathsf{At}^p_X(\Gamma_1)$ and and an injective function from $\mathsf{At}^p_X(D_1, \ldots, D_n)$ to the multiset union of $\mathsf{At}^{\neg p}_X(S, \Gamma_0, \Gamma_2)$ and $\mathsf{At}^p_X(C)$. We refer to our Agda formalization for more details on how to establish these stronger variable conditions.

Example 6. *Let us illustrate the interpolation procedure on a simple example. We compute the stoup Maehara interpolant of the end-sequent in the derivation*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{X \mid \vdash X}\,\mathsf{ax}}{- \mid X \vdash X}\,\mathsf{pass} \quad \dfrac{\dfrac{}{Y \mid \vdash Y}\,\mathsf{ax} \quad \dfrac{\dfrac{}{W \mid \vdash W}\,\mathsf{ax}}{- \mid W \vdash W}\,\mathsf{pass}}{Y \mid W \vdash Y \otimes W}\,\otimes\mathsf{R}}{X \multimap Y \mid X, W \vdash Y \otimes W}\,\multimap\mathsf{L}}{- \mid X \multimap Y, X, W \vdash Y \otimes W}\,\mathsf{pass} \qquad \dfrac{}{Z \mid \vdash Z}\,\mathsf{ax}}{(Y \otimes W) \multimap Z \mid X \multimap Y, X, W \vdash Z}\,\multimap\mathsf{L} \tag{7}$$

with the partition $\langle [X \multimap Y], [X, W] \rangle$.

Following the procedure in the proof of Theorem 5, we are in the case when the last

rule is $\multimap\mathsf{L}$ *and both lists in the partition* $\langle [X \multimap Y], [X, W] \rangle$ *move to the context of the left premise. This means that we need to apply the* cMMIP *procedure to the derivation* $\mathsf{pass}(\multimap\mathsf{L}(\mathsf{pass\ ax}, \otimes\mathsf{R}(\mathsf{ax}, \mathsf{pass\ ax}))) : - \mid X \multimap Y, X, W \vdash Y \otimes W$ *(witnessing the left premise of* $\multimap\mathsf{L}$*) with the partition* $\langle [X \multimap Y], [X, W], [\,] \rangle$*. This produces*

- *a partition* $\langle [X], [W] \rangle$ *of* $[X, W]$*,*
- *a list of interpolant formulae* $[X, W]$*, and*
- *derivations* $g_1' = \mathsf{pass\ ax} : - \mid X \vdash X$*, and* $g_2' = \mathsf{pass\ ax} : - \mid W \vdash W$*, and* $h' = \mathsf{pass}(\multimap\mathsf{L}(\mathsf{pass\ ax}, \otimes\mathsf{R}(\mathsf{ax}, \mathsf{pass\ ax}))) : - \mid X \multimap Y, X, W \vdash Y \otimes W$

satisfying the variable multiplicity condition. Next, we need to apply the sMIP *procedure on the derivation* $\mathsf{ax} : Z \mid\ \vdash Z$ *with the partition* $\langle [\,], [\,] \rangle$ *which produces two derivations* $\mathsf{ax} : Z \mid\ \vdash Z$ *and* $\mathsf{ax} : Z \mid\ \vdash Z$*. Then we obtain the desired interpolant formula* $X \multimap (W \multimap Z)$ *and the desired derivations*

$$g \quad = \quad \dfrac{\dfrac{\dfrac{\overset{h'}{- \mid X \multimap Y, X, W \vdash Y \otimes W} \quad \dfrac{}{Z \mid\ \vdash Z}\,\mathsf{ax}}{(Y \otimes W) \multimap Z \mid X \multimap Y, X, W \vdash Z}\,\multimap\mathsf{L}}{(Y \otimes W) \multimap Z \mid X \multimap Y, X \vdash W \multimap Z}\,\multimap\mathsf{R}}{(Y \otimes W) \multimap Z \mid X \multimap Y \vdash X \multimap (W \multimap Z)}\,\multimap\mathsf{R}$$

$$h \quad = \quad \dfrac{\dfrac{g_1'}{- \mid X \vdash X} \quad \dfrac{\dfrac{\overset{g_2'}{- \mid W \vdash W} \quad \dfrac{}{Z \mid\ \vdash Z}\,\mathsf{ax}}{W \multimap Z \mid W \vdash Z}\,\multimap\mathsf{L}}{X \multimap (W \multimap Z) \mid X, W \vdash Z}\,\multimap\mathsf{L}$$

Notice that this is crucially different from the result that Maehara's method would produce on the corresponding derivation in the associative Lambek calculus (with $\otimes$*). The translation of derivation (7) in the associative Lambek calculus is*

$$\dfrac{\dfrac{\dfrac{}{X \vdash X}\,\mathsf{ax} \quad \dfrac{\dfrac{}{Y \vdash Y}\,\mathsf{ax} \quad \dfrac{}{W \vdash W}\,\mathsf{ax}}{Y, W \vdash Y \otimes W}\,\otimes\mathsf{R}}{Y/X, X, W \vdash Y \otimes W}\,/\mathsf{L} \quad \dfrac{}{Z \vdash Z}\,\mathsf{ax}}{Z/(Y \otimes W), Y/X, X, W \vdash Z}\,/\mathsf{L}$$

Using the Maehara interpolation procedure defined in [11], the resulting interpolant formula would be $Z/(X \otimes W)$*. Again,* X *and* Y *can be tensored in the latter formula since the Lambek calculus admits a general left rule for* $\otimes$*.*

We conclude this section by showing how Craig interpolation follows from stoup Maehara interpolation.

Definition 7. *A formula* A *provable in* `SkNMILL` *if and only if there is a derivation* $f : - \mid\ \vdash A$ *in* `SkNMILL`*.*

Theorem 8. *For any formulae A and C, if $A \multimap C$ is provable in* `SkNMILL`*, then there exists a formula D such that both $A \multimap D$ and $D \multimap C$ are provable, and* $\mathsf{var}(D) \subseteq \mathsf{var}(A) \cap \mathsf{var}(C)$.

