Intersection type assignment systems for intuitionistic sequent calculus

José Espírito Santo[1], Jelena Ivetić[2], Silvia Likavec[2]

[1] Mathematics Department, University of Minho, Portugal
jes@math.uminho.pt

[2] Faculty of Engineering, University of Novi Sad, Serbia
jelena@imft.ftn.ns.ac.yu,likavec@di.unito.it

Abstract

We present and analyse various intersection type assignment systems for the λ^{Gtz}-calculus, a calculus that embodies the Curry-Howard correspondence for intuitionistic sequent calculus. Three systems $\lambda^{\mathsf{Gtz}}\cap$, $\lambda^{\mathsf{Gtz}}\cap_{\circ}$ and $\lambda^{\mathsf{Gtz}}\cap_{\sharp}$ successfully characterise the strongly normalising terms in λ^{Gtz}. The first one is presented as a refinement of two previous, unsuccessful attempts. The latter two try to reduce to a minimum the use of the meta-level type equivalence. The management of intersection is in all successful systems built in the ordinary logical rules.

Keywords: intersection types, intuitionistic logic, sequent calculus.

1 Introduction

The intersection type assignment systems were introduced by Coppo and Dezani [2, 3], Barendregt et al. [1], Copo et al. [4], Pottinger [16], and Sallé [18]. They extend the simple type assignment system $\lambda\rightarrow$ so that a refined study of both syntax and semantics of the ordinary λ-calculus is possible (*e.g.* characterisation of normalising terms, analysis of models).

Meanwhile, the ordinary λ-calculus has been extended in several ways, as an answer to stimuli coming from different sources. For instance, the λ-calculi with explicit substitutions have a computer science motivation (more precisely the implementation of functional programs and other symbolic systems). Many other extensions are motivated by logic and, more specifically, by the extension of the Curry-Howard correspondence. Examples are Parigot's $\lambda\mu$-calculus (for classical natural deduction), Herbelin's $\overline{\lambda}$-calculus (for a fragment of the intuitionistic sequent calculus), Joachimski-Matthes' λ-calculus with generalised application (for von Plato's system of natural deduction), and Curien-Herbelin's $\overline{\lambda}\mu\tilde{\mu}$-calculus (for classical sequent calculus). As an answer to this expansion of the field of application, intersection type assignment systems are being defined and studied for almost all of the mentioned extensions [14, 15, 5, 6, 13].

Our main concern in this paper is the design of the type assignment systems for the λ^{Gtz} calculus, introduced in [7], and corresponding under the Curry-Howard correspondence to the intuitionistic sequent calculus. The original calculus had a system of simple types and was later extended in [8] to an intersection type assignment system $\lambda^{\mathsf{Gtz}}\cap$ which characterises the strongly normalising terms (*i.e.* terms representing sequent calculus derivations on which cut-elimination always terminates).

In $\lambda^{\mathsf{Gtz}}\cap$, the management of intersection is built in the ordinary logical rules, and the system relies on the meta-level treatment of $\leq$ and equivalence of types. The purpose of this paper is to present $\lambda^{\mathsf{Gtz}}\cap$ and report on various alternative formulations of the system. Two of them are not successful and we explain why they fail and how they lead us to the system $\lambda^{\mathsf{Gtz}}\cap$. The two last (and novel) alternatives are successful both in the sense that their induced notion of typeability is the same as that of $\lambda^{\mathsf{Gtz}}\cap$, and that equivalence of types is reduced to a minimum.

The paper is organised as follows. Syntax and reduction rules of the untyped λ^{Gtz}-calculus are given in Section 2. In Section 3 we introduce and study some properties of four intersection type assignment systems that extend the system of simple types for the λ^{Gtz}-calculus. We start with the unsuccessful variants $\lambda^{\mathsf{Gtz}}\cap_{\mathsf{l}}$ and $\lambda^{\mathsf{Gtz}}\cap_{\mathsf{R}}$, followed by $\lambda^{\mathsf{Gtz}}\cap$, and finally we consider the novel systems $\lambda^{\mathsf{Gtz}}\cap_{\circ}$ and $\lambda^{\mathsf{Gtz}}\cap_{\sharp}$. We conclude in Section 4 and give some directions for future work.

2 The syntax of the λ^{Gtz}-calculus

The λ^{Gtz}-calculus was proposed by Espirito Santo [7] as a modification of Herbelin's $\bar{\lambda}$-calculus [10].

The abstract syntax of λ^{Gtz} is given by:

$$\begin{array}{lrcl} \text{Terms} & t,u,v & ::= & x \,|\, \lambda x.t \,|\, tk \\ \text{Contexts} & k & ::= & \widehat{x}.t \,|\, u :: k \end{array}$$

A term is either a variable, an abstraction or a *cut tk*. A context is either a *selection* or a *context cons(tructor)*. Depending on the form of k, a cut may be an explicit substitution $t(\widehat{x}.v)$ or a multiary generalized application $t(u_1 :: \cdots u_m :: \widehat{x}.v)$, $m \geq 1$. In the last case, if $m = 1$, we get a generalized application $t(u :: \widehat{x}.v)$; if $v = x$, we get a multiary application $t[u_1, \cdots, u_m]$ ($\widehat{x}.x$ can be seen as the empty list of arguments). In $\lambda x.t$ and $\widehat{x}.t$, λx and $\widehat{x}$ bind the variable x in t. The scope of binders extends to the right as much as possible. Free variables are the variables not bound by abstraction or by selection operator and Barendregt's convention should be applied in both cases.

