

The Logic of Intersection Subtyping

Olivier Laurent 

CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP, UMR 5668, Lyon, France

Abstract

The subtyping relation of programming languages can be analysed as an entailment relation by means of proof theory. We are interested in two main families of systems: intersection types and polymorphic subtyping. They share the fact that implication has some distributivity property: over intersection in the first case and over universal quantification in the second one.

We introduce a restriction of the second-order (full) Lambek calculus which is stable under cut-elimination and conservatively extends these two subtyping relations. This new system **IS** is an intuitionistic non-commutative linear sequent calculus which provides a natural logical setting for the study of subtyping relations.

We recover sequent calculi from the literature (as well as new variants) as restrictions of **IS** (thanks to the proof-theoretical analysis of the system: admissible rules, invertibility, focusing, etc.), so that **IS** appears as a unifying logic for subtyping. We also develop translations relating **IS** with relevant logic, the (unconstrained) Lambek calculus or cyclic linear logic.

2012 ACM Subject Classification Theory of computation → Proof theory; Theory of computation → Linear logic; Theory of computation → Lambda calculus

Keywords and phrases Intersection types, subtyping, polymorphic subtyping, Lambek calculus, linear logic, cut elimination

Digital Object Identifier 10.4230/LIPIcs.LICS.2026.62

Supplementary Material *Software (Source Code)*: <https://github.com/olaure01/islogic/tree/LICS2026> [16], archived at `swh:1:dir:682600883bd8fd60057d332c6a87a39c047f22e7`

Acknowledgements Thanks to A. Dudenhefner for many fruitful discussions about intersection types, and to the anonymous reviewers for their suggestions for improving the presentation.

1 Introduction

It is standard to move from logics and theories to (pre)order relations by considering the Lindenbaum algebra. This is at the heart of algebraic logic and algebraic semantics. In the opposite direction, one can use logical systems to implement some families of (pre)orders. A pioneering work in that direction is due to P. Whitman [35] who used a cut-free sequent calculus to prove properties about free lattices (see [30] for some extensions).

We want to apply this approach to subtyping relations. Subtyping is a central notion in a great number of programming languages. It has been the subject of many abstract studies, looking for properties of subtyping, algorithmic analyses, semantic interpretations, etc. Our goal is to strengthen the logical view of subtyping.

A first step in extending [35] to types and subtyping is to introduce the arrow type constructor ($\rightarrow$). It comes as a binary constructor with contra-variance on the left and co-variance on the right which can be simply formalized by the rule:

$$\frac{C \leq A \quad B \leq D}{A \rightarrow B \leq C \rightarrow D} \rightarrow \leq \rightarrow$$

We are interested in two main families of subtyping systems: intersection types [3] and polymorphic subtyping [24]. They introduce connectives $\cap$ and $\forall$ coming with distributivity properties with respect to $\rightarrow$:

$$(A_1 \rightarrow B_1) \cap (A_2 \rightarrow B_2) \leq (A_1 \cap A_2) \rightarrow (B_1 \cap B_2) \quad \text{and} \quad \forall X (A \rightarrow B) \leq (\forall X A) \rightarrow (\forall X B).$$



© Olivier Laurent;

licensed under Creative Commons License CC-BY 4.0

41st Annual Symposium on Logic in Computer Science (LICS 2026).

Editors: Claudia Faggian and Joost-Pieter Katoen; Article No. 62; pp. 62:1–62:26

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

By interpreting $\forall$ as an infinitary intersection, these two inequations can be seen as instances of the same one. For this reason, we will use **intersection subtyping** as a generic name for subtyping systems satisfying such a distributivity inequation. The extension of [35] with $(\rightarrow \leq \rightarrow)$ is not expressive enough for interpreting intersection subtyping: the distributivity inequations do not hold. Two distinct lines of work (one for $\cap$ and the other for $\forall$) have provided answers to this problem.

In the realm of intersection types, H. Barendregt, M. Coppo and M. Dezani-Ciancaglini [3] defined an intersection type system, now known under the name **BCD**. This system relies on a subtyping relation, and most of the properties of the BCD typing system are related with properties of BCD subtyping. Because of the presence of a transitivity rule (violating the sub-formula property) in their presentation of the subtyping system, it is not easy to reason by induction on derivations. To address this issue, B. Pierce [26] proposed an algorithm based on a transitivity-free presentation of BCD subtyping (see [7, Figure 12] for a deductive-system-style description of the algorithm). The link with logic and proof-theory is not explicitly mentioned there, and it is a point we want to stress: BCD subtyping can be interpreted as a sub-structural logic.

In the realm of polymorphic subtyping, three key steps can be identified in the history of its proof-theoretical analysis. First, J. Mitchell [24] described an “Hilbert-style” system (sound and complete w.r.t. the semantics of “simple inference models”). Second, G. Longo, K. Milsted and S. Soloviev [23] turned it into a “quasi-sequent-calculus” system (based on a deep inference rule for $\forall$). A key contribution is to get a sub-formula property thanks to the admissibility of the transitivity rule in their system. Finally, J. Tiuryn [32] gave (shallow) natural deduction and sequent calculus systems equivalent to the previous ones. There, the proof-theoretical aspect of the work is explicit and the last sentence of [32] is: “Thus our logic can be qualified as noncommutative linear intuitionistic logic.”

Let us describe the ingredients of these logical systems and their common points leading to a **logic of intersection subtyping**, with the principle that trying to avoid the use of an explicit transitivity rule becomes admissibility of the cut rule (leading to the sub-formula property). A first folklore remark is that Whitman’s system [35] coincides with the two-sided version of additive linear logic with units [12]. Once the rule $(\rightarrow \leq \rightarrow)$ is added, we then have to deal with the distributivity properties. Such properties naturally occur in intuitionistic logics (for example $A \rightarrow (B \wedge C) \simeq (A \rightarrow B) \wedge (A \rightarrow C)$ in intuitionistic logic, or $A \multimap (B \& C) \simeq (A \multimap B) \& (A \multimap C)$ in intuitionistic linear logic). Technically, this is provided by intuitionistic sequents which allow multiple formulas as hypotheses. At this point, we must be very careful not to be too expressive. Properties such as $B \leq A \rightarrow B$, $A \rightarrow B \rightarrow C \leq B \rightarrow A \rightarrow C$ or $(A \rightarrow A) \rightarrow B \leq B$ are not desired ones in the realm of subtyping. This is where the conclusion sentence of [32] is crucial: we need the logic to be *intuitionistic* (to get distributivity properties), *linear* (to avoid reuse or deletion of hypotheses) and *non-commutative* (to preserve the order of function arguments). There is a very well known such logic which goes back to 1958: *the Lambek calculus* [15]! The key remark is that, while being very close to what we are looking for, the Lambek calculus is still too expressive: it is not a conservative extension of BCD subtyping or Mitchell’s polymorphic subtyping. A counter-example is $A \rightarrow B \leq (C \rightarrow A) \rightarrow C \rightarrow B$. In fact the sequent calculus of [32] contains an additional constraint which we will move to the setting of the Lambek calculus: on the left of sequents, $A \rightarrow B$ can only be introduced as the leftmost formula and A must come with a singleton context:

$$\frac{D \leq A \quad B, \Sigma \leq C}{A \rightarrow B, D, \Sigma \leq C} \rightarrow S$$

Restricting the $(\rightarrow L)$ rule to this $(\rightarrow S)$ rule is what we call the **S-constraint** (“S” for *subtyping*). Our logic **IS** is then obtained by applying the S-constraint to the Lambek calculus (with one additional subtlety on the choice of connectives: the multiplicative conjunction as well as one arrow must be removed).

Both Pierce’s system [26] and Tiuryn’s system [32] are sub-systems of **IS**, which then appears as a unifying logic for subtyping. This exhibits a connection between two systems known for more than 40 years: the Lambek Calculus (introduced for linguistics) and BCD subtyping (introduced for models of the untyped λ -calculus).

The properties of **IS**, of its sub-systems, their equivalence with previous systems from the literature and with other logics, come from its proof-theoretical analysis. The central result is cut admissibility for **IS** which must be driven carefully since the full intuitionistic cut rule known to be admissible in the Lambek calculus [15] is *not* admissible in **IS**. Its appropriate “S-restriction” has to be considered:

$$\frac{\Gamma \leq A \quad \Delta, A, \Sigma \leq C \quad \Delta \neq \varepsilon \Rightarrow |\Gamma| = 1}{\Delta, \Gamma, \Sigma \leq C} \text{Scut}$$

which is still stronger than transitivity of $\leq$ (consider $|\Gamma| = 1$ with Δ and Σ empty).

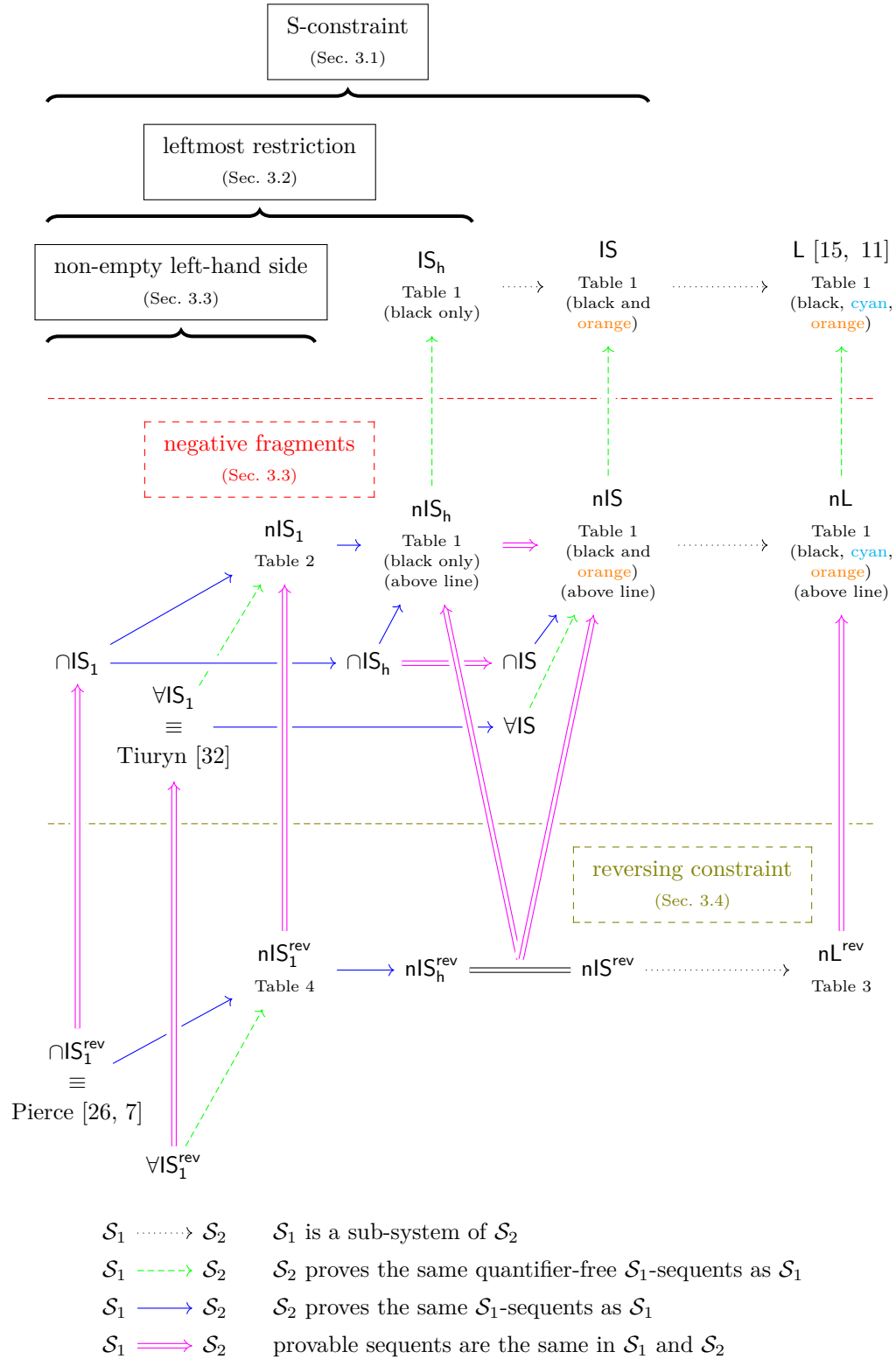
Contents. After some discussions about notations and the structure of sequents (Section 2), we introduce the second-order (full) Lambek calculus **L** (*i.e.* containing all the connectives from propositional linear logic as well as second-order quantifiers) together with a syntactic proof of cut admissibility (built through local cut-reduction steps). By removing the connectives $\otimes$ and $\multimap$ and by imposing the S-constraint, we define the new logic **IS** and prove it is a sub-system of **L** stable under the cut-reduction steps when restricted to the (*Scut*) rule (Section 3.1). In a second step, we study a variant of **IS** where left rules can only be applied on leftmost formulas of sequents (Section 3.2). This is again stable under cut-reduction steps from **L**, and does not impact provability when we consider negative connectives only (Section 3.3). Finally we go deeper in the proof-theoretical analysis of (sub-systems of) **IS** by analysing invertible/reversible rules and by developing a focusing theory (Section 3.4).

