Intersection Types for the Resource Control Lambda Calculi

Intersection types for the resource control lambda calculi

S. Ghilezan[1;?], J. Ivetić[1;?], P. Lescanne[2], S. Likavec[3;?]

[1] University of Novi Sad, Faculty of Technical Sciences, Serbia
gsilvia@uns.ac.rs jelenaivetic@uns.ac.rs
[2] University of Lyon, École Normal Supérieure de Lyon, France
pierre.lescanne@ens-lyon.fr
[3] Dipartimento di Informatica, Università di Torino, Italy
likavec@di.unito.it

Abstract. We propose intersection type assignment systems for two resource control term calculi: the lambda calculus and the sequent lambda calculus with explicit operators for weakening and contraction. These resource control calculi, λ_{r} and $\lambda^{\mathsf{Gtz}}_{\mathsf{r}}$, respectively, capture the computational content of intuitionistic natural deduction and intuitionistic sequent logic with explicit structural rules. Our main contribution is the characterisation of strong normalisation of reductions in both calculi. We first prove that typability implies strong normalisation in λ_{r} by adapting the reducibility method. Then we prove that typability implies strong normalisation in $\lambda^{\mathsf{Gtz}}_{\mathsf{r}}$ by using a combination of well-orders and a suitable embedding of $\lambda^{\mathsf{Gtz}}_{\mathsf{r}}$-terms into λ_{r}-terms which preserves types and enables the simulation of all its reductions by the operational semantics of the λ_{r}-calculus. Finally, we prove that strong normalisation implies typability in both systems using head subject expansion.

Introduction

It is well known that simply typed λ-calculus captures the computational content of intuitionistic natural deduction through Curry-Howard correspondence [21]. This connection between logic and computation can be extended to other calculi and logical systems [19]: Parigot's $\lambda\mu$-calculus [28] corresponds to classical natural deduction, whereas in the realm of sequent calculus, Herbelin's $\bar{\lambda}$-calculus [20], Espírito Santo's λ^{Gtz}-calculus [14], Barbanera and Berardi's symmetric calculus [3] and Curien and Herbelin's $\bar{\lambda}\mu\tilde{\mu}$-calculus [11] correspond to its intuitionistic and classical versions. Extending λ-calculus (λ^{Gtz}-calculus) with explicit operators for weakening and contraction brings the same correspondence to intuitionistic natural deduction (intuitionistic sequent calculus) with explicit structural rules, as investigated in [22, 23, 18].

Among many extensions of the simple type discipline is the one with intersection types, originally introduced in [9, 10, 29, 33] in order to characterise termination properties of term calculi [36, 16, 17]. The extension of Curry-Howard correspondence to other formalisms brought the need for intersection types into many different settings [13, 24–26].

? Partially supported by the Ministry of Education and Science of Serbia, projects III44006 and ON174026

Our work is inspired by Kesner and Lengrand's work on resource operators for λ-calculus [22]. Their linear λlxr calculus introduces operators for substitution, erasure and duplication, preserving at the same time strong normalisation, confluence and subject reduction property of its predecessor λx [8].

Explicit control of erasure and duplication leads to decomposing of reduction steps into more atomic steps, thus revealing the details of computation which are usually left implicit. Since erasing and duplicating of (sub)terms essentially changes the structure of a program, it is important to see how this mechanism really works and to be able to control this part of computation. We choose a direct approach to term calculi, namely lambda calculus and sequent lambda calculus, rather than taking a more common path through linear logic [1, 7]. In practice, for instance in the description of compilers by rules with binders [31, 32], the implementation of substitutions of linear variables by inlining is simple and efficient when substitution of duplicated variables requires the cumbersome and time consuming mechanism of pointers and it is therefore important to tightly control duplication. On the other hand, precise control of erasing does not require a garbage collector and prevents memory leaking.

We introduce the intersection types into λ_{r} and $\lambda_{\mathsf{r}}^{\mathsf{Gtz}}$, λ-calculus and λ^{Gtz}-calculus with explicit rules for weakening and contraction. To the best of our knowledge, this is a first treatment of intersection types in the presence of resource control operators. Our intersection type assignment systems $\lambda_{\mathsf{r}}\cap$ and $\lambda_{\mathsf{r}}^{\mathsf{Gtz}}\cap$ integrate intersection into logical rules, thus preserving syntax-directedness of the system. We assign restricted form of intersection types, namely strict types, therefore minimizing the need for pre-order on types. Using these intersection type assignment systems we prove that terms in both calculi enjoy the strong normalisation property if and only if they are typable.

We first prove that typability implies strong normalisation in λ_{r}-calculus by adapting the reducibility method for explicit resource control operators. Then we prove strong normalisation for $\lambda_{\mathsf{r}}^{\mathsf{Gtz}}$ by using a combination of well-orders and a suitable embedding of $\lambda_{\mathsf{r}}^{\mathsf{Gtz}}$-terms into λ_{r}-terms which preserves types and enables the simulation of all its reductions by the operational semantics of the λ_{r}-calculus. Finally, we prove that strong normalisation implies typability in both systems using head subject expansion.

The paper is organised as follows. In Section 1 we extend the λ-calculus and λ^{Gtz}-calculus with explicit operators for weakening and contraction obtaining λ_{r}-calculus and $\lambda_{\mathsf{r}}^{\mathsf{Gtz}}$-calculus, respectively. Intersection type assignment systems with strict types are introduced to these calculi in Section 2. In Section 3 we first prove that typability implies strong normalization in λ_{r}-calculus by adapting the reducibility method. Then we prove that typability implies strong normalization in $\lambda_{\mathsf{r}}^{\mathsf{Gtz}}$-calculus by using a combination of well-orders and a suitable embedding of $\lambda_{\mathsf{r}}^{\mathsf{Gtz}}$-terms into λ_{r}-terms which preserves types and enables the simulation of all its reductions by the operational semantics of the λ_{r}-calculus. Section 4 gives a proof of strong normalization of typable terms for both calculi using head subject expansion. We conclude in Section 5.

1 Untyped resource control calculi

1.1 Resource control lambda calculus λ_{r}

The *resource control* lambda calculus, λ_{r}, is an extension of the λ-calculus with explicit operators for weakening and contraction. It corresponds to the λcw-calculus of Kesner and Renaud, proposed in [23] as a vertex of "the prismoid of resources".

The *pre-terms* of λ_{r}-calculus are given by the following abstract syntax:

$$\text{Pre-terms} \quad f ::= x \,|\, \lambda x{:}f \,|\, ff \,|\, x \odot f \,|\, x <^{x_1}_{x_2} f$$

where x ranges over a denumerable set of term variables. $\lambda x{:}f$ is an *abstraction*, ff is an *application*, $x \odot f$ is a *weakening* and $x <^{x_1}_{x_2} f$ is a *contraction*. The contraction operator is assumed to be insensitive to order of the arguments x_1 and x_2 i.e. $x <^{x_1}_{x_2} f = x <^{x_2}_{x_1} f$.