Reduction rules of λ^{Gtz} are the following:

$$\begin{array}{lrcl} (\beta) & (\lambda x.t)(u :: k) & \rightarrow & u(\widehat{x}.tk) \\ (\pi) & (tk)k' & \rightarrow & t(k@k') \\ (\sigma) & t\widehat{x}.v & \rightarrow & v[x := t] \\ (\mu) & \widehat{x}.xk & \rightarrow & k, \text{ if } x \notin k \end{array}$$

where $v[x := t]$ denotes meta-substitution defined as usual, and $k@k'$ is defined by

$$(u :: k)@k' = u :: (k@k') \qquad (\widehat{x}.t)@k' = \widehat{x}.tk'.$$

The rules β, π, and σ aim at eliminating all cuts but those of the trivial form $y(u_1 :: \cdots u_m :: \widehat{x}.v)$ (for some $m \geq 1$). The rule β generates a substitution but it is the rule σ that executes it, on the meta-level. The rule π simplifies the head of a cut (t is the *head* of tk). The rule μ has a structural character and it either performs a trivial substitution in the reduction $t(\widehat{x}.xk) \rightarrow tk$, or it minimizes the use of the generality feature in the reduction $t(u_1 \cdots u_m :: \widehat{x}.xk) \rightarrow t(u_1 \cdots u_m :: k)$.

This set of reduction rules has a logical motivation, as they correspond to cut-elimination steps (or, in the case of μ, to a certain trivial manipulation of sequent derivations). But it turns out that, even in the untyped case, these reduction rules are interesting on their own, being capable of simulating ordinary β-reduction, and therefore giving a decomposition of the atomic step of computation of the ordinary λ-calculus.

3 Type assignment systems

3.1 Simply typed λ^{Gtz}-calculus

The basic type assignment system for the λ^{Gtz}-calculus is the one with simple types, introduced by Espirito Santo in [7]. The set of types is defined as follows:

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

where X ranges over a denumerable set of type atoms.

A *type assignment* is an expression of the form $t : A$, where t is either a term or a context and A is a type. A context Γ is a set $\{x_1 : A_1, \ldots, x_n : A_n\}$ of type assignments with different term variables. $Dom\Gamma = \{x_1, \ldots, x_n\}$. A context extension $\Gamma, x : A$ denotes the set $\Gamma \cup \{x : A\}$, where $x \notin Dom\Gamma$.

There are two kinds of type assignment:

- $\Gamma \vdash t : A$ - a type assignment for terms;

- $\Gamma; B \vdash k : A$ - a type assignment for contexts.

Notice the special place between the symbols ; and $\vdash$, called the *stoup*, which contains a selected formula with which we continue computation.

The type assignment system $\lambda^{\mathsf{Gtz}} \rightarrow$ is given in Figure 1.

$$\frac{}{\Gamma, x : A \vdash x : A}\ (Ax)$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \rightarrow B}\ (\rightarrow_R) \qquad \frac{\Gamma \vdash t : A \quad \Gamma; B \vdash k : C}{\Gamma; A \rightarrow B \vdash t :: k : C}\ (\rightarrow_L)$$

$$\frac{\Gamma \vdash t : A \quad \Gamma; A \vdash k : B}{\Gamma \vdash tk : B}\ (Cut) \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma; A \vdash \widehat{x}.t : B}\ (Sel)$$

Figure 1: $\lambda^{\mathsf{Gtz}} \rightarrow$: simply typed λ^{Gtz}-calculus

$\lambda^{\mathsf{Gtz}} \rightarrow$ satisfies subject reduction, and the proof of this property shows which proof transformations are associated with each reduction rule. β corresponds to the key-step in cut-elimination, whereas σ and π correspond to right and left permutation of cuts, respectively. Rule μ undoes the sequence of two inference steps consisting of unselecting the stoup formula, without contraction, and, immediately after, selecting the same formula.

Espirito Santo proved strong normalisation for this system in [7], by translating it into a λ-calculus with "delayed" substitutions. But, as with simply typed λ-calculus, the basic type assignment system cannot characterise all strongly normalising terms. For example, the term $\lambda x.x(x :: \widehat{y}.y)$ (which corresponds to the term $\lambda x.xx$ in simply typed λ-calculus) does not have a type in $\lambda^{\mathsf{Gtz}} \rightarrow$, although it is a normal form.

3.2 Intersection types for the λ^{Gtz}-calculus

In order to characterise strong normalization in the λ^{Gtz}-calculus, the standard technique was to introduce intersection types to the system. But the construction of the appropriate type assignment system for λ^{Gtz} was not a straightforward process: it consisted of three attempts presented in the following subsections, followed by the additional successful system. In all of these systems, the set of types is defined as follows:

$$A, B\ ;::=\ X \mid A \rightarrow B \mid A \cap B$$

where X ranges over a denumerable set of type atoms.

