A Decidable Subtyping Logic for Intersection and Union Types (full version)

Luigi Liquori, Claude Stolze

A Decidable Subtyping Logic for Intersection and Union Types

Luigi Liquori[1] Claude Stolze[2]

Université Côte d'Azur, Inria, France

Abstract

Proof-functional logical connectives allow reasoning about the structure of logical proofs, in this way giving to the latter the status of *first-class* objects. This is in contrast to classical *truth-functional* connectives where the meaning of a compound formula is dependent only on the truth value of its subformulas.
We present a proof-functional logic and we give a semantics using *Mints' realizers* accounting for intersection types, union types, and subtyping. The semantics interprets the type ω as the set universe, the $\rightarrow$ type as a function space, the $\cap$ and $\cup$ types as set intersection and set union, respectively, and the subtype relation $\leqslant$ as a subset operator.
Using the proof-as-types and terms-as-propositions paradigms, we extend the typed calculus previously defined by the authors with a decidable subtyping relation and we show this calculus to be isomorphic to the Barbanera-Dezani-Ciancaglini-de'Liguoro type assignment system.
A subtyping algorithm is presented and proved to be sound. Hindley gave a subtyping algorithm for intersection types but, as far as we know, there is no system in the literature also including union types.

Keywords: Logics and Lambda calculus, Types and Subtypes.

1 Introduction

This paper is a contribution to the study of *subtyping* in presence of *intersection* and *union* types and the role of such type system in *logical* investigations; it is a natural follow up of the recent paper by the authors [DdLS16].

Intersection types were first introduced as a form of *ad hoc* polymorphism in (pure) lambda-calculi *à la* Curry. The paper by Barendregt, Coppo, and Dezani [BCDC83] is a classic reference, and Hindley [Hin84] gives a useful introduction and bibliography. Union types were later introduced as a *dual* of intersection by MacQueen, Plotkin, and Sethi [MPS86]: Barbanera, Dezani, and de' Liguoro [BDCd95] is a definitive reference. As intersection and union types had their classical development for (undecidable) type assignment systems, many papers moved from intersection and union type theories to (typed) lambda-calculi *à la* Church: the programming language Forsythe, by Reynolds [Rey88,Rey96], is probably the first

[1] Email: Luigi.Liquori@inria.fr
[2] Email: Claude.Stolze@inria.fr

reference for intersection types, while Pierce's dissertation combines also unions and intersections [Pie91b,Pie91a]. The logical relation between type assignment systems and typed systems featuring intersection and union types were studied in [LR07,DL10,DdLS16].

Proof-functional connectives represent evidence as a "polymorphic" construction, that is, the *same* evidence can be used as a proof for different sentences. Pottinger [Pot80] first introduced a conjunction, called *strong conjunction* $\cap$, requiring more than the existence of constructions proving the left and the right hand side of the conjuncts. According to Pottinger: *"The intuitive meaning of* $\cap$ *can be explained by saying that to assert* $A \cap B$ *is to assert that one has a reason for asserting* A *which is also a reason for asserting* B*"*. This interpretation makes inhabitants of $A \cap B$ as uniform evidence for both A and B. Later, Lopez-Escobar [LE85] presented the first proof-functional logic with strong conjunction as a special case of ordinary conjunction. Mints [Min89] presented a logical interpretation of strong conjunction (*a.k.a.* intersection types) using *realizers*: the logical predicate $r_{A\cap B}[M]$ is true if the pure lambda-term M is a realizer (also read as "M is a method to assess σ") for either the formula $r_A[M]$ *and* $r_B[M]$. Inspired by this, Barbanera and Martini tried to answer to the question of realizing other "proof-functional" connectives, like *strong implication*, or Lopez-Escobar's *strong equivalence* or *provable type isomorphism* of Bruce, Di Cosmo and Longo [BL85,BCL92]. Recently [DdLS16] extended the logical interpretation with union types as another proof-functional operator, the *strong union* $\cup$. Paraphrasing Pottinger's point of view, we could say that the intuitive meaning of $\cup$ is that if we have a reason to assert A (or B), then the same reason will also assert $A \cup B$. This interpretation makes inhabitants of $(A \cup B) \supset C$ be uniform evidence for both $A \supset C$ and $B \supset C$. Symmetrically to intersection, and extending the Mints' logical interpretation, the logical predicate $r_{A\cup B}[M]$ succeeds if the pure lambda-term M is a realizer for either the formula $r_A[M]$ *or* $r_B[M]$.

1.1 Contributions.

This paper focus on the logical and algorithmic aspects of subtyping in presence of union and intersection types: our interest is not only theoretical but also pragmatic since it open the door to using those type operators in proof-assistants and logical frameworks, like LF, or Coq, or Isabelle. We also inspect the relationship between pure (*à la* Curry) and typed (*à la* Church) lambda-calculi and their corresponding proof-functional logics as dictated by the well-known Curry-Howard [How80] correspondence. We'll present and explore the relationships between the following four formal systems:

- $\Lambda^{\cap\cup}_{\mathsf{u}\leqslant}$, the type assignment system with intersection and union types for pure lambda-calculus with subsumption rule and the type theory Ξ, as defined in [BDCd95];
- $\Lambda^{\cap\cup}_{\mathsf{t}\leqslant}$, an extension of the type system with intersection and union types for the typed lambda-calculus, as defined in [DL10], with subtyping as explicit coercions;
- $\mathcal{L}^{\cap\cup}_{\leqslant}$, an extension proof-functional logic $\mathcal{L}^{\cap\cup}$ of [DdLS16] with *ad hoc* predicates for subtyping;
- $\mathsf{NJ}(\beta)$, a natural deduction system for derivations in first-order intuitionistic logic

with untyped lambda-terms.

Judgements in these systems take the following four forms below. On the right-hand sides of the turnstiles, M is an untyped lambda-term, Δ is a simply-typed lambda-term with strong conjunction, strong disjunction, and explicit coercions, and σ is a simple type formed using $\to, \cap,$ and $\cup$. The $r_\sigma[M]$ are typing predicates to be realized.

$$\begin{array}{llllll}
\Lambda^{\cap\cup}_{\mathsf{u}\leqslant} & B, & x_\iota & :\tau & \vdash & M : \sigma \\
\Lambda^{\cap\cup}_{\mathsf{t}\leqslant} & \Gamma^{@}, & x_\iota \mathbin{@} \iota & :\tau & \vdash & M \mathbin{@} \Delta : \sigma \\
\mathcal{L}^{\cap\cup}_{\leqslant} & \Gamma, & \iota & :\tau & \vdash & \Delta : \sigma \\
\mathsf{NJ}(\beta) & G_B, & r_\tau[x_\iota] & & \vdash & r_\sigma[M]
\end{array}$$

Note that B (resp. Γ) is obtained from $\Gamma^{@}$ by erasing all the $@\iota$ (resp. "$x@$"), and G_B is obtained by B by "realizing" all the $x_\iota{:}\tau$.

Our first contribution is to extend the typed lambda-calculus $\Lambda^{\cap\cup}_{\mathsf{t}}$ of [DL10] with *explicit coercions,* keeping decidability of type checking, and showing the isomorphism with the $\Lambda^{\cap\cup}_{\mathsf{u}\leqslant}$ type assignment system of [BDCd95].

Our second contribution is to show that the extended $\mathcal{L}^{\cap\cup}_{\leqslant}$ logic of subtyping corresponds to a realizability logic, so proposing a complete analysis of the relationship between Curry-style and Church-style typing and the associated logic for intersection, union, and subtyping.

Our third contribution is to present a decidable algorithm for subtyping in presence of intersection and union types. To our knowledge, this is the first subtyping algorithm combining both union and intersection. The algorithm is conceived to work for the minimal (sub)type theory Ξ (*i.e.* axioms 1 to 14, as presented in [BDCd95]), the latter theory being compatible with a set-based interpretation of $\to, \cap, \cup, \leqslant$ with function space, set intersection, and set union, respectively.

1.2 Related Work

We shortly list the main research lines involving type (assignment) systems with intersection, union and subtyping for (un)typed lambda-calculi, proof-functional logics containing "strong-operators", and realizability.

The formal investigation of soundness and completeness for a notion of realizability was initiated by Lopez-Escobar [LE85] and subsequently refined by Mints [Min89]. Following our previous paper [DdLS16], it is Mints' approach that we build on here.

Barbanera and Martini [BM94] studied three proof-functional operators, namely the *strong conjunction*, the *relevant implication* (related with Meyer-Routley's [MR72] system B^+), and the *strong equivalence* connective for double implication, relating those connectives with a suitable type assignments system, a realizability semantics and a completeness theorem.

Dezani-Ciancaglini, Ghilezan, and Venneri [DCGV97], investigated a *Curry-Howard* interpretation of intersection and union types (for Combinatory Logic): using the well-understood relation between *combinatory logic* and lambda-calculus, they encode type-free lambda-terms in suitable combinatory logic formulas and then

$$\frac{}{B \vdash M : \omega}\ (\omega) \qquad \frac{x{:}\sigma \in B}{B \vdash x : \sigma}\ (Var)$$

$$\frac{B \vdash M : \sigma_1 \quad B \vdash M : \sigma_2}{B \vdash M : \sigma_1 \cap \sigma_2}\ (\cap I) \qquad \frac{B \vdash M : \sigma_1 \cap \sigma_2 \quad i = 1,2}{B \vdash M : \sigma_i}\ (\cap E_i)$$

$$\frac{B \vdash M : \sigma_i \quad i = 1,2}{B \vdash M : \sigma_1 \cup \sigma_2}\ (\cup I_i) \qquad \frac{\begin{array}{l} B, x{:}\sigma_1 \vdash M : \sigma_3 \\ B, x{:}\sigma_2 \vdash M : \sigma_3 \quad B \vdash N : \sigma_1 \cup \sigma_2 \end{array}}{B \vdash M[N/x] : \sigma_3}\ (\cup E)$$

Fig. 1. The Intersection and Union Type Assignment System $\Lambda^{\cap\cup}_{\mathsf{u}}$ [BDCd95] (main rules).

$$\frac{\Gamma^{@}, x_\iota @ \iota{:}\sigma_1 \vdash M @ \Delta : \sigma_2}{\Gamma^{@} \vdash \lambda x_\iota . M @ \lambda \iota{:}\sigma_1 . \Delta : \sigma_1 \to \sigma_2}\ (\to I) \qquad \frac{\Gamma^{@} \vdash M @ \Delta : \sigma_i \quad i \in \{1,2\}}{\Gamma^{@} \vdash M @ \mathsf{in_i}\ \Delta : \sigma_1 \cup \sigma_2}\ (\cup I_i)$$

$$\frac{\Gamma^{@} \vdash M @ \Delta_1 : \sigma_1 \quad \Gamma^{@} \vdash M @ \Delta_2 : \sigma_2}{\Gamma^{@} \vdash M @ (\Delta_1 \cap \Delta_2) : \sigma_1 \cap \sigma_2}\ (\cap I) \qquad \frac{\Gamma^{@} \vdash M @ \Delta : \sigma_1 \cap \sigma_2 \quad i \in \{1,2\}}{\Gamma^{@} \vdash M @ \mathsf{pr_i}\ \Delta : \sigma_i}\ (\cap E_i)$$

$$\frac{\Gamma^{@}, x_\iota @ \iota{:}\sigma_1 \vdash M @ \Delta_1 : \sigma_3 \quad \Gamma^{@}, x_\iota @ \iota{:}\sigma_2 \vdash M @ \Delta_2 : \sigma_3 \quad \Gamma^{@} \vdash N @ \Delta_3 : \sigma_1 \cup \sigma_2}{\Gamma^{@} \vdash M[N/x_\iota] @ (\lambda \iota{:}\sigma_1 . \Delta_1 \cup \lambda \iota{:}\sigma_2 . \Delta_2)\ \Delta_3 : \sigma_3}\ (\cup E)$$

Fig. 2. The Typed Calculus $\Lambda^{\cap\cup}_{\mathsf{t}}$ [DL10] (main rules).

type them using intersection and union types. This is a complementary approach to the realizability-based one here and in [DdLS16].