The set of free variables of a pre-term f, denoted by $Fv(f)$, is defined as follows:

$$Fv(x) = x; \quad Fv(\lambda x{:}f) = Fv(f) \setminus \{x\}; \quad Fv(fg) = Fv(f) \cup Fv(g);$$
$$Fv(x \odot f) = \{x\} \cup Fv(f); \quad Fv(x <^{x_1}_{x_2} f) = \{x\} \cup Fv(f) \setminus \{x_1; x_2\}:$$

In $x <^{x_1}_{x_2} f$, the contraction binds the variables x_1 and x_2 and a free variable x is introduced. The operator $x \odot f$ also introduces a free variable x. In order to avoid parentheses, we let the scope of all binders extend to the right as much as possible.

The set of λ_{r}*-terms*, denoted by Λ_{r} and ranged over by $M; N; P, M_1; :::$. is a subset of the set of pre-terms, defined in Figure 1.

$$\frac{}{x \in \Lambda_{\mathsf{r}}} \qquad \frac{f \in \Lambda_{\mathsf{r}} \quad x \in Fv(f)}{\lambda x{:}f \in \Lambda_{\mathsf{r}}}$$

$$\frac{f \in \Lambda_{\mathsf{r}} \quad g \in \Lambda_{\mathsf{r}} \quad Fv(f) \cap Fv(g) = \emptyset}{fg \in \Lambda_{\mathsf{r}}}$$

$$\frac{f \in \Lambda_{\mathsf{r}} \quad x \Subset Fv(f)}{x \odot f \in \Lambda_{\mathsf{r}}} \qquad \frac{f \in \Lambda_{\mathsf{r}} \quad x_1; x_2 \in Fv(f) \quad x \Subset Fv(f)}{x <^{x_1}_{x_2} f \in \Lambda_{\mathsf{r}}}$$

Fig. 1. $\Lambda_{\mathsf{r}} : \lambda_{\mathsf{r}}$-terms

Informally, we say that a term is a pre-term in which in every subterm every free variable occurs exactly once, and every binder binds (exactly one occurrence of) a free variable. This notion corresponds to the notion of linear terms in [22]. In that sense, only linear expressions are in the focus of our investigation. This assumption is not a restriction, since every non linear λ-term has its linear correspondent, as illustrated by the following example.

Example 1. Pre-terms $\lambda x{:}y$ and $\lambda x{:}xx$ are not λ_{r}-terms, on the other hand pre-terms $\lambda x{:}(x \odot y)$ and $\lambda x{:}x <^{x_1}_{x_2} (x_1 x_2)$ are λ_{r}-terms.

In the sequel, we use the notation $X \odot M$ for $x_1 \odot ::: x_n \odot M$ and $X <^{Y}_{Z} M$ for $x_1 <^{y_1}_{z_1} ::: x_n <^{y_n}_{z_n} M$, where X, Y and Z are lists of the size n, consisting of all distinct variables $x_1; :::; x_n; y_1; :::; y_n; z_1; :::; z_n$.

$$
\begin{array}{llll}
(\beta) & (\lambda x{:}M)N \to M[N{=}x] & & \\
(\gamma_1) & x <^{x_1}_{x_2} (\lambda y{:}M) \to \lambda y{:}x <^{x_1}_{x_2} M & (\omega_1) & \lambda x{:}(y \odot M) \to y \odot (\lambda x{:}M);\ x \neq y \\
(\gamma_2) & x <^{x_1}_{x_2} (MN) \to (x <^{x_1}_{x_2} M)N;\ \text{if } x_1;x_2 \in Fv(M) & (\omega_2) & (x \odot M)N \to x \odot (MN) \\
(\gamma_3) & x <^{x_1}_{x_2} (MN) \to M(x <^{x_1}_{x_2} N);\ \text{if } x_1;x_2 \in Fv(N) & (\omega_3) & M(x \odot N) \to x \odot (MN) \\
(\gamma\omega_1) & x <^{x_1}_{x_2} (y \odot M) \to y \odot (x <^{x_1}_{x_2} M);\ y \neq x_1;x_2 & (\gamma\omega_2) & x <^{x_1}_{x_2} (x_1 \odot M) \to M[x{=}x_2]
\end{array}
$$

Fig. 2. Reduction rules of λ_{r} -calculus

The reduction rules of λ_{r} -calculus are presented in Figure 2.

The inductive definition of the meta operator $[\,=\,]$, representing the substitution of free variables, is given in Figure 3. In this definition, the terms N_1 and N_2 are obtained from N by renaming of all the free variables in N by fresh variables.

$$
\begin{array}{ll}
x[N{=}x] \triangleq N & (y \odot M)[N{=}x] \triangleq y \odot M[N{=}x];\ x \neq y \\
(\lambda y{:}M)[N{=}x] \triangleq \lambda y{:}M[N{=}x];\ x \neq y & (x \odot M)[N{=}x] \triangleq Fv(N) \odot M \\
(MP)[N{=}x] \triangleq M[N{=}x]P,\ x \in Fv(M) & (y <^{y_1}_{y_2} M)[N{=}x] \triangleq y <^{y_1}_{y_2} M[N{=}x];\ x \neq y \\
(MP)[N{=}x] \triangleq MP[N{=}x];\ x \in Fv(P) & (x <^{x_1}_{x_2} M)[N{=}x] \triangleq Fv(N) <^{Fv(N_1)}_{Fv(N_2)} M[N_1{=}x_1][N_2{=}x_2]
\end{array}
$$

Fig. 3. Substitution in λ_{r} -calculus

In the λ_{r} , one works modulo equivalencies given in Figure 4.

$$
\begin{array}{ll}
x \odot (y \odot M) \equiv y \odot (x \odot M) & x <^{x_1}_{x_2} M \equiv x <^{x_2}_{x_1} M \\
x <^{y}_{z} (y <^{u}_{v} M) \equiv x <^{y}_{u} (y <^{z}_{v} M) & x <^{x_1}_{x_2} (y <^{y_1}_{y_2} M) \equiv y <^{y_1}_{y_2} (x <^{x_1}_{x_2} M);\ x \neq y_1;y_2;\ y \neq x_1;x_2 \\
M[(y \odot N){=}x] \equiv y \odot M[N{=}x] & M[(y <^{y_1}_{y_2} N){=}x] \equiv y <^{y_1}_{y_2} M[N{=}x];\ y_1;y_2 \in Fv(N)
\end{array}
$$

Fig. 4. Equivalences in λ_{r} -calculus

1.2 Resource control sequent lambda calculus $\lambda^{\mathsf{Gtz}}_{\mathsf{r}}$

The *resource control lambda Gentzen* calculus $\lambda^{\mathsf{Gtz}}_{\mathsf{r}}$ is derived from the λ^{Gtz}-calculus (more precisely its confluent sub-calculus $\lambda^{\mathsf{Gtz}}_{V}$) by adding the explicit operators for weakening and contraction. It is proposed in [18]. The abstract syntax of $\lambda^{\mathsf{Gtz}}_{\mathsf{r}}$ pre-expressions is the following:

$$
\begin{array}{ll}
\text{Pre-values} & F ::= x \,|\, \lambda x{:}f \,|\, x \odot f \,|\, x <^{x_1}_{x_2} f \\
\text{Pre-terms} & f ::= F \,|\, fc \\
\text{Pre-contexts} & c ::= \widehat{x}{:}f \,|\, f :: c \,|\, x \odot c \,|\, x <^{x_1}_{x_2} c
\end{array}
$$

where x ranges over a denumerable set of term variables.

