Subject Reduction vs Intersection / Union Types in \(\bar{\lambda} \mu \bar{\mu}\)

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 \(\bar{\lambda} \mu \bar{\mu}\) [9], 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.

**keywords:** sequent calculus, intersection types, union types, classical logic

Introduction

One of the main contributions of intersection types [8] for the \(\lambda\)-calculus [7] 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 \(\bar{\lambda} \mu \bar{\mu}\) [9], [11] presented a notion of intersection and union type assignment for that calculus.

Another main contribution of intersection types is that of the construction of filter semantics [8], i.e. to build a \(\lambda\)-model using assignable types. With this goal in mind, in this paper we revisit the system of [11], adding \(\top\) as the maximal and \(\bot\) as the minimal type, and the derivation rules \((\top)\), \((\bot)\), that are added for the purpose of subject expansion (the property that types are preserved also going backwards with respect to reduction, sometimes also called completeness); we also add the rules \((\leq L)\) and \((\leq R)\) as a correction of the system of [11], in that they are needed for the proof of subject reduction (the converse of completeness, also called soundness).

The notion of type (or better, 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 [18]) of both intersection and union. In this regard, [11] contained some flaws, and a later attempt by the same authors [13] to achieve the strong normalisation results had those flaws as well; this paper will address and partially correct these. We should point out that also a previous attempt to tackle the same kind of problem (but in the context of \(\mathcal{X}\), a different sequent calculus) by the author of this paper [3] was flawed in the same way. 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.

Another approach, in spirit close to ours, is that mentioned in [17]; as we will see, the differences between that approach and the one chosen here are subtle, but important. In particular, in [17], intersection and union types do not bring the expected properties: not all normal forms can be typed in that system and it is not closed for subject-expansion, so it does certainly not type exactly the strongly

---

\(^1\)Normally, \(\omega\) is used; here we reserve Greek characters for context variables.
normalisable terms.

In this paper we will show that the – at the time – surprising loss of soundness for the system with intersection and union types in [6] 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 that the problem is much more profound, and also arises 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 [16, 23, 21], 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 [10, 14].

The sequent calculus, introduced by Gentzen in [15], 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. A number of variants exist for Gentzen’s calculus for classical logic \(\text{LK}\); the variant of \(\text{LK}\) one we will consider in this paper, and that lies at the basis of the calculus \(\text{\overline{\lambda}\mu\tilde{\mu}}\) is the calculus known as \(G_3\) [19], which is defined by:

\[
\begin{align*}
(Ax) : & \quad \Gamma, A \vdash A, \Delta \\
(\rightarrow L) : & \quad \Gamma \vdash A, \Delta \quad \Gamma, B \vdash \Delta \quad \frac{}{\Gamma, A \rightarrow B \vdash \Delta} \\
(cut) : & \quad \Gamma, A, \Delta \quad \Gamma, A \vdash \Delta \\
(\rightarrow R) : & \quad \Gamma, A \vdash B, \Delta \\
& \quad \frac{}{\Gamma \vdash A \rightarrow B, \Delta}
\end{align*}
\]

It allows sequents of the form \(A_1, \ldots, A_n \vdash B_1, \ldots, B_m\), where \(A_1, \ldots, A_n\) is to be understood as \(A_1 \land \ldots \land A_n\) and \(B_1, \ldots, B_m\) is to be understood as \(B_1 \lor \ldots \lor B_m\), and has implicit weakening and contraction.

In this paper we will treat \(\overline{\lambda}\mu\tilde{\mu}\) as a pure, untyped calculus, and ignore its origin in that we define a notion of sequent-style intersection-union type assignment for \(\overline{\lambda}\mu\tilde{\mu}\); as mentioned above, intersection types are notorious for lacking a solid background in logic [18]. In order to achieve a natural notion of type assignment that is closed for expansion, we will also need to add union types.

The system we define in this paper is set up to be a conservative extension of the BCD intersection type assignment system for the \(\lambda\)-calculus [8], 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 [20, 1, 2]), that, in the context of the \(\lambda\)-calculus more or less coincide; the most important difference is normally the language of types (full BCD [8, 20], or strict types [1, 2]), and the availability of a contra-variant \(\preceq\)-relation [8, 2] or not [20, 1]. Surprisingly, this is no longer true when bringing intersection types (and union) to the context of sequent calculi; both BCD-types and a contra variant relation are needed, as will be shown in this paper.

The system presented here is also a natural extension of the system considered in [9], i.e. the basic implicative system for Classical Logic, but extended with intersection and union types and the type constants \(\top\) and \(\bot\). 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 \(\mu\)-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.
In [17], 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 4. There the CBN 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 in Section 4. However, this does not seem to affect the validity of the subject reduction claim in [17]; 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.

In this paper we will revisit the systems of [11] and [13], and discuss the problem of subject reduction and expansion in the context of those systems in detail. We will then define a notion of type assignment that is an extension of [11] by adding a contra-variant \(\leq\)-relation on types, and show that subject reduction is still problematic. After that we will focus on two sub-sytems, \(\bowtie v\) and \(\bowtie n\), 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 [12, 13]), and pose completely different side-conditions on rules (wrt [17]), 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 or \(\lambda\mu\bar{\mu}\) using this system.

It might seem that the restriction flattens the system too much; however, as intersection is not problematic for CBN-reduction, we can still embed the \(\bowtie\)-system for the \(\lambda\)-calculus whilst preserving the types, so our system \(\bowtie n\) is as least as expressive; also, union types are still useful for \(\mu\)-reductions. On the other hand, union types are not problematic for CBV-reduction, and intersection can then be used for \(\bar{\mu}\)-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 type assignment that is closed for conversion, even when restricting to CBV of CBN. So we cannot construct a filter model using types for \(\lambda\mu\bar{\mu}\) using this system.

This paper is constructed as follows: in Section 1 we define the \(\lambda\mu\bar{\mu}\)-calculus, and encode the \(\lambda\)-calculus into it. In Section ?? we will present the intersection-union type assignment system \(\mathcal{M}^{\cap\cup}\) of [11] and show, via detailed examples, that subject reduction does not hold. This will be followed in Section ?? by the system \(\mathcal{M}^{\cap}\) of [13]; we will show that apart from subject-reduction, also subject expansion fails there. In Section 2 we will revisit the BCD system of [8], from which in Section 3 we will construct our notion of type assignment, and prove some standard properties, as well as that the encoding of the \(\lambda\)-calculus preserves BCD-types. We will also show subject expansion, and soundness for rule \((\rightarrow)\); we will show that contraction via rules \((\mu)\) and \((\bar{\mu})\) is not sound. In Section 4 we will define two restrictions of our system, \(\bowtie n\) and \(\bowtie v\), and will show that call-by-name and call-by-value, respectively, are sound for these notions; completeness will fail.

1 The calculus \(\overline{\lambda\mu\bar{\mu}}\)

In its typed version, \(\overline{\lambda\mu\bar{\mu}}\) [9] is a proof-term syntax for a classical sequent calculus. For the \(\overline{\lambda\mu\bar{\mu}}\)-calculus there are two sets of variables: \(x, y, z\), etc., label the types of the hypotheses and \(a, b, \gamma, \) etc., label the types of the conclusions. Moreover, the syntax of \(\overline{\lambda\mu\bar{\mu}}\) 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 [9])**

\[
\begin{align*}
  c & ::= \langle v \mid e \rangle & \text{(commands)} \\
  v & ::= x \mid \lambda x.v \mid \mu\beta.c & \text{(terms)} \\
  e & ::= a \mid v\cdot e \mid \bar{\mu}x.c & \text{(contexts)}
\end{align*}
\]

With conventional notations about contexts, \(v\cdot e\) is to be thought of as \(e[[\ ]\ v]\).
Here $\mu_\alpha c$, $\bar{\mu}x.c'$ and $\lambda_\mu v$ respectively bind $\alpha$ in $c$, $x$ in $c'$ and $y$ in $v$; as usual, we will consider terms, contexts and commands up to $\alpha$-conversion, writing them in a way satisfying Barendregt’s convention, and consider substitution to be capture avoiding.

**Definition 1.2** *(Typing for $\bar{\lambda}\mu\bar{\mu}$ [9])*

\[
\text{(cut)} : \quad \Gamma \vdash v : A \mid \Delta \quad \Gamma \mid e : A \vdash \Delta \\
\text{(Ax-t)} : \quad \frac{\Gamma, x : A \vdash x : A \mid \Delta}{\langle v \mid e \rangle : \Gamma \vdash \Delta} \quad \text{(Ax-c)} : \quad \frac{\Gamma \mid \alpha : A \vdash \alpha : A, \Delta}{\langle v \mid e \rangle : \Gamma \vdash \Delta}
\]

\[
\text{(RI)} : \quad \frac{\Gamma \vdash \lambda x.v : A \rightarrow B \mid \Delta}{\langle v \mid e \rangle \rightarrow B : \Gamma \vdash \Delta} \quad \text{(LI)} : \quad \frac{\Gamma \vdash v : A \mid \Delta \quad \Gamma \mid e : B \vdash \Delta}{\langle v \mid e \rangle : \Gamma \vdash \Delta}
\]

\[
\text{(}\mu\text{)} : \quad \frac{c : \Gamma \vdash \alpha : A, \Delta}{\Gamma \vdash \mu_\alpha c : A \mid \Delta} \quad \text{(}\bar{\mu}\text{)} : \quad \frac{c : \Gamma \vdash \bar{\mu}x.c : A \mid \Delta}{\Gamma \vdash \langle v \mid e \rangle : \Gamma \vdash \Delta}
\]

Note that the type of a context is the type that a term is expected to have in order to fit in the hole ‘·’.

We see here how a term (context) is built either by introducing ‘→’ on the right-hand side (left-hand side) of a sequent, or just by activating one conclusion (hypothesis) from a sequent typing a command: $\mu_\alpha c$ is inherited from $\lambda_\mu$, as is $\langle v \mid \alpha \rangle$ which corresponds to $[\alpha]v$, and $\bar{\mu}x.c$ is to be thought of as `let $x = [\ ]$ in $c$.

Commands can be computed (thus eliminating the cut in the corresponding proof):

**Definition 1.3** *(Reduction in $\bar{\lambda}\mu\bar{\mu}$ [9, 17])* The reduction rules are defined by:

\[
\text{logical rules} \quad \text{extensional rules}
\]

\[
\text{(}\rightarrow\text{)} : \quad \langle \lambda x.v_1 \mid v_2.e \rangle \rightarrow \langle v_2 \mid \bar{\mu}x.(\langle v_1 \mid e \rangle) \rangle \quad (\eta) : \quad \lambda x_\mu \beta.(\langle v \mid x.\beta \rangle \rightarrow v \mid x, \beta \not\in \text{fv}(v) \rangle
\]

\[
(\mu) : \quad \langle \mu_\beta.c \mid e \rangle \rightarrow \langle c[e/\beta] \rangle \quad (\eta_\mu) : \quad \mu_\beta.(\langle v \mid \alpha \rangle \rightarrow v \mid \alpha \not\in \text{fv}(v) \rangle
\]

\[
(\bar{\mu}) : \quad \langle v \mid \bar{\mu}x.c \rangle \rightarrow \langle c[v/x] \rangle \quad (\eta\bar{\mu}) : \quad \bar{\mu}x.(\langle x \mid e \rangle \rightarrow e \mid x \not\in \text{fv}(e))
\]

We have a critical pair in the command $\langle \mu_\alpha.c \mid \bar{\mu}x.c_2 \rangle$; since cut-elimination of the classical sequent calculus is not confluent, neither is reduction in $\bar{\lambda}\mu\bar{\mu}$. Notice also that the rules $\text{(}\rightarrow\text{)}$, $\text{(}\mu\text{)}$, and $\text{(}\bar{\mu}\text{)}$ reduce commands to commands, that rules $\text{(}\eta\text{)}$ and $\text{(}\eta\mu\text{)}$ reduce a term to a term, and that rule $\text{(}\eta\bar{\mu}\text{)}$ reduces a context to a context. If not mentioned explicitly, we will assume that reduction is done only via the logical rules.

Herbelin’s $\bar{\lambda}\mu\bar{\mu}$-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 [22]). Indeed, the CBV reduction $\rightarrow_V$ is obtained by forbidding a $\bar{\mu}$-reduction when the redex is also a $\mu$-reduct, whereas the CBN reduction $\rightarrow_N$ forbids a $\mu$-reduction when the redex is also a $\bar{\mu}$-reduct.

**Definition 1.4** *(CBV and CBN [9, 17])* i) Values $V$ are defined by $V ::= x \mid \lambda x.v$, and slots\(^2\) $E$ are defined by $E ::= \alpha \mid v \cdot e$.

ii) CBV-reduction is defined by replacing rule $\text{(}\bar{\mu}\text{)}$ by: $\text{(}\bar{\mu}_V\text{)} : \langle V \mid \bar{\mu}x.c \rangle \rightarrow c[V/x]$.

iii) CBN-reduction is defined by replacing rule $\text{(}\mu\text{)}$ by: $\text{(}\mu_S\text{)} : \langle \mu_\beta.c \mid E \rangle \rightarrow c[E/\beta]$.

The remainder of this paper will be dedicated to the definition of a notion of intersection type assignment on $\bar{\lambda}\mu\bar{\mu}$. This will be defined as a natural extension of a system with intersection types for the $\lambda$-calculus; we will start by briefly summarising the latter.

---

\(^2\)In [17], slots are called *linear evaluation contexts*. 
Definition 1.5 (Lambda terms and beta-contraction [7]) i) The set \( \Lambda \) of lambda terms is defined by the syntax:
\[
M, N ::= x \mid \lambda x. M \mid MN
\]

ii) The reduction relation \( \rightarrow_\beta \) is defined as the contextual (i.e. compatible [7]) closure of the rule:
\[
(\lambda x. M) N \rightarrow_\beta M[N/x]
\]
The relation \( \rightarrow_\beta \) is the reflexive and transitive closure of \( \rightarrow_\beta \), and the \( =_\beta \) is the equivalence relation generated by \( \rightarrow_\beta \).

iii) Call-by-value reduction is defined by limiting the reduction rule \( \rightarrow_\beta \) to contract only if the right-hand term is a value,
\[
(\lambda x. M) V \rightarrow_\beta M[V/x]
\]
i.e. if \( V \) is either a variable or an abstraction.

Essentially following [9], we define an encoding of the \( \lambda \)-calculus into \( \overline{\lambda} \mu \tilde{\mu} \):

Definition 1.6 An interpretation \( I \) of the \( \lambda \)-calculus into \( \overline{\lambda} \mu \tilde{\mu} \) can be defined as follows:
\[
I x = x \quad I(\lambda x. M) = \lambda x. I M \quad I(MN) = \mu a. (I M|I N \cdot a)
\]

We can even represent substitution explicitly (so interpret \( \lambda x [?] \)), by adding
\[
I(M < x = N>) = \mu \gamma.(I N | \tilde{\mu} x. (I M | \gamma))
\]

Correctness of this encoding is easy to prove:

Lemma 1.7 \( I(\lambda x. M) N \rightarrow I M[I N / x] \).
Proof: \( I(\lambda x. M) N = \mu a. (I x. I M|I N \cdot a) = \mu a. (\lambda x. I M|I N \cdot a) \rightarrow \mu a. (I M|\tilde{\mu} x. (I M | \alpha)) \rightarrow_\eta \mu I M[I N / x]. \)

Notice that we need \( \eta \mu \)-reduction to achieve this.

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

Theorem 1.8 i) If \( M \rightarrow_\beta N \), then \( I M \rightarrow I N \).
ii) If \( M \rightarrow_\gamma N \), then \( I M \rightarrow I N \).
Proof: Both parts follow by induction on the definition of \( \rightarrow_\beta \), using Lemma 3.7. For part 2, we also need to check that \( I(\lambda x. M) N \rightarrow_\gamma I M[I N / x] \) only if \( N \) is a value. Well, then either \( N \equiv x \) or \( N \equiv \lambda y. N' \), and for both cases \( I N \) is a value. Then the \( \tilde{\mu} \)-reduction in the proof of Lemma 3.7 is permitted, making the reduction CBV.

2 The system \( M^{\cap \cup} \) of intersection and union type assignment for \( \overline{\lambda} \mu \tilde{\mu} \) [11]

A first notion of intersection and union type assignment for \( \overline{\lambda} \mu \tilde{\mu} \) was presented in [11]; this was followed by [13], which presented improvements/variants of the original system. In this paper, we will revisit these systems, in order, albeit in a presentation that is adapted to our approach, and discuss their 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:

$$A, B ::= \emptyset \mid (A \rightarrow B) \mid (A \cap B) \mid (A \cup B)$$

ii) A context $\Gamma$ of term-variables ($\Delta$ 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 ‘$\rightarrow$’, so $A \cap B \rightarrow C \cap D$ is used for $((A \cap B) \rightarrow (C \cap D))$, $A \rightarrow B \cap C \rightarrow D$ for $(A \rightarrow ((B \cap C) \rightarrow D))$, and $(A \rightarrow B) \cap C \rightarrow D$ for $(((A \rightarrow B) \cap C) \rightarrow D)$. We will sometimes write the omitable brackets to enhance readability.

