Sound and Complete Typing for $\lambda\mu$*

(ITRS'10, EPTCS volume 45, page 31-44, 2010)

Steffen van Bakel

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

Abstract

In this paper we define intersection and union type assignment for Parigot's calculus $\lambda\mu$. We show that this notion is complete (*i.e.* closed under subject-expansion), and show also that it is sound (*i.e.* closed under subject-reduction). This implies that this notion of intersection-union type assignment is suitable to define a semantics.

keywords: Classical Logic, $\lambda\mu$-Calculus, intersection types, union types.

Introduction

The Intersection Type Discipline has proven to be an expressive tool for studying termination and semantics for the λ-calculus [12, 10]. Intersection type assignment is defined as an extension of the standard, implicative type assignment known as Curry's system [15] (see also [24]), which expresses function composition and application; the extension made consists of relaxing the requirement that a parameter for a function should have a single type, adding the type constructor $\cap$ next to $\rightarrow$. This simple extension allows for a great leap in complexity: not only can a (filter) model be built for the λ-calculus using intersection types, also strong normalisation (termination) can be characterised via assignable types; however, type assignment becomes undecidable. The literature on intersection types is vast; it was first defined by Coppo and Dezani-Ciancaglini in [13] and its development took place over a number of years, culminating in the paper by Barendregt, Coppo, and Dezani-Ciancaglini [11], and has been explored by many people since.

It is natural to ask if these results can be achieved for other calculi (reduction systems) as well, and in previous papers the author investigated (in collaboration) Term Rewriting Systems [6], and Object Oriented Calculi [5]; Maffeis looked at intersection types in the context of the π-calculus [26]. In order to come to a characterisation of strong normalisation for Curien and Herbelin's (untyped) sequent calculus $\overline{\lambda}\mu\tilde{\mu}$ [14], Dougherty, Ghilezan and Lescanne presented System $\mathcal{M}^{\cap\cup}$ [17], that defines a notion of intersection and union typing for that calculus; in a later paper [18], they presented an improved version of their original system.

In [3], the author revisited System $\mathcal{M}^{\cap\cup}$, and showed that that system was neither *sound* (*i.e.* closed under reduction), nor *complete* (*i.e.* closed under reverse reduction); the same holds for the system presented in [18]. To address completeness, [3] adds $\top$ as the maximal and $\bot$ as the minimal type, and extends the set of derivation rules; however, soundness is shown to be impossible to achieve without restricting typeability (effectively making less terms typeable). In [4], the author attempted to solve the same issue, but this time in the context of the sequent

* Small mistakes in proofs are corrected here.

calculus $\mathcal{X}$, as defined by Lengrand [25], and later studied by Lescanne and the author [7, 8]; $\mathcal{X}$ is a sequent calculus in that it enjoys the Curry-Howard isomorphism with respect to the implicative fragment of Gentzen's LK [20]. The advantage of using the sequent approach is that it is now possible to explore the duality of intersection and union fully, through which we can study and explain various anomalies of union type assignment [32, 9] and quantification [22, 27]. Also for $\mathcal{X}$, the completeness result follows relatively easily, but soundness can only be shown for restricted systems (effectively call-by-name and call-by-value reduction, but it might be possible that other sound restrictions exist as well). The main conclusion of those papers is that, in symmetric calculi (like $\overline{\lambda}\mu\tilde{\mu}$ and $\mathcal{X}$) it is inevitable that intersection and union are truly dual, and that the very nature of those calculi makes a sound and complete system unachievable.

In this paper we will continue on this path and bring intersection types to the context of *classical logic*, by presenting a notion of intersection and union type assignment for the (untyped) calculus $\lambda\mu$, that was first defined by Parigot in [31], and was later extensively studied by various authors.

Intersection and union types have also been studied in the context of the λ-calculus in [9]; also for the system defined in that paper soundness is lost, which can only be recovered by limiting to parallel reduction, *i.e.* all residuals of a redex need to be contracted in parallel. The problem of loss of soundness also appears in other contexts, such as that of ML with side-effects [22, 33, 27], and that of using intersection and union types in an operational setting [16, 19]. As here, also there the cause of the problem is that the type-assignment rules are not fully logical, making the context calls (which form part of the reduction in $\mathcal{X}$) unsafe; this has, in part, already been observed in [23] in the context of Curien and Herbelin's calculus $\overline{\lambda}\mu\tilde{\mu}$ [14]. This also explains why, for ML with side-effects, quantification is no longer sound [22, 27]: also the $(\forall I)$ and $(\forall E)$ rules of ML are not logical.

In the view of those failures, the result presented here comes as a surprise. We will define a notion of type assignment for $\lambda\mu$ that uses intersection and union types, and show that it is *both* sound *and* complete. The system presented is a natural extension of the strict intersection type assignment system as defined in [2]; this implies that intersection models the distribution of arguments in a parameter call. But it is also a natural extension of the system for $\lambda\mu$, and in order to achieve completeness for structural reduction, as in the papers mentioned above, union types are added. However, the union types are no longer dual to intersection types; union types play only a marginal role, as was also the intention of [18]. Contrary to that paper, however, we do not see union as negated intersection, but see a union type as a strict type; in particular, we do not allow the normal $(\cup I)$ and $(\cup E)$ rules as used in [9], which we know create the same soundness problem. Moreover, although one can link intersection types with the logical connector *and*, the union types we use here have *no* relation with *or*; one could argue that therefore perhaps *union* is not the right name to use for this type constructor, but we will stick with it nonetheless.

The limited view of union types is mirrored by $\lambda\mu$'s limited (with respect to $\overline{\lambda}\mu\tilde{\mu}$ and $\mathcal{X}$) notion of context[1]. In $\lambda\mu$, we distinguish *control structures* as those terms that start with a context switch $\mu\alpha.[\beta]M$, followed by a number of arguments; since union types allow us to express that the various continuations (all called α) need not have the same type, we use a different formulation for rule $(\rightarrow E)$, which has an implicit use of union elimination (see Definition 3.2). The type system defined here will be shown to be the natural one, in that intersection and union play their expected roles for completeness. Because the use of intersection and union is limited in that a context variable cannot have an intersection type, and although we allow

[1] In particular, $\overline{\lambda}\mu\tilde{\mu}$'s $\tilde{\mu}x.c$ is not represented.

union types for term variables, we do not have the normal union elimination rule; thanks to these two restrictions, we can show soundness as well.

1 The calculus $\lambda\mu$

Parigot's $\lambda\mu$-calculus [29] is a proof-term syntax for classical logic, expressed in Natural Deduction, defined as an extension of the Curry type assignment system for the λ-calculus. We quickly revise some basic notions:

Definition 1.1 (LAMBDA TERMS AND β-CONTRACTION [10]) *i*) *λ-terms* are defined by:

$$M,N \ ::= \ x \mid \lambda x.M \mid MN$$

ii) The reduction relation $\rightarrow_\beta$ is defined as the contextual closure of the rule:

$$(\lambda x.M)N \ \rightarrow_\beta \ M[N/x]$$

Curry (or simple) type assignment for the λ-calculus is defined as:

Definition 1.2 *i*) Let φ range over a countable (infinite) set of type-variables. The set of *Curry-types* is defined by the grammar:

$$A,B \ ::= \ \varphi \mid (A{\rightarrow}B)$$

ii) *Curry-type assignment* is defined by the following natural deduction system.

$$(Ax):\ \frac{}{\Gamma,x{:}A \vdash x:A} \qquad (\rightarrow I):\ \frac{\Gamma,x{:}A \vdash M:B}{\Gamma \vdash \lambda x.M:A{\rightarrow}B} \qquad (\rightarrow E):\ \frac{\Gamma \vdash M:A{\rightarrow}B \quad \Gamma \vdash N:A}{\Gamma \vdash MN:B}$$

With $\lambda\mu$ Parigot created a multi-conclusion typing system which corresponds to classical logic; the derivable statements have the shape $\Gamma \vdash_{\lambda\mu} M : A \mid \Delta$, where A is the main conclusion of the statement, expressed as the *active* conclusion, and Δ contains the alternative conclusions, consisting of pairs of Greek characters and types; the left-hand context Γ, as usual, contains pairs of Roman characters and types, and represents the types of the free term variables of M. As with Implicative Intuitionistic Logic, the reduction rules for the terms that represent the proofs correspond to proof contractions; the difference is that the reduction rules for the λ-calculus are the *logical* reductions, *i.e.* deal with the elimination of a type construct that has been introduced directly above. In addition to these, Parigot expresses also the *structural* rules, where elimination takes place for a type constructor that appears in one of the alternative conclusions (the Greek variable is the name given to a subterm): he therefore needs to express that the focus of the derivation (proof) changes, and this is achieved by extending the syntax with two new constructs $[\alpha]M$ and $\mu\alpha.M$ that act as witness to *deactivation* and *activation*, which together move the focus of the derivation.