A *pre-value* can be a variable, an abstraction, a weakening or a contraction; a *pre-term* is either a value or a cut (an application). A *pre-context* is one of the following: a selection, a context constructor (usually called cons), a weakening on pre-context or a contraction on a pre-context. Pre-terms and pre-contexts are together referred to as the *pre-expressions* and will be ranged over by E. Pre-contexts $x \odot c$ and $x <^{x_1}_{x_2} c$ behave exactly like corresponding pre-terms $x \odot f$ and $x <^{x_1}_{x_2} f$ in the untyped calculus, so they will not be treated separately. The set of free variables of a pre-expression is defined analogously to the free variables in λ_{r} -calculus with the following additions:

$$Fv(fc) = Fv(f) \cup Fv(c); \quad Fv(\widehat{x}{:}f) = Fv(f) \setminus \{x\}; \quad Fv(f :: c) = Fv(f) \cup Fv(c){:}$$

Like in the case of λ_{r} -calculus, the set of $\lambda^{\mathsf{Gtz}}_{\mathsf{r}}$-expressions (namely values, terms and contexts), denoted by $\Lambda^{\mathsf{Gtz}}_{\mathsf{r}} \cup \Lambda^{\mathsf{Gtz}}_{\mathsf{r};C}$, is a subset of the set of pre-expressions, defined as in Figure 1 plus:

$$\frac{f \in \Lambda^{\mathsf{Gtz}}_{\mathsf{r}} \quad x \in Fv(f)}{\widehat{x}{:}f \in \Lambda^{\mathsf{Gtz}}_{\mathsf{r};C}} \qquad \frac{f \in \Lambda^{\mathsf{Gtz}}_{\mathsf{r}} \quad c \in \Lambda^{\mathsf{Gtz}}_{\mathsf{r};C} \quad Fv(f) \cap Fv(c) = \emptyset}{f :: c \in \Lambda^{\mathsf{Gtz}}_{\mathsf{r};C}}$$

Values are denoted by T; terms by $t; u; v{:}{:}{:}$, contexts by $k; k'; {:}{:}{:}$ and expressions by $e; e'$.

The computation over the set of $\lambda^{\mathsf{Gtz}}_{\mathsf{r}}$-expressions reflects the cut-elimination process. Four groups of reductions in $\lambda^{\mathsf{Gtz}}_{\mathsf{r}}$-calculus are given in Figure 5.

(β)	$(\lambda x{:}t)(u :: k) \to u(\widehat{x}{:}tk)$	(σ)	$T(\widehat{x}{:}v) \to v[T{=}x]$
(π)	$(tk)k' \to t(k@k')$	(μ)	$\widehat{x}{:}xk \to k$
(γ_1)	$x <^{x_1}_{x_2} (\lambda y{:}t) \to \lambda y{:}x <^{x_1}_{x_2} t$	(ω_1)	$\lambda x{:}(y \odot t) \to y \odot (\lambda x{:}t); \quad x \neq y$
(γ_2)	$x <^{x_1}_{x_2} (tk) \to (x <^{x_1}_{x_2} t)k; \quad \text{if } x_1; x_2 \in Fv(t)$	(ω_2)	$(x \odot t)k \to x \odot (tk)$
(γ_3)	$x <^{x_1}_{x_2} (tk) \to t(x <^{x_1}_{x_2} k); \quad \text{if } x_1; x_2 \in Fv(k)$	(ω_3)	$t(x \odot k) \to x \odot (tk)$
(γ_4)	$x <^{x_1}_{x_2} (\widehat{y}{:}t) \to \widehat{y}{:}(x <^{x_1}_{x_2} t)$	(ω_4)	$\widehat{x}{:}(y \odot t) \to y \odot (\widehat{x}{:}t); \quad x \neq y$
(γ_5)	$x <^{x_1}_{x_2} (t :: k) \to (x <^{x_1}_{x_2} t) :: k; \quad \text{if } x_1; x_2 \in Fv(t)$	(ω_5)	$(x \odot t) :: k \to x \odot (t :: k)$
(γ_6)	$x <^{x_1}_{x_2} (t :: k) \to t :: (x <^{x_1}_{x_2} k); \quad \text{if } x_1; x_2 \in Fv(k)$	(ω_6)	$t :: (x \odot k) \to x \odot (t :: k)$
$(\gamma\omega_1)$	$x <^{x_1}_{x_2} (y \odot e) \to y \odot (x <^{x_1}_{x_2} e) \qquad x_1 \neq y \neq x_2$	$(\gamma\omega_2)$	$x <^{x_1}_{x_2} (x_1 \odot e) \to e[x{=}x_2]$

Fig. 5. Reduction rules of $\lambda^{\mathsf{Gtz}}_{\mathsf{r}}$-calculus

The first group consists of β, π, σ and μ reductions from λ^{Gtz}. New reductions are added to deal with explicit contraction (γ reductions) and weakening (ω reductions). The groups of γ and ω reductions consist of rules that perform propagation of contraction into the expression and extraction of weakening out of the expression. This discipline allows us to optimize the computation by delaying the duplication of terms on the one hand, and by performing the erasure of terms as soon as possible on the other.

The meta-substitution $v[T{=}x]$ is defined as in Figure 3 with the following additions:

$$\begin{aligned}
(tk)[u{=}x] &= t[u{=}x]k; \;\; x \in Fv(t) \qquad & (tk)[u{=}x] &= tk[u{=}x]; \;\; x \in Fv(k) \\
(\widehat{y}{:}t)[u{=}x] &= \widehat{y}{:}t[u{=}x] & & \\
(t :: k)[u{=}x] &= t[u{=}x] :: k; \;\; x \in Fv(t) \qquad & (t :: k)[u{=}x] &= t :: k[u{=}x]; \;\; x \in Fv(k)
\end{aligned}$$

In the π rule, the meta-operator @, called *append*, joins two contexts and is defined as:

$$(\widehat{x}{:}t)@k' = \widehat{x}{:}tk' \qquad (u :: k)@k' = u :: (k@k')$$
$$(x \odot k)@k' = x \odot (k@k') \qquad (x <^{y}_{z} k)@k' = x <^{y}_{z} (k@k'){:}$$

2 Intersection type assignment systems for resource control

In this section we introduce intersection type assignment systems which assign *strict types* to λ_{r} -terms and $\lambda^{\mathsf{Gtz}}_{\mathsf{r}}$-expressions. Strict types were proposed in [36] and already used in [15] for characterisation of strong normalisation in λ^{Gtz}-calculus.