1 First attempt: Intuitive system $\lambda^{\mathsf{Gtz}}\cap_{\mathsf{I}}$

Our first (and the most natural) attempt consisted of simply adding standard typing rules for the intersection operator to the existing Espirito Santo's basic type assignment system λ^{Gtz}, following the characteristic symmetry of the sequent calculus.

Pre-order $\leq$ over the set of types is defined as the smallest relation satisfying the following properties:

1. $A \leq A$
2. $A \cap B \leq A$, $A \cap B \leq B$
3. $(A \rightarrow B) \cap (A \rightarrow C) \leq A \rightarrow (B \cap C)$
4. $A \leq B$, $B \leq C \Rightarrow A \leq C$
5. $A \leq B$, $A \leq C \Rightarrow A \leq B \cap C$
6. $A' \leq A$, $B \leq B' \Rightarrow A \rightarrow B \leq A' \rightarrow B'$

Two types are equivalent, $A \sim B$, if and only if $A \leq B$ and $B \leq A$.

$$\frac{}{\Gamma, x : A \vdash x : A}\ (Ax)$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}\ (\to_R) \qquad \frac{\Gamma \vdash t : A \quad \Gamma; B \vdash k : C}{\Gamma; A \to B \vdash t :: k : C}\ (\to_L)$$

$$\frac{\Gamma \vdash t : A \quad \Gamma; A \vdash k : B}{\Gamma \vdash tk : B}\ (Cut) \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma; A \vdash \widehat{x}.t : B}\ (Sel)$$

$$\frac{\Gamma \vdash t : A \quad \Gamma \vdash t : B}{\Gamma \vdash t : A \cap B}\ (\cap_R) \qquad \frac{\Gamma, x : A_1 \vdash t : B}{\Gamma, x : A_1 \cap A_2 \vdash t : B}\ (\cap_L)$$

$$\frac{\Gamma \vdash t : A, \ A \leq B}{\Gamma \vdash t : B}\ (\leq_R)$$

Figure 2: First attempt: intuitive system $\lambda^{\mathsf{Gtz}}\cap_{\mathsf{I}}$

The type assignment system $\lambda^{\mathsf{Gtz}}\cap_{\mathsf{I}}$ is given in Figure 2.

Basis expansion and Bases intersection lemmas can be easily proved for the proposed system, where bases intersection is defined as usual. The following rules are admissible in $\lambda^{\mathsf{Gtz}}\cap_{\mathsf{I}}$.

Proposition 1 ($\leq$ RULES)

(i) If $\Gamma, x : A \vdash t : C$ *and* $B \leq A$*, then* $\Gamma, x : B \vdash t : C$.
If $\Gamma, x : A; C \vdash k : D$ *and* $B \leq A$*, then* $\Gamma, x : B; C \vdash k : D$.

(ii) If $\Gamma; C \vdash k : A$ *and* $A \leq B$*, then* $\Gamma; C \vdash k : B$.

Proposition 2 ($\cap$ RULES)

(i) If $\Gamma \vdash t : A_1 \cap A_2$*, then* $\Gamma \vdash t : A_i$*, for each* $i \in \{1, 2\}$.

(ii) If $\Gamma; A_1 \vdash \widehat{x}.t : B$*, then* $\Gamma; A_1 \cap A_2 \vdash \widehat{x}.t : B$.

(iii) If $\Gamma; C \vdash k : A$ *and* $\Gamma; C \vdash k : B$*, then* $\Gamma; C \vdash k : A \cap B$.

This system has two problems. The first one is that the second statement from Proposition 2 holds only for the selection, while it is not possible to prove a similar statement for the context of the form $k \equiv t :: k_1$ (since type changes are not allowed in the stoup in any of the typing rules). The second one is that in the presence of $(\cap_R)$ rule all terms could have intersection types. These problems make it impossible to formulate the Generation lemma which would enable us to "reverse" the rules of the type assignment system and which is usually necessary for proving the Subject reduction and Subject expansion properties. Hence, the main tool for further proofs is missing and forces us to search for a new intersection type assignment system for the λ^{Gtz}-calculus.

2 Second attempt: Restrictive system $\lambda^{\mathsf{Gtz}}\cap_{\mathsf{R}}$

Having realized that the above presented system is inappropriate, mainly because it allows too much due to the overly permissive typing rules, we turned to a more restrictive approach and designed a system inspired by the type assignment system for classical sequent $\lambda\mu\tilde{\mu}$-calculus proposed by Dougherty et al. in [6]. In this system pre-order $\leq$ on types is completely omitted, as well as RHS introduction of intersection, which turned out to be the problematic rule in the previous system $\lambda^{\mathsf{Gtz}}\cap_{\mathsf{I}}$ (indeed only LHS intersection introduction is important in the system, whereas RHS intersection introduction was only added for symmetry reasons). To regain the broken symmetry of the system, we replaced LHS intersection introduction with upgraded rules (Ax) and $(\to_L)$, in which intersection is implicitly introduced. This system assigns types to the same set of terms as the previous one, but it is more restrictive since the set of the types that can be assigned to a certain term is smaller. For example, in the previous system the type of the abstraction could be both $A \cap B$ and $A \to B$, whereas in this one it can only be $A \to B$. The system, denoted by $\lambda^{\mathsf{Gtz}}\cap_{\mathsf{R}}$, is given in Figure 3, where $\cap A_i$ abbreviates $\cap_{i=1}^{n} A_i$, for some $n \geq 1$.

