Subject Reduction vs Intersection / Union Types in $\overline{\lambda}\mu\tilde{\mu}$

Extended abstract

(Visions of Computer Science, BCS International Academic Conference, London, pp. 249-258, 2008)

Steffen van Bakel

Department of Computing, Imperial College London, 180 Queen's Gate, London SW7 2BZ, UK
svb@doc.ic.ac.uk

Abstract

This paper studies intersection and union type assignment for the calculus $\overline{\lambda}\mu\tilde{\mu}$ [6], a proof-term syntax for Gentzen's classical sequent calculus. We show that this notion is closed for subject-expansion, and show that it needs to be restricted to satisfy subject-reduction as well, even when limiting reduction to (confluent) call-by-name or call-by-value reduction, making it unsuitable to define a semantics.

Introduction

The role of Classical Logic in computer science is changing drastically over the last few years. Given the direct relation between the Lambda Calculus [4] and intuitionistic logic, for many years it was believed that only the *constructive* logics had any real computational content, and only after Griffin's discovery of the relation between double-negation elimination [14] and Felleisen's control operators [12] did the research community become aware of the computational advantages of Classical Logic. Since then, many researchers have focussed on different calculi, trying to exploit the Curry-Howard isomorphism for various classical logics, both in sequent style and in natural deduction, for computer science: amongst many, we mention [22, 21, 6, 24, 25]. In this paper we contribute to that line of research by studying the $\overline{\lambda}\mu\tilde{\mu}$-calculus [6], which enjoys a Curry-Howard isomorphism for implicative G_3 [18], an variant of Gentzen's system LK [13].

One of the great advantages of using classical logic is the capture of not only parameter passing, but also context management: for programs based on full classical logic, a parameter call is truly dual to a context call. This gives rise to far richer programming paradigms, which need to be fully investigated and exploited. This paper tries to prepare the ground for semantics, by defining a notion of intersection-union typing for $\overline{\lambda}\mu\tilde{\mu}$, and to study its properties.

One of the main contributions of intersection types [5] for the λ-calculus [4] is the characterisation of strong normalisation, which states that, in a system without the type constant $\top$[1] the typeable terms are exactly the strongly normalisable ones [1]. This result has since then been achieved in many ways for different calculi, and in order to come to a similar characterisation for the (untyped) sequent calculus $\overline{\lambda}\mu\tilde{\mu}$ [6], [8] presented a notion of intersection and union typing for that calculus. Another main contribution is that of the construction of *filter semantics*, i.e. to build a λ-model using assignable types. With this goal in mind, in this paper we revisit the system of [8], adding $\top$ as the maximal and $\bot$ as the minimal type, and

[1] Normally, ω is used; here we reserve Greek characters for context variables.

the derivation rules $(\top)$, $(\bot)$, that are added for the purpose of *subject expansion* (also called *completeness*).

The notion of typing (i.e. context assignment) we present here will be shown to be the natural system, in that intersection, $\top$, union, and $\bot$ play their expected roles for subject expansion, our first step towards the construction of a filter model. However, we show that subject reduction no longer holds, and will argue that this is caused by the non-logical foundation (i.e. typeable terms no longer correspond to proofs [17]) of both intersection and union. We will show that we can partially recover from these failures by restricting to either call-by-name or call-by-value reduction, but that we also need to restrict the applicability of either union or intersection. This then, unfortunately, demolishes the subject-expansion result, making the system not suitable to define semantics.

In this paper we will show that the –at the time– surprising loss of soundness for the system with intersection and union types in [3] is, in fact, natural and inevitable. Also, working with intersection and union in the context of these highly symmetric sequent calculi makes clear that these are truly dual; we will show that it is not union alone that causes problems, but they also arise when dealing with intersection. Although the idea behind both intersection and union might be (the logical) *and* and *or*, the fact that they are both *not* logical destroys the soundness, both for a system based on intersection, as for a system based on union. This also explains why, for ML with side-effects [15, 26, 20], quantification is no longer sound: also the $(\forall I)$ and $(\forall E)$ rules of ML are not logical. This problem also appears when using intersection and union types in an operational setting [7, 11].

The system presented here is also a natural extension of the system considered for $\overline{\lambda}\mu\tilde{\mu}$ in [6], i.e. the basic implicative system for Classical Logic, but extended with intersection and union types and the type constants $\top$ and $\bot$. That system is based on the sequent calculus, introduced by Gentzen in [13], is a logical system in which the rules only introduce connectives (but on both sides of a sequent), in contrast to natural deduction which uses introduction and elimination rules. The only way to eliminate a connective is to eliminate the whole formula in which it appears, via an application of the (cut)-rule.

In this paper we will treat $\overline{\lambda}\mu\tilde{\mu}$ as a pure, untyped calculus, and define a notion of sequent-style intersection-union typing for $\overline{\lambda}\mu\tilde{\mu}$. The system is set up to be a conservative extension of Krivine's *Système D* intersection type assignment system for the λ-calculus [19] ($\vdash_D$), in that lambda terms typeable in that system translate to $\overline{\lambda}\mu\tilde{\mu}$-terms, while preserving the type. There are many different notions of intersection type assignment in existence (see also [5, 1, 2]), that, in the context of the λ-calculus more or less coincide; the most important difference is normally the language of types (full BCD [5, 19], or strict types [1, 2]), and the availability of a contra-variant $\leq$-relation [5, 2] or not [19, 1]. Surprisingly, this is no longer true when bringing intersection types (and union) to the context of sequent calculi; BCD-types are needed, as will be shown in this paper. The main results of this paper are that this notion is closed for expansion and that it needs to be restricted to become closed for reduction; characterisation of strong normalisation is still an open issue.

We will show in this paper that there are two causes for the loss of subject reduction:

(a)) Performing a μ-reduction towards an introduced intersection (i.e. via rule $(\cap R)$), and

(b)) Performing a $\tilde{\mu}$-reduction towards an introduced union (via rule $(\cup L)$).

Any solution for subject reduction for a system with intersection and union types for $\overline{\lambda}\mu\tilde{\mu}$ will need to solve *both* problems.

To address the problem of subject-reduction, in this paper we will revisit the system of [8], and discuss the problem of subject reduction and expansion in the context of that system in detail. We will then define a notion of typing that is an extension of [8] by adding $\top$ and $\bot$,

and show that subject reduction is problematic. After that we will focus on two sub-sytems, $\vdash_{\mathbf{V}}$ and $\vdash_{\mathbf{N}}$, that will successfully tackle that problem; for CBV we limit $(\cap R)$ to values, and for CBN limit $(\cup L)$ to slots. These solutions are crucially different from any other published in the past, in that we do not limit the syntax of types at all (as in [10, 9]), and pose completely different side-conditions on rules (wrt [16]), in fact generalising the there claimed result. As far as we know, these are the first correct presentations of sound restrictions of a fully expressive system with intersection and union types for $\overline{\lambda}\mu\tilde{\mu}$.

