Intersection and Union Types for $\mathcal{X}$

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 presents a notion of intersection and union type assignment for the calculus $\mathcal{X}$, a substitution free language that can be used to describe the behaviour of functional programming languages at a very low level of granularity, and has first been defined in [14,5]. $\mathcal{X}$ has been designed to give a Curry-Howard-de Bruijn correspondence to the sequent calculus for classical logic.

In this paper we will define a notion of sequent-style intersection type assignment on $\mathcal{X}$ that needs union types, and show that this notion is closed for both subject-reduction and subject-expansion. We will also show that it is an extension of the Strict system for λ-calculus.

Introduction

This paper will present a notion of intersection and union type assignment for the (untyped) calculus $\mathcal{X}$, as first defined in [14] and later extensively studied in [5]. The origin of $\mathcal{X}$ lies within the quest for a language designed to give a Curry-Howard-de Bruijn correspondence to the sequent calculus for Classical Logic. $\mathcal{X}$ is defined as a substitution-free programming language that, perhaps surprisingly, is extremely well equipped to describe the behaviour of functional programming languages at a very low level of granularity (see [14,5]).

The Curry-Howard property strongly links typeable programs and provable theorems, and can be understood as follows:

> (CURRY-HOWARD ISOMORPHISM) "*Terms as Proofs, Types as Propositions.*" Let M be a term, and A a type, then M is of type A if and only if A, read as a logical formula, is provable in the corresponding logic, using a proof which structure corresponds to M.

[1] Email: svb@doc.ic.ac.uk

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

A sequent style implicative classical logic can be defined by:

$$(ax): \frac{}{\Gamma, A \vdash A, \Delta} \qquad (cut): \frac{\Gamma \vdash A, \Delta \quad \Gamma, A \vdash \Delta}{\Gamma \vdash \Delta}$$

$$(LI): \frac{\Gamma \vdash A, \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A{\rightarrow}B \vdash \Delta} \qquad (RI): \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A{\rightarrow}B, \Delta}$$

Starting from different approaches in that area [10,15], in [14] the calculus $\mathcal{X}$ was introduced, and shown to be equivalent to the $\overline{\lambda}\mu\tilde{\mu}$-calculus of [10] in terms of expressivity. Using this correspondence, a strong normalization result is shown for $\overline{\lambda}\mu\tilde{\mu}$. In fact, [14] did not study any property of untyped $\mathcal{X}$, but focused only on its type aspects in connection with the sequent calculus.

As far as the Curry-Howard isomorphism is concerned, $\mathcal{X}$ stands out in that it is the first calculus to achieve that in full for a classical logic. For example, in $\overline{\lambda}\mu\tilde{\mu}$, all provable propositions can be inhabited, but not all terms correspond to proofs, and in $\lambda\mu$, not all proofs can be represented, since there reduction is confluent.

When studying $\mathcal{X}$ as an untyped language, soon the unexpected special properties surfaced: it became apparent that $\mathcal{X}$ provides an excellent general purpose machine, very well suited to encode various calculi (for details, see [5]). Amongst the calculi studied in that paper, the Calculus of Explicit Substitutions $\lambda\mathbf{x}$ stands out. In fact, a 'subatomic' level was reached by decomposing explicit substitutions into smaller components. Even more, the calculus is actually symmetric [7]; the '*cut*', represented by $P\widehat{\alpha} \dagger \widehat{x}Q$ represents, in a sense, the explicit substitution of P for x in Q, but also that of Q for α in P.

Perhaps the main feature of $\mathcal{X}$ is that it constitutes a *variable* and *substitution*-free method of computation. Rather than having variables like x representing places where terms can be inserted, in $\mathcal{X}$ the symbol x represents a *socket*, to which a term can be *attached*. The definition of reduction on $\mathcal{X}$ shows nicely how the interaction between the two subtly and gently percolates through the terms.

Although the origin of $\mathcal{X}$ is a logic, and one could expect it to be close to λ-calculus, it is in fact specified as a *conditional term rewriting system*; the only non-standard aspect is that it treats *three* different classes of variables (for plugs, sockets, and nets).

In this paper we will treat $\mathcal{X}$ as a pure, untyped calculus, and ignore its origin in that we define a notion of sequent-style intersection type assignment on $\mathcal{X}$; intersection types are notorious for lacking a solid background in logic. We will see that, in view of the special nature of $\mathcal{X}$ as an input-output calculus, we will need to add also union types. The notion of intersection type assignment for $\mathcal{X}$ as defined in this paper is inspired by the system of [12] (the precise relation between the two –through the interpretation functions as defined in [14,5]– needs to be studied, and is left for further research).

It is a conservative extension of the Strict Intersection Type Assignment System of [1] (see also [2,3]), in that lambda terms typeable in that system translate to $\mathcal{X}$-nets, while preserving the type. It is also a natural extension of the system

considered in [5], i.e. the basic implicative system for Classical Logic, but extended with (strict) intersection and union types and the type constant $\top$. The main results of this paper are that this notion is closed for both subject-reduction and subject-expansion.

As was the case for systems with intersection types for λ-calculus [9,2], in order to get a notion of type assignment that is closed for η-reduction, we would need to introduce a $\leq$-relation on types which is contra-variant in the arrow; this is not part of the present system.

This paper is constructed as follows. Section 1 presents the syntax and reduction system of the calculus $\mathcal{X}$. In Section 2 we define the basic system of type assignment for $\mathcal{X}$, then in Section 3 we will embed λ-calculus into $\mathcal{X}$ and discuss η-reduction, and, in Section 4, present the Strict Intersection System for λ-calculus. Then, in Section 5, we will define a notion of type assignment on $\mathcal{X}$ that uses strict intersection and union types, give an extended example, and show that type assignment in the strict system is preserved by the translation of λ-calculus into $\mathcal{X}$. Finally, in Section 6, we will show that the notion of type assignment introduced in this paper is closed for both subject-reduction and expansion, but not for η-reduction.

1 The $\mathcal{X}$-calculus

In this section we will give the definition of the $\mathcal{X}$-calculus, that was proven to be a fine-grained implementation model for various well-known calculi in [5]. Its features two separate categories of 'connectors', *plug* and *socket*, that act as input and output channel, and is defined without a notion of substitution (implicit or explicit).

Definition 1.1 (SYNTAX) The nets of the $\mathcal{X}$-calculus are defined by the following syntax, where $x, y, \ldots$ range over the infinite set of *sockets*, and α, β over the infinite set of *plugs*.

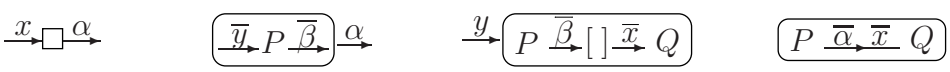
$$P, Q ::= \langle x.\alpha \rangle \mid \widehat{y}P\widehat{\beta}\cdot\alpha \mid P\widehat{\beta}\,[y]\,\widehat{x}Q \mid P\widehat{\alpha}\dagger\widehat{x}Q$$

In this definition, the $\widehat{\cdot}$ symbolizes that the socket or plug underneath is bound in the net. The *free sockets* $fc(P)$ and *free plugs* $fs(P)$ in a net P are defined as usual. The set of *free connectors* is the union of those of free sockets and plugs: $fc(P) = fs(P) \cup fp(P)$

A connector (socket or plug) which is not free is called *bound*. We will, as usual, identify porcesses that only differ in the names of bound connectors (α-conversion).

We use the following terminology for our nets: $\langle x.\alpha \rangle$ is called a *capsule*, $(\widehat{y}P\widehat{\beta}\cdot\alpha)$ an *export* net, $(P\widehat{\beta}\,[y]\,\widehat{x}Q)$ a *mediator*, and $(P\widehat{\alpha}\dagger\widehat{x}Q)$ a *cut*.

Diagrammatically, we represent these nets as:



The nets of $\mathcal{X}$ can be seen as a collection (heap) of wires (streams, strings), each with an input and an output. When two heaps interact, they do so through *one* input and *one* output name *only*, that are *bound* in the interaction. This interaction can be possible via a net like $(P\widehat{\alpha} \dagger \widehat{x}Q)$ that expresses an active computation; it will try to connect the wires ending with α in the heap called P to the wires beginning with x in the heap called Q. On the other hand, they can be bound as in $(P\widehat{\beta}\,[y]\,\widehat{x}Q)$, which expresses two heaps that try to connect the wires ending with β and beginning with x, but that need another heap to mediate between them; this new net will need to interact via the input name y (which might appear in P and Q as well). Also, a heap P that is willing to interact via the names y and β can itself be made available (exported) via the name α, as in $(\widehat{y}P\widehat{\beta}\cdot\alpha)$.

At any given moment, all names of unconnected inputs and outputs in a heap make up the collection of *free names*, that are *inactive* during the computational step; the bound names are all involved in some interaction.

The calculus, defined by the reduction rules below, explains in detail how cuts are distributed through nets to be eventually evaluated at the level of capsules. Reduction is defined by giving how the basic syntactic structures that are well-connected interact, and specifies how to deal with propagating active nodes in the computation to points where they can interact.

Definition 1.2 (Reduction: **Logical rules**)

$$
\begin{array}{rr}
(\textit{cap}): & \langle y.\alpha\rangle\widehat{\alpha} \dagger \widehat{x}\langle x.\beta\rangle \rightarrow \langle y.\beta\rangle \\
(\textit{exp}): & (\widehat{y}P\widehat{\beta}\cdot\alpha)\widehat{\alpha} \dagger \widehat{x}\langle x.\gamma\rangle \rightarrow \widehat{y}P\widehat{\beta}\cdot\gamma, \\
(\textit{med}): & \langle y.\alpha\rangle\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\beta}\,[x]\,\widehat{z}P) \rightarrow Q\widehat{\beta}\,[y]\,\widehat{z}P \\
(\textit{exp-imp}): & (\widehat{y}P\widehat{\beta}\cdot\alpha)\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}P) \rightarrow \begin{cases} Q\widehat{\gamma} \dagger \widehat{y} \\ (Q\widehat{\gamma} \dagger \widehat{y} \end{cases}
\end{array}
$$

The first three logical rules above specify a renaming (reconnecting) procedure, whereas the last rule specifies the basic computational step: it links a function, available over the unique plug α, to an open adjacent mediator position in the net that ports the unique socket x.

We now extend the syntax with two *flagged*, or *active* cuts:

$$P ::= \ldots \mid P_1\widehat{\alpha} \nearrow \widehat{x}P_2 \mid P_1\widehat{\alpha} \nwarrow \widehat{x}P_2$$

Terms constructed from the restricted syntax without those flagged cuts are called *pure* (the diagrammatical representation of flagged cuts is the same as that for unflagged cuts). These flagged cuts either reduce to normal cuts when dealing with a capsule, or are propagated through the net.

Definition 1.3 (Reduction: **Activating the cuts**) We define the following two activation rules.

$$
\begin{array}{l}
(\textit{act}\text{-L}): P\widehat{\alpha} \dagger \widehat{x}Q \rightarrow P\widehat{\alpha} \nearrow \widehat{x}Q \textit{ if } P \textit{ does not introduce } \alpha \\
(\textit{act}\text{-R}): P\widehat{\alpha} \dagger \widehat{x}Q \rightarrow P\widehat{\alpha} \nwarrow \widehat{x}Q \textit{ if } Q \textit{ does not introduce } x
\end{array}
$$

where:

(P *introduces* x): Either $P = P\widehat{\alpha}\,[x]\,\widehat{y}Q$, and x does not occur free in P, Q, or

$P = \langle x.\delta \rangle$.

(P *introduces* δ): Either $P = \widehat{x}Q\widehat{\beta}{\cdot}\delta$, and δ does not occur free in Q, or $P = \langle x.\delta \rangle$.

The activated cuts (obtained from cuts to which the logical rules cannot be applied) are introduced to obtained a fine-tuned reduction system. An activated cut is processed, by 'pushing' it, systematically, in the direction indicated by the tilting of the dagger, through its syntactic structure, until a cut is created that involves a capsule. The cut is then deactivated, such that a logical rule can be applied or, else, the 'pushing' can go on, but now in the other direction.