The syntax of types is defined as follows:

$$\begin{array}{ll} \text{Strict types } \sigma ::= & p \mid \alpha \to \sigma \\ \text{Types} \qquad \alpha ::= & \sigma \mid \sigma \cap \alpha \end{array}$$

where p ranges over a denumerable set of type atoms. We denote types with $\alpha;\beta;\gamma::$ and strict types with $\sigma;\tau;\upsilon:::$. We assume that intersection operator is idempotent, commutative and associative. Due to this property, equivalent terms have the same type.

Definition 1. *(i) A* basic type assignment *is an expression of the form* $x : \alpha$*, where* x *is a term variable and* α *is a type.*

(ii) A basis Γ *is a set* $\{x_1 : \alpha_1;:::;x_n : \alpha_n\}$ *of basic type assignments, where all term variables are different.* $Dom(\Gamma) = \{x_1;:::;x_n\}$*. A basis extension* $\Gamma;x : \alpha$ *denotes the set* $\Gamma \cup \{x : \alpha\}$*, where* $x \notin Dom(\Gamma)$:

(iii) A bases intersection *is* $\cap\Gamma_i = \{x : \cap\alpha_i \mid x : \alpha_i \in \Gamma_i\}$*; where for all* i*;* j*,* $Dom(\Gamma_i) = Dom(\Gamma_j)$:

2.1 Intersection types for λ_{r}

The type assignment system $\lambda_{\mathsf{r}}\cap$ is given in Figure 6.

$$\frac{}{x : \cap\sigma_i \vdash x : \sigma_i}\ (Ax)$$

$$\frac{\Gamma;x : \alpha \vdash M : \sigma}{\Gamma \vdash \lambda x{:}M : \alpha \to \sigma}\ (\to_I) \qquad \frac{\Gamma \vdash M : \cap\alpha_i \to \sigma \quad \Delta_i \vdash N : \alpha_i}{\Gamma;\cap\Delta_i \vdash MN : \sigma}\ (\to_E)$$

$$\frac{\Gamma;x : \alpha;y : \beta \vdash M : \sigma}{\Gamma;z : \alpha\cap\beta \vdash z <^{x}_{y} M : \sigma}\ (Cont) \qquad \frac{\Gamma \vdash M : \sigma}{\Gamma;x : \alpha \vdash x \odot M : \sigma}\ (Weak)$$

Fig. 6. $\lambda_{\mathsf{r}}\cap$: λ_{r} -calculus with intersection types

The Generation lemma induced by the proposed system is the following:

Proposition 2 (Generation lemma for $\lambda_{\mathsf{r}}\cap$).

(i) $\Gamma \vdash \lambda x{:}M : \beta$ *iff there exist* α *and* σ *such that* $\beta \equiv \alpha \to \sigma$ *and* $\Gamma; x : \alpha \vdash M : \sigma$:
(ii) $\Gamma \vdash MN : \sigma$ *iff* $\Gamma = \Gamma'; \cap\Delta_i$ *and there exists a type* $\cap\alpha_i$ *such that* $\Gamma' \vdash M : \cap\alpha_i \to \sigma$ *and for all* i $\Delta_i \vdash N : \alpha_i$:
(iii) $\Gamma \vdash z <^{x}_{y} M : \sigma$ *iff there exist* $\Gamma'; \alpha; \beta$ *such that* $\Gamma = \Gamma'; z : \alpha \cap \beta$ *and* $\Gamma'; x : \alpha; y : \beta \vdash M : \sigma$:
(iv) $\Gamma \vdash x \odot M : \sigma$ *iff there exist* $\Gamma'; \beta$ *such that* $\Gamma = \Gamma'; x : \beta$ *and* $\Gamma' \vdash M : \sigma$:

The proposed system satisfies the following properties.

Proposition 3. *If* $M \to M'$ *then* $Fv(M) = Fv(M')$:

Proposition 4. *(i) If* $\Gamma \vdash M :$ *, then* $Dom(\Gamma) = Fv(M)$:
(ii) If $\Gamma_1 \vdash M : \sigma$ *and* $\Gamma_2 \vdash M : \sigma$*, then* $\Gamma_1 \cap \Gamma_2 \vdash M : \sigma$.

Proposition 5 (Substitution lemma). *If* $\Gamma; x : \cap\alpha_i \vdash M : \sigma$ *and for all* i, $\Delta_i \vdash N : \alpha_i$, *then* $\Gamma; \cap\Delta_i \vdash M[N{=}x] : \sigma$:

Proposition 6 (Subject reduction and equivalence). *For every* λ_{r}*-term* M*: if* $\Gamma \vdash M : \sigma$ *and* $M \to M'$ *or* $M \equiv M$*, then* $\Gamma \vdash M' : \sigma$:

2.2 Intersection types for $\lambda^{\mathsf{Gtz}}_{\mathsf{r}}$

The type assignment system $\lambda^{\mathsf{Gtz}}_{\mathsf{r}}\cap$ is given in Figure 7.

$$\frac{}{x : \cap\sigma_i \vdash x : \sigma_i}\ (Ax)$$

$$\frac{\Gamma; x : \alpha \vdash t : \sigma}{\Gamma \vdash \lambda x{:}t : \alpha \to \sigma}\ (\to_R) \qquad \frac{\Gamma_i \vdash t : \alpha_i \quad \Delta; \sigma \vdash k : \tau}{\cap\Gamma_i; \Delta; \cap\alpha_i \to \sigma \vdash t :: k : \tau}\ (\to_L)$$

$$\frac{\Gamma_i \vdash t : \alpha_i \quad \Delta; \cap\alpha_i \vdash k : \sigma}{\cap\Gamma_i; \Delta \vdash tk : \sigma}\ (Cut) \qquad \frac{\Gamma; x : \alpha \vdash t : \sigma}{\Gamma; \alpha \vdash \widehat{x}{:}t : \sigma}\ (Sel)$$

$$\frac{\Gamma; x : \alpha; y : \beta \vdash t : \sigma}{\Gamma; z : \alpha \cap \beta \vdash z <^{x}_{y} t : \sigma}\ (Cont_t) \qquad \frac{\Gamma \vdash t : \sigma}{\Gamma; x : \alpha \vdash x \odot t : \sigma}\ (Weak_t)$$

$$\frac{\Gamma; x : \alpha; y : \beta; \gamma \vdash k : \sigma}{\Gamma; z : \alpha \cap \beta; \gamma \vdash z <^{x}_{y} k : \sigma}\ (Cont_k) \qquad \frac{\Gamma; \gamma \vdash k : \sigma}{\Gamma; x : \alpha; \gamma \vdash x \odot k : \sigma}\ (Weak_k)$$

Fig. 7. $\lambda^{\mathsf{Gtz}}_{\mathsf{r}}\cap$: $\lambda^{\mathsf{Gtz}}_{\mathsf{r}}$-calculus with intersection types

The Generation lemma induced by the proposed system is the following:

Proposition 7 (Generation lemma for $\lambda^{\mathsf{Gtz}}_{\mathsf{r}}\cap$).