We will now present the variant of $\lambda\mu$ we consider in this paper, as considered by Parigot in [30]; for convenience, we split terms into two categories: we define *terms*, and *control structure*:

Definition 1.3 (SYNTAX OF $\lambda\mu$) The *$\lambda\mu$-terms* we consider are:

$$M,N ::= x \mid \lambda x.M \mid MN \mid \mu\alpha.[\beta]M.$$

We also define *control structure* as a subset of terms: $C ::= \mu\alpha.[\beta]M \mid CM$.

To shorten proofs and notation, we will treat $\mu\alpha.M$ as a term as well, whenever convenient.

As usual, $\lambda x.M$ binds x in M, and $\mu\alpha.[\beta]M$ binds α in M, and the notions of free and bound variables are defined accordingly; the notion of α-conversion extends naturally to bound names, and we assume Barendregt's convention on free and bound variables.

In $\lambda\mu$, reduction of terms is expressed via implicit substitution; as usual, $M\{N/x\}$ stands for the substitution of all occurrences of x in M by N, and $M\{N{\cdot}\gamma/\alpha\}$ stands for the term obtained from M in which every (pseudo) sub-term of the form $[\alpha]M'$ is substituted by $[\gamma](M'N)$ (γ is a fresh variable) (in Parigot's notation: $(\mu\alpha.[\beta]M)N \rightarrow \mu\gamma.[\beta]M[[\gamma]PN/[\alpha]P])$.

We define formally how to preform the μ-substitution; this is convenient in later proofs.

Definition 1.4 We define $M\{N{\cdot}\gamma/\alpha\}$ by induction over the structure of terms by:

$$\begin{array}{rcll} x\,\{N{\cdot}\gamma/\alpha\} & \triangleq & x & \\ (\lambda x.M)\,\{N{\cdot}\gamma/\alpha\} & \triangleq & \lambda x.(M\{N{\cdot}\gamma/\alpha\}) & \\ (PQ)\,\{N{\cdot}\gamma/\alpha\} & \triangleq & P\{N{\cdot}\gamma/\alpha\}\;Q\{N{\cdot}\gamma/\alpha\} & \\ (\mu\delta.[\alpha]M)\,\{N{\cdot}\gamma/\alpha\} & \triangleq & \mu\delta.[\gamma](M\{N{\cdot}\gamma/\alpha\}N) & \\ (\mu\delta.[\beta]M)\,\{N{\cdot}\gamma/\alpha\} & \triangleq & \mu\delta.[\beta](M\{N{\cdot}\gamma/\alpha\}) & \beta \neq \alpha \end{array}$$

We have the following rules of computation in $\lambda\mu$:

Definition 1.5 ($\lambda\mu$ REDUCTION) Parigot defines a number of reduction rules; two *computational rules*:

$$\begin{array}{rrcl} \textit{logical } (\beta): & (\lambda x.M)N & \rightarrow & M[N/x] \\ \textit{structural } (\mu): & (\mu\alpha.[\beta]M)N & \rightarrow & \mu\gamma.([\beta]M\{N{\cdot}\gamma/\alpha\}) \end{array}$$

as well as the *simplification rules*:

$$\begin{array}{rrcll} \textit{renaming}: & \mu\alpha[\beta](\mu\gamma.[\delta]M) & \rightarrow & \mu\alpha.[\delta]M[\beta/\gamma] & \\ \textit{erasing}: & \mu\alpha.[\alpha]M & \rightarrow & M & \textit{if } \alpha \textit{ does not occur in } M. \\ \eta\mu: & \mu\alpha.[\beta]M & \rightarrow & \lambda x\mu\gamma.[\beta]M\{x{\cdot}\gamma/\alpha\} & \end{array}$$

which are added mainly to simplify the presentation of his results[2].

Reduction on $\lambda\mu$-terms is defined as the compatible closure of these rules.

It is possible to define more reduction rules, but Parigot refrained from that since he aimed at defining a confluent reduction system.

The intuition behind the structural rule is given by de Groote [21]: *"in a $\lambda\mu$-term $\mu\alpha.M$ of type $A{\rightarrow}B$, only the subterms named by α are* really *of type $A{\rightarrow}B$ (...); hence, when such a μ-abstraction is applied to an argument, this argument must be passed over to the sub-terms named by α."* In this paper, we will only deal with the logical, structural and renaming rule; this is also the restriction made by de Groote in [21].

Type assignment for $\lambda\mu$ is defined by the following natural deduction system; there is a *main*, or *active*, conclusion, labelled by a term of this calculus, and the alternative conclusions are labelled by the set of Greek variables $\alpha,\beta,etc.$

Definition 1.6 (TYPING RULES FOR $\lambda\mu$) Our types are those of Definition 1.2, extended with the type constant $\bot$ that is essentially added to express negation, *i.e.*:

$$A,B \;::=\; \varphi \mid \bot \mid (A{\rightarrow}B) \quad (A \neq \bot)$$

[2] In fact, Parigot formulates the renaming rule as $[\beta](\mu\gamma.M) \rightarrow M[\beta/\gamma]$; since $[\beta](\mu\gamma.M)$ is not a term, we write the rule differently.

The type assignment rules are:

$$(Ax):\ \frac{}{\Gamma \vdash x : A \mid \Delta}\,(x{:}A \in \Gamma) \qquad (\mu):\ \frac{\Gamma \vdash M : B \mid \alpha{:}A,\beta{:}B,\Delta}{\Gamma \vdash \mu\alpha.[\beta]M : A \mid \beta{:}B,\Delta} \quad \frac{\Gamma \vdash M : A \mid \alpha{:}A,\Delta}{\Gamma \vdash \mu\alpha.[\alpha]M : A \mid \Delta}$$

$$(\to I):\ \frac{\Gamma, x{:}A \vdash M : B \mid \Delta}{\Gamma \vdash \lambda x.M : A \to B \mid \Delta} \qquad (\to E):\ \frac{\Gamma \vdash M : A \to B \mid \Delta \quad \Gamma \vdash N : A \mid \Delta}{\Gamma \vdash MN : B \mid \Delta}$$

We can think of $[\alpha]M$ as storing the type of M amongst the alternative conclusions by giving it the name α - the set of Greek variables is called the set of *context* variables (or *names*).

As an example illustrating the fact that this system is more powerful than the system for the λ-calculus, here is a proof of Peirce's Law (due to Ong and Steward [28]):

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x{:}(A\to B)\to A \vdash x : (A\to B)\to A \mid \alpha{:}A}\,(Ax) \quad \dfrac{\dfrac{\dfrac{}{x{:}(A\to B)\to A, y{:}A \vdash y : A \mid \alpha{:}A,\beta{:}B}\,(Ax)}{x{:}(A\to B)\to A, y{:}A \vdash \mu\beta.[\alpha]y : B \mid \alpha{:}A}\,(\mu)}{x{:}(A\to B)\to A \vdash \lambda y.\mu\beta.[\alpha]y : A\to B \mid \alpha{:}A}\,(\to I)}{x{:}(A\to B)\to A \vdash x(\lambda y.\mu\beta.[\alpha]y) : A \mid \alpha{:}A}\,(\to E)}{x{:}(A\to B)\to A \vdash \mu\alpha.[\alpha](x(\lambda y.\mu\beta.[\alpha]y)) : A \mid}\,(\mu)}{\vdash \lambda x.\mu\alpha.[\alpha](x(\lambda y.\mu\beta.[\alpha]y)) : ((A\to B)\to A)\to A \mid}\,(\to I)$$

Notice that $\perp$ plays no part in this proof. Indeed, we can define the set of types without $\perp$; the underlying logic of such a system then corresponds to *minimal classical logic* [1].

Since we allow $\perp$ as a proper type, we can even express negation (of course, it is also implicitly present in the right-hand type environment), and can give a derivation for $\neg\neg A\to A$, so can express *double negation elimination*; so in $\lambda\mu$ we can represent full Classical Logic.

$$\dfrac{\dfrac{\dfrac{}{y{:}(A\to\perp)\to\perp \vdash y : (A\to\perp)\to\perp \mid \beta{:}\perp}\,(Ax) \quad \dfrac{\dfrac{\dfrac{}{x{:}A \vdash x : A \mid \alpha{:}A,\delta{:}\perp}\,(Ax)}{x{:}A \vdash \mu\delta.[\alpha]x : \perp \mid \alpha{:}A}\,(\mu)}{\vdash \lambda x.\mu\delta.[\alpha]x : A\to\perp \mid \alpha{:}A}\,(\to I)}{\dfrac{y{:}(A\to\perp)\to\perp \vdash y(\lambda x.\mu\delta.[\alpha]x) : \perp \mid \alpha{:}A,\beta{:}\perp}{y{:}(A\to\perp)\to\perp \vdash \mu\alpha.[\beta]y(\lambda x.\mu\delta.[\alpha]x) : A \mid \beta{:}\perp}\,(\mu)}\,(\to E)}{\vdash \lambda y.\mu\alpha.[\beta]y(\lambda x.\mu\delta.[\alpha]x) : ((A\to\perp)\to\perp)\to A \mid \beta{:}\perp}\,(\to I)$$

Notice that this term is not closed, since β is free, albeit of type $\perp$.