Proof. Suppose that $A \multimap C$ is provable, then by Definition 7, there is a derivation $f : - \mid \ \vdash A \multimap C$. By invertibility of the rule $\multimap\mathsf{R}$, we obtain a derivation $f' : - \mid A \vdash C$. Then by running the sMIP procedure on f' with the partition $\langle [A], [\,] \rangle$, we get

- a formula D,
- $g' : - \mid A \vdash D$,
- $h' : D \mid \ \vdash C$, such that
- $\sigma_X(D) \leq \sigma_X(A)$ and $\sigma_X(D) \leq \sigma_X(C)$.

The formulae $A \multimap D$ and $D \multimap C$ are proved by the derivations $\multimap\mathsf{R}\ g' : - \mid \ \vdash A \multimap D$ and $\multimap\mathsf{R}(\mathsf{pass}\ h') : - \mid \ \vdash D \multimap C$, respectively. The variable condition is implied by the variable multiplicity condition. □

The nature of sequents in `SkNMILL` explains why Craig interpolation holds for `SkNMILL`, yet cMIP does not. The antecedent of a sequent $S \mid \Gamma \vdash C$ is not merely a list with a distinguished element at the leftmost position but carries an implicit structure, akin to a tree associated to the left. Recall Counterexample 4, the antecedent of the sequent $X \mid Y, Z \vdash (X \otimes Y) \otimes Z$ is the tree $((X, Y), Z)$. Given the partition $\langle [\,], (Y, Z), [\,] \rangle$, cMIP would require identifying a subderivation where the tree (Y, Z) appears. However, such a subderivation does not appear in the derivation for $X \mid Y, Z \vdash (X \otimes Y) \otimes Z$.

On the other hand, considering the derivation

$$\dfrac{\dfrac{}{X \mid \ \vdash X}\,\mathsf{ax} \qquad \dfrac{\dfrac{\dfrac{}{Y \mid \ \vdash Y}\,\mathsf{ax} \quad \dfrac{\dfrac{}{Z \mid \ \vdash Z}\,\mathsf{ax}}{- \mid Z \vdash Z}\,\mathsf{pass}}{Y \mid Z \vdash Y \otimes Z}\,\otimes\mathsf{R}}{- \mid Y, Z \vdash Y \otimes Z}\,\mathsf{pass}}{X \mid Y, Z \vdash X \otimes (Y \otimes Z)}\,\otimes\mathsf{R}$$

and the partition $\langle [\,], [Y, Z], [\,] \rangle$, we can find the interpolant formula $Y \otimes Z$ and two derivations $g : - \mid Y, Z \vdash Y \otimes Z$ and $h : X \mid Y \otimes Z \vdash X \otimes (Y \otimes Z)$ because the tree (Y, Z) has appeared in the derivation. Thus, while cMIP might hold in specific, carefully chosen instances depending on the full derivation context, it does not hold generally.

Therefore, in cMMIP, we relax the condition by allowing interpolants to be a list of formulae so that we can prove Craig interpolation for `SkNMILL`.

For sMIP, the interpolant tree always exists. For any derivation $f : S \mid \Gamma_0, \Gamma_1 \vdash C$ and any partition $\langle \Gamma_0, \Gamma_1 \rangle$, $((S, \Gamma_0), \Gamma_1)$ is a tree associated to the left where the interpolant tree (S, Γ_0) has appeared. For example, consider the derivation above and the partition $\langle [Y], [Z] \rangle$. The interpolant tree (X, Y) has appeared in the endsequent.

6 Proof-Relevant Interpolation

So far we have established a procedure sMIP for effectively splitting a derivation $f : S \mid \Gamma_1, \Gamma_2 \vdash C$ in two derivations $g : S \mid \Gamma_1 \vdash D$ and $h : - \mid \Gamma_2 \vdash C$, with D being "minimal" in the sense of satisfying an appropriate multiplicity condition. A natural question arises: what happens when we compose derivations g and h using the admissible scut rule? Intuition suggests that we should get back the original derivation f, at least modulo η-conversions and permutative conversions. This is in fact what happens, and this section is dedicated to proving this result.

Analogously, the cMMIP procedure splits a derivation $f : S \mid \Gamma_0, \Gamma_1, \Gamma_2 \vdash C$ in a tuple of derivations $[g_i : - \mid \Delta_i \vdash D_i]_i$ and $h : S \mid \Gamma_0, D_1, \ldots, D_n, \Gamma_2 \vdash C$, with $D_1, \ldots, D_n$ satisfying an appropriate multiplicity condition. If we compose $[g_i]$ and h using the admissible ccut^* rule, we get back the original derivation modulo $\mathring{=}$.

Similar questions have been considered by Čubrić [24] in the setting of intuitionistic propositional logic and by Saurin [25] for (extensions) of classical linear logic. They call *proof-relevant interpolation* the study of interpolation procedures in relationship to cut rules and equivalence of proofs, like our $\mathring{=}$. In particular, Čubrić and Saurin show that interpolation procedures are in a way "right inverses" of cut rules. Here we show the same for `SkNMILL`: the sMIP procedure is a right inverse of scut, while the cMMIP procedure is a right inverse of ccut^*.

Theorem 9.

(i) Let $g : S \mid \Gamma_0 \vdash D$ and $h : D \mid \Gamma_1 \vdash C$ be the derivations obtained by applying the sMIP procedure on a derivation $f : S \mid \Gamma \vdash C$ with the partition $\langle \Gamma_0, \Gamma_1 \rangle$. Then $\mathsf{scut}(g, h) \mathring{=} f$.

(ii) Let $g_i : - \mid \Delta_i \vdash D_i$ for $i \in [1, \ldots, n]$ and $h : S \mid \Gamma_0, D_1, \ldots, D_n, \Gamma_2 \vdash C$ be derivations obtained by applying the cMMIP procedure on a derivation $f : S \mid \Gamma \vdash C$ with the partition $\langle \Gamma_0, \Gamma_1, \Gamma_2 \rangle$. Then $\mathsf{ccut}^*([g_i], h) \mathring{=} f$.

Proof. Similar to the proof of Theorem 5, statements (i) and (ii) are proved by mutual induction on the structure of derivations. We focus on the proof of statement (i), since (ii) is proved in a similar manner. We refer the interested reader to our Agda formalization for all the technical details.