(i) $\Gamma \vdash \lambda x{:}t : \beta$ *iff there exist* α *and* σ *such that* $\beta \equiv \alpha \to \sigma$ *and* $\Gamma; x : \alpha \vdash t : \sigma$:

(ii) $\Gamma; \gamma \vdash t :: k : \tau$ *iff* $\Gamma = \cap\Gamma_i; \Delta$, $\gamma \equiv \cap\alpha_i \to \sigma$; *and* $\Gamma_i \vdash t : \alpha_i; \forall i$ *and* $\Delta; \sigma \vdash k : \tau$.

(iii) $\Gamma \vdash tk : \sigma$ *iff* $\Gamma = \cap\Gamma_i; \Delta$ *and there exists a type* $\cap\alpha_i$ *such that* $\Gamma_i \vdash t : \alpha_i$; $\forall i$ *and* $\Delta; \cap\alpha_i \vdash k : \sigma$:

(iv) $\Gamma; \alpha \vdash \widehat{x}{:}t : \sigma$ *iff* $\Gamma; x : \alpha \vdash t : \sigma$:

(v) $\Gamma \vdash z <^{x}_{y} t : \sigma$ *iff there exist* $\Gamma'; \alpha; \beta$ *such that* $\Gamma = \Gamma'; z : \alpha \cap \beta$ *and* $\Gamma'; x : \alpha; y : \beta \vdash t : \sigma$:

(vi) $\Gamma \vdash x \odot t : \sigma$ *iff there exist* $\Gamma'; \beta$ *such that* $\Gamma = \Gamma'; x : \beta$ *and* $\Gamma' \vdash t : \sigma$:

(vii) $\Gamma; \varepsilon \vdash z <^{x}_{y} k : \sigma$ *iff there exist* $\Gamma'; \alpha; \beta$ *such that* $\Gamma = \Gamma'; z : \alpha \cap \beta$ *and* $\Gamma; x : \alpha; y : \beta; \varepsilon \vdash k : \sigma$:

(viii) $\Gamma; \gamma \vdash x \odot k : \sigma$ *iff there exist* $\Gamma; \beta$ *such that* $\Gamma = \Gamma'; x : \beta$ *and* $\Gamma; \gamma \vdash k : \sigma$:

3 Typability ⇒ SN in both systems

3.1 Typeability ⇒ SN in $\lambda_{\mathtt{r}}\cap$

The main idea of the reducibility method, introduced in Tait [35] for proving the strong normalization property for the simply typed lambda calculus, is to interpret types by suitable sets of lambda terms which satisfy certain realizability properties.

In the remainder of the paper we consider $\Lambda_{\mathtt{r}}$ as the *applicative structure* whose domain are $\lambda_{\mathtt{r}}$-terms and where the application is just the application of $\lambda_{\mathtt{r}}$-terms. We recall some notions from [4]. The set of *strongly normalizing terms* is defined as

$$\mathcal{SN} = \{M \in \Lambda_{\mathtt{r}} \mid \neg(\exists M_1; M_2; \ldots \in \Lambda_{\mathtt{r}})\, M \to M_1 \to M_2 \to \ldots\}:$$

Definition 8. *For* $\mathcal{M}; \mathcal{N} \subseteq \Lambda_{\mathtt{r}}$, *we define* $\mathcal{M} \longrightarrow \mathcal{N} \subseteq \Lambda_{\mathtt{r}}$ *as*

$$\mathcal{M} \longrightarrow \mathcal{N} = \{N \in \Lambda_{\mathtt{r}} \mid \forall M \in \mathcal{M}: \quad (fv(\mathcal{M}) \cap fv(\mathcal{N}) = \emptyset \quad \Rightarrow \quad NM \in \mathcal{N})\}:$$

Definition 9. *The type interpretation* $[\![-]\!] : Types \to 2^{\Lambda_{\mathtt{r}}}$ *is defined by:*

(I1) $[\![p]\!] = \mathcal{SN}$, *where p is a type atom;*

(I2) $[\![\sigma \cap \alpha]\!] = [\![\sigma]\!] \cap [\![\alpha]\!]$;

(I3) $[\![\alpha \to \sigma]\!] = ([\![\alpha]\!] \longrightarrow [\![\sigma]\!]) = \{M \in \Lambda_{\mathtt{r}} \mid \forall N \in [\![\alpha]\!] \quad MN \in [\![\sigma]\!]\}$.

Next, we introduce the notions of *saturation property*, obtained by extending the saturation property given in [5], and *weakening property*. To this aim we introduce the following notation: if R denotes the set of reductions given in Figure 2, $r \in R \setminus (\beta)$, then $redex_r$ ($contr_r$) denote the left (right) hand side of the reduction r (its redex and contractum, respectively).

Definition 10.

- *A set* $X \subseteq \mathcal{SN}$ *satisfies the* saturation property, *notation* $\mathsf{SAT}(\mathcal{X})$, *if*
  - $\mathsf{VAR}(\mathcal{X})$: $(\forall n \geq 0)\ (\forall x \in \mathtt{var})\ (\forall M_1; \ldots; M_n \in \mathcal{SN})$
    $(x \cap fv(M_1) \cap \ldots \cap fv(M_n) = \emptyset \quad \Rightarrow \quad xM_1 \ldots M_n \in \mathcal{X}$:

- $\mathsf{SAT}_\beta(\mathcal{X})$:[4] $(\forall n \geq 0)\,(\forall M_1;\ldots;M_n \in \mathcal{SN})$
  $M[N{=}x]M_1 \ldots M_n \in \mathcal{X} \;\Rightarrow\; (\lambda x{:}M)NM_1 \ldots M_n \in \mathcal{X}$:
- $\mathsf{SAT}_r(\mathcal{X})$: $(\forall n \geq 0)\,(\forall M_1;\ldots;M_n \in \mathcal{SN})$
  $contr_r M_1 \ldots M_n \in \mathcal{X} \;\Rightarrow\; redex_r M_1 \ldots M_n \in \mathcal{X}$:

– *A set* $X \subseteq \mathcal{SN}$ *satisfies the* weakening property, *notation* $\mathsf{WEAK}(\mathcal{X})$,
- $\mathsf{WEAK}(\mathcal{X})$: $(\forall x \in \mathtt{var})\; M \in \mathcal{X};\; x \notin Fv(M) \;\Rightarrow\; x \odot M \in \mathcal{X}$:

Definition 11 (®-Saturated set). *A set* $\mathcal{X} \subseteq \Lambda_{\mathtt{r}}$ *is called* ®-saturated*, if it satisfies the saturation and weakening properties.*

Proposition 12. *Let* $\mathcal{M};\mathcal{N} \subseteq \Lambda_{\mathtt{r}}$.