We will consider a pre-order on types which takes into account the idempotence, 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:

$$A \cap B \leq A, B \quad A \leq C \& A \leq B \Rightarrow A \leq B \cap C$$

$$A, B \leq A \cup B \quad A \leq C \& B \leq C \Rightarrow A \cup B \leq C$$

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:

$$A \leq B \leq A \Rightarrow A \sim B$$

$$A \sim C \& B \sim D \Rightarrow A \rightarrow B \sim C \rightarrow D$$

The set $\mathcal{T}$ will be considered modulo $\sim$; then $\leq$ becomes a partial order. Since intersection and union are associative and commutative, we will write $\cap_n A_i$ for $A_1 \cap \cdots \cap A_n$, as well as $\cup_n A_i$ for $A_1 \cup \cdots \cup A_n$.

The following is easy to show:

**Lemma 2.3**  
i) If $A \leq_\cap B$, then $A = \cap_n A_i, B = \cap_m B_j$, and for every $B_j$ there exists an $A_i$ such that $A_i = B_j$.

ii) If $A \leq_\cup B$, then $A = \cup_n A_i, B = \cup_m B_j$, and for every $A_i$ there exists a $B_j$ such that $A_i = B_j$.

**Proof:** Straightforward. \hfill $\square$

We will now define a notion of intersection-union type assignment for $\bar{\lambda}\mu\bar{\mu}$, which corresponds to system $\mathcal{M}^{\cap\cup}$ as defined in [11].

**Definition 2.4 (System $\mathcal{M}^{\cap\cup}$ [11])**

$$\frac{\Gamma_1 \vdash \mu \nu : A \mid \Delta_1 \quad \Gamma_2 \mid e : A \vdash \Delta_2}{\langle \nu \mid e \rangle : \Gamma_1 \cap \Gamma_2 \vdash \mu \nu \Delta_1 \cup \Delta_2}$$

**(cut)**
(Ax-t) : \[ \Gamma, x:A \vdash M x : B \mid \Delta \quad (A \leq \cap B) \]

(\neg R) : \[ \Gamma, x:A \vdash v : B \mid \Delta \quad \Gamma \vdash \lambda x.v : A \rightarrow B \mid \Delta \]

(\mu) : \[ c : \Gamma \vdash \mu \alpha:A, \Delta \quad \Gamma \vdash \mu \alpha.c : A \mid \Delta \]

(\cap R) : \[ \Gamma \vdash v : A_j \mid \Delta \quad (\forall j \in \nu (n \geq 0)) \quad \Gamma \vdash v : \cap A_{ij} \mid \Delta \]

(\cup R) : \[ \Gamma \vdash v : A_j \mid \Delta \quad (j \in \nu) \quad \Gamma \vdash v : \cup A_{ij} \mid \Delta \]

Notice that instead of the rules (Ax-t) and (Ax-c), the system of [11] contains the rules (e-ax) and (r-ax) that only allow selection of a type out of a context, instead of an selection out of an intersection or union type as well; to be able to select such a subtype, it also had rules (\cap L-var) and (\cup R-var); our rules combine these, with the same functionality.

Also, their notion of type assignment 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 contra-variant type inclusion relation (which would state that \( C \leq A \) \& \( B \leq D \implies A \rightarrow B \leq C \rightarrow D \)); this combined with the full BCD-intersection types [8] gives a notion of type assignment that is closest to Krivine's system D [20].

Notice that type assignment is not preserved for rule (\eta): for example, we can derive

\[
\frac{x:A \cap C \vdash x : A \mid \beta:B}{y:A \vdash y : A \rightarrow B \mid \Delta} \quad \frac{\beta:B \vdash \beta:B}{\langle A \cap C \rightarrow B \mid \Delta}
\]

but cannot show \( y:A \rightarrow B \vdash y : (A \cap C) ightarrow B \mid \Delta \); we would need a \leq\)-relation on types that is contra-variant over arrow types.

Similar to the results we will show below (Theorem 3.12), it is straightforward to show that this system is closed for expansion.

**Theorem 2.5** (Subject Expansion) *Let \( c_1 \rightarrow c_2 \) if \( c_2 : \Gamma \vdash \Delta \) then \( c_1 : \Gamma \vdash \Delta \).*

However, it is not closed for reduction, so we can’t show that, if \( c_1 \rightarrow c_2 \) and \( c_1 : \Gamma \vdash \Delta \) then also \( c_2 : \Gamma \vdash \Delta \). To illustrate this, we will show two counterexamples in detail.

**Example 2.6** Let

\[
c_1 = \langle \mu \alpha. \langle x | x \cdot \alpha \rangle | \mu y. \langle y | y \cdot \beta \rangle \rangle,
\]

\[
c_2 = \langle x | x \cdot \mu y. \langle y | y \cdot \beta \rangle \rangle,
\]

\[
c_3 = \langle x | x \cdot (\mu \alpha. \langle x | x \cdot \alpha \rangle \cdot \beta) \rangle
\]

Then it is straightforward to verify that \( c_1 \rightarrow_{\nu} c_2 \), and \( c_1 \rightarrow_{\pi} c_3 \). Let the derivation \( D_1 \) be constructed as follows:
Moreover, let $\mathcal{D}_2$ be

$$
\begin{array}{c}
x: A \vdash_M x: A \\
x: A \mid x: A \vdash_M a: C \\
\langle x | x \cdot a \rangle: x: A \cap (A \rightarrow C) \vdash_M a: C \\
x: A \cap (A \rightarrow C) \vdash_M a: C \\
\langle x | x \cdot a \rangle: x: A \cap (A \rightarrow C) \vdash_M a: C \\
x: A \cap (A \rightarrow C) \vdash_M a: C \\
\langle y | y \cdot \beta \rangle: (C \cap C) \vdash_M \beta: D \\
\end{array}
$$

and $\mathcal{D}_3$ be

$$
\begin{array}{c}
x: A \vdash_M x: A \\
x: A \mid x: A \vdash_M a: C \\
\langle x | x \cdot a \rangle: x: A \cap (A \rightarrow C) \vdash_M a: C \\
x: A \cap (A \rightarrow C) \vdash_M a: C \\
\langle x | x \cdot a \rangle: x: A \cap (A \rightarrow C) \vdash_M a: C \\
x: A \cap (A \rightarrow C) \vdash_M a: C \\
\langle y | y \cdot \beta \rangle: (C \cap C) \vdash_M \beta: D \\
\end{array}
$$

then a derivation for the first term $c_1$ is:

$$
\begin{array}{c}
\vdash_M (\cap \mathcal{R}) \\
\vdash_M
\end{array}
$$

When reducing $c_1$ to $c_3$, there is no problem. First, we reduce $c_1$ to $\langle \mu a. \langle x | x \cdot a \rangle | \mu a. \langle x | x \cdot a \rangle \cdot \beta \rangle$ via rule $\rightarrow_\beta$:

$$
\begin{array}{c}
x: A \vdash_M x: A \\
x: A \mid x: A \vdash_M a: C \\
\langle x | x \cdot a \rangle: x: A \cap (A \rightarrow C) \vdash_M a: C \\
x: A \cap (A \rightarrow C) \vdash_M a: C \\
\langle x | x \cdot a \rangle: x: A \cap (A \rightarrow C) \vdash_M a: C \\
x: A \cap (A \rightarrow C) \vdash_M a: C \\
\langle \mu a. \langle x | x \cdot a \rangle | \mu a. \langle x | x \cdot a \rangle \cdot \beta \rangle: x: A \cap (A \rightarrow C) \cap (A \rightarrow C) \vdash_M \beta: D \\
\end{array}
$$

(notice that we have made $\mathcal{D}_3$ explicit). We then contract the $\mu a$ redex to obtain $\langle x | x \cdot (\mu a. \langle x | x \cdot a \rangle \cdot \beta) \rangle$:
most derive:

\[ x : A \land (A \to C) \vdash_{\mathcal{M}} \mu a. \langle x \cdot x \cdot \alpha : C \mid x \cdot A \vdash_{\mathcal{M}} \beta : D \]

\[ x : A \vdash_{\mathcal{M}} x : A \]

\[ x : A \land (A \to C) \vdash_{\mathcal{M}} \mu a. \langle x \cdot x \cdot \alpha : C \mid x \cdot A \vdash_{\mathcal{M}} \beta : D \]

\[ x : A \vdash_{\mathcal{M}} x : A \to C \to D \]

\[ \langle x \cdot x \cdot \mu a. \langle x \cdot x \cdot \alpha : C \mid x \cdot A \vdash_{\mathcal{M}} \beta : D \]

When contracting the \( \mu a \) redex on the other hand, the context \( \bar{\mu} y. \langle y \cdot y \cdot \beta \rangle \) will be inserted for the context variable \( \alpha \). But we cannot derive:

\[ \langle x \cdot x \cdot \bar{\mu} y. \langle y \cdot y \cdot \beta \rangle \rangle : x : A \land (A \to C) \land (A \to C \to D) \vdash_{\mathcal{M}} \beta : D \]

We would expect the derivation \( D_1 \) to be inserted for \( \alpha \) (in one way or another; since intersection and union are involved in this system, it might be that a derivation gets divided into pieces before being inserted, as we have seen above) in the derivation for \( \langle x \cdot x \cdot \alpha \rangle \). However, notice that we do not have a sub-derivation for \( \mid \alpha : (C \to D) \land C \mid \mathcal{M} : \alpha : (C \to D) \land C \), but only two for both \( \mid \alpha : C \mid \mathcal{M} : \alpha : C \land D \). This means that a derivation for the contractum cannot be simply constructed from the one above by ‘cut and paste’: we need to build a new derivation.

Now we cannot split the intersection needed to derive \( D_1 \): both types \( C \) and \( C \to D \) are needed to type \( \bar{\mu} y. \langle y \cdot y \cdot \beta \rangle \), so we cannot derive either \( \mid \bar{\mu} y. \langle y \cdot y \cdot \beta \rangle : C \to D \vdash_{\mathcal{M}} \beta : D \), nor \( \mid \bar{\mu} y. \langle y \cdot y \cdot \beta \rangle : C \vdash_{\mathcal{M}} \beta : D \).

Given the structure of the rules in system \( \vdash \), for \( \langle x \cdot x \cdot \bar{\mu} y. \langle y \cdot y \cdot \beta \rangle \rangle \) this implies that for \( c_2 \) we can at most derive:

\[ \langle x \cdot x \cdot \bar{\mu} y. \langle y \cdot y \cdot \beta \rangle \rangle \mid \bar{\mu} y. \langle y \cdot y \cdot \beta \rangle : (C \to D) \land C \mid \mathcal{M} : \beta : D \]

Notice that this is incomparable to the desired result: there is no way we can transform \( A, A \to C \) and \( A \to C \to D \) (i.e. \( A \land (A \to C) \land (A \to C \to D) \)) into \( A \land (A \to (C \to D) \land C) \). We would need the contravariant relation \( \leq \) on types as defined below (in Definition 3.2) to achieve this; since that relation is not considered in [11], this is not possible. So we have a type assignment for \( c_1 \) that is a valid type assignment also for \( c_3 \), but not for \( c_2 \).

By the subject expansion result, the above type assignment for \( c_2 \) is also a type assignment for \( c_1 \), so, in the opposite direction, no problem exists; we can derive \( c_1 : x : A \land (A \to ((C \to D) \land C)) \mid \mathcal{M} : \beta : D \):

\[ \langle x \cdot x \cdot \bar{\mu} y. \langle y \cdot y \cdot \beta \rangle \rangle \mid \bar{\mu} y. \langle y \cdot y \cdot \beta \rangle : (C \to D) \land C \mid \mathcal{M} : \beta : D \]

A particular problem with this last type assignment is that we (have shown that we) can derive it for both \( c_1 \) and \( c_2 \), but cannot derive it for \( c_3 \); a derivation for \( c_3 \) necessarily should have the shape as shown above. There is no way we can derive \( c_3 : x : A \land (A \to (C \to D) \land C) \mid \mathcal{M} : \beta : D \); again, we could solve this by \( \leq \), but this forms no part of the system.
Notice that the typing for $c_2$ shows the usefulness of allowing intersection types on the right of an arrow.

Summarising, this counterexample shows the following things:

i) there exist terms $c_1, c_2$, and $c_3$ such that $c_1 \rightarrow c_2$ and $c_1 \rightarrow c_3$, and

ii) there are contexts $\Gamma_1$ and $\Delta_1$ such that $c_1 : \Gamma_1 \vdash \Delta_1$ and $c_2 : \Gamma_1 \vdash \Delta_1$, but not $c_3 : \Gamma_1 \vdash \Delta_1$, and

iii) there are contexts $\Gamma_2$ and $\Delta_2$ such that $c_1 : \Gamma_2 \vdash \Delta_2$ and $c_3 : \Gamma_2 \vdash \Delta_2$, but not $c_2 : \Gamma_2 \vdash \Delta_2$.

iv) for all contexts $\Gamma, \Delta$ such that either $c_2 : \Gamma \vdash \Delta$ or $c_3 : \Gamma \vdash \Delta$, we have $c_1 : \Gamma \vdash \Delta$.

v) there are no contexts $\Gamma, \Delta$ such that $c_1 : \Gamma \vdash \Delta$ and $c_2 : \Gamma \vdash \Delta$ and $c_3 : \Gamma \vdash \Delta$.

We have seen that part of the encountered problems can be overcome by using a contra-variant $\leq$ relation, as we will do in Section 3.

The problem of loss of soundness exists also for $\mu$-reduction:

**Example 2.7** Take the $\mu$-reduction

$$\langle \mu \delta \cdot \langle \lambda x \mu \beta \cdot \langle x | \delta \rangle \ | \delta \rangle \ | \mu z \cdot \langle \lambda v \cdot z | \gamma \rangle \rangle \rightarrow \langle \lambda v \cdot \mu \delta \cdot \langle \lambda x \mu \beta \cdot \langle x | \delta \rangle \ | \delta \rangle \ | \gamma \rangle \rangle$$

Let $D_1$ be the derivation

$$
\begin{array}{c}
\frac{x:A \vdash x:A}{\langle x | \delta \rangle : x:A \vdash \beta:B, \delta:A} \\
\frac{x:A \vdash \beta:B, \delta:A}{\langle x | \delta \rangle : x:A \vdash \beta:B, \delta:A} \\
\frac{\langle x | \delta \rangle : x:A \vdash \beta:B, \delta:A}{\langle \lambda x \mu \beta \cdot \langle x | \delta \rangle : A \vdash \delta:A} \\
\frac{\langle \lambda x \mu \beta \cdot \langle x | \delta \rangle : A \vdash \delta:A}{\langle \lambda x \mu \beta \cdot \langle x | \delta \rangle : A \vdash \delta:A} \\
\frac{\langle \lambda x \mu \beta \cdot \langle x | \delta \rangle : A \vdash \delta:A}{\langle \lambda x \mu \beta \cdot \langle x | \delta \rangle : A \vdash \delta:A} \\
\frac{\langle \lambda x \mu \beta \cdot \langle x | \delta \rangle : A \vdash \delta:A}{\langle \lambda x \mu \beta \cdot \langle x | \delta \rangle : A \vdash \delta:A} \\
\frac{\langle \lambda x \mu \beta \cdot \langle x | \delta \rangle : A \vdash \delta:A}{\langle \lambda x \mu \beta \cdot \langle x | \delta \rangle : A \vdash \delta:A} \\
\frac{\langle \lambda x \mu \beta \cdot \langle x | \delta \rangle : A \vdash \delta:A}{\langle \lambda x \mu \beta \cdot \langle x | \delta \rangle : A \vdash \delta:A} \\
\end{array}
$$

Notice that this derivation shows that both types $A$ and $A \rightarrow B$ are needed here; we cannot derive either $\vdash \mu \delta \cdot \langle \lambda x \mu \beta \cdot \langle x | \delta \rangle \ | \delta \rangle : A$ nor $\vdash \mu \delta \cdot \langle \lambda x \mu \beta \cdot \langle x | \delta \rangle \ | \delta \rangle : A \rightarrow B$.