The proof relies on the computational behaviour of the admissible rules scut and ccut. The reader might want to consult our previous work [18, 21] for the explicit construction of the cut rules that we employ in this proof.

Case $f = \mathsf{ax}$. The goal reduces to $\mathsf{scut}(\mathsf{ax}, \mathsf{ax}) \mathring{=} \mathsf{ax}$, which holds by definition of scut.

Case $f = \mathsf{IR}$. The goal reduces to $\mathsf{scut}(\mathsf{IR}, \mathsf{IL}\ \mathsf{IR}) \mathring{=} \mathsf{IR}$, which holds by definition of scut.

Case $f = \mathsf{IL}\ f'$. The goal reduces to $\mathsf{scut}(\mathsf{IL}\ g', h') \mathring{=} \mathsf{IL}\ f'$[1]. By definition of scut we have $\mathsf{scut}(\mathsf{IL}\ g', h') = \mathsf{IL}\ (\mathsf{scut}(g', h'))$. By inductive hypothesis on f', we have $\mathsf{scut}(g', h') \mathring{=} f'$ and then by congruence of $\mathring{=}$, we obtain $\mathsf{IL}(\mathsf{scut}(g', h')) \mathring{=} \mathsf{IL}\ f'$, as desired.

Cases $f = \otimes\mathsf{L}\ f'$ and $f = \multimap\mathsf{R}\ f'$. Analogous to the previous case. Though the case of $\multimap\mathsf{R}$ requires also an application of Proposition 3.

Case $f = \mathsf{pass}\ f'$. Two cases determined by whether Γ_0 is empty or not.

[1] Here g' and h' are as in the proof of Theorem 5. We follow the same convention for the forthcoming cases too, where names of derivations match the ones in the proof of Theorem 5.

- In the first case, the goal reduces to $\mathsf{scut}(\mathsf{IR}, \mathsf{IL}(\mathsf{pass}\ f')) \mathrel{\mathring{=}} \mathsf{pass}\ f'$, which holds by definition of scut.
- In the second case, the goal reduces to $\mathsf{scut}(\mathsf{pass}\ g', h') \mathrel{\mathring{=}} \mathsf{pass}\ f'$. By definition of scut we have $\mathsf{scut}(\mathsf{pass}\ g', h') = \mathsf{pass}(\mathsf{scut}(g', h'))$. By inductive hypothesis on f' and congruence, the latter is $\mathring{=}$-related to $\mathsf{pass}\ f'$.

Case $f = \otimes\mathsf{R}(f', f'')$. Two cases determined by whether Γ_0 is fully contained in the context of the left premise or not.

- In the first case, the goal reduces to $\mathsf{scut}(g', \otimes\mathsf{R}(h', f'')) \mathrel{\mathring{=}} \otimes\mathsf{R}(f', f'')$. By Proposition 3, we have $\mathsf{scut}(g', \otimes\mathsf{R}(h', f'')) \mathrel{\mathring{=}} \otimes\mathsf{R}(\mathsf{scut}(g', h'), f'')$. By inductive hypothesis on f' and congruence, the latter is $\mathring{=}$-related to $\otimes\mathsf{R}(f', f'')$.
- In the second case, the goal reduces to showing that the derivation $\mathsf{scut}(\otimes\mathsf{R}(g, h'), \otimes\mathsf{L}\ (\otimes\mathsf{R}(h', \mathsf{pass}`h'')))$ is $\mathring{=}$-related to $\otimes\mathsf{R}(f', f'')$. This is witnessed by the following sequence of equivalences:

$$
\begin{aligned}
&\mathsf{scut}(\otimes\mathsf{R}(g, h'), \otimes\mathsf{L}(\otimes\mathsf{R}(h', \mathsf{pass}`h''))) \\
&\quad = \mathsf{scut}(g', \otimes\mathsf{R}(h', \mathsf{scut}(g'', h''))) && \text{(by definition of } \mathsf{scut}) \\
&\quad \mathrel{\mathring{=}} \otimes\mathsf{R}(\mathsf{scut}(g', h'), \mathsf{scut}(g'', h'')) && \text{(by Proposition 3)} \\
&\quad \mathrel{\mathring{=}} \otimes\mathsf{R}(f', f'') && \text{(by ind. hyp. on } f' \text{ and } f'' \\
& && \text{and congruence)}
\end{aligned}
$$

Case $f = \multimap\mathsf{L}(f', f'')$. Two cases determined by whether Γ_1 is fully contained in the context of the right premise or not.

- In the first case, the goal reduces to $\mathsf{scut}(\multimap\mathsf{L}(f', g''), h'') \mathrel{\mathring{=}} \multimap\mathsf{L}(f', f'')$. By definition of scut, we have $\mathsf{scut}(\multimap\mathsf{L}(f', g''), h'') = \multimap\mathsf{L}(f', (\mathsf{scut}(g'', h''))$. By inductive hypothesis on f'' and congruence, the latter is $\mathring{=}$-related to $\multimap\mathsf{L}(f', f'')$.
- In the second case, the goal reduces to showing that the derivation $\mathsf{scut}(\multimap\mathsf{R}^*(\multimap\mathsf{L}(h', g'')), \multimap\mathsf{L}^*([g_i'], h''))$ is $\mathring{=}$-related to $\multimap\mathsf{L}(f', f'')$. This is witnessed by the following sequence of equivalences:

$$
\begin{aligned}
&\mathsf{scut}(\multimap\mathsf{R}^*(\multimap\mathsf{L}(h', g'')), \multimap\mathsf{L}^*([g_i'], h'')) \\
&\quad \mathrel{\mathring{=}} \mathsf{scut}(\mathsf{ccut}^*([g_i'], \multimap\mathsf{L}(h', g'')), h'') && \text{(by Proposition 2)} \\
&\quad = \multimap\mathsf{L}(\mathsf{ccut}^*([g_i'], h'), \mathsf{scut}(h'', g'')) && \text{(by definition of } \mathsf{scut} \text{ and } \mathsf{ccut}^*) \\
&\quad \mathrel{\mathring{=}} \multimap\mathsf{L}(f', f'') && \text{(by ind. hyp. on } f' \text{ and } f'' \\
& && \text{and congruence)}
\end{aligned}
$$

The final step employs the "inductive hypothesis" on f', which in this case means the validity of statement (ii) for derivation f' (remember that statements (i) and (ii) are proved simultaneously by structural induction on derivations).