As intersection is not problematic for CBN-reduction, we can still embed the $\vdash_{D}$-system for the λ-calculus whilst preserving the types, so our system $\vdash_{\mathbf{N}}$ is as least as expressive; also, union types are still useful for μ-reductions. On the other hand, union types are not problematic for CBV-reduction, and intersection can then be used for $\tilde{\mu}$-reductions. However, limiting the applicability of either rule $(\cap R)$ or $(\cup L)$ disrupts the proof of subject-expansion: this implies that it is impossible to define a notion of typing that is closed for *conversion*, even when restricting to CBV of CBN. So we cannot construct a filter model using types for $\overline{\lambda}\mu\tilde{\mu}$ using this system.

1 The calculus $\overline{\lambda}\mu\tilde{\mu}$

In its typed version, $\overline{\lambda}\mu\tilde{\mu}$ is a proof-term syntax for a classical sequent calculus. As in $\lambda\mu$ [22], for the $\overline{\lambda}\mu\tilde{\mu}$-calculus there are two sets of variables: x, y, z, etc., label the types of the hypotheses and α, β, γ, etc., label the types of the conclusions. Moreover, the syntax of $\overline{\lambda}\mu\tilde{\mu}$ has three different categories: commands, terms, and contexts (or co-terms). Correspondingly, they are typed by three kinds of sequents: the usual sequents $\Gamma \vdash \Delta$ type commands, while the sequents typing terms (resp. contexts) are of the form $\Gamma \vdash A \mid \Delta$ (resp. $\Gamma \mid A \vdash \Delta$), marking the conclusion (resp. hypothesis) A as *active*.

Definition 1.1 (COMMANDS, TERMS, AND CONTEXTS [6])

$$\underset{\textit{commands}}{c ::= \langle v \mid e \rangle} \qquad \underset{\textit{terms}}{v ::= x \mid \lambda x.v \mid \mu\beta.c} \qquad \underset{\textit{contexts}}{e ::= \alpha \mid v \cdot e \mid \tilde{\mu}x.c}$$

Here $\mu\alpha.c$, $\tilde{\mu}x.c'$ and $\lambda y.v$ respectively bind α in c, x in c' and y in v; as usual, we will consider terms, contexts and commands up to α-conversion, and consider substitution to be capture avoiding.

Typing for $\overline{\lambda}\mu\tilde{\mu}$ is defined by:

Definition 1.2 (TYPING FOR $\overline{\lambda}\mu\tilde{\mu}$ [6])

$$(cut): \frac{\Gamma \vdash_{\mathcal{M}} v : A \mid \Delta \quad \Gamma \mid e : A \vdash_{\mathcal{M}} \Delta}{\langle v \mid e \rangle : \Gamma \vdash_{\mathcal{M}} \Delta}$$

$$(Ax\text{-}R): \frac{}{\Gamma, x{:}A \vdash_{\mathcal{M}} x : A \mid \Delta} \qquad (Ax\text{-}L): \frac{}{\Gamma \mid \alpha : A \vdash_{\mathcal{M}} \alpha{:}A, \Delta}$$

$$(RI): \frac{\Gamma, x{:}A \vdash_{\mathcal{M}} v : B \mid \Delta}{\Gamma \vdash_{\mathcal{M}} \lambda x.v : A \to B \mid \Delta} \qquad (LI): \frac{\Gamma \vdash_{\mathcal{M}} v : A \mid \Delta \quad \Gamma \mid e : B \vdash_{\mathcal{M}} \Delta}{\Gamma \mid v \cdot e : A \to B \vdash_{\mathcal{M}} \Delta}$$

$$(\mu): \frac{c : \Gamma \vdash_{\mathcal{M}} \alpha{:}A, \Delta}{\Gamma \vdash_{\mathcal{M}} \mu\alpha.c : A \mid \Delta} \qquad (\tilde{\mu}): \frac{c : \Gamma, x{:}A \vdash_{\mathcal{M}} \Delta}{\Gamma \mid \tilde{\mu}x.c : A \vdash_{\mathcal{M}} \Delta}$$

Commands can be computed (thus eliminating the cut in the corresponding proof): We have a critical pair in the command $\langle \mu\alpha.c_1 \,|\, \tilde{\mu}x.c_2\rangle$; since *cut*-elimination of the classical sequent calculus is not confluent, neither is reduction in $\overline{\lambda}\mu\tilde{\mu}$.

Notice that $\overline{\lambda}\mu\tilde{\mu}$ has both *explicit* and *implicit* variables: the implicit variables are for example in $v{\cdot}e$, where the hole ($\cdot$, which acts as input) does not have an identity, and in $\lambda x.v$ where the context (output) is anonymous. We can make these variables explicit by, respectively, $\tilde{\mu}y\langle y \,|\, v{\cdot}e\rangle$ and $\mu\alpha.\langle \lambda x.v \,|\, \alpha\rangle$; in case the variable y (α) does not occur in $v{\cdot}e$ ($\lambda x.v$), these terms are η redexes, but, in general, the implicit variable can be made to correspond to one that already occurs. The type assignment system below (Definition 2.3) is designed to be indifferent to these steps.

Herbelin's $\overline{\lambda}\mu\tilde{\mu}$-calculus expresses the duality of LK's left and right introduction in a very symmetrical syntax. But the duality goes beyond that: for instance, the symmetry of the reduction rules display syntactically the duality between the call-by-value (CBV) and call-by-name (CBN) evaluations (see also [25]). Indeed, the CBV reduction $\to_{\mathrm{v}}$ is obtained by forbidding a $\tilde{\mu}$-reduction when the redex is also a μ-redex, whereas the CBN reduction $\to_{\mathrm{N}}$ forbids a μ-reduction when the redex is also a $\tilde{\mu}$-redex.

Definition 1.4 (CBV AND CBN [6, 16]) *Values* V are defined by $V ::= x \mid \lambda x.v$, and *slots*[2] E are defined by $E ::= \alpha \mid v \cdot e$.

- CBV-reduction is defined by replacing rule $(\tilde{\mu})$ by: $(\tilde{\mu}_{\mathrm{v}}) : \langle V \,|\, \tilde{\mu}x.c\rangle \;\to\; c[V/x]$;
- CBN-reduction is defined by replacing rule (μ) by: $(\mu_{\mathrm{N}}) : \langle \mu\beta.c \,|\, E\rangle \;\to\; c[E/\beta]$.