In Section 4, we work on the links with intersection types. After recalling the main ingredients of intersection type systems (and in particular the BCD system [3]), we show how the (restriction without bounded variables of the) algorithm introduced in [26] coincides with backward proof-search in a sub-system of **IS**, and how it leads to a simple proof of the β -condition. A faithful embedding of this system into the standard Lambek calculus is defined thanks to a Girard-style translation of implication, leading to a new provability equivalence between BCD and **L** (and also cyclic linear logic) (Section 4.6). The known relation between BCD and relevant logic **B+** [34] is also refined (Section 4.7). We conclude this section with two variants of **IS** allowing to represent the subtyping relations behind the Scott (Section 4.8.1) and Park (Section 4.8.2) models of the untyped λ -calculus. Again we get a sub-formula property and then direct proofs of the validity of the β - and η -conditions.

In Section 5, we consider polymorphic subtyping. We first remark that we can find the sequent calculus introduced in [32] as a sub-system of **IS** (thus with a new proof of cut admissibility through **IS**). The theory of focusing developed for **IS**, allows us to refine this system without touching provability.

In Appendix A, we provide a few ingredients about a Rocq formalization of the main results of the paper. This formalization provides all details about the proofs.

A summary of the relations between the main systems considered in the paper is presented in Figure 1.



■ **Figure 1** Summary of the Main Systems.

2 Preliminaries and Notations

2.1 Formulas

Since we want to relate systems coming from different communities, we face conflicts in notations. We choose linear-logic-style notations for the first part of the paper since it is the most common setting where all the connectives we need are used together (and this also avoids conflicts with classical or intuitionistic logic where properties are really different). Given a set $\mathcal{A}$ of **atoms** denoted X, Y, Z , etc, we consider **formulas** (or types) belonging to: $A, B ::= X \mid A \multimap B \mid B \multimap A \mid A \otimes B \mid 1 \mid A \& B \mid \top \mid A \oplus B \mid 0 \mid \forall X A \mid \exists X A$.

In the second part of the paper, when we restrict the set of connectives and focus on relations with subtyping, we will move to more “type-style” notations. Anyway, here is an equivalence table for connectives:

Types	Lambek	Linear Logic	Types	Lambek	Linear Logic
$A \rightarrow B$	$= B / A$	$= B \multimap A$	$A \cap B$	$= A \wedge B$	$= A \& B$
	$A \setminus B$	$= A \multimap B$	Ω	$= \top$	$= \top$
	$A \bullet B$	$= A \otimes B$	$A \cup B$	$= A \vee B$	$= A \oplus B$
	1	$= 1$	$\emptyset$	$= \perp$	$= 0$
			$\forall X A$	$= \forall X A$	$= \forall X A$
			$\exists X A$	$= \exists X A$	$= \exists X A$

Despite the number of connectives and notations used, note that no ambiguity occurs in this table: when a symbol appears twice, it is always for representing the same constructor. Product or sum types are not considered here as, regarding subtyping, they can be considered as abstract constructors (see for example [8, 5, 19]).

When writing formulas, $\multimap$ and $\rightarrow$ associate to the right while $\multimap$ associates to the left, and the other connectives (in particular $\&$, $\cap$ and $\forall$) associate more than $\multimap$, $\multimap$ and $\rightarrow$.

2.2 Sequents

In most systems we consider, sequents are intuitionistic non-commutative sequents, *i.e.* of the shape $\Gamma \vdash A$ with Γ a *list* of formulas. Regarding list notations, ε is the **empty list**, $|\Gamma|$ is the **length** of Γ , $\bar{\Gamma}$ is the **reverse** of Γ , and if μ is a function from formulas to formulas (for example a unary connective), $\mu\Gamma$ is the **map** construction obtained by applying μ to each element of Γ .

A deduction system satisfies the **sub-formula property** if all the formulas appearing in a derivation are sub-formulas of the formulas of its conclusion sequent. This sub-formula property is often easy to derive in cut-free sequent calculi. In the propositional case, it leads to conservativity properties for sub-systems obtained by restricting the set of connectives (and then a rule admissible in the big system is also admissible in the sub-system).

3 Lambek Calculus

The logical systems we are going to consider share some key (sub)structural properties:

- *intuitionistic*: sequents have the shape $\Gamma \vdash A$;
- *non-commutative*: context are lists, there is no exchange rule;
- *linear*: no (weakening or contraction) structural rule.

The most standard such system is the Lambek calculus [15]. We consider the (full) Lambek calculus L which extends the original one with many connectives (including second-order

■ **Table 1** Second-Order (Full) Lambek Calculus $\mathbf{L}$ (all rules and contexts: cyan, orange and black)
 – System $\mathbf{IS}$ (orange and black parts) – System $\mathbf{IS}_h$ (black parts only). The negative restriction ($\mathbf{nL}$, $\mathbf{nIS}$, $\mathbf{nIS}_h$) of each system is obtained by considering only the content above the horizontal line.

$\frac{}{A \vdash A} ax$	$\frac{}{\Gamma \vdash \top} \top R$	$\frac{\Gamma \vdash A}{\Gamma \vdash \forall X A} \forall R$	$\frac{\text{orange } \Gamma, A[B/X], \Delta \vdash C}{\text{orange } \Gamma, \forall X A, \Delta \vdash C} \forall L$
$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \& B} \& R$	$\frac{\text{orange } \Gamma, A, \Delta \vdash C}{\text{orange } \Gamma, A \& B, \Delta \vdash C} \& L_1$	$\frac{\text{orange } \Gamma, B, \Delta \vdash C}{\text{orange } \Gamma, A \& B, \Delta \vdash C} \& L_2$	
$\frac{\Gamma, A \vdash B}{\Gamma \vdash B \multimap A} \multimap R$	$\frac{\text{cyan } \Gamma \vdash A \quad \text{cyan } \Delta, B, \Sigma \vdash C}{\text{cyan } \Delta, B \multimap A, \Gamma, \Sigma \vdash C} \multimap L$	$\frac{D \vdash A \quad B, \Sigma \vdash C}{B \multimap A, D, \Sigma \vdash C} \multimap S$	
$\frac{\text{cyan } A, \Gamma \vdash B}{\text{cyan } \Gamma \vdash A \multimap B} \multimap R$	$\frac{\text{cyan } \Gamma \vdash A \quad \text{cyan } \Delta, B, \Sigma \vdash C}{\text{cyan } \Delta, \Gamma, A \multimap B, \Sigma \vdash C} \multimap L$		
$\frac{}{\text{orange } \Gamma, 0, \Delta \vdash C} 0L$	$\frac{\Gamma \vdash A[B/X]}{\Gamma \vdash \exists X A} \exists R$	$\frac{\text{orange } \Gamma, A, \Delta \vdash C}{\text{orange } \Gamma, \exists X A, \Delta \vdash C} \exists L$	
$\frac{\Gamma \vdash A}{\Gamma \vdash A \oplus B} \oplus R_1$	$\frac{\Gamma \vdash B}{\Gamma \vdash A \oplus B} \oplus R_2$	$\frac{\text{orange } \Gamma, A, \Delta \vdash C \quad \text{orange } \Gamma, B, \Delta \vdash C}{\text{orange } \Gamma, A \oplus B, \Delta \vdash C} \oplus L$	
$\frac{\text{cyan } \Gamma \vdash A \quad \text{cyan } \Delta \vdash B}{\text{cyan } \Gamma, \Delta \vdash A \otimes B} \otimes R$	$\frac{\text{cyan } \Gamma, A, B, \Delta \vdash C}{\text{cyan } \Gamma, A \otimes B, \Delta \vdash C} \otimes L$	$\frac{}{\vdash 1} 1R$	$\frac{\text{orange } \Gamma, \Delta \vdash C}{\text{orange } \Gamma, 1, \Delta \vdash C} 1L$
with X not free in Γ, Δ and C for the $(\forall R)$ and $(\exists L)$ rules			

quantifiers $\forall$ and $\exists$, see [11] for the propositional part). The obtained system can also be seen as the non-commutative (*i.e.* exchange-free) variant of intuitionistic multiplicative additive linear logic (introduced later by J.-Y. Girard [12]).

We consider here the variant of $\mathbf{L}$ which allows empty left-hand sides in sequents (sometimes denoted $\mathbf{L}^*$). Some additional discussion on that point will appear in Remark 8. The rules of the system are presented on Table 1 (taking into account all colors, and where $(\multimap S)$ is just a particular instance of $(\multimap L)$ to be used later).

The system $\mathbf{L}$ is given without cut rule. A major consequence, which comes from a direct analysis of the shape of the rules, is that $\mathbf{L}$ satisfies the sub-formula property in the propositional case. It is very important for expressiveness that some cut rule remains admissible, so that cut-free provability and provability with cuts coincide.

► **Theorem 1** (Cut Admissibility in $\mathbf{L}$). *The intuitionistic cut rule is admissible in $\mathbf{L}$:*

$$\frac{\Gamma \vdash A \quad \Delta, A, \Sigma \vdash C}{\Delta, \Gamma, \Sigma \vdash C} cut$$

Proof. This result can be found in the literature for the original Lambek calculus [15, Section 9] or for its extension with other propositional connectives [11, Section 4.1.1]. Those proofs rely on induction on the size of cut formulas which cannot be directly adapted here because of the use of second-order quantifiers which may increase the size of cut formulas in the calls to the induction hypothesis. To avoid this problem we want to define a measure on proofs not depending on formulas themselves (this idea is, for example, at the heart of the

cut-elimination proofs for light linear logics [13]). Standard such measures are the *size* of a proof (*i.e.* its number of rules, obtained by summing up the sizes of the two premises at each binary rule), and also the *height* of a proof (*i.e.* the length of its longest branch, obtained by taking the maximum of the heights of the two premises at each binary rule). None of them can be directly used here. This is why we consider a mixed notion (used in [21]), the **weight** $w(\pi)$ of a proof π . As for the *size* and the *height* of a proof, the weight of a 0-ary rule is 1, and we add 1 when we cross a unary rule. The case of binary rules is not uniform anymore. We use the *sum* of the weights plus 1 for the “multiplicative binary rules” ($\otimes R$), ($\multimap L$) and ($\multimap R$). We use the *max* of the weights plus 1 for the “additive binary rules” ($\& R$) and ($\oplus L$).

$$w\left(\frac{\frac{\pi_1}{\Gamma \vdash A} \quad \frac{\pi_2}{\Delta, B, \Sigma \vdash C}}{\Delta, B \multimap A, \Gamma, \Sigma \vdash C} \multimap L\right) = w(\pi_1) + w(\pi_2) + 1$$

$$w\left(\frac{\frac{\pi_1}{\Gamma \vdash A} \quad \frac{\pi_2}{\Gamma \vdash B}}{\Gamma \vdash A \& B} \& R\right) = \max(w(\pi_1), w(\pi_2)) + 1$$

Now the theorem is obtained by showing that, given a proof π_1 of $\Gamma \vdash A$ and a proof π_2 of $\Delta, A, \Sigma \vdash C$, we can build a proof π of $\Delta, \Gamma, \Sigma \vdash C$ with $w(\pi) < w(\pi_1) + w(\pi_2)$. This is done by induction on $w(\pi_1) + w(\pi_2)$. The required local cut-reduction steps are included in the Rocq formalization (see Appendix A). ◀

Here is a typical application of cut admissibility.

► **Lemma 2** (Invertibility/Reversibility of $\multimap$). *If $\Gamma \vdash B \multimap A$ is provable in $\mathbf{L}$, then $\Gamma, A \vdash B$ as well.*

Proof. Using Theorem 1, we have:

$$\frac{\Gamma \vdash B \multimap A \quad \frac{\frac{A \vdash A}{B \multimap A, A \vdash B} \multimap L \quad \frac{B \vdash B}{\Gamma, A \vdash B} ax}{\Gamma, A \vdash B} cut}{\Gamma, A \vdash B} cut$$

(we use dashed lines for admissible rules which are not explicitly part of the system). ◀

3.1 The S-constraint

The key ingredient of the present work, which we call the **S-constraint** (with “S” for *subtyping*), is to restrict the ($\multimap L$) rule of $\mathbf{L}$ to the particular case ($\multimap S$) where Γ is a singleton and Δ is empty:

$$\frac{D \vdash A \quad B, \Sigma \vdash C}{B \multimap A, D, \Sigma \vdash C} \multimap S$$

The (*cut*) rule is no longer admissible in the induced system (see Example 3).