$$\frac{}{\Gamma, x : \cap A_i \vdash x : A_i}\ (Ax)$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}\ (\to_R) \qquad \frac{\Gamma \vdash u : A_i\ \forall i \quad \Gamma; B \vdash k : C}{\Gamma; \cap A_i \to B \vdash u :: k : C}\ (\to_L)$$

$$\frac{\Gamma \vdash t : A \quad \Gamma; A \vdash k : B}{\Gamma \vdash tk : B}\ (Cut) \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma; A \vdash \widehat{x}.t : B}\ (Sel)$$

Figure 3: Second attempt: restrictive system $\lambda^{\mathsf{Gtz}}\cap_{\mathsf{R}}$

Basis expansion and Bases intersection lemmas hold for this system as well and the rule $(\cap_L)$ from the previous system is now admissible. Since there is exactly one rule for deriving each sequent the system is syntax-directed so it is trivial to formulate and prove the following Generation lemma.

Proposition 3 (GENERATION LEMMA)

(i) $\Gamma \vdash x : A$ *iff* $x : A \cap A_i \in \Gamma$, $i = 1, \ldots, n$ *for some* $n \geq 0$.

(ii) $\Gamma \vdash \lambda x.t : A$ *iff* $A \equiv \cap B_i \to C$ *and* $\Gamma, x : \cap B_i \vdash t : C$.

(iii) $\Gamma; A \vdash \widehat{x}.t : B$ *iff* $\Gamma, x : A \vdash t : B$.

(iv) $\Gamma \vdash tk : A$ *iff there exists a type* B *such that* $\Gamma \vdash t : B$ *and* $\Gamma; B \vdash k : A$.

(v) $\Gamma; T \vdash t :: k : C$ *iff* $T \equiv \cap A_i \to B$ *and* $\Gamma; B \vdash k : C$ *and for all* i, $\Gamma \vdash t : A_i$.

The basic properties we wanted to prove were Subject reduction and Subject expansion. We proved Substitution lemma, analogous to the one from λ-calculus and the following Append lemma.

Lemma 4 (APPEND LEMMA) *If* $\Gamma; C \vdash k : B$ *and* $\Gamma; B \vdash k' : A$, *then* $\Gamma; C \vdash k@k' : A$.

However, when trying to prove Subject reduction, we are stuck already at the first reduction rule β. Supposing that $\Gamma \vdash (\lambda x.t)(u :: k) : A$, we want to prove that $\Gamma \vdash u\widehat{x}.tk : A$. From $\Gamma \vdash (\lambda x.t)(u :: k) : A$ and using Generation lemma (iv) it follows that there exists a type B such that $\Gamma \vdash \lambda x.t : B$ and $\Gamma; B \vdash u :: k : A$. From the last sequent, using Generation lemma (v) it follows that $B = \cap C_i \to D$, $\Gamma; D \vdash k : A$ and for all i, $\Gamma \vdash u : C_i$. From $\Gamma \vdash \lambda x.t : B$, using Generation lemma (ii) it follows that $\Gamma, x : \cap C_i \vdash t : D$. Now we have to assign a type to term $u\widehat{x}.(tk)$:

$$\frac{\Gamma \vdash u : C_i \qquad \dfrac{\dfrac{\Gamma, x : \cap C_i \vdash t : D \quad \Gamma, x : \cap C_i; D \vdash k : A}{\Gamma, x : \cap C_i \vdash tk : A}\ (Cut)}{\Gamma; \cap C_i \vdash \widehat{x}.tk : A}\ (Sel)}{???}\ (Cut)$$

The last (Cut) rule is impossible to apply, since the types C_i and $\cap C_i$ do not match.

There are two solutions to this problem: we can either change the β reduction rule or we can again change the type system. The first solution can be achieved by replacing the β reduction rule with a larger computational step - $(\beta + \sigma)$ reduction rule as follows:

$$(\lambda x.t)(u :: k) \ \to_\beta \ t[x := u]k.$$

With this reduction rule, it is possible to prove Subject reduction for the rules β, σ and π without changing the type system. The μ reduction is of a different nature and for this reduction it is possible to prove the following proposition.

Proposition 5 *If* $\Gamma; \cap B_i \vdash \widehat{x}.xk : A$*, then* $\Gamma; B_i \vdash k : A$*, for some* i*.*

But such a modification implies losing the possibility to delay substitution and the call-by-value computational side of λ^{Gtz}. Also, Subject expansion property (needed for characterising strong normalisation) does not hold.

Hence, in order to obtain type assignment system which characterizes all strongly normalising terms, we had to change the type assignment system again. For more details about the system $\lambda^{\mathsf{Gtz}}\cap_{\mathsf{R}}$we refer the reader to [9, 11].