Definition 1.4 (REDUCTION: **Propagation rules**) **Left propagation**

$$
\begin{array}{rrll}
(d\text{L}): & \langle y.\alpha\rangle\widehat{\alpha} \nleftarrow \widehat{x}P \to \langle y.\alpha\rangle\widehat{\alpha} \dagger \widehat{x}P & & \\
(\mathit{cap}\nleftarrow): & \langle y.\beta\rangle\widehat{\alpha} \nleftarrow \widehat{x}P \to \langle y.\beta\rangle & & \beta \neq \alpha \\
(\mathit{exp\text{-}outs}\nleftarrow): & (\widehat{y}P'\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \nleftarrow \widehat{x}P \to (\widehat{y}(P'\widehat{\alpha} \nleftarrow \widehat{x}P)\widehat{\beta}{\cdot}\gamma)\widehat{\gamma} \dagger \widehat{x}P, & & \gamma \textit{ fresh} \\
(\mathit{exp\text{-}ins}\nleftarrow): & (\widehat{y}P'\widehat{\beta}{\cdot}\gamma)\widehat{\alpha} \nleftarrow \widehat{x}P \to \widehat{y}(P'\widehat{\alpha} \nleftarrow \widehat{x}P)\widehat{\beta}{\cdot}\gamma, \gamma \neq \alpha & & \\
(\mathit{imp}\nleftarrow): & (Q\widehat{\beta}\,[z]\,\widehat{y}P)\widehat{\alpha} \nleftarrow \widehat{x}P \to (Q\widehat{\alpha} \nleftarrow \widehat{x}P)\widehat{\beta}\,[z]\,\widehat{y}(P\widehat{\alpha} \nleftarrow \widehat{x}P) & & \\
(\mathit{cut}\nleftarrow): & (Q\widehat{\beta} \dagger \widehat{y}P)\widehat{\alpha} \nleftarrow \widehat{x}P \to (Q\widehat{\alpha} \nleftarrow \widehat{x}P)\widehat{\beta} \dagger \widehat{y}(P\widehat{\alpha} \nleftarrow \widehat{x}P) & &
\end{array}
$$

Right propagation

$$
\begin{array}{rrll}
(d\text{R}): & P\widehat{\alpha} \nrightarrow \widehat{x}\langle x.\beta\rangle \to P\widehat{\alpha} \dagger \widehat{x}\langle x.\beta\rangle & & \\
(\nrightarrow\mathit{cap}): & P\widehat{\alpha} \nrightarrow \widehat{x}\langle y.\beta\rangle \to \langle y.\beta\rangle & & y \neq x \\
(\nrightarrow\mathit{exp}): & P\widehat{\alpha} \nrightarrow \widehat{x}(\widehat{y}P'\widehat{\beta}{\cdot}\gamma) \to \widehat{y}(P\widehat{\alpha} \nrightarrow \widehat{x}P')\widehat{\beta}{\cdot}\gamma & & \\
(\nrightarrow\mathit{imp\text{-}outs}): & P\widehat{\alpha} \nrightarrow \widehat{x}(Q\widehat{\beta}\,[x]\,\widehat{y}P) \to P\widehat{\alpha} \dagger \widehat{z}((P\widehat{\alpha} \nrightarrow \widehat{x}Q)\widehat{\beta}\,[z]\,\widehat{y}(P\widehat{\alpha} \nrightarrow \widehat{x}P)) & & z \textit{ fresh} \\
(\nrightarrow\mathit{imp\text{-}ins}): & P\widehat{\alpha} \nrightarrow \widehat{x}(Q\widehat{\beta}\,[z]\,\widehat{y}P) \to (P\widehat{\alpha} \nrightarrow \widehat{x}Q)\widehat{\beta}\,[z]\,\widehat{y}(P\widehat{\alpha} \nrightarrow \widehat{x}P) & & z \neq x \\
(\nrightarrow\mathit{cut}): & P\widehat{\alpha} \nrightarrow \widehat{x}(Q\widehat{\beta} \dagger \widehat{y}P) \to (P\widehat{\alpha} \nrightarrow \widehat{x}Q)\widehat{\beta} \dagger \widehat{y}(P\widehat{\alpha} \nrightarrow \widehat{x}P) & &
\end{array}
$$

We write $\to$ for the (transitive, compatible) reduction relation that only reduces active cuts (and those to which the logical rules can be applied).

Although the origin of $\mathcal{X}$ is a logic, and one could expect it to be close to λ-calculus, it is in fact specified as a *conditional term rewriting system*; the only non-standard aspect is that it treats *three* different classes of variables (for plugs, sockets, and nets). The implications of this observation are left for further research; a first result can be found in [6].

The rules (*exp-outs*$\nleftarrow$) and ($\nrightarrow$*imp-outs*) deserve some attention. Here the exposed occurence of the plug α (socket x) need not be the only one. The cut wants to connect all its occurences to the corresponding socket (plug), so, first all *other* occurences, i.e. not to top one, are eliminated (notice that reducing the cut will erase the bound connectors), at the end of which the remaining occurence is now *introduced* in the term, and can be dealt with (if the other side of the cut is of the right shape). We rename to avoid α (x) to occur both bound and free; but, in fact, no confusion is possible, so the α-conversion here is almost cosmetical.

The reduction relation $\to$ is not confluent; this comes in fact from the critical pair that activates a cut $P\widehat{\alpha} \dagger \widehat{x}Q$ in two ways if P does not introduce α and Q does not introduce x. When activating according to the first criterium, the reduction will

connect the wires in P that end with α with the wires from Q that begin with x; if α does not appear in P, this will return P. When using the second, the reduction will connect the wires in Q that begin with x with the wires from P that end with α; if x does not appear in Q, this will return Q. As an example, consider that (when $\beta \neq \gamma$, and $y \neq z$)

$$(\widehat{x}\langle x.\alpha\rangle\widehat{\alpha}\cdot\gamma)\widehat{\beta} \dagger \widehat{z}\langle y.\delta\rangle \rightarrow \begin{cases} (\widehat{x}\langle x.\alpha\rangle\widehat{\alpha}\cdot\gamma)\widehat{\beta} \nwarrow \widehat{z}\langle y.\delta\rangle \rightarrow \langle y.\delta\rangle \\ (\widehat{x}\langle x.\alpha\rangle\widehat{\alpha}\cdot\gamma)\widehat{\beta} \nleftarrow \widehat{z}\langle y.\delta\rangle \rightarrow \widehat{x}\langle x.\alpha\rangle\widehat{\alpha}\cdot\gamma \end{cases}$$

So, if activation in both directions is possible, the end result of the reduction can be different.

In [5] some basic properties were shown, which essentially show that the calculus is well-behaved, and the relation between $\mathcal{X}$ and a number of other calculi. These results motivate the formulation of new rules:

Lemma 1.5 ([5]) *The following reduction rules are admissible:*

$$\begin{array}{rl} (\nleftarrow gc) : & P\widehat{\alpha} \nleftarrow \widehat{x}Q \rightarrow P, \text{ if } \alpha \notin fp(P),\ P \text{ pure}. \\ (\nwarrow gc) : & P\widehat{\alpha} \nwarrow \widehat{x}Q \rightarrow Q, \text{ if } x \notin fs(Q),\ P \text{ pure} \\ (ren\text{-L}) : & P\widehat{\delta} \dagger \widehat{z}\langle z.\alpha\rangle \rightarrow P[\alpha/\delta],\ P \text{ pure} \\ (ren\text{-R}) : & \langle z.\alpha\rangle\widehat{\alpha} \dagger \widehat{x}P \rightarrow P[z/x],\ P \text{ pure} \end{array}$$

2 Typing for $\mathcal{X}$

The notion of *Curry* type assignment as presented in [14,5] on $\mathcal{X}$ is defined in such a way that it gives a straightforward Curry-Howard isomorphism and Gentzen's sequent calculus for (implicative) Classical Logic [13].

Definition 2.1 (Curry types and Contexts) (i) The set of *Curry* types is defined by the grammar:

$$A, B ::= \varphi \mid A \rightarrow B$$

The Curry types considered in this paper are normally known as *Curry* types.

(ii) A *context of variables* Γ is a mapping from variables to types, denoted as a finite set of *statements* $x{:}A$, such that the *subject* of the statements (x) are distinct. We write $\Gamma, x{:}A$ for the context of names defined by:

$$\begin{array}{rll} \Gamma, x{:}A & = \Gamma \cup \{x{:}A\}, & \text{if } \Gamma \text{ is not defined on } x \\ & = \Gamma, & \text{otherwise} \end{array}$$

So, when writing a context as $\Gamma, x{:}A$, this implies that $x{:}A \in \Gamma$, or Γ is not defined on x.

Contexts of *names* are defined in a similar way.

The normal view for derivable statements in Gentzen's system, like $\Gamma \vdash \Delta$, is that the formulae in the context Γ are all assumptions and are therefore supposed to hold, and at least one of the formulea in Δ is a consequence of Γ. However, not all the formulae in Δ necessarily follow from Γ. In other words, the formulae in the context Γ are connected through the logical 'and', whereas those in Δ are

connected through the logical 'or'. This is reflected in the type assignment rules of the system of Definition 5.1.

Definition 2.2 (Curry Typing for $\mathcal{X}$ [5]) (i) *Type judgements* are expressed via a ternary relation $M :\cdot \Gamma \vdash_C \Delta$, where Γ is a context of *variables* and Δ is a context of *names*, and M is a term. We say that M is the *witness* of this judgement.

(ii) *Curry type assignment for* $\mathcal{X}$ is defined by the following sequent calculus:

$$(cap): \frac{}{\langle y.\alpha\rangle :\cdot \Gamma, y{:}A \vdash_C \alpha{:}A, \Delta}$$

$$(cut): \frac{M :\cdot \Gamma \vdash_C \alpha{:}A, \Delta \quad N :\cdot \Gamma, x{:}A \vdash_C \Delta}{M\widehat{\alpha} \dagger \widehat{x}N :\cdot \Gamma \vdash_C \Delta}$$

$$(med): \frac{M :\cdot \Gamma \vdash_C \alpha{:}A, \Delta \quad N :\cdot \Gamma, x{:}B \vdash_C \Delta}{M\widehat{\alpha}\,[y]\,\widehat{x}N :\cdot \Gamma, y{:}A{\rightarrow}B \vdash_C \Delta}$$

$$(exp): \frac{M :\cdot \Gamma, x{:}A \vdash_C \alpha{:}B, \Delta}{\widehat{x}M\widehat{\alpha}{\cdot}\beta :\cdot \Gamma \vdash_C \beta{:}A{\rightarrow}B, \Delta}$$

We write $M :\cdot \Gamma \vdash_C \Delta$ if there exists a derivation that has this judgement in the bottom line, and write $\mathsf{D} :: M :\cdot \Gamma \vdash_C \Delta$ if we want to name that derivation.

3 The relation with the Lambda Calculus

The remainder of this paper will be dedicated to the definition of a notion of intersection type assignment on $\mathcal{X}$. The definition will be such that it will be a natural extension of a system with intersection types for λ-calculus; we will start by briefly summarizing the latter. We assume the reader to be familiar with the λ-calculus [8]; we just recall the definition of lambda terms and β-contraction. We will write $\underline{n}$ for $\{1, \ldots, n\}$, where $n \geq 0$.

Definition 3.1 (Lambda terms and β-contraction [8]) (i) The set Λ of *lambda terms* is defined by the syntax:

$$M ::= x \mid \lambda x.M \mid M_1 M_2$$

(ii) The reduction relation $\rightarrow_\beta$ is defined as the contextual (i.e. compatible [8]) closure of the rule:

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

The relation $\twoheadrightarrow_\beta$ is the reflexive and transitive closure of $\rightarrow_\beta$, and the $=_\beta$ is the equivalence relation generated by $\twoheadrightarrow_\beta$

We can define the direct encoding of λ-calculus into $\mathcal{X}$:

Definition 3.2 ([5]) The interpretation of lambda terms into terms of $\mathcal{X}$ via the

plug α, $[\![M]\!]_{\mathrm{N}}\alpha$, is defined by:

$$\begin{aligned}
[\![x]\!]_{\mathrm{N}}\alpha &= \langle x.\alpha\rangle \\
[\![\lambda x.M]\!]_{\mathrm{N}}\alpha &= \widehat{x}[\![M]\!]_{\mathrm{N}}\beta\widehat{\beta}\cdot\alpha \\
[\![MN]\!]_{\mathrm{N}}\alpha &= [\![M]\!]_{\mathrm{N}}\gamma\widehat{\gamma} \dagger \widehat{x}([\![N]\!]_{\mathrm{N}}\beta\widehat{\beta}\ [x]\ \widehat{y}\langle y.\alpha\rangle),\ \textit{with } x \textit{ fresh}
\end{aligned}$$

Notice that every sub-term of $[\![M]\!]_{\mathrm{N}}\alpha$ has exactly one free plug.