Barbanera, Dezani-Ciancaglini, and de'Liguoro [BDCd95] introduced an untyped lambda-calculus $\Lambda^{\cap\cup}_{\mathsf{u}\leqslant}$ with related type assignment system featuring intersection and union types, and a powerful subtyping relation. The previous work [DL10] presented a typed calculus $\Lambda^{\cap\cup}_{\mathsf{t}}$ (without subtyping) that explored the relationship between the proof-functional intersections and unions and the truth-functional (strong) products and (strong) sums. In [DdLS16] we introduced the new notion of *essence* $\wr\Delta\wr$ of a typed lambda-term Δ, used to connect $\Lambda^{\cap\cup}_{\mathsf{t}}$ and $\Lambda^{\cap\cup}_{\mathsf{u}}$. Specifically,

$$\Gamma^{@} \vdash M @ \Delta : \sigma \text{ if and only if } \Gamma \vdash \Delta : \sigma \text{ and } \wr\Delta\wr =_\beta M.$$

We proved the isomorphism between $\Lambda^{\cap\cup}_{\mathsf{t}}$ and $\Lambda^{\cap\cup}_{\mathsf{u}}$, and we showed that $\mathcal{L}^{\cap\cup}$ can be thought of as a proof-functional logic. The present paper extends all the systems and logics of [DdLS16] and presents a comparative analysis of the (sub)type theories Ξ and Π of [BDCd95]: this motivates the use of the (sub)type theory Ξ with their natural correspondence with $\mathsf{NJ}(\beta)$.

2 System

The pseudo-syntax of σ, M, Δ, and the derived $M @ \Delta$ are defined using the following three syntactic categories:

$$\frac{\Gamma, \iota{:}\sigma_1 \vdash \Delta : \sigma_2}{\Gamma \vdash \lambda\iota{:}\sigma_1.\Delta : \sigma_1 \to \sigma_2}\ (\to I) \qquad \frac{\Gamma \vdash \Delta_1 : \sigma_1 \to \sigma_2 \quad \Gamma \vdash \Delta_2 : \sigma_1}{\Gamma \vdash \Delta_1\,\Delta_2 : \sigma_2}\ (\to E)$$

$$\frac{\Gamma \vdash \Delta_1 : \sigma_1 \quad \Gamma \vdash \Delta_2 : \sigma_2 \quad \wr\Delta_1\wr =_\beta \wr\Delta_2\wr}{\Gamma \vdash \Delta_1 \cap \Delta_2 : \sigma_1 \cap \sigma_2}\ (\cap I) \qquad \frac{\Gamma \vdash \Delta : \sigma_1 \cap \sigma_2 \quad i \in \{1,2\}}{\Gamma \vdash \mathsf{pr_i}\,\Delta : \sigma_i}\ (\cap E_i)$$

$$\frac{\Gamma \vdash \Delta : \sigma_i \quad i \in \{1,2\}}{\Gamma \vdash \mathsf{in_i}\,\Delta : \sigma_1 \cup \sigma_2}\ (\cup I_i) \qquad \frac{\begin{array}{l}\Gamma, \iota{:}\sigma_1 \vdash \Delta_1 : \sigma_3 \quad \wr\Delta_1\wr =_\beta \wr\Delta_2\wr \\ \Gamma, \iota{:}\sigma_2 \vdash \Delta_2 : \sigma_3 \quad \Gamma \vdash \Delta_3 : \sigma_1 \cup \sigma_2\end{array}}{\Gamma \vdash (\lambda\iota{:}\sigma_1.\Delta_1 \cup \lambda\iota{:}\sigma_2.\Delta_2)\,\Delta_3 : \sigma_3}\ (\cup E)$$

Fig. 3. The proof-functional logic $\mathcal{L}^{\cap\cup}$ (main rules).

$$\frac{}{\sigma \leqslant \sigma \cap \sigma}\ (1) \qquad \frac{\sigma_1 \leqslant \sigma_2 \quad \tau_1 \leqslant \tau_2}{\sigma_1 \cup \tau_1 \leqslant \sigma_2 \cup \tau_2}\ (8)$$

$$\frac{}{\sigma \cup \sigma \leqslant \sigma}\ (2) \qquad \frac{\sigma \leqslant \tau \quad \tau \leqslant \rho}{\sigma \leqslant \rho}\ (9)$$

$$\frac{i \in \{1,2\}}{\sigma_1 \cap \sigma_2 \leqslant \sigma_i}\ (3) \qquad \frac{}{\sigma \cap (\tau \cup \rho) \leqslant (\sigma \cap \tau) \cup (\sigma \cap \rho)}\ (10)$$

$$\frac{i \in \{1,2\}}{\sigma_i \leqslant \sigma_1 \cup \sigma_2}\ (4) \qquad \frac{}{(\sigma \to \tau) \cap (\sigma \to \rho) \leqslant \sigma \to (\tau \cap \rho)}\ (11)$$

$$\frac{}{\sigma \leqslant \omega}\ (5) \qquad \frac{}{(\sigma \to \rho) \cap (\tau \to \rho) \leqslant (\sigma \cup \tau) \to \rho}\ (12)$$

$$\frac{}{\sigma \leqslant \sigma}\ (6) \qquad \frac{}{\omega \leqslant \omega \to \omega}\ (13)$$

$$\frac{\sigma_1 \leqslant \sigma_2 \quad \tau_1 \leqslant \tau_2}{\sigma_1 \cap \tau_1 \leqslant \sigma_2 \cap \tau_2}\ (7) \qquad \frac{\sigma_2 \leqslant \sigma_1 \quad \tau_1 \leqslant \tau_2}{\sigma_1 \to \tau_1 \leqslant \sigma_2 \to \tau_2}\ (14)$$

Fig. 4. The (sub)type theory Ξ.

$$\frac{B \vdash M : \sigma \quad \sigma \leqslant \tau}{B \vdash M : \tau}\ (\leqslant) \qquad \frac{\Gamma^{@} \vdash M@\Delta : \sigma \quad \sigma \leqslant \tau}{\Gamma^{@} \vdash M@(\tau)\Delta : \tau}\ (\leqslant) \qquad \frac{\Gamma \vdash \Delta : \sigma \quad \sigma \leqslant \tau}{\Gamma \vdash (\tau)\Delta : \sigma}\ (\leqslant)$$

Fig. 5. Subsumption rule and Explicit Coercion Rules.

$$\begin{array}{l}\sigma ::= \omega \mid \phi \mid \sigma \to \sigma \mid \sigma \cap \sigma \mid \sigma \cup \sigma \\ M ::= x \mid \lambda x.M \mid M\,M \\ \Delta ::= \iota \mid \lambda\iota{:}\sigma.\Delta \mid \Delta\,\Delta \mid \Delta \cap \Delta \mid \Delta \cup \Delta \mid \mathsf{pr_1}\,\Delta \mid \mathsf{pr_2}\,\Delta \mid \mathsf{in_1}\,\Delta \mid \mathsf{in_2}\,\Delta \mid (\sigma)\Delta\end{array}$$

where ϕ denotes arbitrary constant types and ω denotes a special type that is inhabited by all terms. The operators $\cup$ and $\cap$ are well-known to be associative and commutative, so we'll note $\cup_i \sigma_i$ (resp. $\cap_i \sigma_i$) are shortcuts for $\sigma_1 \cup \ldots \cup \sigma_i$ (resp. $\sigma_1 \cap \ldots \cap \sigma_i$).

Figure 1 presents the main rules of the type assignment system of [BDCd95]: note that the type inference rules are not syntax-directed. Figure 2 presents the main rules of the typed calculus $\Lambda_{\mathsf{t}}^{\cap\cup}$ of [DL10]: note that this type system is completely syntax directed. Figure 3 presents the main rules of the proof-functional

logic, as presented in [DdLS16], where the essence function is presented in Definition 2.1, showing the syntactic relation between type free and typed lambda-terms.

Definition 2.1 [Proof Essence]
The essence function between pure and typed lambda-terms is defined as follows:

$$
\begin{array}{rcl}
\wr \iota \wr & \triangleq & x_\iota \\
\wr \lambda \iota{:}\sigma_1.\Delta \wr & \triangleq & \lambda x_\iota.\wr\Delta\wr \\
\wr \Delta_1\ \Delta_2 \wr & \triangleq & \wr\Delta_1\wr\wr\Delta_2\wr \\
\wr (\lambda \iota{:}\sigma_1.\Delta_1 \cup \lambda \iota{:}\sigma_2.\Delta_2)\ \Delta_3 \wr & \triangleq & \wr\Delta_1\wr[\wr\Delta_3\wr/x_\iota] \quad \text{if } \wr\Delta_1\wr =_\beta \wr\Delta_2\wr \\
\wr \Delta_1 \cap \Delta_2 \wr & \triangleq & \wr\Delta_1\wr \quad \text{if } \wr\Delta_1\wr =_\beta \wr\Delta_2\wr \\
\wr \mathsf{pr}_\mathsf{i}\ \Delta \wr & \triangleq & \wr\Delta\wr \\
\wr \mathsf{in}_\mathsf{i}\ \Delta \wr & \triangleq & \wr\Delta\wr \\
\wr (\sigma)\Delta \wr & \triangleq & \wr\Delta\wr
\end{array}
$$

The logic $\mathcal{L}^{\cap\cup}$, introduced in [DdLS16] is a *proof-functional* logic, in the sense of Pottinger [Pot80] and Lopez-Escobar [LE85]: formulas encode, using the Curry-Howard isomorphism, *derivations* $\mathcal{D} : B \vdash M : \sigma$ in the type assignment system $\Lambda^{\cap\cup}_{\mathsf{u}}$ which are, in turn, isomorphic to typed judgments $\Gamma^{@} \vdash M@\Delta : \sigma$ of $\Lambda^{\cap\cup}_{\mathsf{t}}$. The next theorem recall the above properties: the key concept is the essence partial map $\wr{-}\wr$. This is, to the best of our knowledge, the first attempt to interpret union, intersection, and subtyping as explicit coercions as proof-functional connectives.

Theorem 2.2 (Equivalence)

(i) $B \vdash M : \sigma$ *iff* $\Gamma^{@} \vdash M@\Delta : \sigma$ *and* $\wr\Delta\wr =_\beta M$, *where* B *is obtained by erasing all the* $@\iota$ *in* $\Gamma^{@}$;

(ii) $\Gamma^{@} \vdash M@\Delta : \sigma$ *iff* $\Gamma \vdash \Delta : \sigma$ *and* $\wr\Delta\wr =_\beta M$ *where* Γ *is obtained by erasing all the* $x@$ *in* $\Gamma^{@}$;

(iii) $B \vdash M : \sigma$ *iff* $\Gamma \vdash \Delta : \sigma$ *and* $\wr\Delta\wr =_\beta M$, *where* B *is obtained by substituting all the* ι *with* x_ι.

Proof. *Using Theorem 10 of [DL10].* □

It is worth noticing that if we drop the restriction concerning the "essence" in rules $(\cap I)$ and $(\cup E)$ in the system $\mathcal{L}^{\cap\cup}$ and replace $\sigma \cap \tau$ by $\sigma \times \tau$, and $\sigma \cup \tau$ by $\sigma + \tau$ then we get a simply typed lambda-calculus with product and sums, namely a truth-functional intuitionistic propositional logic with implication, conjunction, and disjunction in disguise.