□

7 Agda Formalization

In this section we discuss some details of the Agda formalization. We start by recalling some definitions and proceed to explain how formulae and sequents are implemented as types.

Basic notions and notation. The type $\mathsf{Maybe}\ A$ is the type of optional elements of A, i.e. either $\mathsf{just}\ x : \mathsf{Maybe}\ A$, for some element $x : A$, or $\mathsf{nothing} : \mathsf{Maybe}\ A$. The type $\mathsf{List}\ A$ is the type of ordered lists of elements in A, which is inductively generated by the empty list constructor $[\,] : \mathsf{List}\ A$ and the *cons* constructor $(::) : A \to \mathsf{List}\ A \to \mathsf{List}\ A$ which appends an element of A to the front of a previously constructed list. We write $xs +\!\!+ ys : \mathsf{List}\ A$ for the concatenation of $xs, ys : \mathsf{List}\ A$, and $x \in xs$ for the type of occurrences of x in the list xs. The universe of types is called Set, so given a type $A : \mathsf{Set}$ we also have $\mathsf{Maybe}\ A : \mathsf{Set}$ and $\mathsf{List}\ A : \mathsf{Set}$.

Given $x, y : A$, their judgemental equality is denoted $x = y$ and their propositional equality is $x \equiv y$. For example, given lists $xs, ys, zs : \mathsf{List}\ A$, we have the judgemental equality $[\,] +\!\!+ xs = xs$ (since $+\!\!+$ is defined by recursion on the first argument) and the propositional equalities $xs +\!\!+ [\,] \equiv xs$ and $(xs +\!\!+ ys) +\!\!+ zs \equiv xs +\!\!+ (ys +\!\!+ zs)$, witnessing right unitality and associativity of $+\!\!+$. Agda currently allows the extension of its evaluation relation with new computation rules using the flag `--rewriting`. This enabled us to turn the two latter propositional equalities into judgemental ones. Notice that this is only used to *simplify* the statements and proofs of cut admissibility and Maehara interpolation (and their properties), but all the results that we formalized are also valid with right unitality and associativity of $+\!\!+$ being propositional instead of judgemental.

Given two types A, B, we define the type $A \hookrightarrow B$ of injective functions between A and B, which consists of pairs of a function $f : A \to B$ and en element of type $(x\ y : A)\ (eq : f\ x \equiv f\ y) \to x \equiv y$, which evidences the injectivity of f.

Formulae, stoups and contexts. In the formalization we fix a type At of atomic formulae. The type Fma of formulae is the following inductive type:

$$\begin{aligned}
&\mathsf{data}\ \mathsf{Fma} : \mathsf{Set}\ \mathsf{where}\\
&\quad \mathsf{at} : \mathsf{At} \to \mathsf{Fma}\\
&\quad \mathsf{I} : \mathsf{Fma}\\
&\quad _\otimes_ : \mathsf{Fma} \to \mathsf{Fma} \to \mathsf{Fma}\\
&\quad _\multimap_ : \mathsf{Fma} \to \mathsf{Fma} \to \mathsf{Fma}
\end{aligned}$$

Underscores are used to represent infix operators, i.e. $A \otimes B$ and $A \multimap B$ are formulae for all $A, B : \mathsf{Fma}$. The type of stoups is $\mathsf{Stp} = \mathsf{Maybe}\ \mathsf{Fma}$ and the type of contexts is $\mathsf{Cxt} = \mathsf{List}\ \mathsf{Fma}$.

Sequents. Given a stoup $S : \mathsf{Stp}$, a context $\Gamma : \mathsf{Cxt}$ and a formula $C : \mathsf{Fma}$, the type $S \mid \Gamma \vdash C$ consists of all the possible proofs of the corresponding sequent. This type is inductively generated by the rules in (1), i.e. there is a constructor associated to each of the 8 rules. For example, the inference rule $\multimap\mathsf{L}$ is implemented as a constructor

$$\begin{aligned}
\multimap\mathsf{L}\ :\ &\{\Gamma\ \Delta : \mathsf{Cxt}\}\ \{A\ B\ C : \mathsf{Fma}\}\\
&\to (f : \mathsf{nothing} \mid \Gamma \vdash A)\ (g : \mathsf{just}\ B \mid \Delta \vdash C)\\
&\to \mathsf{just}\ (A \multimap B) \mid \Gamma +\!\!+ \Delta \vdash C
\end{aligned}$$

Curly brackets are used in Agda to denote implicit arguments. Therefore $\multimap\mathsf{L}\ f\ g$ is a term of type $\mathsf{just}\ (A \multimap B) \mid \Gamma +\!\!+ \Delta \vdash C$, whenever we are give terms $f : \mathsf{nothing} \mid \Gamma \vdash A$ and $g : \mathsf{just}\ B \mid \Delta \vdash C$.

Equivalence of derivations. Given two derivations of the same sequent $f, g : S \mid \Gamma \vdash C$, the type $f \stackrel{\circ}{=} g$ consists of all possible ways to identify f and g using: (i) the generating equations in Figures 1 and 2, (ii) equations establishing that $\stackrel{\circ}{=}$ is an equivalence relation and (iii) equations establishing that $\stackrel{\circ}{=}$ is a congruence. Therefore $\stackrel{\circ}{=}$ is implemented as an inductive type family indexed over pairs of derivations. In the Agda definition of $\stackrel{\circ}{=}$, there is: (i) a constructor for each equation in Figures 1 and 2, (ii) constructors for reflexivity, symmetry and transitivity of $\stackrel{\circ}{=}$, and (iii) constructors evidencing the compatibility of $\stackrel{\circ}{=}$ with the inference rules of `SkNMILL`. We present three indicative examples, one for each class of equational generators: the constructor $\otimes$Rpass associated to the first permutative conversion in Figure 1; the constructor sym$\stackrel{\circ}{=}$ associated to symmetry of $\stackrel{\circ}{=}$; the constructor cong$\multimap$L associated to the compatibility of $\stackrel{\circ}{=}$ wrt. $\multimap$L. In the types of the constructors below, and further in this section, we omit some of the implicit arguments to shorten the types and improve readability.