The image of Λ in to $\mathcal{X}$ is not extensional:

$$\begin{aligned}
&[\![\lambda x.yx]\!]_{\mathrm{N}}\alpha \triangleq \widehat{x}[\![yx]\!]_{\mathrm{N}}\beta\widehat{\beta}\cdot\alpha && \triangleq \\
&\widehat{x}([\![y]\!]_{\mathrm{N}}\gamma\widehat{\gamma} \dagger \widehat{z}([\![x]\!]_{\mathrm{N}}\delta\widehat{\delta}\ [z]\ \widehat{w}\langle w.\beta\rangle))\widehat{\beta}\cdot\alpha && \triangleq \\
&\widehat{x}(\langle y.\gamma\rangle\widehat{\gamma} \dagger \widehat{z}(\langle x.\delta\rangle\widehat{\delta}\ [z]\ \widehat{w}\langle w.\beta\rangle))\widehat{\beta}\cdot\alpha && \rightarrow \\
&\widehat{x}(\langle x.\delta\rangle\widehat{\delta}\ [y]\ \widehat{w}\langle w.\beta\rangle)\widehat{\beta}\cdot\alpha && \neq \langle y.\alpha\rangle = [\![y]\!]_{\mathrm{N}}\alpha
\end{aligned}$$

Directly translating the η-reduction rule '$\lambda x.Mx \rightarrow M$ *if* $x \notin$ *fv*(M)' into $\mathcal{X}$ would give:

$$\widehat{x}([\![M]\!]_{\mathrm{N}}\gamma\widehat{\gamma} \dagger \widehat{z}(\langle x.\delta\rangle\widehat{\delta}\ [z]\ \widehat{w}\langle w.\beta\rangle))\widehat{\beta}\cdot\alpha \rightarrow [\![M]\!]_{\mathrm{N}}\alpha$$

or, in a more general notation:

Definition 3.3 (η-REDUCTION ON $\mathcal{X}$) $\quad \widehat{x}(P\widehat{\gamma} \dagger \widehat{z}(\langle x.\delta\rangle\widehat{\delta}\ [z]\ \widehat{w}\langle w.\beta\rangle))\widehat{\beta}\cdot\alpha \rightarrow P\widehat{\gamma} \dagger \widehat{z}$

Notice that we need to create a cut on the right-hand side to make sure that the result of P is available on the plug α.

This is of course a true extension of the notion of reduction, but we can 'justify' it by taking P to be the interpretation of abstraction (with $x \notin$ *fv*(Q)):

$$\begin{aligned}
&\widehat{x}((\widehat{v}Q\widehat{\epsilon}\cdot\gamma)\widehat{\gamma} \dagger \widehat{z}(\langle x.\delta\rangle\widehat{\delta}\ [z]\ \widehat{w}\langle w.\beta\rangle))\widehat{\beta}\cdot\alpha && \rightarrow \\
&\widehat{x}(\langle x.\delta\rangle\widehat{\delta} \dagger \widehat{v}(Q\widehat{\epsilon} \dagger \widehat{w}\langle w.\beta\rangle))\widehat{\beta}\cdot\alpha && \rightarrow \\
&\widehat{x}(\langle x.\delta\rangle\widehat{\delta} \dagger \widehat{v}(Q[\beta/\epsilon]))\widehat{\beta}\cdot\alpha && \rightarrow \\
&\widehat{x}(Q[\beta/\epsilon][x/v])\widehat{\beta}\cdot\alpha && = (\alpha\textit{-conversion}) \\
&\widehat{v}N\widehat{\epsilon}\cdot\alpha
\end{aligned}$$

and that $(\widehat{v}N\widehat{\epsilon}\cdot\gamma)\widehat{\gamma} \dagger \widehat{z}\langle z.\alpha\rangle \rightarrow \widehat{v}N\widehat{\epsilon}\cdot\alpha$, exactly as expected.

As shown in [5], the notion of Curry type assignment for λ-calculus, $\Gamma \vdash_{\lambda} M : A$ is strongly linked to the one defined for $\mathcal{X}$. In [5], the following relation is shown between λ-calculus and $\mathcal{X}$:

Theorem 3.4 ([5]) (i) *If* $M \rightarrow_{\beta} N$, *then* $[\![M]\!]_{\mathrm{N}}\alpha \rightarrow [\![N]\!]_{\mathrm{N}}\alpha$.
(ii) *if* $\Gamma \vdash_{\lambda} M : A$, *then* $[\![M]\!]_{\mathrm{N}}\alpha :\cdot \Gamma \vdash_{\mathcal{C}} \alpha{:}A$.

4 Intersection Type Assignment for the Lambda Calculus

The notion of intersection type assignment for $\mathcal{X}$ as defined in the next section is inspired by the system of [12] (the precise relation between the two – through the interpretation functions as defined in [14,5] – needs to be studied, and is left for further research). It is a conservative extension of the Strict Intersection Type Assignment System of [1] (see also [2,3]), in that lambda terms typeable in that system translate to $\mathcal{X}$ nets, while preserving the type.

In this section, we will present that strict system; it can be seen as a restricted version of the BCD-system as presented in [9]. The major feature of this restricted system, compared to the BCD-system, is that the $\leq$-relation on types is not contra-variant over arrow types. Also, the $\leq$ relation on types is no longer contra-variant on the argument type in arrow-types, but restricted to the one induced by $A \cap B \leq A$ and taking $\top$ to be the maximal type.

We now come to the definition of strict intersection types.

Definition 4.1 (Strict types, statements, and contexts) (i) Let Φ be a countable (infinite) set of type-variables, ranged over by φ. $\mathcal{T}_S$, the set of *strict types*, and the set $\mathcal{T}$ of *intersection types*, both ranged over by $A, B, \ldots$, are defined through:

- The set $\mathcal{T}_S$ of *strict types* is inductively defined by:

$$A, B ::= \varphi \mid ((A_1 \cap \cdots \cap A_n) \rightarrow B), (n \geq 0)$$

- The set $\mathcal{T}$ of *strict intersection types* is defined by:

$$\{A_1 \cap \cdots \cap A_n \mid n \geq 0 \ \& \ \forall i \in \underline{n} \ [A_i \text{ is a strict type}]\}$$

We will write $\top$ for the empty intersection type.

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

(iii) A *context* Γ is a partial mapping from term variables to intersection types, so we can use Γx for the type stored for x in Γ. As standard, a context is also represented as a set of statements with only distinct variables as subjects. We will write $x \notin \Gamma$ if Γ is not defined on x.

(iv) For contexts $\Gamma_1, \ldots, \Gamma_n$, the context $\Gamma_1 \cap \cdots \cap \Gamma_n$ is defined by: $x{:}A_1 \cap \cdots \cap A_m \in \Gamma_1 \cap \cdots \cap \Gamma_n$ if and only if $\{x{:}A_1, \ldots, x{:}A_m\}$ is the set of all statements with strict predicate about x that occur in $\Gamma_1 \bigcup \cdots \bigcup \Gamma_n$.

(v) We write $\Gamma \cap x{:}A$ for the context $\Gamma \cap \{x{:}A\}$, i.e., the context defined by:

$$\begin{array}{rl} \Gamma \cap x{:}A = \Gamma \cup \{x{:}A\}, & \text{if } x \notin \Gamma \\ = \Gamma \backslash x \cup \{x{:}A \cap B\}, & \text{if } x{:}B \in \Gamma \end{array}$$

We will often write $\Gamma, x{:}A$ for $\Gamma \cap x{:}A$ when $x \notin \Gamma$.

Notice that strict types are either type-variables, φ, or arrow types. In an arrow type, the type on the right of the arrow type constructor is always strict; the type on the left of the arrow is an intersection type, but since $\mathcal{T}_S$ is a proper subset of $\mathcal{T}$, it can be strict.

In the notation of types, as usual, right-most outer-most brackets will be omitted. Also, we write $\cap_{\underline{n}} A_i$ for the type $A_1 \cap \cdots \cap A_n$.

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

Definition 4.2 (Relations on types) (i) The relation $\leq$ is defined as the least

pre-order (i.e. reflexive and transitive relation) on $\mathcal{T}_S$ such that:

$$\begin{array}{rcll} & & \cap_{\underline{n}} A_i & \leq & A_i, \text{ for all } i \in \underline{n}, & (n \geq 1) \\ B \leq A_i, \text{ for all } i \in \underline{n} & \Rightarrow & B \leq \cap_{\underline{n}} A_i, & & & (n \geq 0) \end{array}$$

(ii) The equivalence relation $\sim$ on types is defined by: $A \sim B \iff A \leq B \leq A$, and we will consider types modulo $\sim$.

(iii) We write $\Gamma \leq \Gamma'$ if and only if for every $x{:}A' \in \Gamma'$ there is an $x{:}A \in \Gamma$ such that $A \leq A'$, and $\Gamma \sim \Gamma' \iff \Gamma \leq \Gamma' \leq \Gamma$.

$\mathcal{T}$ may be considered modulo $\sim$; then $\leq$ becomes a partial order.

Notice that $A \leq A$, and $A \leq \top$, for all A; 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 $A{\cap}B{\cap}C$ rather than $(A{\cap}B){\cap}C$. Moreover, we will assume, unless stated explicitly otherwise, that in $\cap_{\underline{n}} A_i$ each A_i is a strict type.

The definition of the $\leq$-relation as given in [9] (apart from dealing with intersection types occurring on the right of the arrow type constructor) or [2] also contained the alternative:

$$C \leq A \,\&\, B \leq D \;\Rightarrow\; A{\to}B \leq C{\to}D$$

This was added mainly to obtain a notion of type assignment closed for η-reduction (i.e. β-reduction extended with $\lambda x.Mx \to_\eta M$, if x is not free in M), a feature that is not considered here.

Definition 4.3 (STRICT TYPE ASSIGNMENT AND DERIVATIONS) (i) *Strict intersection type assignment* and *strict intersection derivations* are defined by the following natural deduction system (where A in rules $(\to E)$ and $(\to I)$ is in $\mathcal{T}$):

$$(ax): \frac{}{x{:}\cap_{\underline{n}} A_i \vdash_\lambda x : A_i}\ (i \in \underline{n}) \qquad (\cap I): \frac{\Gamma \vdash_\lambda M : A_1 \ldots \Gamma \vdash_\lambda M : A_n}{\Gamma \vdash_\lambda M : \cap_{\underline{n}} A_i}\ (n \geq 0)$$

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

(ii) We write $\Gamma \vdash_\lambda M : A$ if this statement is derivable using a strict intersection derivation, and write $\mathrm{D} :: \Gamma \vdash_\lambda M : A$ to specify that this result was obtained through the derivation D.

Notice that, since $\top$ is considered to be the empty intersection, the derivation rule

$$(\top): \frac{}{\Gamma \vdash_\lambda M : \top}$$

is implicit in rule $(\cap I)$.

Theorem 4.4 (cf. [1,4]) *The following rules are admissible:*

$$(\leq): \frac{\Gamma \vdash_\lambda M : A}{\Gamma' \vdash_\lambda M : B}\,(\Gamma' \leq \Gamma, A \leq B) \qquad (=_\beta): \frac{\Gamma \vdash_\lambda M : A}{\Gamma \vdash_\lambda N : A}\,(M =_\beta N)$$

$$(\mathit{cut}): \frac{\Gamma, x{:}A \vdash_\lambda M : B \quad \Gamma \vdash_\lambda N : A}{\Gamma \vdash_\lambda M[N/x] : B}$$

5 Intersection and Union Type Assignment for $\mathcal{X}$

The notion of intersection type assignment on $\mathcal{X}$ that we will present in this section is a natural extension of the system considered in [5], i.e. the basic implicative system for Classical Logic, but extended with intersection and union types and the type constant $\top$.