The remainder of this paper will be dedicated to the definition of a notion of intersection typing on $\overline{\lambda}\mu\tilde{\mu}$. This will be defined as a natural extension of a system with intersection types for the λ-calculus; we will start by briefly summarising the latter. We assume the reader to be familiar with the λ-calculus [4]. As done in [6], we define an encoding of the λ-calculus into $\overline{\lambda}\mu\tilde{\mu}$:

Definition 1.5 $\llbracket x \rrbracket_\lambda \;\triangleq\; x \qquad \llbracket (\lambda x.M) \rrbracket_\lambda \;\triangleq\; \lambda x.\llbracket M \rrbracket_\lambda \qquad \llbracket (MN) \rrbracket_\lambda \;\triangleq\; \mu\alpha.\langle \llbracket M \rrbracket_\lambda \,|\, \llbracket N \rrbracket_\lambda \cdot \alpha\rangle$

Correctness of this encoding is easy to prove; $\eta\mu$-reduction is needed here:

Lemma 1.6 $\llbracket (\lambda x.M)N \rrbracket_\lambda \;\to\; \llbracket M \rrbracket_\lambda [\llbracket N \rrbracket_\lambda / x]$.

Using this lemma, we can prove the following relation between the λ-calculus and $\overline{\lambda}\mu\tilde{\mu}$:

Theorem 1.7 *i) If* $M \to_\beta N$, *then* $\llbracket M \rrbracket_\lambda \to \llbracket N \rrbracket_\lambda$.
ii) If $M \to_{\mathrm{v}} N$, *then* $\llbracket M \rrbracket_\lambda \to_{\mathrm{v}} \llbracket N \rrbracket_\lambda$.

The notion of intersection type assignment for $\overline{\lambda}\mu\tilde{\mu}$ as defined in the next section is a conservative extension of Système $\mathcal{D}$ [19] – which is a restricted version of the BCD-system presented in [5], not closed for η-reduction – in that we can translate lambda terms typeable in that system to $\overline{\lambda}\mu\tilde{\mu}$ terms whilst preserving types. The BCD-system treats the two type constructors '$\to$' and '$\cap$' the same, allowing, in particular, intersection to occur at the right of arrow types; this general treatment is not necessary within the context of the λ-calculus (see [2]), but for $\overline{\lambda}\mu\tilde{\mu}$, as we will see in Example 4.1, to type all normal forms it is natural to have an intersection

[2] In [16], slots are called *linear evaluation contexts*.

type occur on the right-hand side of an arrow type. We will not consider a relation on types that is contra-variant on the arrow, since we are not interested in modelling extensionality.

Definition 1.8 *i*) Let $\mathcal{V}$ be a countable (infinite) set of type-variables, ranged over by φ. The set $\mathcal{T}$ of *intersection types*, ranged over by $A, B \ldots$, is defined through:

$$A, B \ ::= \ \varphi \mid \top \mid (A \to B) \mid (A \cap B)$$

ii) A *statement* is an expression of the form $M : A$, with $M \in \Lambda$, and $A \in \mathcal{T}$. M is the *subject* and A the *predicate* of $M : A$.

iii) A *context* Γ is a partial mapping from term variables to intersection types, and we write $x{:}A \in \Gamma$ if $\Gamma\, x = A$. We will write $x \notin \Gamma$ if Γ is not defined on x, and $\Gamma \backslash x$ when we remove x from the domain of Γ.

We will consider a pre-order on types which takes into account the idem-potence, commutativity and associativity of the intersection type constructor, and defines $\top$ to be the maximal element. We write $\underline{n}$ for the set $\{1, \ldots, n\}$. It is easy to show that the type constructor $\cap$ is associative and commutative, and we will write $\cap_n \mathsf{A}_i$ for $A_1 \cap \cdots \cap A_n$, and consider $\top$ to be the empty intersection: $\top = \cap_0 \mathsf{A}_i$.

Definition 1.9 *Intersection type assignment* and *derivations* are defined by the following (sequent style) natural deduction system.

$$(Ax) : \frac{}{\Gamma, x{:}A \vdash_D x : A} \qquad (\to I) : \frac{\Gamma, x{:}A \vdash_D M : B}{\Gamma \vdash_D \lambda x.M : A \to B} \qquad (\cap E) : \frac{\Gamma \vdash_D M : \cap_n \mathsf{A}_i}{\Gamma \vdash_D M : A_j} \ (j \in \underline{n}, n \geq 1)$$

$$(\to E) : \frac{\Gamma \vdash_D M : A \to B \quad \Gamma \vdash_D N : A}{\Gamma \vdash_D MN : B} \qquad (\cap I) : \frac{\Gamma \vdash_D M : A_j \quad (\forall j \in \underline{n})}{\Gamma \vdash_D M : \cap_n \mathsf{A}_i} \ (n \geq 0)$$

2 The system $\mathcal{M}^{\cap\cup}$ of intersection and union typing for $\overline{\lambda}\mu\tilde{\mu}$

The first notion of intersection and union typing for $\overline{\lambda}\mu\tilde{\mu}$ was presented in [8]. In this paper, we will revisit this system, albeit in a presentation that is adapted to our approach, and discuss its results and shortcomings.

We start with the definition of intersection/union types.

Definition 2.1 (Intersection and Union Types, Contexts) *i*) The set $\mathcal{T}_{\cap\cup}$ of *intersection-union types* (we will write $\mathcal{T}$ for short), ranged over by $A, B, \ldots$ is inductively defined by:

$$A, B \ ::= \ \varphi \mid (A \to B) \mid (A \cap B) \mid (A \cup B)$$

ii) A *context* Γ of term-variables (Δ of context-variables) is a partial mapping from term-variables (context-variables) to types, represented as a set of statements with only distinct variables as subjects.

We will omit unnecessary brackets in types; the type constructors '$\cap$' and '$\cup$' will bind more strongly than '$\to$', so $A \cap B \to C \cap D$ is used for $((A \cap B) \to (C \cap D))$, $A \to B \cap C \to D$ for $(A \to ((B \cap C) \to D))$, and $(A \to B) \cap C \to D$ for $(((A \to B) \cap C) \to D)$. We will sometimes write the omitable brackets to enhance readability.

We will consider a pre-order on types which takes into account the idem-potence, commutativity and associativity of the intersection and union type constructors.

Definition 2.2 (Relations on types) *i*) The relation $\leq$ is defined as the least pre-order on $\mathcal{T}$ such that:

$$\begin{array}{llll} A \leq \top & A\cap B \leq A,B & A\leq C \,\&\, A\leq B \;\Rightarrow\; A\leq B\cap C \\ \bot \leq A & A,B \leq A\cup B & A\leq C \,\&\, B\leq C \;\Rightarrow\; A\cup B\leq C \end{array}$$

We will write $A \leq_\cap B$ for the relation on types defined by the first line, and $A \leq_\cup B$ for the one defined by the second.

ii) The equivalence relation $\sim$ on types is defined by:

$$\begin{array}{c} A \leq B \leq A \;\Rightarrow\; A \sim B \\ A \sim C \;\&\; B \sim D \;\Rightarrow\; A\to B \sim C\to D \end{array}$$

Since we want the structure of types and derivations to correspond as much as possible, the set $\mathcal{T}$ will *not* be considered modulo . Since intersection and union are associative and commutative, we will write $\cap_n \mathsf{A}_i$ for $\mathsf{A}_1 \cap \cdots \cap \mathsf{A}_n$, as well as $\cup_{\underline{n}} A_i$ for $\mathsf{A}_1 \cup \cdots \cup \mathsf{A}_n$, and will only allow permutations of types within an intersection or within a union.