The whole picture is now ready to be extended with the subtyping relation, as introduced in [BCDC83] and extended in [BDCd95]. Subtyping is a preorder over types, and it is written as $\sigma \leqslant \tau$: we use the standard terminology of "(sub)type theories" for any collection of inequalities between types satisfying natural closure conditions. As such, the (sub)type theory, called Ξ in [BDCd95], is defined by the subtyping axioms and inference rules defined in Figure 4. The theory Ξ suggests the interpretation of ω as the *set universe*, of $\cap$ as the *set intersection*, of $\cup$ as the

$$\frac{G_\Gamma \vdash_{\mathsf{NJ}(\beta)} \bot}{G_\Gamma \vdash_{\mathsf{NJ}(\beta)} A}\ (\bot) \qquad \frac{A \in G_\Gamma}{G_\Gamma \vdash_{\mathsf{NJ}(\beta)} A}\ (Hyp)$$

$$\frac{G_\Gamma, A \vdash_{\mathsf{NJ}(\beta)} B}{G_\Gamma \vdash_{\mathsf{NJ}(\beta)} A \supset B}\ (\supset I) \qquad \frac{G_\Gamma \vdash_{\mathsf{NJ}(\beta)} A \supset B \quad G_\Gamma \vdash_{\mathsf{NJ}(\beta)} A}{G_\Gamma \vdash_{\mathsf{NJ}(\beta)} B}\ (\supset E)$$

$$\frac{G_\Gamma \vdash_{\mathsf{NJ}(\beta)} A \quad G_\Gamma \vdash_{\mathsf{NJ}(\beta)} B}{G_\Gamma \vdash_{\mathsf{NJ}(\beta)} A \wedge B}\ (\wedge I) \qquad \frac{G_\Gamma \vdash_{\mathsf{NJ}(\beta)} A_1 \wedge A_2 \quad i = 1,2}{G_\Gamma \vdash_{\mathsf{NJ}(\beta)} A_i}\ (\wedge E_i)$$

$$\frac{G_\Gamma \vdash_{\mathsf{NJ}(\beta)} A_i \quad i = 1,2}{G_\Gamma \vdash_{\mathsf{NJ}(\beta)} A_1 \vee A_2}\ (\vee I_i) \qquad \frac{\begin{array}{l} G_\Gamma, A \vdash_{\mathsf{NJ}(\beta)} C \\ G_\Gamma, B \vdash_{\mathsf{NJ}(\beta)} C \end{array} \quad G_\Gamma \vdash_{\mathsf{NJ}(\beta)} A \vee B}{G_\Gamma \vdash_{\mathsf{NJ}(\beta)} C}\ (\vee E)$$

$$\frac{G_\Gamma \vdash_{\mathsf{NJ}(\beta)} A}{G_\Gamma \vdash_{\mathsf{NJ}(\beta)} \forall x.A}\ (\forall I) \qquad \frac{G_\Gamma \vdash_{\mathsf{NJ}(\beta)} \forall x.A}{G_\Gamma \vdash_{\mathsf{NJ}(\beta)} A[M/x]}\ (\forall E)$$

$$\frac{G_\Gamma \vdash_{\mathsf{NJ}(\beta)} A[M/x]}{G_\Gamma \vdash_{\mathsf{NJ}(\beta)} \exists x.A}\ (\exists I) \qquad \frac{G_\Gamma \vdash_{\mathsf{NJ}(\beta)} \exists x.A \quad G_\Gamma, A \vdash_{\mathsf{NJ}(\beta)} B}{G_\Gamma \vdash_{\mathsf{NJ}(\beta)} B}\ (\exists E)$$

$$\frac{G_\Gamma \vdash_{\mathsf{NJ}(\beta)} \mathbf{P}_\phi(M) \quad M =_{\beta\eta} N}{G_\Gamma \vdash_{\mathsf{NJ}(\beta)} \mathbf{P}_\phi(N)}\ (\beta\eta) \qquad \frac{}{G_\Gamma \vdash_{\mathsf{NJ}(\beta)} \top}\ (\top)$$

Fig. 6. The Logic $\mathsf{NJ}(\beta)$

set union, and of $\leqslant$ as a sound (but not complete) *subset relation*, respectively. In the following, we write $\sigma \sim \tau$ iff $\sigma \leqslant \tau$ and $\tau \leqslant \sigma$. The astute reader will also notice that distributivity of union over intersection and intersection over union, *i.e.*

$$\sigma \cup (\tau \cap \rho) \sim (\sigma \cup \tau) \cap (\sigma \cup \rho) \quad \text{and} \quad \sigma \cap (\tau \cup \rho) \sim (\sigma \cap \tau) \cup (\sigma \cap \rho)$$

are derivable (see, *e.g.* derivation in [BDCd95], page 9).

Once the subtyping preorder has been defined, a classical subsumption or an explicit coercion rule, as shown in Figure 5, completes the presentation of the type assignment $\Lambda^{\cap\cup}_{\mathsf{u}\leqslant}$, of the typed system $\Lambda^{\cap\cup}_{\mathsf{t}\leqslant}$, and of the logic $\mathcal{L}^{\cap\cup}_{\leqslant}$, respectively. We sometimes write $\vdash_{\leqslant}$ for judgments using rules of Figure 5, if it is not clear from the context.

Theorem 2.3 (Conservativity) *The typed system $\Lambda^{\cap\cup}_{\mathsf{t}\leqslant}$ and the logic $\mathcal{L}^{\cap\cup}_{\leqslant}$, both obtained by extending with the (sub)type theory Ξ of Figure 4 and with explicit coercions of Figure 5, preserve parallel subject reduction, Church-Rosser, strong normalization, unicity of typing, decidability of type reconstruction and of type checking, judgment decidability and isomorphism of typed-untyped derivations.*

Proof. *The proof proceeds by upgrading Theorem 2 of [DdLS16] with subtyping as explicit coercions. Note that the extension of Theorem 2.2 also holds.* □

3 Realizers

We start this section by recalling the logic $\mathsf{NJ}(\beta)$: by NJ we mean the natural deduction presentation of the intuitionistic first-order predicate calculus [Pra65]. Derivations in NJ are trees of judgments $G \vdash_{\mathsf{NJ}} A$, where G is a set of undischarged assumptions, rather than trees of formulas as in Gentzen's original formulation.

Definition 3.1 (Logic $\mathsf{NJ}(\beta)$)
The system $\mathsf{NJ}(\beta)$, extending NJ, is the natural deduction system for first-order intuitionistic logic shown in Figure 6.

In [DdLS16], we provided a foundation for the proof-functional logic $\mathcal{L}^{\cap\cup}$ by extending Mints' provable realizability to cope with union types. More precisely, we write $r_\sigma[M]$ to denote a formula in $\mathsf{NJ}(\beta)$, realized by the pure lambda-term M of type σ. Observe that M is "distilled" by applying the essence function to the typed lambda-term Δ, which faithfully encodes the type assignment derivation $B \vdash \wr\Delta\wr : \sigma$ in $\Lambda^{\cap\cup}_{\mathsf{u}}$: this enforces the proof-functional nature of $\mathcal{L}^{\cap\cup}$.

The following definition is a reminder of the notion of realizer, as first introduced for intersection types by Mints [Min89], and extended by the authors in [DdLS16].

Definition 3.2 (Mints' realizers in $\mathsf{NJ}(\beta)$)
Let $\mathbf{P}_\phi(x)$ be a unary predicate for each atomic type ϕ. Then we define the predicates $r_\sigma[x]$ for each type σ by induction over σ, as follows:

$$\begin{array}{lll} r_\phi[x] \triangleq \mathbf{P}_\phi(x) & & r_{\sigma_1\to\sigma_2}[x] \triangleq \forall y.r_{\sigma_1}[y] \supset r_{\sigma_2}[x\,y] \\ r_\omega[x] \triangleq \top & & r_{\sigma_1\cup\sigma_2}[x] \triangleq r_{\sigma_1}[x] \vee r_{\sigma_2}[x] \\ & & r_{\sigma_1\cap\sigma_2}[x] \triangleq r_{\sigma_1}[x] \wedge r_{\sigma_2}[x] \end{array}$$

where $\supset$ denotes implication, $\wedge$ and $\vee$ are the logical connectives for conjunction and disjunction respectively, that must be kept distinct from $\cap$ and $\cup$. Formulas have the shape $r_\sigma[M]$, whose intended meaning is that M is a method for σ in the intersection-union type discipline with subtyping.

To prove soundness and completeness between $\mathcal{L}^{\cap\cup}_{\leqslant}$ and $\mathsf{NJ}(\beta)$, we will make use of a "bridge" system, a modification of $\Lambda^{\cap\cup}_{\mathsf{u}\leqslant}$, where $Dom(B)$ is contains type-free lambda-terms M instead of simple variables: note that a similar technique was used in [BCDC83] (see Corollary 4.10 there) to prove the conservativity of $\Lambda^{\cap\cup}_{\mathsf{u}\leqslant}$ over the Simple Curry Type Assignment System.

Definition 3.3 (Large Type Assignment System $\vdash^*$)

(i) A *large* context, denoted by $\mathbf{B}$, is any arbitrary set of statements $M{:}\sigma$. To emphasize the difference, ordinary contexts B are called *small*;

(ii) Let $\vdash^*$ denote the type assignment system $\Lambda^{\cap\cup}_{\mathsf{u}\leqslant}$ of Figure 1 where all contexts are large and the following rule is added:

$$\frac{\mathbf{B} \vdash^* N : \sigma \quad M =_{\beta\eta} N}{\mathbf{B} \vdash^* M : \sigma}\ (Eq_{\beta\eta})$$

We define $G_{\mathbf{B}} \triangleq r_{\sigma_1}[M_1], \ldots, r_{\sigma_n}[M_n]$ and $\mathbf{B} \triangleq \{M_1{:}\sigma_1, \ldots, M_n{:}\sigma_n\}$.

The following lemma shows some properties of $\vdash^*$: a natural conservativity property and a subtyping property that cannot be proved in the type assignment system of Figure 1.

Lemma 3.4 (Conservativity *w.r.t.* [BDCd95])
If $B \vdash_{\leqslant} M : \sigma$, then for any N such that $M =_{\beta\eta} N$, we have that $B \vdash^* N : \sigma$. Conversely, if $B \vdash^* N : \sigma$, then there exists M such that $M =_{\beta\eta} N$ and $B \vdash_{\leqslant} M : \sigma$.

Proof. *If $B \vdash_{\leqslant} M : \sigma$, then $B \vdash^* M : \sigma$, and for any $M =_{\beta\eta} N$ we can show that $B \vdash^* N : \sigma$ thanks to the rule $(Eq_{\beta\eta})$. The other part of the proof is proved by showing that the $(Eq_{\beta\eta})$ rule can be safely be postponed at the end of the derivation (see Lemma A.1 in the appendix for referees).* □

We can now state that the large type assignment system $\vdash^*$ is sound and complete *w.r.t.* Mints' realizers in $\mathsf{NJ}(\beta)$.

Theorem 3.5 ($\vdash^*$ *versus* $\mathsf{NJ}(\beta)$) $\mathbf{B} \vdash^* M : \sigma$ *iff* $G_{\mathbf{B}} \vdash_{\mathsf{NJ}(\beta)} r_\sigma[M]$.