(i) $\mathcal{SN}$ *is ®-saturated.*
(ii) If $\mathcal{M}$ *and* $\mathcal{N}$ *are ®-saturated, then* $\mathcal{M} \longrightarrow \mathcal{N}$ *is ®-saturated.*
(iii) If $\mathcal{M}$ *and* $\mathcal{N}$ *are ®-saturated, then* $\mathcal{M} \cap \mathcal{N}$ *is ®-saturated.*
(iv) For all types $\varphi \in Types$, $[\![\varphi]\!]$ *is ®-saturated.*

We further define a *valuation of terms* $[\![-]\!]_\rho : \Lambda_{\mathtt{r}} \to \Lambda_{\mathtt{r}}$ and the *semantic satisfiability relation* $\models$ which connects the type interpretation with the term valuation.

Definition 13. *Let* $\rho : \mathtt{var} \to \Lambda_{\mathtt{r}}$ *be a valuation of term variables in* $\Lambda_{\mathtt{r}}$. *For* $M \in \Lambda_{\mathtt{r}}$, *with* $Fv(M) = x_1;\ldots;x_n$ *the* term valuation $[\![-]\!]_\rho : \Lambda_{\mathtt{r}} \to \Lambda_{\mathtt{r}}$ *is defined as:*

(i) $[\![x]\!]_\rho = \rho(x)$;
(ii) $[\![MN]\!]_\rho \equiv \begin{cases} [\![M]\!]_\rho [\![N]\!]_\rho; & \text{if } Fv([\![M]\!]_\rho) \cap Fv([\![N]\!]_\rho) = \emptyset \\ Y <^{Y'}_{Y''} ([\![M]\!]_{\rho(Y'=Y)} [\![N]\!]_{\rho(Y''=Y)}); & \text{if } Fv([\![M]\!]_\rho) \cap Fv([\![N]\!]_\rho) = \{y_1;\ldots;y_k\} \end{cases}$
where $Y = \{y_1;\ldots;y_k\}$, $Y' = \{y'_1;\ldots;y'_k\}$ *and* $Y'' = \{y''_1;\ldots;y''_k\}$ *and* $\rho(Y'{=}Y)$ *denotes* $\rho(y'_1{=}y_1;\ldots;y'_k{=}y_k)$ *(similarly for* $\rho(Y''{=}Y)$*).*
(iii) $[\![\lambda x{:}M]\!]_\rho \equiv \lambda x{:}[\![M]\!]_{\rho(x=x)}$:
(iv) $[\![x \odot M]\!]_\rho \equiv Fv(\rho(x)) \odot [\![M]\!]_\rho$:
(v) $[\![z <^x_y M]\!]_\rho \equiv Fv(\rho(z)) <^{Fv(N_1)}_{Fv(N_2)} [\![M]\!]_{\rho(N_1=x;N_2=y)}$
where N_1 *and* N_2 *are obtained from* $\rho(z)$ *by renaming its free variables.*

Lemma 14.

(i) $[\![M]\!]_{\rho(N=x)} \equiv [\![M]\!]_{\rho(x=x)}[N{=}x]$:
(ii) $[\![z <^x_y M]\!]_{\rho(N=z)} \equiv (z <^x_y [\![M]\!]_{\rho(x=x;y=y)})[N{=}z]$:
(iii) $[\![M]\!]_{\rho(N=x;N=y)} \equiv Fv(N) <^{Fv(N')}_{Fv(N'')} [\![M]\!]_{\rho(N'=x;N''=y)}$; *where* N' *and* N'' *are obtained from N by renaming all free variables of N with fresh variables.*

Proof. By induction on the construction of M. For the cases (i)-(iv) we consider only the base cases when M is a variable, other cases being straightforward using IH.

(i) $[\![y]\!]_{\rho(N=x)} = y[N{=}x;\rho(y){=}y] = \rho(y)$.
$[\![y]\!]_{\rho(x=x)}[N{=}x] = y[x{=}x;\rho(y){=}y][N{=}x] = \rho(y)$.

[4] Notice that we do not need a condition that $N \in \mathcal{SN}$ in $\mathsf{SAT}_\beta(\mathcal{X})$ since we only work with linear terms, hence if the contractum $M[N{=}x] \in \mathcal{SN}$, then $N \in \mathcal{SN}$.

(ii) Using (i) and the definition of substitution.
$[\![z <_y^x M]\!]_{\rho(N=z)} = [\![z <_y^x M]\!]_{\rho(z=z)}[N=z] = (z <_y^x [\![M]\!]_{\rho(x=x;y=y)})[N=z] =$
$Fv(N) <_{Fv(N_2)}^{Fv(N_1)} [\![M]\!]_{\rho(x=x;y=y)}[N_1=x][N_2=y] = Fv(N) <_{Fv(N_2)}^{Fv(N_1)} [\![M]\!]_{\rho(N_1=x;N_2=y)} =$
$Fv(N) <_{Fv(N_2)}^{Fv(N_1)} [\![M]\!]_{\rho(x=x;y=y)}[N_1=x][N_2=y] = (z <_y^x [\![M]\!]_{\rho(x=x;y=y)})[N=z]$:
(iii) By straightforward application od Definition 13.

Definition 15.

(i) $\rho \models M : \alpha \iff [\![M]\!]_\rho \in [\![\alpha]\!]$;
(ii) $\rho \models \Gamma \iff (\forall (x:\alpha) \in \Gamma) \quad \rho(x) \in [\![\alpha]\!]$;
(iii) $\Gamma \models M : \alpha \iff (\forall \rho; \rho \models \Gamma \Rightarrow \rho \models M : \alpha)$.

Proposition 16 (Soundness of $\lambda_{\mathsf{r}} \cap$). *If* $\Gamma \vdash M : \alpha$, *then* $\Gamma \models M : \alpha$.

Proof. By induction on the derivation of $\Gamma \vdash M : \alpha$. The cases (Ax) and $(\to_I)$ are analogous to the corresponding rules in ordinary λ calculus. We prove the statement for the remaining inference rules.