We will now present a notion of intersection-union typing for $\overline{\lambda}\mu\tilde{\mu}$, system $\mathcal{M}^{\cap\cup}$, as defined in [8], adapted to our purposes.

Definition 2.3 ($\cap\vdash\cup$ typing for $\overline{\lambda}\mu\tilde{\mu}$)

$$(cut):\ \frac{\Gamma_1 \vdash_{\mathcal{M}} v : A \mid \Delta_1 \quad \Gamma_2 \mid e : A \vdash_{\mathcal{M}} \Delta_2}{\langle v \,|\, e\rangle : \Gamma_1\cap\Gamma_2 \vdash_{\mathcal{M}} \Delta_1\cup\Delta_2}$$

$$(Ax\text{-}R):\ \frac{}{\Gamma\cap x{:}A \vdash_{\mathcal{M}} x : A \mid \Delta} \qquad (Ax\text{-}L):\ \frac{}{\Gamma \mid \alpha : A \vdash_{\mathcal{M}} \alpha{:}A\cup\Delta}$$

$$(\to R):\ \frac{\Gamma, x{:}A \vdash_{\mathcal{M}} v : B \mid \Delta}{\Gamma \vdash_{\mathcal{M}} \lambda x.v : A\to B \mid \Delta} \qquad (\to L):\ \frac{\Gamma_1 \vdash_{\mathcal{M}} v : A \mid \Delta_1 \quad \Gamma_2 \mid e : B \vdash_{\mathcal{M}} \Delta_2}{\Gamma_1\cap\Gamma_2 \mid v\cdot e : A\to B \vdash_{\mathcal{M}} \Delta_1\cup\Delta_2}$$

$$(\mu):\ \frac{c : \Gamma \vdash_{\mathcal{M}} \alpha{:}A, \Delta}{\Gamma \vdash_{\mathcal{M}} \mu\alpha.c : A \mid \Delta} \qquad (\tilde{\mu}):\ \frac{c : \Gamma, x{:}A \vdash_{\mathcal{M}} \Delta}{\Gamma \mid \tilde{\mu}x.c : A \vdash_{\mathcal{M}} \Delta}$$

$$(\cap R):\ \frac{\Gamma_i \vdash_{\mathcal{M}} v : A_i \mid \Delta_i \quad (\forall i \in \underline{n})}{\cap_n\Gamma_i \vdash_{\mathcal{M}} v : \cap_n \mathsf{A}_i \mid \cup_{\underline{n}}\Delta_i}\ (n \geq 0) \qquad (\cup L):\ \frac{\Gamma_i \mid e : A_i \vdash_{\mathcal{M}} \Delta_i \quad (\forall i \in \underline{n})}{\cap_n\Gamma_i \mid e : \cup_{\underline{n}} A_i \vdash_{\mathcal{M}} \cup_{\underline{n}}\Delta_i}\ (n \geq 0)$$

$$(\cup R):\ \frac{\Gamma_i, x{:}B_i \vdash_{\mathcal{M}} v : A_j \mid \Delta_i}{\cap_n\Gamma_i, x{:}\cup_{\underline{n}} B_i \vdash_{\mathcal{M}} v : \cup_{\underline{n}} A_i \mid \cup_{\underline{n}}\Delta_i}\ (j \in \underline{n}) \qquad (\cap L):\ \frac{\Gamma_i \mid e : A_j \vdash_{\mathcal{M}} \alpha{:}B_i, \Delta_i}{\cap_n\Gamma_i \mid e : \cap_n \mathsf{A}_i \vdash_{\mathcal{M}} \alpha{:}\cap_n \mathsf{B}_i, \cup_{\underline{n}}\Delta_i}\ (j \in \underline{n})$$

This system is a multiplicative variant of system $\mathcal{M}^{\cap\cup}$ of [8], but for the fact that we deal with $(\top)$ and $(\bot)$, and generalise rules $(\cup R)$ and $(\cap L)$. Notice that the rules $(\top)$ and $(\bot)$ are special cases of $(\cap R)$ and $(\cup L)$ by taking $n = 0$ as well and conveniently ignoring the fact that an intersection (or union) of an empty collection of sets is the empty set.

Notice that, in rule $(Ax\text{-}R)$ (and $(Ax\text{-}L)$), a type is selected from within an intersection (a union) for a term (context) variable. To obtain the same functionality for the *implicit* variables, the system adds rules $(\cup R)$ (and $(\cap L)$).

Also, their notion of typing is defined without any relation on types, it is just stated that types are considered to be defined modulo commutativity and associativity for $\cap$ and $\cup$; this corresponds to the type inclusion relation we use here. Notice we do not have a contravariant type inclusion relation (which would state that $C \leq A \;\&\; B \leq D \Rightarrow A\to B \leq C\to D$); this combined with the full BCD-intersection types gives a notion of typing that is closest to Krivine's system $\mathcal{D}$ (see Theorem 3.2).

3 Properties for System $\vdash_{\mathcal{M}}$

In this section we study some of the properties of $\vdash_{\mathcal{M}}$ in detail.

As for the strict system, we can easily show the following.

Lemma 3.1 The following rules are admissible.

$$(\cap E): \frac{\Gamma \vdash_{\mathcal{M}} v : \cap_n \mathsf{A}_i \mid \Delta}{\Gamma \vdash_{\mathcal{M}} v : A_j \mid \Delta}\ (j \in \underline{n}) \qquad (\cup E): \frac{\Gamma \mid e : \cup_{\underline{n}} A_i \vdash_{\mathcal{M}} \Delta}{\Gamma \mid e : A_j \vdash_{\mathcal{M}} \Delta}\ (j \in \underline{n})$$

We can now show that typing is preserved by the interpretation of the λ-calculus:

Theorem 3.2 *If $\Gamma \vdash_{D} M : A$ then $\Gamma \vdash_{\mathcal{M}} [\![M]\!]_{\lambda} : A \mid$.*

Notice that typing is not preserved for the rule (η): for example, we can derive

$$y{:}A{\to}B \vdash_{\mathcal{M}} \lambda x.\mu\beta.\langle y \mid x{\cdot}\beta\rangle : (A \cap C){\to}B \mid$$

but cannot show $y{:}A{\to}B \vdash_{\mathcal{M}} y : (A \cap C){\to}B \mid$; we would need a $\leq$-relation on types that is contra-variant over arrow types.

It is easy to show the following completeness result.