Proof. *The complete proof given in the appendix for referees.* □

Informally speaking, $r_\sigma[M]$ can be interpreted as "M is an element of the set σ", and the judgment $\sigma_1 \leqslant \sigma_2$ in the (sub)type theory Ξ can be interpreted as $r_{\sigma_1}[x] \vdash_{\mathsf{NJ}(\beta)} r_{\sigma_2}[x]$. The next lemma relates the large system with the typed system $\Lambda^{\cap\cup}_{\mathsf{u}\leqslant}$ and the logic $\mathcal{L}^{\cap\cup}_{\leqslant}$ as follows:

Lemma 3.6

(i) $\Gamma^{@} \vdash_{\leqslant} M\,@\Delta : \sigma$ *iff* $B \vdash^* N : \sigma$ *and* $M =_{\beta\eta} N$ *where B is the $@\iota$-erasing of $\Gamma^{@}$;*

(ii) $\Gamma \vdash_{\leqslant} \Delta : \sigma$ *iff* $B \vdash^* M : \sigma$ *and* $\wr\Delta\wr =_\beta M$ *where B is Γ where all occurrence of ι are substituted with x_ι.*

Proof. *By Theorems 2.2 and 2.3, and Lemma 3.4.* □

As a simple consequence of Theorem 3.5 and Lemma 3.6, we can now state one of the main results of this paper:

Theorem 3.7 (Soundness and Completeness of $\mathsf{NJ}(\beta)$ and $\mathcal{L}^{\cap\cup}_{\leqslant}$)

(i) *If $\Gamma \vdash_{\leqslant} \Delta : \sigma$ then $G_\Gamma \vdash_{\mathsf{NJ}(\beta)} r_\sigma[\wr\Delta\wr]$.*

(ii) *If $G_\Gamma \vdash_{\mathsf{NJ}(\beta)} r_\sigma[M]$ then there exists Δ such that $\Gamma \vdash_{\leqslant} \Delta : \sigma$ and $\wr\Delta\wr =_{\beta\eta} M$.*

Remark 3.8
The type assignment system $\Lambda^{\cap\cup}_{\mathsf{u}\leqslant}$ of [BDCd95] was based on the (sub)type theory Ξ of Figure 4 (see Definition 3.6 of [BDCd95]): the paper also introduced a stronger (sub)type theory, called Π, by adding the extra axiom

$$(15) \qquad \mathbf{P}(\sigma) \Rightarrow \sigma \to \tau \cup \rho \leqslant (\sigma \to \rho) \cup (\sigma \to \rho),$$

where $\mathbf{P}(\sigma)$ is true if σ syntactically corresponds to an Harrop formula. However, in $\mathsf{NJ}(\beta)$, the judgment $r_{\sigma\to(\tau\cup\rho)}[x] \vdash_{\mathsf{NJ}(\beta)} r_{(\sigma\to\tau)\cup(\sigma\to\rho)}[x]$ is *not* derivable because the judgment $A \supset (B \vee C) \vdash_{\mathsf{NJ}(\beta)} (A \supset B) \vee (A \supset C)$ is *not* derivable in NJ. As such, the (sub)type theory Π cannot be overlapped with an interpretation of (sub)types

as (sub)sets, as the following example show. The identity function $\lambda x.x$ inhabits the function set $\{a, b\} \to \{a\} \cup \{b\}$ but, by axiom (15), it should also inhabits $\{a, b\} \to \{a\}$ or $\{a, b\} \to \{b\}$, which is clearly not the case.

4 Subtyping algorithm

The previous section showed that the proof-functional logic $\mathcal{L}^{\cap\cup}_{\leqslant}$ is sound and complete *w.r.t.* the logic $\mathsf{NJ}(\beta)$. The truth of the sequent "$\Gamma \vdash_{\leqslant} \Delta : \sigma$", complicates its decidability because of the presence of the predicate $\sigma \leqslant \tau$ as a premise in rule ($\leqslant$) of Figure 5: in fact, the subtype system *is not* an algorithm because of the presence of reflexivity and transitivity rules that are not syntax-directed. The same subtyping premise can affect the decidability of type checking of $\Lambda^{\cap\cup}_{\mathsf{t}\leqslant}$. This section presents a decidable algorithm $\mathcal{A}$ for subtyping in the (sub)type theory Ξ: as far as we know, this is the first attempt to study decidability of subtyping in presence of union and intersection types. The algorithm $\mathcal{A}$ needs four decidable subroutines, all implemented using term rewriting systems:

- $\mathcal{R}_1$, to simplify the shape of types containing the ω type: its complexity is linear;
- $\mathcal{R}_2$ (well-known), to transform a type in its *conjunctive normal form*, denoted by CNF, *i.e.* types being, roughly, *intersection of unions*: its complexity is exponential;
- $\mathcal{R}_3$ (well-known), to transform a type in its *disjunctive normal form*, denoted by DNF, *i.e.* types being, roughly, *union of intersections*: its complexity is exponential;
- $\mathcal{R}_4$, to transform a type in its *arrow normal form*, denoted by ANF, *i.e.* types being, roughly, arrow types where all the domains are intersection of ANF and all the codomains are union of ANF: its complexity is exponential.

Our algorithm $\mathcal{A}$ is proved to be sound *w.r.t.* the (sub)type theory Ξ. In what follows we use the following useful shorthands:

$$\cap_i(\cup_j \sigma_{i,j}) \triangleq \cap_1(\cup_1 \sigma_{1,1} \ldots \cup_j \sigma_{1,j}) \ldots \cap_i (\cup_1 \sigma_{i,1} \ldots \cup_j \sigma_{i,j}), \text{ and}$$

$$\cup_i(\cap_j \sigma_{i,j}) \triangleq \cup_1(\cap_1 \sigma_{1,1} \ldots \cap_j \sigma_{1,j}) \ldots \cup_i (\cap_1 \sigma_{i,1} \ldots \cap_j \sigma_{i,j})$$

Those shorthands can also apply to unions of unions, intersections of intersections, intersections of arrows, etc.

Definition 4.1 (Subroutine $\mathcal{R}_1$)
The term rewriting system $\mathcal{R}_1$ is defined as follows:

- $\omega \cap \sigma$ and $\sigma \cap \omega$ rewrite to σ;
- $\omega \cup \sigma$ and $\sigma \cup \omega$ rewrite to ω;
- $\sigma \to \omega$ rewrites to ω.

It is easy to verify that $\mathcal{R}_1$ terminates and his complexity is linear.

The next definition recall the usual *conjunctive/disjunctive normal form* with corresponding subroutines $\mathcal{R}_2$ and $\mathcal{R}_3$, and introduce the *arrow normal form* with his corresponding subroutine $\mathcal{R}_4$.

Definition 4.2 (Subroutines $\mathcal{R}_2$ and $\mathcal{R}_3$)

- A type is in *conjunctive normal form* (CNF) if it has the form $\cap_i(\cup_j \sigma_{i,j})$, and all the $\sigma_{i,j}$ are either atomic types, arrow types, or ω.
- The term rewriting system $\mathcal{R}_2$ rewrites a type in its CNF; it is defined as follows:
  - $\sigma \cup (\tau \cap \rho)$ rewrites to $(\sigma \cup \tau) \cap (\sigma \cup \rho)$.
  - $(\sigma \cap \tau) \cup \rho$ rewrites to $(\sigma \cup \rho) \cap (\tau \cup \rho)$.
- A type is in *disjunctive normal form* (DNF) if it has the form $\cup_i(\cap_j \sigma_{i,j})$, and all the $\sigma_{i,j}$ are either atomic types, arrow types, or ω.
- The term rewriting system $\mathcal{R}_3$ rewrites a type in its DNF; it is defined as follows:
  - $\sigma \cap (\tau \cup \rho)$ rewrites to $(\sigma \cap \tau) \cup (\sigma \cap \rho)$;
  - $(\sigma \cup \tau) \cap \rho$ rewrites to $(\sigma \cap \rho) \cup (\tau \cup \rho)$.

It is well documented in the literature that $\mathcal{R}_2$ and $\mathcal{R}_3$ terminate and that the complexity of those algorithms is exponential.

Definition 4.3 (Subroutine $\mathcal{R}_4$)

- A type is in *arrow normal form* (ANF) if :
  - it is an atomic type or ω;
  - it is an arrow type in the form $(\cap_i \sigma_i) \to (\cup_j \tau_j)$, where the σ_i and τ_j are ANFs;
- The term rewriting system $\mathcal{R}_4$ rewrites an arrow type into an *intersection* of ANF; it is defined as follows:
  - $\sigma \to \tau$ rewrites to $\mathcal{R}_3(\sigma) \to \mathcal{R}_2(\tau)$;
  - $\cup_i \sigma_i \to \cap_h \tau_h$ rewrites to $\cap_i(\cap_h(\sigma_i \to \tau_h))$.

Since $\mathcal{R}_2$ and $\mathcal{R}_3$ terminate, also that $\mathcal{R}_4$ terminates and his complexity is exponential.

Lemma 4.4 *For all the term rewriting systems $\mathcal{R}_{1,2,3,4}$ we have that $\mathcal{R}(\sigma) \sim \sigma$.*

Proof. *Each rewriting rule rewrites a term into an equivalent ($\sim$) term.* □

The next definition introduce the "preprocessing" of types necessary to feed correctly the algorithm $\mathcal{A}$.

Definition 4.5

- A type is in *conjunctive-arrow normal form* (CANF) if it is in CNF and all the arrow type subterms are in ANF. The composition of the term rewriting systems $\mathcal{R}_2 \circ \mathcal{R}_4 \circ \mathcal{R}_1$ rewrite a type into its CANF.
- A type is in *disjunctive-arrow normal form* (DANF) if it is in DNF and all the arrow type subterms are in ANF. The composition of the term rewriting systems $\mathcal{R}_3 \circ \mathcal{R}_4 \circ \mathcal{R}_1$ rewrite a type into its DANF.

4.1 The algorithm $\mathcal{A}$

Our algorithm $\mathcal{A}$ accepts or rejects an $\sigma \leqslant \tau$ formula: in a nutshell the algorithm $\mathcal{A}$ proceeds as follows: using $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{R}_3$ and $\mathcal{R}_4$, we first "preprocess" σ into a DANF and τ into a CANF producing a (normalized) subtyping statement of the shape $\cup_i(\cap_j \sigma_{i,j}) \leqslant \cap_h(\cup_k \tau_{h,k})$, where all the $\sigma_{i,j}, \tau_{h,k}$ are in ANF; then we call $\mathcal{A}_1$.

More precisely, $\mathcal{A}$ is composed by two mutually inductive functions, called $\mathcal{A}_1$ and $\mathcal{A}_2$.

Definition 4.6 (Main function $\mathcal{A}_1$)
input: $\cup_i(\cap_j \sigma_{i,j}) \leqslant \cap_h(\cup_k \tau_{h,k})$ where all the $\sigma_{i,j}, \tau_{h,k}$ are ANF; **output:** bool.

- if $\cap_h(\cup_k \tau_{h,k})$ is ω, then accept;
- if, for all i and h, there exists some j and some k, such that $\mathcal{A}_2(\sigma_{i,j} \leqslant \tau_{h,k})$ is true, then accept, else reject.

Definition 4.7 (Subtyping function $\mathcal{A}_2$)
input: $\sigma \leqslant \tau$, where σ and $\tau \not\equiv \omega$ are ANFs; **output:** bool.

- Case $\omega \leqslant \phi$: reject;
- Case $\omega \leqslant \sigma \to \tau$: reject;
- Case $\phi \leqslant \phi'$: accept if $\phi \equiv \phi'$, else reject;
- Case $\phi \leqslant \sigma \to \tau$: reject;
- Case $\sigma \to \tau \leqslant \phi$: reject;
- Case $\sigma \to \tau \leqslant \sigma' \to \tau'$: accept if $\mathcal{A}_1(\sigma' \leqslant \sigma)$ and $\mathcal{A}_1(\tau \leqslant \tau')$, else reject.

Algorithms $\mathcal{A}_1$, and $\mathcal{A}_2$ are polynomial, but the preprocessing of $\sigma \leqslant \tau$ using the subroutines $\mathcal{R}_{1,2,3,4}$ makes the whole exponential.

The following two technical lemmas are useful to prove soundness of the algorithm $\mathcal{A}_1$.

Lemma 4.8

(i) $\sigma \cup \tau \leqslant \rho \Longleftrightarrow \sigma \leqslant \rho$ *and* $\tau \leqslant \rho$

(ii) $\sigma \leqslant \tau \cap \rho \Longleftrightarrow \sigma \leqslant \tau$ *and* $\sigma \leqslant \rho$

Proof. *The two parts can be proved by examining the subtyping rules of the (sub)type theory* Ξ. □

Lemma 4.9
If all the σ_i *and* τ_j *are ANFs, then*

(i) $\exists j, \cap_i \sigma_i \leqslant \tau_j \Longrightarrow \cap_i \sigma_i \leqslant \cup_j \tau_j$

(ii) $\exists i, \sigma_i \leqslant \cup \tau_j \Longrightarrow \cap_i \sigma_i \leqslant \cup_j \tau_j$

Proof. *The two parts can be proved by induction on the subtyping rules of the (sub)type theory* Ξ *using the ANF definition.* □

Theorem 4.10 ($\mathcal{A}_1, \mathcal{A}_2$'s Soundness)

(i) *Let* σ *(resp.* τ*) be in DANF (resp. CANF). If* $\mathcal{A}_1(\sigma \leqslant \tau)$ *then* $\sigma \leqslant \tau$.

(ii) *Let* σ *and* $\tau \not\equiv \omega$ *be in ANF. If* $\mathcal{A}_2(\sigma \leqslant \tau)$ *then* $\sigma \leqslant \tau$.