$$
\begin{aligned}
\otimes\mathsf{Rpass} \;&:\; (f : \mathsf{just}\ A' \mid \Gamma \vdash A)\ (g : \mathsf{nothing} \mid \Delta \vdash B) \\
&\to \otimes\mathsf{R}\ (\mathsf{pass}\ f)\ g \stackrel{\circ}{=} \mathsf{pass}\ (\otimes\mathsf{R}\ f\ g) \\
\mathsf{sym}\stackrel{\circ}{=} \;&:\; (f\ g : S \mid \Gamma \vdash C)\ (eq : f \stackrel{\circ}{=} g) \to g \stackrel{\circ}{=} f \\
\mathsf{cong}\multimap\mathsf{L} \;&:\; (f\ g : \mathsf{nothing} \mid \Gamma \vdash A)\ (f'\ g' : \mathsf{just}\ B \mid \Delta \vdash C) \\
&\to (eq : f \stackrel{\circ}{=} g)\ (eq' : f' \stackrel{\circ}{=} g') \to \multimap\mathsf{L}\ f\ f' \stackrel{\circ}{=} \multimap\mathsf{L}\ g\ g'
\end{aligned}
$$

Cut admissibility. In Agda, the two admissible cut rules in (3) becomes two definable terms:

$$
\begin{aligned}
&\mathsf{scut} : (f : S \mid \Gamma \vdash A)\ (g : \mathsf{just}\ A \mid \Delta \vdash C) \to S \mid \Gamma +\!\!+ \Delta \vdash C \\
&\mathsf{ccut} : (f : \mathsf{nothing} \mid \Gamma \vdash A)\ (g : S \mid \Delta_0 +\!\!+ A :: \Delta_1 \vdash C) \to S \mid \Delta_0 +\!\!+ \Gamma +\!\!+ \Delta_1 \vdash C
\end{aligned}
$$

The construction of $\mathsf{scut}\ f\ g$ proceeds by pattern-matching (i.e. structural recursion) on the first argument f and, when f begins with the application of a right introduction rule, continues by pattern-matching on the second argument g. The construction of $\mathsf{ccut}\ f\ g$ should proceed by pattern-matching on the second argument g, but Agda disallows this since it is unable to unify the context $\Delta_0 +\!\!+ A :: \Delta_1$ with the ones appearing in the conclusion of inference rules. This issue is easily fixable by relaxing the context in the type of derivation g to be an arbitrary context Δ, which we subsequently equate to $\Delta_0 +\!\!+ A :: \Delta_1$, as follows:

$$
\begin{aligned}
\mathsf{ccut}' \;&:\; (f : \mathsf{nothing} \mid \Gamma \vdash A)\ (g : S \mid \Delta \vdash C)\ (eq : \Delta \equiv \Delta_0 +\!\!+ A :: \Delta_1) \\
&\to S \mid \Delta_0 +\!\!+ \Gamma +\!\!+ \Delta_1 \vdash C
\end{aligned}
$$

The construction of $\mathsf{ccut}'\ f\ g\ eq$ proceeds by pattern-matching on the second argument g and, when g begins with the application of a left introduction rule, continues by pattern-matching on the first argument f. The functions scut and ccut' need to be mutually-defined, as evidenced by their pen-and-paper definition in [14, 21]. The term $\mathsf{ccut}\ f\ g$ is then definable as $\mathsf{ccut}'\ f\ g\ \mathsf{refl}$, with refl being the reflexivity constructor of the propositional identity type.

Maehara interpolation. The interpolations properties sMIP and cMMIP, not including the variable multiplicity condition (which is discussed later), are implemented in Agda as the following record types:

```
record sMIP (S : Stp) (Γ₀ Γ₁ : Cxt) (C : Fma) : Set where
  field
    D : Fma
    g : S | Γ₀ ⊢ D
    h : just D | Γ₁ ⊢ C

record cMMIP (S : Stp) (Γ₀ Γ₁ Γ₂ : Cxt) (C : Fma) : Set where
  field
    Ds : List Fma
    gs : Γ₁ ⊢* Ds
    h : S | Γ₀ ++ Ds ++ Γ₂ ⊢ D
```

Elements of type sMIP $S\ \Gamma_0\ \Gamma_1\ C$ are triples consisting of a formula D and two derivations g and h. Elements of cMMIP $S\ \Gamma_0\ \Gamma_1\ \Gamma_2\ C$ are also triples, but consisting of a list of formulae Ds (the interpolant formulae $D_1, \ldots, D_n$ in Theorem 5), a sequence of derivations gs (the derivations $g_1, \ldots, g_n$ in Theorem 5) and a derivation h. Given a context Δ and a list of formulae As, the type $\Delta \vdash^* As$ is inductively defined by the two constructors $[\,]^*$ and $\_ ::^* \_$ below:

```
[ ]* : [ ] ⊢* [ ]
_ ::* _ : (f : nothing | Δ₀ ⊢ A) (fs : Δ₁ ⊢* As) → Δ₀ ++ Δ₁ ⊢* A :: As
```

In other words, an element of type $\Delta \vdash^* As$, with As being the list of formulae $A_1, \ldots, A_n$, consists in a partition of Δ in n parts $\Delta_1, \ldots, \Delta_n$ and a sequence of derivations of type nothing $\mid \Delta_i \vdash A_i$, one for each $1 \leq i \leq n$. This implies that in our Agda implementation of cMMIP, the partition of Γ_1 and the derivations g_i in the statement of cMMIP in Theorem 5 are both collected in the field $gs : \Gamma_1 \vdash^* Ds$.