Let $D_2$ be:

$$
\begin{array}{c}
\frac{\langle \lambda v \cdot z | \gamma \rangle : z:A \vdash \beta:B, \gamma:C \rightarrow A \rightarrow B}{\langle \lambda v \cdot z | \gamma \rangle : z:A \vdash \beta:B, \gamma:C \rightarrow A \rightarrow B} \\
\frac{\langle \lambda v \cdot z | \gamma \rangle : z:A \vdash \beta:B, \gamma:C \rightarrow A \rightarrow B}{\langle \lambda v \cdot z | \gamma \rangle : z:A \vdash \beta:B, \gamma:C \rightarrow A \rightarrow B} \\
\frac{\langle \lambda v \cdot z | \gamma \rangle : z:A \vdash \beta:B, \gamma:C \rightarrow A \rightarrow B}{\langle \lambda v \cdot z | \gamma \rangle : z:A \vdash \beta:B, \gamma:C \rightarrow A \rightarrow B} \\
\frac{\langle \lambda v \cdot z | \gamma \rangle : z:A \vdash \beta:B, \gamma:C \rightarrow A \rightarrow B}{\langle \lambda v \cdot z | \gamma \rangle : z:A \vdash \beta:B, \gamma:C \rightarrow A \rightarrow B} \\
\frac{\langle \lambda v \cdot z | \gamma \rangle : z:A \vdash \beta:B, \gamma:C \rightarrow A \rightarrow B}{\langle \lambda v \cdot z | \gamma \rangle : z:A \vdash \beta:B, \gamma:C \rightarrow A \rightarrow B} \\
\frac{\langle \lambda v \cdot z | \gamma \rangle : z:A \vdash \beta:B, \gamma:C \rightarrow A \rightarrow B}{\langle \lambda v \cdot z | \gamma \rangle : z:A \vdash \beta:B, \gamma:C \rightarrow A \rightarrow B} \\
\frac{\langle \lambda v \cdot z | \gamma \rangle : z:A \vdash \beta:B, \gamma:C \rightarrow A \rightarrow B}{\langle \lambda v \cdot z | \gamma \rangle : z:A \vdash \beta:B, \gamma:C \rightarrow A \rightarrow B} \\
\frac{\langle \lambda v \cdot z | \gamma \rangle : z:A \vdash \beta:B, \gamma:C \rightarrow A \rightarrow B}{\langle \lambda v \cdot z | \gamma \rangle : z:A \vdash \beta:B, \gamma:C \rightarrow A \rightarrow B} \\
\frac{\langle \lambda v \cdot z | \gamma \rangle : z:A \vdash \beta:B, \gamma:C \rightarrow A \rightarrow B}{\langle \lambda v \cdot z | \gamma \rangle : z:A \vdash \beta:B, \gamma:C \rightarrow A \rightarrow B} \\
\end{array}
$$

We can type the left-hand side as follows:

$$
\begin{array}{c}
\begin{array}{c}
D_1 \\
\vdash \mu \delta \cdot \langle \lambda x \mu \beta \cdot \langle x | \delta \rangle \ | \delta \rangle : A \vdash \beta:B, \gamma:C \rightarrow A \rightarrow B \\
\vdash \mu \delta \cdot \langle \lambda x \mu \beta \cdot \langle x | \delta \rangle \ | \delta \rangle : A \vdash \beta:B, \gamma:C \rightarrow A \rightarrow B \\
\vdash \mu \delta \cdot \langle \lambda x \mu \beta \cdot \langle x | \delta \rangle \ | \delta \rangle : A \vdash \beta:B, \gamma:C \rightarrow A \rightarrow B \\
\vdash \mu \delta \cdot \langle \lambda x \mu \beta \cdot \langle x | \delta \rangle \ | \delta \rangle : A \vdash \beta:B, \gamma:C \rightarrow A \rightarrow B \\
\vdash \mu \delta \cdot \langle \lambda x \mu \beta \cdot \langle x | \delta \rangle \ | \delta \rangle : A \vdash \beta:B, \gamma:C \rightarrow A \rightarrow B \\
\vdash \mu \delta \cdot \langle \lambda x \mu \beta \cdot \langle x | \delta \rangle \ | \delta \rangle : A \vdash \beta:B, \gamma:C \rightarrow A \rightarrow B \\
\vdash \mu \delta \cdot \langle \lambda x \mu \beta \cdot \langle x | \delta \rangle \ | \delta \rangle : A \vdash \beta:B, \gamma:C \rightarrow A \rightarrow B \\
\vdash \mu \delta \cdot \langle \lambda x \mu \beta \cdot \langle x | \delta \rangle \ | \delta \rangle : A \vdash \beta:B, \gamma:C \rightarrow A \rightarrow B \\
\vdash \mu \delta \cdot \langle \lambda x \mu \beta \cdot \langle x | \delta \rangle \ | \delta \rangle : A \vdash \beta:B, \gamma:C \rightarrow A \rightarrow B \\
\end{array}
\end{array}
$$

When building a derivation for

$$\langle \lambda v \cdot \mu \delta \cdot \langle \lambda x \mu \beta \cdot \langle x | \delta \rangle \ | \delta \rangle \ | \gamma \rangle : \vdash \gamma:C \rightarrow A \rightarrow B \rangle$$
we would expect the derivation $D$ to be inserted for $z$ in the derivation for $\langle \lambda v.z | \gamma \rangle$. But, because of the inseparability of $A$ and $A \rightarrow B$ for $\mu \delta.\langle \lambda x \mu \beta.\langle x | \delta \rangle | \delta \rangle$, this is impossible; we can at most derive (notice that we have weakened the left context in $D$ with $v:C$):

$$\frac{\langle \lambda x \mu \beta.\langle x | \delta \rangle | \delta \rangle : v:C \vdash_\delta \gamma : A \cup (A \rightarrow B)}{v:C \vdash_\delta \lambda v.\mu \delta.\langle \lambda x \mu \beta.\langle x | \delta \rangle | \delta \rangle : A \cup (A \rightarrow B) | \gamma : C \rightarrow A \cup (A \rightarrow B) \vdash_\delta \gamma : C \rightarrow A \cup (A \rightarrow B)}$$

Again, by the subject-expansion result, we can also derive:

$$\frac{\lambda \mu \delta.\langle \lambda x \mu \beta.\langle x | \delta \rangle | \delta \rangle : \vdash_\delta \gamma : C \rightarrow A \cup (A \rightarrow B) | \gamma : C \rightarrow A \cup (A \rightarrow B) \vdash_\delta \gamma : C \rightarrow A \cup (A \rightarrow B)}{z : A \cup (A \rightarrow B) \vdash_\mu \lambda v.z : A \cup (A \rightarrow B) | \gamma : C \rightarrow A \cup (A \rightarrow B) \vdash_\delta \gamma : C \rightarrow A \cup (A \rightarrow B)}$$

Notice that now we cannot achieve the desired result through $\leq$; it is easy to verify that

$$(C \rightarrow A) \cup (C \rightarrow A \rightarrow B) \leq C \rightarrow A \cup (A \rightarrow B)$$

but not vice-versa, as is needed here.

The conclusion of these two counterexamples has to be that it is profoundly impossible to achieve soundness (or subject reduction) in the system $M^{(\cup)}$.

As we have seen, adding contra-variant $\leq$ to our system would fix part of the here noted problems, and that is exactly what we will do in Section 3.

3 The system $M^{\cap}$ of [13]

A normal reason to set up a notion of type assignment with intersection types is to prove a characterisation result for strong normalisation, and that was one of the motivations for [11]. In order to achieve this result, the authors of that paper decided to modify the system they defined in [11] quite drastically, adding negation, and expressing union types via an intersection and negation. This is accompanied by transforming derivable statements from sequent style to natural deduction style, writing only a context to the left and to have a single result (conclusion).

We will briefly revisit that system here (adapting notation and syntax), and show that the above counter examples are valid in that modified system as well.

It is important to point out that [13] presents a slightly different version of $\overline{\mu} \mu \overline{\nu}$’s reduction relation (Definition 1.3): instead of rule (→) it uses [9]’s rule (→'): $\langle \lambda x.v_1 | v_2.e \rangle \rightarrow \langle v_1[v_2/x] | e \rangle$. For example, for the full reduction as we consider here, we can reduce as follows:

$$\langle \lambda y.\mu \beta.\langle y | y \cdot \beta \rangle | \mu \gamma.\langle x | x \cdot \gamma \rangle \cdot \alpha \rangle \rightarrow \langle \mu \gamma.\langle x | x \cdot \gamma \rangle | \overline{\mu} y.\langle \mu \beta.\langle y | y \cdot \beta \rangle | \alpha \rangle \rangle$$

Now this latter term can be reduced in two ways: the first is:

$$\langle \mu \gamma.\langle x | x \cdot \gamma \rangle | \overline{\mu} y.\langle \mu \beta.\langle y | y \cdot \beta \rangle | \alpha \rangle \rangle \rightarrow (\mu) \langle x | x \cdot \overline{\mu} y.\langle \mu \beta.\langle y | y \cdot \beta \rangle | \alpha \rangle \rangle$$

which corresponds to a CBV reduction.
The other contraction gives:

\[
\langle \mu \gamma \cdot (x \cdot x') \mid \tilde{\mu} y. \langle \mu \beta \cdot (y \cdot y') \mid \alpha \rangle \rangle \rightarrow (\tilde{\mu}) \langle \mu \beta \cdot (\mu \gamma \cdot (x \cdot x') \mid \mu \gamma \cdot (x \cdot x') \cdot \beta) \mid \alpha \rangle
\]

which corresponds to a CBN reduction. Using the reduction rule \((\rightarrow')\), we can only reduce:

\[
\langle \lambda y. \mu \beta \cdot (y \cdot y') \mid \mu \gamma \cdot (x \cdot x') \cdot \alpha \rangle \rightarrow \langle \mu \beta \cdot (\mu \gamma \cdot (x \cdot x') \mid \mu \gamma \cdot (x \cdot x') \cdot \beta) \mid \alpha \rangle
\]

So \([13]\) partially (i.e. essentially) restricts reduction to CBN.

**Definition 3.1** (System \(M^\cap [13]\))

\[\begin{align*}
\text{i)} & \quad \text{Types are defined by} \\
& \quad \quad A, B ::= \varphi \mid A \rightarrow B \mid A \cap B \mid A^o
\end{align*}\]

Intersection is associative and commutative, and \((A^o)^o = A\).

\[\begin{align*}
\text{ii)} & \quad \text{The type assignment rules are defined by:} \\
& \quad \quad (\text{ax}) : \quad \sum_i x: \mu \gamma \cdot A_i \vdash \mu \alpha \cdot x : A_i \\
& \quad \quad (\text{cut}) : \quad \sum_i \alpha \cdot x : A_i \vdash \beta \cdot e : B^o \\
& \quad \quad (\rightarrow) : \quad \sum_i x: A \vdash \beta \cdot e : B \\
& \quad \quad (\leftrightarrow) : \quad \sum_i x: A \vdash \beta \cdot e : B \\
& \quad \quad (\mu) : \quad \sum_i x: A \vdash \beta \cdot e : \bot \\
& \quad \quad (\tilde{\mu}) : \quad \sum_i x: A \vdash \beta \cdot e : \bot
\end{align*}\]

Notice that union types are omitted, but that they are ‘silently’ present as negated intersection types; it is the view of the authors of \([13]\) that union is not the dual of intersection, a view we don’t share here. Moreover, notice that rule \((\cap R)\) is missing altogether: this in particular creates a problem with the subject expansion result that is claimed in \([13]\), but collapses. There is only an implicit \((\cap R)\) in rule \((\rightarrow e)\), much as there was one in rule \((\rightarrow E)\) in the first presentation of the strict type assignment system in \([1]\); that system, however, also had an implicit general \((\cap I)\) rule that is missing here.

As for the subject reduction and expansion (soundness and completeness) results stated in \([13]\), Example ?? essentially provides also a counter example against both. We will construct those, translating the derivations of that example to fit the new system.

**Example 3.2** Notice that, since the first derivation in Example ?? for \(c_1\) uses rule \((\cap R)\), we cannot simulate that derivation in \(M^\cap\), so cannot derive

\[
x : A \cap (A \rightarrow C) \cap (A \rightarrow C \rightarrow D), \beta : D^o \vdash_c \langle \mu a. \langle x \cdot x' \rangle \mid \tilde{\mu} y. \langle y \cdot y' \rangle \rangle : \bot
\]

We can, of course, derive that for \(c_3\)!

\[
\begin{array}{c}
\text{We can, of course, derive that for } c_3! \\
\end{array}
\]

This now gives an immediate counter example against the subject expansion result of \([13]\): \(c_1\) reduces to \(c_3\), but not every type assignment for \(c_3\) is valid for \(c_1\).
We can assigning $c_2$’s type to $c_1$. First we construct $D$:

\[
\begin{align*}
\frac{y : C \vdash y : C}{y : C \to D \vdash y : C} & \quad \frac{\beta : D^\circ \vdash \beta : D^\circ}{y, \beta \vdash (C \to D)^\circ} \\
\frac{y : (C \to D)^\circ \vdash C, \beta : D^\circ \vdash y \cdot \beta}{\beta : D^\circ \vdash \mu y.(y \cdot \beta) : ((C \to D)^\circ \cap C)^\circ}
\end{align*}
\]

then a derivation for $c_1$ is:

\[
\begin{align*}
x : A \vdash x : A & \quad a : (\vdash (C \to D)^\circ \cap C)^\circ \vdash x : (A \to (C \to D)^\circ) \\
x : A \vdash x : A & : (A \to (C \to D)^\circ) \\
x : A \vdash x : A & : (A \to (C \to D)^\circ) \\
& \vdash (C \to D)^\circ \cap C)
\end{align*}
\]

\[
\begin{align*}
x : A \vdash x : A & : (A \to (C \to D)^\circ) \\
\vdash (C \to D)^\circ \cap C) & : (C \to D)^\circ \cap C)
\end{align*}
\]

\[
\begin{align*}
x : A \vdash x : A & : (A \to (C \to D)^\circ) \\
\vdash (C \to D)^\circ \cap C) & : (C \to D)^\circ \cap C)
\end{align*}
\]

So the only statement derivable in the system of [13] for $c_1$ is $c_2$’s, which cannot be derived for $c_3$, for reasons similar to those discussed in Example ??; this gives a counterexample against subject reduction.

So both subject reduction and expansion of [13] collapse.

As already noted in [17], the problem lies in the fact that, although both intersection and union might be similar to the (logical) and and or, they are both not logical [18]. In fact, when deriving

\[
\begin{align*}
& c_1 : \Gamma \vdash_c \alpha : A, \Delta \\
\Gamma & \vdash_c \mu \alpha . c_1 : A | \Delta \\
& \vdash (\cap) \quad c_2 : \Gamma, x : A \cap B \vdash_c x : B | \Delta
\end{align*}
\]

in the derivation for $c_2 : \Gamma, x : A \cap B | \Delta$, there will be two ‘leaves’ for $x$, one deriving $x : A \cap B | \Delta$, and the other deriving $x : A \cap B | \Delta$. Then, as shown in Lemma 4.3, performing the $\mu$ reduction poses no problems, since the two derivations for $\Gamma \vdash_c \mu \alpha . c_1 : A | \Delta$ and $\Gamma \vdash_c \mu \alpha . c_1 : B | \Delta$ will be safely ‘inserted’ yielding a derivation for $c_2 [\mu \alpha . c_1 / x] : \Gamma | \Delta$. Contracting the $\mu$ redex, on the other hand, poses a problem. Since there is no ‘leaf’ for $\alpha$ that derives $\alpha : A \cap B | \Delta$, we cannot ‘insert’ the derivation for $\Gamma | \mu \alpha . c_2 : A \cap B | \Delta$ for the ‘leaves’ for $\alpha$: this derivation does not ‘distribute’.

A similar argument holds regarding union types and $\mu$ reduction.

4 The BCD Intersection Type Assignment for the $\lambda$-calculus [8]

The notion of intersection type assignment for $\lambda \mu \tilde{\mu}$ as defined in the next section is a conservative extension of the BCD-system presented in [8], in that we can translate lambda terms typeable in that system to $\lambda \mu \tilde{\mu}$ 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 $\lambda$-calculus (see [2]), but for $\lambda \mu \tilde{\mu}$, as we saw in
Example ??, to type all normal forms it is natural to have an intersection type occur on the right-hand side of an arrow type.

**Definition 4.1** (Intersection types, statements, and contexts)  
1) Let \( \mathcal{V} \) be a countable (infinite) set of type-variables, ranged over by \( \varphi \). The set \( \mathcal{T} \) of intersection types, ranged over by \( A, B, \ldots \), is defined through the grammar:

\[
A, B ::= \varphi \mid \top \mid (A \rightarrow B) \mid (A \cap B)
\]

2) 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 \( M : A \).

3) A context \( \Gamma \) is a partial mapping from term variables to intersection types, and we write \( x : A \in \Gamma \) if \( \Gamma x = A \). We will write \( x \not\in \Gamma \) if \( \Gamma \) is not defined on \( x \), and \( \Gamma \setminus x \) when we remove \( x \) from the domain of \( \Gamma \).

In the notation of types, as usual, right-most outer-most brackets will be omitted.