Theorem 3.3 (Subject expansion) *Let $c_1 \to c_2$: if $c_2 : \Gamma \vdash_{\mathcal{M}} \Delta$ then $c_1 : \Gamma \vdash_{\mathcal{M}} \Delta$.*

As for the system suggested in [16] (there only the intersection and union rules for a CBN and CBV restriction are given), the added rules are:

$$(\cap L): \frac{\Gamma \mid e : A_i \vdash_{\mathcal{M}} \Delta}{\Gamma \mid e : A_1 \cap A_2 \vdash_{\mathcal{M}} \Delta}\ (i = 1,2) \qquad (\cap R): \frac{\Gamma \vdash_{\mathcal{M}} v : A_1 \mid \Delta \quad \Gamma \vdash_{\mathcal{M}} v : A_2 \mid \Delta}{\Gamma \vdash_{\mathcal{M}} v : A_1 \cap A_2 \mid \Delta}$$

$$(\cup L): \frac{\Gamma \mid e : A_1 \vdash_{\mathcal{M}} \Delta \quad \Gamma \mid e : A_2 \vdash_{\mathcal{M}} \Delta}{\Gamma \mid e : A_1 \cup A_2 \vdash_{\mathcal{M}} \Delta} \qquad (\cap L): \frac{\Gamma \vdash_{\mathcal{M}} v : A_i \mid \Delta}{\Gamma \vdash_{\mathcal{M}} v : A_1 \cup A_2 \mid \Delta}\ (i = 1,2)$$

With these rules, none of the above result can be achieved; for example, there we cannot derive $\vdash_{\mathcal{M}} [\![\lambda x.xx]\!]_{\lambda} : ((A{\to}B) \cap A){\to}B \mid$, so cannot type all normal forms. Notice the role of intersection on term variables and union on context variables in Theorem 3.3; we cannot prove such a property for the system of [16].

As we will show in the next section, we cannot prove a general subject *reduction* result. However, we can show that $\tilde{\mu}$-reduction towards an intersection and μ-reduction towards a union are safe, as well as rule $(\to)$; we will start with the latter:

Theorem 3.4 (Soundness for rule $(\to)$) *If $\langle \lambda x.v_1 \mid v_2{\cdot}e\rangle : \Gamma \vdash_{\mathcal{M}} \Delta$ then $\langle v_2 \mid \tilde{\mu}x.\langle v_1 \mid e\rangle\rangle : \Gamma \vdash_{\mathcal{M}} \Delta$.*

The following lemmas show that μ-contraction to any type A, even a union, such that $c : \Gamma \vdash_{\mathcal{M}} \alpha{:}A, \Delta$ is sound, as is a $\tilde{\mu}$-contraction to any type A, even an intersection, such that $c : \Gamma, x{:}A \vdash_{\mathcal{M}} \Delta$.

Lemma 3.5 (Safe term substitution lemma) *Let $B \in \mathcal{T}$.*

i) If $c : \Gamma, x{:}B \vdash_{\mathcal{M}} \Delta$ and $\Gamma \vdash_{\mathcal{M}} v : B \mid \Delta$, then $c[v/x] : \Gamma \vdash_{\mathcal{M}} \Delta$.

ii) If $\Gamma, x{:}B \vdash_{\mathcal{M}} v' : A \mid \Delta$ and $\Gamma \vdash_{\mathcal{M}} v : B \mid \Delta$, then $\Gamma \vdash_{\mathcal{M}} v'[v/x] : A \mid \Delta$.

iii) If $\Gamma, x{:}B \mid e : A \vdash_{\mathcal{M}} \Delta$ and $\Gamma \vdash_{\mathcal{M}} v : B \mid \Delta$, then $\Gamma \mid e[v/x] : A \vdash_{\mathcal{M}} \Delta$.

Lemma 3.6 (Safe context substitution lemma) *Let* $B \in \mathcal{T}$.

i) If $c : \Gamma \vdash_{\mathcal{M}} \alpha{:}B, \Delta$ *and* $\Gamma \mid e : B \vdash_{\mathcal{M}} \Delta$, *then* $c[e/\alpha] : \Gamma \vdash_{\mathcal{M}} \Delta$.

ii) If $\Gamma \vdash_{\mathcal{M}} v : A \mid \alpha{:}B, \Delta$ *and* $\Gamma \mid e : B \vdash_{\mathcal{M}} \Delta$, *then* $\Gamma \vdash_{\mathcal{M}} v[e/\alpha] : A \mid \Delta$.

iii) If $\Gamma \mid e' : A \vdash_{\mathcal{M}} \alpha{:}B, \Delta$ *and* $\Gamma \mid e : B \vdash_{\mathcal{M}} \Delta$, *then* $\Gamma \mid e'[e/\alpha] : A \vdash_{\mathcal{M}} \Delta$.

The promised reduction results for both $\tilde{\mu}$ and μ reduction are in the first parts, respectively, of these lemmas.

4 The problem with subject reduction

In this section we will focus on soundness, i.e. investigate when assignable types (and contexts) are preserved by reduction.

Example 4.1 Take the commands $c_1 = \langle \mu\alpha.\langle x \,|\, x{\cdot}\alpha\rangle \,|\, \tilde{\mu}y.\langle y \,|\, y{\cdot}\beta\rangle\rangle$, $c_2 = \langle x \,|\, x{\cdot}(\mu\alpha.\langle x \,|\, x{\cdot}\alpha\rangle{\cdot}\beta)\rangle$, and $c_3 = \langle x \,|\, x \cdot \tilde{\mu}y.\langle y \,|\, y{\cdot}\beta\rangle\rangle$, and notice that $c_1 \to_{\mathrm{v}} c_2$, and $c_1 \to_{\mathrm{N}} c_3$. We can easily derive

$$c_2 : x{:}A \cap (A \to ((C \to D) \cap C)) \vdash_{\mathcal{M}} \beta{:}D$$

and

$$c_3 : x{:}A \cap (A \to C) \cap (A \to C \to D) \vdash_{\mathcal{M}} \beta{:}D,$$

and (by subject expansion) can derive both typings also for c_1, but subject reduction fails: we cannot derive c_3's typing for c_2, but can only derive c_2's typing for c_3.

Notice that the typing for c_2 shows the usefulness of allowing intersection types on the right of an arrow.

The example above shows a counterexample for CBV reduction, but we can also give one for CBN:

Example 4.2 Take the commands

$$\begin{array}{rcl} c_1 & = & \langle \mu\delta.\langle \lambda x.\mu\beta.\langle x \,|\, \delta\rangle \,|\, \delta\rangle \,|\, \tilde{\mu}z.\langle \lambda v\mu\alpha.\langle z \,|\, \alpha\rangle \,|\, \gamma\rangle\rangle, \\ c_2 & = & \langle \lambda v.\mu\alpha.\langle \mu\delta.\langle \lambda x.\mu\beta.\langle x \,|\, \delta\rangle \,|\, \delta\rangle \,|\, \alpha\rangle \,|\, \gamma\rangle, \text{ and} \\ c_3 & = & \langle \lambda v.\mu\alpha.\langle \lambda x\mu\beta.\langle x \,|\, \tilde{\mu}z.\langle \lambda w.\mu\alpha.\langle z \,|\, \alpha\rangle \,|\, \gamma\rangle\rangle \,|\, \alpha\rangle \,|\, \gamma\rangle \end{array}$$

then $c_1 \to_{\mathrm{N}} c_2$ and $c_1 \to_{\mathrm{v}} c_3$. We can derive

$$c_2 : \vdash_{\mathcal{M}} \gamma{:}C \to A \cup (A \to B)$$

and

$$c_3 : \vdash_{\mathcal{M}} \gamma{:}(C \to A) \cup (C \to A \to B)$$

and both these typings also for c_1, as well as c_2's for c_3, but not c_3's for c_2.