The initial aim of this work was to define a system using intersection types only, but, when trying to prove the conversion results of the next section, problems were encountered. These were mainly due to the fact that, in that approach, when a plug carried an intersection type $A{\cap}B$, if was not sure if this was derived by combining two derivations, one with A, and the other with B, so, in particular, step ($\lightning$*imp-outs*) of the proof of Theorem 6.1 was troublesome.

Also, just forcing intersection types only on the system violates the normal interpretation of the system of Classical Logic of Definition 2.2. The normal view for statements like $\Gamma \vdash \Delta$ is that the formulae in the context Γ are all necessary for the result, and not all the formulae in Δ necessarily follow from Γ; in other words, the formulae in the context Γ are connected through the logical 'and', whereas those in Δ are connected through the logical 'or'. So, also inspired by [12], a system was set up that allowed *only* intersection types for sockets, and *only* union types for plugs, but this soon proved to be too restrictive. Intersection types are sometimes needed on plugs, and union types can be needed on sockets.

These observations then led to the present definition. Essentially, the choice above still stands: intersection types for sockets, and union types for plugs, and obsolete types can be added at will via the rules ($\cap L$) and ($\cup R$), respectively. However, a union type like $A{\cup}B$ for sockets is allowed, but only if both A and B can be justified (see rule ($\cup L$)); similarly, an intersection type like $A{\cap}B$ for plugs is only allowed if both A and B can be justified (see rule ($\cap R$)).

The following definition of strict types is a natural extension of the notion of strict types of the previous section, by adding union as a type constructor.

Definition 5.1 (Types, statements, and contexts) (i)

(a) The set $\mathcal{T}_S$ of *strict types* is inductively defined by:

$$\begin{aligned} A, B ::= {} & \varphi \mid ((A_1 {\cap} \cdots {\cap} A_n) \to B),\ (n \geq 0) \\ & \mid ((A_1 {\cup} \cdots {\cup} A_n) \to B),\ (n \geq 0) \end{aligned}$$

(b) The set $\mathcal{T}$ of *types* is defined by:

$$\{A_1 \cap \cdots \cap A_n \mid n \geq 0 \ \& \ \forall i \in \underline{n}\ [A_i \text{ is a strict type}]\} \cup$$
$$\{A_1 \cup \cdots \cup A_n \mid n \geq 0 \ \& \ \forall i \in \underline{n}\ [A_i \text{ is a strict type}]\}$$

We will write $\top$ for the empty intersection type.

(ii) Statements and contexts are defined as in Definition 4.1.

(iii) For contexts of sockets $\Gamma_1, \ldots, \Gamma_n$, the context $\Gamma_1 \cap \cdots \cap \Gamma_n$ is defined by: $x{:}A_1 \cap \cdots \cap A_m \in \Gamma_1 \cap \cdots \cap \Gamma_n$ if and only if $\{x{:}A_1, \ldots, x{:}A_m\}$ is the set of all statements about x that occur in $\{v{:}C \mid \exists i \in \Gamma_i\ [v{:}C \in \Gamma_i]\}$. We write $\Gamma \cap x{:}A$ for the context of sockets $\Gamma \cap \{x{:}A\}$, i.e., the context defined by (where $\bigcup$ is the operation of union on sets):

$$\begin{aligned} \Gamma \cap x{:}A &= \Gamma \bigcup \{x{:}A\}, && \text{if } x \notin \Gamma \\ &= \Gamma \backslash x \bigcup \{x{:}A \cap B\}, && \text{if } x{:}B \in \Gamma \end{aligned}$$

We will often write $\Gamma, x{:}A$ for $\Gamma \cap x{:}A$ when $x \notin \Gamma$.

(iv) For contexts of plugs, $\Delta_1, \ldots, \Delta_n$, the context $\Delta_1 \cup \cdots \cup \Delta_n$ is defined by: $\alpha{:}A_1 \cup \cdots \cup A_m \in \Delta_1 \cup \cdots \cup \Delta_n$ if and only if $\{\alpha{:}A_1, \ldots, \alpha{:}A_m\}$ is the set of all statements about α that occur in $\{\beta{:}C \mid \exists i \in \Delta_i\ [\beta{:}C \in \Delta_i]\}$. We write $\alpha{:}A \cup \Delta$ for the context of sockets $\{\alpha{:}A\} \cup \Delta$, i.e., the context defined by:

$$\begin{aligned} \alpha{:}A \cup \Delta &= \{\alpha{:}A\} \bigcup \Delta, && \text{if } \alpha \notin \Delta \\ &= \{\alpha{:}A \cup B\} \bigcup \Delta \backslash \alpha, && \text{if } \alpha{:}B \in \Delta \end{aligned}$$

We will often write $\alpha{:}A, \Delta$, for $\alpha{:}A \cup \Delta$ when $\alpha \notin \Delta$.

In order to not have derivations littered with applications of the *Weakening* rule, we allow rules to *combine* the contexts of the subterms involved; this does not exclude the normal approach, since the contexts can be equal. The most important thing to notice is that, by rule (ax), only *strict* types are added to contexts, and that, via the rules, *intersection types* are built of contexts of sockets, and *union types* are built for contexts of plugs.

However, a union type can appear in a contexts of sockets, but only via the rule $(\cup L)$; similarly, an intersection type can appear in a contexts of plugs, but only via the rule $(\cap R)$. This restriction helps to avoid the famous subject reduction problem of systems with union types.

Definition 5.1 (Intersection and Union Typing for $\mathcal{X}$) (i) *Intersection type judgements* are expressed via a ternary relation $P :\cdot \Gamma \vdash \Delta$, where Γ is a context of *sockets* and Δ is a context of *plugs*, and P is a net. We say that P is the *witness* of this judgement.

(ii) *Intersection and union type assignment for* $\mathcal{X}$ is defined by the following sequent calculus:

$$(ax): \frac{}{< y, \alpha > :\cdot \Gamma \cap y{:}A \vdash \alpha{:}A \cup \Delta}\ (A \in \mathcal{T}_S) \qquad (\rightarrow R): \frac{P :\cdot \Gamma, x{:}A \vdash \alpha{:}B, \Delta}{\widehat{x}P\widehat{\alpha}\cdot\beta :\cdot \Gamma \vdash \beta{:}A \rightarrow B \cup \Delta}$$

$$(\to\!L) : \frac{P :\cdot \Gamma_1 \vdash \alpha{:}A, \Delta_1 \quad Q :\cdot \Gamma_2, x{:}B \vdash \Delta_2}{P\widehat{\alpha}\,[y]\,\widehat{x}Q :\cdot \Gamma_1 \cap \Gamma_2 \cap y{:}A\to B \vdash \Delta_1 \cup \Delta_2}$$

$$(\mathit{cut}) : \frac{P :\cdot \Gamma_1 \vdash \alpha{:}A, \Delta_1 \quad Q :\cdot \Gamma_2, x{:}A \vdash \Delta_2}{P\widehat{\alpha} \dagger \widehat{x}Q :\cdot \Gamma_1 \cap \Gamma_2 \vdash \Delta_1 \cup \Delta_2}$$

$$(\cap\!L) : \frac{P :\cdot \Gamma, x{:}A \vdash \Delta}{P :\cdot \Gamma, x{:}A \cap B \vdash \Delta} \qquad (\cap\!R) : \frac{P :\cdot \Gamma_1 \vdash \alpha{:}A, \Delta_1 \quad P :\cdot \Gamma_2 \vdash \alpha{:}B, \Delta_2}{P :\cdot \Gamma_1 \cap \Gamma_2 \vdash \alpha{:}A \cap B, \Delta_1 \cup \Delta_2}$$

$$(\cup\!L) : \frac{P :\cdot \Gamma_1, x{:}A \vdash \Delta_1 \quad P :\cdot \Gamma_2, x{:}B \vdash \Delta_2}{P :\cdot \Gamma_1 \cap \Gamma_2, x{:}A \cup B \vdash \Delta_1 \cup \Delta_2} \qquad (\cup\!R) : \frac{P :\cdot \Gamma \vdash \alpha{:}A, \Delta}{P :\cdot \Gamma \vdash \alpha{:}A \cup B, \Delta}$$

$$() : \frac{}{P :\cdot \Gamma, x{:}\top \vdash \Delta} \qquad () : \frac{}{P :\cdot \Gamma \vdash \alpha{:}\top, \Delta}$$

We write $P :\cdot \Gamma \vdash \Delta$ if there exists a derivation that has this judgement in the bottom line, and write $\mathsf{D} :: P :\cdot \Gamma \vdash \Delta$ if we want to name that derivation.

Notice that, in $P :\cdot \Gamma \vdash \Delta$, there is no notion of type for P itself, instead the derivable statement shows how P is connectable; Γ and Δ carry the types of the free connectors in P, as unordered sets.

Lemma 5.2 (WEAKENING) *The following rule is admissible:*

$$(W) : \frac{P :\cdot \Gamma \vdash \Delta}{P :\cdot \Gamma \cap x{:}A \vdash \alpha{:}B \cup \Delta}$$

We can now show that typeability is preserved by $[\![\cdot]\!]_{\mathrm{N}}\alpha$:

Theorem 5.3 *If* $\Gamma \vdash_\lambda M : A$, *then* $[\![M]\!]_{\mathrm{N}}\alpha :\cdot \Gamma \vdash \alpha{:}A$.

Proof: By induction on the structure of derivations in $\vdash_\lambda$.

(*ax*): Then $M \equiv x$, and $\Gamma = \Gamma', x{:}\cap_{\underline{n}} A_i$, and $A = A_i \in \mathcal{T}_S$ for some $1 \leq i \leq n$. Take

$$\Gamma'' = \Gamma, x{:}A_1 \cap \cdots \cap A_{i-1} \cap A_{i+1} \cap \cdots \cap A_n$$

so $\Gamma'' \cap x{:}A_i = \Gamma$, then

$$\frac{}{[\![x]\!]_{\mathrm{N}}\alpha :\cdot \Gamma'' \cap x{:}A_i \vdash \alpha{:}A_i}\,(\mathit{ax})$$

($\to\!I$): Then $M \equiv \lambda x.N$, $A = C \to D$, and $\Gamma, x{:}C \vdash_\lambda N : D$. Then $\mathsf{D} :: [\![N]\!]_{\mathrm{N}}\beta :\cdot \Gamma, x{:}C \vdash \beta{:}D$ exists by induction, and we can construct:

$$\frac{\begin{array}{c}\mathsf{D}\\ [\![N]\!]_{\mathrm{N}}\beta :\cdot \Gamma, x{:}C \vdash \beta{:}D\end{array}}{\widehat{x}[\![N]\!]_{\mathrm{N}}\beta\widehat{\beta}\cdot\alpha :\cdot \Gamma \vdash \alpha{:}C \to D}\,(\to\!R)$$

Notice that $\widehat{x}[\![N]\!]_{\mathrm{N}}\beta\widehat{\beta}\cdot\alpha = [\![\lambda x.N]\!]_{\mathrm{N}}\alpha$.

($\cap I$): Then $A = \cap_{\underline{n}} A_i$, and we have $\Gamma \vdash_\lambda M : A_i$ for all $i \in \underline{n}$. By induction, $[\![M]\!]_{\mathrm{N}}\alpha :\cdot \Gamma \vdash \alpha{:}A_i$ for all $i \in \underline{n}$, so, by rule ($\cap R$), also $[\![M]\!]_{\mathrm{N}}\alpha :\cdot \Gamma \vdash \alpha{:}A$.

($\top$): Notice that $[\![P]\!]_{\mathrm{N}}\alpha :\cdot \Gamma \vdash \alpha{:}\top$ by rule ().