3 System $\lambda^{\mathsf{Gtz}}\cap$

The appropriate type assignment system, in which Subject reduction and Subject expansion at the root position hold for the original reductions of λ^{Gtz}, was introduced in Espirito Santo et al. [8]. In order to assign the same type to β-redex $(\lambda x.t)(u :: k)$ and its contractum $u\widehat{x}.(tk)$ (as required by Subject reduction) we need to implicitly introduce intersection in the (Cut) rule. The necessity for certain equivalencies among types showed up, so we returned $\leq$ relation. But $\leq$ relation is not explicitly introduced into typing rules, its only rôle is in defining equivalence, so that the equivalent types can be interchangeable in derivations. The important rôle belongs to the following equivalence: $\cap(A \to B_i) \sim A \to \cap B_i$. The type assignment system $\lambda^{\mathsf{Gtz}}\cap$ is given in Figure 4.

$$\frac{}{\Gamma, x : \cap A_i \vdash x : A_i}\ (Ax)$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}\ (\to_R) \qquad \frac{\Gamma \vdash u : A_i\ \ \forall i \quad \Gamma; B \vdash k : C}{\Gamma; \cap A_i \to B \vdash u :: k : C}\ (\to_L)$$

$$\frac{\Gamma \vdash t : A_i,\ \forall i \quad \Gamma; \cap A_i \vdash k : B}{\Gamma \vdash tk : B}\ (Cut) \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma; A \vdash \widehat{x}.t : B}\ (Sel)$$

Figure 4: $\lambda^{\mathsf{Gtz}}\cap$: type assignment system for λ^{Gtz}-calculus

The following rules are admissible in $\lambda^{\mathsf{Gtz}}\cap$.

Proposition 6
1. *If* $\Gamma, x : A_i \vdash t : C$ *then* $\Gamma, x : \cap A_i \vdash t : C$.
2. *If* $\Gamma, x : A_i; D \vdash k : C$ *then* $\Gamma, x : \cap A_i; D \vdash k : C$.

Basis expansion and Bases intersection lemmas hold for $\lambda^{\mathsf{Gtz}}\cap$. Generation lemma differs from the previous one only in the following.

Proposition 7 (GENERATION LEMMA)
(iv) $\Gamma \vdash tk : A$ *iff there exists a type* $\cap B_i$, $i = 1, \ldots, n$ *such that for all* i $\Gamma \vdash t : B_i$ *and* $\Gamma; \cap B_i \vdash k : A$.

With this system we finally succeeded in characterising strong normalisation in λ^{Gtz}, i.e., the terms are strongly normalising if and only if they can be assigned a type in $\lambda^{\mathsf{Gtz}}\cap$ (for more details see [8]).

Example: In λ-calculus with intersection types, the term $\lambda x.xx$ has the type $(A \cap (A \to B)) \to B$. The corresponding term in λ^{Gtz}-calculus is $\lambda x.x(x :: \widehat{y}.y)$. Although being a normal form this term is not typeable in the simply typed λ^{Gtz}-calculus. It is typeable in $\lambda^{\mathsf{Gtz}}\cap$ in the following way:

$$\dfrac{\dfrac{\dfrac{}{x:A\cap(A\to B)\vdash x:A\to B}\,(Ax) \qquad \dfrac{\dfrac{}{x:A\cap(A\to B)\vdash x:A}\,(Ax) \quad \dfrac{\dfrac{}{x:A\cap(A\to B),y:B\vdash y:B}\,(Ax)}{x:A\cap(A\to B);B\vdash \widehat{y}.y:B}\,(Sel)}{x:A\cap(A\to B);A\to B\vdash (x::\widehat{y}.y):B}\,(\to_L)}{x:A\cap(A\to B)\vdash x(x::\widehat{y}.y):B}\,(Cut)}{\vdash \lambda x.x(x::\widehat{y}.y):(A\cap(A\to B))\to B}\,(\to_R).$$

4 Systems $\lambda^{\mathsf{Gtz}}\cap_\circ$ and $\lambda^{\mathsf{Gtz}}\cap_\sharp$

In $\lambda^{\mathsf{Gtz}}\cap$, the whole business of equivalence (and $\leq$) is left to the meta-level. This defect is reduced to a minimum in the new systems $\lambda^{\mathsf{Gtz}}\cap_\circ$ and $\lambda^{\mathsf{Gtz}}\cap_\sharp$, which we now present. In these systems we distinguish two kinds of types: *proper types* and *strict types*. $\lambda^{\mathsf{Gtz}}\cap_\circ$ (resp. $\lambda^{\mathsf{Gtz}}\cap_\sharp$) is a system for assigning proper (resp. strict) types.

Proper and strict types are defined as follows:

$$\begin{array}{lccl} \textit{Proper Types} & S,T,U & ::= & \cap_{i=1}^{n} a_i \ (n\geq 1) \\[1em] \textit{Strict Types} & a,b,c & ::= & p \mid S\to b \end{array}$$

If we impose $n=1$, then $\mathit{ProperTypes}=\mathit{StrictTypes}=\mathit{SimpleTypes}$. By allowing $n\geq 1$ one has:

$$\mathit{SimpleTypes}\subset\mathit{StrictTypes}\subset\mathit{ProperTypes}\subset\mathit{Types}\ \ .$$

At the level of proper types, we work modulo commutativity and idempotency of $\cap$. So we can think of S as a non-empty, finite set of strict types, and use the set-theoretical notations $a\in S$, $S\subseteq T$, and $S\cup T$. For instance, if $S=a\cap b$ and $T=a$, then $a\in T$, $a\in S$, $T\subseteq S$, and $S\cup T=S=a\cap b\cap a$. Notice that it is natural to regard $a\cap b\cap a$ as $S\cap T$!