► **Example 3.** We have:

$$\frac{\frac{\frac{A \vdash A}{B \multimap A, A \vdash B} \multimap S \quad \frac{B \vdash B}{C \multimap B, B \vdash C} \multimap S}{C \multimap B, B \multimap A, A \vdash C} cut \quad \frac{C \multimap B, B \multimap A, A \vdash C}{C \multimap B, B \multimap A \vdash C \multimap A} \multimap R}{C \multimap B \vdash C \multimap A \multimap (B \multimap A)} \multimap R$$

but $C \multimap B \vdash C \multimap A \multimap (B \multimap A)$ has no cut-free proof in $\mathbf{L}$ which satisfies the S-constraint (see Example 12).

In order to define a meaningful restriction of $\mathbf{L}$ based on the S-constraint, we then have to be careful. This is why we define the logical **system** $\mathbf{IS}$ to be the restriction of $\mathbf{L}$:

- without the connectives $\otimes$ and $\multimap$;
- with $(\multimap L)$ replaced with its restriction $(\multimap S)$.

The rules of $\mathbf{IS}$ are obtained from Table 1 by *removing* the *cyan* rules (in particular the $(\multimap L)$ rule is removed but its restriction $(\multimap S)$ remains). Even if the (cut) rule is not admissible in $\mathbf{IS}$ (Example 3), a restriction called $(Scut)$ is however admissible:

$$\frac{\Gamma \vdash A \quad \Delta, A, \Sigma \vdash C \quad \Delta \neq \varepsilon \Rightarrow |\Gamma| = 1}{\Delta, \Gamma, \Sigma \vdash C} Scut$$

While being less powerful than plain (cut) because of the side condition, $(Scut)$ still has important particular cases such as:

$$\frac{D \vdash A \quad \Delta, A, \Sigma \vdash C}{\Delta, D, \Sigma \vdash C} scut \quad \frac{\Gamma \vdash A \quad A, \Sigma \vdash C}{\Gamma, \Sigma \vdash C} tcut \quad \frac{D \vdash A \quad A \vdash C}{D \vdash C} trans$$

which make it powerful enough in practice for the system $\mathbf{IS}$ to have nice properties. The “s” in $(scut)$ is for *substitution cut*. The “t” in $(tcut)$ is for *transitivity cut*. Both contain $(trans)$ which refers to transitivity of the (pre)order relation on formulas induced by provability. In practice, the goal is to have $(trans)$ as a property of subtyping. But when trying to prove its admissibility, we need to strengthen the induction hypothesis to both $(scut)$ and $(tcut)$. Then, for controlling the number of cases to consider, it is comfortable to factorize them through $(Scut)$. The usual approach would be to use the full intuitionistic (cut) but it is not admissible here (see Example 3).

► **Theorem 4** (Cut Admissibility in $\mathbf{IS}$). *The $(Scut)$ rule is admissible in $\mathbf{IS}$.*

Proof. The proof happens to be the same as for Theorem 1! Indeed, when restricted to $(Scut)$, the transformation steps used for Theorem 1 happen to preserve the S-constraint and to require only induction hypotheses satisfying the side condition $\Delta \neq \varepsilon \Rightarrow |\Gamma| = 1$. The very nice interaction between the S-constraint and $(Scut)$ is the key ingredient of $\mathbf{IS}$, and is made possible by the absence of $\otimes$ and $\multimap$ in $\mathbf{IS}$. ◀

It is not so common to find a structural restriction of a logical system which is stable under cut elimination (let us mention the restriction to intuitionistic logic inside the sequent calculus of classical logic or the η -constraint in $\mathbf{LK}^{tq}$ [9]).

The S-constraint obviously impacts provability. Typical sequents which are provable in $\mathbf{L}$ but not in $\mathbf{IS}$ are: $B \multimap (A \multimap A) \vdash B$ and $C \multimap B \vdash C \multimap A \multimap (B \multimap A)$ (see Example 12).

3.2 The Leftmost Restriction of $\mathbf{IS}$

Half of the S-constraint is to require that we only apply the left rule for $\multimap$ to the leftmost formula of the sequent. We consider now the generalization of this restriction to every left rule. Take this as an intellectual exercise for now, the reason for considering this constraint will become natural in Section 3.3 (see Proposition 7).

The induced system $\mathbf{IS}_h$ (with “h” for *head* since left rules are applied to the head of the context list only) is presented on Table 1, where one considers *black parts only*: the *cyan* rules are removed as well as the *orange* contexts. This new constraint is compatible with cut admissibility.

► **Theorem 5** (Cut Admissibility). *The $(Scut)$ rule is admissible in $\mathbf{IS}_h$.*

Proof. Same as for Theorem 4, since the constraint of IS_h (no orange context) is preserved under all the cut-reduction steps. $\blacktriangleleft$

Note that, by moving from IS to IS_h , we loose some provability.

► **Example 6.** $(C \multimap A) \& (C \multimap B) \vdash C \multimap (A \oplus B) \& D$ is provable in IS but not in IS_h . We have in IS :

$$\frac{\frac{\frac{\overline{A \vdash A} \text{ ax}}{C \multimap A, A \vdash C} \multimap S \quad \frac{\overline{C \vdash C} \text{ ax}}{(C \multimap A) \& (C \multimap B), A \vdash C} \& L_1 \quad \frac{\frac{\overline{B \vdash B} \text{ ax}}{C \multimap B, B \vdash C} \multimap S \quad \frac{\overline{C \vdash C} \text{ ax}}{(C \multimap A) \& (C \multimap B), B \vdash C} \& L_2}{(C \multimap A) \& (C \multimap B), A \oplus B \vdash C} \oplus L}{(C \multimap A) \& (C \multimap B), (A \oplus B) \& D \vdash C} \& L_1}{(C \multimap A) \& (C \multimap B) \vdash C \multimap ((A \oplus B) \& D)} \multimap R$$

but proving $(C \multimap A) \& (C \multimap B), (A \oplus B) \& D \vdash C$ in IS_h would require to apply a rule on the leftmost $\&$, leading to $C \multimap A, (A \oplus B) \& D \vdash C$ or to $C \multimap B, (A \oplus B) \& D \vdash C$ which are both not provable (they are not even provable in linear logic).

3.3 Negative Fragments

From the focusing theory of linear logic [2], we know that connectives can be split into two categories: the **negative** ones with right-hand side invertible/reversible rules and the **positive** ones satisfying the focusing property on the right-hand side of sequents. Applied to L , this dichotomy gives us $\multimap$, $\multimap$, $\&$, $\top$ and $\forall$ as negative connectives and $\otimes$, 1 , $\oplus$, 0 and $\exists$ as positive ones.

In order to get additional proof-theoretical properties in the systems considered so far, we now look at restricting them to their fragments (*i.e.* restrictions on formulas) using negative connectives only. Because of the non-commutative aspects of the systems under study, it happens that dropping not only $\otimes$, 1 , $\oplus$, 0 and $\exists$ but also $\multimap$ gives even stronger properties (and is expressive enough for the relations we are going to draw with systems from the literature). We then consider nL (resp. nIS , etc.) as the restriction of L (resp. IS , etc.) using only $\multimap$, $\&$, $\top$ and $\forall$ as connectives (*i.e.* taking into account only the content above the horizontal line in Table 1). As a first consequence, it happens that the leftmost restriction (Section 3.2) does not impact provability anymore in nIS .

► **Proposition 7.** nIS and nIS_h prove exactly the same sequents.

Proof. The rules of nIS are admissible in nIS_h thanks to Theorem 5:

$$\frac{\frac{\overline{A \vdash A} \text{ ax}}{A \& B \vdash A} \& L_1 \quad \Gamma, A, \Delta \vdash C}{\Gamma, A \& B, \Delta \vdash C} \quad \frac{\frac{\overline{A[B/X] \vdash A[B/X]} \text{ ax}}{\forall X A \vdash A[B/X]} \forall L \quad \Gamma, A[B/X], \Delta \vdash C}{\Gamma, \forall X A, \Delta \vdash C} \forall L \quad \blacktriangleleft$$

► **Remark 8 (Empty left-hand sides).** Remember the Lambek calculus was first introduced with the constraint that the left-hand sides of sequents should never be empty [15]. This makes in particular $B \multimap (A \multimap A) \vdash B$ not provable, meaning that the variant allowing empty left-hand sides is not conservative over the original one. In the system IS , the situation is different.

The only rule of IS which may have a premise with an empty left-hand side and a conclusion with a non-empty left-hand side is $(1L)$, while in the case of L this is also possible with the $(\multimap L)$ rule. It is possible to prove that IS is conservative over its restriction using

■ **Table 2** System $\mathbf{nIS}_1$.

$\frac{}{A \triangleleft \vdash A} ax$	$\frac{C \triangleleft \Gamma, A \vdash B}{C \triangleleft \Gamma \vdash B \multimap A} \multimap R$	$\frac{D \triangleleft \vdash A \quad B \triangleleft \Sigma \vdash C}{B \multimap A \triangleleft D, \Sigma \vdash C} \multimap S$
$\frac{C \triangleleft \Gamma \vdash A \quad C \triangleleft \Gamma \vdash B}{C \triangleleft \Gamma \vdash A \& B} \& R$	$\frac{A_i \triangleleft \Delta \vdash C}{A_1 \& A_2 \triangleleft \Delta \vdash C} \& L_i$	$\frac{}{C \triangleleft \Gamma \vdash \top} \top R$
$\frac{C \triangleleft \Gamma \vdash A}{C \triangleleft \Gamma \vdash \forall X A} \forall R \quad (X \text{ not free in } C, \Gamma)$	$\frac{A[B/x] \triangleleft \Delta \vdash C}{\forall X A \triangleleft \Delta \vdash C} \forall L$	

sequents with non-empty left-hand sides only: they both prove the same sequents $\Gamma \vdash A$ as soon as Γ is not empty. In the case of the negative fragments ($\mathbf{nIS}$ or $\mathbf{nIS}_h$), the $(1L)$ rule is not here anymore and a proof of a sequent with non-empty left-hand side contains only sequents with non-empty left-hand sides.

The only rules of $\mathbf{nIS}$ which may have a conclusion with an empty left-hand side while all their premises have non-empty left-hand sides are $(\top R)$ (when it introduces a conclusion $\vdash \top$) and $(\multimap R)$ (when it moves from $B \vdash C$ to $\vdash C \multimap B$). By controlling these two rules, it is then possible to ensure that all sequents have non-empty left-hand sides.

In order to have direct links with subtyping systems (*i.e.* $A, B_1, \dots, B_n \vdash B$ interpreted as $A \leq B_1 \rightarrow \dots \rightarrow B_n \rightarrow B$), we will mostly restrict proofs to sequents with non-empty left-hand sides (an exception being Theorem 27).

We denote by $\mathbf{nIS}_1$ the restriction of $\mathbf{nIS}_h$ to proofs containing only sequents with at least one formula on their left-hand side. Following Remark 8, it is the same as restricting conclusion sequents of proofs to have non-empty left-hand sides. In particular, no sequent $\vdash A$ is provable in $\mathbf{nIS}_1$, and a sequent $\Gamma \vdash A$ with non-empty Γ is provable in $\mathbf{nIS}_1$ if and only if it is provable in $\mathbf{nIS}_h$ if and only if it is provable in $\mathbf{nIS}$. Moreover we get that $(Scut)$ is admissible in $\mathbf{nIS}_1$.

Using Remark 8 again, the rules of $\mathbf{nIS}_1$ are obtained from those of $\mathbf{nIS}_h$ by restricting the $(\top R)$ and $(\multimap R)$ rules only. In $\mathbf{nIS}_1$, each sequent can be written $A, \Gamma \vdash B$ with A and B formulas, and Γ a (possibly empty) list of formulas. Moreover, $\mathbf{nIS}_1$ being a restriction of $\mathbf{nIS}_h$, left rules are applied on A only (never in Γ). To stress this particular status of A in sequents, we will often write them $A \triangleleft \Gamma \vdash B$ (which is just a notation for $A, \Gamma \vdash B$). The rules of $\mathbf{nIS}_1$ are recalled on Table 2 with this notation used. Note, in the $(\multimap R)$ rule, the formula moving to the right (when reading the rule top-down) cannot be the leftmost formula of the premise which is how we ensure that we do not have a conclusion with an empty left-hand side.

3.4 Reversing and Focusing

Going one step further in the analysis of the structure of proofs in $\mathbf{nL}$ and $\mathbf{nIS}$, we now consider focusing [2]. Very roughly, the operation of turning a proof into a focused one can be decomposed into two transformations (see for example [17]): moving all the invertible/reversible rules down (*reversing*) and grouping positive rules together (*focusing*). In the systems $\mathbf{nL}$ and $\mathbf{nIS}$, which contain only negative connectives, the invertible/reversible rules are the right ones (and the positive rules are the left ones). In such an intuitionistic system, the result of reversing a proof (*i.e.* of moving the reversible rules down) can be characterised by the constraint that all left rules have an atom as right-hand side of their conclusion sequent.