($\rightarrow$*E*): Then $M \equiv M_1 M_2$, and there exists B such that both $\Gamma \vdash_\lambda M_1 : B{\rightarrow}A$ and $\Gamma \vdash_\lambda M_2 : B$. By induction, both $\mathrm{D}_1 :: \llbracket M_1 \rrbracket_{\mathrm{N}} \gamma :\cdot \Gamma \vdash \gamma{:}B{\rightarrow}A$ and $\mathrm{D}_2 :: \llbracket M_2 \rrbracket_{\mathrm{N}} \beta :\cdot \Gamma \vdash \beta{:}B$ exist, and we can construct:

$$\dfrac{\dfrac{\mathrm{D}_1}{\llbracket M_1 \rrbracket_{\mathrm{N}} \gamma :\cdot \Gamma \vdash \gamma{:}B{\rightarrow}A} \qquad \dfrac{\dfrac{\mathrm{D}_2}{\llbracket M_2 \rrbracket_{\mathrm{N}} \beta :\cdot \Gamma \vdash \beta{:}B} \qquad \dfrac{}{< y, \alpha > :\cdot y{:}A \vdash \alpha{:}A}\,(ax)}{\llbracket M_2 \rrbracket_{\mathrm{N}} \beta \widehat{\beta}\,[x]\ \widehat{y}{<}\, y, \alpha > :\cdot \Gamma, x{:}B{\rightarrow}A \vdash \alpha{:}A}\,(\rightarrow L)}{\llbracket M_1 \rrbracket_{\mathrm{N}} \gamma \widehat{\gamma} \dagger \widehat{x}(\llbracket M_2 \rrbracket_{\mathrm{N}} \beta \widehat{\beta}\,[x]\ \widehat{y}{<}\, y, \alpha >) :\cdot \Gamma \vdash \alpha{:}A}\,(cut)$$

Notice that $\llbracket M_1 M_2 \rrbracket_{\mathrm{N}} \alpha = \llbracket M_1 \rrbracket_{\mathrm{N}} \gamma \widehat{\gamma} \dagger \widehat{x}(\llbracket M_2 \rrbracket_{\mathrm{N}} \beta \widehat{\beta}\,[x]\ \widehat{y}{<}\, y, \alpha >)$, and that, by construction, $x, y \notin \Gamma$.

Example 5.4 (WHY STRICT TYPES) The real motivation for using strict types rather than the – perhaps more easier to understand – normal types defined by the syntax

$$A, B ::= \varphi \mid A \cap B \mid A \cup B$$

is that we would have loss of the subject reduction property, as in the system of [11] defined for λ-calculus (there the solution is to use *Harrop* types).

Using essentially the same rules as in Definition 5.1 (using normal types rather than strict types, of course), we can derive:

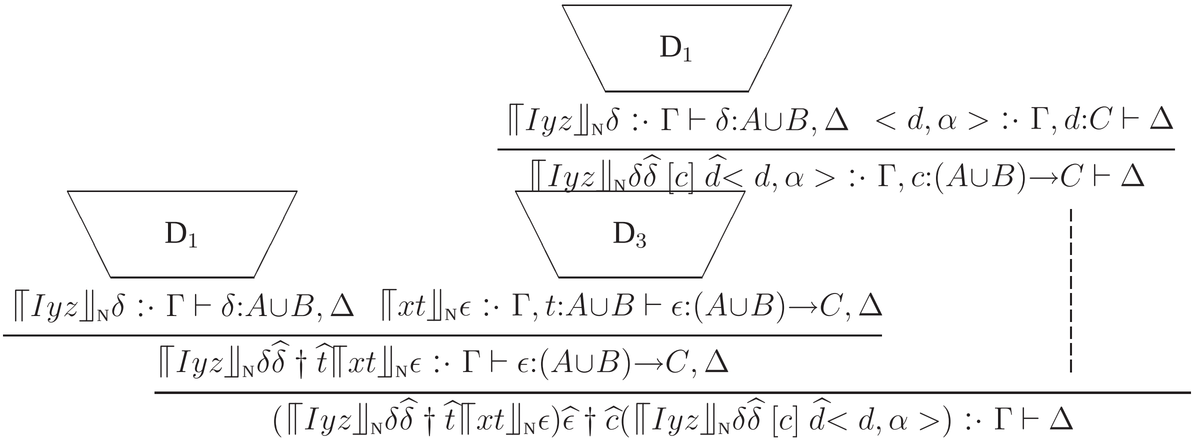
$$\dfrac{\dfrac{\mathrm{D}_1}{\llbracket I y z \rrbracket_{\mathrm{N}} \delta :\cdot \Gamma \vdash \delta{:}A \cup B, \Delta} \qquad \dfrac{\dfrac{\mathrm{D}_2^A}{\llbracket x t t \rrbracket_{\mathrm{N}} \beta :\cdot \Gamma, t{:}A \vdash \Delta} \qquad \dfrac{\mathrm{D}_2^B}{\llbracket x t t \rrbracket_{\mathrm{N}} \beta :\cdot \Gamma, t{:}B \vdash \Delta}}{\llbracket x t t \rrbracket_{\mathrm{N}} \beta :\cdot \Gamma, t{:}A \cup B \vdash \Delta}}{\llbracket I y z \rrbracket_{\mathrm{N}} \delta \widehat{\delta} \setminus \widehat{t} \llbracket x t t \rrbracket_{\mathrm{N}} \alpha :\cdot \Gamma \vdash \Delta}$$

where $\Gamma = x{:}(A{\rightarrow}A{\rightarrow}C) \cap (B{\rightarrow}B{\rightarrow}C), y{:}D{\rightarrow}(A \cup B), z{:}D$, and $\Delta = \alpha{:}C$. In this derivation, D_2^A and D_2^B are identical, but for the type used for t, and the choice for $x{:}A{\rightarrow}A{\rightarrow}C$ or $x{:}B{\rightarrow}B{\rightarrow}C$. Now

$$\begin{aligned}
&\llbracket I y z \rrbracket_{\mathrm{N}} \delta \widehat{\delta} \setminus \widehat{t} \llbracket x t t \rrbracket_{\mathrm{N}} \alpha \\
&\quad = \llbracket I y z \rrbracket_{\mathrm{N}} \delta \widehat{\delta} \setminus \widehat{t}(\llbracket x t \rrbracket_{\mathrm{N}} \epsilon \widehat{\epsilon} \dagger \widehat{c}(< t, \mu > \widehat{\mu}\,[c]\ \widehat{d}{<}\, d, \alpha >)) \\
&\quad \rightarrow (\llbracket I y z \rrbracket_{\mathrm{N}} \delta \widehat{\delta} \setminus \widehat{t} \llbracket x t \rrbracket_{\mathrm{N}} \epsilon) \widehat{\epsilon} \dagger \widehat{c}(\llbracket I y z \rrbracket_{\mathrm{N}} \delta \widehat{\delta} \setminus \widehat{t}(< t, \mu > \widehat{\mu}\,[c]\ \widehat{d}{<}\, d, \alpha >)) \\
&\quad \rightarrow (\llbracket I y z \rrbracket_{\mathrm{N}} \delta \widehat{\delta} \setminus \widehat{t} \llbracket x t \rrbracket_{\mathrm{N}} \epsilon) \widehat{\epsilon} \dagger \widehat{c}((\llbracket I y z \rrbracket_{\mathrm{N}} \delta \widehat{\delta} \setminus \widehat{t}{<}\, t, \mu >) \widehat{\mu}\,[c]\ \widehat{d}(\llbracket I y z \rrbracket_{\mathrm{N}} \delta \widehat{\delta} \setminus \widehat{t}{<}\, d, \alpha >)) \\
&\quad \rightarrow (\llbracket I y z \rrbracket_{\mathrm{N}} \delta \widehat{\delta} \setminus \widehat{t} \llbracket x t \rrbracket_{\mathrm{N}} \epsilon) \widehat{\epsilon} \dagger \widehat{c}(\llbracket I y z \rrbracket_{\mathrm{N}} \delta \widehat{\delta}\,[c]\ \widehat{d}{<}\, d, \alpha >)
\end{aligned}$$

Now the last term above is not typeable. This can be observed from the fact that

the derivation would need to have the following shape:



but the subderivation D_3 does not exist: picking either A or B for t gives derivations for

$$[\![xt]\!]_{\mathrm{N}}\epsilon \,:\cdot\, \Gamma, t{:}A \vdash \epsilon{:}A{\to}C, \Delta \text{ and } [\![xt]\!]_{\mathrm{N}}\epsilon \,:\cdot\, \Gamma, t{:}B \vdash \epsilon{:}B{\to}C, \Delta$$

Notice that the types 'derived' for the plug ϵ differ, and do not permit the application of rule $(\cup L)$; there is no way around this problem.

In fact, the proofs of Theorems 6.1 and 6.2 strongly depend on the fact that, when we have a derivation for $P \,:\cdot\, \Gamma, t{:}A{\cup}B \vdash \Delta$, then rule $(\cup L)$ has been used to 'insert' the union type.

6 Preservance of types under conversion

In this section, we will perform the main 'soundness' checks of the system with intersection and union types as introduced for $\mathcal{X}$ above. We will show that the notion of type assignment is closed for both reduction and expansion, but that it is not with respect to the notion of η-reduction as introduced in Definition 3.3.

Theorem 6.1 (SUBJECT REDUCTION) *If $P \,:\cdot\, \Gamma \vdash \Delta$, and $P \to Q$, then $Q \,:\cdot\, \Gamma \vdash \Delta$.*
Proof: See the appendix.

Theorem 6.2 (SUBJECT EXPANSION) *If $Q \,:\cdot\, \Gamma \vdash \Delta$, and $P \to Q$, then $P \,:\cdot\, \Gamma \vdash \Delta$.*
Proof: See the appendix.

We will finish the presentation of the results of this paper by looking at the η-reduction rule, and show, as can be expected – seen that the notion of type assignment defined here for $\mathcal{X}$ is a natural extension of the strict system for λ-calculus – that type assignment in the system as presented here is not preserved by this rule.

Example 6.3 Take the η-rule for $\mathcal{X}$ as defined in Definition 3.3:

$$\widehat{x}(P\widehat{\gamma} \dagger \widehat{z}(<x,\delta>\widehat{\delta}\,[z]\,\widehat{w}<w,\beta>))\widehat{\beta}{\cdot}\alpha \to P\widehat{\gamma} \dagger \widehat{z}<z,\alpha>$$

Let A be strict; the following is a possible derivation for the left-hand side:

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\text{D}\\ P :\cdot \Gamma \vdash \gamma{:}A{\to}B, \Delta\end{array} \quad \dfrac{}{< x, \delta > :\cdot x{:}A{\cap}C \vdash \delta{:}A} \quad \dfrac{}{< w, \beta > :\cdot w{:}B \vdash \beta{:}B}}{< x, \delta > \widehat{\delta}\, [z]\, \widehat{w} < w, \beta > :\cdot x{:}A{\cap}C, z{:}A{\to}B \vdash \beta{:}B}}{P\widehat{\gamma} \dagger \widehat{z}(< x, \delta > \widehat{\delta}\, [z]\, \widehat{w} < w, \beta >) :\cdot \Gamma, x{:}A{\cap}C \vdash \beta{:}B.\Delta}}{\widehat{x}(P\widehat{\gamma} \dagger \widehat{z}(< x, \delta > \widehat{\delta}\, [z]\, \widehat{w} < w, \beta >))\widehat{\beta}{\cdot}\alpha :\cdot \Gamma \vdash \alpha{:}(A{\cap}C){\to}B, \Delta}$$

However, using the same derivation for P, we cannot derive

$$P\widehat{\gamma} \dagger \widehat{z} < z, \alpha > :\cdot \Gamma \vdash \alpha{:}(A{\cap}C){\to}B$$

At most, we can derive:

$$\dfrac{\begin{array}{c}\text{D}\\ P :\cdot \Gamma \vdash \gamma{:}A{\to}B, \Delta\end{array} \quad \dfrac{}{< z, \alpha > :\cdot z{:}A{\to}B \vdash \alpha{:}A{\to}B}}{P\widehat{\gamma} \dagger \widehat{z} < z, \alpha > :\cdot \Gamma \vdash \alpha{:}A{\to}B, \Delta}$$

As was the case for systems with intersection types for λ-calculus [9,2], in order to get a notion of type assignment that is closed for η-reduction, we would need to introduce a $\leq$-relation on types which is contra-variant in the arrow (see the discussion after Definition 4.2).