De Groote [21] considers a variant of $\lambda\mu$ which separates the naming and μ-binding features[3]. This gives a considerable different system, that allows for $\neg\neg A\to A$ to be inhabited via (the closed term) $\lambda y.\mu\alpha.y(\lambda x.[\alpha]x)$. De Groote's variant of $\lambda\mu$ [21] uses the syntax

$$M,N ::= x \mid \lambda x.M \mid MN \mid \mu\alpha.M \mid [\beta]M$$

and splits rule (μ) into

$$(\mu):\ \frac{\Gamma \vdash M : \perp \mid \alpha{:}A,\Delta}{\Gamma \vdash \mu\alpha.M : A \mid \Delta} \qquad (\perp):\ \frac{\Gamma \vdash M : A \mid \beta{:}A,\Delta}{\Gamma \vdash [\beta]M : \perp \mid \beta{:}A,\Delta}$$

[3] Notice that then Parigot's renaming rule is correct. We could have presented our results for this more permissive system, but would have had to sacrifice soundness and completeness for the renaming rule. Notice that we would still have soundness and completeness for the two computational rules, which are arguably the most important.

In this system we can derive

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{y{:}(A\to\bot)\to\bot \vdash y : (A\to\bot)\to\bot \mid}\,(Ax) \qquad \dfrac{\dfrac{\dfrac{}{x{:}A \vdash x : A \mid \alpha{:}A}\,(Ax)}{x{:}A \vdash [\alpha]x : \bot \mid \alpha{:}A}\,(\bot)}{\vdash \lambda x.[\alpha]x : A\to\bot \mid \alpha{:}A}\,(\to I)}{y{:}(A\to\bot)\to\bot \vdash y(\lambda x.[\alpha]x) : \bot \mid \alpha{:}A}\,(\to E)}{y{:}(A\to\bot)\to\bot \vdash \mu\alpha.y(\lambda x.[\alpha]x) : A \mid}\,(\mu)}{\vdash \lambda y.\mu\alpha.y(\lambda x.[\alpha]x) : ((A\to\bot)\to\bot)\to A \mid}\,(\to I)
$$

For the moment, we will deal with Parigot's original system only; we aim to extend our results to de Groote's variant in future work.

2 The Strict Intersection Type Assignment System for the λ-calculus

The remainder of this paper will be dedicated a notion of intersection/union typing on $\lambda\mu$. This will be defined as a natural extension of the Strict Intersection System [2] for the λ-calculus. Before we come to that, we will briefly summarise the latter.

Definition 2.1 (STRICT TYPES) *i*) Let φ range over an infinite, enumerable set of type variables. The set $\mathcal{T}_s$ of *strict types*, ranged over by A, B, *etc* is defined through the grammar:

$$A, B \ ::= \ \varphi \mid \top\to B \mid (A_1\cap\cdots\cap A_n)\to B \quad (n\geq 1)$$

The set $\mathcal{T}$ of *intersection types* is defined as the union of $\{\top\}$ and the closure of $\mathcal{T}_s$ under intersection; we will use A, B, *etc* for intersection types as well, and mention which set they belong to when necessary.

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 *type-environment* Γ is a partial mapping from term variables to intersection types, and we write $x{:}A\in\Gamma$ if $\Gamma(x) = A$.

So if we write a type as $A\to B$, then $B\in\mathcal{T}_{\mathbf{S}}$, and $A\in\mathcal{T}$.

In the notation of types, as usual, right-most outer-most parentheses in arrow types will be omitted, and we assume $\cap$ to bind stronger than $\to$. From hereon, we will write $\underline{n}$ for the set $\{1,\ldots,n\}$.

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.

Definition 2.2 *i*) The relation '$\leq$' is defined as the least pre-order on $\mathcal{T}$ such that:

$$
\begin{array}{rcll}
A_1\cap\cdots\cap A_n & \leq & A_i, & \text{for all } i\in\underline{n}, n\geq 1 \\
B\leq A_i, \ \text{for all } i\in\underline{n} & \Rightarrow & B\leq A_1\cap\cdots\cap A_n, & n\geq 0
\end{array}
$$

ii) On $\mathcal{T}$, the relation '$\sim$' is defined by:

$$A\leq B\leq A \ \Rightarrow \ A\sim B \qquad\qquad A\sim B \wedge C\sim D \ \Leftrightarrow \ A\to C\sim B\to D$$

iii) The relations '$\leq$' and '$\sim$', are extended to contexts by: $\Gamma\leq\Gamma'$ if and only if for every $x{:}A'\in\Gamma'$ there is an $x{:}A\in\Gamma$ such that $A\leq A'$, and: $\Gamma\sim\Gamma' \Leftrightarrow \Gamma\leq\Gamma'\leq\Gamma$.

$\mathcal{T}$ will be considered modulo $\sim$; then $\leq$ becomes a partial order. It is easy to show that both $(A\cap B)\cap C \sim A\cap(B\cap C)$ and $A\cap B \sim B\cap A$, so the type constructor $\cap$ is associative and commutative, and we will write $\cap_n A_i$ for $A_1\cap\cdots\cap A_n$, and consider $\top$ to be the empty intersection: $\top = \cap_0 A_i$. Moreover, we will assume, unless stated explicitly otherwise, that in $\cap_n A_i$ each A_i is strict.

Definition 2.3 The *strict type assignment* is defined by the following natural deduction system (where all types mentioned are strict, with the exception of A in rule $(\to I)$ and $(\to E)$):

$$(\cap E):\ \frac{}{\Gamma,x{:}\cap_n A_i \vdash x : A_j}\ (n\geq 1, 1\leq j\leq n) \qquad (\cap I):\ \frac{\Gamma\vdash M : A_i \quad (\forall i\in\underline{n})}{\Gamma\vdash M:\cap_n A_i}\ (n\geq 0)$$

$$(\to I):\ \frac{\Gamma,x{:}A\vdash M:B}{\Gamma\vdash \lambda x.M : A\to B} \qquad (\to E):\ \frac{\Gamma\vdash M:A\to B \quad \Gamma\vdash N:A}{\Gamma\vdash MN:B}$$

We will write $\Gamma\vdash M:A$ for statements that are derived using these rules.

Notice that $\Gamma\vdash M:\top$ for all Γ, M by rule $(\cap I)$.

Properties of this system have been studied in [2].

3 Intersection and union type assignment for $\lambda\mu$

We will now define a notion of type assignment for $\lambda\mu$ that uses intersection and union types.

We see the context variables α as names for possible continuations that in the philosophy of intersection types need not all be typed with the same type; we therefore allow multiple types for a context variable in the environment Δ, grouped using a new type constructor, which we call union.

Binding a context variable then generates a *context switch* $\mu\alpha.[\beta]M$, which naturally has a union type $\cup_n A_i$; reduction of the term $(\mu\alpha.[\beta]M)N$ will bring the operand N to each of the pseudo subterms in M of the shape $[\alpha]Q$ ('named' with α), where Q has type A_i; since N gets placed behind Q, this implies that $A_i = C_i\to B_i$ and that therefore the type for α should be $\cup_n(C_i\to B_i)$; this then also implies that N should have all the types C_i $(\forall i\in\underline{n})$; rule $(\to E)$ as below expresses exactly that. The only 'functionality' we need for union types therefore is the ability to choose a collection of types for α amongst those stored in Δ; this is represented by rule $(\cup E)$.