■ **Table 3** Reversed Negative Lambek Calculus $\mathbf{nL}^{\text{rev}}$.

$\frac{}{Z \vdash Z} ax$	$\frac{\Gamma, A \vdash B}{\Gamma \vdash B \multimap A} \multimap R$	$\frac{\Gamma \vdash A \quad B, \Sigma \vdash Z}{B \multimap A, \Gamma, \Sigma \vdash Z} \multimap L$
$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \& B} \& R$	$\frac{A_i, \Delta \vdash Z}{A_1 \& A_2, \Delta \vdash Z} \& L_i$	$\frac{}{\Gamma \vdash \top} \top R$
$\frac{\Gamma \vdash A}{\Gamma \vdash \forall X A} \forall R$ (X not free in Γ)	$\frac{A[B/X], \Delta \vdash Z}{\forall X A, \Delta \vdash Z} \forall L$	

■ **Table 4** System $\mathbf{nIS}_1^{\text{rev}}$.

$\frac{}{C \triangleleft \vdash Z} ax$	$\frac{C \triangleleft \Gamma, A \vdash B}{C \triangleleft \Gamma \vdash B \multimap A} \multimap R$	$\frac{D \triangleleft \vdash A \quad B \triangleleft \Sigma \vdash Z}{B \multimap A \triangleleft D, \Sigma \vdash Z} \multimap S$
$\frac{C \triangleleft \Gamma \vdash A \quad C \triangleleft \Gamma \vdash B}{C \triangleleft \Gamma \vdash A \& B} \& R$	$\frac{A_i \triangleleft \Delta \vdash Z}{A_1 \& A_2 \triangleleft \Delta \vdash Z} \& L_i$	$\frac{}{C \triangleleft \Gamma \vdash \top} \top R$
$\frac{C \triangleleft \Gamma \vdash A}{C \triangleleft \Gamma \vdash \forall X A} \forall R$ (X not free in C, Γ)	$\frac{A[B/X] \triangleleft \Delta \vdash Z}{\forall X A \triangleleft \Delta \vdash Z} \forall L$	

This comes naturally with the fact of restricting the (ax) rule in the same way. For systems satisfying the leftmost restriction (Section 3.2), it then happens that positive rules cannot commute, so we get that reversing leads to focusing (somehow for free).

Formally, we consider the following **reversing constraint**: *the (ax) rule is restricted to atoms, and all the left rules have their main formula at the leftmost position in the sequent and an atom as right-hand side formula*. This reversing constraint can be applied to $\mathbf{nL}$, $\mathbf{nIS}$, $\mathbf{nIS}_h$ and $\mathbf{nIS}_1$, leading to $\mathbf{nL}^{\text{rev}}$, $\mathbf{nS}^{\text{rev}}$, $\mathbf{nS}_h^{\text{rev}}$ and $\mathbf{nIS}_1^{\text{rev}}$ (see Figure 1). Note that $\mathbf{nS}^{\text{rev}} = \mathbf{nS}_h^{\text{rev}}$ (since the leftmost restriction is included in the reversing constraint). We present explicitly the two extremal cases $\mathbf{nL}^{\text{rev}}$ (Table 3) and $\mathbf{nIS}_1^{\text{rev}}$ (Table 4) (where Z denotes an atom).

We want to show that $\mathbf{nL}$ and $\mathbf{nL}^{\text{rev}}$ (resp. $\mathbf{nIS}_1$ and $\mathbf{nIS}_1^{\text{rev}}$, etc.) have the same provability. One direction is immediate since the reversed system is included in the original one. In the opposite direction, we need some admissibility proofs. Once the equivalence is settled, we can derive cut admissibility in the reversed systems.

► **Proposition 9.** *The rules of $\mathbf{nL}$ are admissible in $\mathbf{nL}^{\text{rev}}$.*

Proof. Regarding the left rules, let us consider the case of ($\multimap L$). If we have a proof π_1 of $\Gamma \vdash A$ and a proof π_2 of $\Delta, B, \Sigma \vdash C$, we go by induction on π_2 to derive $\Delta, B \multimap A, \Gamma, \Sigma \vdash C$. We consider the case of each possible last rule for π_2 . For example if π_2 is obtained from π'_2 by applying a ($\multimap R$) rule, we can build:

$$\frac{\frac{\pi_1 \quad \pi'_2}{\Gamma \vdash A \quad \Delta, B, \Sigma, E \vdash D} \text{ IH}}{\Delta, B \multimap A, \Gamma, \Sigma \vdash E \multimap D} \multimap R$$

Using the admissibility of the left rules, we can now prove the case of (ax) by a simple induction on A . ◀

► **Proposition 10** (Cut Admissibility for nL^{rev}). *The (cut) rule is admissible in nL^{rev} .*

► **Proposition 11.** *The rules of nIS_1 as well as the (Scut) rule are admissible in $\text{nIS}_1^{\text{rev}}$.*

► **Example 12.** $B \multimap (A \multimap A) \triangleleft \vdash B$ is not provable in $\text{nIS}_1^{\text{rev}}$ when A and B are atoms: there is no proof since there is no possible last rule. Similarly, $C \multimap B \triangleleft \vdash C \multimap A \multimap (B \multimap A)$ is not provable in $\text{nIS}_1^{\text{rev}}$ when A , B and C are atoms. A proof would end with:

$$\frac{\frac{\frac{B \multimap A \triangleleft \vdash B}{C \multimap B \triangleleft B \multimap A, A \vdash C} \multimap S}{C \multimap B \triangleleft B \multimap A \vdash C \multimap A} \multimap R}{C \multimap B \triangleleft \vdash C \multimap A \multimap (B \multimap A)} \multimap R$$

but both top sequents are not provable.

Assuming A , B and C are atoms, the two considered sequents have at least one formula on the left-hand side and do not contain any quantifier thus we have:

$$\begin{aligned} \text{provable in } \text{nIS}_1^{\text{rev}} &\iff \text{provable in } \text{nIS}_1 && (\text{Proposition 11}) \\ &\iff \text{provable in } \text{nIS}_h && (\text{Remark 8}) \\ &\iff \text{provable in } \text{nIS} && (\text{Proposition 7}) \\ &\iff \text{provable in } \text{IS} && (\text{sub-formula property}) \end{aligned}$$

This means $B \multimap (A \multimap A) \vdash B$ and $C \multimap B \vdash C \multimap A \multimap (B \multimap A)$ are not provable in IS .

We have met many constraints used to define various sub-systems of L . A global map of the relations is presented in Figure 1. We can find in particular the path used in Example 12: a negative quantifier-free sequent with at least one formula on its left-hand side is provable in IS if and only if it is provable in $\text{nIS}_1^{\text{rev}}$.

It is now time to draw the bridge with subtyping! We have seen that a sequent $A \vdash B$ is derivable in nIS if and only if it is derivable in $\text{nIS}_1^{\text{rev}}$. Let us now interpret $A \vdash B$ as a subtyping relation $A \leq B$. By restricting the language to $\multimap$, $\&$ and $\top$ (which defines the fragment $\cap\text{IS}$ of IS) on one side and to $\multimap$ and $\forall$ (which defines the fragment $\forall\text{IS}$ of IS) on the other side, we get subtyping relations studied in the literature in the setting of intersection types and of polymorphic subtyping (we have not found research results mixing intersections and polymorphism in this kind of structural study of subtyping). To make the link with the literature simpler, we now move to notations from type theory (see Section 2.1):

$$B \multimap A \mapsto A \rightarrow B, \quad A \& B \mapsto A \cap B, \quad \top \mapsto \Omega, \quad \forall X A \mapsto \forall X A.$$

A sequent $A \triangleleft B_1, \dots, B_n \vdash B$ can be understood as representing $A \leq B_1 \rightarrow \dots \rightarrow B_n \rightarrow B$.

We will consider in Section 4 the case of intersection types and $\cap\text{IS}$. In Section 5, we will move to polymorphic subtyping and consider $\forall\text{IS}$. As an application of Proposition 11, we get conservativity results for $\cap\text{IS}$ and $\forall\text{IS}$ (see Figure 1 for a more complete picture):

► **Theorem 13.** $A \vdash B$ in $\cap\text{IS}$ (resp. $\forall\text{IS}$) if and only if $A \vdash B$ in $\cap\text{IS}_1^{\text{rev}}$ (resp. $\forall\text{IS}_1^{\text{rev}}$).

4 Intersection Types

Regarding intersection types, we will mostly focus on the BCD intersection type system [3] but, in order to stress the key role played by the subtyping relation, we first consider a slightly more general notion of *intersection type system* where subtyping appears as a parameter. We refer to [1] and [4, Chapters 12–14], for a more detailed (and more general) presentation of this approach.

■ **Table 5** Bounded Meet-Semilattice.

$\frac{}{A \leq A}$	$\frac{A \leq B \quad B \leq C}{A \leq C}$	$\frac{A \leq C \quad B \leq D}{A \cap B \leq C \cap D}$	$\frac{}{A \leq \Omega}$
$\frac{}{A \cap B \leq A}$	$\frac{}{A \cap B \leq B}$	$\frac{}{A \leq A \cap A}$	

■ **Table 6** Intersection Typing System.

$\frac{}{\rho, x : A \vdash x : A} \text{ var}$	$\frac{\rho, x : A \vdash t : B}{\rho \vdash \lambda x. t : A \rightarrow B} \text{ abs}$	$\frac{\rho \vdash t : A \rightarrow B \quad \rho \vdash u : A}{\rho \vdash t u : B} \text{ app}$
$\frac{\rho \vdash t : A \quad \rho \vdash t : B}{\rho \vdash t : A \cap B} \text{ inter}$	$\frac{}{\rho \vdash t : \Omega} \text{ omg}$	$\frac{\rho \vdash t : A \quad A \leq B}{\rho \vdash t : B} \text{ sub}$

4.1 Bounded Meet-Semilattices

As pre-orders (reflexive transitive or irreflexive transitive relations) appear more natural than orders here, we will almost always forget the prefix “pre” (we use “lattice” instead of “pre-lattice”, etc). A **bounded meet-semilattice** is a pre-order with binary meets (denoted with $\cap$) and a top element (denoted Ω). In a bounded meet-semilattice, all finite meets exist. Table 5 provides an axiomatic presentation of bounded meet-semilattices. We use the notation $A \sim B$ if both $A \leq B$ and $B \leq A$ are derivable.

Using Table 5, one can prove that, in any bounded meet-semilattice, $\cap$ is associative, commutative and idempotent, with Ω as unit. If I is a (possibly empty) finite set $\{i_1, \dots, i_n\}$, we use the notation $\bigcap_{i \in I} A_i$ to represent the equivalence class of $A_{i_1} \cap (A_{i_2} \cap \dots (A_{i_n} \cap \Omega))$ with respect to associativity, commutativity and unitality (we do not take idempotency into account here). This will be used to state results and properties which do not depend on a specific choice of an element of this class. This is in particular the case for results valid up to $\sim$ (see β -condition, η -condition or Theorem 26 below).

4.2 Intersection Type Systems

We are interested in typing systems for the λ -calculus where types are the formulas of $\cap$ IS:

$$A ::= X \mid \Omega \mid A \cap A \mid A \rightarrow A$$

Terms are usual λ -terms and typing judgements are of the shape $\rho \vdash t : A$ where ρ is a mapping from term variables to types with finite support, t is a λ -term and A is a type.

We now assume given a bounded meet-semilattice structure $\mathcal{T}$ on types such that $\cap$ defines (binary) meets and Ω is the top element. $\mathcal{T}$ is the **intersection subtyping** relation. The typing rules are presented on Table 6, where $\leq$ denotes the order relation induced by $\mathcal{T}$ (and remember we use $\sim$ for the induced equivalence relation). The first three rules are exactly those of the simply typed λ -calculus and the (*sub*) rule is the only one depending on $\mathcal{T}$.

It is known from the literature (see [4, 18] for example), that **subject β -expansion** holds for any $\mathcal{T}$:

$$\text{if } t \text{ } \beta\text{-reduces to } u \text{ then } \rho \vdash u : A \implies \rho \vdash t : A$$

Assuming moreover that the properties of Tables 5 and 7 hold for $\mathcal{T}$ (*i.e.* that axioms hold and rules are admissible in $\mathcal{T}$), then we have **subject η -reduction**:

$$\text{if } t \text{ } \eta\text{-reduces to } u \text{ then } \rho \vdash t : A \implies \rho \vdash u : A$$

Additional assumptions on the parameter $\mathcal{T}$ lead to subject β -reduction and/or subject η -expansion, thus leading to models of β -conversion and/or η -conversion. It is a particularly nice consequence that it is possible to focus on the properties of $\mathcal{T}$ only, without coming back to the typing rules themselves. This moves the analysis from intersection typing to intersection subtyping, thus stressing the key role of subtyping in systems with intersection types. In the next two sections, we recall key conditions we can ask about $\mathcal{T}$, then we focus on subtyping.