So we cannot achieve soundness (or subject reduction) in this system for $\overline{\lambda}\mu\tilde{\mu}$.

Performing a μ-reduction towards an intersection type, or performing a $\tilde{\mu}$-reduction towards a union type, is problematic. Let's first of all analyse the situation. If an intersection has been

used to type a command, we find ourselves in either one of the following two situations:

$$\dfrac{\dfrac{\dfrac{\boxed{\mathcal{D}_i}}{c : \Gamma_i \vdash_{\mathcal{M}} \alpha{:}A_i, \Delta_i}(\mu)}{\Gamma_i \vdash_{\mathcal{M}} \mu\alpha.c : A_i \mid \Delta_i}(\forall i \in \underline{n})}{\cap_n \Gamma_i \vdash_{\mathcal{M}} \mu\alpha.c : \cap_n A_i \mid \cup_{\underline{n}} \Delta_i}(\cap R) \quad \dfrac{\boxed{\mathcal{D}}}{\Gamma \mid e : \cap_n A_i \vdash_{\mathcal{M}} \Delta}}{\langle \mu\alpha.c \,|\, e\rangle : \cap_n \Gamma_i \cap \Gamma \vdash_{\mathcal{M}} \cup_{\underline{n}} \Delta_i \cup \Delta}(cut)$$

$$\dfrac{\dfrac{\dfrac{\boxed{\mathcal{D}'}}{c : \Gamma \vdash_{\mathcal{M}} \alpha{:}\cap_n A_i, \Delta}(\mu)}{\Gamma \vdash_{\mathcal{M}} \mu\alpha.c : \cap_n A_i \mid \Delta} \quad \dfrac{\boxed{\mathcal{D}}}{\Gamma \mid e : \cap_n A_i \vdash_{\mathcal{M}} \Delta}}{\langle \mu\alpha.c \,|\, e\rangle : \cap_n \Gamma_i \cap \Gamma \vdash_{\mathcal{M}} \cup_{\underline{n}} \Delta_i \cup \Delta}(cut)$$

Only the first case is problematic, since from the second we get $c[e/\alpha] : \cap_n \Gamma_i \cap \Gamma \vdash_{\mathcal{M}} \cup_{\underline{n}} \Delta_i \cup \Delta$ by Lemma 3.6. For the first case, we normally cannot remove any of the types in $\cap_n A_i$ in the derivation $\mathcal{D}$, so we cannot safely propagate $\mathcal{D}$ into the various $\mathcal{D}_i$. So this is the problem case that should be caught.

This problem also appears when dealing with quantification, as in ML: assume we have a derivation for (example essentially by Alexander J. Summers: see [23] for a detailed treatment of quantification in sequent calculi).

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x{:}\phi, y{:}\phi \vdash_{\mathcal{M}} x : \phi \mid}}{x{:}\phi \vdash_{\mathcal{M}} \lambda y.x : \phi \to \phi \mid}(\to R) \quad \vdots \quad \dfrac{}{\mid \gamma : \phi\to\phi \vdash_{\mathcal{M}} \beta{:}\phi, \gamma{:}\phi\to\phi}}{\langle \lambda y.x \,|\, \gamma\rangle : x{:}\phi \vdash_{\mathcal{M}} \beta{:}\phi, \gamma{:}\phi\to\phi}(cut)}{x{:}\phi \vdash_{\mathcal{M}} \mu\beta.\langle \lambda y.x \,|\, \gamma\rangle : \phi \mid \gamma{:}\phi\to\phi}(\mu)}{\vdash_{\mathcal{M}} \lambda x.\mu\beta.\langle \lambda y.x \,|\, \gamma\rangle : \phi\to\phi \mid \gamma{:}\phi\to\phi}(\to R) \quad \dfrac{\dfrac{}{y{:}\forall\phi.\phi\to\phi \vdash_{\mathcal{M}} y : \forall\phi.\phi\to\phi \mid}(\forall E) \quad \dfrac{}{\mid \delta : \phi'\to\phi' \vdash_{\mathcal{M}} \delta{:}\phi'\to\phi'}}{y{:}\forall\phi.\phi\to\phi \mid y{\cdot}\delta : (\phi'\to\phi')'\to\phi'\to\phi' \vdash_{\mathcal{M}} \delta{:}\phi'\to\phi'}(\to L)$$

$$\vdots \qquad \vdots$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mid \gamma : \phi\to\phi \vdash_{\mathcal{M}} \gamma{:}\phi\to\phi}}{\langle \lambda x.\mu\beta.\langle \lambda y.x \,|\, \gamma\rangle \,|\, \gamma\rangle : \Gamma \vdash_{\mathcal{M}} \gamma{:}\phi\to\phi}(cut)}{\Gamma \vdash_{\mathcal{M}} \mu\gamma.\langle \lambda x.\mu\beta.\langle \lambda y.x \,|\, \gamma\rangle \,|\, \gamma\rangle : \phi\to\phi \mid}(\mu)}{\Gamma \vdash_{\mathcal{M}} \mu\gamma.\langle \lambda x.\mu\beta.\langle \lambda y.x \,|\, \gamma\rangle \,|\, \gamma\rangle : \forall\phi.\phi\to\phi \mid}(\forall R) \quad \dfrac{\dfrac{\dfrac{}{y{:}\forall\phi.\phi\to\phi \vdash_{\mathcal{M}} y : \forall\phi.\phi\to\phi \mid}(\forall E)}{y{:}\forall\phi.\phi\to\phi \vdash_{\mathcal{M}} y : (\phi'\to\phi')'\to\phi'\to\phi' \mid}}{\langle y \,|\, y{\cdot}\delta\rangle : \Gamma, y{:}y{:}\forall\phi.\phi\to\phi \vdash_{\mathcal{M}} \delta{:}\phi'\to\phi'}(cut)}{y{:}\forall\phi.\phi\to\phi \mid \tilde{\mu}y.\langle y \,|\, y{\cdot}\delta\rangle : \forall\phi.\phi\to\phi \vdash_{\mathcal{M}} \delta{:}\phi'\to\phi'}(\tilde{\mu})$$

$$\dfrac{}{\langle \mu\gamma.\langle \lambda x.\mu\beta.\langle \lambda y.x \,|\, \gamma\rangle \,|\, \gamma\rangle \,|\, \tilde{\mu}y.\langle y \,|\, y{\cdot}\delta\rangle\rangle : y{:}\forall\phi.\phi\to\phi \vdash_{\mathcal{M}} \delta{:}\phi\to\phi}(cut)$$

then the μ-reduction will cause problems. Notice that $\gamma{:}\phi\to\phi$ is contracted, so in fact occurs twice: bringing the quantifier inside, as would be required when μ-reducing, would force two independent closures of the type $\phi\to\phi$ for γ; this is impossible, because the 'other' γ will have the ϕ free, which violates the side-condition on rule $\forall R$.