Definition 8 Let S, T be proper types.

i) $S\cap T:=S\cup T$.

ii) $S\to T:=\cap_{b\in T}(S\to b)$.

Definition 9 A function $(\_)^\circ:\mathit{Types}\to\mathit{ProperTypes}$ is defined by $p^\circ=p$, $(A\to B)^\circ=A^\circ\to B^\circ$, and $(A\cap B)^\circ=A^\circ\cap B^\circ$.

Lemma 10 $A\sim A^\circ$.

Proof: By induction on A. Immediate by IH and the fact that $\sim$ is a congruence. ■

In $\lambda^{\mathsf{Gtz}}\cap_\circ$ bases are sets of declarations $x:S$ where all term variables are different. Sequents have two forms: $\Gamma\vdash t:T$ and $\Gamma;S\vdash k:T$. Typing rules are given in Figure 5.

Bases in $\lambda^{\mathsf{Gtz}}\cap_\sharp$ are as in $\lambda^{\mathsf{Gtz}}\cap_\circ$. Sequents in $\lambda^{\mathsf{Gtz}}\cap_\sharp$ have the forms $\Gamma\vdash t:b$ and $\Gamma;S\vdash k:b$. Typing rules are given in Figure 6.

Both in $\lambda^{\mathsf{Gtz}}\cap_\circ$ and $\lambda^{\mathsf{Gtz}}\cap_\sharp$, we have no $\leq$ and no equivalence (except that, at the level of proper types, we work modulo commutativity and idempotency of $\cap$).

Proposition 11

(i) If $\lambda^{\mathsf{Gtz}}\cap_\sharp$ derives $\Gamma\vdash t:a$, then $\lambda^{\mathsf{Gtz}}\cap_\circ$ derives $\Gamma\vdash t:a$.

(ii) If $\lambda^{\mathsf{Gtz}}\cap_\circ$ derives $\Gamma\vdash t:S$, then $\lambda^{\mathsf{Gtz}}\cap$ derives $\Gamma\vdash t:S$.

$$\frac{S \supseteq T}{\Gamma, x:S \vdash x:T}\ (Ax)$$

$$\frac{\Gamma, x:S \vdash t:T}{\Gamma \vdash \lambda x.t : S \to T}\ (\to_R) \qquad \frac{\Gamma \vdash u:a,\ \forall a \in S \qquad \Gamma;T \vdash k:U}{\Gamma; S \to T \vdash u::k:U}\ (\to_L)$$

$$\frac{\Gamma \vdash t:a,\ \forall a \in S \qquad \Gamma;S \vdash k:U}{\Gamma \vdash tk:U}\ (Cut) \qquad \frac{\Gamma, x:S \vdash v:U}{\Gamma; S \vdash \widehat{x}.v:U}\ (Sel)$$

Figure 5: $\lambda^{\mathsf{Gtz}}\cap_{\circ}$: proper type assignment system for λ^{Gtz}-calculus

$$\frac{b \in S}{\Gamma, x:S \vdash x:b}\ (Ax)$$

$$\frac{\Gamma, x:S \vdash t:b}{\Gamma \vdash \lambda x.t : S \to b}\ (\to_R) \qquad \frac{\Gamma \vdash u:a,\ \forall a \in S \qquad \Gamma;T \vdash k:b}{\Gamma; S \to T \vdash u::k:b}\ (\to_L)$$

$$\frac{\Gamma \vdash t:a,\ \forall a \in S \qquad \Gamma;S \vdash k:b}{\Gamma \vdash tk:b}\ (Cut) \qquad \frac{\Gamma, x:S \vdash v:b}{\Gamma; S \vdash \widehat{x}.v:b}\ (Sel)$$

Figure 6: $\lambda^{\mathsf{Gtz}}\cap_{\sharp}$: strict type assignment system for λ^{Gtz}-calculus

Proof: (i) A typing derivation in $\lambda^{\mathsf{Gtz}}\cap_{\sharp}$ is a typing derivation in $\lambda^{\mathsf{Gtz}}\cap_{\circ}$. (ii) A typing derivation in $\lambda^{\mathsf{Gtz}}\cap_{\circ}$ is a typing derivation in $\lambda^{\mathsf{Gtz}}\cap$. ■

Proposition 12 If $\lambda^{\mathsf{Gtz}}\cap_{\circ}$ derives $\Gamma \vdash t:T$, then, for all $b \in T$, $\lambda^{\mathsf{Gtz}}\cap_{\sharp}$ derives $\Gamma \vdash t:b$;