4.3 The β -Condition

The **β -condition** is the following property:

$$\bigcap_{i \in I} A_i \rightarrow B_i \leq A \rightarrow B \implies \exists J \subseteq I, A \leq \bigcap_{j \in J} A_j \wedge \bigcap_{j \in J} B_j \leq B \quad (\beta)$$

Note that both I and J may be empty sets in (β) . The exact formulation of this condition (also called *β -soundness*) may vary in the literature. We rely here on the fact that we are only interested in theories validating the properties of Tables 5 and 7.

The β -condition provides a sufficient (but not necessary) condition for *subject β -reduction* to hold (see for example [4]).

► **Proposition 14** (Subject β -Reduction). *If $\mathcal{T}$ satisfies the properties of Tables 5 and 7 and the β -condition, then subject β -reduction holds.*

4.4 The η -Condition

The **η -condition** is the following property:

$$\forall X \in \mathcal{A}, \exists I, \exists (A_i)_{i \in I}, \exists (B_i)_{i \in I}, X \sim \bigcap_{i \in I} A_i \rightarrow B_i \quad (\eta)$$

Again, the exact formulation of this condition (also called *η -soundness*) may vary in the literature. We rely here on the fact that we are only interested in theories validating the properties of Tables 5 and 7.

The η -condition provides a necessary and sufficient condition for *subject η -expansion* to hold (see for example [4]).

► **Proposition 15** (Subject η -Expansion). *If $\mathcal{T}$ satisfies the properties of Tables 5 and 7 then the η -condition holds if and only if subject η -expansion holds.*

The β - and η -conditions provide modularity in the study of (BCD-style) intersection type systems: one can focus on the subtyping part independently of the typing part, and use Propositions 14 and 15 to get properties of typing from subtyping. From now on, we will thus focus on the study of the subtyping relation. Our goal is to provide simple presentations of the BCD subtyping relation described on Tables 5 and 7, so that the β -condition should come very easily. We are mostly interested in the BCD system which does not validate subject η -expansion. Section 4.8 will more quickly present two variants for which subject $\beta\eta$ -conversion holds.

■ **Table 7** BCD Subtyping.

$\frac{C \leq A \quad B \leq D}{A \rightarrow B \leq C \rightarrow D}$	$\frac{}{(C \rightarrow A) \cap (C \rightarrow B) \leq C \rightarrow (A \cap B)}$	$\frac{}{\Omega \leq \Omega \rightarrow \Omega}$
--	---	--

4.5 The BCD Subtyping Relation

The **BCD typing system** is the instance of the system of Section 4.2 given by the subtyping relation defined by *exactly* the axioms and rules of Tables 5 and 7 (called the **BCD subtyping relation**).

► **Example 16.** From Tables 5 and 7 we can deduce $\Omega \sim \Omega \rightarrow \Omega$ (since both inequalities are provided). BCD subtyping also contains $(C \rightarrow A) \cap (C \rightarrow B) \leq C \rightarrow (A \cap B)$, which can be extended to $(C \rightarrow A) \cap (C \rightarrow B) \sim C \rightarrow (A \cap B)$ since the converse inequality is derivable.

In the spirit of Sections 4.2–4.4, the following properties of the BCD system are well known (see [3, Lemma 2.4(ii)] and [5, Lemma 3.12]).

► **Lemma 17** (β -Condition for BCD Subtyping [3]). *The BCD subtyping system satisfies the β -condition.*

► **Lemma 18** (η -Condition and BCD Subtyping). *The η -condition is not satisfied in the BCD subtyping system.*

The proof of Lemma 17 cannot be obtained directly by induction on a derivation of $\bigcap_{i \in I} A_i \rightarrow B_i \leq A \rightarrow B$ because of the transitivity rule (which in particular leads to the loss of the sub-formula property). The induction hypothesis then needs to be strengthened. In order to gain modularity in the analysis of the BCD subtyping relation, but also for defining algorithms to check subtyping, alternative presentations have been proposed [26, 7]. They rely on cut-free sequent-calculus-style descriptions. This approach can be traced back to [35] in the restricted case of bounded meet-semilattices.

While we followed a different path, it is important to notice that the system proposed in [7, Figure 12] directly corresponds to $\cap \text{IS}_1^{\text{rev}}$ (Table 4 without the quantifier rules). Let us make two remarks here. First, following [26], transitivity of the subtyping relation in [7] is proved by introducing an auxiliary notion of *reflexive supertypes* which looks much less modular than our generic approach by cut admissibility using standard techniques from proof theory. Second, one of our motivations was to be able to rely on the sub-formula property to prove properties of the subtyping relation (such as the β -condition) and to get results on λ -models, while [26, 7] were more interested in the algorithmic computation of the relation.

► **Theorem 19.** $A \leq B$ in BCD if and only if $A \triangleleft \vdash B$ in $\cap \text{IS}_1^{\text{rev}}$.

Proof. We can replace the reflexivity and transitivity proofs of [7] by Proposition 11. ◀

As an application, it is nice to see that the β -condition can be derived directly by induction on a cut-free derivation in a system like $\cap \text{IS}$ (or $\cap \text{IS}_1^{\text{rev}}$).

► **Lemma 20** (β -Condition for $\cap \text{IS}_1^{\text{rev}}$). *The following property holds in $\cap \text{IS}_1^{\text{rev}}$:*

$$\bigcap_{i \in I} A_i \rightarrow B_i \triangleleft A, \Gamma \vdash B \implies \exists J \subseteq I, A \triangleleft \vdash \bigcap_{j \in J} A_j \wedge \bigcap_{j \in J} B_j \triangleleft \Gamma \vdash B$$

The β -condition from Section 4.3 corresponds to the particular case of an empty Γ .

On the algorithmic side, [26] shows how $\cap\text{IS}_1^{\text{rev}}$ can be directly turned into an algorithm for deciding $A \leq B$. It corresponds to running a backward (*i.e.* bottom-up) proof search in this focused proof system. While there is a known quadratic time upper bound [6], the exact complexity of the problem of deciding BCD subtyping remains open, as far as we know. Let us mention, following a suggestion by Andrej Dudenhefner, that one can interpret the proof-search algorithm from the perspective of an alternating Turing machine (see also [10, Figure 2]) using only a logarithmic space. This gives again a polynomial time upper bound since $\text{ALOGSPACE} = \text{PTIME}$. The logarithmic space bound comes from the fact that there is an injection from sequents which may appear in a (bottom-up tentative) derivation of $A \triangleleft \vdash B$ in $\cap\text{IS}_1^{\text{rev}}$, into pairs of occurrences of a sub-formula of A and a sub-formula of B (see Appendix B).

We are now going to investigate relations between $\cap\text{IS}$ and other logical systems. First, in Section 4.6, we show how $\cap\text{IS}$, which is a restriction of L , can be faithfully embedded in it. Second, taking inspiration from [34], we draw a connection with the relevant logic $\text{B}+$ in Section 4.7.

4.6 From $\cap\text{IS}_1$ back to Lambek

We have already seen sequents, like $B \rightarrow C \vdash (A \rightarrow B) \rightarrow A \rightarrow C$, which are provable in L but not in $\cap\text{IS}$ (Examples 3 and 12). While IS and L share the property of being non-commutative linear intuitionistic systems, in IS (and even more in IS_h) we have added restrictions on left rules, notably by applying them only to the leftmost formula of sequents. It is natural to try to understand if such a constraint can be internalised in L , that is to look for a translation of $\cap\text{IS}_1$ into L which would be conservative (as already recalled, the trivial identity embedding is not). We now present such a conservative embedding from $\cap\text{IS}_1$ into L .

If R is a formula, we use the notation $\neg_R A$ for $A \rightarrow R$. Given two formulas R and S , we define:

$$\begin{aligned} X^\bullet &= X & (A \& B)^\bullet &= A^\bullet \& B^\bullet \\ (A \rightarrow B)^\bullet &= \neg_R \neg_S A^\bullet \rightarrow B^\bullet & \top^\bullet &= \top \end{aligned}$$

and $(A \triangleleft \Gamma \vdash B)^\bullet = A^\bullet, \neg_R \neg_S \Gamma^\bullet \vdash B^\bullet$. Note that, since the Lambek calculus satisfies the sub-formula property, a proof of a sequent of the shape $A^\bullet, \neg_R \neg_S \Gamma^\bullet \vdash B^\bullet$ does not contain any use of $\multimap$, $\otimes$, 1 , $\oplus$, 0 , $\forall$ or $\exists$. Also we can see there is no A and B such that $A^\bullet = \neg_R \neg_S B^\bullet$ or $A^\bullet = \neg_S B^\bullet$ (since 3 divides the number of $\rightarrow$ in both $A^\bullet$ and $B^\bullet$).

This translation is inspired by Girard's translation [12] from intuitionistic logic to linear logic: $A \Rightarrow B \mapsto \Box A \multimap B$. In Girard's case $\Box := !$, here we use a different modality $\Box := \neg_R \neg_S$, which satisfies the admissibility of the rule $\frac{A \vdash B}{\Box A \vdash \Box B} \Box$ (a property required for correctness):

$$\frac{\frac{\frac{A \vdash B}{\neg_S B, A \vdash S} \rightarrow S}{\neg_S B \vdash \neg_S A} \rightarrow R \quad \frac{R \vdash R}{R \vdash R} ax}{\frac{\neg_R \neg_S A, \neg_S B \vdash R}{\neg_R \neg_S A \vdash \neg_R \neg_S B} \rightarrow R} \rightarrow S$$

A key part of the conservativity result will be to prove that this admissible rule is invertible (see Lemma 22).

► **Proposition 21** (Correctness of $(_)\bullet$). *For all R and S , if $A \triangleleft \Gamma \vdash B$ is provable in $\cap\mathcal{IS}_1$ then $A^\bullet, \neg_R \neg_S \Gamma^\bullet \vdash B^\bullet$ is provable in $\mathcal{L}$.*

Regarding the converse direction (Proposition 24), we first need a few lemmas. We now assume R and S are atoms.

► **Lemma 22.** *If $R \neq S$ and $\neg_R \neg_S \Gamma, \neg_S A, \Delta \vdash X$ is provable in $\mathcal{L}$ then $\Delta, \Gamma \vdash A$ and Γ is empty if $X = S$ and Γ is a singleton if $X = R$.*

► **Lemma 23.** *Assuming that R and S do not occur in Γ and Δ , if $\neg_R \neg_S \Gamma^\bullet, \Delta^\bullet \vdash X$ is provable in $\mathcal{L}$ then X is neither R nor S .*

► **Proposition 24** (Conservativity of $(_)\bullet$). *Given two distinct atoms R and S not occurring in A , B and Γ , if $A^\bullet, \neg_R \neg_S \Gamma^\bullet \vdash B^\bullet$ is provable in $\mathcal{L}$ then $A \triangleleft \Gamma \vdash B$ is provable in $\cap\mathcal{IS}_1$.*

Proof. By induction on the proof of $A^\bullet, \neg_R \neg_S \Gamma^\bullet \vdash B^\bullet$. We look at the last rule. In all cases, we rely on the induction hypothesis in a direct manner, except for the $(\rightarrow L)$ rule. In this case, the main formula can be $A^\bullet$ or in $\neg_R \neg_S \Gamma^\bullet$:

- in the first case, we get $A = C \rightarrow D$ and $\Gamma = \Delta, \Sigma$ with $\neg_R \neg_S \Delta^\bullet \vdash \neg_R \neg_S C^\bullet$ (then $\neg_R \neg_S \Delta^\bullet, \neg_S C^\bullet \vdash R$ by Lemma 2) and $D^\bullet, \neg_R \neg_S \Sigma^\bullet \vdash B^\bullet$, so that, by Lemma 22, we have $\Delta = E$ and $E^\bullet \vdash C^\bullet$ and using the induction hypothesis twice, we can build:

$$\frac{\frac{E \triangleleft \vdash C \quad D \triangleleft \Sigma \vdash B}{C \rightarrow D \triangleleft E, \Sigma \vdash B} \rightarrow S}{\quad} \rightarrow S$$

- in the second case, we get $\Gamma = \Delta, C, \Sigma, \Xi$ and $\neg_R \neg_S \Sigma^\bullet \vdash \neg_S C^\bullet$ which is not possible by Lemmas 2 and 23. ◀

From these results we can give an alternative proof of $(Scut)$ admissibility for $\cap\mathcal{IS}_1$ (we already knew it as a consequence of Theorem 5).