- The last rule applied is $(\to_E)$, i.e., $\Gamma \vdash M : \cap\alpha_i \to \sigma; \Delta_i \vdash N : \alpha_i \Rightarrow \Gamma; \cap\Delta_i \vdash MN : \sigma$: By the IH $\Gamma \models M : \cap\alpha_i \to \sigma$ and $\Delta_i \models N : \alpha_i; \forall i$. Suppose that $\rho \models \Gamma; \cap\Delta_i$, then $\rho \models \Gamma$ and $\rho \models \cap\Delta_i$. From $\rho \models \Gamma$, using the IH we deduce that $[\![M]\!]_\rho \in [\![\cap\alpha_i \to \sigma]\!]$. From $\rho \models \cap\Delta_i$, we deduce that $\rho \models \Delta_i; \forall i$ (since every variable $x : \alpha \in \cap\Delta_i$ is of the form $x : \cap\alpha_i; x : \alpha_i \in \Delta_i$), hence using the IH we deduce that $[\![N]\!]_\rho \in [\![\alpha_i]\!]; \forall i$. This means that $[\![N]\!]_\rho \in \cap[\![\alpha_i]\!]_\rho = [\![\cap\alpha_i]\!]_\rho$. Using Definition 13(ii) we obtain that $[\![M]\!]_\rho [\![N]\!]_\rho = [\![MN]\!]_\rho \in [\![\sigma]\!]$.

- The last rule applied is $(Weak)$, i.e., $\Gamma \vdash M : \alpha \Rightarrow \Gamma; x : \beta \vdash x \odot M : \alpha$. By the IH $\Gamma \models M : \alpha$. Suppose that $\rho \models \Gamma; x : \beta \Leftrightarrow \rho \models \Gamma$ and $\rho \models x : \beta$. From $\rho \models \Gamma$ we obtain $[\![M]\!]_\rho \in [\![\alpha]\!]$. Using the weakening property WEAK and Definition 13(iv) we obtain $Fv(\rho(x)) \odot [\![M]\!]_\rho = [\![x \odot M]\!]_\rho \in [\![\alpha]\!]$, since $Fv(\rho(x)) \cap Fv([\![M]\!]_\rho) = \emptyset$.

- The last rule applied is $(Cont)$, i.e., $\Gamma; x : \alpha; y : \beta \vdash M : \gamma \Rightarrow \Gamma; z : \alpha \cap \beta \vdash z <_y^x M : \gamma$. By the IH $\Gamma; x : \alpha; y : \beta \models M : \gamma$. Suppose that $\rho \models \Gamma; z : \alpha \cap \beta$, in order to prove $[\![z <_y^x M]\!]_\rho \in [\![\gamma]\!]$. This means that $\rho \models \Gamma$ and $\rho \models z : \alpha \cap \beta \Leftrightarrow \rho(z) \in [\![\alpha]\!]$ and $\rho(z) \in [\![\beta]\!]$. For the sake of simplicity let $\rho(z) \equiv N$. We define a new ρ' such that $\rho' = \rho(N=x; N=y)$. Then $\rho' \models \Gamma; x : \alpha; y : \beta$ since $x; y \notin Dom(\Gamma)$, $N \in [\![\alpha]\!]$ and $N \in [\![\beta]\!]$. By the IH $[\![M]\!]_{\rho'} \in [\![\gamma]\!]$. By the definition of term valuation (Definition 13), Lemma 14(i), (ii) and (iii) and the definition of substitution we obtain $[\![M]\!]_{\rho'} = [\![M]\!]_{\rho(N=x;N=y)} = Fv(N) <_{Fv(N'')}^{Fv(N')} [\![M]\!]_{\rho(N'=x;N''=y)} = Fv(N) <_{Fv(N'')}^{Fv(N')} [\![M]\!]_{\rho(x=x;y=y)}[N'=x][N''=y] = (z <_y^x [\![M]\!]_{\rho(x=x;y=y)})[N=z] = ([\![z <_y^x M]\!]_{\rho(z=z)})[N=z] = [\![z <_y^x M]\!]_{\rho(N=z)} = [\![z <_y^x M]\!]_\rho$, since $\rho(z) = N$. Hence, $[\![z <_y^x M]\!]_\rho \in [\![\gamma]\!]$. □

Theorem 17 ($\mathcal{SN}$ for $\lambda_{\mathsf{r}} \cap$). *If* $\Gamma \vdash M : \alpha$, *then M is strongly normalizing, i.e.* $M \in \mathcal{SN}$.

Proof. Suppose $\Gamma \vdash M : \alpha$. By Proposition 16 $\Gamma \models M : \alpha$. According to Definition 15(iii), this means that $(\forall \rho \models \Gamma) \quad \rho \models M : \alpha$. We can choose a particular $\rho_0(x) = x$ for all $x \in \mathtt{var}$. By Proposition 12(iv), $[\![\beta]\!]$ is saturated for each type β, hence $x = [\![x]\!]_\rho \in [\![\beta]\!]$ (variable condition for $n = 0$). Therefore, $\rho_0 \models \Gamma$ and we can conclude that $[\![M]\!]_{\rho_0} \in [\![\alpha]\!]$. On the other hand, $M = [\![M]\!]_{\rho_0}$ and $[\![\alpha]\!] \subseteq \mathcal{SN}$ (Proposition 12), hence $M \in \mathcal{SN}$. □

3.2 Typeability ⇒ SN in $\lambda_{\mathsf{r}}^{\mathsf{Gtz}}\cap$

In this section, we prove the strong normalisation property of the $\lambda_{\mathsf{r}}^{\mathsf{Gtz}}$-calculus with intersection types. The termination is proved by showing that the reduction on the set $\Lambda_{\mathsf{r}}^{\mathsf{Gtz}} \cup \Lambda_{\mathsf{r};C}^{\mathsf{Gtz}}$ of the typeable $\lambda_{\mathsf{r}}^{\mathsf{Gtz}}$-expressions is included in a particular well-founded relation, which we define as the lexicographic product of three well-founded component relations. The first one is based on the mapping of $\lambda_{\mathsf{r}}^{\mathsf{Gtz}}$-expressions into λ_{r} -terms. We show that this mapping preserves types and that all $\lambda_{\mathsf{r}}^{\mathsf{Gtz}}$-reductions can be simulated by the reductions or identities of the λ_{r} -calculus. The other two well-founded orders are based on the introduction of quantities designed to decrease a global measure associated with specific $\lambda_{\mathsf{r}}^{\mathsf{Gtz}}$-expressions during the computation.