In such a system, in the style of [2], the (ax)-rule of Definition 5.1 could be replaced by:

$$(ax) : \dfrac{}{< y, \alpha > :\cdot \Gamma{\cap}y{:}A \vdash \alpha{:}B{\cup}\Delta}\ (A \leq B; A, B \textit{ strict})$$

With this new rule, we can derive the desired result:

$$\dfrac{\begin{array}{c}\text{D}\\ P :\cdot \Gamma \vdash \gamma{:}A{\to}B, \Delta\end{array} \quad \dfrac{}{< z, \alpha > :\cdot z{:}A{\to}B \vdash \alpha{:}(A{\cap}C){\to}B}}{P\widehat{\gamma} \dagger \widehat{z} < z, \alpha > :\cdot \Gamma \vdash \alpha{:}(A{\cap}C){\to}B, \Delta}$$

If this indeed gives a sound system (and a true extension of the system of [9,2]), will be left for future research.

Future work

There exists a whole plethora of directions of research that need exploration for $\mathcal{X}$. The one started with this paper, a notion of type assignment using intersection types, will need to be more strongly linked to existing systems, like those of [9,2,12]. Using those results, we want to look at the problem of termination, semantics, approximation, etc.

Also, in view of the striking similarities between the nets of $\mathcal{X}$ and the processes of the π-calculus, perhaps a suitable notion of type assignment using both intersection and union types can be defined for the latter.

Acknowledgements

I am greatly indebted to Pierre Lescanne, who not only suggested to use union types, but also drew my attention to the paper [12]. The solution for various problems noted there for a system with intersection types for Curien and Herbelein's calculus $\overline{\lambda}\mu\tilde{\mu}$ gave the final inspiration towards the type assignment rules defined here.

My thanks go also to Mariangiola Dezani, who pointed out a flaw in an earlier version 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. Strongly Normalising Cut-Elimination with Strict Intersection Types. In *Electronic Proceedings of International Workshop* Intersection Types and Related Systems 2002 *(ITRS '02), Copenhagen, Denmark*, volume 70.1 of *Electronic Notes in Theoretical Computer Science*, 2002.

[4] S. van Bakel. Cut-Elimination in the Strict Intersection Type Assignment System is Strongly Normalising. To appear in: *Notre Dame Journal of Formal Logic*, 2004.

[5] S. van Bakel, S. Lengrand, and P. Lescanne. The language §: computation and sequent calculus in classical logic. *Submitted*, 2004.

[6] S. van Bakel and J. Raghunandan. Implementing §. In *Electronic Proceedings of* TermGraph'04, *Rome, Italy*, Electronic Notes in Theoretical Computer Science, 2004. To appear.

[7] F. Barbanera and S. Berardi. A symmetric lambda calculus for classical program extraction. *Information and Computation*, 125(2):103–117, 1996.

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

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

[10] Pierre-Louis Curien and Hugo Herbelin. The duality of computation. In *Proceedings of the 5 th ACM SIGPLAN International Conference on Functional Programming (ICFP'00)*, pages 233–243. ACM, 2000.

[11] M. Dezani-Ciancaglini, U. de' Liguoro, and A. Piperno. Intersection and union types: syntax and semantics. *Information and Computation*, 119:202–230, 1995.

[12] Dan Dougherty, Sivia Ghilezan, and Pierre Lescanne. Characterizing strong normalization in a language with control operators. To appear, 2004.

[13] G. Gentzen. Investigations into logical deductions. In M.E. Szabo, editor, *Gentzen collected works*. North Holland, 1969. First published in 1935.

[14] Stéphane Lengrand. Call-by-value, call-by-name, and strong normalization for the classical sequent calculus. In Bernhard Gramlich and Salvador Lucas, editors, *Electronic Notes in Theoretical Computer Science*, volume 86. Elsevier, 2003.

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

A Conversion proofs

We start with the proof of a subject-reduction result. Notice the use of derivation rules $(\cup L)$ in the case for (*exp-outs*$\not\prec$) shown, and $(\cap R)$ in the one for ($\not\setminus$*imp-outs*).

Theorem A.1 (SUBJECT REDUCTION) *If* $P :\cdot\ \Gamma \vdash \Delta$, *and* $P \to Q$, *then* $Q :\cdot\ \Gamma \vdash \Delta$.

Proof: By induction on the definition of $\to$, where we focus on the rules: the proof consists of showing, for each rule, the 'minimal' derivation for the left-hand side, and that, using the restrictions that poses, we can type the right-hand side. We only show the interesting cases.

(*Logical rules*): (*exp-imp*): $(\widehat{y}P\widehat{\beta}\cdot\alpha)\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R) \to (Q\widehat{\gamma} \dagger \widehat{y}P)\widehat{\beta} \dagger \widehat{z}R$, with $\alpha \notin \mathit{fp}(R), x \notin \mathit{fs}(Q,R)$.

$$\dfrac{\dfrac{\mathrm{D}_1}{P :\cdot\ \Gamma_1, y{:}A \vdash \beta{:}B, \Delta_1}}{\widehat{y}P\widehat{\beta}\cdot\alpha :\cdot\ \Gamma_1 \vdash \alpha{:}A{\to}B, \Delta_1} \qquad \dfrac{\dfrac{\mathrm{D}_2}{Q :\cdot\ \Gamma_2 \vdash \gamma{:}A, \Delta_2} \quad \dfrac{\mathrm{D}_3}{R :\cdot\ \Gamma_3, z{:}B \vdash \Delta_3}}{Q\widehat{\gamma}\,[x]\,\widehat{z}P :\cdot\ \Gamma_2{\cap}\Gamma_3, x{:}A{\to}B \vdash \Delta_2{\cup}\Delta_3}$$

$$\dfrac{}{(\widehat{y}P\widehat{\beta}\cdot\alpha)\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R) :\cdot\ \Gamma_1{\cap}\Gamma_2{\cap}\Gamma_3 \vdash \Delta_1{\cup}\Delta_2{\cup}\Delta_3}$$

Notice that $y, \beta \notin \mathit{fc}(Q,R)$.

$$\dfrac{\dfrac{\dfrac{\mathrm{D}_2}{Q :\cdot\ \Gamma_2 \vdash \gamma{:}A, \Delta_2} \quad \dfrac{\mathrm{D}_1}{P :\cdot\ \Gamma_1, y{:}A \vdash \beta{:}B, \Delta_1}}{Q\widehat{\gamma} \dagger \widehat{y}P :\cdot\ \Gamma_1{\cap}\Gamma_2 \vdash \beta{:}B, \Delta_1{\cup}\Delta_2} \qquad \dfrac{\mathrm{D}_3}{R :\cdot\ \Gamma_3, z{:}B \vdash \Delta_3}}{(Q\widehat{\gamma} \dagger \widehat{y}P)\widehat{\beta} \dagger \widehat{z}R :\cdot\ \Gamma_1{\cap}\Gamma_2{\cap}\Gamma_3 \vdash \Delta_1{\cup}\Delta_2{\cup}\Delta_3}$$

(*CBV propagation*): (*exp-outs*$\not\prec$): $(\widehat{y}P\widehat{\beta}\cdot\alpha)\widehat{\alpha} \not\prec \widehat{x}Q \to (\widehat{y}(P\widehat{\alpha} \not\prec \widehat{x}Q)\widehat{\beta}\cdot\gamma)\widehat{\gamma} \dagger \widehat{x}Q$, with γ fresh. Notice that α is not introduced in $\widehat{y}P\widehat{\beta}\cdot\alpha$, so might appear inside P; also, $y, \beta \notin \mathit{fc}(Q)$.

$$\dfrac{\dfrac{\dfrac{\mathrm{D}_1}{P :\cdot\ \Gamma_1, y{:}A \vdash \beta{:}B, \alpha{:}C, \Delta_1}}{\widehat{y}P\widehat{\beta}\cdot\alpha :\cdot\ \Gamma_1 \vdash \alpha{:}(A{\to}B){\cup}C, \Delta_1} \quad \dfrac{\dfrac{\mathrm{D}_2}{Q :\cdot\ \Gamma_2, x{:}A{\to}B \vdash \Delta_2} \quad \dfrac{\mathrm{D}_3}{Q :\cdot\ \Gamma_3, x{:}C \vdash \Delta_3}}{Q :\cdot\ \Gamma_2{\cap}\Gamma_3, x{:}(A{\to}B){\cup}C \vdash \Delta_2{\cup}\Delta_3}\,(\cup L)}{(\widehat{y}P\widehat{\beta}\cdot\alpha)\widehat{\alpha} \not\prec \widehat{x}Q :\cdot\ \Gamma_1{\cap}\Gamma_2{\cap}\Gamma_3 \vdash \Delta_1{\cup}\Delta_2{\cup}\Delta_3}$$

$$\dfrac{\dfrac{\dfrac{\mathsf{D}_1 \qquad \mathsf{D}_2}{P :\cdot \Gamma_1, y{:}A \vdash \beta{:}B, \alpha{:}A{\to}B, \Delta_1 \quad Q :\cdot \Gamma_2, x{:}A{\to}B \vdash \Delta_2}}{P\widehat{\alpha} \not\leftarrow \widehat{x}Q :\cdot \Gamma_1{\cap}\Gamma_2, y{:}A \vdash \beta{:}B, \Delta_1{\cup}\Delta_2}}{\widehat{y}(P\widehat{\alpha} \not\leftarrow \widehat{x}Q)\widehat{\beta}{\cdot}\gamma :\cdot \Gamma_1{\cap}\Gamma_2 \vdash \gamma{:}A{\to}B, \Delta_1{\cup}\Delta_2} \qquad \dfrac{\mathsf{D}_3}{Q :\cdot \Gamma_3, x{:}A{\to}B \vdash \Delta_3}}{(\widehat{y}(P\widehat{\alpha} \not\leftarrow \widehat{x}Q)\widehat{\beta}{\cdot}\gamma)\widehat{\gamma} \dagger \widehat{x}Q :\cdot \Gamma_1{\cap}\Gamma_2{\cap}\Gamma_3 \vdash \Delta_1{\cup}\Delta_2{\cup}\Delta_3}$$

(*CBN propagation*): ($\not\leftarrow$*imp-outs*):
$P\widehat{\alpha} \not\leftarrow \widehat{x}(Q\widehat{\beta}\,[x]\,\widehat{y}R) \;\to\; P\widehat{\alpha} \dagger \widehat{v}((P\widehat{\alpha} \not\leftarrow \widehat{x}Q)\widehat{\beta}\,[v]\,\widehat{y}(P\widehat{\alpha} \not\leftarrow \widehat{x}R))$, with v fresh. Notice that $y, \beta \notin \mathit{fc}(P)$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathsf{D}_1 \qquad \mathsf{D}_2}{P :\cdot \Gamma_1 \vdash \alpha{:}A{\to}B, \Delta_1 \quad P :\cdot \Gamma_2 \vdash \alpha{:}C, \Delta_2}(\cap R)}{P :\cdot \Gamma_1{\cap}\Gamma_2 \vdash \alpha{:}A{\to}B{\cap}C, \Delta_1{\cup}\Delta_2} \qquad \dfrac{\mathsf{D}_3 \qquad \mathsf{D}_4}{Q :\cdot \Gamma_3, x{:}C \vdash \beta{:}A, \Delta_3 \quad R :\cdot \Gamma_4, y{:}B, x{:}C \vdash \Delta_4}}{P\widehat{\alpha} \not\leftarrow \widehat{x}(Q\widehat{\beta}\,[x]\,\widehat{y}R) :\cdot \Gamma_1{\cap}\cdots{\cap}\Gamma_4 \vdash \Delta_1{\cup}\cdots{\cup}\Delta_4}$$

$$\dfrac{\dfrac{\dfrac{\mathsf{D}_1}{P :\cdot \Gamma_1 \vdash \alpha{:}A{\to}B, \Delta_1} \qquad \dfrac{\dfrac{\mathsf{D}_2 \qquad \mathsf{D}_3}{P :\cdot \Gamma_2 \vdash \alpha{:}C, \Delta_2 \quad Q :\cdot \Gamma_3, x{:}C \vdash \beta{:}A, \Delta_3} \qquad \dfrac{\mathsf{D}_2 \qquad \mathsf{D}_4}{P :\cdot \Gamma_2 \vdash \alpha{:}C, \Delta_2 \quad R :\cdot \Gamma_4, y{:}B, x{:}C \vdash \Delta_4}}{P\widehat{\alpha} \not\leftarrow \widehat{x}Q :\cdot \Gamma_2{\cap}\Gamma_3 \vdash \beta{:}A, \Delta_2{\cup}\Delta_3 \qquad P\widehat{\alpha} \not\leftarrow \widehat{x}R :\cdot \Gamma_2{\cap}\Gamma_4, y{:}B \vdash \Delta_2{\cup}\Delta_4}}{(P\widehat{\alpha} \not\leftarrow \widehat{x}Q)\widehat{\beta}\,[v]\,\widehat{y}(P\widehat{\alpha} \not\leftarrow \widehat{x}R) :\cdot \Gamma_2{\cap}\Gamma_3{\cap}\Gamma_4, v{:}A{\to}B \vdash \Delta_2{\cup}\Delta_3{\cup}\Delta_4}}{P\widehat{\alpha} \dagger \widehat{v}((P\widehat{\alpha} \not\leftarrow \widehat{x}Q)\widehat{\beta}\,[v]\,\widehat{y}(P\widehat{\alpha} \not\leftarrow \widehat{x}R)) :\cdot \Gamma_1{\cap}\cdots{\cap}\Gamma_4 \vdash \Delta_1{\cup}\cdots{\cup}\Delta_4}$$

Next we show a subject-expansion result. Notice the use of the derivation rules ($\cup L$) in the cases (*exp-outs*$\not\leftarrow$), (*imp*$\not\leftarrow$), and (*cut*$\not\leftarrow$), and that of ($\cap R$) in the cases ($\not\leftarrow$*imp-outs*), ($\not\leftarrow$*imp-ins*), and ($\not\leftarrow$*cut*).