We will consider a pre-order on types which takes into account the idempotence, commutativity and associativity of the intersection type constructor, and defines \( \top \) to be the maximal element.

**Definition 4.2** The relation \( \leq \) is defined as the least pre-order (i.e. reflexive and transitive relation) on \( \mathcal{T} \) such that:

\[
A \cap B \leq A, B \quad (A \rightarrow B) \cap (A \rightarrow C) \leq A \rightarrow B \cap C \\
A \leq \top \quad C \leq A \& C \leq B \Rightarrow C \leq A \cap B \\
\top \leq \top \rightarrow \top \quad A \leq C \& B \leq D \Rightarrow A \rightarrow B \leq C \rightarrow D
\]

and the relation \( \sim \) is defined by:

\[
A \leq B \leq A \Rightarrow A \sim B
\]

\( \mathcal{T} \) will be considered modulo \( \sim \); then \( \leq \) becomes a partial order. We will write \( n \) for the set \( \{1, \ldots , n\} \). It is easy to show that both \( (A \cap B) \cap C \sim B \cap (A \cap C) \) 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 not an intersection type.

**Definition 4.3** Intersection type assignment and derivations are defined by the following (sequent style) natural deduction system.

\[
(Ax) : \frac{\Gamma, x : A \vdash \gamma, x : A}{\Gamma, x : A \vdash x : A} \quad (-I) : \frac{\Gamma, x : A \vdash \gamma, M : B}{\Gamma \vdash \lambda x.M : A \rightarrow B} \quad (\leq) : \frac{\Gamma \vdash \gamma, M : A}{\Gamma \vdash \gamma, M : B} (A \leq B)
\]

\[
(-E) : \frac{\Gamma \vdash \gamma, M : A \rightarrow B \quad \Gamma \vdash \gamma, N : A}{\Gamma \vdash \gamma, MN : B} \quad (\cap I) : \frac{\Gamma \vdash \gamma, M : A_i \quad (\forall i \in n)}{\Gamma \vdash \gamma, M : \cap_n A_i} (n \geq 0)
\]

We will write \( \Gamma \vdash \gamma, M : A \) for statements that are derived using these rules.

## 5 Extentional Intersection and union type assignment for \( \mathcal{T}_\mu \)

In the sections above, we have seen that subject reduction is lost for the systems presented in [11] and [13]. It might be clear that the approach of [13] did not improve the situation in that there also subject expansion was lost, and we will therefore not consider the union-through-negation approach; this brings us back to the approach of [11].

We have seen, in Example ??, three terms \( c_1, c_2 \) and \( c_3 \), such that
are associative and commutative (easy to verify), we will write

\[ c_1 \rightarrow_n c_2, \quad c_1 \rightarrow c_3, \]

\[ c_1 : \Gamma_2 \vdash \Delta_2, \quad c_1 : \Gamma_3 \vdash \Delta_3, \quad c_2 : \Gamma_2 \vdash \Delta_2, \text{ and } c_3 : \Gamma_3 \vdash \Delta_3, \]

\[ \text{but not } c_3 : \Gamma_2 \vdash \Delta_2 \text{ nor } c_2 : \Gamma_3 \vdash \Delta_3. \]

This implies that, even when restricting to either CBN or CBV, subject reduction will collapse. We have also seen that, when adding a full contra-variant \( \leq \) relation on types, with corresponding \((\leq)\) type assignment rules, this problem disappears. This is the direction we will explore in this section, and will show that also this system is closed for expansion; however, we will see, by giving counterexamples in the next section, that it is also not closed for reduction.

**Definition 5.1 (Intersection and Union Types, Contexts)**

1. 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:

\[
A, B ::= \top \mid \bot \mid (A \rightarrow B) \mid (A \cap B) \mid (A \cup B)
\]

2. A context \( \Gamma \) of term-variables (\( \Delta \) 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 consider a pre-order on types which takes into account the idem-potence, commutativity and associativity of the intersection and union type constructors, and defines \( \top \) to be the maximal element, and \( \bot \) to be the minimal.

**Definition 5.2 (Relations on Types)**

1. The relation \( \leq \) is defined as the least pre-order on \( \mathcal{T} \) such that:

\[
\begin{align*}
A \cap B \leq A & \quad A \cap B \leq B \quad A \leq A \cup B \quad B \leq A \cup B \\
A \leq \top & \quad \top \leq \top \leq \top \quad \bot \leq A \quad \bot \rightarrow \bot \leq \bot \\
A \leq C & \text{ and } A \leq B \Rightarrow A \leq B \cap C & \quad A \leq C \text{ and } B \leq C \Rightarrow A \cup B \leq C \\
(A \rightarrow B) \cap (A \rightarrow C) \leq A \rightarrow B \cap C & \quad (A \rightarrow C) \cup (B \rightarrow C) \leq A \cup B \rightarrow C \\
C \leq A \text{ and } B \leq D \Rightarrow A \rightarrow B \leq C \rightarrow D
\end{align*}
\]

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

\[
A \leq B \leq A \Rightarrow A \sim B
\]

Notice that, since the relation defined here is a natural extension of that in Definition 2.2, we are free to use the same symbol.

2. The equivalence relation \( \sim \) on types is defined

\[
A \leq B \leq A \Rightarrow A \sim B
\]

The set \( \mathcal{T} \) can be considered modulo \( \sim \); then \( \leq \) becomes a partial order. Since intersection and union are associative and commutative (easy to verify), we will write \( \bigcap \, A_i \) for \( A_1 \cap \cdots \cap A_n \), and \( \top \) (top) for the empty intersection type, as well as \( \bigcup \, A_i \) for \( A_1 \cup \cdots \cup A_n \), and \( \bot \) (bottom) for the empty union type. The following is easy to show:

**Lemma 5.3**

1. If \( \bigcap_i A_i \leq \bigcap_i B_j \) then, for every \( B_j \), there exists an \( A_{i_j} \) such that \( A_{i_j} \leq B_j \).

2. If \( \bigcup_i A_i \leq \bigcup_i B_j \) then, for every \( A_i \), there exists a \( B_{i_j} \) such that \( A_i \leq B_{i_j} \).

3. \( B \leq \bigcap_i A_i \) if and only if \( B \leq A_i \), for all \( i \in n \).

4. \( B \leq \bigcup_i A_i \) if and only if \( B \leq A_j \), for some \( j \in n \).

5. \( \bigcap_i B_i \leq A \) if and only if \( B_i \leq A \), for some \( i \in n \).

6. \( \bigcup_i B_i \leq A \) if and only if \( B_i \leq A \), for all \( i \in n \).
\textbf{Lemma 5.5} i) Notice that, in particular, \( \lambda \cdot \mu \)

\textbf{Proof:} Straightforward.

\textbf{Definition 5.4} i) For contexts of term-variables \( \Gamma_1, \ldots, \Gamma_n \), the context \( \Gamma_1 \cap \cdots \cap \Gamma_n \) is defined by:\n\( x:A_1 \cap \cdots \cap A_m \in \Gamma_1 \cap \cdots \cap \Gamma_n \) if and only if \( \{ x:A_1, \ldots, x:A_m \} \) is the set of all statements about \( x \) that occur in \( \cup_n \Gamma_i \), where \( \cup \) is set-union. We write \( \Gamma \cap x:A \) for the context of term-variables \( \Gamma \cap \{ x:A \} \), and will write \( \Gamma, x:A \) for \( \Gamma \cap x:A \) when \( x \not\in \Gamma \), and write \( \cap_n \Gamma_i \) for \( \Gamma_1 \cap \cdots \cap \Gamma_n \).

\( \iamb 2 \) Similarly, for contexts of context-variables, \( \Delta_1, \ldots, \Delta_n \), the context \( \Delta_1 \cup \cdots \cup \Delta_n \) is defined by:\n\( \alpha:A_1 \cup \cdots \cup A_m \in \Delta_1 \cup \cdots \cup \Delta_n \) if and only if \( \{ \alpha:A_1, \ldots, \alpha:A_m \} \) is the set of all statements about \( \alpha \) that occur in \( \cup_n \Delta_i \). We write \( \alpha:A \cup \Delta \) for the context of term-variables \( \{ \alpha:A \} \cup \Delta \), and write \( \cup_n \Delta_i \) for \( \Delta_1 \cup \cdots \cup \Delta_n \).

\( \iamb 3 \) We extend \( \leq \) to contexts by: \( \Gamma_1 \leq \Gamma_2 \) if for every \( x:B \in \Gamma_2 \) there exists \( x:A \in \Gamma_2 \) such that \( A \leq B \).

Similarly, we write \( \Delta_1 \leq \Delta_2 \) if for every \( \alpha:A \in \Delta_1 \) there exists \( \alpha:B \in \Delta_2 \) such that \( A \leq B \).

Notice that, in particular, \( \Gamma_1 \leq \Gamma_2 \) when \( \Gamma_2 \subseteq \Gamma_1 \), and that \( \Delta_1 \leq \Delta_2 \) when \( \Delta_1 \subseteq \Delta_2 \).

\textbf{Lemma 5.5} i) \( \Gamma_1 \cap \Gamma_2 \leq \Gamma_i \), for both \( i = 1, 2 \).

ii) \( \Delta_i \leq \Delta_1 \cup \Delta_2 \), for both \( i = 1, 2 \).

iii) \( \text{If } \Gamma \leq \Gamma_1 \cap \Gamma_2, \text{ then } \Gamma \leq \Gamma_i \), for both \( i = 1, 2 \).

iv) \( \Delta_1 \cup \Delta_2 \leq \Delta, \text{ then } \Delta_i \leq \Delta, \text{ for both } i = 1, 2 \).

The type assignment (or better, context assignment) rules as defined below are set up first and for all to preserve intersection types that are assignable to (interpretations of) lambda terms, as well as strive for closure of context assignment with respect to reduction and expansion (i.e. sound and complete).

We will now define our notion of intersection-union type assignment for \( \lambda \mu \).

\textbf{Definition 5.6} (Intersection and Union Typing for \( \lambda \mu \))

\[
\text{(cut)} : \quad \frac{\Gamma \vdash \Delta_1 \quad \Gamma_2 \vdash \Delta_2}{\langle v \mid e \rangle : \Gamma\cap\Gamma_2 \vdash \Delta_1 \cup \Delta_2}
\]

\[
\text{(Ax-t)} : \quad \frac{\Gamma, x:A \vdash \Delta}{\Gamma, x:A \vdash \Delta}
\]

\[
\text{(Ax-c)} : \quad \frac{\Gamma \vdash A \vdash \Delta}{\alpha : A \vdash \alpha : A, \Delta}
\]

\[
\text{(Ax-c)} : \quad \frac{\Gamma \vdash A \vdash \Delta}{\alpha : A \vdash \alpha : A, \Delta}
\]

\[
\text{(-R)} : \quad \frac{\Gamma, x:A \vdash \Delta}{\Gamma, x:A \vdash \Delta}
\]

\[
\text{(µ)} : \quad \frac{\Gamma \vdash \alpha:A, \Delta}{\Gamma \vdash \alpha:A, \Delta}
\]

\[
\text{(Ax-L)} : \quad \frac{\Gamma \vdash A \vdash \Delta}{\Gamma \cap \Gamma_2 \vdash \VarPhi \vdash \Delta}
\]

\[
\text{(-L)} : \quad \frac{\Gamma \vdash A \vdash \Delta}{\Gamma \cap \Gamma_2 \vdash \Delta}
\]

\[
\text{(\cap R)} : \quad \frac{\Gamma \vdash A_i \vdash \Delta_i \quad (\forall i \in \mathbb{N})}{\cap_n \Gamma_i \vdash \cap_n A_i \vdash \Delta_i (n \geq 1)}
\]

\[
\text{(\cup L)} : \quad \frac{n \geq 1}{\cap_n \Gamma_i \vdash \cap_n A_i \vdash \cup_n \Delta_i}
\]

\[
\text{(\cap L)} : \quad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \cap \vdash \Delta}
\]

\[
\text{(\cup R)} : \quad \frac{\Gamma \vdash A \vdash \Delta}{\Gamma \vdash \cup \vdash \Delta (A \leq B)}
\]

\[
\text{(\cup L)} : \quad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \cup \vdash \Delta (B \leq A)}
\]

We call a derivation ending in either \( \text{(cut)} \), \( \text{(Ax-t)} \), \( \text{(Ax-c)} \), \( \text{(-R)} \), \( \text{(-L)} \), \( \text{((µ)} \) or \( \text{(\cap L)} \) a proper derivation.

This system is quite different from the system \( M^{\cap \cup} \) of \cite{11}; we need to point out that 1) our rules \( \text{((L)} \) and \( \text{(R)} \) are there present in an implicit \( \text{(\cap E)} \) that is only applicable to term-variables, and \( \text{((\cup E)} \) that is applicable only to context-variables, and in the rules \( \text{((\cap R)} \) and \( \text{((\cup L)} \), and 2) we deal with \( \text{((T)} \) and \( \text{((\perp)} \).
Notice that the rules \( \land \) and \( \bot \) are special cases of \( \cap \) and \( \cup \) by taking \( n = 0 \) as well and ignoring the fact that an intersection (or union) of an empty collection of sets is the empty set.

This system does not have choice:

**Example 5.7** Take the term \( \mu \delta. \langle \lambda x \mu \beta.(x | \delta) | \delta \rangle \); we can type this term as follows:

\[
\begin{array}{c}
\text{x:A} \vdash x : A \\
\text{\delta:A} \vdash \beta:B, \delta:A \\
\langle x | \delta \rangle : x : A \vdash \beta:B, \delta:A \\
\text{x:A} \vdash \mu \beta.(x | \delta) : B | \delta:A \\
\text{\delta:A} \vdash \lambda x \mu \beta.(x | \delta) : A \rightarrow B | \delta:A \\
\text{\delta:A} \vdash \lambda \delta:A \rightarrow B | \delta:B, \delta:A \rightarrow B \\
\langle \lambda x \mu \beta.(x | \delta) | \delta \rangle : \vdash \delta:A \cup (A \rightarrow B) \\
\text{\delta:A} \vdash \mu \delta. \langle \lambda x \mu \beta.(x | \delta) | \delta \rangle : A \rightarrow B | \\
\end{array}
\]

As with the necessity of intersection to type \( \langle x | x \cdot \alpha \rangle \), we need both types for \( \delta \) to type the whole term. It is therefore impossible to derive either \( \vdash \mu \delta. \langle \lambda x \mu \beta.(x | \delta) | \delta \rangle : A \rightarrow B \) or \( \vdash \mu \delta. \langle \lambda x \mu \beta.(x | \delta) | \delta \rangle : A \rightarrow B \).