Definition 3.1 (THE SYSTEM $\vdash^{\cap\cup}_{\lambda\mu}$) *i*) The set of strict types we consider for the intersection-union type assignment is:

$$\begin{array}{rcll} A,B & ::= & \varphi \mid B_1\cup\cdots\cup B_m \mid \sigma\to B & (n,m\geq 0)\\ \sigma,\tau & ::= & A_1\cap\cdots\cap A_n & (n\geq 0)\end{array}$$

As above, we call $A_1\cap\cdots\cap A_n$ (with $n\geq 0$) an intersection type, and call $B_1\cup\cdots\cup B_m$ (with $m\geq 0$) a *union* type; we use $\top$ for the empty intersection type, and $\bot$ for the empty union type.

ii) The relation $\leq$ of Definition 2.2 is extended to intersection-union types by:

$$\begin{array}{rcll} A_1\cap\cdots\cap A_n & \leq & A_i, & \text{for all } i\in\underline{n}, n\geq 1\\ B\leq A_i,\ \text{for all } i\in\underline{n} & \Rightarrow & B\leq A_1\cap\cdots\cap A_n, & n\geq 0\\ B_j & \leq & B_1\cup\cdots\cup B_m, & \text{for all } j\in\underline{m}, m\geq 1\\ B_j\leq A,\ \text{for all } j\in\underline{m} & \Rightarrow & B_1\cup\cdots\cup B_m\leq A, & m\geq 0\end{array}$$

On $\mathcal{T}$, the relation '$\sim$' is defined by the same way as in Definition 2.2.

iii) A *left environment* Γ is a partial mapping from term variables to intersections of strict types, and we write $x{:}A \in \Gamma$ if $\Gamma(x) = A$. Similarly, a *right environment* Δ contains only strict types, which can be union types.

iv) The relations '$\leq$' and '$\sim$', are extended to left and right environments by: $\Gamma \leq \Gamma'$ if and only if for every $x{:}\sigma' \in \Gamma'$ there is an $x{:}\sigma \in \Gamma$ such that $\sigma \leq \sigma'$, and $\Gamma \sim \Gamma' \iff \Gamma \leq \Gamma' \leq \Gamma$, and $\Delta \leq \Delta'$ if for every $\alpha{:}A \in \Delta$ there exists $\alpha{:}A' \in \Delta'$ such that $A \leq A'$, and $\Delta \sim \Delta' \iff \Delta \leq \Delta' \leq \Delta$.

Notice that we consider union types to be strict as well; this implies that we allow an intersection of union types, a union of union types, but not a union of intersection types.

Definition 3.2 (The system $\vdash^{\cap\cup}_{\lambda\mu}$) Intersection-union type assignment for $\lambda\mu$ is defined via:

$$(\cap E): \frac{}{\Gamma, x{:}\cap_n A_i \vdash x : A_i \mid \Delta} \qquad (\cap I): \frac{\Gamma \vdash M : A_i \mid \Delta \quad (\forall i \in \underline{n})}{\Gamma \vdash M : \cap_n A_i \mid \Delta}\,(n \geq 0, n \neq 1)$$

$$(\to I): \frac{\Gamma, x{:}\sigma \vdash M : B \mid \Delta}{\Gamma \vdash \lambda x.M : \sigma \to B \mid \Delta} \qquad (\to E): \frac{\Gamma \vdash M : \cup_n(\sigma_i \to B_i) \mid \Delta \quad \Gamma \vdash N : \sigma_i \mid \Delta \ (\forall i \in \underline{n})}{\Gamma \vdash MN : \cup_n B_i \mid \Delta}\,(n \geq 1)$$

$$(\cup E): \frac{\Gamma \vdash M : \cup_m B_j \mid \beta{:}\cup_n A_i, \alpha{:}B, \Delta}{\Gamma \vdash \mu\alpha.[\beta]M : B \mid \beta{:}\cup_n A_i, \Delta}\,(\cup_m B_j \leq \cup_n A_i) \qquad \frac{\Gamma \vdash M : \cup_m B_j \mid \beta{:}\cup_n A_i, \Delta}{\Gamma \vdash \mu\beta.[\beta]M : \cup_n A_i \mid \Delta}\,(\cup_m B_j \leq \cup_n A_i)$$

We write $\Gamma \vdash^{\cap\cup}_{\lambda\mu} M : A \mid \Delta$ if this statement is derivable using these rules.

We will normally not distinguish between the two variants of $(\cup E)$.

Notice that the traditional $(\to E)$ of Definition 2.3 is obtained by taking $n = 1$. Moreover, all σ_i can be intersection types, so each can be $\top$; this is why that rule is not formulated using $\Gamma \vdash^{\cap\cup}_{\lambda\mu} N : \cap_n \sigma_i \mid \Delta$. If $x{:}\cup_m B_j \in \Gamma$, then we can only derive $\Gamma \vdash^{\cap\cup}_{\lambda\mu} x : \cup_m B_j \mid \Delta$, *i.e.* we have no way of eliminating a union assigned to a term variable. Moreover, we have no traditional rules $(\cup I)$ and $(\cup E)$ on terms, which would be formulated (as in [9]), via

$$(\cup I): \frac{\Gamma \vdash M : A \mid \Delta}{\Gamma \vdash M : A \cup B \mid \Delta} \qquad (\cup E): \frac{\Gamma \vdash N : A \cup B \mid \Delta \quad \Gamma, x{:}A \vdash M : C \mid \Delta \quad \Gamma, x{:}B \vdash M : C \mid \Delta}{\Gamma \vdash M\{N/x\} : C \mid \Delta}$$

These create the subject-reduction problem dealt with in that paper by limiting to parallel reduction.

Notice that both the strict system for the λ-calculus and the system for $\lambda\mu$ are true subsystems; the first by not allowing union types, or alternative conclusions, the second by limiting to Curry types.

Lemma 3.3 (Generation lemma) • *If $\Gamma \vdash^{\cap\cup}_{\lambda\mu} x : A \mid \Delta$, then there exists $x{:}\sigma \in \Gamma$ such that $\sigma \leq A$.*

- *If $\Gamma \vdash^{\cap\cup}_{\lambda\mu} \lambda x.M : A \mid \Delta$, then there exists σ_i, C_i $(\forall i \in \underline{n})$ such that $A = \cap_n(\sigma_i \to C_i)$, and, for all $i \in \underline{n}$, $\Gamma, x{:}\sigma_i \vdash^{\cap\cup}_{\lambda\mu} M : C_i \mid \Delta$.*
- *If $\Gamma \vdash^{\cap\cup}_{\lambda\mu} MN : A \mid \Delta$, then $A = \cup_n A_i$, and for every $i \in \underline{n}$ there exists $\sigma_i \in \mathcal{T}$ such that $\Gamma \vdash^{\cap\cup}_{\lambda\mu} M : \cup_n(\sigma_i \to A_i) \mid \Delta$ and $\Gamma \vdash^{\cap\cup}_{\lambda\mu} N : \sigma_i \mid \Delta$.*
- *If $\Gamma \vdash^{\cap\cup}_{\lambda\mu} \mu\alpha.[\beta]M : A \mid \Delta$, then there are A_i $(\forall i \in \underline{n})$ such that $A = \cap_n A_i$, and, for every $i \in \underline{n}$, there are m_i, m'_i with $m'_i \leq m_i$ and B^i_j $(\forall j \in \underline{m_i})$ such that $\Gamma \vdash^{\cap\cup}_{\lambda\mu} M : \cup_{m'_i} B^i_k \mid \beta{:}\cup_{m_i} B^i_j, \alpha{:}A_i, \Delta$.*

Proof. By easy induction. $\square$

The system $\vdash^{\cap\cup}_{\lambda\mu}$ does not have *choice*, *i.e.* we cannot show that, if $\Gamma \vdash^{\cap\cup}_{\lambda\mu} M : A \cup B \mid \Delta$, then either $\Gamma \vdash^{\cap\cup}_{\lambda\mu} M : A \mid \Delta$ of $\Gamma \vdash^{\cap\cup}_{\lambda\mu} M : B \mid \Delta$ as would hold in an intuitionistic system. Take:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x{:}A \vdash x : A \mid \beta{:}B,\delta{:}A \cup (A \to B)}\,(\cap E)}{x{:}A \vdash \mu\beta.[\delta]x : B \mid \delta{:}A \cup (A \to B)}\,(\cup E)}{\vdash \lambda x.\mu\beta.[\delta]x : A \to B \mid \delta{:}A \cup (A \to B)}\,(\to R)}{\vdash \mu\delta.[\delta](\lambda x.\mu\beta.[\delta]x) : A \cup (A \to B) \mid}\,(\cup E)$$

Notice that we cannot derive $\vdash^{\cap\cup}_{\lambda\mu} \mu\delta.[\delta](\lambda x.\mu\beta.[\delta]x) : A \mid$, nor $\vdash^{\cap\cup}_{\lambda\mu} \mu\delta.[\delta](\lambda x.\mu\beta.[\delta]x) : A \to B \mid$, since the two occurrences of $[\delta]$ need to be typed differently, but with related types. This is comparable to both A and $A \to B$ to be needed as assumption for x to type $\lambda x.xx$.

We can show that a general $(\cap E)$ (for all terms) is admissible.

Lemma 3.4 *If* $\Gamma \vdash^{\cap\cup}_{\lambda\mu} M : \cap_n A_i \mid \Delta$, *then* $\Gamma \vdash^{\cap\cup}_{\lambda\mu} M : A_i \mid \Delta$, *for all* $i \in \underline{n}$.

Proof. Easy. □

The following result is standard.