Proof. *The proof proceeds by mutual induction, where base case is in proving part* (ii).

(i) *By case analysis on the algorithm* $\mathcal{A}_1$ *using Lemmas 4.8 and 4.9 and part (ii);*

(ii) *By case analysis on the algorithm* $\mathcal{A}_2$*, and by looking at the subtyping rules.*

□

The completeness of the algorithm is proved in Theorem A.16 in the appendix for referees.

5 Conclusions

We mention some future research directions.

Strong/Relevant Implication is another proof-functional connective: as well explained in [BM94], it can be viewed as a special case of implication "whose related function space is the simplest one, namely the one containing only the *identity* function". Relevant implication is well-known in the literature, corresponding to Meyer and Routley's Minimal Relevant Logic B^+ [MR72]. Following our parallelism between type systems for lambda calculi *à la* Curry, *à la* Church, and logics, we could conjecture that strong implication, denoted by $\supset_r$ in the logic, by $\to_r$ in the type theory, and by λ_r in the typed lambda calculus, can lead to the following type (assignment) rules, proof-functional logical inference, and Mints' realizer in $\mathsf{NJ}(\beta)$, respectively:

$$\frac{B \vdash \mathsf{I} : \sigma \to \tau}{B \vdash \mathsf{I} : \sigma \to_r \tau} \quad \frac{\Gamma^{@}, x_\iota @ \iota{:}\sigma \vdash x_\iota @ \Delta : \tau}{\Gamma^{@} \vdash \lambda x_\iota . x_\iota @ \lambda_r \iota{:}\sigma . \Delta : \sigma \to_r \tau} \quad \frac{\Gamma, \iota{:}\sigma \vdash \Delta : \tau \quad \wr \Delta \wr =_\beta \iota}{\Gamma \vdash \lambda_r \iota{:}\sigma . \Delta : \sigma \to_r \tau} \quad \frac{G_B \vdash r_{\sigma\to\tau}[\mathsf{I}]}{G_B \vdash r_{\sigma\to_r\tau}[\mathsf{I}]}$$

As showed in Remark 3.8, even a stronger (sub)type theory of Ξ (*i.e.* the theory Π of [BDCd95]) cannot be overlapped with a sound and *complete* interpretation of (sub)types as (sub)sets. We also conjecture that, by extending the proof-functional logic with relevant implication ($\mathcal{L}^{\cap\cup}_{\leqslant\to_r}$), we could to achieve completeness, by combining explicit coercions and relevant abstractions as the following derivation shows:

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash \iota : \sigma \quad \sigma \leqslant \tau}{\Gamma \vdash (\tau)\iota : \sigma} \qquad \wr(\tau)\iota\wr =_\beta \iota}{\Gamma \vdash \lambda_r \iota{:}\sigma . (\tau)\iota : \sigma \to_t \tau} \qquad \Gamma \vdash \Delta : \sigma}{\Gamma \vdash (\lambda_r \iota{:}\sigma . (\tau)\iota)\, \Delta : \tau}$$

Dependent Types / Logical Frameworks. Our aim is to build a small logical framework, *à la* the Edinburgh Logical Framework [HHP93], featuring dependent types and proof-functional logical connectives compatible with a set-based interpretation of $\to, \cap, \cup$ with function space, set intersection, and set union, respectively, where Zermelo-Fraenkel axiom of subsets can be recovered by combination of $\to_r$ and $\leqslant$, as shown above. We conjecture that, in addition to the usual machinery dealing with dependent types, the following typing rules can be good candidates for a

proof-functional LF extension:

$$\frac{\Gamma, \iota{:}\sigma \vdash \Delta : \tau \quad \wr\Delta\wr =_\beta x_i}{\Gamma \vdash \lambda^r \iota{:}\sigma.\Delta : \Pi^r \iota{:}\sigma.\tau}\ (\Pi^r I) \qquad \frac{\Gamma \vdash \Delta_1 : \sigma_1 \quad \Gamma \vdash \Delta_2 : \sigma_2 \quad \wr\Delta_1\wr =_\beta \wr\Delta_2\wr}{\Gamma \vdash \Delta_1 \cap \Delta_2 : \sigma_1 \cap \sigma_2}\ (\cap I)$$

$$\frac{\begin{array}{ll}\Gamma \vdash \Delta_1 : \Pi\iota'{:}\sigma_1.\sigma_3[\mathsf{in}_1\ \iota'/\iota] & \wr\Delta_1\wr =_\beta \wr\Delta_2\wr \\ \Gamma \vdash \Delta_2 : \Pi\iota'{:}\sigma_2.\sigma_3[\mathsf{in}_2\ \iota'/\iota] & \Gamma \vdash \Delta_3 : \sigma_1 \cup \sigma_2\end{array}}{\Gamma \vdash (\Delta_1 \cup \Delta_2)\ \Delta_3 : \sigma_3[\Delta_3/\iota]}\ (\cup E)$$

Studying the behavior of proof-functional connectives would be beneficial to existing interactive theorem provers such as Coq [Coq17] or Isabelle [Isa17], and dependently typed programming languages such as Agda [Agd17], Beluga [Bel17], Epigram [Epi17], or Idris [Idr17].

Prototype Implementation. We are current implementing a small kernel for a logical framework featuring union and intersection types, as the $\Lambda_{\mathsf{t}}^{\cap\cup}$ calculus and the proof-functional logic $\mathcal{L}^{\cap\cup}$ does. The actual type system also features an experimental implementation of dependent-types *à la* LF following the above type rules, and of a *Read-Eval-Print-Loop* (REPL). We will put our future efforts to integrate our algorithm $\mathcal{A}$ to the type checker engine.

The aim of the prototype is to check the expressiveness of the proof-functional nature of the logical engine in the sense that when the user must prove *e.g.* a strong conjunction formula $\sigma_1 \cap \sigma_2$ obtaining (mostly interactively) a witness Δ_1 for σ_1, the prototype can "squeeze" the essence M of Δ_1 to accelerate, and in some case automatize, the construction of a witness Δ_2 proof for the formula σ_2 having the same essence M of Δ_1. Existing proof assistants could get some benefit if extended with a proof-functional logic. We are also started an encoding of the proof-functional operators of intersection and union in Coq. The actual state of the prototype can be retrieved at `https://github.com/cstolze/Bull`.

Acknowledgment. We are grateful to Daniel Dougherty for useful suggestions and a careful reading of the document.

References

[Agd17] The Agda Programming Language. `http://wiki.portal.chalmers.se/agda/pmwiki.php`, 2017.

[Bar84] Henk P. Barendregt. *The λ-Calculus.* Studies in logic and the foundations of mathematics, North-Holland, 1984.

[BCDC83] Henk Barendregt, Mario Coppo, and Mariangiola Dezani-Ciancaglini. A Filter Lambda Model and the Completeness of Type Assignment. *Journal of Symbolic Logic*, 48(4):931–940, 1983.

[BCL92] Kim B. Bruce, Roberto Di Cosmo, and Giuseppe Longo. Provable isomorphisms of types. *Mathematical Structures in Computer Science*, 2(2):231–247, 1992.

[BDCd95] Franco Barbanera, Mariangiola Dezani-Ciancaglini, and Ugo de'Liguoro. Intersection and union types: syntax and semantics. *Inf. Comput.*, 119(2):202–230, 1995.

[Bel17] The Beluga Programming Language. `http://complogic.cs.mcgill.ca/beluga/`, 2017.

[BL85] Kim B. Bruce and Giuseppe Longo. Provable isomorphisms and domain equations in models of typed languages (preliminary version). In *Proceedings of STOC*, pages 263–272, 1985.

[BM94] Franco Barbanera and Simone Martini. Proof-functional connectives and realizability. *Archive for Mathematical Logic*, 33:189–211, 1994.

[Coq17] The Coq Proof Assistant. https://coq.inria.fr/, 2017.

[DCGV97] Mariangiola Dezani-Ciancaglini, Silvia Ghilezan, and Betti Venneri. The "relevance" of intersection and union types. *Notre Dame Journal of Formal Logic*, 38(2):246–269, 1997.

[DdLS16] Daniel J. Dougherty, Ugo de'Liguoro, Luigi Liquori, and Claude Stolze. A realizability interpretation for intersection and union types. In *Programming Languages and Systems - 14th Asian Symposium, APLAS*, volume 10017 of *Lecture Notes in Computer Science*, pages 187–205. Springer, 2016.

[DL10] Daniel J. Dougherty and Luigi Liquori. Logic and computation in a lambda calculus with intersection and union types. In *Logic for Programming, Artificial Intelligence, and Reasoning - 16th International Conference, LPAR*, volume 6355 of *Lecture Notes in Computer Science*, pages 173–191. Springer, 2010.

[Epi17] The Epigram Programming Language. https://code.google.com/archive/p/epigram/, 2017.

[HHP93] Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. *J. ACM*, 40(1):143–184, 1993.

[Hin84] J. Roger Hindley. Coppo-Dezani types do not correspond to propositional logic. *Theor. Comput. Sci.*, 28:235–236, 1984.

[How80] William A. Howard. The Formulae-as-Types Notion of Construction. In *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic press, 1980.

[Idr17] The Idris Programming Language. http://www.idris-lang.org/, 2017.

[Isa17] The Isabelle Proof Assistant. https://isabelle.in.tum.de/, 2017.

[LE85] Edgar G. K. Lopez-Escobar. Proof functional connectives. In *Methods in Mathematical Logic*, volume 1130 of *Lecture Notes in Mathematics*, pages 208–221. Springer-Verlag, 1985.

[LR07] Luigi Liquori and Simona Ronchi Della Rocca. Intersection typed system *à la* Church. *Information and Computation*, 9(205):1371–1386, 2007.

[Min89] Grigori Mints. The completeness of provable realizability. *Notre Dame Journal of Formal Logic*, 30(3):420–441, 1989.

[MPS86] David B. MacQueen, Gordon D. Plotkin, and Ravi Sethi. An ideal model for recursive polymorphic types. *Information and Control*, 71(1/2):95–130, 1986.

[MR72] Robert K Meyer and Richard Routley. Algebraic analysis of entailment I. *Logique et Analyse*, 15:407–428, 1972.

[Pie91a] Benjamin C. Pierce. *Programming with intersection types, union types, and bounded polymorphism.* PhD thesis, Technical Report CMU-CS-91-205. Carnegie Mellon University, 1991.

[Pie91b] Benjamin C. Pierce. Programming with intersection types, union types, and polymorphism. Technical Report CMU-CS-91-106, Carnegie Mellon University, 1991.

[Pot80] Garrel Pottinger. A type assignment for the strongly normalizable λ-terms. In *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 561–577. Academic Press, 1980.

[Pra65] Dag Prawitz. *Natural deduction: a proof-theoretical study.* PhD thesis, Almqvist & Wiksell, 1965.

[Pra71] Dag Prawitz. Ideas and results in proof theory. In *Proceedings of the Second Scandinavian Logic Symposium.* North-Holland, 1971.

[Rey88] John C. Reynolds. Preliminary design of the programming language Forsythe. Report CMU–CS–88–159, Carnegie Mellon University, Pittsburgh, Pennsylvania, June 21, 1988.

[Rey96] John C. Reynolds. Design of the programming language Forsythe. Report CMU–CS–96–146, Carnegie Mellon University, Pittsburgh, Pennsylvania, June 28, 1996.

A Appendix (for Referees)

A.1 Proofs on the realizers system

Lemma A.1 (Postponement of the $(Eq_{\beta\eta})$ rule)
If $B \vdash^* M : \sigma$, then there exists N such that $B \vdash_{\leqslant} N : \sigma$ of $\Lambda^{\cap\cup}_{\mathsf{u}\leqslant}$ and $M =_{\beta\eta} N$.