Theorem A.2 (SUBJECT EXPANSION) *If* $Q :\cdot \Gamma \vdash \Delta$, *and* $P \to Q$, *then* $P :\cdot \Gamma \vdash \Delta$.

Proof: By induction on the definition of $\to$, where we focus on the rules: the proof consists of showing, for each rule, the 'minimal' derivation for the right-hand side, and that, using the restrictions that poses, we can type the left-hand side. We will only show the interesting cases.

(*CBV propagation*): (*cap*$\not\leftarrow$): $\langle y.\beta\rangle\widehat{\alpha} \not\leftarrow \widehat{x}P \to \langle y.\beta\rangle$, with $\beta \neq \alpha$. Notice the use

of the type $\top$.

$$\dfrac{}{\langle y.\beta\rangle \;:\cdot\; y{:}A \vdash \beta{:}A} \qquad \dfrac{\dfrac{\dfrac{}{\langle y.\beta\rangle \;:\cdot\; y{:}A \vdash \beta{:}A}}{\langle y.\beta\rangle \;:\cdot\; y{:}A \vdash \alpha{:}\top, \beta{:}A}\,(W) \qquad \dfrac{}{P \;:\cdot\; x{:}\top \vdash \emptyset}\,()}{\langle y.\beta\rangle\widehat{\alpha} \nearrow \widehat{x}P \;:\cdot\; y{:}A \vdash \beta{:}A}$$

(*exp-outs*$\nearrow$): $(\widehat{y}P\widehat{\beta}\cdot\alpha)\widehat{\alpha} \nearrow \widehat{x}Q \to (\widehat{y}(P\widehat{\alpha} \nearrow \widehat{x}Q)\widehat{\beta}\cdot\gamma)\widehat{\gamma} \dagger \widehat{x}Q$, with γ fresh. Notice that $y, \beta \notin \mathit{fc}(Q)$.

$$\dfrac{\dfrac{\dfrac{\boxed{\mathsf{D}_1} \qquad \boxed{\mathsf{D}_2}}{P \;:\cdot\; \Gamma_1, y{:}A \vdash \alpha{:}C, \beta{:}B, \Delta_1 \quad Q \;:\cdot\; \Gamma_2, x{:}C \vdash \Delta_2}}{P\widehat{\alpha} \nearrow \widehat{x}Q \;:\cdot\; \Gamma_1{\cap}\Gamma_2, y{:}A \vdash \beta{:}B, \Delta_1{\cup}\Delta_2}}{\widehat{y}(P\widehat{\alpha} \nearrow \widehat{x}Q)\widehat{\beta}\cdot\gamma \;:\cdot\; \Gamma_1{\cap}\Gamma_2 \vdash \gamma{:}A{\to}B, \Delta_1{\cup}\Delta_2} \qquad \dfrac{\boxed{\mathsf{D}_3}}{Q \;:\cdot\; \Gamma_3, x{:}A{\to}B \vdash \Delta_3}}{(\widehat{y}(P\widehat{\alpha} \nearrow \widehat{x}Q)\widehat{\beta}\cdot\gamma)\widehat{\gamma} \nearrow \widehat{x}Q \;:\cdot\; \Gamma_1{\cap}\Gamma_2{\cap}\Gamma_3 \vdash \Delta_1{\cup}\Delta_2{\cup}\Delta_3}$$

$$\dfrac{\dfrac{\boxed{\mathsf{D}_1}}{P \;:\cdot\; \Gamma_1, y{:}A \vdash \alpha{:}C, \beta{:}B, \Delta_1}}{\widehat{y}P\widehat{\beta}\cdot\alpha \;:\cdot\; \Gamma_1 \vdash \alpha{:}(A{\to}B){\cup}C, \Delta_1} \qquad \dfrac{\dfrac{\boxed{\mathsf{D}_2} \qquad \boxed{\mathsf{D}_3}}{Q \;:\cdot\; \Gamma_2, x{:}C \vdash \Delta_2 \quad Q \;:\cdot\; \Gamma_3, x{:}A{\to}B \vdash \Delta_3}}{Q \;:\cdot\; \Gamma_2{\cap}\Gamma_3, x{:}(A{\to}B){\cup}C \vdash \Delta_2{\cup}\Delta_3}\,(\cup L)}{(\widehat{y}P\widehat{\beta}\cdot\alpha)\widehat{\alpha} \nearrow \widehat{x}Q \;:\cdot\; \Gamma \vdash \Delta}$$

(*imp*$\nearrow$): $(P\widehat{\beta}\,[z]\,\widehat{y}Q)\widehat{\alpha} \nearrow \widehat{x}R \to (P\widehat{\alpha} \nearrow \widehat{x}R)\widehat{\beta}\,[z]\,\widehat{y}(Q\widehat{\alpha} \nearrow \widehat{x}R)$. Notice that $y, \beta \notin \mathit{fc}(R)$.

$$\dfrac{\dfrac{\dfrac{\boxed{\mathsf{D}_1} \qquad \boxed{\mathsf{D}_2}}{P \;:\cdot\; \Gamma_1 \vdash \alpha{:}C, \beta{:}A, \Delta_1 \quad R \;:\cdot\; \Gamma_2, x{:}C \vdash \Delta_2} \qquad \dfrac{\boxed{\mathsf{D}_3} \qquad \boxed{\mathsf{D}_4}}{Q \;:\cdot\; \Gamma_3, y{:}B \vdash \alpha{:}D, \Delta_3 \quad R \;:\cdot\; \Gamma_4, x{:}D \vdash \Delta_4}}{P\widehat{\alpha} \nearrow \widehat{x}R \;:\cdot\; \Gamma_1{\cap}\Gamma_2 \vdash \beta{:}A, \Delta_1{\cup}\Delta_2 \qquad Q\widehat{\alpha} \nearrow \widehat{x}R \;:\cdot\; \Gamma_3{\cap}\Gamma_4, y{:}B \vdash \Delta_3{\cup}\Delta_4}}{(P\widehat{\alpha} \nearrow \widehat{x}R)\widehat{\beta}\,[z]\,\widehat{y}(Q\widehat{\alpha} \nearrow \widehat{x}R) \;:\cdot\; \Gamma_1{\cap}\cdots{\cap}\Gamma_4{\cap}z{:}A{\to}B \vdash \Delta_1{\cup}\cdots{\cup}\Delta_4}$$

$$\dfrac{\dfrac{\dfrac{\boxed{\mathsf{D}_1} \qquad \boxed{\mathsf{D}_3}}{P \;:\cdot\; \Gamma_1 \vdash \alpha{:}C, \beta{:}A, \Delta_1 \quad Q \;:\cdot\; \Gamma_3, y{:}B \vdash \alpha{:}D, \Delta_3} \qquad \dfrac{\boxed{\mathsf{D}_2} \qquad \boxed{\mathsf{D}_4}}{R \;:\cdot\; \Gamma_2, x{:}C \vdash \Delta_2 \quad R \;:\cdot\; \Gamma_4, x{:}D \vdash \Delta_4}}{P\widehat{\beta}\,[z]\,\widehat{y}Q \;:\cdot\; \Gamma_1{\cap}\Gamma_3{\cap}z{:}A{\to}B \vdash \alpha{:}C{\cup}D, \Delta_1{\cup}\Delta_3 \qquad R \;:\cdot\; \Gamma_2{\cap}\Gamma_4, x{:}C{\cup}D \vdash \Delta_2{\cup}\Delta_4}\,(\cup L)}{(P\widehat{\beta}\,[z]\,\widehat{y}Q)\widehat{\alpha} \nearrow \widehat{x}R \;:\cdot\; \Gamma_1{\cap}\cdots{\cap}\Gamma_4{\cap}z{:}A{\to}B \vdash \Delta_1{\cup}\cdots{\cup}\Delta_4}$$

(*cut*$\nearrow$): $(P\widehat{\beta} \dagger \widehat{y}Q)\widehat{\alpha} \nearrow \widehat{x}R \to (P\widehat{\alpha} \nearrow \widehat{x}R)\widehat{\beta} \dagger \widehat{y}(Q\widehat{\alpha} \nearrow \widehat{x}R)$. Notice that $y, \beta \notin \mathit{fc}(R)$.

$$\dfrac{\dfrac{\dfrac{\boxed{\mathsf{D}_1} \qquad \boxed{\mathsf{D}_2}}{P \;:\cdot\; \Gamma_1 \vdash \alpha{:}C, \beta{:}B, \Delta_1 \quad R \;:\cdot\; \Gamma_2, x{:}C \vdash \Delta_2} \qquad \dfrac{\boxed{\mathsf{D}_3} \qquad \boxed{\mathsf{D}_4}}{Q \;:\cdot\; \Gamma_3, y{:}B \vdash \alpha{:}D, \Delta_3 \quad R \;:\cdot\; \Gamma_4, x{:}D \vdash \Delta_4}}{P\widehat{\alpha} \nearrow \widehat{x}R \;:\cdot\; \Gamma_1{\cap}\Gamma_2 \vdash \beta{:}B, \Delta_1{\cup}\Delta_2 \qquad Q\widehat{\alpha} \nearrow \widehat{x}R \;:\cdot\; \Gamma_3{\cap}\Gamma_4, y{:}B \vdash \Delta_3{\cup}\Delta_4}}{(P\widehat{\alpha} \nearrow \widehat{x}R)\widehat{\beta} \dagger \widehat{y}(Q\widehat{\alpha} \nearrow \widehat{x}R) \;:\cdot\; \Gamma_1{\cap}\cdots{\cap}\Gamma_4 \vdash \Delta_1{\cup}\cdots{\cup}\Delta_4}$$

$$\dfrac{\dfrac{\overset{\mathrm{D}_1}{\vartriangle}\quad \overset{\mathrm{D}_3}{\vartriangle}}{P :\cdot \Gamma_1 \vdash \alpha{:}C, \beta{:}B, \Delta_1 \quad Q :\cdot \Gamma_3, y{:}B \vdash \alpha{:}D, \Delta_3}}{P\widehat{\beta} \dagger \widehat{y}Q :\cdot \Gamma_1 \cap \Gamma_3 \vdash \alpha{:}C \cup D, \Delta_1 \cup \Delta_3} \quad \dfrac{\overset{\mathrm{D}_2}{\vartriangle}\quad \overset{\mathrm{D}_4}{\vartriangle}}{R :\cdot \Gamma_2, x{:}C \vdash \Delta_2 \quad R :\cdot \Gamma_4, x{:}D \vdash \Delta_4}}{R :\cdot \Gamma_2 \cap \Gamma_4, x{:}C \cup D \vdash \Delta_2 \cup \Delta_4}\,(\cup L)}{(P\widehat{\beta} \dagger \widehat{y}Q)\widehat{\alpha} \nleftarrow \widehat{x}R :\cdot \Gamma_1 \cap \cdots \cap \Gamma_4 \vdash \Delta_1 \cup \cdots \cup \Delta_4}$$