The proof of Theorem 5 in Agda becomes the construction of the two following functions, defined mutually by pattern matching on their first argument f:

```
smip  : (f : S | Γ ⊢ C) (eq : Γ ≡ Γ₀ ++ Γ₁) → sMIP S Γ₀ Γ₁ C
cmmip : (f : S | Γ ⊢ C) (eq : Γ ≡ Γ₀ ++ Γ₁ ++ Γ₂) → cMMIP S Γ₀ Γ₁ Γ₂ C
```

Variable multiplicity condition. We recursively define a function atom collecting the atomic formulae appearing in a formula:

```
atom : Fma → List At
atom (at X)   = X :: [ ]
atom I        = [ ]
atom (A ⊗ B)  = atom A ++ atom B
atom (A ⊸ B)  = atom A ++ atom B
```

Extensions of this function that collect atomic formulae in stoups and contexts, called atomS and atomC, can be easily defined from atom.

The variable conditions of the interpolation properties sMIP and cMMIP are implemented in Agda as the following record types:

```
record sVar (S : Stp) (Γ₀ Γ₁ : Cxt) (C D : Fma) : Set where
  field
    varg : (X : At) → (X ∈ atom D) ↪ (X ∈ atomS S ++ atomC Γ₀)
    varh : (X : At) → (X ∈ atom D) ↪ (X ∈ atomC Γ₁ ++ atom C)

record cVar (S : Stp) (Γ₀ Γ₁ Γ₂ : Cxt) (C Ds : Fma) : Set where
  field
    vargs : (X : At) → (X ∈ atomC Ds) ↪ (X ∈ atomC Γ₁)
    varh : (X : At)
      → (X ∈ atomC Ds) ↪ (X ∈ atomS S ++ atomC (Γ₀ ++ Γ₂) ++ atom C)
```

The projection varg returns an injective function that assigns to each occurrence of an atomic formula X in D an occurrence of X in S, Γ_0, i.e. it is a faithful implementation of the condition $\sigma_X(D) \leq \sigma_X(S, \Gamma_0)$. The other projections of types sVar and cVar appropriately represent the variable multiplicity conditions of sMIP and cMMIP. The variable conditions are established by constructing the following pair of functions, defined recursively on the argument f. In the types below, the formula sMIP.D (smip f eq) is the interpolant computed by the stoup Maehara interpolation procedure on input f. Similarly, sMIP.Ds (cmmip f eq) is the sequence of interpolant formulae produced by the context Maehara multi-interpolation procedure on input f.

```
svar  :  (f : S | Γ ⊢ C) (eq : Γ ≡ Γ₀ ++ Γ₁)
      →  sVar S Γ₀ Γ₁ C (sMIP.D (smip f eq))
cvar  :  (f : S | Γ ⊢ C) (eq : Γ ≡ Γ₀ ++ Γ₁ ++ Γ₂)
      →  cVar S Γ₀ Γ₁ C (sMIP.Ds (cmmip f eq))
```

Proof-relevant interpolation. Given a derivation $f : S \mid \Gamma \vdash C$ and an equality proof $eq : \Gamma \equiv \Gamma_0 +\!\!+ \Gamma_1$, evidencing the partition of context Γ in two parts Γ_0 and Γ_1, we first apply the stoup Maehara interpolation procedure smip on them, obtaining an interpolant formula and two derivations sMIP.g (smip f eq) and sMIP.h (smip f eq) (the g and h in the statement of sMIP in Theorem 5). Then we apply the cut procedure scut on these, resulting in a derivation which is $\doteq$-related to the original f. In Agda, proving proof-relevant stoup interpolation translates to the construction of a term scutsmip with the following type:

```
scutsmip  :  (f : S | Γ ⊢ C)(eq : Γ ≡ Γ₀ ++ Γ₁)
          →  scut (sMIP.g (smip f eq)) (sMIP.h (smip f eq))
             ≐
             subst (λx. S | x ⊢ C) eq f
```

Notice that f is a term of type $S \mid \Gamma \vdash C$, while the derivation on the left-hand-side of $\mathring{=}$ has type $S : \Gamma_0 +\!\!+ \Gamma_1 \vdash C$. In order to state that the two derivations are $\mathring{=}$-related we need to substitute in the type of f using the equality proof $eq : \Gamma \equiv \Gamma_0 +\!\!+ \Gamma_1$. The definition of scutsmip proceeds by pattern-matching on the argument f.

Proving the analogous property for context Maehara multi-interpolation requires a bit of preparation. We define the context multi-cut rule introduced in (6), which is an iterated version of ccut taking a sequence of derivations $fs : \Gamma \vdash^* As$ in input instead of a single derivation. This is constructed by pattern-matching on the first argument fs.

$$\begin{aligned} \mathsf{ccut}^* \;:\; & (fs : \Gamma \vdash^* As)\ (g : S \mid \Delta \vdash C)\ (eq : \Delta \equiv \Delta_0 +\!\!+ As +\!\!+ \Delta_1) \\ & \to S \mid \Delta_0 +\!\!+ \Gamma +\!\!+ \Delta_1 \vdash C \end{aligned}$$

In Agda, proof-relevant context interpolation translates to the construction of a term ccutcmmip with the following type, defined by pattern-matching on the argument f:

$$\begin{aligned} \mathsf{ccutcmmip} \;:\; & (f : S \mid \Gamma \vdash C)(eq : \Gamma \equiv \Gamma_0 +\!\!+ \Gamma_1 +\!\!+ \Gamma_2) \\ & \to \mathsf{ccut}^*\ (\mathsf{cMMIP}.gs\ (\mathsf{cmmip}\ f\ eq))\ (\mathsf{cMMIP}.h\ (\mathsf{cmmip}\ f\ eq)) \\ & \quad \mathring{=} \\ & \quad \mathsf{subst}\ (\lambda x.\ S \mid x \vdash C)\ eq\ f \end{aligned}$$

8 Conclusions and Future Work

This paper describes a proof of Craig interpolation for the semi-substructural logic SkNMILL. It employs proof-theoretic techniques, since it relies on cut elimination and it manipulates sequent calculus derivations directly. As common when proving Craig interpolation for other substructural logics, the proof strategy follows a variant of Maehara's method [9]: we prove a stoup Maehara interpolation property (similar to the one for the Lambek calculus [10]) and a context Maehara multi-interpolation property (similar to the one for the product-free fragment of the Lambek calculus [12]) simultaneously by mutual structural induction.