**Lemma 5.8 (Generation Lemma)**

1. \( \langle \nu | e \rangle : \Gamma \vdash \Delta \text{ if and only if there exists } A \text{ such that } \Gamma \vdash \nu : A \rightarrow \Delta \text{ and } \Gamma \vdash e : A \rightarrow \Delta. \)
2. \( \Gamma, x : A \vdash x : B | \Delta \text{ if and only if } A \leq B. \)
3. \( \Gamma \vdash \lambda x : A \rightarrow \Delta \text{ if and only if there are } A_i (i \in \text{n}) \text{ such that } \cap_i A_i \leq A, \text{ and for all } i \in \text{n} \text{ there exist } B_i, C_i \text{ such that } A_i = B_i \rightarrow C_i, \text{ and } \Gamma, x : B_i \vdash \nu : C_i | \Delta. \)
4. \( \Gamma \vdash \mu x : A \rightarrow \Delta \text{ if and only if there are } A_i (i \in \text{n}) \text{ such that } \cap_i A_i \leq A, \text{ and } c : \Gamma \vdash \alpha : A_i | \Delta \text{ for all } i \in \text{n}. \)
5. \( \Gamma \vdash \alpha : A \vdash \alpha : B, \Delta \text{ if and only if } A \leq B. \)
6. \( \Gamma \vdash e : A \rightarrow \Delta \text{ if and only if there are } A_i (i \in \text{n}) \text{ such that } A \leq \cup_i A_i, \text{ and for all } i \in \text{n} \text{ there are } B_i, C_i \text{ such that } A_i = B_i \rightarrow C_i, \text{ and } \Gamma \vdash e : C_i | \Delta. \)
7. \( \Gamma \vdash x : A \rightarrow \Delta \text{ if and only if there are } A_i (i \in \text{n}) \text{ such that } A \leq \cup_i A_i, \text{ and, for all } i \in \text{n}, c : \Gamma, x : A_i \vdash \Delta. \)
8. \( \text{If } \Gamma \vdash e : \text{n} \cap_i A_i \vdash \Delta, \text{ then there exists } j \in \text{n} \text{ such that } \Gamma \vdash E : A_j \vdash \Delta. \)
9. \( \text{If } \Gamma \vdash V : \text{n} \cap_i A_i | \Delta, \text{ then there exists } j \in \text{n} \text{ such that } \Gamma \vdash V : A_j | \Delta. \)

**Proof:** Easy.

The following properties hold:

**Lemma 5.9 (Thinning)**

1. \( c : \Gamma \vdash \Delta \text{ and } \Gamma' = \{ x : B \in \Gamma | x \in \text{fv}(c) \} \text{ and } \Delta' = \{ \alpha : B \in \Gamma | \alpha \in \text{fv}(c) \}, \text{ then also } c : \Gamma' \vdash \Delta'. \)

**Proof:** Easy.

**Lemma 5.10 (Weakening)**

1. \( c : \Gamma \vdash \Delta \text{ and } \Gamma' \leq \Gamma \text{ and } \Delta \leq \Delta', \text{ then also } c : \Gamma' \vdash \Delta'. \)
2. \( \text{If } \Gamma \vdash v : A | \Delta \text{ and } \Gamma' \leq \Gamma \text{ and } A \leq B, \text{ and } \Delta \leq \Delta', \text{ then also } \Gamma' \vdash v : B | \Delta'. \)
3. \( \text{If } \Gamma \vdash e : A \rightarrow \Delta \text{ and } \Gamma' \leq \Gamma \text{ and } B \leq A, \text{ and } \Delta \leq \Delta', \text{ then also } \Gamma' \vdash e : B \rightarrow \Delta'. \)

**Proof:** By simultaneous induction (on the structure of derivations).

1. \( \text{If } c : \Gamma \vdash \Delta \text{ then } c = \langle \nu | e \rangle, \text{ and there exists } \mu \gamma, \mu \beta, A_1, A_2 \text{ such that } \Gamma = \Gamma_1 \cap \Gamma_2, \Delta = \Delta_1 \cup \Delta_2, \Gamma_1 \vdash v : A | \Delta_1 \text{ and } \Gamma_2 \vdash e : A \rightarrow \Delta_2. \text{ Then, by induction } \nu, \text{ we have } \langle \nu | e \rangle : \Gamma' \vdash \Delta'. \)
2. \( \text{We have five cases } (\cap) \text{ and } (\land) \text{ are treated together): }\)

\( (A \cdot t) : \text{ then } v = x, \text{ and } x : A \in \Gamma; \text{ since } \Gamma' \leq \Gamma, \text{ there exists } x : C \in \Gamma' \text{ such that } C \leq A \leq B; \text{ by } (A \cdot t), \Gamma' \vdash x : C | \Delta'. \text{ Then by rule } (\leq R), \Gamma' \vdash x : B | \Delta'. \)
there exists an

\[ \top \]

We have five cases.

* \( B = E \rightarrow F \), with \( E \leq C \) and \( D \leq F \): since \( \Gamma', x : E \leq \Gamma, x : C \), and, by induction ??, we get \( \Gamma', x : E \vdash v' : F \mid \Delta' \). Then \( \Gamma' \vdash \lambda x.v' : E \rightarrow F \mid \Delta' \) by rule \((-R)\).

* \( B = \cap_n B_i \) (\( n \geq 0 \)) with \( A \leq B_i \) for all \( i \in \mathbb{N} \) by induction and \((\cap R)\).

* \( B = \cup_n B_i \) (\( n \geq 1 \)), with \( A \leq B_j \) for some \( j \in \mathbb{N} \) by induction and \((\leq R)\).

\((\mu)\) : then \( v = \mu x.c \) and \( c : \Gamma \vdash \alpha : A, \Delta \). Since \( \alpha : A, \Delta \leq \alpha : B, \Delta' \), by induction ??, \( c : \Gamma' \vdash \alpha : B, \Delta' \), and \( \Gamma' \vdash \mu x.c : B \mid \Delta' \) by rule \((\mu)\).

\((\cap R)\) : then \( A = \cap_n A_i \), \( \Gamma = \cap_n \Gamma_i \), \( \Delta = \cup_n \Delta_i \), and \( \Gamma_i \vdash v : A_i \mid \Delta_i \). Since \( \cap_n A_i \leq B \), \( B = \cap_m B_j \), and for every \( j \in m \) there exists an \( i \) such that \( A_j \leq B_j \). Since \( \Gamma \leq \cap_m \Gamma_i \leq \Gamma_i \), and \( \Delta_i \leq \cup_n \Delta_i \leq \Delta' \) for all \( i \in m \) by induction, \( \Gamma' \vdash v : B_j \mid \Delta' \). Then, by \((\cap R)\), also \( \Gamma' \vdash v : \cap_n B_j \mid \Delta' \).

\((\leq R)\) : then there exists \( C \leq A \), such that \( \Gamma \vdash v : C \mid \Delta \). By induction, \( \Gamma' \vdash v : C \mid \Delta' \), and by \((\leq R)\), \( \Gamma' \vdash v : B \mid \Delta' \).

\( \text{iii} \) We have five cases.

\((A x-c)\) : then \( e = \alpha \), and \( \alpha : A \in \Delta \); since \( \Delta \leq \Delta' \), there exists \( \alpha : C \in \Delta' \) such that \( A \leq C \); by \((A x-c)\), \( \Gamma' \vdash \alpha : C \mid \Delta' \). Since \( B \leq C \), by rule \((\leq L)\), also \( \Gamma' \vdash \alpha : B \mid \Delta' \).

\((\rightarrow L)\) : then \( e = v \rightarrow v' \). Then there exist \( \Gamma_1, \Gamma_2, \Delta_1, \Delta_2 \) and \( C, D \) such that \( \Gamma = \Gamma_1 \cap \Gamma_2 \), \( \Delta = \Delta_1 \cup \Delta_2 \), \( A = C \rightarrow D \), and both \( \Gamma_1 \vdash v' \mid \Delta_1 \) and \( \Gamma_2 \vdash e' \mid D \cup \Delta_2 \). We continue by induction on \( \leq \). Since \( B \leq A \), we have

* \( B = E \rightarrow F \), with \( C \leq E \) and \( F \leq D \): Then, by induction ?? \( \Gamma' \vdash v' : E \mid \Delta' \) and by induction ?? \( \Gamma' \vdash e' : F \mid \Delta' \). Then \( \Gamma' \vdash v \cdot e' : E \rightarrow F \mid \Delta' \) by rule \((-L)\).

* \( B = \cap_n B_i \) (\( n \geq 0 \)), with \( B_j \leq A \) for some \( j \in \mathbb{N} \); then by induction, \( \Gamma' \vdash v \cdot e' : B_j \mid \Delta' \), and by \((\leq L)\) also \( \Gamma' \vdash v \cdot e' : B \mid \Delta' \).

* \( B = \cup_n B_i \) (\( n \geq 0 \)) with \( B_j \leq A \) for all \( i \in \mathbb{N} \) by induction and rule \((\cup L)\).

\((\bar{\mu})\) : then \( e = \bar{\mu} y.c \), and \( c : \Gamma, y : A \vdash \Delta \). Since \( \Gamma, x : B \leq \Gamma, x : A \), by induction ??, \( c : \Gamma, y : B \vdash \Delta \), and we get \( \Gamma \vdash \bar{\mu} y.c : B \mid \Delta \) via rule \((\bar{\mu})\).

\((\cup L)\) : then \( A = \cup_n A_i \), \( \Gamma = \cap_n \Gamma_i \), \( \Delta = \cup_n \Delta_i \), and \( \Gamma_i \vdash e : A_i \mid \Delta_i \) for all \( i \in \mathbb{N} \); since \( B \leq \cup_n A_i \), \( B = \cup_m B_j \), and for all \( j \in m \) there exists \( A_j \) such that \( B_j \leq A_j \). Then \( \Gamma' \vdash e : B_j \mid \Delta' \) for all \( j \in m \) by induction, and \( \Gamma \vdash e : B_j \mid \Delta \) by \((\cup L)\).

\((\leq L)\) : then \( A \leq C \), and \( \Gamma \vdash e : C \mid \Delta' \). By induction, \( \Gamma' \vdash e : C \mid \Delta' \), and since \( B \leq A \leq C \), by \((\leq L)\) also \( \Gamma' \vdash e : B \mid \Delta' \).

Notice that the traditional notion of weakening is a consequence of the above lemma, since, for example, \( \Gamma, x : A \leq \Gamma \).

We can show that type assignment is preserved by the interpretation of the \( \lambda \)-calculus in \( \lambda \mu \bar{\mu} \):

**Theorem 5.11** If \( \Gamma \vdash M : A \) then \( \Gamma \vdash [M] : A \).

**Proof:** By easy induction on the structure of derivations in \( \Gamma \).

\((A x)\) : Then \( M = x \), and \( x : A \in \Gamma \); by rule \((A x-t)\), \( \Gamma \vdash x : A \).

\((T)\) : Immediate, by rule \((T)\).

\((-I)\) : Then \( M = \lambda x. N \), \( A = B \rightarrow C \), and \( \Gamma, x : B \vdash N : C \). By induction, \( \Gamma, x : B \vdash [N] : C \); then, by rule \((-R)\), also \( \Gamma \vdash [\lambda x. N] : A \).

\((-E)\) : Then \( M = PQ \), and there is a \( B \) such that \( \Gamma \vdash P : B \rightarrow A \) and \( \Gamma \vdash Q : B \). By induction, both
Proposition 5.12  i) If \( \Gamma \vdash p^1 : B \rightarrow A \) \( \) and \( \Gamma \vdash q^1 : B \), and we can construct:

\[
\begin{array}{c}
\Gamma \vdash q^1 : B \\
\Gamma \vdash p^1 : B \rightarrow A \\
\hline
\Gamma \vdash \{p^1\} q^1, \alpha : A \vdash A
\end{array}
\]

(\( \cap l \)) : Then \( \Gamma \vdash a_i \), and for every \( i \in n \), \( \Gamma \vdash M : A_i \). Then, by induction, \( \Gamma \vdash \nabla M : A_i \) For all

(\( \leq \)) : Then \( \Gamma \vdash M : B \), for some \( B \leq A \). Then, by induction, \( \Gamma \vdash \nabla M : B \), and by rule \( (\leq R) \), also \( \Gamma \vdash \nabla M : A \).

Proof: Without loss of generality, we assume that this the derivation is proper.

i) If \( \Gamma \vdash \lambda x \mu \beta \langle \psi | x, \beta \rangle : A \mid \Delta \), then \( \Gamma \vdash \psi : A \mid \Delta \).

ii) If \( \Gamma \vdash \mu \alpha \langle \psi | \alpha \rangle : A \mid \Delta \), with \( \alpha \) not in \( \psi \), then \( \Gamma \vdash \psi : A \mid \Delta \).

iii) If \( \Gamma \vdash \bar{\mu} x \langle \psi | e \rangle : A \vdash \Delta \), with \( x \) not in \( e \), then \( \Gamma \vdash e : A \vdash \Delta \).

Then, by Lemma 3.5, \( B \leq F \) and \( G \leq D \), and \( E \leq F \rightarrow G \leq B \rightarrow D \leq B \rightarrow C \), so by rule \( (\leq R) \), also \( \Gamma \vdash \psi : B \rightarrow C \mid \Delta \).

ii) If \( \Gamma \vdash \mu \alpha \langle \psi | \alpha \rangle : A \mid \Delta \), then this is derived as follows:

\[
\begin{array}{c}
\Gamma \vdash \psi : B \mid \Delta \\
\Gamma \vdash \alpha : B \vdash \alpha : A, \Delta
\end{array}
\]

Then, by Lemma 3.5, \( B \leq A \), and by rule \( (\leq R) \), also \( \Gamma \vdash \psi : A \mid \Delta \).

iii) If \( \Gamma \vdash \bar{\mu} x \langle \psi | e \rangle : A \vdash \Delta \), then this is derived as follows:

\[
\begin{array}{c}
\Gamma, x : A \vdash x : B \mid \Delta \\
\Gamma \vdash e : B \vdash \Delta
\end{array}
\]

\( \Gamma \vdash \bar{\mu} x \langle \psi | e \rangle : A \vdash \Delta \)
Then, by Lemma 3.5, \( A \leq B \), and by rule \((\leq L)\), \( \Gamma \mid e : A \vdash \Delta \).

The following expansion properties are easy to show:

**Lemma 5.13 (Term Expansion Lemma)**

1) If \( c[v/x] : \Gamma \vdash \Delta \) there exists \( B \in \mathcal{T} \) such that \( c : \Gamma, x : B \vdash \Delta \) and \( \Gamma \vdash v : B \mid \Delta \).

2) If \( \Gamma \vdash v'[v/x] : A \mid \Delta \) then there exists \( B \in \mathcal{T} \) such that \( \Gamma, x : B \vdash v' : A \mid \Delta \) and \( \Gamma \vdash v : B \mid \Delta \).

3) If \( \Gamma \mid e[v/x] : A \vdash \Delta \) then there exists \( B \in \mathcal{T} \) such that \( \Gamma, x : B \vdash e : A \vdash \Delta \) and \( \Gamma \vdash v : B \mid \Delta \).

**Proof:** By simultaneous induction.

1) If \( c[v/x] : \Gamma \vdash \Delta \) then \( c = \langle v' \mid e' \rangle \), and there exists \( C \in \mathcal{T} \) such that \( \Gamma \vdash v'[v/x] : C \mid \Delta \) and \( \Gamma \vdash e'[v/x] : C \mid \Delta \). Then, by induction 2 there exists \( B_1 \) such that \( \Gamma, x : B_1 \vdash v' : C \mid \Delta \) and \( \Gamma \vdash v : B_1 \mid \Delta \) and by induction 3 there exists \( B_2 \) such that \( \Gamma, x : B_2 \vdash e' : C \mid \Delta \) and \( \Gamma \vdash v : B_2 \mid \Delta \). Then, by Lemma 3.5 and rule \((cut)\), we have \( \langle v' \mid e' \rangle : \Gamma, x : B_1 \cap B_2 \vdash \Delta \), and, by rule \((\cap R)\), \( \Gamma \vdash v : B_1 \cap B_2 \mid \Delta \).

2) We have four cases:

   (\( v' = x \)) : Then \( \Gamma \vdash v : A \mid \Delta \). Take \( B = A \); by rule \((Ax-t)\), \( \Gamma, x : A \vdash x : A \mid \Delta \).

   (\( v' = y \neq x \)) : Then, trivially, \( \Gamma \vdash y : A \mid \Delta \). Take \( B = \top \), then by weakening, \( \Gamma, x : \top \vdash y : A \mid \Delta \); by rule \((\top)\), we have \( \Gamma \vdash v : \top \mid \Delta \).

   (\( v' = \lambda y.v'' \)) : Then, by Lemma 3.5, \( A = \cap_i A_i \), and for all \( i \in \mathcal{N} \) there exist \( C_i, D_i, E_i \) such that \( A_i = (C_i \rightarrow D_i) \cup E_i \), and \( \Gamma, y : C_i \vdash v''[v/x] : D_i \mid \Delta \). By induction 2, there exists \( B_i \) such that \( \Gamma, y : C_i, x : B_i \vdash v'' : D_i \mid \Delta \) and \( \Gamma, y : C_i \vdash v : B_i \mid \Delta \). Then, \( \Gamma, x : B_1 \vdash \lambda y.v'' : (C_i \rightarrow D_i) \cup E_i \mid \Delta \) by rules \((\cap R)\) and \((\cup R)\), and \( \Gamma, x : B_i \vdash \lambda y.v'' : A \mid \Delta \) by rule \((\cap R)\); by thinning \( \Gamma \vdash v : B_i \mid \Delta \), and \( \Gamma \vdash v : \cap_i B_i \mid \Delta \) by rule \((\cap R)\).

   (\( v' = \mu \alpha. c \)) : Then by Lemma 3.5, \( A = \cap_i A_i \), and for all \( i \in \mathcal{N} \) there exist \( C_i, D_i \) such that \( A_i = C_i \cup D_i \), and \( c[v/x] : \Gamma \vdash \alpha : C_i, \Delta \). Then by induction \( I \), for all \( i \in \mathcal{N} \), there exists \( B_i \) such that \( c : \Gamma, x : B_i \vdash \alpha : C_i, \Delta \) and \( \Gamma \vdash v : B_i \mid \alpha : A, \Delta \). Then, \( \Gamma, x : \cap_i B_i \vdash \mu \alpha. c : A \mid \Delta \) by rules \((\mu)\), \((\cup R)\), and \((\cap R)\), and \( \Gamma \vdash v : \cap_i B_i \mid \Delta \) by thinning and rule \((\cap R)\).

3) We have three cases:

   (\( e = \alpha \)) : Then \( \Gamma \mid \alpha : A \vdash \Delta \). Take \( B = \top \), then \( \Gamma, x : \top \mid \alpha : A \vdash \Delta \) by weakening, and, by rule \((\top)\), \( \Gamma \vdash v : \top \mid \Delta \).