Lemma 3.5 (Thinning & Weakening) *i) Let* $\Gamma \vdash^{\cap\cup}_{\lambda\mu} M : A \mid \Delta$; *take* $\Gamma' = \{ x{:}\sigma \in \Gamma \mid x \in fv(M) \}$ *and* $\Delta' = \{ \alpha{:}\sigma \in \Delta \mid \alpha \in fv(M) \}$, *then* $\Gamma' \vdash^{\cap\cup}_{\lambda\mu} M : A \mid \Delta'$.
ii) Let $\Gamma \vdash^{\cap\cup}_{\lambda\mu} M : A \mid \Delta$, *and* $\Gamma' \leq \Gamma$ *and* $\Delta \leq \Delta'$, *then* $\Gamma' \vdash^{\cap\cup}_{\lambda\mu} M : A \mid \Delta'$.

Proof. By easy induction. □

As a consequence, the following rules are admissible:

$$(Th): \ \dfrac{\Gamma \vdash M : A \mid \Delta}{\{ x{:}\sigma \in \Gamma \mid x \in fv(M) \} \vdash M : A \mid \{ \alpha{:}\sigma \in \Delta \mid \alpha \in fv(M) \}}$$

$$(Wk): \ \dfrac{\Gamma \vdash M : A \mid \Delta}{\Gamma' \vdash M : A \mid \Delta'}\ (\Gamma' \leq \Gamma, \Delta \leq \Delta')$$

4 Subject reduction and expansion

We will now show our main results, by showing that our notion of type assignment is sound and complete. We start by showing two variants of the substitution lemma.

Lemma 4.1 (Term substitution lemma) *Let* A *be strict;* $\Gamma \vdash^{\cap\cup}_{\lambda\mu} M\{N/x\} : A \mid \Delta$ *if and only if there exists* $\sigma \in \mathcal{T}$ *such that* $\Gamma, x{:}\sigma \vdash^{\cap\cup}_{\lambda\mu} M : A \mid \Delta$ *and* $\Gamma \vdash^{\cap\cup}_{\lambda\mu} N : \sigma \mid \Delta$.

Proof. By induction on the structure of terms.

$(M \equiv x)$: $(\Rightarrow)$: If $\Gamma \vdash^{\cap\cup}_{\lambda\mu} x\{N/x\} : A$, then $\Gamma, x{:}A \vdash^{\cap\cup}_{\lambda\mu} x : A$ and $\Gamma \vdash^{\cap\cup}_{\lambda\mu} N : A$.

$(\Leftarrow)$: If $\Gamma \vdash^{\cap\cup}_{\lambda\mu} x : A \mid \Delta$, then there exists A_i $(\forall i \in \underline{n})$ such that $A = A_k$ from some $k \in \underline{n}$, and $\Gamma = \Gamma', x{:}\cap_n A_i$, so $\Gamma', x{:}\cap_n A_i \vdash^{\cap\cup}_{\lambda\mu} x : A_k \mid \Delta$. From $\Gamma \vdash^{\cap\cup}_{\lambda\mu} N : \cap_n A_i \mid \Delta$ and Lemma 3.4, we have $\Gamma \vdash^{\cap\cup}_{\lambda\mu} N : A \mid \Delta$, so $\Gamma \vdash^{\cap\cup}_{\lambda\mu} x\{N/x\} : A \mid \Delta$.

$(M \equiv y \neq x)$: $(\Rightarrow)$: By Lemma 3.5, since $y\{N/x\} \equiv y$, and $x \notin fv(y)$.

$(\Leftarrow)$: Take $C = \top$; by Lemma 3.5, $\Gamma, x{:}\top \vdash^{\cap\cup}_{\lambda\mu} y : A \mid \Delta$.

$(M' = PQ)$: Let $A = \cup_r A_j$, with $r \geq 1$. Notice that $(PQ)\{N/x\} = P\{N/x\}Q\{N/x\}$.

$(\Rightarrow)$: Then, by Lemma 3.3, there are $\tau_j \in \mathcal{T}$ $(\forall j \in \underline{r})$ such that $\Gamma \vdash^{\cap\cup}_{\lambda\mu} P[N/x] : \cup_r(\tau_j \to A_j) \mid \Delta$ and $\Gamma \vdash^{\cap\cup}_{\lambda\mu} Q[N/x] : \tau_j \mid \Delta$, for all $j \in \underline{r}$. Then by induction, there are $\sigma_1, \sigma_2^1, \ldots, \sigma_2^r$ such that:

- $\Gamma, x{:}\sigma_1 \vdash^{\cap\cup}_{\lambda\mu} P : \cup_r(\tau_j \to A_j) \mid \Delta$ and $\Gamma \vdash^{\cap\cup}_{\lambda\mu} N : \sigma_1 \mid \Delta$, as well as
- $\Gamma, x{:}\sigma_2^j \vdash^{\cap\cup}_{\lambda\mu} Q : \tau_j \mid \Delta$ and $\Gamma \vdash^{\cap\cup}_{\lambda\mu} N : \sigma_2^j \mid \Delta$, for all $j \in \underline{r}$.

Take $\sigma = \sigma_1 \cap \sigma_2^1 \cap \cdots \cap \sigma_2^r$; then by weakening and $(\to E)$, we get $\Gamma, x{:}\sigma \vdash^{\cap\cup}_{\lambda\mu} PQ : A \mid \Delta$; notice that $\Gamma \vdash^{\cap\cup}_{\lambda\mu} N : \sigma \mid \Delta$ by $(\cap I)$.

$(\Leftarrow)$: If $\Gamma, x{:}\sigma \vdash^{\cap\cup}_{\lambda\mu} PQ : \cup_r A_j \mid \Delta$, then by Lemma 3.3 there exists $\tau_j \in \mathcal{T}$ $(j \in \underline{r})$ such that $\Gamma, x{:}\sigma \vdash^{\cap\cup}_{\lambda\mu} P : \cup_r(\tau_j \to A_j) \mid \Delta$ and $\Gamma, x{:}\sigma \vdash^{\cap\cup}_{\lambda\mu} Q : \tau_j \mid \Delta$, for $j \in \underline{r}$. Then, by induction, we have both $\Gamma \vdash^{\cap\cup}_{\lambda\mu} P[N/x] : \cup_r(\tau_j \to A_j) \mid \Delta$ and $\Gamma \vdash^{\cap\cup}_{\lambda\mu} Q[N/x] : \tau_j \mid \Delta$ for all $j \in \underline{r}$; the result follows by $(\to E)$.

$(M \equiv \lambda y.M'$; $M \equiv \mu\alpha.[\beta]M')$: By induction.

Because of Lemma 3.4, we can extend the above results also to the case that A is an intersection type; notice that this is implicitly used in the third case, where D_j can be an intersection type.

Dually, we have:

Lemma 4.2 (Structural substitution lemma) *$\Gamma \vdash^{\cap\cup}_{\lambda\mu} M\{N\cdot\gamma/\alpha\} : C \mid \gamma{:}\cup_n B_i, \Delta$ if and only if there are σ_i $(\forall i \in \underline{n})$ such that, for all $i \in \underline{n}$, $\Gamma \vdash^{\cap\cup}_{\lambda\mu} N : \sigma_i \mid \Delta$, and $\Gamma \vdash^{\cap\cup}_{\lambda\mu} M : C \mid \alpha{:}\cup_n(\sigma_i \to B_i), \Delta$.*

Proof. We only show the interesting cases.

$(M = x)$: Then $x\{N\cdot\gamma/\alpha\} = x$; as above the result follows, in either direction, by thinning and weakening.

$(M = \lambda x.M')$: By induction.

$(M = PQ)$: Then $PQ\{N\cdot\gamma/\alpha\} = P\{N\cdot\gamma/\alpha\}\ Q\{N\cdot\gamma/\alpha\}$; assume C is strict, $C = \cup_r C_j$, with $r \geq 1$.

$(\Rightarrow)$: Then, by Lemma 3.3, there exists $\tau_j \in \mathcal{T}$ $(\forall j \in \underline{r})$ such that $\Gamma \vdash^{\cap\cup}_{\lambda\mu} P\{N\cdot\gamma/\alpha\} : \cup_r(\tau_j \to C_j) \mid \gamma{:}\cup_n B_i, \Delta$ and $\Gamma \vdash^{\cap\cup}_{\lambda\mu} Q\{N\cdot\gamma/\alpha\} : \tau_j \mid \gamma{:}\cup_n B_i, \Delta$, for $j \in \underline{r}$. Then by induction, there are σ_i $(\forall i \in \underline{k})$ and σ_i' $(\forall i \in \underline{l})$ with $k + l = m$ such that:

- $\Gamma \vdash^{\cap\cup}_{\lambda\mu} P : \cup_r(\tau_j \to C_j) \mid \alpha{:}\cup_k(\sigma_i \to B_i), \Delta$ and, for all $i \in \underline{k}$, $\Gamma \vdash^{\cap\cup}_{\lambda\mu} N : \sigma_i \mid \Delta$, as well as
- $\Gamma \vdash^{\cap\cup}_{\lambda\mu} Q : \tau_j \mid \alpha{:}\cup_l(\sigma_i' \to B_i), \Delta$ for all $j \in \underline{r}$, and, for all $i \in \underline{l}$, $\Gamma \vdash^{\cap\cup}_{\lambda\mu} N : \sigma_i' \mid \Delta$.