Following the category-theoretic considerations of Čubrić [24], we proved a proof-relevant form of interpolation, showing that the interpolation procedures are right inverses of the corresponding admissible cut rules. The main aspect missing in our work (and, as far as we know, the whole literature on Craig interpolation) is the characterization of the Craig interpolant via a universal property, in the sense of category theory. In fact, Čubrić's characterization of interpolants is merely an *existence* property, there is no mention of a *uniqueness* property. This is analogous to the distinction between *weak* and *non-weak* (co)limits in category theory. Alternatively, we may ask whether the interpolation procedures are also *left* inverses of the cut rules.

These questions naturally lead to another one: what is the correct notion of "equality" between interpolants? First, notice that in SkNMILL interpolants satisfying sMIP for a fixed sequent $S \mid \Gamma \vdash C$ and partition $\langle \Gamma_1, \Gamma_2 \rangle$ of Γ can be organized in a set of triples:

$$\{(D, g : S \mid \Gamma_1 \vdash D, h : D \mid \Gamma_2 \vdash C) \;\mid\; \forall X. \sigma_X(D) \leq \sigma_X(S, \Gamma_0) \& \sigma_X(D) \leq \sigma_X(\Gamma_1, C)\}$$

Many equivalence relations can potentially be defined on these triples:

- $(D, g, h) \sim (D', g', h')$ if and only if $D = D'$, $g = g'$ and $h = h'$. This option is definitely too strict, since it does not account for the fact that we consider derivations that differ by some η-conversion or permutative conversion as equal. A better option would then be:
- $(D, g, h) \sim (D', g', h')$ if and only if $D = D'$, $g \mathrel{\mathring{=}} g'$ and $h \mathrel{\mathring{=}} h'$. This might still be too restrictive. For example, in the Lambek calculus, we can run Maehara's method on two derivations f and f' for the same sequent that only differ by a permutative conversion, and obtain different interpolant formulae D and D'. The same phenomenon could also happen in `SkNMILL`. A relaxation of this notion would then be:
- $(D, g, h) \sim (D', g', h')$ if and only if there is an isomorphism $d : D \mid \ \vdash D'$ such that $\mathsf{scut}(g, d) \mathrel{\mathring{=}} g'$ and $h \mathrel{\mathring{=}} \mathsf{scut}(d, h')$. By isomorphism here we mean that there exists a derivation $d' : D' \mid \ \vdash D$ such that $\mathsf{scut}(d, d') \mathrel{\mathring{=}} \mathsf{ax}$ and $\mathsf{scut}(d', d) \mathrel{\mathring{=}} \mathsf{ax}$. Yet another weaker option could be:
- $\sim$ is the equivalence relation generated by the relation: $(D, g, h) \sim_0 (D', g', h')$ if and only if there exists a derivation $d : D \mid \ \vdash D'$ such that $\mathsf{scut}(g, d) \mathrel{\mathring{=}} g'$ and $h \mathrel{\mathring{=}} \mathsf{scut}(d, h')$. More explicitly, triples (D, g, h) and (D', g', h') are $\sim$-related when there exists a list of formulae $D_1, \ldots, D_n$ and a "zigzag" of derivations like

$$d_1 : D \mid \ \vdash D_1, \quad d_2 : D_2 \mid \ \vdash D_1, \quad d_3 : D_2 \mid \ \vdash D_3, \quad \ldots \quad d_n : D_n \mid \ \vdash D'$$

  such that, when appropriately composed with these d_i-s, g is $\mathring{=}$-related to g' and h is $\mathring{=}$-related to h'. At first sight, this might seem like a weird notion of equality between interpolants, but it is in fact a very natural one to require from a category-theoretic perspective, since it would be the one characterizing interpolants as some kind of colimit/coend.

The attentive reader might have noticed that, while we consider sets of derivations quotiented by the congruence relation $\mathring{=}$, we do not prove that the interpolation procedures of Theorem 5 are well-defined wrt. $\mathring{=}$, e.g. that sMIP sends $\mathring{=}$-related derivations to the "same" triple (D, g, h). The reason for this omission comes again from that fact that we do not yet know what is the appropriate notion of "sameness" in this case.

We do not want to continue our speculations in this conclusive section and leave further investigations on this topic to future work. Our first step will be understanding the universal property of interpolants for logics in which the Maehara interpolation property is simpler than in `SkNMILL`, e.g. in the associative Lambek calculus. This approach could be seen as a step towards the universal proof theory for substructural logics in the sense of [28].

Another venue of future work would be the extension of the results of this paper to other semi-substructural logics, e.g. extensions with a notion of skew exchange [29] or additive connectives [22]. In the latter setting we might also ask whether the logic satisfies a uniform interpolation property in the sense of [30].

Acknowledgments

This work was supported by the Estonian Research Council grant PSG749. Cheng-Syuan Wan's presentation of this work and participation in The Proof Society Workshop were supported by the EU COST action CA19135 (CERCIRAS).

References

[1] Craig, W.: Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. Journal of Symbolic Logic **22**(3), 269–285 (1957) https://doi.org/10.2307/2963594

[2] Beth, E.W.: On Padoa's method in the theory of definition. Indagationes Mathematicae (Proceedings) **56**, 330–339 (1953) https://doi.org/10.1016/s1385-7258(53)50042-3

[3] Henzinger, T.A., Jhala, R., Majumdar, R., McMillan, K.L.: Abstractions from proofs. ACM SIGPLAN Notices **39**(1), 232–244 (2004) https://doi.org/10.1145/982962.964021

[4] Lambek, J.: The mathematics of sentence structure. American Mathematical Monthly **65**(3), 154–170 (1958) https://doi.org/10.2307/2310058

[5] Lambek, J.: On the calculus of syntactic types. In: Jakobson, R. (ed.) Structure of Language and Its Mathematical Aspects, pp. 166–178. AMS, Providence (1961). https://doi.org/10.1090/psapm/012

[6] Girard, J.-Y.: Linear logic. Theoretical Computer Science **50**, 1–102 (1987) https://doi.org/10.1016/0304-3975(87)90045-4

[7] Fussner, W., Santschi, S.: Interpolation in linear logic and related systems. ACM Transactions on Computational Logic **25**(4), 1–19 (2024) https://doi.org/10.1145/3680284