   (\( e = v' e' \)) : Then by Lemma 3.5, there are \( n \geq 0 \), \( C_i, D_i (i \in \mathcal{N}) \) such that \( A \leq \cup_i (C_i \rightarrow D_i) \), \( \Gamma \vdash v'[v/x] : C_i \mid \Delta \), and \( \Gamma \vdash e'[v/x] : D_i \mid \Delta \) for all \( i \in \mathcal{N} \). Then, by induction 2 there exists \( B_i \) such that \( \Gamma, x : B_i \vdash v' : C_i \mid \Delta \) and \( \Gamma \vdash v : B_i \mid \Delta \) and by induction 3 there exists \( B_2 \) such that both \( \Gamma, x : B_i \vdash e' : D_i \mid \Delta \) and \( \Gamma \vdash v : B_i \mid \Delta \). Then, \( \Gamma, x : \cap_i B_i \vdash v' e' : C_i \mid \Delta \) by rule \((\cap L)\), and \( \Gamma, x : \cup_i (B_i \cap B_i) \vdash v' e' : A \mid \Delta \) by rules \((\cup L)\) and \((\leq L)\); via rule \((\cap L)\) we obtain \( \Gamma \mid \cap_i B_i \vdash B_i \mid \Delta \).

   (\( e = \mu y.c \)) : Then by Lemma 3.5, there are \( n \geq 0 \), \( A_i (i \in \mathcal{N}) \) such that \( A \leq \cup_i A_i \), and \( c[v/x] : \Gamma, y : A_i \vdash \Delta ; \) by induction \( I \) there exists \( B_i \) such that \( c : \Gamma, y : A_i, x : B_i \vdash \Delta \) and \( \Gamma, y : A_i \vdash v : B_i \mid \Delta \). Then, by rules \((\mu)\), \((\cup L)\), and \((\leq L)\), \( \Gamma \mid \mu \alpha. c : A \vdash \alpha \cup_i B_i, \Delta \) and, by thinning and rule \((\cup L)\), \( \Gamma \vdash v : \cup_i B_i \mid \Delta \).

**Lemma 5.14 (Context Expansion Lemma)**

1) If \( c[e/a] : \Gamma \vdash \Delta \) then there exists \( B \in \mathcal{T} \) such that \( c : \Gamma \vdash \alpha : B, \Delta \) and \( \Gamma \mid e : B \vdash \Delta \).

2) If \( \Gamma \vdash v'[e/a] : A \mid \Delta \) then there exists \( B \in \mathcal{T} \) such that \( \Gamma \vdash v' : A \mid \alpha : B, \Delta \) and \( \Gamma \vdash e : B \vdash \Delta \).

3) If \( \Gamma \mid e'[e/a] : A \vdash \Delta \) then there exists \( B \in \mathcal{T} \) such that \( \Gamma \mid e' : A \vdash \alpha : B, \Delta \) and \( \Gamma \vdash e : B \vdash \Delta \).

**Proof:** By simultaneous induction.

1) If \( c[e/a] : \Gamma \vdash \Delta \) then \( c = \langle v' \mid e' \rangle \), and, by Lemma 3.5, there exists \( C \) such that \( \Gamma \vdash v'[e/a] : C \mid \Delta \) and \( \Gamma \vdash e'[e/a] : C \mid \Delta \). Then, by induction 2 there exists \( B_1 \) such that \( \Gamma \vdash v' : C \mid \alpha : B_1, \Delta \) and \( \Gamma \mid e : B_1 \vdash \Delta \) and by induction 3 there exists \( B_2 \) such that \( \Gamma \mid e' : C \vdash \alpha : B_2, \Delta \) and \( \Gamma \mid e : B_2 \vdash \Delta \).
Then, by Lemma 3.10 and rule (cut), we have \( \langle v' | e' \rangle : \Gamma \vdash \alpha : B_1 \cup B_2, \Delta \), and, by rule (\( \cup L \)), \( \Gamma \vdash e : B_1 \cup B_2 \vdash \Delta \).

ii) We have three cases:

\[(v' = x) : \text{Take } B = \bot \text{; by rule (\( \bot \))}, \text{we have } \Gamma \vdash e : \bot \vdash \Delta, \text{and obtain } \Gamma \vdash x : A \parallel \bot, \Delta \text{ by weakening.}\]

\[(v' = \lambda y.v' \parallel) : \text{Then, by Lemma 3.5, there are } n \geq 0, C_i, D_i (i \in \mathbb{N}) \text{ such that } \cap_n (C_i \rightarrow D_i) \subseteq A, \text{ and } \Gamma, y : C_i \vdash v' [e / \alpha] : D_i | \Delta \text{ for all } i \in \mathbb{N}; \text{ by induction 2, there exists } B_i \text{ such that both } \Gamma, y : C_i \vdash v' : D_i | \alpha : B_i, \Delta \text{ and } \Gamma, y : C_i \vdash e : B_i \vdash \Delta. \text{ Then } \Gamma \vdash \lambda y.v' : C_\bot \vdash D_\bot | \alpha : B_\bot, \Delta \text{ by rule (\( \Rightarrow R \))}, \text{ and by thinning } \Gamma \vdash e : B_\bot \vdash \Delta. \text{ Then } \Gamma \vdash \lambda y.v' : A | \alpha : \cup_n B_i, \Delta \text{ by rules (\( \cap R \)) and (\( \leq R \))}, \text{ and by rule (\( \cup L \)), } \Gamma \vdash e : \cup_n B_i \vdash \Delta.\]

\[(v' = \mu \beta.c) : \text{Then, by Lemma 3.5, there are } n \geq 0, A_i (i \in \mathbb{N}) \text{ such that } \cap_n A_i \subseteq A, \text{ and } \Gamma, y : C_i \vdash e [e / \alpha] : D_i | \Delta \text{ for all } i \in \mathbb{N}. \text{ Then, by induction 2 there exists } B_i \text{ such that } \Gamma, y : C_i \vdash e : D_i | \alpha : B_i, \Delta \text{ and by induction 3 there exists } B_i \text{ such that both } \Gamma \vdash e' : C_\bot \vdash D_\bot | \alpha : B_\bot, \Delta \text{ and } \Gamma \vdash e : B_\bot \vdash \Delta. \text{ Then } \Gamma \vdash v' : C_i \vdash D_i | \alpha : B_i \cup B_i \vdash \Delta \text{ by rule (\( \Rightarrow L \))}, \text{ and, by rule (\( \cup L \)) and (\( \leq L \)), } \Gamma \vdash v' : A \parallel \cup_n (B_i \cup B_i), \Delta \text{ via rule (\( \cup L \)) we obtain } \Gamma \vdash e : \cup_n (B_i \cup B_i) \vdash \Delta. \]

\[(v' = \mu \beta.c) : \text{Then by Lemma 3.5, there are } n \geq 0, A_i (i \in \mathbb{N}) \text{ such that } A \subseteq \cup_n A_i, \text{ and } \Gamma \vdash e [e / \alpha] : y : A_i \vdash \Delta; \text{ by induction 1 there exists } B_i \text{ such that } c : \Gamma, y : A_i \vdash \alpha : B_i, \Delta \text{ and } \Gamma, y : A_i \vdash e : B_i \vdash \Delta. \text{ Then, by rules (\( \mu \)), (\( \cup L \)) and (\( \leq L \)), } \Gamma \vdash \mu x : c : A \parallel \cup_n A_i, \Delta \text{, and by thinning and rule (\( \cup L \)), } \Gamma \vdash e : \cup_n B_i \vdash \Delta.\]

So intersections and \( \top \) are crucial for Lemma 3.10, and union and \( \bot \) for Lemma 3.11.

**Theorem 5.15 (Subject Expansion)** Let \( c_1 \rightarrow c_2 : \Gamma \vdash \Delta \text{ then } c_1 : \Gamma \vdash \Delta. \)

**Proof:** By induction on the definition of reduction, where we focus on the logical rules:

\[\langle \lambda x.v_1 | v_2 \parallel e \rangle \rightarrow \langle v_2 | \mu x.\langle v_1 | e \rangle \rangle : \text{Assume } \langle v_2 | \mu x.\langle v_1 | e \rangle \rangle : \Gamma \vdash \Delta, \text{ then this result is derived by:}\]

\[
\begin{array}{c}
\Gamma, x : A_i \vdash v_1 : B_i | \Delta \\
\Gamma \vdash v_2 : A | \Delta \\
\langle v_1 | e \rangle : \Gamma, x : A_i \vdash \Delta \\
\Gamma \vdash \mu x.\langle v_1 | e \rangle : A_i \vdash \Delta \\
\langle v_2 | \mu x.\langle v_1 | e \rangle \rangle : \Gamma \vdash \Delta
\end{array}
\]

\[
\begin{array}{c}
\Gamma \vdash e : B_i \vdash \Delta \\
\end{array}
\]

(\text{notice that we can assume that } x \text{ does not occur free in } v_2 \text{ or } e). \text{ Then } A \subseteq \cup_n A_i, \text{ so } A \subseteq A_j \text{ for
some $j \in \mathbb{N}$, and we can construct

\[
\frac{\Gamma, x:A_j \vdash v_1:B_j | \Delta}{\Gamma \vdash \lambda x.v_1:A \rightarrow B_j | \Delta} \quad \frac{\Gamma \vdash v_2:A | \Delta}{\Gamma \vdash v_2:A_j | \Delta} \quad \frac{\Gamma | e:B_j \vdash \Delta}{\Gamma | v_2 \cdot e:A_j \rightarrow B_j \vdash \Delta}
\]

\[
\langle \lambda x.v_1 | v_2 \cdot e \rangle : \Gamma \vdash \Delta
\]

\[
(\langle v | \bar{\mu}x.c \rangle \rightarrow c[v/x]): \quad \text{By Lemma 3.10, there exists a } B \in T \text{ such that } c : \Gamma, x:B \vdash \Delta, \text{ and } \Gamma \vdash v : B \mid \Delta.
\]

Then we can derive:

\[
\frac{\Gamma \vdash v : B \mid \Delta}{\Gamma, x:B \vdash \Delta} \quad \frac{\Gamma \vdash \bar{\mu}x.c : B \vdash \Delta}{\langle v | \bar{\mu}x.c \rangle : \Gamma, x:B \vdash \Delta}
\]

\[
(\langle \mu \beta.c | e \rangle \rightarrow c[e/\beta]): \quad \text{By Lemma 3.11, there exists a } B \in T \text{ such that } c : \Gamma \vdash \beta:B, \Delta, \text{ and } \Gamma \mid e : B \vdash \Delta.
\]

Then we can derive:

\[
\frac{\Gamma \vdash \mu \beta.c : B \mid \Delta}{\Gamma \vdash e : B \mid \Delta} \quad \frac{\langle \mu \beta.c | e \rangle : \Gamma, x:B \vdash \Delta}{\langle \mu \beta.c | e \rangle : \Gamma, x:B \vdash \Delta}
\]

\[
\square
\]

**Example 5.16** Suppose a union type has been used to type the command $\langle \mu a.c_1 | \bar{\mu}z.c_2 \rangle$, as follows:

\[
\frac{\Gamma \vdash \mu a.c_1 : A \cup B \mid \Delta}{\langle \mu a.c_1 | \bar{\mu}z.c_2 \rangle : \Gamma \vdash \Delta}
\]

\[
\frac{\Gamma \mid \bar{\mu}z.c_2 : A \vdash \Delta}{\Gamma \mid \bar{\mu}z.c_2 : A \cup B \vdash \Delta} \quad (\cup L)
\]

we want to derive the same result for $\langle \mu \delta.c_1 | \bar{\mu}x.\langle x | \bar{\mu}z.c_2 \rangle \rangle$ (notice that this term $\bar{\mu}$-reduces to the term above):

\[
\frac{\Gamma, x:A \vdash x : A \mid \Delta}{\langle x | \bar{\mu}z.c_2 \rangle : \Gamma, x:A \vdash \Delta} \quad \frac{\Gamma, x:B \vdash x : B \mid \Delta}{\langle x | \bar{\mu}z.c_2 \rangle : \Gamma, x:B \vdash \Delta} \quad (\cup L)
\]

\[
\frac{\Gamma \vdash \mu a.c_1 : A \cup B \mid \Delta}{\langle \mu \delta.c_1 | \bar{\mu}x.\langle x | \bar{\mu}z.c_2 \rangle \rangle : \Gamma \vdash \Delta}
\]

As we will show in the next section, we cannot prove a general subject reduction result. However, we can show that the rule $(\rightarrow)$, $\bar{\mu}$-reduction towards an intersection, and $\mu$-reduction towards a union are safe. First we deal with rule $(\rightarrow)$:

**Theorem 5.17** (Soundness for rule $\rightarrow$) If $\langle \lambda x.v_1 | v_2 \cdot e \rangle : \Gamma \vdash \Delta$ then $\langle v_2 | \bar{\mu}x.\langle v_1 | e \rangle \rangle : \Gamma \vdash \Delta$. 

22
Proof: If \( \langle \lambda x. v_1 \mid v_2 \cdot e \rangle : \Gamma \vdash \Delta \), then by Lemma 3.5, this derivation is constructed as follows:

\[
\begin{array}{l}
\vdash D_1^1 & \quad \Gamma, x : C_i \vdash v_1 : D_i \vdash \Delta \\
\vdash D_1^2 & \quad \Gamma \vdash \lambda x. v_1 : C_i \rightarrow D_i \vdash \Delta \\
\vdash D_1^3 & \quad \Gamma \vdash \lambda x. v_1 : A \vdash \Delta \\
\vdash D_1^4 & \quad \Gamma \vdash \lambda x. v_1 : A_i \vdash \Delta \\
\vdash D_1^5 & \quad \Gamma \vdash \lambda x. v_1 : A \vdash \Delta \\
\vdash D_1^6 & \quad \Gamma \vdash \lambda x. v_1 : \cap_i A_i \vdash \Delta \\
\vdash D_1^7 & \quad \Gamma \vdash \lambda x. v_1 : \lambda \Gamma \vdash \Delta \\
\vdash D_1^8 & \quad \Gamma \vdash \lambda x. v_1 : \mu \beta \vdash \Delta \\
\vdash D_1^9 & \quad \Gamma \vdash \lambda x. v_1 : \mu \beta \vdash \Delta \\
\vdash D_1^{10} & \quad \Gamma \vdash \lambda x. v_1 : \mu \beta \vdash \Delta \\
\end{array}
\]

Notice that \( \cap_i (C_i \rightarrow D_i) \leq A \leq \sqcup_i (E_j \rightarrow F_j) \), so, in particular, for every \( i \in \Pi \) there exists a \( j \in \Pi \) such that \( C_i \rightarrow D_i \leq E_j \rightarrow F_j \), so that \( E_j \leq C_i \) and \( D_i \leq F_j \). But then we can derive:

\[
\begin{array}{l}
\vdash D_2^1 & \quad \Gamma, x : C_i \vdash v_1 : D_i \vdash \Delta \\
\vdash D_2^2 & \quad \Gamma, x : C_i \vdash v_1 : \Delta \\
\vdash D_2^3 & \quad \Gamma \vdash v_2 : \Delta \\
\vdash D_2^4 & \quad \Gamma \vdash v_2 : C_i \vdash \Delta \\
\vdash D_2^5 & \quad \Gamma \vdash v_2 : \Delta \\
\vdash D_2^6 & \quad \Gamma \vdash \lambda x. v_1 : \Delta \\
\vdash D_2^7 & \quad \Gamma \vdash \lambda x. v_1 : A \vdash \Delta \\
\vdash D_2^8 & \quad \Gamma \vdash \lambda x. v_1 : \Delta \\
\vdash D_2^9 & \quad \Gamma \vdash \lambda x. v_1 : \Delta \\
\vdash D_2^{10} & \quad \Gamma \vdash \lambda x. v_1 : \Delta \\
\end{array}
\]

\( \langle v_2 \mid \mu x. (\langle v_1 \mid e \rangle) \rangle : \Gamma \vdash \Delta \)

\( \square \)

Unfortunately, we cannot show the same result for the other two main reduction rules, as we will illustrate in the next two examples. This is mainly due to the fact that \( \Gamma \vdash \mu x. c : A \vdash \Delta \) does not always imply \( \vdash \lambda x. c : A \vdash \Delta . \) and \( \Gamma \vdash \mu x. c : A \vdash \Delta \) does not always imply \( \vdash \lambda x. c : A \vdash \Delta . \)