Similarly, for union, we have the situations:

$$\dfrac{\boxed{\mathcal{D}} \quad \dfrac{\boxed{\mathcal{D}'}}{c : \Gamma_i, x{:}\cup_{\underline{n}} A_i \vdash_{\mathcal{M}} \Delta_i}}{\Gamma \vdash_{\mathcal{M}} v : \cup_{\underline{n}} A_i \mid \Delta \quad \Gamma_i \mid \tilde{\mu}x.c : \cup_{\underline{n}} A_i \vdash_{\mathcal{M}} \Delta_i}(\tilde{\mu})}{\langle v \,|\, \tilde{\mu}x.c\rangle : \cap_n \Gamma_i \cap \Gamma \vdash_{\mathcal{M}} \cup_{\underline{n}} \Delta_i \cup \Delta}(cut)$$

$$\dfrac{\boxed{\mathcal{D}}\ \Gamma \vdash_{\mathcal{M}} v : \cup_{\underline{n}} A_i \mid \Delta \qquad \dfrac{\dfrac{\boxed{\mathcal{D}_i}\ c : \Gamma_i, x{:}A_i \vdash_{\mathcal{M}} \Delta_i}{\Gamma_i \mid \tilde{\mu}x.c : A_i \vdash_{\mathcal{M}} \Delta_i}\,(\tilde{\mu})}{\cap_n \Gamma_i \mid \tilde{\mu}x.c : \cup_{\underline{n}} A_i \vdash_{\mathcal{M}} \cup_{\underline{n}} \Delta_i}\,(\forall i \in \underline{n})\,(\cup L)}{\langle v \mid \tilde{\mu}x.c\rangle : \cap_n \Gamma_i \cap \Gamma \vdash_{\mathcal{M}} \cup_{\underline{n}} \Delta_i \cup \Delta}\,(cut)$$

As above, only the second situation is possibly problematic, and propagating $\mathcal{D}$ into the $\mathcal{D}_i$ should be avoided.

In [16], the subject reduction problem is recognised, and two systems are proposed that solve the problem, one for CBV, and one for CBN reduction, as we will do in Section 5. There the CBV restriction is defined by limiting rule $(\cup R)$ to values, and the CBN restriction by limiting rule $(\cap L)$ to slots. How this will achieve subject-reduction is not made clear; it is remarkable that the suggested solution is almost orthogonal to what we define below. However, this does not seem to affect the validity of the subject reduction claim in [16]; in fact, we conjecture that, given the applicability of rule $(\cap L)$ only to contexts and rule $(\cup R)$ only to terms, subject reduction even holds for the *unrestricted* system of that paper.

5 Sound restrictions of $\vdash$

In this section, we will study two restrictions for which we can show a subject reduction result. First of all, we can show that the rule $(\rightarrow)$, $\tilde{\mu}$-reduction towards an intersection, and μ-reduction towards a union are safe.

Unfortunately, as suggested above (and noted in [16], and observed by others), we cannot show the above soundness result for the other two main reduction rules, $\rightarrow_{\mu}$ or $\rightarrow_{\tilde{\mu}}$. To show the crucial defects, we presented two counterexamples in Examples 4.1 and 4.2. This is mainly due to the fact that $\Gamma \vdash_{\mathcal{M}} \mu\alpha.c : A \mid \Delta$ does not always imply $c : \Gamma \vdash_{\mathcal{M}} \alpha{:}A, \Delta$, and $\Gamma \mid \tilde{\mu}x.c : A \vdash_{\mathcal{M}} \Delta$ does not always imply $c : \Gamma, x{:}A \vdash_{\mathcal{M}} \Delta$.

Summarising, it is problematic to: *i) perform a μ-reduction towards an intersection type, introduced via $(\cap R)$, or ii) perform a $\tilde{\mu}$-reduction towards a union type, introduced via $(\cup L)$.*

It seems reasonable to allow full $(\cap R)$ for CBN; assuming we use the restriction mentioned above on $(\cup L)$, how do we stop reducing into an intersection? Remember that this only happens in case $\langle \mu\alpha.c \mid e\rangle$, with e a slot; in that case, the intersection type on the right is superfluous, and only one proper type inside it is needed.

Definition 5.1 The notion of typing $\vdash_{\mathbf{N}}$ is defined as $\vdash_{\mathcal{M}}$, by changing rules $(\cup L)$ and $(\cup R)$:

$$(\cup L):\ \frac{\Gamma_i \mid E : A_i \vdash_{\mathbf{N}} \Delta_i \quad (\forall i \in \underline{n})}{\cap_n \Gamma_i \mid E : \cup_{\underline{n}} A_i \vdash_{\mathbf{N}} \cup_{\underline{n}} \Delta_i}\,(n \geq 1) \qquad (\cup R):\ \frac{\Gamma \vdash_{\mathcal{M}} v : A_j \mid \Delta}{\Gamma \vdash_{\mathcal{M}} v : \cup_{\underline{n}} A_i \mid \Delta}\,(j \in \underline{n}, n \geq 1)$$

That subject-expansion now fails is immediate, since the condition on rule $(\cup L)$ (the context is a slot) is not preserved by expansion. We can now show:

Theorem 5.2 (SOUNDNESS FOR $\rightarrow_{\mathbf{N}}$ IN $\vdash_{\mathbf{N}}$) *If $c_1 : \Gamma \vdash_{\mathbf{N}} \Delta$ and $c_1 \rightarrow_{\mathbf{N}} c_2$, then $c_2 : \Gamma \vdash_{\mathbf{N}} \Delta$.*

For CBV, we take the dual approach, and restrict the use of $(\cap R)$. To avoid a μ-reduction into an intersection, we either do not allow intersections for cuts at all (which, by the example above, is clearly to restrictive), or make sure that no intersection is introduced for $\mu\alpha.c$ via the restriction on $(\cap R)$. The only thing to solve is then to avoid the contraction of $\langle v \mid \tilde{\mu}x.c\rangle$ into a union, which is solved by the fact that then v is a value.

Definition 5.3 The notion of typing $\vdash_{\mathrm{v}}$ is defined as $\vdash_{\mathcal{M}}$, by changing rules $(\cap R)$ and $(\cap L)$:

$$(\cap L):\ \frac{\Gamma \mid e : A_j \vdash_{\mathcal{M}} \Delta}{\Gamma \mid e : \cap_n \mathsf{A}_i \vdash_{\mathcal{M}} \Delta}\ (j \in \underline{n}, n \geq 1) \qquad (\cap R):\ \frac{\Gamma_i \vdash_{\mathrm{v}} V : A_i \mid \Delta_i \quad (\forall i \in \underline{n})}{\cap_n \Gamma_i \vdash_{\mathrm{v}} V : \cap_n \mathsf{A}_i \mid \cup_{\underline{n}} \Delta_i}\ (n \geq 0)$$

That subject-expansion now fails is again immediate, since the condition on rule $(\cap R)$ (the term is a value) is not preserved by expansion.