Proof. Assume we have a proof of $\Gamma \vdash A$ and a proof of $\Delta, A, \Sigma \vdash C$ in $\cap\mathcal{IS}_1$, with $\Delta \neq \varepsilon \Rightarrow |\Gamma| = 1$. We know Γ is not empty (as a property of $\cap\mathcal{IS}_1$) and thus can be written $\Gamma = D, \Gamma'$ with $\Delta \neq \varepsilon \Rightarrow \Gamma' = \varepsilon$. By Proposition 21, we have proofs of $D^\bullet, \neg_R \neg_S \Gamma'^\bullet \vdash A^\bullet$ and either $A^\bullet, \neg_R \neg_S \Sigma^\bullet \vdash C^\bullet$ if $\Delta = \varepsilon$, or $B^\bullet, \neg_R \neg_S \Delta'^\bullet, \neg_R \neg_S A^\bullet, \neg_R \neg_S \Sigma^\bullet \vdash C^\bullet$ if $\Delta = B, \Delta'$.

In the first case, we can build:

$$\frac{\frac{D^\bullet, \neg_R \neg_S \Gamma'^\bullet \vdash A^\bullet}{D^\bullet, \neg_R \neg_S \Gamma'^\bullet, \neg_R \neg_S \Sigma^\bullet \vdash C^\bullet} \quad \frac{A^\bullet, \neg_R \neg_S \Sigma^\bullet \vdash C^\bullet}{\quad} cut}{\quad}$$

which leads to a proof of $D, \Gamma', \Sigma \vdash C = \Delta, \Gamma, \Sigma \vdash C$ in $\cap\mathcal{IS}_1$ by Proposition 24.

In the second case, $\Gamma' = \varepsilon$ and we can build:

$$\frac{\frac{\frac{D^\bullet \vdash A^\bullet}{\neg_R \neg_S D^\bullet \vdash \neg_R \neg_S A^\bullet} \square \quad \frac{B^\bullet, \neg_R \neg_S \Delta'^\bullet, \neg_R \neg_S A^\bullet, \neg_R \neg_S \Sigma^\bullet \vdash C^\bullet}{B^\bullet, \neg_R \neg_S \Delta'^\bullet, \neg_R \neg_S D^\bullet, \neg_R \neg_S \Sigma^\bullet \vdash C^\bullet} cut}{\quad}$$

which leads to a proof of $B, \Delta', D, \Sigma \vdash C = \Delta, \Gamma, \Sigma \vdash C$ in $\cap\mathcal{IS}_1$ by Proposition 24. ◀

Relying on the fact that we do not use the unit 0 in $\cap\mathcal{IS}_1$, we can go one step further and relate $\cap\mathcal{IS}_1$ with **cyMALL** (the cyclic variant of classical multiplicative additive linear logic [36]), where the intuitionistic constraint has been released.

► **Proposition 25.** *Given two distinct atoms R and S not occurring in A and B , $A \triangleleft \vdash B$ is provable in $\cap\mathcal{IS}_1$ if and only if $\vdash A^{\bullet\perp}, B^\bullet$ is provable in **cyMALL**.*

■ **Table 8** Minimal Relevant Logic $(\vee, t)$ -free $B+$ with Ω .

Axioms:				
$A \rightarrow A$	$A \cap B \rightarrow A$	$(A \rightarrow B) \cap (A \rightarrow C) \rightarrow A \rightarrow B \cap C$		
	$A \cap B \rightarrow B$	$\Omega \rightarrow A \rightarrow \Omega$		
Rules:				
$\frac{A}{B}$	$\frac{A \rightarrow B}{(C \rightarrow A) \rightarrow (C \rightarrow B)}$	$\frac{A \rightarrow B}{(B \rightarrow C) \rightarrow (A \rightarrow C)}$	$\frac{A \quad B}{A \cap B}$	$\overline{\Omega}$

Proof. By Propositions 21 and 24, we have an equivalence between $A \triangleleft \vdash B$ in $\cap IS_1$ and $A^\bullet \vdash B^\bullet$ in L . Now, if Γ and C do not contain the unit 0, then $\Gamma \vdash C$ in L iff $\vdash \bar{\Gamma}^\perp, C$ in $cyMALL$. This is a non-commutative variant of [29, Proposition 3.8], see for example [14, 21]. ◀

To sum up, given two distinct atoms R and S (used in $(_)^\bullet$), and two formulas A and B of $\cap IS$ not containing R and S , the following statements are all equivalent:
 $A \leq B$ in BCD , $A \vdash B$ in IS , $A \vdash B$ in $\cap IS$, $A \triangleleft \vdash B$ in $\cap IS_1$, $A \triangleleft \vdash B$ in $\cap IS_1^{rev}$,
 $A^\bullet \vdash B^\bullet$ in L , $\vdash A^{\bullet\perp}, B^\bullet$ in $cyMALL$.

We do not know if, when extending the translation with $(\forall X A)^\bullet = \forall X A^\bullet$, conservativity (Proposition 24) from $\cap IS_1$ into L holds or not.

4.7 Link with Relevant Logic

The $(\vee, t)$ -free variant of the minimal relevant logic $B+$ [27] is the logic given by Ω -free axioms and rules of Table 8 (to avoid yet another change of notations for the connectives, we use $\cap$ and Ω where $\wedge$ and $\top$ would be more common for $B+$). R. Meyer and B. Venneri [34] have identified a strong relation between Ω -free BCD and this $(\vee, t)$ -free $B+$. Their key result is the following theorem.

► **Theorem 26** (Coincidence between BCD and $B+$ [34, Theorem 3.4]). *If $A \leq B$ (with A and B not containing Ω) in BCD then $A \rightarrow B$ is provable in $B+$.*

Conversely, if C (without Ω) is provable in $B+$, then it is of the shape $C = \bigcap_{i \in I} A_i \rightarrow B_i$, and for each $i \in I$, we have $A_i \leq B_i$ in BCD .

By Theorem 19, we then also get an equivalence between $\cap IS_1$ and $B+$ in this Ω -free case. We are going to slightly refine this correspondence in two different directions:

- first we take Ω into consideration by introducing it in $B+$ (see red part in Table 8) through one new axiom and one new rule which are the natural nullary versions of the corresponding binary $\cap$ cases;
- second, we move from $\cap IS_1$ to $\cap IS_h$, which makes the correspondence even more natural (remember for example that $\vdash A \rightarrow A$ is provable in $\cap IS_h$, but not in $\cap IS_1$).

► **Theorem 27** (Coincidence between $\cap IS_h$ and $B+$ with Ω). *$\vdash A$ is provable in $\cap IS_h$ if and only if A is provable in $B+$ with Ω .*

Proof. In both directions, we go by induction on the hypothesis derivation. From $\cap IS_h$ to $B+$, we prove the generalised statement:

$$A_1, \dots, A_n \vdash A \text{ provable in } \cap IS_h \implies A_1 \rightarrow \dots \rightarrow A_n \rightarrow A \text{ provable in } B+ \text{ with } \Omega. \quad \blacktriangleleft$$

■ **Table 9** $\cap\text{IS}$ Scott Deduction System.

$\frac{\triangleleft \vdash A \quad B \triangleleft \vdash X}{A \rightarrow B \triangleleft \Gamma \vdash X} \mathcal{AR}$	$\frac{}{X \triangleleft \Gamma \vdash X} \mathcal{AL}$
$\frac{\Pi \triangleleft \Gamma, A \vdash B}{\Pi \triangleleft \Gamma \vdash A \rightarrow B} \rightarrow R$	$\frac{C \triangleleft \vdash A \quad B \triangleleft \Gamma \vdash D}{A \rightarrow B \triangleleft C, \Gamma \vdash D} \rightarrow S$
$\frac{\Pi \triangleleft \Gamma \vdash A \quad \Pi \triangleleft \Gamma \vdash B}{\Pi \triangleleft \Gamma \vdash A \cap B} \cap R$	$\frac{A_i \triangleleft \Gamma \vdash C}{A_1 \cap A_2 \triangleleft \Gamma \vdash C} \cap L_i$ $\frac{}{\Pi \triangleleft \Gamma \vdash \Omega} \Omega R$

In this way, we obtain a presentation of the $B+$ system which satisfies the sub-formula property (we have found no such presentation in the literature, but we might have missed it). As an additional remark, remember that provability in $\cap\text{IS}_h$ and $\cap\text{IS}$ are the same (Proposition 7). Then, for A a formula of $\cap\text{IS}$:

$$A \text{ in } B+ \iff \vdash A \text{ in } \cap\text{IS}_h \iff \vdash A \text{ in } \cap\text{IS} \iff \vdash A \text{ in } \text{IS}$$

4.8 Atoms and η -Invariance

As already mentioned, the η -condition is not satisfied by the BCD subtyping relation (Lemma 18). However this is a necessary and sufficient condition for an extension of BCD to induce a model of the λ -calculus which is invariant under η -expansion (Proposition 15). Various such extensions have been considered in the literature. We focus on two key models: the Scott model (corresponding to $X \sim \Omega \rightarrow X$) and the Park model (corresponding to $X \sim X \rightarrow X$).

We have to adapt the $\cap\text{IS}$ system to cover these new equations. However the proof techniques then remain the same by relying on proof-theoretical properties, such as cut admissibility.

4.8.1 The Scott System

The Scott model [31] (see also [4, Chapter 13]) can be presented as a BCD-style intersection type system by considering a single atom o satisfying $o \sim \Omega \rightarrow o$. As it makes no technical difference, we directly consider here the generalisation to an arbitrary set $\mathcal{A}$ of atoms all satisfying: $X \sim \Omega \rightarrow X$ (*Scott*). Such an equation immediately guarantees the η -condition.

We now present a variant of the $\cap\text{IS}_1$ system corresponding to this new theory. In order to preserve the sub-formula property, we slightly extend the notion of sequent: $A \triangleleft \Gamma \vdash C$ is now more than just a notation for $A, \Gamma \vdash C$. We consider sequents of the shape $\Pi \triangleleft \Gamma \vdash C$ where we have three separated areas: Π is either empty or a singleton formula, Γ is a list of formulas and C is a formula. In particular, $\triangleleft A, \Gamma \vdash C$ and $A \triangleleft \Gamma \vdash C$ are two *distinct* valid sequents. The rules of the **system** $\cap\text{IS Scott}$ are presented on Table 9.

In sequents, the use of an empty Π is internalising Ω in a way which is compatible with the sub-formula property (otherwise the $(\mathcal{AR})$ rule would need to use $\Omega \triangleleft \vdash A$ as a premise and would violate the sub-formula property).

► **Lemma 28.** *In $\cap\text{IS Scott}$, $\triangleleft \Gamma \vdash C$ if and only if $\Omega \triangleleft \Gamma \vdash C$.*

The new system allows us to derive the (*Scott*) equation:

$$\frac{\overline{X \triangleleft \Omega \vdash X} \mathcal{AL}}{X \triangleleft \vdash \Omega \rightarrow X} \rightarrow R \quad \frac{\overline{\triangleleft \vdash \Omega} \Omega R \quad \overline{X \triangleleft \vdash X} \mathcal{AL}}{\Omega \rightarrow X \triangleleft \vdash X} \mathcal{AR}$$

Before stating the link with the more traditional BCD-style approach, we give some admissible rules.

► **Lemma 29** (Scott Weakenings). *The following two weakening rules are admissible in $\cap$ IS Scott:*

$$\frac{X \triangleleft \Gamma \vdash C}{X \triangleleft A, \Gamma \vdash C} \quad \frac{\Pi \triangleleft \Gamma \vdash X}{\Pi \triangleleft \Gamma, A \vdash X}$$

► **Theorem 30** (Scott Cuts Admissibility). *The following three cut rules are admissible in the $\cap$ IS Scott system:*

$$\frac{\Pi \triangleleft \Gamma \vdash B \quad B \triangleleft \Delta \vdash C}{\Pi \triangleleft \Gamma, \Delta \vdash C} \quad \frac{A \triangleleft \vdash B \quad \Pi \triangleleft \Gamma, B, \Delta \vdash D}{\Pi \triangleleft \Gamma, A, \Delta \vdash D} \quad \frac{\triangleleft \vdash B \quad \Pi \triangleleft \Gamma, B \vdash X}{\Pi \triangleleft \Gamma \vdash X}$$

The equivalence with the BCD-style presentation then follows.

► **Proposition 31.** *$A \leq B$ in BCD extended with (*Scott*) if and only if $A \triangleleft \vdash B$ in $\cap$ IS Scott.*

Finally, the sub-formula property of $\cap$ IS Scott comes with a simple direct proof of the validity of the β -condition.