Proof: We also prove that, if $\lambda^{\mathsf{Gtz}}\cap_{\circ}$ derives $\Gamma;S \vdash k:T$, then, for all $b \in T$, $\lambda^{\mathsf{Gtz}}\cap_{\sharp}$ derives $\Gamma;S \vdash k:b$. The proof is by simultaneous induction on $\Gamma \vdash t:T$ and $\Gamma;S \vdash k:T$. Cases according to the last typing rule used. All cases are straightforward. The only case slightly interesting is $\to_R$, which we prove. By IH we know that, for each $b \in T$, $\lambda^{\mathsf{Gtz}}\cap_{\sharp}$ derives $\Gamma, x:S \vdash t:b$. So, for each $b \in T$, $\lambda^{\mathsf{Gtz}}\cap_{\sharp}$ derives $\Gamma \vdash \lambda x.t : S \to b$. Since $S \to T = \cap_{b \in T}(S \to b)$, we actually have that, for each $c \in S \to T$, $\lambda^{\mathsf{Gtz}}\cap_{\sharp}$ derives $\Gamma \vdash \lambda x.t : c$. ■

The following rules are admissible in $\lambda^{\mathsf{Gtz}}\cap_{\circ}$.

Proposition 13
(i) If $\Gamma \vdash t:T$ and $a \in T$, then $\Gamma \vdash t:a$.
(ii) If $\Gamma;S \vdash k:T$ and $a \in T$, then $\Gamma;S \vdash k:a$.

Proof: Follows from the statements proved in the previous proposition, together with the fact that derivations in $\lambda^{\mathsf{Gtz}}\cap_{\sharp}$ are derivations in $\lambda^{\mathsf{Gtz}}\cap_{\circ}$. ■

We define $\Gamma^{\circ} = \{(x:A^{\circ}) : (x:A) \in \Gamma\}$. We now see that, if $\lambda^{\mathsf{Gtz}}\cap$ derives $\Gamma \vdash t:A$, then $\lambda^{\mathsf{Gtz}}\cap_{\circ}$ assigns an equivalent type to t, in base Γ°.

Proposition 14 If $\lambda^{\mathsf{Gtz}}\cap$ derives $\Gamma \vdash t:A$, then $\lambda^{\mathsf{Gtz}}\cap_{\circ}$ derives $\Gamma^{\circ} \vdash t:A^{\circ}$.

Proof: We also prove that, if $\lambda^{\mathsf{Gtz}}\cap$ derives $\Gamma;B \vdash k:A$, then $\lambda^{\mathsf{Gtz}}\cap_{\circ}$ derives $\Gamma^{\circ};B^{\circ} \vdash k:A^{\circ}$. The proof is by simultaneous induction on $\Gamma \vdash t:A$ and $\Gamma;B \vdash k:A$. Cases according to the last typing rule used.

Case (Ax). We want to prove that $\lambda^{\mathsf{Gtz}}\cap_{\circ}$ derives $\Gamma^{\circ}, x:\cap_{i=1}^{n} A_i^{\circ} \vdash x:A_j^{\circ}$, with $j \in \{1,\cdots,n\}$. This sequent is derived in $\lambda^{\mathsf{Gtz}}\cap_\circ$ with an application of (Ax), because $\cap_{i=1}^n A_i^\circ \supseteq A_j^\circ$.

Cases $(\rightarrow_R)$ and (Sel). Straightforward.

Case $(\rightarrow_L)$. We are given by IHs $\Gamma^\circ \vdash u : A_i^\circ$ (for each $i \in \{1, \cdots, n\}$) and $\Gamma^\circ; B^\circ \vdash k : C^\circ$. We have to derive the sequent $\Gamma^\circ; (\cap_{i=1}^n A_i \rightarrow B)^\circ \vdash u :: k : C^\circ$ in $\lambda^{\mathsf{Gtz}}\cap_\circ$. Let $S = \cap_{i=1}^n A_i^\circ$. We now claim that, for each $a \in S$, $\Gamma^\circ \vdash u : a$. Let $a \in S$. Then there is $i \in \{1, \cdots, n\}$ such that $a \in A_i^\circ$ (because in fact $S = \cup_{i=1}^n A_i^\circ$). From $\Gamma^\circ \vdash u : A_i^\circ$ and $a \in A_i^\circ$ and Proposition 13 we conclude $\Gamma^\circ \vdash u : a$. The claim is proved. From the claim and $\Gamma^\circ; B^\circ \vdash k : C^\circ$ we obtain, with one application of $\rightarrow_L$, the sequent $\Gamma^\circ; S \rightarrow B^\circ \vdash u :: k : C^\circ$. This is what we want, because $(\cap_{i=1}^n A_i \rightarrow B)^\circ = (\cap_{i=1}^n A_i^\circ \rightarrow B^\circ) = S \rightarrow B^\circ$.

Case (Cut). Similar to case $\rightarrow_L$. ■

The main result is given in the following theorem.

Theorem 15 *Let t be a λ^{Gtz}-term. The following are equivalent:*

(i) t is typeable in $\lambda^{\mathsf{Gtz}}\cap$;

(ii) t is typeable in $\lambda^{\mathsf{Gtz}}\cap_\circ$;

(iii) t is typeable in $\lambda^{\mathsf{Gtz}}\cap_\sharp$.

Proof: Immediate from Propositions 11, 12, and 14. ■

Hence we get two alternative characterisations of the strongly normalising terms of λ^{Gtz}.