(*CBN propagation*): ($\nleftarrow$*cap*): $P\widehat{\alpha} \nleftarrow \widehat{x}\langle y.\beta\rangle \to \langle y.\beta\rangle,\ y \neq x$. Again, notice the use of the type $\top$.

$$\dfrac{\dfrac{}{\langle y.\beta\rangle :\cdot y{:}B \vdash \beta{:}B} \quad \dfrac{\dfrac{}{P :\cdot \emptyset \vdash \alpha{:}\top}\,()\quad \dfrac{\dfrac{}{\langle y.\beta\rangle :\cdot y{:}B \vdash \beta{:}B}}{\langle y.\beta\rangle :\cdot y{:}B, x{:}\top \vdash \beta{:}B}}{P\widehat{\alpha} \nleftarrow \widehat{x}\langle y.\beta\rangle :\cdot y{:}B \vdash \beta{:}B}\,(W)$$

($\nleftarrow$*imp-outs*): $P\widehat{\alpha} \nleftarrow \widehat{x}(Q\widehat{\beta}\,[x]\,\widehat{y}R) \to P\widehat{\alpha} \dagger \widehat{v}((P\widehat{\alpha} \nleftarrow \widehat{x}Q)\widehat{\beta}\,[v]\,\widehat{y}(P\widehat{\alpha} \nleftarrow \widehat{x}R))$, with v fresh. Notice that $y, \beta \notin \mathit{fc}(P)$.

$$\dfrac{\dfrac{\overset{\mathrm{D}_1}{\vartriangle}}{P :\cdot \Gamma_1 \vdash \alpha{:}A{\to}B, \Delta_1} \quad \dfrac{\dfrac{\overset{\mathrm{D}_2}{\vartriangle}\quad \overset{\mathrm{D}_3}{\vartriangle}}{P :\cdot \Gamma_2 \vdash \alpha{:}C, \Delta_2 \quad Q :\cdot \Gamma_3, x{:}C \vdash \beta{:}A, \Delta_3}}{P\widehat{\alpha} \nleftarrow \widehat{x}Q :\cdot \Gamma_2 \cap \Gamma_3 \vdash \beta{:}A, \Delta_2 \cup \Delta_3} \quad \dfrac{\overset{\mathrm{D}_4}{\vartriangle}\quad \overset{\mathrm{D}_5}{\vartriangle}}{P :\cdot \Gamma_4 \vdash \alpha{:}D, \Delta_4 \quad R :\cdot \Gamma_5, x{:}D, y{:}B \vdash \Delta_5}}{P\widehat{\alpha} \nleftarrow \widehat{x}R :\cdot \Gamma_4 \cap \Gamma_5, y{:}B \vdash \Delta_4 \cup \Delta_5}}{(P\widehat{\alpha} \nleftarrow \widehat{x}Q)\widehat{\beta}\,[v]\,\widehat{y}(P\widehat{\alpha} \nleftarrow \widehat{x}R) :\cdot \Gamma_2 \cap \cdots \cap \Gamma_5, v{:}A{\to}B \vdash \Delta_2 \cup \cdots \cup \Delta_5}}{P\widehat{\alpha} \dagger \widehat{v}((P\widehat{\alpha} \nleftarrow \widehat{x}Q)\widehat{\beta}\,[v]\,\widehat{y}(P\widehat{\alpha} \nleftarrow \widehat{x}R)) :\cdot \Gamma_1 \cap \cdots \cap \Gamma_5 \vdash \Delta_1 \cup \cdots \cup \Delta_5}$$

$$\dfrac{\dfrac{\overset{\mathrm{D}_1}{\vartriangle}\quad \overset{\mathrm{D}_2}{\vartriangle}\quad \overset{\mathrm{D}_4}{\vartriangle}}{P :\cdot \Gamma_1 \vdash \alpha{:}A{\to}B, \Delta_1 \quad P :\cdot \Gamma_2 \vdash \alpha{:}C, \Delta_2 \quad P :\cdot \Gamma_4 \vdash \alpha{:}D, \Delta_4}\,(\cap R)}{P :\cdot \Gamma_1 \cap \Gamma_2 \cap \Gamma_4 \vdash \alpha{:}(A{\to}B) \cap C \cap D, \Delta_1 \cup \Delta_2 \cup \Delta_4} \quad \dfrac{\overset{\mathrm{D}_3}{\vartriangle}\quad \overset{\mathrm{D}_5}{\vartriangle}}{Q :\cdot \Gamma_3, x{:}C \vdash \beta{:}A, \Delta_3 \quad R :\cdot \Gamma_5, x{:}D, y{:}B \vdash \Delta_5}}{Q\widehat{\beta}\,[x]\,\widehat{y}R :\cdot \Gamma_3 \cap \Gamma_5, x{:}A{\to}B \cap C \cap D \vdash \Delta_3 \cup \Delta_5}}{P\widehat{\alpha} \nleftarrow \widehat{x}(Q\widehat{\beta}\,[x]\,\widehat{y}R) :\cdot \Gamma_1 \cap \cdots \cap \Gamma_5 \vdash \Delta_1 \cup \cdots \cup \Delta_5}$$

($\nleftarrow$*imp-ins*): $P\widehat{\alpha} \nleftarrow \widehat{x}(Q\widehat{\beta}\,[z]\,\widehat{y}R) \to (P\widehat{\alpha} \nleftarrow \widehat{x}Q)\widehat{\beta}\,[z]\,\widehat{y}(P\widehat{\alpha} \nleftarrow \widehat{x}R),\ x \neq z$. Notice that $y, \beta \notin \mathit{fc}(P)$.

$$\dfrac{\dfrac{\overset{\mathrm{D}_1}{\vartriangle}\quad \overset{\mathrm{D}_2}{\vartriangle}}{P :\cdot \Gamma_1 \vdash \alpha{:}C, \Delta_1 \quad Q :\cdot \Gamma_2, x{:}C \vdash \beta{:}A, \Delta_2}}{P\widehat{\alpha} \nleftarrow \widehat{x}Q :\cdot \Gamma_1 \cap \Gamma_2 \vdash \beta{:}A, \Delta_1 \cup \Delta_2} \quad \dfrac{\overset{\mathrm{D}_3}{\vartriangle}\quad \overset{\mathrm{D}_4}{\vartriangle}}{P :\cdot \Gamma_3 \vdash \alpha{:}D, \Delta_3 \quad R :\cdot \Gamma_4, x{:}D, y{:}B \vdash \Delta_4}}{P\widehat{\alpha} \nleftarrow \widehat{x}R :\cdot \Gamma_3 \cap \Gamma_4, y{:}B \vdash \Delta_3 \cup \Delta_4}}{(P\widehat{\alpha} \nleftarrow \widehat{x}Q)\widehat{\beta}\,[z]\,\widehat{y}(P\widehat{\alpha} \nleftarrow \widehat{x}R) :\cdot \Gamma_1 \cap \cdots \cap \Gamma_4 \cap z{:}A{\to}B \vdash \Delta_1 \cup \cdots \cup \Delta_4}$$

$$\dfrac{\dfrac{\overset{\mathsf{D}_1}{\vdots}\quad \overset{\mathsf{D}_3}{\vdots}}{P \mathrel{:\cdot} \Gamma_1 \vdash \alpha{:}C, \Delta_1 \quad P \mathrel{:\cdot} \Gamma_3 \vdash \alpha{:}D, \Delta_3}\,(\cap R) \quad \dfrac{\overset{\mathsf{D}_2}{\vdots}\quad \overset{\mathsf{D}_4}{\vdots}}{Q \mathrel{:\cdot} \Gamma_2, x{:}C \vdash \beta{:}A, \Delta_2 \quad R \mathrel{:\cdot} \Gamma_4, x{:}D, y{:}B \vdash \Delta_4}}{P \mathrel{:\cdot} \Gamma_1 \cap \Gamma_3 \vdash \alpha{:}C \cap D, \Delta_1 \cup \Delta_3 \quad Q\widehat{\beta}\,[z]\,\widehat{y}R \mathrel{:\cdot} \Gamma_2 \cap \Gamma_4 \cap z{:}A \to B, x{:}C \cap D \vdash \Delta_2 \cup \Delta_4}$$
$$P\widehat{\alpha} \not\leftarrow \widehat{x}(Q\widehat{\beta}\,[z]\,\widehat{y}R) \mathrel{:\cdot} \Gamma_1 \cap \cdots \cap \Gamma_4 \cap z{:}A \to B \vdash \Delta_1 \cup \cdots \cup \Delta_4$$

($\nleftarrow$*cut*): $P\widehat{\alpha} \nleftarrow \widehat{x}(Q\widehat{\beta} \dagger \widehat{y}R) \to (P\widehat{\alpha} \nleftarrow \widehat{x}Q)\widehat{\beta} \dagger \widehat{y}(P\widehat{\alpha} \nleftarrow \widehat{x}R)$. Notice that $y, \beta \notin$ *fc*(P).

$$\dfrac{\dfrac{\overset{\mathsf{D}_1}{\vdots}\quad \overset{\mathsf{D}_2}{\vdots}}{P \mathrel{:\cdot} \Gamma_1 \vdash \alpha{:}A, \Delta_1 \quad Q \mathrel{:\cdot} \Gamma_2, x{:}A \vdash \beta{:}B, \Delta_2}{P\widehat{\alpha} \nleftarrow \widehat{x}Q \mathrel{:\cdot} \Gamma_1 \cap \Gamma_2 \vdash \beta{:}B, \Delta_1 \cup \Delta_2} \quad \dfrac{\overset{\mathsf{D}_3}{\vdots}\quad \overset{\mathsf{D}_4}{\vdots}}{P \mathrel{:\cdot} \Gamma_3 \vdash \alpha{:}C, \Delta_3 \quad R \mathrel{:\cdot} \Gamma_4, x{:}C, y{:}B \vdash \Delta_4}{P\widehat{\alpha} \nleftarrow \widehat{x}R \mathrel{:\cdot} \Gamma_3 \cap \Gamma_4, y{:}B \vdash \Delta_3 \cup \Delta_4}}{(P\widehat{\alpha} \nleftarrow \widehat{x}Q)\widehat{\beta} \dagger \widehat{y}(P\widehat{\alpha} \nleftarrow \widehat{x}R) \mathrel{:\cdot} \Gamma_1 \cap \cdots \cap \Gamma_4 \vdash \Delta_1 \cup \cdots \cup \Delta_4}$$

$$\dfrac{\dfrac{\overset{\mathsf{D}_1}{\vdots}\quad \overset{\mathsf{D}_3}{\vdots}}{P \mathrel{:\cdot} \Gamma_1 \vdash \alpha{:}A, \Delta_1 \quad P \mathrel{:\cdot} \Gamma_3 \vdash \alpha{:}C, \Delta_3}\,(\cap R) \quad \dfrac{\overset{\mathsf{D}_2}{\vdots}\quad \overset{\mathsf{D}_4}{\vdots}}{Q \mathrel{:\cdot} \Gamma_2, x{:}A \vdash \beta{:}B, \Delta_2 \quad R \mathrel{:\cdot} \Gamma_4, x{:}C, y{:}B \vdash \Delta_4}}{P \mathrel{:\cdot} \Gamma_1 \cap \Gamma_3 \vdash \alpha{:}A \cap C, \Delta_1 \cup \Delta_3 \quad Q\widehat{\beta} \dagger \widehat{y}R \mathrel{:\cdot} \Gamma_2 \cap \Gamma_4, x{:}A \cap C \vdash \Delta_2 \cup \Delta_4}$$
$$P\widehat{\alpha} \nleftarrow \widehat{x}(Q\widehat{\beta} \dagger \widehat{y}R) \mathrel{:\cdot} \Gamma_1 \cap \cdots \cap \Gamma_4 \cap z{:}A \to B \vdash \Delta_1 \cup \cdots \cup \Delta_4$$