Example 5.18 Take \( c_1 = \langle \mu a. \langle \lambda x \mu b. \langle v \mid \alpha \rangle \mid e \rangle \mid \mu y. \langle y \mid \delta \rangle \rangle \) where \( \alpha \) does not occur in \( v \) or \( e \), and \( c_2 = \langle \lambda x \mu b. \langle v \mid \mu y. \langle y \mid \delta \rangle \rangle \mid e \rangle \rangle \); notice that \( c_1 \rightarrow \mu c_2 \). Assuming we have \( D_1 :: \Gamma \vdash v : A \rightarrow B \vdash \beta \epsilon \), \( D_2 :: \Gamma \vdash v : A \vdash \beta \epsilon \), \( D_3 :: \Gamma \vdash e : C \rightarrow E \vdash \Delta \), and \( D_4 :: \Gamma \vdash e : D \rightarrow F \vdash \Delta \), then we can type \( c_1 \) as follows. First, let \( D_5 \) be

\[
\begin{array}{l}
\vdash y : A \vdash y : A \vdash \delta \vdash B \\
\vdash \langle y \mid \delta \rangle : y : (A \rightarrow B) \rightarrow A | \Delta \vdash \delta : B \\
\vdash \mu y. \langle y \mid \delta \rangle : (A \rightarrow B) \rightarrow A | \Delta \vdash \delta : B \\
\end{array}
\]
then we can construct:

\[
\Gamma \vdash A \rightarrow B \mid \beta : E, \Delta \\
\Gamma, x : C \vdash v : A \rightarrow B \mid \beta : E, \Delta \\
\Gamma, x : C \vdash \mu \beta. (v : A \rightarrow B, \Delta) \mid \beta : E, \Delta \\
\Gamma, x : C \vdash \lambda \mu \beta. (v : A \rightarrow B, \Delta) : C \rightarrow E \mid \alpha : A \rightarrow B, \Delta \\
\Gamma \vdash \lambda \mu \beta. (v : A \rightarrow B, \Delta) : C \rightarrow E \mid \alpha : A \rightarrow B, \Delta \\
\Gamma, x : C \vdash \alpha : A \rightarrow B, \Delta \\
\Gamma, x : C \vdash \beta : E, \Delta \\
\Gamma, x : C \vdash v : A \rightarrow B, \Delta \\
\Gamma, x : C \vdash \mu \beta. (v : A \rightarrow B, \Delta) \mid \beta : E, \Delta \\
\Gamma, x : C \vdash \lambda \mu \beta. (v : A \rightarrow B, \Delta) : C \rightarrow E \mid \alpha : A \rightarrow B, \Delta \\
\Gamma \vdash \lambda \mu \beta. (v : A \rightarrow B, \Delta) : C \rightarrow E \mid \alpha : A \rightarrow B, \Delta \\
\Gamma, x : C \vdash \alpha : A \rightarrow B, \Delta 
\]

Now we expect the derivation for \( c_2 : \Gamma \vdash \delta : B \lor A \rightarrow B \Delta \) to be composed out of \( D_1 \) to \( D_5 \); however, since the intersection type in derivation \( D_5 \) cannot be split, we can at most derive:

\[
\Gamma, x : C \vdash \beta : E, \Delta \\
\Gamma, x : C \vdash v : A \rightarrow B \mid \beta : E, \Delta \\
\Gamma, x : C \vdash \mu \beta. (v : A \rightarrow B, \Delta) \mid \beta : E, \Delta \\
\Gamma, x : C \vdash \lambda \mu \beta. (v : A \rightarrow B, \Delta) : C \rightarrow E \mid \alpha : A \rightarrow B, \Delta \\
\Gamma \vdash \lambda \mu \beta. (v : A \rightarrow B, \Delta) : C \rightarrow E \mid \alpha : A \rightarrow B, \Delta \\
\Gamma, x : C \vdash \alpha : A \rightarrow B, \Delta \\
\Gamma, x : C \vdash \beta : E, \Delta \\
\Gamma, x : C \vdash v : A \rightarrow B, \Delta \\
\Gamma, x : C \vdash \mu \beta. (v : A \rightarrow B, \Delta) \mid \beta : E, \Delta \\
\Gamma, x : C \vdash \lambda \mu \beta. (v : A \rightarrow B, \Delta) : C \rightarrow E \mid \alpha : A \rightarrow B, \Delta \\
\Gamma \vdash \lambda \mu \beta. (v : A \rightarrow B, \Delta) : C \rightarrow E \mid \alpha : A \rightarrow B, \Delta \\
\Gamma, x : C \vdash \alpha : A \rightarrow B, \Delta 
\]

We could only solve this problem via \( (\leq) \) if \( C \cap D \rightarrow E \lor F \) is smaller than \( (C \rightarrow E) \lor (D \rightarrow F) \); this is not the case. We can show \( (C \rightarrow E) \lor (D \rightarrow F) \leq C \cap D \rightarrow E \lor F \), but that is of no use here.

**Example 5.19** Likewise, take \( c_1 = \mu \delta. (\lambda \mu \beta. (x : \delta) \mid \delta) \mid \mu \beta. (v : y \cdot e) \), with \( y \) not free in \( v \) and \( e \), \( c_2 = (v : \mu \delta. (\lambda \mu \beta. (x : \delta) \mid \delta) \cdot e) \) and notice that \( c_1 \vdash \mu c_2 \). Assume we have \( D_1 : \Gamma \vdash v : A \rightarrow C \mid \Delta, \Gamma \vdash e : C \mid \Delta, \Gamma \vdash v : (A \rightarrow B) \rightarrow D \mid \Delta, \Gamma \vdash e : D \mid \Delta \); let \( D_5 \) be constructed as follows:

\[
x : A \vdash x : A \\
\delta : A \vdash \beta : B, \delta : A \\
\langle x : \delta \rangle : x : A \vdash \beta : B, \delta : A \\
x : A \vdash \mu \beta. (x : \delta) : B \mid \beta : A \\
\Gamma \vdash \lambda \mu \beta. (x : \delta) : A \rightarrow B \mid \delta : A \\
\Gamma \vdash \mu \delta. (\lambda \mu \beta. (x : \delta) \mid \delta) \mid \delta : A \rightarrow B \\
\Gamma \vdash \mu \delta. (\lambda \mu \beta. (x : \delta) \mid \delta) : A \cup (A \rightarrow B) \\
\]
then we can derive:

\[
\begin{array}{c}
\frac{\text{D}}{\Gamma \vdash v : A \rightarrow D} & \frac{\text{D}'}{\Gamma \vdash v : A \rightarrow D} \\
\frac{\text{D}_1}{\Gamma \vdash \text{v} : A \rightarrow C \mid \Delta} & \frac{\text{D}_3}{\Gamma \vdash \text{v} : \text{(A} \rightarrow \text{B)} \rightarrow \text{D} \mid \Delta} \\
\frac{\text{y} : A \vdash \text{y} : A}{} & \frac{\text{y} : A \rightarrow B \vdash \text{y} : A \rightarrow B}{} \\
\frac{\Gamma \vdash e : C \mid \Delta}{} & \frac{\Gamma \vdash e : D \mid \Delta}{} \\
\end{array}
\]

\[
\frac{\langle v \mid y \cdot e \rangle : \text{Γ} \vdash A \mid \Delta}{\Gamma \vdash \text{μ} \cdot y. \langle v \mid y \cdot e \rangle : A \mid \Delta} (\text{μL})
\]

Now we would expect the derivation for \(c_2 : \Gamma \vdash \Delta\) to be constructed out of \(\text{D}_1\) to \(\text{D}_5\), but we can’t derive it; since the union type derived in \(\text{D}_5\) cannot be split, we can at most derive:

\[
\begin{array}{c}
\text{D}_1 & \text{D}_3 & \text{D}_5 & \text{D}_2 & \text{D}_4 \\
\end{array}
\]

\[
\frac{\Gamma \vdash \text{v} : A \rightarrow C \mid \Delta}{\frac{\Gamma \vdash \text{v} : (A \rightarrow C) \cup (\text{(A} \rightarrow \text{B)} \rightarrow \text{D}) \mid \Delta}{\vdots}}
\]

\[
\frac{\vdots}{\Gamma \vdash \text{v} : (A \rightarrow C) \cup (\text{(A} \rightarrow \text{B)} \rightarrow \text{D}) \mid \Delta}
\]

\[
\frac{\Gamma \vdash e : C \mid \Delta}{\frac{\Gamma \vdash e : C \cup D \mid \Delta}{\vdots}}
\]

\[
\frac{\vdots}{\Gamma \vdash e : C \cup D \mid \Delta}
\]

\[
\frac{\Gamma \vdash \mu \cdot \delta. \langle \lambda x \mu. \langle x \mid \delta \rangle \mid \delta \rangle : A \cup (A \rightarrow B) \mid \Delta}{\vdots}
\]

\[
\frac{\Gamma \vdash e : A \cup (A \rightarrow B) \mid \Delta}{\vdots}
\]

\[
\frac{\Gamma \vdash e : \text{A} \cup (A \rightarrow B) \mid \Delta}{\vdots}
\]

\[
\frac{\vdots}{\vdots}
\]

\[
\vdots
\]

Notice that, again, \(\leq\) is of no use here: we can show \((A \rightarrow C) \cup (\text{(A} \rightarrow \text{B)} \rightarrow \text{D}) \leq A \cup (A \rightarrow B) \rightarrow C \cup D\), but not the needed \(A \cup (A \rightarrow B) \rightarrow C \cup D \leq (A \rightarrow C) \cup (\text{(A} \rightarrow \text{B)} \rightarrow \text{D})\).

Looking at the essence of the examples, it seems that performing a \(\mu\)-reduction towards an intersection type, or performing a \(\mu\)-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:

\[
\begin{array}{c}
\text{D}_i & \text{D'} & \text{D}
\end{array}
\]

\[
\frac{c : \Gamma_i \vdash \alpha : A_i \mid \Delta_i}{\vdots}
\]

\[
\frac{\vdots}{\vdots}
\]

\[
\frac{\vdots}{\vdots}
\]

\[
\begin{array}{c}
\text{D}_i & \text{D'} & \text{D}_i & \text{D}_i
\end{array}
\]

\[
\frac{c : \Gamma_i \vdash \alpha : A_i \mid \Delta_i}{\vdots}
\]

\[
\frac{\vdots}{\vdots}
\]

\[
\frac{\vdots}{\vdots}
\]

\[
\frac{\vdots}{\vdots}
\]

Only the first case is problematic, since from the second we get \(c[e / \alpha] : \cap_n \Gamma_i \vdash \cap_n \Delta_i \cup \Delta\) by Lemma 4.4. For the first case, we normally cannot remove any of the types in \(\cap_n A_i\) in the derivation \(\text{D}\), so we cannot safely propagate \(\text{D}\) into the various \(\text{D}_i\). So this is the problem case that should be caught.

Similarly, for union, we have the situations:

\[
\begin{array}{c}
\text{D} & \text{D'} & \text{D} & \text{D}
\end{array}
\]

\[
\frac{c : \Gamma_i \vdash x : A_i \mid \Delta_i}{\vdots}
\]

\[
\frac{\vdots}{\vdots}
\]

\[
\frac{\vdots}{\vdots}
\]

\[
\frac{\vdots}{\vdots}
\]

\[
\frac{\vdots}{\vdots}
\]

\[
\frac{\vdots}{\vdots}
\]

\[
\frac{\vdots}{\vdots}
\]

\[
\frac{\vdots}{\vdots}
\]

\[
\frac{\vdots}{\vdots}
\]

\[
\frac{\vdots}{\vdots}
\]

\[
\frac{\vdots}{\vdots}
\]

\[
\frac{\vdots}{\vdots}
\]

\[
\frac{\vdots}{\vdots}
\]
As above, only the second situation is possibly problematic, and propagating \( D \) into the \( D_i \) should be avoided.

Summarising, it is problematic to: 1) perform a \( \mu \)-reduction towards an intersection type, introduced via \( (\cap R) \), or 2) perform a \( \bar{\mu} \)-reduction towards a union type, introduced via \( (\cup L) \). Below, we will define a restriction of the notion of type assignment that will avoid these reductions.

6 Sound restrictions

In this section we will investigate if we can at least partially recover the soundness property by limiting reduction; we will look at both CBV and CBN. We will recover from the subject reduction only partially, in that we will obtain a soundness result for both notions of reduction, but at the price of sacrificing completeness.

The general idea will be to make sure that a reduction towards an introduced intersection will be blocked by allowing rule \( (\cap R) \) only for values; that way, if \( \langle \nu | e \rangle \) is typeable, and \( (\cap R) \) is used to type \( \nu \), then \( \nu \) is not a \( \mu \)-abstraction, so a reduction towards \( \nu \) is excluded. Likewise, a reduction towards an introduced union will be blocked by allowing rule \( (\cup L) \) only for slots. In fact, we will enforce witness reduction by modifying the system in such a way that

\[
\Gamma \vdash \mu a : C \mid \Delta \\
\text{c : } \Gamma \vdash a : C, \Delta
\]

becomes admissible.

It might seem tempting to define the system immediately with both these restrictions in place. However, it would be too restrictive: for example, we could not encode the lambda calculus whilst preserving assignable intersection types, whatever system used there: to type \( (\lambda x.xx)(yy) \), for example, we must derive

\[
\begin{array}{c}
y : C \rightarrow A \rightarrow B \vdash \gamma, y : C \rightarrow A \\
x : A \rightarrow B \vdash x : A \rightarrow B
\end{array}
\]

\[
\begin{array}{c}
y : C \rightarrow A \rightarrow B \vdash \gamma, y : C \rightarrow A \\
\end{array}
\]

\[
\begin{array}{c}
x : (A \rightarrow B) \cap A \vdash xx : B
\end{array}
\]

\[
\begin{array}{c}
y : (C \rightarrow A) \cap C \vdash yy : A
\end{array}
\]

\[
\begin{array}{c}
y : (C \rightarrow A) \cap (C \rightarrow A) \cap (A \rightarrow B) \cap A \\
\end{array}
\]

\[
\begin{array}{c}
\langle y | y \cdot \gamma \rangle : y : (A \rightarrow B) \cap A
\end{array}
\]

\[
\begin{array}{c}
\mu \gamma : y : (A \rightarrow B) \cap A
\end{array}
\]

\[
\begin{array}{c}
y : D \vdash y : D
\end{array}
\]

\[
\begin{array}{c}
y : D \vdash y : C \vdash y : A \rightarrow B \cap A
\end{array}
\]

\[
\begin{array}{c}
y : D \vdash y : A \rightarrow B \cap A
\end{array}
\]

\[
\begin{array}{c}
\langle y | y \cdot \gamma \rangle : y : (A \rightarrow B) \cap A
\end{array}
\]

\[
\begin{array}{c}
\mu \gamma : y : (A \rightarrow B) \cap A
\end{array}
\]

Notice that \( (C \rightarrow A) \cap (C \rightarrow A) \leq C \rightarrow (A \rightarrow B) \cap A \).
Notice that $\bar{(\lambda x.xx)(yy)}$ reduces as follows:

$$\mu\alpha. (\lambda x.\mu\beta. (x | x \beta) | \mu\gamma. (y | y \gamma) \cdot \alpha) \rightarrow \mu\alpha. (\mu\gamma. (y | y \gamma) | \tilde{\mu} x. (x | x \alpha))$$

In CBV the last term reduces to $\mu\alpha. (y | y \tilde{\mu} x. (x | x \alpha))$, and in CBN to $\mu\alpha. (y | y \mu\gamma. (y | y \gamma) \cdot \alpha)$ (these are essentially the terms $c_1$, $c_2$ and $c_3$ of Example ??).

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

For CBV, we take the dual approach, and restrict the use of $(\cap R)$. To avoid a $\mu$-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 $\langle v | \tilde{\mu} x.c \rangle$ into a union, which is solved by the fact that then $v$ is a value.

**Definition 6.1 (CBN Intersection and Union Typing for $\bar{\lambda\mu\tilde{\mu}}$)** The notion of type assignment $\vdash_n$ is defined as $\vdash$, by changing rule $(\cup L)$ into:

$$(\cup L): \quad \frac{\Gamma_i \vdash E : A_i \vdash_n \Delta_i \quad (\forall i \in n) \quad (n \geq 1)}{\cap_n \Gamma_i \vdash E : \cup_n A_i \vdash_n \cup_n \Delta_i}$$

For CBV, we define:

**Definition 6.2 (CBV Intersection and Union Type Assignment for $\bar{\lambda\mu\tilde{\mu}}$)** The notion of type assignment $\vdash_v$ is defined as $\vdash$, by changing rule $(\cap R)$ into:

$$(\cap R): \quad \frac{\Gamma_i \vdash V : A_i \vdash \Delta_i \quad (\forall i \in n) \quad (n \geq 0)}{\cap_n \Gamma_i \vdash V : \cap_n A_i \vdash \cap_n \Delta_i}$$

That subject-expansion now fails is immediate, since the condition on these rules (either the term is a value or the context is a slot) are not preserved by expansion.