4 Conclusion

In this work we described our quest for the intersection type assignment system $\lambda^{\mathsf{Gtz}}\cap$ and offered two new, equivalent systems $\lambda^{\mathsf{Gtz}}\cap_\circ$ and $\lambda^{\mathsf{Gtz}}\cap_\sharp$. All the systems $\lambda^{\mathsf{Gtz}}\cap$, $\lambda^{\mathsf{Gtz}}\cap_\circ$, and $\lambda^{\mathsf{Gtz}}\cap_\sharp$ successfully characterise strongly normalising terms of the λ^{Gtz}-calculus.

The characterisation of weak normalisation in the λ^{Gtz}-calculus is still an open problem that might be the first direction for future research. This might lead us to a design of a more "natural" type assignment system which characterises strong normalisation (along the lines of the first type system $\lambda^{\mathsf{Gtz}}\cap_{\mathsf{I}}$). Introduction of some additional operators, such as the operators of explicit contraction and explicit weakening, might broaden the expressiveness of the system.

In [8] a characterisation is given of those λJ-terms [12] or λx-terms [17] which are strongly normalising *as sequent terms*, that is, as λ^{Gtz}-terms reducing inside λ^{Gtz}. Such characterisation is given in terms of typeability in cetain subsystems of $\lambda^{\mathsf{Gtz}}\cap$. An interesting exercise would be to obtain an alternative characterisation in terms of subsystems of $\lambda^{\mathsf{Gtz}}\cap_\circ$ or $\lambda^{\mathsf{Gtz}}\cap_\sharp$.

References

[1] H. P. Barendregt, M. Coppo, and M. Dezani-Ciancaglini. A filter lambda model and the completeness of type assignment. *J. of Symbolic Logic*, 48(4):931–940 (1984), 1983.

[2] M. Coppo and M. Dezani-Ciancaglini. A new type-assignment for lambda terms. *Archiv für Mathematische Logik*, 19:139–156, 1978.

[3] M. Coppo and M. Dezani-Ciancaglini. An extension of the basic functionality theory for the λ-calculus. *Notre Dame J. of Formal Logic*, 21(4):685–693, 1980.

[4] M. Coppo, M. Dezani-Ciancaglini, and B. Venneri. Principal type schemes and λ-calculus semantics. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 535–560. Academic Press, London, 1980.

[5] D. Dougherty, S. Ghilezan, and P. Lescanne. Intersection and union types in the $\overline{\lambda}\mu\widetilde{\mu}$-calculus. In M. Coppo and F. Damiani, editors, *Intersection types and related systems 2004*, volume 136 of *ENTCS*, pages 153–172. Elsevier, 2005.

[6] D. Dougherty, S. Ghilezan, and P. Lescanne. Characterizing strong normalization in the Curien-Herbelin symmetric lambda calculus: extending the Coppo-Dezani heritage. In S. Berardi and U. de' Liquoro, editors, *Theoretical Computer Science*, volume Festschrift Coppo, Dezani, Ronchi. To appear.

[7] J. Espírito Santo. Completing Herbelin's programme. In S. R. D. Rocca, editor, *Proceedings of TLCA'07*, volume 4583 of *LNCS*, pages 118–132. Springer-Verlag, 2007.

[8] J. Espírito Santo, S. Ghilezan, and J. Ivetić. Characterising strongly normalising intuitionistic sequent terms. In *International Workshop TYPES'07 (Selected Papers)*, LNCS. Springer-Verlag, 2008.

[9] S. Ghilezan and J. Ivetić. Intersection types for λ^{gtz} calculus. *Publications de l'Institute Mathematique, Serbian Academy of Sciences and Arts*, 82(96):159–164, 2007.

[10] H. Herbelin. A lambda calculus structure isomorphic to Gentzen-style sequent calculus structure. In *Computer Science Logic, CSL 1994*, volume 933 of *LNCS*, pages 61–75. Springer-Verlag, 1995.

[11] J. Ivetić. Formal calculi for intuitionistic logic. Master's thesis, University of Novi Sad, 2008.

[12] F. Joachimski and R. Matthes. Standardization and confluence for ΛJ. In *Proceedings of RTA 2000*, volume 1833 of *LNCS*, pages 141–155. Springer, 2000.

[13] K. Kikuchi. Simple proofs of characterizing strong normalisation for explicit substitution calculi. In F. Baader, editor, *Proceedings of RTA'07*, LNCS, pages 257–272. Springer-Verlag, 2007.

[14] S. Lengrand, P. Lescanne, D. Dougherty, M. Dezani-Ciancaglini, and S. van Bakel. Intersection types for explicit substitutions. *Inf. Comput.*, 189(1):17–42, 2004.

[15] R. Matthes. Characterizing strongly normalizing terms of a calculus with generalized applications via intersection types. In *ICALP Satellite Workshops*, pages 339–354, 2000.

[16] G. Pottinger. A type assignment for the strongly normalizable λ-terms. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 561–577. Academic Press, London, 1980.

[17] K. Rose. Explicit substitutions: Tutorial & survey. Technical Report LS-96-3, BRICS, 1996.

[18] P. Sallé. Une extension de la théorie des types en lambda-calcul. In G. Ausiello and C. Böhm, editors, *Fifth International Conference on Automata, Languages and Programming*, volume 62 of *LNCS*, pages 398–410. Springer-Verlag, 1978.