[8] Kihara, H., Ono, H.: Interpolation properties, Beth definability properties and amalgamation properties for substructural logics. Journal of Logic and Computation **20**(4), 823–875 (2009) https://doi.org/10.1093/logcom/exn084

[9] Maehara, S.: Craig's interpolation theorem. Mathematics **12**(4), 235–237 (1961) https://doi.org/10.11429/sugaku1947.12.235

[10] Ono, H.: Proof-theoretic methods in nonclassical logic –an introduction. In: Theories of Types and Proofs, pp. 207–254 (1998). https://doi.org/10.2969/msjmemoirs/00201c060

[11] Moot, R., Retoré, C.: The Logic of Categorial Grammars: A Deductive Account of Natural Language Syntax and Semantics. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31555-8

[12] Pentus, M.: Product-free Lambek calculus and context-free grammars. Journal of Symbolic Logic **62**(2), 648–660 (1997) https://doi.org/10.2307/2275553

[13] Kanazawa, M.: Computing interpolants in implicational logics. Annals of Pure and Applied Logic **142**(1-3), 125–201 (2006) https://doi.org/10.1016/J.APAL.2005.12.014

[14] Uustalu, T., Veltri, N., Wan, C.-S.: Proof theory of skew non-commutative MILL. In: Indrzejczak, A., Zawidzki, M. (eds.) Proceedings of 10th International Conference on Non-classical Logics: Theory and Applications, NCL 2022. Electronic Proceedings in Theoretical Computer Science, vol. 358, pp. 118–135. Open Publishing Association, Australia (2022). https://doi.org/10.4204/eptcs.358.9

[15] Girard, J.-Y.: A new constructive logic: Classical logic. Mathematical Structures in Computer Science **1**(3), 255–296 (1991) https://doi.org/10.1017/s0960129500001328

[16] Szlachányi, K.: Skew-monoidal categories and bialgebroids. Advances in Mathematics **231**(3–4), 1694–1730 (2012) https://doi.org/10.1016/j.aim.2012.06.027

[17] Zeilberger, N.: A sequent calculus for a semi-associative law. Logical Methods in Computer Science **15**(1) (2019) https://doi.org/10.23638/lmcs-15(1:9)2019

[18] Uustalu, T., Veltri, N., Zeilberger, N.: The sequent calculus of skew monoidal categories. In: Casadio, C., Scott, P.J. (eds.) Joachim Lambek: The Interplay of Mathematics, Logic, and Linguistics. Outstanding Contributions to Logic, vol. 20, pp. 377–406. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-66545-6_11

[19] Uustalu, T., Veltri, N., Zeilberger, N.: Deductive systems and coherence for skew prounital closed categories. In: Sacerdoti Coen, C., Tiu, A. (eds.) Proceedings of 15th Workshop on Logical Frameworks and Meta-Languages: Theory and Practice, LFMTP 2020. Electronic Proceedings in Theoretical Computer Science, vol. 332, pp. 35–53. Open Publishing Association, Australia (2021). https://doi.org/10.4204/eptcs.332.3

[20] Veltri, N.: Maximally multi-focused proofs for skew non-commutative MILL. In: Hansen, H.H., Scedrov, A., Queiroz, R.J.G.B. (eds.) Proceedings of 29th International Workshop on Logic, Language, Information, and Computation, WoLLIC 2023. Lecture Notes in Computer Science, vol. 13923, pp. 377–393. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-39784-4_24

[21] Wan, C.-S.: Semi-substructural logics à la Lambek. In: Indrzejczak, A., Zawidzki, M. (eds.) Proceedings of 11th International Conference on Non-classical Logics: Theory and Applications, NCL 2024. Electronic Proceedings in Theoretical Computer Science, vol. 415, pp. 195–213. Open Publishing Association, Australia (2024). https://doi.org/10.4204/eptcs.415.18

[22] Veltri, N., Wan, C.-S.: Semi-substructural logics with additives. In: Temur Kutsia, D.M. Daniel Ventura, Morales, J.F. (eds.) Proceedings of 18th International Workshop on Logical and Semantic Frameworks, with Applications and 10th Workshop on Horn Clauses for Verification and Synthesis, LSFA/HCVS 2023. Electronic Proceedings in Theoretical Computer Science, pp. 63–80. Open Publishing Association, Australia (2023). https://doi.org/10.4204/eptcs.402.8

[23] Andreoli, J.-M.: Logic programming with focusing proofs in linear logic. Journal of Logic and Computation **2**(3), 297–347 (1992) https://doi.org/10.1093/logcom/2.3.297

[24] Čubrić, D.: Interpolation property for bicartesian closed categories. Archive for Mathematical Logic **33**(4), 291–319 (1994) https://doi.org/10.1007/bf01270628

[25] Saurin, A.: Interpolation as cut-introduction. manuscript (2024). Available at https://www.irif.fr/_media/users/saurin/pub/interpolation_as_cut_introduction.pdf

[26] Abrusci, V.M.: Non-commutative intuitionistic linear logic. Mathematical Logic Quarterly **36**(4), 297–318 (1990) https://doi.org/10.1002/malq.19900360405

[27] Roorda, D.: Resource logics: proof-theoretical investigations. PhD thesis, FWI, Universiteit van Amsterdam (1991). Available at https://pure.knaw.nl/ws/portalfiles/portal/1821261/Roorda_1991_Resource_Logics_Proof_theoretical_Investigations.pdf

[28] Tabatabai, A.A., Jalali, R.: Universal proof theory: Semi-analytic rules and craig interpolation. Annals of Pure and Applied Logic **176**(1), 103509 (2025) https://doi.org/10.1016/j.apal.2024.103509

[29] Veltri, N.: Coherence via focusing for symmetric skew monoidal and symmetric skew closed categories. Journal of Logic and Computation, 059 (2024) https://doi.org/10.1093/logcom/exae059

[30] Alizadeh, M., Derakhshan, F., Ono, H.: Uniform Interpolation in Substructural Logics. The Review of Symbolic Logic **7**(3), 455–483 (2014) https://doi.org/10.1017/s175502031400015x