Then by weakening and $(\to E)$, we get $\Gamma \vdash^{\cap\cup}_{\lambda\mu} PQ : \cup_r C_j \mid \alpha{:}\cup_k(\sigma_i \to B_i) \cup \cup_l(\sigma_i' \to B_i), \Delta$; notice that $\Gamma \vdash^{\cap\cup}_{\lambda\mu} N : \rho \mid \Delta$ for all $\rho \in \{\sigma_i\ (\forall i \in \underline{k}),\ \sigma_i'\ (\forall i \in \underline{l})\}$.

$(\Leftarrow)$: If $\Gamma \vdash^{\cap\cup}_{\lambda\mu} PQ : \cup_r C_j \mid \alpha{:}\cup_n(\sigma_i \to B_i), \Delta$, then there are $\tau_j \in \mathcal{T}$ $(\forall j \in \underline{r})$ such that $\Gamma \vdash^{\cap\cup}_{\lambda\mu} P : \cup_r(\tau_j \to C_j) \mid \alpha{:}\cup_n(\sigma_i \to B_i), \Delta$ and $\Gamma \vdash^{\cap\cup}_{\lambda\mu} Q : \tau_j \mid \alpha{:}\cup_n(\sigma_i \to B_i), \Delta$, for $j \in \underline{r}$. Then, by induction, $\Gamma \vdash^{\cap\cup}_{\lambda\mu} P\{N\cdot\gamma/\alpha\} : \cup_r(\tau_j \to C_j) \mid \gamma{:}\cup_n B_i, \Delta$ and $\Gamma \vdash^{\cap\cup}_{\lambda\mu} Q\{N\cdot\gamma/\alpha\} : \tau_j \mid \gamma{:}\cup_n B_i, \Delta$ for all $j \in \underline{r}$; the result follows by $(\to E)$.

$(M = \mu\beta.[\alpha]M')$: $(\Rightarrow)$: Notice that $\mu\beta.[\alpha]M'\{N\cdot\gamma/\alpha\} = \mu\beta.[\gamma](M'\{N\cdot\gamma/\alpha\}N)$ by definition. By Lemma 3.3, from $\Gamma \vdash^{\cap\cup}_{\lambda\mu} \mu\beta.[\gamma](M'\{N\cdot\gamma/\alpha\}N) : C \mid \gamma{:}\cup_n B_i, \Delta$, there are $r \leq n$ and ρ_l, D_l $(\forall l \in \underline{r})$ such that, without loss of generality, $\cup_r D_l \subseteq \cup_n B_i$, and the derivation is shaped like (notice that we can assume $\gamma, \beta \notin fv(N)$):

$$\dfrac{\dfrac{\dfrac{\boxed{}}{\Gamma \vdash M'\{N\cdot\gamma/\alpha\} : \cup_r(\rho_l \to D_l) \mid \gamma{:}\cup_n B_i, \beta{:}C, \Delta} \quad \dfrac{\dfrac{\boxed{}}{\Gamma \vdash N : \rho_l \mid \Delta}}{\Gamma \vdash N : \rho_l \mid \gamma{:}\cup_n B_i, \beta{:}C, \Delta}\,(Wk) \quad (\forall l \in \underline{r})}{\Gamma \vdash M'\{N\cdot\gamma/\alpha\}N : \cup_r D_l \mid \gamma{:}\cup_n B_i, \beta{:}C, \Delta}\,(\to E)}{\Gamma \vdash \mu\beta.[\gamma](M'\{N\cdot\gamma/\alpha\}N) : C \mid \gamma{:}\cup_n B_i, \Delta}\,(\cup E)$$

Then, by induction, there exist σ_i $(\forall i\in\underline{n})$ such that $\Gamma \vdash^{\cap\cup}_{\lambda\mu} M' : \cup_r(\rho_l\to D_l) \mid \alpha{:}\cup_n(\sigma_i\to B_i)$, $\beta{:}C,\Delta$ and, for all $i\in\underline{n}$, $\Gamma\vdash^{\cap\cup}_{\lambda\mu} N:\sigma_i \mid \Delta$, and we can construct:

$$\dfrac{\dfrac{\Gamma\vdash M' : \cup_r(\rho_l\to D_l)\mid \alpha{:}\cup_{n-r}(\sigma_i\to B_i),\beta{:}C,\Delta}{\Gamma\vdash M' : \cup_r(\rho_l\to D_l)\mid \alpha{:}\cup_r(\rho_l\to D_l)\cup\cup_{n-r}(\sigma_i\to B_i),\beta{:}C,\Delta}\,(Wk)}{\Gamma\vdash \mu\beta.[\alpha]M' : C\mid \alpha{:}\cup_r(\rho_l\to D_l)\cup\cup_{n-r}(\sigma_i\to B_i),\Delta}\,(\cup E)$$

Notice that also $\Gamma\vdash^{\cap\cup}_{\lambda\mu} N:\tau\mid\Delta'$ for every $\tau\in\{\rho_1,\ldots,\rho_r,\sigma_1,\ldots,\sigma_n\}$.

($\Leftarrow$): If $\Gamma\vdash^{\cap\cup}_{\lambda\mu} \mu\beta.[\alpha]M' : C\mid\alpha{:}\cup_n(\sigma_i\to B_i),\Delta$ and $\Gamma\vdash^{\cap\cup}_{\lambda\mu} N:\sigma_i\mid\Delta'$ for every $i\in\underline{n}$, then, by Lemma 3.3, the first derivation is constructed as follows:

$$\dfrac{\Gamma\vdash M' : \cup_r(\sigma_i\to B_l)\mid\alpha{:}\cup_n(\sigma_i\to B_i),\beta{:}C,\Delta}{\Gamma\vdash\mu\beta.[\alpha]M' : C\mid\alpha{:}\cup_n(\sigma_i\to B_i),\Delta}\,(\cup E)$$

for some $r\leq n$. Then, by induction, $\Gamma\vdash^{\cap\cup}_{\lambda\mu} M'\{N\cdot\gamma/\alpha\} : \cup_r(\sigma_i\to B_i)\mid\gamma{:}\cup_n B_i,\beta{:}C,\Delta$, and we can construct:

$$\dfrac{\dfrac{\Gamma\vdash M'\{N\cdot\gamma/\alpha\} : \cup_r(\sigma_i\to B_i)\mid\gamma{:}\cup_n B_i,\beta{:}C,\Delta \quad \Gamma\vdash N:\sigma_i\mid\Delta\ (\forall i\in\underline{r})}{\Gamma\vdash(M'\{N\cdot\gamma/\alpha\})N : \cup_r B_i\mid\gamma{:}\cup_n B_i,\beta{:}C,\Delta}\,(\to E)}{\Gamma\vdash\mu\beta.[\gamma]M'\{N\cdot\gamma/\alpha\}N : C\mid\gamma{:}\cup_n B_i,\Delta}\,(\cup E)$$

□

Using these two lemmas, we can prove the two main results of this paper:

Theorem 4.3 (Subject expansion) *If $P\to_{\lambda\mu} Q$, and $\Gamma\vdash^{\cap\cup}_{\lambda\mu} Q:A\mid\Delta$ (A strict), then $\Gamma\vdash^{\cap\cup}_{\lambda\mu} P:A\mid\Delta$.*

Proof. By induction on the definition of reduction, where we focus on the reduction rules.

($(\lambda x.M)N\to M[N/x]$): If $\Gamma\vdash^{\cap\cup}_{\lambda\mu} M\{N/x\} : A\mid\Delta$, then by Lemma 4.1 there exists a $\sigma\in\mathcal{T}$ such that $\Gamma,x{:}\sigma\vdash_{\lambda\mu} M:A\mid\Delta$ and $\Gamma\vdash^{\cap\cup}_{\lambda\mu} N:\sigma\mid\Delta$; then, by applying rule $(\to I)$ to the first result we get $\Gamma\vdash^{\cap\cup}_{\lambda\mu}\lambda x.M:\sigma\to A\mid\Delta$ and then by $(\to E)$ we get $\Gamma\vdash^{\cap\cup}_{\lambda\mu}(\lambda x.M)N:A\mid\Delta$.