Theorem 5.4 (Soundness for $\to_{\mathrm{v}}$ in $\vdash_{\mathrm{v}}$) *If $c : \Gamma \vdash_{\mathrm{v}} \Delta$ and $c \to_{\mathrm{v}} c'$, then $c' : \Gamma \vdash_{\mathrm{v}} \Delta$.*

6 Concluding remarks

It is worthwhile to observe that the system presented above is just *one* of the possible ways of dealing with intersection and union in the context of $\overline{\lambda}\mu\tilde{\mu}$. In fact, allowing intersection and union for activated formulae only, actually leans strongly on the similarity between those type constructs and the logical & and $\vee$.

We have seen that it is straightforward to define a natural notion of typing to the sequent calculus $\overline{\lambda}\mu\tilde{\mu}$ that uses intersection and union types. This system was shown natural in that we were able to show that subject expansion - the main reason to use either intersection or union - follows easily, and that both intersection and union play their natural and crucial role in that proof.

However, this ease is not paired with soundness with respect to subject reduction. As in similar notions for the λ-calculus, combining union and intersection types breaks the soundness of the system; unlike the λ-calculus, however, also intersection is problematic. We have isolated the problem cases, and seen that it is exactly the non-logical behaviour of *both* type constructors that causes the problem. We have looked at restrictions for either cbn or cbv reduction that overcome this defect, but all with the loss of the subject expansion result.

Acknowledgements

I would like to thank Silvia Ghilezan, Pierre Lescanne, Mariangiola Dezani, Alexander J. Summers and especially Dan Dougherty for detailed, critical and constructive discussions on the results of this paper.

References

[1] S. van Bakel. Complete restrictions of the Intersection Type Discipline. *Theoretical Computer Science*, 102(1):135–163, 1992.

[2] S. van Bakel. Intersection Type Assignment Systems. *Theoretical Computer Science*, 151(2):385–435, 1995.

[3] F. Barbanera, M. Dezani-Ciancaglini, and U. de'Liguoro. Intersection and Union Types: Syntax and Semantics. *Information and Computation*, 119(2):202–230, 1995.

[4] H. Barendregt. *The Lambda Calculus: its Syntax and Semantics*. North-Holland, Amsterdam, revised edition, 1984.

[5] H. Barendregt, M. Coppo, and M. Dezani-Ciancaglini. A filter lambda model and the completeness of type assignment. *Journal of Symbolic Logic*, 48(4):931–940, 1983.

[6] P.-L. Curien and H. Herbelin. The Duality of Computation. In *Proceedings of the 5th ACM SIGPLAN International Conference on Functional Programming (ICFP'00)*, volume 35.9 of *ACM Sigplan Notices*, pages 233–243. ACM, 2000.

[7] R. Davies and F. Pfenning. Intersection types and computational effects. In *Proceedings of the 5th ACM SIGPLAN International Conference on Functional Programming (ICFP'00)*, volume 35.9 of *ACM Sigplan Notices*, pages 198–208. ACM Press, 2000.

[8] D. Dougherty, S. Ghilezan, and P Lescanne. Intersection and Union Types in the $\overline{\lambda}\mu\tilde{\mu}$-calculus. In *Electronic Proceedings of 3rd International Workshop* Intersection Types and Related Systems *(ITRS'04), Turku, Finland*, volume 136 of *Electronic Notes in Theoretical Computer Science*, pages 228–246, 2004.

[9] D. Dougherty, S. Ghilezan, and P. Lescanne. Characterizing strong normalization in the Curien-Herbelin symmetric lambda calculus: extending the Coppo-Dezani heritage. *Theoretical Computer Science*, 398, 2008.

[10] D. Dougherty, S. Ghilezan, P. Lescanne, and S. Likavec. Strong Normalization of the Dual Classical Sequent Calculus. In *Proceedings of 12th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'05)*, volume 3835 of *Lecture Notes in Computer Science*, pages 169–183, 2005.

[11] J. Dunfield and F. Pfenning. Tridirectional typechecking. In *Proceedings of the 27th ACM Symposium on Principles of Programming Languages (POPL'00)*, pages 281–292. ACM Press, 2000.

[12] M. Felleisen, D.P. Friedman, E. Kohlbecker, and B. Duba. A syntactic theory of sequential control. *Theoretical Computer Science*, 52:205–237, 1987.

[13] G. Gentzen. Investigations into logical deduction. In *The Collected Papers of Gerhard Gentzen*. Ed M. E. Szabo, North Holland, 68ff (1969), 1935.

[14] T. Griffin. A formulae-as-types notion of control. In *Proceedings of the 17th Annual ACM Symposium on Principles Of Programming Languages, Orlando (Fla., USA)*, pages 47–58, 1990.

[15] B. Harper and M Lillibridge. ML with callcc is unsound. Post to TYPES mailing list, July 8, 1991.

[16] H. Herbelin. *C'est maintenant qu'on calcule: au cœur de la dualité*. Mémoire d'habilitation, Université Paris 11, Décembre 2005.

[17] J.R. Hindley. Coppo-Dezani Types do not Correspond to Propositional Logic. *Theoretical Computer Science*, 28:235–236, 1984.

[18] S.C. Kleene. *Introduction to Metamathematics*. North Holland, Amsterdam, 1952.

[19] J.-L Krivine. *Lambda calculus, types and models*. Ellis Horwood, 1993.

[20] R. Milner, M. Tofte, R. Harper, and D. MacQueen. *The Definition of Standard ML*. MIT Press, 1990. Revised edition.

[21] C.-H. L. Ong and C.A. Stewart. A Curry-Howard foundation for functional computation with control. In *Proceedings of the 24th Annual ACM Symposium on Principles Of Programming Languages, Paris (France)*, pages 215–227, 1997.

[22] M. Parigot. An algorithmic interpretation of classical natural deduction. In *Proceedings of 3rd International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'92)*, volume 624 of *Lecture Notes in Computer Science*, pages 190–201. Springer Verlag, 1992.

[23] A.J. Summers. *Curry-Howard Term Calculi for Gentzen-Style Classical Logic*. PhD thesis, Imperial College London, 2008.

[24] C. Urban. *Classical Logic and Computation*. PhD thesis, University of Cambridge, October 2000.

[25] P. Wadler. Call-by-Value is Dual to Call-by-Name. In *Proceedings of the eighth ACM SIGPLAN international conference on Functional programming*, pages 189 – 201, 2003.

[26] A.K Wright. Simple imperative polymorphism. *Lisp and Symbolic Computation*, 8(4):343–355, 1995.