Proof. *We proceed by induction on the derivation tree. As a representative case, we get by induction hypothesis that:*

$$\frac{\begin{array}{ccccccc} B,x{:}\sigma_1 \vdash_{\leqslant} M_1 : \sigma_3 & M =_{\beta\eta} M_1 & B,x{:}\sigma_2 \vdash_{\leqslant} M_2 : \sigma_3 & M =_{\beta\eta} M_2 & B \vdash_{\leqslant} N' : \sigma_1 \cup \sigma_2 & N =_{\beta\eta} N' \\ \vdots & & \vdots & & \vdots & \\ B,x{:}\sigma_1 \vdash^* M : \sigma_3 & & B,x{:}\sigma_2 \vdash^* M : \sigma_3 & & B \vdash^* N : \sigma_1 \cup \sigma_2 & \end{array}}{B \vdash^* M[N/x] : \sigma_3}\ (\cup E)$$

Using the Church-Rosser property on the Gross-Knuth parallel reduction (as defined in [Bar84], 13.2.7), we know that there is some M' such that M_1 and M_2 reduce in parallel to M'. As a consequence, we have that using subject (parallel) reduction of $\Lambda^{\cap\cup}_{\mathsf{u}\leqslant}$, we get $B, x{:}\sigma_1 \vdash_{\leqslant} M' : \sigma_3$ and $B, x{:}\sigma_2 \vdash_{\leqslant} M' : \sigma_3$, and therefore by $(\cup E)$ we get $B \vdash_{\leqslant} M'[N'/x] : \sigma_3$ and $M'[N'/x] =_{\beta\eta} M[N/x]$, as desired. □

Lemma A.2 (Substitution)
If $\mathbf{B}, x{:}\sigma \vdash^* M : \tau$ and $\mathbf{B} \vdash^* N : \sigma$, then $\mathbf{B} \vdash^* M[N/x] : \tau$

Proof. *By induction on the derivation tree.* □

As a step towards the proof of Theorem 3.5, consider the following new elimination rule for the large type assignment system:

$$\frac{\mathbf{B}, N{:}\sigma_1 \vdash^* M : \sigma_3 \quad \mathbf{B}, N{:}\sigma_2 \vdash^* M : \sigma_3 \quad \mathbf{B} \vdash^* N : \sigma_1 \cup \sigma_2}{\mathbf{B} \vdash^* M : \sigma_3}\ (\cup E^*)$$

This rule is an admissible rule, as the following lemma states:

Lemma A.3 *The rule $(\cup E^*)$ is admissible.*

Proof. *Let M and N such that N does not appears in M. We suppose we have a derivation tree $\mathcal{D}$ for $\mathbf{B}, N{:}\sigma_1 \vdash^* M[N/x] : \sigma_3$, a derivation tree $\mathcal{D}'$ for $\mathbf{B}, N{:}\sigma_2 \vdash^* M[N/x] : \sigma_3$ and a derivation tree $\mathcal{D}''$ for $\mathbf{B} \vdash^* N : \sigma_1 \cup \sigma_2$. We have to show that $\mathbf{B} \vdash^* M[N/x] : \sigma_3$.*

Let $S = \{\mathcal{D}_i\}_{1\leqslant i\leqslant n}$ (resp $S' = \{\mathcal{D}'_i\}_{1\leqslant j\leqslant n'}$) be the set of all the minimal (in the sense of tree inclusion) subtrees of $\mathcal{D}$ (resp. $\mathcal{D}'$) whose goal has the shape $\mathbf{B}, N{:}\sigma_1, B_i \vdash^* N : \tau_i$ (resp. $\mathbf{B}, N{:}\sigma_2, B'_j \vdash^* N : \tau'_j$) for some B_i, τ_i (resp. B'_j, τ'_j). Note that S and S' may be empty, and that B_i and B'_i are small bases. In the rest of the proof, B_i, B'_j, τ_i and τ'_i will refer to the corresponding bases and types in the $\mathcal{D}_i$ and $\mathcal{D}'_i$ derivation trees. Let τ (resp. τ') the intersection of all the τ_i and σ_1 (resp. τ'_i and σ_2). It is clear that, for any i and j, $\tau \leqslant \tau_i$ and $\tau' \leqslant \tau'_j$. Notice that none of the free variable of N appears in any of the B_i or B'_i, therefore if τ_i (resp. τ'_j) is not σ_1 (resp. σ_2), then we have that $\mathbf{B} \vdash^* N : \tau_i$ (resp. $\mathbf{B} \vdash^* N : \tau'_j$). Without loss of generality, we can assume that σ_1 is one of the τ_i and that σ_2 is one of the τ'_j. We construct a derivation of $\mathbf{B} \vdash^* M[N/x] : \sigma_3$ ending by:

$$\frac{\mathbf{B}, x{:}\tau \vdash^* M : \sigma_3 \quad \mathbf{B}, x{:}\tau' \vdash^* M : \sigma_3 \quad \mathbf{B} \vdash^* N : \tau \cup \tau'}{\mathbf{B} \vdash^* M[N/x] : \sigma_3}\ (\cup E)$$

We now have to prove that:

- $\mathbf{B}, x{:}\tau \vdash^* M : \sigma_3$*: we modify* $\mathcal{D}$ *by replacing* N *by* x *and by replacing all the* $\mathcal{D}_i \in S$ *subtrees by:*

$$\frac{\dfrac{}{\mathbf{B}, B_i, x{:}\tau \vdash^* x{:}\tau}\ (Var) \qquad \tau \leqslant \tau_i}{\mathbf{B}, B_i, x{:}\tau \vdash^* x{:}\tau_i}\ (\leqslant)$$

- $\mathbf{B}, x{:}\tau' \vdash^* M : \sigma_3$*: we proceed in a similar way;*
- $\mathbf{B} \vdash^* N : \tau \cup \tau'$*: by distributing the intersection over the union in* $\tau \cup \tau'$*, we can show that*

$$\tau \cup \tau' \sim (\tau_1 \cup \tau'_1) \cap (\tau_1 \cup \tau'_2) \cap \ldots \cap (\tau_1 \cup \tau'_{n'}) \cap (\tau_2 \cup \tau'_1) \cap \ldots \cap (\tau_n \cup \tau'_{n'})$$

We now have to show that:
· *for any* i, j, $\mathbf{B} \vdash^* N : \tau_i \cup \tau'_j$*, where* $\tau_i \not\equiv \sigma_1$*: obtained since* $\mathbf{B} \vdash^* N : \tau_i$*;*
· $\mathbf{B} \vdash^* N : \tau_i \cup \tau'_j$*, where* $\tau'_j \not\equiv \sigma_2$*: obtained since* $\mathbf{B} \vdash^* N : \tau'_j$*;*
· $\mathbf{B} \vdash^* N : \sigma_1 \cup \sigma_2$*: by the derivation tree* $\mathcal{D}''$*.*

□

Lemma A.4 *If* $\sigma \leqslant \tau$*, then we can derive* $x{:}\sigma \vdash^* x{:}\tau$ *without the* $(\leqslant)$ *rule.*

Proof. *By induction on the subtyping rules.* □

Lemma A.5 (Swapping)
In the large type assignment system $\vdash^*$*, the rules* $(\leqslant)$ *and* $(\cup E)$ *can be safely replaced by the rule* $(\cup E^*)$*.*

Proof.

- $(\leqslant)$ *rule. If* $\sigma \leqslant \tau$*, then by Lemma A.4, for any* $\mathbf{B}$*, we have that* $\mathbf{B}, x{:}\sigma \vdash^* x : \tau$*. Then, by Lemma A.2, if* $\mathbf{B} \vdash^* M : \sigma$*, then we have that* $\mathbf{B} \vdash^* M : \tau$*.*
- $(\cup E)$ *rule. By hypothesis, we have that* $\mathbf{B}, N{:}\sigma_1, x{:}\sigma_1 \vdash^* M : \sigma_3$ *and* $\mathbf{B}, N{:}\sigma_1, x{:}\sigma_2 \vdash^* M : \sigma_3$*, therefore, using the Substitution Lemma A.2 we have that* $\mathbf{B}, N{:}\sigma_1 \vdash^* M[N/x] : \sigma_3$ *and* $\mathbf{B}, N{:}\sigma_2 \vdash^* M[N/x] : \sigma_3$*. We conclude by using* $(\cup E^*)$*.*

□

Thanks to Lemma A.5, we can now consider that, in $\vdash^*$, rules $(\leqslant)$ and $(\cup E)$ are replaced by the rule $(\cup E^*)$.

Lemma A.6 *If* $\mathbf{B} \vdash^* M : \sigma$*, then* $G_{\mathbf{B}} \vdash_{\mathsf{NJ}(\beta)} r_\sigma[M]$*.*

Proof. *By induction on the derivation tree.* □

Lemma A.7 *If* $G_{\mathbf{B}} \vdash_{\mathsf{NJ}(\beta)} r_\sigma[M]$*, then* $\mathbf{B} \vdash^* M : \sigma$*.*

Proof. *Recall that* $\mathsf{NJ}(\beta)$ *is a particular case of systems called* $\mathbf{I(S)}$ *in [Pra71], which enjoys the property of being strongly normalizable. The normal form of a*

derivation, called "fully normal derivation" by Prawitz, is split into branches that contain a topmost "analytical part" consisting of elimination rules, an intermediate "minimum part" consisting of rules of the Post system and $(\bot)$, and a final "synthetical part" (ending with the very conclusion of the branch) only consisting of introduction rules. This implies the subformula property. Note that the major premise of the elimination rules contains a subformula of some assumption.

We proceed by induction on the fully-normal derivation of $G_{\mathbf{B}} \vdash r_\sigma[M]$. As $G_{\mathbf{B}}$ does not contain the symbol $\bot$ or $\exists$, the fully-normal derivation cannot contain the rules $(\bot)$ or $(\exists E)$. It also cannot contain the rule $(\exists I)$.

It is clear that the rules $(Hyp), (\wedge I), (\wedge E_i), (\vee I_i), (\vee E), (\beta\eta), (\top)$ of Figure 6 are respectively translated into the rules $(Var), (\cap I), (\cap E_i), (\cup I_i), (\cup E^*), (Eq_{\beta\eta}), (\omega)$ of $\vdash^*$.

From the Definition 3.2 of the Mints' realizers, we can verify that, in the assumptions, an implication is always enclosed by a $\forall$ quantifier, and reciprocally, a $\forall$ quantifier always encloses an implication.

Therefore, we can verify that if the $(\forall I)$ is used in the synthetical part, then the conclusion has the shape $G_{\mathbf{B}} \vdash \forall y.r_{\sigma_1}[y] \supset r_{\sigma_2}[M\,y]$. The rule above cannot be an elimination rule. Indeed, from the shape of the assumptions this elimination rule would be $(\forall E)$, which is absurd because the tree is fully normalized. Therefore the derivation tree ends necessarily by:

$$\dfrac{\dfrac{G_{\mathbf{B}}, r_{\sigma_1}[y] \vdash r_{\sigma_2}[M\,y]}{G_{\mathbf{B}} \vdash r_{\sigma_1}[y] \supset r_{\sigma_2}[M\,y]}\,(\supset I)}{G_{\mathbf{B}} \vdash \forall y.r_{\sigma_1}[y] \supset r_{\sigma_2}[M\,y]}\,(\forall I)$$

This part of the tree can be translated into:

$$\dfrac{\dfrac{\mathbf{B}, y{:}\sigma_1 \vdash^* My : \sigma_2}{\mathbf{B} \vdash^* \lambda y.My : \sigma_1 \to \sigma_2}\,(\to I)}{\mathbf{B} \vdash^* M : \sigma_1 \to \sigma_2}\,(Eq_{\beta\eta})$$

If the $(\supset E)$ rule is used in the derivation tree, then the major premise uses an elimination rule (because the tree is fully normalized), and this rule is necessarily $(\forall E)$ (because of the shape of the assumptions). For the same reason, we also know that the conclusion has necessarily the shape $G_{\mathbf{B}} \vdash_{\mathsf{NJ}(\beta)} r_{\sigma_2}[M\,N]$. Therefore the derivation tree ends by:

$$\dfrac{\dfrac{G_{\mathbf{B}} \vdash_{\mathsf{NJ}(\beta)} \forall y.r_{\sigma_1}[y] \supset r_{\sigma_2}[M\,y]}{G_{\mathbf{B}} \vdash_{\mathsf{NJ}(\beta)} r_{\sigma_1}[N] \supset r_{\sigma_2}[M\,N]}\,(\forall E) \qquad G_{\mathbf{B}} \vdash_{\mathsf{NJ}(\beta)} r_{\sigma_1}[N]}{G_{\mathbf{B}} \vdash_{\mathsf{NJ}(\beta)} r_{\sigma_2}[M\,N]}\,(\supset E)$$

This part of the tree can be translated into the $(\to E)$ rule. □

Finally, Theorem 3.5 is deducible from Lemmas A.6 and A.7.