► **Lemma 32** (β -Condition for $\cap$ IS Scott). *The following property holds in $\cap$ IS Scott:*

$$\bigcap_{i \in I} A_i \rightarrow B_i \triangleleft A, \Gamma \vdash B \implies \exists J \subseteq I, A \triangleleft \vdash \bigcap_{j \in J} A_j \wedge \bigcap_{j \in J} B_j \triangleleft \Gamma \vdash B$$

Proof. Very similar to the proof of Lemma 20, with only one case to add when the last rule is ($\mathcal{AR}$). I is a singleton $\{i\}$ with $\triangleleft \vdash A_i$ and $B_i \triangleleft \vdash X$. We choose $J = \{i\}$. By Lemma 28, we have $\Omega \triangleleft \vdash A_i$ thus $A \triangleleft \vdash A_i$. By Lemma 29, we have $B_i \triangleleft \Gamma \vdash X$. ◀

4.8.2 The Park System

Another well-known model is the Park model [25] (see also [4, Chapter 13]) given as an extension of BCD with only one atom o satisfying $o \sim o \rightarrow o$. As it makes no technical difference, we directly consider here the generalisation to an arbitrary set $\mathcal{A}$ of atoms all satisfying: $X \sim X \rightarrow X$ (*Park*). Such an equation immediately guarantees the η -condition.

For the associated variant of the $\cap$ IS system, we go back to the same sequents as in $\cap$ IS₁: $A \triangleleft \Gamma \vdash B$. The rules of the **system $\cap$ IS Park** are presented on Table 10.

The new system allows us to derive the (*Park*) equation:

$$\frac{\overline{X \triangleleft \vdash X} \mathcal{AL}}{\overline{X \triangleleft X \vdash X} \mathcal{AL}} \rightarrow R \quad \frac{\overline{X \triangleleft \vdash X} \mathcal{AL} \quad \overline{X \triangleleft \vdash X} \mathcal{AL}}{X \rightarrow X \triangleleft \vdash X} \mathcal{AR}$$

Here are some admissible rules.

► **Lemma 33** (Park Weakenings). *The following two weakening rules are admissible in $\cap$ IS Park:*

$$\frac{X \triangleleft \Gamma \vdash C \quad A \triangleleft \vdash X}{X \triangleleft A, \Gamma \vdash C} \quad \frac{C \triangleleft \Gamma \vdash X \quad A \triangleleft \vdash X}{C \triangleleft \Gamma, A \vdash X}$$

■ **Table 10** $\cap$ IS Park Deduction System.

$\frac{X \triangleleft \vdash A \quad B \triangleleft \vdash X \quad \dots C_i \triangleleft \vdash X \quad \dots}{A \rightarrow B \triangleleft C_1, \dots, C_k \vdash X} AR$		$\frac{\dots C_i \triangleleft \vdash X \quad \dots}{X \triangleleft C_1, \dots, C_k \vdash X} AL$
$\frac{C \triangleleft \Gamma, A \vdash B}{C \triangleleft \Gamma \vdash A \rightarrow B} \rightarrow R$	$\frac{C \triangleleft \vdash A \quad B \triangleleft \Gamma \vdash D}{A \rightarrow B \triangleleft C, \Gamma \vdash D} \rightarrow S$	
$\frac{C \triangleleft \Gamma \vdash A \quad C \triangleleft \Gamma \vdash B}{C \triangleleft \Gamma \vdash A \cap B} \cap R$	$\frac{A_i \triangleleft \Gamma \vdash C}{A_1 \cap A_2 \triangleleft \Gamma \vdash C} \cap L_i$	$\frac{}{C \triangleleft \Gamma \vdash \Omega} \Omega R$

► **Theorem 34** (Park Cuts Admissibility). *The following three cut rules are admissible in the $\cap$ IS Park system:*

$$\frac{A \triangleleft \Gamma \vdash B \quad B \triangleleft \Delta \vdash C}{A \triangleleft \Gamma, \Delta \vdash C} \quad \frac{A \triangleleft \vdash B \quad C \triangleleft \Gamma, B, \Delta \vdash D}{C \triangleleft \Gamma, A, \Delta \vdash D} \quad \frac{X \triangleleft \vdash B \quad C \triangleleft \Gamma, B \vdash X}{C \triangleleft \Gamma \vdash X}$$

The equivalence with the BCD-style presentation then follows.

► **Proposition 35.** *$A \leq B$ in BCD extended with (Park) if and only if $A \triangleleft \vdash B$ in $\cap$ IS Park.*

Finally, the sub-formula property of $\cap$ IS Park comes with a simple direct proof of the validity of the β -condition.

► **Lemma 36** (β -Condition for $\cap$ IS Park). *The following property holds in $\cap$ IS Park:*

$$\bigcap_{i \in I} A_i \rightarrow B_i \triangleleft A, \Gamma \vdash B \implies \exists J \subseteq I, A \triangleleft \vdash \bigcap_{j \in J} A_j \wedge \bigcap_{j \in J} B_j \triangleleft \Gamma \vdash B$$

Proof. Very similar to the proof of Lemma 20, with only one case to add when the last rule is (AR). I is a singleton $\{i\}$ with $X \triangleleft \vdash A_i$ and for each $C \in B_i, A, \Gamma$, we have $C \triangleleft \vdash X$. We choose $J = \{i\}$. We have $A \triangleleft \vdash X$ and $X \triangleleft \vdash A_i$ thus $A \triangleleft \vdash A_i$. By Lemma 33, we have:

$$\frac{B_i \triangleleft \vdash X \quad \dots C \triangleleft \vdash X \quad \dots (C \in \Gamma)}{B_i \triangleleft \Gamma \vdash X} \quad \blacktriangleleft$$

5 Polymorphic Subtyping

Polymorphic subtyping has been the subject of many works in the last forty years. We are interested in systems validating the following distributivity property:

$$\forall X (A \rightarrow B) \leq (\forall X A) \rightarrow (\forall X B)$$

We focus here on a line of work analysing such systems through logical tools [24, 23, 32]. It was initiated with an axiomatized, Hilbert style, presentation [24] (see Table 11), satisfying soundness and completeness w.r.t. the semantics of “simple inference models”.

In a second step, the Hilbert-style presentation was refined into an “half-way sequent calculus” [23] (see Table 12): it manipulates judgements/sequents of the shape $A \leq B$, with a “deep” rule for $\forall$ introduction on the right. A key contribution is that transitivity is not part of the presentation anymore and becomes an admissible rule.

■ **Table 11** Mitchell’s System [24].

$\frac{}{A \leq A}$	$\frac{A \leq B \quad B \leq C}{A \leq C}$	$\frac{C \leq A \quad B \leq D}{A \rightarrow B \leq C \rightarrow D}$	$\frac{A \leq B}{\forall X A \leq \forall X B}$
$\frac{}{\forall X (A \rightarrow B) \leq (\forall X A) \rightarrow (\forall X B)}$	$\frac{}{\forall X A \leq A[C/X]}$	$\frac{}{A \leq \forall X A} \quad X \notin A$	

■ **Table 12** Longo–Milsted–Soloviev’s System [23].

$\frac{}{A \leq A}$	$\frac{C \leq A \quad B \leq D}{A \rightarrow B \leq C \rightarrow D}$	$\frac{A[C/X] \leq B}{\forall X A \leq B}$	$\frac{C \leq A_1 \rightarrow \dots \rightarrow A_n \rightarrow B}{C \leq A_1 \rightarrow \dots \rightarrow A_n \rightarrow \forall X B}$ $X \text{ not free in } C, A_1, \dots, A_n$
---------------------	--	--	--

The third step [32] was to remove this “deep” rule and to use (standard/shallow) natural deduction and sequent calculus. This was made possible thanks to the use of intuitionistic sequents of the shape $A_1, \dots, A_n \leq B$ ($n > 0$). Cut admissibility is first proved in the natural deduction system (using substitutions) and then extended to the sequent calculus by means of an embedding into natural deduction and the use of some additional cut elimination steps.

It happens the proposed sequent calculus is exactly $\forall\text{IS}_1$ (Table 2 without the rule for $\top/\Omega$ and $\&/\cap$). We are not going to provide much details here since we can simply reap the fruits of Section 3. If the link with the Lambek calculus was not explicitly developed in [32], the main characteristics were already stressed in its last sentence: “Thus our logic can be qualified as noncommutative linear intuitionistic logic.” Regarding cut admissibility, our approach from Section 3 works directly on the sequent calculus in a more standard manner. As another input of our approach, we can constrain the structure of derivations without loosing provability thanks to $\forall\text{IS}_1^{\text{rev}}$. The goal is not really to deal with proof search and complexity here, since we know that anyway we have undecidability [33].

6 Conclusions and Future Works

By introducing the S-constraint in the Lambek calculus, we have defined the unifying *intersection subtyping logic* IS. We have found back theories from intersection types and polymorphic subtyping. The contribution is mostly theoretical by establishing strong bridges between three lines of work initiated in the last century: Lambek calculus (from 1958 [15, 11]), BCD-style intersection subtyping (from 1983 [3, 26, 7]) and polymorphic subtyping (from 1988 [24, 23, 32]).

We have highlighted immediate impacts on the existing theory: simplified relation with the logic B+, focused system for polymorphic subtyping, etc. Another consequence is the possibility to consider interactions between $\&/\cap$ and $\forall$, such as $\forall X (A \cap B) \sim (\forall X A) \cap (\forall X B)$. Beyond that, we have tried to state and prove results about IS and its sub-systems in their most general form so that future possible extensions of subtyping could appear in this unified framework by taking advantage of the other available connectives. Regarding expressiveness, one should also consider now integrating additional type constructions (base types, products, sums, records, variants, unions, etc.) into IS (see for example [5, 19] or [26, Section 5]). Providing transitivity-free presentations will basically help proving any property about subtyping thanks to simpler inductive approaches.

A third stage will then be to consider applications to programming languages through inference, algorithmic aspects, complexity questions, etc. Remember the exact complexity of BCD subtyping is still unknown. The link we established with the Lambek calculus draws yet another link with a PTIME problem [28].

Regarding intersection types, we have introduced variants of IS for the Scott and Park models, which allow us to deal with the appropriate recursive equations on atoms. Many denotational models from the literature (see for example [4, Figure 13.1]) require to mix equations on atoms with primitive subtyping relations between those atoms. We plan to extend our systems to incorporate uniformly these two ingredients so that we could be able to cover a much larger set of models (see [22] for a solution based on non-wellfounded proofs).

References

- 1 Fabio Alessi, Franco Barbanera, and Mariangiola Dezani-Ciancaglini. Intersection types and lambda models. *Theoretical Computer Science*, 355(2):108–126, April 2006. doi:10.1016/j.tcs.2006.01.004.
- 2 Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation*, 2(3):297–347, 1992. doi:10.1093/logcom/2.3.297.
- 3 Henk Barendregt, Mario Coppo, and Mariangiola Dezani-Ciancaglini. A filter lambda model and the completeness of type assignment. *Journal of Symbolic Logic*, 48:931–940, 1983. doi:10.2307/2273659.
- 4 Henk Barendregt, Wil Dekkers, and Richard Statman. *Lambda Calculus with Types*. Perspectives in Logic. Cambridge University Press, 2013. doi:10.1017/CB09781139032636.
- 5 Jan Bessai, Tzu-Chun Chen, Andrej Dudenhefner, Boris Döder, Ugo de Liguoro, and Jakob Rehof. Mixin composition synthesis based on intersection types. *Logical Methods in Computer Science*, 14:37, February 2018. doi:10.23638/LMCS-14(1:18)2018.
- 6 Jan Bessai, Jakob Rehof, and Boris Döder. Fast verified BCD subtyping. In Tiziana Margaria, Susanne Graf, and Kim G. Larsen, editors, *Models, Mindsets, Meta: The What, the How, and the Why Not? – Essays Dedicated to Bernhard Steffen on the Occasion of His 60th Birthday*, volume 11200 of *Lecture Notes in Computer Science*, pages 356–371. Springer, June 2019. doi:10.1007/978-3-030-22348-9_21.
- 7 Xuan Bi, Bruno C. d. S. Oliveira, and Tom Schrijvers. The essence of nested composition. In Todd Millstein, editor, *32nd European Conference on Object-Oriented Programming (ECOOP)*, volume 109 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:33. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ECOOP.2018.22.
- 8 Viviana Bono and Jerzy Tiuryn. Products and polymorphic subtypes. *Fundamenta Informaticae*, 51(1–2):13–41, 2002. doi:10.3233/FUN-2002-511-203.
- 9 Vincent Danos, Jean-Baptiste Joinet, and Harold Schellinx. A new deconstructive logic: linear logic. *Journal of Symbolic Logic*, 62(3):755–807, September 1997. doi:10.2307/2275572.
- 10 Boris Döder, Moritz Martens, Jakob Rehof, and Pawel Urzyczyn. Bounded combinatory logic. In Patrick Cégielski and Arnaud Durand, editors, *Computer Science Logic*, volume 16 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 243–258. European Association for Computer Science Logic, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012. doi:10.4230/LIPIcs.CSL.2012.243.
- 11 Nikolaos Galatos, Peter Jipsen, Tomasz Kowalski, and Hiroakira Ono. *Residuated Lattices: An Algebraic Glimpse at Substructural Logics*. Number 151 in Studies in Logic and the Foundations of Mathematics. Elsevier, April 2007. doi:10.1016/S0049-237X(13)70001-6.
- 12 Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987. doi:10.1016/0304-3975(87)90045-4.
- 13 Jean-Yves Girard. Light linear logic. *Information and Computation*, 143(2):175–204, June 1998. doi:10.1006/inco.1998.2700.