($(\mu\alpha.[\alpha]M)N\to\mu\gamma.[\gamma]M\{N\cdot\gamma/\alpha\}N$): If $\Gamma\vdash^{\cap\cup}_{\lambda\mu}\mu\gamma.[\gamma]M\{N\cdot\gamma/\alpha\}N : A\mid\Delta$, then $A=\cup_n A_i$, and by Lemma 3.3, (wlog) there is $m\leq n$ such that $\Gamma\vdash^{\cap\cup}_{\lambda\mu} M\{N\cdot\gamma/\alpha\}N : \cup_m A_j\mid\gamma{:}\cup_n A_i,\Delta$, and there are τ_j $(\forall j\in\underline{m})$ such that, $\Gamma\vdash^{\cap\cup}_{\lambda\mu} M\{N\cdot\gamma/\alpha\} : \cup_m(\tau_j\to A_j)\mid\gamma{:}\cup_n A_i,\Delta$ and for all $j\in\underline{m}$, $\Gamma\vdash^{\cap\cup}_{\lambda\mu} N:\tau_j\mid\Delta$.

$$\dfrac{\dfrac{\Gamma\vdash M\{N\cdot\gamma/\alpha\} : \cup_m(\tau_j\to A_j)\mid\gamma{:}\cup_n A_i,\Delta \quad \Gamma\vdash N:\tau_j\mid\Delta\ (j\in\underline{m})}{\Gamma\vdash M\{N\cdot\gamma/\alpha\}N : \cup_m A_j\mid\gamma{:}\cup_n A_i,\Delta}\,(m\leq n)}{\Gamma\vdash\mu\gamma.[\gamma]M\{N\cdot\gamma/\alpha\}N : \cup_n A_i\mid\Delta}$$

Then, by Lemma 4.2, there are σ_i $(\forall i\in\underline{n})$ such that for all $i\in\underline{n}$, $\Gamma\vdash^{\cap\cup}_{\lambda\mu} N:\sigma_i\mid\Delta$, and $\Gamma\vdash^{\cap\cup}_{\lambda\mu} M : \cup_m(\tau_j\to A_j)\mid\alpha{:}\cup_n(\sigma_i\to A_i),\Delta$; take $B=\cup_m(\tau_j\to A_j)\cup\cup_n(\sigma_i\to A_i)$, then by weakening, we have $\Gamma\vdash^{\cap\cup}_{\lambda\mu} M : \cup_m(\tau_j\to A_j)\mid\alpha{:}B,\Delta$. Then, by rule $(\cup E)$, $\Gamma\vdash^{\cap\cup}_{\lambda\mu}\mu\alpha.[\alpha]M : B\mid\Delta$, and $\Gamma\vdash^{\cap\cup}_{\lambda\mu}(\mu\alpha.[\alpha]M)N : \cup_n A_i\mid\Delta$ then follows by rule $(\to E)$.

$$\dfrac{\dfrac{\Gamma\vdash M : \cup_m(\tau_j\to A_j)\mid\alpha{:}\cup_m(\tau_j\to A_j)\cup\cup_n(\sigma_i\to A_i),\Delta}{\Gamma\vdash\mu\alpha.[\alpha]M : \cup_m(\tau_j\to A_j)\cup\cup_n(\sigma_i\to A_i)\mid\Delta}\,(\cup E) \quad \Gamma\vdash N:\tau_i\mid\Delta\ (j\in\underline{m}) \quad \begin{array}{c}\Gamma\vdash N:\sigma_i\mid\Delta\ (i\in\underline{n})\\ \vdots\end{array}}{\Gamma\vdash(\mu\alpha.[\alpha]M)N : \cup_n A_i\mid\Delta}\,(\to E)$$

$((\mu\alpha.[\beta]M)N \to \mu\gamma.[\beta]M\{N\cdot\gamma/\alpha\})$: If $\Gamma \vdash^{\cap\cup}_{\lambda\mu} (\mu\alpha.[\beta]M)N : A \mid \Delta$, then $A = \cup_n A_i$, and by $(\to E)$ there are $\sigma_i\,(\forall i\in\underline{n})$ such that $\Gamma \vdash^{\cap\cup}_{\lambda\mu} \mu\alpha.[\beta]M : \cup_n(\sigma_i\to A_i) \mid \Delta$, and $\Gamma \vdash^{\cap\cup}_{\lambda\mu} N : \sigma_i \mid \Delta$ for all $i\in\underline{n}$; then by Lemma 4.2, $\Gamma \vdash^{\cap\cup}_{\lambda\mu} \mu\gamma.[\beta]M\{N\cdot\gamma/\alpha\} : A \mid \Delta$.

$(\mu\alpha.[\beta]\mu\gamma.[\delta]M \to \mu\alpha.[\delta](M[\beta/\gamma]))$: If $\Gamma \vdash^{\cap\cup}_{\lambda\mu} \mu\alpha.[\delta](M[\beta/\gamma]) : A \mid \Delta$, then by rule $(\cup E)$, there exist $\delta{:}\cup_n D_i \in \Delta$ and $m \leq n$ such that $\Gamma \vdash^{\cap\cup}_{\lambda\mu} M[\beta/\gamma] : \cup_m D_j \mid \alpha{:}A,\Delta$. Let $\Delta = \delta{:}\cup_n D_i, \beta{:}\cup_k B_i, \Delta'$. Since M can contain β as well, this means that there are $C_j\,(\forall j\in\underline{k}), E_i\,(\forall i\in\underline{l})$ with $\cup_m C_j \cup \cup_{\underline{k}} E_i = \cup_k B_i$, and we can construct:

$$\dfrac{\dfrac{\Gamma \vdash M : D_k \mid \gamma{:}\cup_m C_j, \delta{:}\cup_n D_i, \beta{:}\cup_{\underline{k}} E_i, \alpha{:}A, \Delta'}{\Gamma \vdash \mu\gamma.[\delta]M : \cup_m C_j \mid \delta{:}\cup_n D_i, \beta{:}\cup_{\underline{k}} E_i, \alpha{:}A, \Delta'}\,(\cup E)}{\Gamma \vdash \mu\alpha.[\beta]\mu\gamma.[\delta]M : A \mid \Delta}\,(\cup E)$$

□

Theorem 4.4 (Subject reduction) *If $P \to_{\lambda\mu} Q$, and $\Gamma \vdash_{\lambda\mu} P : A \mid \Delta$, where$A$ is not an intersection, then $\Gamma \vdash_{\lambda\mu} Q : A \mid \Delta$*

Proof. $((\lambda x.M)N \to M[N/x])$: Let $\Gamma \vdash_{\lambda\mu} (\lambda x.M)N : A \mid \Delta$. Then by Lemma 3.3 there exists $\sigma \in \mathcal{T}$ such that $\Gamma \vdash_{\lambda\mu} \lambda x.M : \sigma\to A \mid \Delta$ and $\Gamma \vdash_{\lambda\mu} N : \sigma \mid \Delta$, and also $\Gamma, x{:}\sigma \vdash_{\lambda\mu} M : A \mid \Delta$. Then, by Lemma 4.1, we have $\Gamma \vdash_{\lambda\mu} M\{N/x\} : A \mid \Delta$.

$((\mu\alpha.[\alpha]M)N \to \mu\gamma.[\gamma]M\{N\cdot\gamma/\alpha\}N)$: If $\Gamma \vdash^{\cap\cup}_{\lambda\mu} (\mu\alpha.[\alpha]M)N : A \mid \Delta$, then by Lemma 3.3 there exist $A_i\,(\forall i\in\underline{n})$ and $\sigma_i\,(\forall i\in\underline{n})$ such that $A = \cup_n A_i$, and $\Gamma \vdash^{\cap\cup}_{\lambda\mu} \mu\alpha.[\alpha]M : \cup_n(\sigma_i\to A_i) \mid \Delta$ and, for all $i\in\underline{n}$, $\Gamma \vdash^{\cap\cup}_{\lambda\mu} N : \sigma_i \mid \Delta$; then (wlog) also $\Gamma \vdash^{\cap\cup}_{\lambda\mu} M : \cup_m(\sigma_j\to A_j) \mid \alpha{:}\cup_n(\sigma_i\to A_i),\Delta$, with $m \leq n$.

$$\dfrac{\dfrac{\Gamma \vdash M : \cup_m(\sigma_j\to A_j) \mid \alpha{:}\cup_n(\sigma_i\to A_i),\Delta}{\Gamma \vdash \mu\alpha.[\alpha]M : \cup_n(\sigma_i\to A_i) \mid \Delta}\,(\cup E) \qquad \Gamma \vdash N : \sigma_i \mid \Delta \;\; (i\in\underline{n})}{\Gamma \vdash (\mu\alpha.[\alpha]M)N : \cup_n A_i \mid \Delta}\,(\to E)$$

Then, by Lemma 4.2, $\Gamma \vdash^{\cap\cup}_{\lambda\mu} M\{N\cdot\gamma/\alpha\} : \cup_m(\sigma_j\to A_j) \mid \gamma{:}\cup_n A_i,\Delta$, and, by $(\to E)$, $\Gamma \vdash^{\cap\cup}_{\lambda\mu} M\{N\cdot\gamma/\alpha\}N : \cup_m(A_i) \mid \gamma{:}\cup_n A_i,\Delta$, so, by rule $(\cup E)$, $\Gamma \vdash^{\cap\cup}_{\lambda\mu} \mu\gamma.[\gamma]M\{N\cdot\gamma/\alpha\} : \cup_n A_i \mid \Delta$.