A.2 Proofs on the subtyping algorithm

Definition A.8

(i) $L^{\geqslant}_{\omega}$ is the least set such that:
- $\omega \in L^{\geqslant}_{\omega}$;
- $\forall \sigma, \tau \in L^{\geqslant}_{\omega}$ we have that $\sigma \cap \tau, \tau \cap \sigma \in L^{\geqslant}_{\omega}$;
- $\forall \sigma \in L^{\geqslant}_{\omega}, \forall \tau$ we have that $\sigma \cup \tau, \tau \cup \sigma \in L^{\geqslant}_{\omega}$;
- $\forall \tau \in L^{\geqslant}_{\omega}, \forall \sigma$ we have that $\sigma \to \tau \in L^{\geqslant}_{\omega}$.

(ii) For any atomic type ϕ, $L^{\geqslant}_{\phi}$ is the least set such that:
- $\phi \in L^{\geqslant}_{\phi}$;
- $\forall \sigma, \tau \in L^{\geqslant}_{\phi}$ we have that $\sigma \cap \tau, \tau \cap \sigma \in L^{\geqslant}_{\phi}$;
- $\forall \sigma \in L^{\geqslant}_{\phi}, \forall \tau$ we have that $\sigma \cup \tau, \tau \cup \sigma \in L^{\geqslant}_{\phi}$;
- $\forall \sigma \in L^{\geqslant}_{\omega}$ we have that $\sigma \in L^{\geqslant}_{\phi}$.

(iii) For any arrow type $\sigma \to \tau$, $L^{\geqslant}_{\sigma\to\tau}$ is the least set such that:
- $\forall \sigma', \tau'$ such that $\sigma' \leqslant \sigma$ and $\tau \leqslant \tau'$, we have that $\sigma' \to \tau' \in L^{\geqslant}_{\sigma\to\tau}$;
- $\forall \sigma', \tau' \in L^{\geqslant}_{\sigma\to\tau}$ we have that $\sigma' \cap \tau', \sigma' \cap \tau' \in L^{\geqslant}_{\sigma\to\tau}$;
- $\forall \sigma' \in L^{\geqslant}_{\sigma\to\tau}, \forall \tau'$ we have that $\sigma' \cup \tau', \tau' \cup \sigma' \in L^{\geqslant}_{\sigma\to\tau}$;
- $\forall \sigma' \in L^{\geqslant}_{\omega}$ we have that $\sigma' \in L^{\geqslant}_{\sigma\to\tau}$.

(iv) For any intersection of types $\cap_i \sigma_i$ such that the σ_i are either atomic types, arrow types, or ω, $L^{\geqslant}_{\cap_i\sigma_i}$ is the least set such that:
- $\forall i, L^{\geqslant}_{\sigma_i} \subseteq L^{\geqslant}_{\cap_i\sigma_i}$
- $\forall \sigma', \tau' \in L^{\geqslant}_{\cap_i\sigma_i}$ we have that $\sigma' \cap \tau', \sigma' \cap \tau' \in L^{\geqslant}_{\cap_i\sigma_i}$;
- $\forall \sigma' \in L^{\geqslant}_{\cap_i\sigma_i}, \forall \tau'$ we have that $\sigma' \cup \tau', \tau' \cup \sigma' \in L^{\geqslant}_{\cap_i\sigma_i}$;
- $\forall \sigma \to \tau, \sigma \to \rho \in L^{\geqslant}_{\cap_i\sigma_i}$ we have that $L^{\geqslant}_{\sigma\to(\tau\cap\rho)} \subseteq L^{\geqslant}_{\cap_i\sigma_i}$
- $\forall \sigma \to \rho, \tau \to \rho \in L^{\geqslant}_{\cap_i\sigma_i}$ we have that $L^{\geqslant}_{(\sigma\cup\tau)\to\tau} \subseteq L^{\geqslant}_{\cap_i\sigma_i}$

(v) $L^{\leqslant}_{\omega}$ is the set of all types

(vi) For any atomic type ϕ, $L^{\leqslant}_{\phi}$ is the least set such that:
- $\phi \in L^{\leqslant}_{\phi}$
- $\forall \sigma \in L^{\leqslant}_{\phi}, \forall \tau$ we have that $\sigma \cap \tau, \tau \cap \sigma \in L^{\leqslant}_{\phi}$
- $\forall \sigma, \tau \in L^{\leqslant}_{\phi}$ we have that $\sigma \cup \tau, \tau \cup \sigma \in L^{\leqslant}_{\phi}$

(vii) For any σ, τ such that $\omega \leqslant \tau$ (*ie.* $\omega \leqslant \sigma \to \tau$), $L^{\leqslant}_{\sigma\to\tau}$ is the set of all types;

(viii) For any $\cap_i\sigma_i, \cup_j\tau_j, \cup_k\rho_k$ such that all the σ_i, τ_j, ρ_k are ANFs and $\omega \not\leqslant \cup_j\tau_j$, we define by mutual induction the sets $L^{\leqslant}_{\cap_i\sigma_i\to\cup_j\tau_j}$ and $L^{\leqslant}_{\cup_k\rho_k}$:
- $\forall k, L^{\leqslant}_{\rho_k} \subseteq L^{\leqslant}_{\cup_k\rho_k}$
- $\forall \sigma \in L^{\leqslant}_{\cup_k\rho_k}, \forall \tau$ we have that $\sigma \cap \tau, \tau \cap \sigma \in L^{\leqslant}_{\cup_k\rho_k}$
- $\forall \sigma, \tau \in L^{\leqslant}_{\cup_k\rho_k}$ we have that $\sigma \cup \tau, \tau \cup \sigma \in L^{\leqslant}_{\cup_k\rho_k}$
- $\forall \sigma \in L^{\geqslant}_{\cap_i\sigma_i}, \forall \tau \in L^{\leqslant}_{\cup_j\tau_j}$ we have that $\sigma \to \tau \in L^{\leqslant}_{\cap_i\sigma_i\to\cup_j\tau_j}$
- $\forall \sigma \in L^{\leqslant}_{\cap_i\sigma_i\to\cup_j\tau_j}, \forall \tau$ we have that $\sigma \cap \tau, \tau \cap \sigma \in L^{\leqslant}_{\cap_i\sigma_i\to\cup_j\tau_j}$
- $\forall \sigma, \tau \in L^{\leqslant}_{\cap_i\sigma_i\to\cup_j\tau_j}$ we have that $\sigma \cup \tau, \tau \cup \sigma \in L^{\leqslant}_{\cap_i\sigma_i\to\cup_j\tau_j}$

Lemma A.9
For any σ such that $L^{\geqslant}_{\sigma}$ is defined, we have:

(i) $\tau \in L^{\geqslant}_{\sigma} \iff \sigma \leqslant \tau$;

(ii) $\sigma \leqslant \tau \cup \rho \iff \sigma \leqslant \tau$ *or* $\sigma \leqslant \rho$.

Proof. *(ii) is deducible from (i) by looking at the construction rules of $L^{\geqslant}_{\sigma}$, so we only have to prove (i).*

- *By induction on the construction rules of $L^{\geqslant}_{\omega}$, for any $\sigma \in L^{\geqslant}_{\omega}$, we have that $\omega \leqslant \sigma$. For the same reasons, if $\sigma \in L^{\geqslant}_{\phi}$, we have that $\phi \leqslant \sigma$, if $\sigma' \in L^{\geqslant}_{\sigma\to\tau}$, we have that $\sigma \to \tau \leqslant \sigma'$, if $\sigma \in L^{\geqslant}_{\cup_i\sigma_i}$, we have that $\cup_i\sigma_i \leqslant \sigma$;*
- *We prove by induction on the subtyping rules of the (sub)type theory Ξ, that for any σ,τ such that $\sigma \leqslant \tau$ and $\sigma \in L^{\geqslant}_{\omega}$, we have that $\tau \in L^{\geqslant}_{\omega}$. Some cases of the inductive proof need a case analysis on the construction rules of $L^{\geqslant}_{\omega}$. Therefore, if $\omega \leqslant \tau$, then $\tau \in L^{\geqslant}_{\omega}$;*
- *For any σ,τ such that $\sigma \leqslant \tau$ and $\sigma \in L^{\geqslant}_{\phi}$, we have two cases:*
  · *If $\omega \leqslant \sigma$, then, as $L^{\geqslant}_{\omega} \subseteq L^{\geqslant}_{\phi}$, we now that $\tau \in L^{\geqslant}_{\omega}$, so it is clear that $\tau \in L^{\geqslant}_{\phi}$;*
  · *If $\omega \not\leqslant \sigma$, we prove by induction on the subtyping rules of the (sub)type theory Ξ that $\tau \in L^{\geqslant}_{\omega}$.*
  Therefore, if $\phi \leqslant \tau$, then $\tau \in L^{\geqslant}_{\phi}$;
- *For any σ',τ' such that $\sigma' \leqslant \tau'$ and $\sigma' \in L^{\geqslant}_{\sigma\to\tau}$, we have two cases:*
  · *If $\omega \leqslant \sigma'$, then it is clear that $\tau' \in L^{\geqslant}_{\sigma\to\tau}$;*
  · *If $\omega \not\leqslant \sigma'$, we prove by induction on the subtyping rules of the (sub)type theory Ξ that $\tau' \in L^{\geqslant}_{\sigma\to\tau}$. Notice that, for rules (12) and (13), we need to use Lemma 4.8. For instance, in rule (12), we suppose $\sigma' \equiv (\sigma_1 \to \tau_1)\cap(\sigma_1 \to \tau_2) \in L^{\geqslant}_{\sigma\to\tau}$. From the construction rules of $L^{\geqslant}_{\sigma\to\tau}$, we know that both $\sigma_1 \to \tau_1 \in L^{\geqslant}_{\sigma\to\tau}$ and $\sigma_1 \to \tau_2 \in L^{\geqslant}_{\sigma\to\tau}$. These types cannot be both in $L^{\geqslant}_{\omega}$, because it would be contrary to the supposition $\omega \not\leqslant \sigma'$. From the construction rules of $L^{\geqslant}_{\sigma\to\tau}$, we know that both $\tau \leqslant \tau_1$ and $\tau \leqslant \tau_2$, so by Lemma 4.8 we get $\tau \leqslant \tau_1 \cap \tau_2$. Moreover, $\sigma_1 \leqslant \sigma$. Therefore $\sigma_1 \to (\tau_1 \cap \tau_2) \in L^{\geqslant}_{\sigma\to\tau}$.*
  As $\sigma \to \tau \in L^{\geqslant}_{\sigma\to\tau}$, we conclude that, if $\sigma \to \tau \leqslant \tau'$, then $\tau' \in L^{\geqslant}_{\sigma\to\tau}$.
- *We prove by induction on the subtyping rules of the (sub)type theory Ξ that for any σ,τ such that $\sigma \leqslant \tau$ and $\sigma \in L^{\geqslant}_{\cap_i\sigma_i}$, we have that $\tau \in L^{\geqslant}_{\cap_i\sigma_i}$. Moreover, $\cap_i\sigma_i \in L^{\geqslant}_{\cap_i\sigma_i}$. Therefore, if $\cap_i\sigma_i \leqslant \tau$, then $\tau \in L^{\geqslant}_{\cap_i\sigma_i}$.*

□

Corollary A.10 *If $\sigma \to \tau \leqslant \sigma' \to \tau'$ and $\omega \not\leqslant \tau'$, then $\sigma' \leqslant \sigma$ and $\tau \leqslant \tau'$.*

Proof. *According to the construction rules of $L^{\geqslant}_{\omega}$, $\sigma' \to \tau' \notin L^{\geqslant}_{\omega}$. Therefore, by looking at the definition of $L^{\geqslant}_{\sigma\to\tau}$, we can say that $\sigma' \leqslant \sigma$ and $\tau \leqslant \tau'$.* □

Lemma A.11 *If $\sigma \sim \omega$, then $\mathcal{R}_1$ rewrites σ as ω.*

Proof. *Since $\sigma \sim \omega$, we have that $\sigma \in L^{\geqslant}_{\omega}$. The proof proceeds by induction on σ.*□

Lemma A.12
For any τ such that $L^{\leqslant}_{\tau}$ is defined, we have:

(i) *$\sigma \in L^{\leqslant}_{\tau} \iff \sigma \leqslant \tau$;*

(ii) *$\sigma \cap \rho \leqslant \tau \iff \sigma \leqslant \tau$ or $\rho \leqslant \tau$.*

Proof. (ii) is deducible from (i) by looking at the construction rules of $L^{\leqslant}_{\sigma}$, so we only have to prove (i).