- 14 François Lamarche and Christian Retoré. Proof nets for the Lambek calculus - an overview. In Michele Abrusci and Claudia Casadio, editors, *Roma workshop: Proofs and Linguistic Categories*, pages 241–262, Bologna, April 1996. CLUEB. URL: <https://www.lirmm.fr/~retore/ARTICLES/LaReRoma96.pdf>.
- 15 Joachim Lambek. The mathematics of sentence structure. *The American Mathematical Monthly*, 65(3):154–170, March 1958. doi:10.1080/00029890.1958.11989160.
- 16 Olivier Laurent. IS Logic. Software, swbId: swb:1:dir:682600883bd8fd60057d332c6a87a39c047f22e7 (visited on 2026-07-01). URL: <https://github.com/olaure01/islogic/tree/LICS2026>, doi:10.4230/artifacts.26957.
- 17 Olivier Laurent. A proof of the focusing property of linear logic. Unpublished note, 2004, revised 2017. URL: <https://perso.ens-lyon.fr/olivier.laurent/11foc2.pdf>.
- 18 Olivier Laurent. A syntactic introduction to intersection types. Unpublished note, April 2012. URL: <http://perso.ens-lyon.fr/olivier.laurent/tutinter.pdf>.
- 19 Olivier Laurent. Intersection subtyping with constructors. In Michele Pagani and Sandra Alves, editors, *Proceedings Twelfth Workshop on Developments in Computational Models and Ninth Workshop on Intersection Types and Related Systems (DCM 2018 and ITRS 2018)*, volume 293 of *Electronic Proceedings in Theoretical Computer Science*, pages 73–84. Open Publishing Association, April 2019. doi:10.4204/EPTCS.293.6.
- 20 Olivier Laurent. An anti-locally-nameless approach to formalizing quantifiers. In Catalin Hritcu and Andrei Popescu, editors, *CPP 2021: Proceedings of the 10th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pages 300–312. ACM, January 2021. doi:10.1145/3437992.3439926.
- 21 Olivier Laurent. Yalla: Yet another deep embedding of linear logic in Rocq. *Journal of Automated Reasoning*, 70(9), June 2026. doi:10.1007/s10817-026-09755-y.
- 22 Olivier Laurent and Jui-Hsuan Wu. Non-wellfounded derivations for intersection subtyping with fixpoints. In Frank Pfenning, editor, *11th International Conference on Formal Structures for Computation and Deduction*, volume 378 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2026. doi:10.4230/LIPIcs.FSCD.2026.22.
- 23 Giuseppe Longo, Kathleen Milsted, and Sergei Soloviev. Coherence and transitivity of subtyping as entailment. *Journal of Logic and Computation*, 10(4):493–526, 2000. doi:10.1093/LOGCOM/10.4.493.
- 24 John C. Mitchell. Polymorphic type inference and containment. *Information and Computation*, 76(2–3):211–249, 1988. doi:10.1016/0890-5401(88)90009-0.
- 25 David Park. The Y-combinator in Scott’s lambda-calculus models. Technical Report 13, University of Warwick, Department of Computer Science, June 1976. URL: <https://wrap.warwick.ac.uk/46310/>.
- 26 Benjamin C. Pierce. A decision procedure for the subtype relation on intersection types with bounded variables. Technical Report CMU-CS-89-169, School of Computer Science, Carnegie Mellon University, August 1989. doi:10.1184/R1/6587339.v1.
- 27 Richard Routley and Robert K. Meyer. The semantics of entailment – III. *Journal of Philosophical Logic*, 1:192–208, May 1972. doi:10.1007/BF00650498.
- 28 Yury Savateev. Lambek grammars with one division are decidable in polynomial time. In Edward A. Hirsch, Alexander A. Razborov, Alexei Semenov, and Anatol Slissenko, editors, *Computer Science – Theory and Applications, Third International Computer Science Symposium (CSR)*, pages 273–282. Springer, 2008. doi:10.1007/978-3-540-79709-8_28.
- 29 Harold Schellinx. Some syntactical observations on linear logic. *Journal of Logic and Computation*, 1(4):537–559, 1991. doi:10.1093/logcom/1.4.537.
- 30 Jürgen Schulte Mönting. Cut elimination and word problems for varieties of lattices. *Algebra Universalis*, 12:290–321, December 1981. doi:10.1007/BF02483891.

- 31 Dana Scott. Continuous lattices. In Lawvere, editor, *Toposes, Algebraic Geometry and Logic*, volume 274 of *Lecture Notes in Mathematics*, pages 97–136. Springer, 1972. doi:10.1007/BFb0073967.
- 32 Jerzy Tiuryn. A sequent calculus for subtyping polymorphic types. *Information and Computation*, 164(2):345–369, 2001. doi:10.1006/inco.2000.2941.
- 33 Jerzy Tiuryn and Pawel Urzyczyn. The subtyping problem for second-order types in undecidable. *Information and Computation*, 179(1):1–18, November 2002. doi:10.1006/inco.2001.2950.
- 34 Betti Venneri. Intersection types as logical formulae. *Journal of Logic and Computation*, 4(2):109–124, April 1994. doi:10.1093/logcom/4.2.109.
- 35 Philip Whitman. Free lattices. *Annals of Mathematics*, 42(1):325–330, January 1941. doi:10.2307/1969001.
- 36 David N. Yetter. Quantales and (noncommutative) linear logic. *Journal of Symbolic Logic*, 55(1):41–64, 1990. doi:10.2307/2274953.

A Rocq Formalization

Most of the results of the present paper are formalized in Rocq. Before addressing our own formalization, we should mention that [7] also comes with a Coq formalization: <http://dx.doi.org/10.4230/DARTS.4.3.5>. In particular, the correctness and completeness of Pierce’s algorithm [26] are proved, and the authors mention that they had to fill some gaps in Pierce’s proofs. This advocates for the use of proof assistants. Additional lemmas were added with respect to the proof of transitivity of the relation induced by the algorithm given in [26]. This property is formalized as `nested-composition/coq/Subtype_Property.v:ASub_trans`. As already mentioned, the proof is based on the introduction of “reflexive supertypes” (same as in [26]), while we go directly by a cut-elimination procedure.

Our Rocq development is available at <https://github.com/olaure01/islogic/tree/LICS2026>. There is nothing particularly original in this formalization. Let us mention that, regarding the representation of quantifiers, we rely on the anti-locally-nameless approach [20]. This is made possible by the fact that we only need second-order quantifiers: no first-order ones here. The anti-locally-nameless approach does not allow to have full interactions between first-order and second-order quantifiers.

Here is a correspondence table for the definitions of the main systems in the Rocq formalization (references have the shape `file_name.v:inductive_definition`):

L	(Table 1, all colors)	<code>lambek_is.v:lprove</code>	(with <code>bp = false</code>)	($\otimes, \multimap$)-free
IS	(Table 1, without cyan)	<code>lambek_is.v:lprove</code>	(with <code>bp = true</code>)	
IS_h	(Table 1, black only)	<code>ish.v:lprove</code>		
nIS_1	(Table 2)	<code>nis.v:lprove</code>		
nL^{rev}	(Table 3)	<code>nlambek.v:rlprove</code>		
nIS_1^{rev}	(Table 4)	<code>nis.v:rlprove</code>		
BCD	(Tables 5 and 7)	<code>bcd.v:bcd_sub</code>	(with <code>lax = nil</code>)	
B+	(Table 8)	<code>relevant.v:bp</code>		
$\cap IS$ Scott	(Table 9)	<code>scott.v:sc_sub</code>		
$\cap IS$ Park	(Table 10)	<code>park.v:pk_sub</code>		
Mitchell	(Table 11)	<code>fis.v:m_sub</code>		
LMS	(Table 12)	<code>fis.v:lms_sub</code>		

Regarding L, we have formalized the ($\otimes, \multimap$)-free case only (thus these two connectives are missing in the cut-elimination proof for L, but they would not introduce anything specific). As indicated above, L and IS are defined (in `lambek_is.v`) as a single inductive type `lprove` with

a Boolean parameter `bp` describing if the S-constraint is activated or not (thus `bp = false` for `L` and `bp = true` for `IS`). In the same spirit, Theorems 1 and 4 are proved simultaneously: now the Boolean parameter `bp` is used to decide if the cut rule under consideration is (*cut*) or (*Scut*).

Regarding the main results of the paper, a correspondence table with `Rocq` statements is given in `references.txt`. Equivalences between systems of Section 5 are formalized in `fis.v`.

B Proof Search in $\cap IS_1^{\text{rev}}$

There is an injection from the set of all sequents which may appear during executions of the backward proof-search algorithm in $\cap IS_1^{\text{rev}}$ on an input $A \triangleleft \vdash B$ into pairs of occurrences of a sub-formula of A and a sub-formula of B . Let C be a sub-formula of A , we call l_C^A the number of times we go to the left of a $\rightarrow$ -connective in the syntax tree of A to reach C from the root, and r_C^A the number of times we go to the right of a $\rightarrow$ -connective *after* the last time we went to the left of a $\rightarrow$. We call Σ_C^A the list of the sub-formulas of A which were at the left of the last r_C^A $\rightarrow$ -connectives we have crossed (by definition of r_C^A , we had to go to the right of these $\rightarrow$ to reach C). We can check that, if $C \triangleleft \Delta \vdash D$ appears in an execution of the algorithm on input $A \triangleleft \vdash B$ then:

- one of C and D is a sub-formula of A and the other is a sub-formula of B ;
- if C is a sub-formula of A (and thus D is a sub-formula of B), $l_C^A = l_D^B$ (and they are even) and $\Sigma_D^B = \Xi, \Delta$ for some Ξ of length r_C^A (in particular, for each D' in Δ , D' is a sub-formula of B and $l_{D'}^B = l_D^B + 1$ and $r_{D'}^B$ is the position of D' in Δ starting from 0 for its head);
- if C is a sub-formula of B (and thus D is a sub-formula of A), $l_C^B = l_D^A$ (and they are odd) and $\Sigma_D^A = \Xi, \Delta$ for some Ξ of length r_C^B (in particular, for each D' in Δ , D' is a sub-formula of A and $l_{D'}^A = l_D^A + 1$ and $r_{D'}^A$ is the position of D' in Δ starting from 0 for its head).

Proof. By induction on the length of a run from $A \triangleleft \vdash B$ to $C \triangleleft \Delta \vdash D$. Let us focus on the case where we move from $E \rightarrow F \triangleleft \Delta \vdash X$ to $hd(\Delta) \triangleleft \vdash E$ and $F \triangleleft tl(\Delta) \vdash X$ (with $\Delta \neq \varepsilon$, thus $\Delta = hd(\Delta), tl(\Delta)$) as last step of the run.

Let us assume X is a sub-formula of B (the case of A is similar). By induction hypothesis, we have $l_{E \rightarrow F}^A = l_X^B$ (and they are even) and $\Sigma_X^B = \Xi, hd(\Delta), tl(\Delta)$ for some Ξ of length $r_{E \rightarrow F}^A$. Regarding $hd(\Delta) \triangleleft \vdash E$, we have $l_{hd(\Delta)}^B = l_X^B + 1 = l_{E \rightarrow F}^A + 1 = l_E^A$ (which are odd) and $\Sigma_E^A = \varepsilon$ with $r_{hd(\Delta)}^B = 0$. Regarding $F \triangleleft tl(\Delta) \vdash X$, we have $l_F^A = l_{E \rightarrow F}^A = l_X^B$ (which are even) and $\Sigma_X^B = \Xi, hd(\Delta), tl(\Delta)$ with $r_F^A = r_{E \rightarrow F}^A + 1$.

See `iforulas.v:sub_coh`, `iis1r.v:trel_subsequents` and `iis1r.v:trel_subsequents_inj` for more details. ◀

We get in particular that $C \triangleleft \Delta \vdash D$ is uniquely determined by knowing C , D and the fact that it appears in an execution of the proof-search algorithm on input $A \triangleleft \vdash B$.

The correctness and completeness of the proof-search algorithm are formalized in `Rocq` in `iis1r.v:is_b_correct`.