$$\dfrac{\dfrac{\Gamma \vdash M\{N\cdot\gamma/\alpha\} : \cup_m(\sigma_j\to A_j) \mid \gamma{:}\cup_n A_i,\Delta \qquad \dfrac{\Gamma \vdash N : \sigma_j \mid \Delta \;\; (j\in\underline{m})}{\Gamma \vdash N : \sigma_j \mid \gamma{:}\cup_n A_i,\Delta}\,(Wk) \;\; (j\in\underline{m})}{\Gamma \vdash M\{N\cdot\gamma/\alpha\}N : \cup_m(A_i) \mid \gamma{:}\cup_n A_i,\Delta}\,(\to E)}{\Gamma \vdash \mu\gamma.[\gamma]M\{N\cdot\gamma/\alpha\}N : \cup_n A_i \mid \Delta}\,(\cup E)$$

$(\mu\alpha.[\beta]\mu\gamma.[\delta]M \to \mu\alpha.[\delta](M[\beta/\gamma]))$: If $\Gamma \vdash^{\cap\cup}_{\lambda\mu} \mu\alpha.[\beta]\mu\gamma.[\delta]M : A \mid \Delta$, the derivation is shaped like:

$$\dfrac{\dfrac{\Gamma \vdash M : D_p \mid \gamma{:}B_l, \delta{:}\cup_n D_i, \beta{:}\cup_{\underline{k}} B_i, \alpha{:}A, \Delta'}{\Gamma \vdash \mu\gamma.[\delta]M : B_l \mid \beta{:}\cup_{\underline{k}} B_i, \alpha{:}A, \Delta'}\,(\cup E)}{\Gamma \vdash \mu\alpha.[\beta]\mu\gamma.[\delta]M : A \mid \beta{:}\cup_{\underline{k}} B_i, \Delta'}\,(\cup E)$$

with $\Delta = \beta{:}\cup_m B_j, \Delta'$, for some $B_i\,(\forall i\in m)$, with $l\in\underline{k}$, and $p\in\underline{n}$. It is straightforward to show that then $\Gamma \vdash^{\cap\cup}_{\lambda\mu} M[\beta/\gamma] : D_p \mid \beta{:}\cup_{\underline{k}} B_i, \alpha{:}A, \Delta'$, and applying rule $(\cup E)$ to this derivation gives $\Gamma \vdash^{\cap\cup}_{\lambda\mu} \mu\alpha.[\delta](M[\beta/\gamma]) : A \mid \Delta$. □

Notice that we cannot show subject reduction for the *erasing* rule. Assume the derivation for $\mu\alpha.[\alpha]M$ with M not a control structure is shaped like

$$\dfrac{\dfrac{\Gamma \vdash M : A_j \mid \alpha{:}\cup_n A_i, \Delta}{\Gamma \vdash [\alpha]M : \bot \mid \alpha{:}\cup_n A_i, \Delta}\,(\cup E)}{\Gamma \vdash \mu\alpha.[\alpha]M : \cup_n A_i \mid \Delta}\,(\mu)$$

Since α does not appear in M, by thinning we can derive $\Gamma \vdash^{\cap\cup}_{\lambda\mu} M : A_j \mid \Delta$, but have no rule to allow us to derive $\Gamma \vdash^{\cap\cup}_{\lambda\mu} M : \cup_n A_i \mid \Delta$ from that.

Conclusion

We have seen that the calculus $\lambda\mu$ is sufficiently limited to allow for the definition of a sound and complete notion of type assignment. This will need to be investigated further, towards the definition of semantics, and characterisation of the termination properties. Also, we need to look at the ignored reduction rules, and see if it is possible to generalise the system such that also these will be preserved, without sacrificing the main results. The approach we use here seems to be promising also for the setting of (restrictions of) $\mathcal{X}$ and $\overline{\lambda}\mu\tilde{\mu}$; we will leave this for future work.

References

[1] Z.M. Ariola & H. Herbelin (2003): *Minimal Classical Logic and Control Operators*. In: *Proceedings of Automata, Languages and Programming, 30th International Colloquium, ICALP'03, 2003*, *Lecture Notes in Computer Science* 2719, Springer, pp. 871–885.

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

[3] S. van Bakel (2010): *Completeness and Partial Soundness Results for Intersection & Union Typing for $\overline{\lambda}\mu\tilde{\mu}$*. *Annals of Pure and Applied Logic* 161, pp. 1400–1430.

[4] S. van Bakel (2010): *Completeness and Soundness results for $\mathcal{X}$ with Intersection and Union Types*. To appear in: Fundamenta Informaticae.

[5] S. van Bakel & U. de'Liguoro (2008): *Logical equivalence for subtyping object and recursive types*. *Theory of Computing Systems* 42(3), pp. 306–348.

[6] S. van Bakel & M. Fernández (1997): *Normalization Results for Typeable Rewrite Systems*. *Information and Computation* 2(133), pp. 73–116.

[7] S. van Bakel, S. Lengrand & P. Lescanne (2005): *The language $\mathcal{X}$: Circuits, Computations and Classical Logic*. In: *Proceedings of Ninth Italian Conference on Theoretical Computer Science* (ICTCS'05), *Lecture Notes in Computer Science* 3701, Springer Verlag, pp. 81–96.

[8] S. van Bakel & P. Lescanne (2008): *Computation with Classical Sequents*. *Mathematical Structures in Computer Science* 18, pp. 555–609.

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

[10] H. Barendregt (1984): *The Lambda Calculus: its Syntax and Semantics*. North-Holland, Amsterdam.

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

[12] A. Church (1936): *A Note on the Entscheidungsproblem*. *Journal of Symbolic Logic* 1(1), pp. 40–41.

[13] M. Coppo & M. Dezani-Ciancaglini (1978): *A New Type Assignment for Lambda-Terms*. *Archive für Mathematischer Logic und Grundlagenforschung* 19, pp. 139–156.

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

[15] H.B. Curry & R. Feys (1958): *Combinatory Logic*, 1. North-Holland, Amsterdam.

[16] R. Davies & F. Pfenning (2001): *A judgmental reconstruction of modal logic*. *Mathematical Structures in Computer Science* 11(4), pp. 511–540.

[17] D. Dougherty, S. Ghilezan & P. Lescanne (2004): *Intersection and Union Types in the* $\overline{\lambda}\mu\tilde{\mu}$*-calculus*. In: *Electronic Proceedings of 2nd International Workshop* Intersection Types and Related Systems *(ITRS'04)*, *Electronic Notes in Theoretical Computer Science* 136, pp. 228–246.

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

[19] J. Dunfield & F. Pfenning (2003): *Type Assignment for Intersections and Unions in Call-by-Value Languages*. In: *Proceedings of 6th International Conference on* Foundations of Software Science and Computational Structures *(FOSSACS'03)*, pp. 250–266.

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

[21] Ph. de Groote (1994): *On the relation between the* $\lambda\mu$*-calculus and the syntactic theory of sequential control*. In: *Proceedings of 5th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'94)*, *Lecture Notes in Computer Science* 822, Springer Verlag, pp. 31–43.

[22] B. Harper & M. Lillibridge (1991): *ML with callcc is unsound*. Post to TYPES mailing list, July 8.

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

[24] J.R. Hindley (1997): *Basic Simple Type Theory*. Cambridge University Press.

[25] S. Lengrand (2003): *Call-by-value, call-by-name, and strong normalization for the classical sequent calculus*. In: *3rd Workshop on Reduction Strategies in Rewriting and Programming (WRS 2003)*, *Electronic Notes in Theoretical Computer Science* 86, Elsevier.

[26] S. Maffeis (2005): *Sequence Types for the pi-calculus*. *Electronic Notes in Theoretical Computer Science* 136, pp. 117–132.

[27] R. Milner, M. Tofte, R. Harper & D. MacQueen (1990): *The Definition of Standard ML*. MIT Press. Revised edition.

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

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

[30] M. Parigot (1993): *Classical Proofs as Programs*. In: *Kurt Gödel Colloquium*, pp. 263–276. Presented at TYPES Workshop, 1992.

[31] M. Parigot (1993): *Strong Normalization for Second Order Classical Natural Deduction*. In: *Proceedings of Eighth Annual IEEE Symposium on Logic in Computer Science, 19-23 June 1993*, pp. 39–46.

[32] B.C. Pierce (1991): *Programming with Intersection Types and Bounded Polymorphism*. Ph.D. thesis, Carnegie Mellon University, School of Computer Science, Pitsburgh. CMU-CS-91-205.

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