To show that these solution indeed yield sound solutions, we first need to check the following properties:

**Lemma 6.3 (Term Substitution Lemma)** Let $B \in \mathcal{T}$.

i) If $c : \Gamma, x : B \vdash \Delta$ and $\Gamma \vdash v : B | \Delta$, then $c[v/x] : \Gamma \vdash \Delta$.

ii) If $\Gamma, x : B \vdash v' : A | \Delta$ and $\Gamma \vdash v : B | \Delta$, then $\Gamma \vdash v'[v/x] : A | \Delta$.

iii) If $\Gamma, x : B | e : A \vdash \Delta$ and $\Gamma \vdash v : B | \Delta$, then $\Gamma \vdash e[v/x] : A \vdash \Delta$.

**Proof:** By simultaneous induction.

i) Then $c = \langle v' | e' \rangle$, and, by Lemma 3.5, we have both $\Gamma, x : B \vdash v' : A | \Delta$ and $\Gamma, x : B | e' : A \vdash \Delta$ for some $A$. Then by induction 2 and 3, both $\Gamma \vdash v'[v/x] : A | \Delta$ and $\Gamma \vdash e'[v/x] : A \vdash \Delta$, and $c[v/x] : \Gamma \vdash \Delta$ by rule $(cut)$.

ii) We have four cases:

1. $(v' = x)$: Then, by Lemma 3.5, $B \leq A$. By rule $($R$)$, also $\Gamma \vdash v : A | \Delta$.

2. $(v' = y \neq x)$: By thinning, $\Gamma \vdash y : A \vdash \Delta$.

3. $(v' = \lambda y.y''')$: Then, by Lemma 3.5, there are $n \geq 0, C_i, D_i (i \in n)$ such that $\cap_n (C_i \rightarrow D_i) \leq A$, and $\Gamma, x : B | y : C_i \vdash v''' : D_i | \Delta$ for all $i \in n$. By induction 2, $\Gamma, y : C_i \vdash v''[v/x] : D_i | \Delta$, so by rule $($R$)$ also $\Gamma \vdash (\lambda y.y''')[v/x] : C_i \rightarrow D_i | \Delta$, and $\Gamma \vdash (\lambda y.y''')[v/x] : A | \Delta$ by rules $($R$)$ and $(R)$.
(v' = μα.c) : Then, by Lemma 3.5, there are n ≥ 0, A_i (i ∈ ℤ) such that ∨ₙ A_i ≤ A, and c : Γ, x : B ⊢ α : A_i, Δ. By induction I, c[ν/x] : Γ ⊢ α : A_i, Δ, so Γ ⊢ (μα.c)[y/x] : A_i | Δ by rule (μ), and the result follows by rules (∨R) and (≤R).

iii) We have three cases:

(e = α) : By thinning, Γ | α : A ⊳ Δ.

(e = ν'·e') : Then, by Lemma 3.5, there are n ≥ 0, C_i, D_i (i ∈ ℤ) such that A ≤ ∨ₙ (C_i → D_i), and both Γ, x : B ⊢ ν' : C_i | Δ and Γ, x : B ⊢ e' : D_i | Δ for all i ∈ ℤ. We have Γ ⊢ ν'[ν/x] : C_i | Δ and Γ | (ν'·e')[ν/x] : C_i → D_i | Δ by rule (→L), and the result follows from rules (∨L) and (≤L).

(e = μy.c) : Then, by Lemma 3.5, there are n ≥ 0, A_i (i ∈ ℤ) such that A ≤ ∨ₙ A_i, and for all i ∈ ℤ, c : Γ, x : B, y : A_i ⊳ Δ. By induction I, c[ν/x] : Γ, y : A_i ⊳ Δ, so, by rule (μi), Γ | (μy.c)[ν/x] : A_i | Δ, and the result follows from rules (∨L) and (≤L).

Lemma 6.4 (Context Substitution Lemma) Let B ∈ T.

i) If c : Γ ⊢ α : B, Δ and Γ | e : B | ≤ Δ, then c[e/α] : Γ ⊳ Δ.

ii) If Γ ⊢ ν : α : B, Δ and Γ | e : B ⊳ Δ, then Γ ⊢ ν[e/α] : A | Δ.

iii) If Γ | e' : A ⊳ α : B, Δ and Γ | e : B ⊳ Δ, then Γ | e'[e/α] : A | Δ.

Proof: By simultaneous induction.

i) Then, by Lemma 3.5, c = (ν'[e'/e]), and we have both Γ ⊢ ν' : α : B, Δ and Γ | e' : A ⊳ α : B, Δ for some A. Then by induction 2 and 3, both Γ ⊢ ν'[e/α] : A | Δ and Γ | e'[e/α] : A ⊳ Δ, and c[e/α] : Γ ⊳ Δ by rule (cut).

ii) We have three cases:

(v = x) : By thinning, Γ ⊢ x : A | Δ.

(v = λy.ν') : Then, by Lemma 3.5, there are n ≥ 0, C_i, D_i (i ∈ ℤ) such that ∨ₙ (C_i → D_i) ≤ A, and Γ, x : B, y : C_i ⊢ ν'' : D_i | Δ for all i ∈ ℤ. By induction 2, Γ, y : C_i ⊢ ν''[e/α] : D_i | Δ, so by rule (→L) also Γ ⊢ (λy.ν'')[e/α] : C_i → D_i | Δ, and the result follows from rules (∨R) and (≤R).

(v = μβ.c) : Then, by Lemma 3.5, there are n ≥ 0, A_i (i ∈ ℤ) such that ∨ₙ A_i ≤ A, and c : Γ ⊢ β : A_i, α : B, Δ for all i ∈ ℤ. By induction I, c[ν/e/α] : Γ ⊢ β : A_i, Δ, so, by rule (μi), Γ ⊢ (μβ.c)[ν/α] : A_i | Δ, and the result follows from rules (∨R) and (≤R).

iii) We have four cases:

(e = α) : Then, by Lemma 3.5, A ≤ B. By rule (≤L), also Γ | e : A ⊳ Δ.

(e = β ≠ α) : By thinning, Γ | β : A ⊳ Δ.

(e = ν'·e') : Then, by Lemma 3.5, there are n ≥ 0, C_i, D_i (i ∈ ℤ) such that A ≤ ∨ₙ (C_i → D_i), and both Γ ⊢ ν' : C_i | α : B, Δ and Γ | e' : D_i | α : B, Δ. Then Γ ⊢ ν'[ν/e/α] : C_i | Δ, by induction 2, and Γ | e'[e/α] : D_i | Δ, by induction 3, so Γ | (ν'·e')[ν/e/α] : C_i → D_i | Δ by rule (→L), and the result follows from rules (∨R) and (≤R).

(e = μy.c) : Then, by Lemma 3.5, there are n ≥ 0, A_i for each i ∈ ℤ such that A ≤ ∨ₙ A_i, c : Γ, y : A_i ⊢ α : B, Δ for all i ∈ ℤ, and by induction I, c[e/α] : Γ, y : A_i ⊳ Δ, so, by rule (μi), Γ | (μy.c)[ν/e/α] : A_i | Δ, and the result follows from rules (∨R) and (≤R).

Notice that these two lemmata hold for all notions of type assignment, i.e. that of Definition 3.4, 4.1, and 4.2.

Using these two results, we can now show:

**Theorem 6.5 (Soundness for →N)** If c_1 : Γ ⊳ N ⊳ N and c_2 : N ⊳ c_2, then c_2 : Γ ⊳ N, Δ.

Proof: We focus on the main reduction rules:

(→) : By Theorem 3.13.
(μN) : Then \( c_1 = \langle μa.c | E \rangle \), and \( c_2 = \langle E/a \rangle \). Now, by Lemma 3.5, there exists \( A \) such that \( Γ \vdash μa.c : A \vdash Δ \) and \( Γ : E : A \vdash Δ \), and there are \( A_i (i \in \mathbb{N}) \), with \( ∩_i A_i \leq A \), and \( c : Γ \vdash a:A_i Δ \) for all \( i \in \mathbb{N} \). Now either:

\( (n ≥ 1) \) : by Lemma 3.5, since \( E \) is a slot, there exists \( j \in \mathbb{N} \) such that \( Γ : E : A_j \vdash Δ \), or

\( (n = 1) \) : then \( c : Γ \vdash a:A_1 Δ \).

In either case, by Lemma 4.4, \( c[E/a] : Γ \vdash Δ \).

(μV) : Then \( c_1 = \langle V/μx.c \rangle \), and \( c_2 = \langle V[x/x] \rangle \). By Lemma 3.5, there exists \( A \) such that \( Γ \vdash V : A \vdash Δ \) and \( Γ \vdash μx.c : A \vdash Δ \), and there are \( A_i (i \in \mathbb{N}) \) such that \( ∪_i A_i \leq A \), and \( c : Γ, x:A_i Δ \). Now since \( (∪L) \) has been restricted to slots, \( n = 1 \), so \( c : Γ, x:A \vdash Δ \), and by Lemma 4.3, \( c[V/x] : Γ \vdash Δ \).

**Theorem 6.6 (Soundness for \( \rightarrow_v \) in \( \vdash_v \))** If \( c : Γ \vdash Δ \) and \( c \rightarrow_v c' \), then \( c' : Γ \vdash Δ \).

**Proof:** By induction on length of reduction sequence; we focus on the main reduction rules.

\( (→) \) : By Theorem 3.13.

(μ) : Then \( c_1 = \langle μa.c | e \rangle \), and \( c_2 = \langle e/a \rangle \). By Lemma 3.5, there exists \( A \) such that \( Γ \vdash μa.c : A \vdash Δ \) and \( Γ : e : A \vdash Δ \), and there are \( A_i (i \in \mathbb{N}) \), with \( ∩_i A_i \leq A \), and \( c : Γ \vdash a:A_i Δ \) for all \( i \in \mathbb{N} \). Since \( (∩R) \) has been restricted to values, \( n = 1 \), so \( c : Γ \vdash a:A_1 Δ \), and by Lemma 4.4, \( c[e/a] : Γ \vdash Δ \).

(μV) : Then \( c_1 = \langle V/μx.c \rangle \), and \( c_2 = \langle V[x/x] \rangle \). By Lemma 3.5, there exists \( A \) such that \( Γ \vdash V : A \vdash Δ \) and \( Γ \vdash μx.c : A \vdash Δ \), and there are \( A_i (i \in \mathbb{N}) \) such that \( ∩_i A_i \leq A \), and \( c : Γ, x:A_i Δ \).

Now either:

\( (n ≥ 1) \) : by Lemma 3.5, since \( V \) is a value, there exists \( j \in \mathbb{N} \) such that \( Γ \vdash V : A_j \vdash Δ \), or

\( (n = 1) \) : then \( c : Γ, x:A \vdash Δ \).

In either case, by Lemma 4.3, \( c[V/x] : Γ \vdash Δ \).

As for the system suggested in [17] (there only the intersection and union rules for a CBN and CBV restriction are given), the added rules are:

\[
\begin{align*}
(∩L) & : \frac{Γ \vdash e : A_1 \vdash Δ \quad Γ \vdash e : A_1 ∩ A_2 \vdash Δ \quad (i = 1, 2)}{Γ \vdash e : A_1 ∩ A_2 \vdash Δ} \\
(∪L) & : \frac{Γ \vdash e : A_1 \vdash Δ \quad Γ \vdash e : A_2 \vdash Δ \quad (i = 1, 2)}{Γ \vdash e : A_1 ∪ A_2 \vdash Δ}
\end{align*}
\]

With these rules, none of the above result can be achieved; for example, there cannot derive \( ⊢ \langle λx.xx \rangle : (A → B) ∩ A → B \), so cannot type all normal forms. Notice the role of intersection on term variables and union on context variables in Theorem 3.12; we cannot prove such a property for the system of [17].

### 7 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 \( \overline{\lambda μ} \). In fact, allowing intersection and union for activated formulae only, actually leans strongly on the similarity between those type constructs and the logical \& and ∨.

An alternative would be to allow union and intersection to be assigned also to inactive statements, by adding a variant of \( (∪L) \) for \( c : Γ \vdash Δ \) and \( Γ \vdash v : A \vdash Δ \), and of \( (∩R) \) for \( c : Γ \vdash Δ \) and \( Γ \vdash e : A \vdash Δ \). However, this would not solve the subject reduction problem at all (and very likely make it worse). The approach followed (essentially, since it deals with a different calculus) in [4], is not to depend on the logical foundation at all any more, and to allow \( (∪L) \) and \( (∩R) \) only for commands:

\[
\begin{align*}
(∩R) & : \frac{c : Γ_i \vdash a : A_i Δ_i \quad (i \in \mathbb{N})}{c : ∩_i Γ_i \vdash a : ∩_i A_i ∩_i Δ_i \quad (n ≥ 0)} \\
(∪L) & : \frac{c : Γ_i, x : A_i Δ_i \quad (i \in \mathbb{N})}{c : ∪_i Γ_i, x : ∪_i A_i \vdash ∪_i Δ_i \quad (n ≥ 0)}
\end{align*}
\]
It is still possible then to preserve the types assignable in the \( \Gamma \)-system, but we would need a different interpretation function.

We have seen that it is straightforward to define a natural notion of type assignment to the sequent calculus \( \lambda \mu \tilde{\mu} \) 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 \( \lambda \)-calculus, combining union and intersection types breaks the soundness of the system; unlike the \( \lambda \)-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.

Although this does not affect the characterisation of strong normalisation, it does have negative implications with the respect to the obtainable results. For example, it is now impossible to define a filter-style semantics [8] using intersection and union types for \( \lambda \mu \tilde{\mu} \), even for the two confluent sub-reduction systems CBV and CBN. We will investigate the other standard properties, like characterisation of strong normalisation, approximation and head-normalisation in future work.

Acknowledgements

I would like to thank Silvia Ghilezan, Pierre Lescanne, Mariangiola Dezani, 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] S. van Bakel. Rank 2 Intersection Type Assignment in Term Rewriting Systems. Fundamenta Informaticae, 2(26):141–166, 1996.
[4] S. van Bakel. Intersection and Union Types for \( \chi \). In Electronic Proceedings of International Workshop Intersection Types and Related Systems 2004 (ITRS’04), Turku, Finland, Electronic Notes in Theoretical Computer Science, 2004.
[5] S. van Bakel. Reduction in \( \chi \) does not agree with Intersection and Union Types. In Electronic Proceedings of International Workshop Intersection Types and Related Systems 2008 (ITRS’08), Turin, Italy, 2008.
[6] F. Barbanera, M. Dezani-Ciancaglini, and U. de’ Liguoro. Intersection and Union Types: Syntax and Semantics. Information and Computation, 119(2):202–230, 1995.
[7] H. Barendregt. The Lambda Calculus: its Syntax and Semantics. North-Holland, Amsterdam, revised edition, 1984.
[8] H. Barendregt, M. Coppo, and M. Dezani-Ciancaglini. A filter lambda model and the completeness of type assignment. The journal of Symbolic Logic, 48(4):931–940, 1983.
[9] R. Bloo and K.H. Rose. Preservation of strong normalisation in named lambda calculi with explicit substitution and garbage collection. In CSN’95 – Computer Science in the Netherlands, pages 62–72, 1995.
[10] P.-L. Curien and H. Herbelin. The Duality of Computation. In Proceedings of the 5th ACM SIGPLAN International Conference on Functional Programming (ICFP’00), pages 233–243. ACM, 2000.
[11] R. Davies and F. Pfenning. Intersection types and computational effects. In Proceedings of the ACM Sigplan International Conference on Functional Programming (ICFP’00), volume 35.9 of ACM Sigplan Notices, pages 198–208. ACM Press, 2000.
[12] D. Dougherty, S. Ghilezan, and P. Lescanne. Intersection and Union Types in the \( \lambda \mu \tilde{\mu} \)-calculus. In Electronic Proceedings of International Workshop Intersection Types and Related Systems 2004 (ITRS’04), Turku, Finland, Electronic Notes in Theoretical Computer Science, 2004.
[13] D. Dougherty, S. Ghilezan, and P. Lescanne. Strong Normalization of the Dual Classical Sequent Calculus. In *Proceedings of LPAR’05*, Lecture Notes in Computer Science, pages 169–182, 2005.

[14] 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*, 2008.

[15] J. Dunfield and F. Pfenning. Tridirectional typechecking. In *Proceedings of SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL’04)*, pages 281–292. ACM Press, 2000.

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

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

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

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

[20] T. Jim. Rank 2 type systems and recursive definitions. Technical Memorandum MIT/LCS/TM-531, Laboratory for Computer Science Massachusetts Institute of Technology, 1995.

[21] A. Kfoury and J. Wells. Principality and decidable type inference for finite-rank intersection types. In *Proceedings of POPL’99: 26th ACM Symposium on the Principles of Programming. Languages*, pages 161–174, 1999.

[22] S.C. Kleene. *Introduction to Metamathematics*. Études et Recherches en Informatique. North Holland, Amsterdam, 1952.

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

[24] D. Leivant. Polymorphic Type Inference. In *Proceedings 10th ACM Symposium on Principles of Programming Languages*, pages 88–98, Austin Texas, 1983.

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

[26] Philip 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.

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