- By induction on the construction rules of $L^{\leqslant}_{\phi}$, if $\sigma \in L^{\leqslant}_{\phi}$, we have that $\sigma \leqslant \phi$. For the same reasons, if $\sigma \in L^{\leqslant}_{\cap_i\sigma_i \to \cup_j\tau_j}$, we have that $\sigma \leqslant \cap_i\sigma_i \to \cup_j\tau_j$, and if $\sigma \in L^{\leqslant}_{\cup_i\sigma_i}$, we have that $\sigma \leqslant \cup_i\sigma_i$;
- We prove by induction on the subtyping rules of the (sub)type theory Ξ that for any σ, τ such that $\tau \leqslant \sigma$ and $\tau \in L^{\leqslant}_{\phi}$, we have that $\sigma \in L^{\leqslant}_{\phi}$. Therefore, if $\sigma \leqslant \phi$, then $\sigma \in L^{\leqslant}_{\phi}$;
- For $L^{\leqslant}_{\omega}$ and $L^{\leqslant}_{\sigma\to\tau}$ where $\omega \leqslant \tau$, the proof is trivial;
- We notice that $\cup_k\rho_k \in L^{\leqslant}_{\cup_k\rho_k}$ and $\cap_i\sigma_i \to \cup_j\tau_j \in L^{\leqslant}_{\cap_i\sigma_i\to\cup_j\tau_j}$;
- We note P_1 the proposition "for any union of ANFs $\cup_k\rho_k$, for any σ, τ, such that $\sigma \leqslant \tau$ and $\tau \in L^{\leqslant}_{\cup_k\rho_k}$ we have that $\sigma \in L^{\leqslant}_{\cup_k\rho_k}$", and P_2 the proposition "for any ANF $\cap_i s_i \to \cup_j\tau_j$, for any σ, τ, such that $\sigma \leqslant \tau$ and $\tau \in L^{\leqslant}_{\cap_i s_i\to\cup_j\tau_j}$ we have that $\sigma \in L^{\leqslant}_{\cap_i s_i\to\cup_j\tau_j}$". We prove by induction on the subtyping rules of the (sub)type theory Ξ that:
  · $P_1 \Longrightarrow P_2$
  · $P_2 \Longrightarrow P_1$
- We prove by mutual induction on the definitions of $L^{\leqslant}_{\cap_i s_i\to\cup_j\tau_j}$ and $L^{\leqslant}_{\cup_k\tau_k}$ that P_1 and P_2 hold. The base case is P_1': "for any union of ANFs (*excluding arrow-types*) $\cup_k\rho_k$, for any σ, τ, such that $\sigma \leqslant \tau$ and $\tau \in L^{\leqslant}_{\cup_k\rho_k}$ we have that $\sigma \in L^{\leqslant}_{\cup_k\rho_k}$". P_1' is provable by induction on the subtyping rules of the (sub)type theory Ξ.

□

Corollary A.13 *If all the σ_i and τ_i are ANFs, then $\cap_i\sigma_i \leqslant \cup_j\tau_j \iff \exists i, j, \sigma_i \leqslant \tau_j$.*

Proof. If $\exists i, j, \sigma_i \leqslant \tau_j$, it is clear that $\cap_{1\leqslant i\leqslant m}\sigma_i \leqslant \cup_{1\leqslant j\leqslant n}\tau_j$. Reciprocally, we proceed by strong recurrence on $m + n$.

- If $m = 1$ and $n = 1$, then it is clear that $\sigma_1 \leqslant \tau_1$;
- If $m = 1$ and $n > 1$, then, by Lemma A.12, either $\cap_{1\leqslant i\leqslant\lfloor\frac{m}{2}\rfloor}\sigma_i \leqslant \tau_1$, or $\cap_{\lfloor\frac{m}{2}\rfloor+1\leqslant i\leqslant m}\sigma_i \leqslant \tau_1$;
- If $m > 1$ and $n > 1$, then, by Lemma A.9, either $\cap_i\sigma_i \leqslant \cap_{1\leqslant j\leqslant\lfloor\frac{n}{2}\rfloor}\tau_j$, or $\cap_i\sigma_i \leqslant \cap_{\lfloor\frac{n}{2}\rfloor+1\leqslant j\leqslant n}\tau_j$.

□

Lemma A.14

- $\mathcal{R}_2 \circ \mathcal{R}_1 = \mathcal{R}_1 \circ \mathcal{R}_2$
- $\mathcal{R}_3 \circ \mathcal{R}_1 = \mathcal{R}_1 \circ \mathcal{R}_3$
- $\mathcal{R}_4 \circ \mathcal{R}_1 = \mathcal{R}_1 \circ \mathcal{R}_4 \circ \mathcal{R}_1$

Proof. *Merging $\mathcal{R}_1$ and $\mathcal{R}_2$ (resp. $\mathcal{R}_3$) is strongly normalizing, and all the critical pairs are locally confluent, so according to Newman's lemma, we get a confluent abstract rewriting system. Therefore, $\mathcal{R}_2 \circ \mathcal{R}_1 = \mathcal{R}_1 \circ \mathcal{R}_2$ (resp. $\mathcal{R}_3 \circ \mathcal{R}_1 = \mathcal{R}_1 \circ \mathcal{R}_3$).*

For any rewriting rule $\mathcal{R}$ in $\mathcal{R}_4$, we have that $\mathcal{R} \circ \mathcal{R}_1 = \mathcal{R}_1 \circ \mathcal{R} \circ \mathcal{R}_1$:

- *$\mathcal{R}_1(\sigma \to \tau)$ is $\mathcal{R}_1(\sigma) \to \mathcal{R}_1(\tau)$, which then rewrites, by the first rule of $\mathcal{R}_4$, to $\mathcal{R}_3 \circ \mathcal{R}_1(\sigma) \to \mathcal{R}_2 \circ \mathcal{R}_1(\tau)$, which is equivalent to $\mathcal{R}_1(\mathcal{R}_3 \circ \mathcal{R}_1(\sigma) \to \mathcal{R}_2 \circ \mathcal{R}_1(\tau))$;*
- *Let $(\cup_j\sigma'_j \to \cap_k\tau'_k)$ be $\mathcal{R}_1(\cup_i\sigma_i \to \cap_h\tau_h)$, which then rewrites, by the second rule*

of $\mathcal{R}_4$, to $\cap_j(\cap_k(\sigma'_j \to \tau'_k))$. All the σ'_j and τ'_k are already normalized by $\mathcal{R}_1$, so the whole expression is already normalized by $\mathcal{R}_1$.

By composing these rewriting rules, we see that $\mathcal{R}_4 \circ \mathcal{R}_1 = \mathcal{R}_1 \circ \mathcal{R}_4 \circ \mathcal{R}_1$. □

Corollary A.15

- $\mathcal{R}_2 \circ \mathcal{R}_4 \circ \mathcal{R}_1 = \mathcal{R}_1 \circ \mathcal{R}_2 \circ \mathcal{R}_4 \circ \mathcal{R}_1$
- $\mathcal{R}_3 \circ \mathcal{R}_4 \circ \mathcal{R}_1 = \mathcal{R}_1 \circ \mathcal{R}_3 \circ \mathcal{R}_4 \circ \mathcal{R}_1$

Proof. *Immediate from Lemma A.14.* □

Theorem A.16 ($\mathcal{A}_1, \mathcal{A}_2$'s Completeness)

(i) *For any type σ', τ' such that $\sigma' \leqslant \tau'$, let $\cup_i(\cap_j \sigma_{i,j}) \equiv \mathcal{R}_3 \circ \mathcal{R}_4 \circ \mathcal{R}_1(\sigma')$ and $\cap_h(\cup_k \tau_{h,k}) \equiv \mathcal{R}_2 \circ \mathcal{R}_4 \circ \mathcal{R}_1(\tau')$. We have that $\mathcal{A}_1(\cup_i(\cap_j \sigma_{i,j}) \leqslant \cap_h(\cup_k \tau_{h,k}))$.*

(ii) *Let σ and $\tau \not\sim \omega$ be ANFs. If $\sigma \leqslant \tau$ then $\mathcal{A}_2(\mathcal{R}_1(\sigma) \leqslant \mathcal{R}_1(\tau))$.*

Proof. *We know by Lemma 4.4 that, as $\sigma' \leqslant \tau'$, $\cup_i(\cap_j \sigma_{i,j}) \leqslant \cup_j \cap_h (\cup_k \tau_{h,k})$. The proof proceeds by mutual induction.*

(i) *For (i), if any of the $\tau_{h,k}$ is equivalent to ω, then $\cup_k \tau_{h,k}$ is equivalent to ω, then by Lemma A.11, $\cup_k \tau_{h,k}$ has been rewritten into ω. As we suppose $\tau_{h,k}$ has not been erased by $\mathcal{R}_1$, it means that $\cap_h(\cup_k \tau_{h,k}) \equiv \omega$. The algorithm is accepting, so we can conclude.*

On the other hand, we know by Corollary A.15 that $\mathcal{R}_1(\sigma_{i,j}) = \sigma_{i,j}$ and $\mathcal{R}_1(\tau_{h,k}) = \tau_{h,k}$, so if none of the $\tau_{h,k}$ are equivalent to ω, we can safely call $\mathcal{A}_2(\sigma_{i,j}, \tau_{h,k})$. Moreover, we know by Lemmas 4.8 and A.13, that for all i and h, there exists some j and some k, such that $\sigma_{i,j} \leqslant \tau_{h,k}$. As the induction hypothesis states that $\mathcal{A}_2$ can decide subtyping for any of the possible $\sigma_{i,j} \leqslant \tau_{h,k}$, we can conclude;

(ii) *For (ii), we proceed by case analysis on the algorithm $\mathcal{A}_2$, and by looking at the subtyping rules.*
- *Case $\omega \leqslant \tau$: by hypothesis, $\omega \not\leqslant \tau$, so this case is impossible;*
- *Case $\phi \leqslant \phi'$: according to Lemma A.9, the only atomic variable ϕ'' such that $\phi \leqslant \phi''$ is ϕ itself;*
- *Case $\sigma \to \tau \leqslant \phi$: impossible by inspecting $L^{\leqslant}_{\phi}$;*
- *Case $\phi \leqslant \sigma \to \tau$: by inspecting $L^{\geqslant}_{\phi}$, we have $\phi \leqslant \sigma \to \tau$ iff $\sigma \to \tau \sim \omega$. However, this is contrary to the hypothesis $\sigma \to \tau \not\sim \omega$. This case is therefore impossible;*
- *Case $\sigma \to \tau \leqslant \sigma' \to \tau'$: we know that $\tau' \not\sim \omega$, therefore, by Corollary A.10, $\tau \leqslant \tau'$, and $\sigma' \leqslant \sigma$. We know that $\sigma, \sigma', \tau, \tau'$ have already been rewritten by $\mathcal{R}_1$, and that they already are in CANF or DANF, therefore it is no need to preprocess them through $\mathcal{R}_2 \circ \mathcal{R}_4 \circ \mathcal{R}_1$, or $\mathcal{R}_3 \circ \mathcal{R}_4 \circ \mathcal{R}_1$. By induction hypothesis, $\mathcal{A}_1$ decides that $\sigma' \leqslant \sigma$ and $\tau \leqslant \tau'$, so we can conclude.*

□