Definition 18. *The mapping* $\lfloor\ \rfloor : \Lambda_{\mathsf{r}}^{\mathsf{Gtz}} \to \Lambda_{\mathsf{r}}$ *is defined together with the auxiliary mapping* $\lfloor\ \rfloor_k : \Lambda_{\mathsf{r};C}^{\mathsf{Gtz}} \to (\Lambda_{\mathsf{r}} \to \Lambda_{\mathsf{r}})$ *in the following way:*

$$\begin{array}{llll}
\lfloor x\rfloor & = x & \lfloor \widehat{x}{:}t\rfloor_k(M) & = (\lambda x{:}\lfloor t\rfloor)M \\
\lfloor \lambda x{:}t\rfloor & = \lambda x{:}\lfloor t\rfloor & \lfloor t :: k\rfloor_k(M) & = \lfloor k\rfloor_k(M\lfloor t\rfloor) \\
\lfloor x \odot t\rfloor & = x \odot \lfloor t\rfloor & \lfloor x \odot k\rfloor_k(M) & = x \odot \lfloor k\rfloor_k(M) \\
\lfloor x <_z^y t\rfloor & = x <_z^y \lfloor t\rfloor & \lfloor x <_z^y k\rfloor_k(M) & = x <_z^y \lfloor k\rfloor_k(M) \\
\lfloor tk\rfloor & = \lfloor k\rfloor_k(\lfloor t\rfloor) &&
\end{array}$$

Lemma 19. *(i)* $Fv(t) = Fv(\lfloor t\rfloor)$, *for* $t \in \Lambda_{\mathsf{r}}^{\mathsf{Gtz}}$.
(ii) $\lfloor v[t{=}x]\rfloor = \lfloor v\rfloor[\lfloor t\rfloor{=}x]$, *for* $v;t \in \Lambda_{\mathsf{r}}^{\mathsf{Gtz}}$.

We prove that the mappings $\lfloor\ \rfloor$ and $\lfloor\ \rfloor_k$ preserve types. In the sequel, the notation $\Lambda_{\mathsf{r}\ (\Gamma' \vdash_{\lambda_{\mathsf{r}}} \alpha)}$ stands for $\{M \mid M \in \Lambda_{\mathsf{r}}\ \&\ \Gamma' \vdash_{\lambda_{\mathsf{r}}} M : \alpha\}$.

Proposition 20 (Type preservation with $\lfloor\ \rfloor$).

(i) If $\Gamma' \vdash t : \alpha$, *then* $\Gamma' \vdash_{\lambda_{\mathsf{r}}} \lfloor t\rfloor : \alpha$.
(ii) If $\Gamma';\alpha \vdash k : \beta$, *then* $\lfloor k\rfloor_k : \Lambda_{\mathsf{r}\ (\Gamma'' \vdash_{\lambda_{\mathsf{r}}} \alpha)} \to \Lambda_{\mathsf{r}\ (\Gamma';\Gamma'' \vdash_{\lambda_{\mathsf{r}}} \beta)}$, *for some* Γ''.

Proof. The proposition is proved by simultaneous induction on derivations. We distinguish cases according to the last typing rule used.

- Cases (Ax), $(\to_R)$, $(Weak_t)$ and $(Cont_t)$ are easy, because the intersection type assignment system of λ_{r} has exactly the same rules.
- Case (Sel): the derivation ends with the rule

$$\frac{\Gamma';x : \alpha \vdash t : \sigma}{\Gamma';\alpha \vdash \widehat{x}{:}t : \sigma}\ (Sel)$$

By IH we have that $\Gamma';x : \alpha \vdash_{\lambda_{\mathsf{r}}} \lfloor t\rfloor : \sigma$. For any $M \in \Lambda^{\mathsf{r}}$ such that $\Gamma'' \vdash_{\lambda_{\mathsf{r}}} M : \alpha$, for some Γ'', we have

$$\frac{\dfrac{\Gamma';x : \alpha \vdash_{\lambda_{\mathsf{r}}} \lfloor t\rfloor : \sigma}{\Gamma' \vdash_{\lambda_{\mathsf{r}}} \lambda x{:}\lfloor t\rfloor : \alpha \to \sigma}\ (\to_I) \qquad \Gamma'' \vdash_{\lambda_{\mathsf{r}}} M : \alpha}{\Gamma';\Gamma'' \vdash_{\lambda_{\mathsf{r}}} (\lambda x{:}\lfloor t\rfloor)M : \sigma}\ (\to_E)$$

Since $(\lambda x.\lfloor t \rfloor)M = \lfloor \widehat{x}{:}t \rfloor_k(M)$, we conclude that $\lfloor \widehat{x}{:}t \rfloor_k : \Lambda_{\mathsf{r}\ (\Gamma'' \vdash_{\lambda_{\mathsf{r}}} \alpha)} \to \Lambda_{\mathsf{r}\ (\Gamma';\Gamma'' \vdash_{\lambda_{\mathsf{r}}} \sigma)}$.

– Case $(\to_L)$: the derivation ends with the rule

$$\frac{\Gamma'_i \vdash t : \alpha_i \quad \Delta;\sigma \vdash k : \beta}{\cap\Gamma'_i;\Delta;\cap\alpha_i \to \sigma \vdash t :: k : \beta}\ (\to_L)$$

By IH we have that $\Gamma'_i \vdash_{\lambda_{\mathsf{r}}} \lfloor t \rfloor : \alpha_i;\ \forall i$. For any $M \in \Lambda^{\mathsf{r}}$ such that $\Gamma''' \vdash_{\lambda_{\mathsf{r}}} M : \cap\alpha_i \to \sigma$, we have

$$\frac{\Gamma''' \vdash_{\lambda_{\mathsf{r}}} M : \cap\alpha_i \to \sigma \quad \Gamma'_i \vdash_{\lambda_{\mathsf{r}}} \lfloor t \rfloor : \alpha_i}{\cap\Gamma'_i;\Gamma''' \vdash_{\lambda_{\mathsf{r}}} M\lfloor t \rfloor : \sigma}\ (\to_E)$$

From the right-hand side premise in the $(\to_L)$ rule, by IH, we get that $\lfloor k \rfloor_k$ is the function with the scope $\lfloor k \rfloor_k : \Lambda_{\mathsf{r}\ (\Gamma'''' \vdash_{\lambda_{\mathsf{r}}} \sigma)} \to \Lambda_{\mathsf{r}\ (\Gamma'''';\Gamma'' \vdash_{\lambda_{\mathsf{r}}} \beta)}$. For $\Gamma'''' \equiv \cap\Gamma'_i;\Gamma'''$ and by taking $M\lfloor t \rfloor$ as the argument of the function $\lfloor k \rfloor_k$, we get $\cap\Gamma'_i;\Delta;\Gamma''' \vdash_{\lambda_{\mathsf{r}}} \lfloor k \rfloor_k(M\lfloor t \rfloor) : \beta$. Since $\lfloor k \rfloor_k(M\lfloor t \rfloor) = \lfloor t :: k \rfloor_k(M)$, we have that $\cap\Gamma'_i;\Delta;\Gamma''' \vdash_{\lambda_{\mathsf{r}}} \lfloor t :: k \rfloor_k(M) : \beta$. This holds for any M of the appropriate type, yielding $\lfloor t :: k \rfloor_k : \Lambda_{\mathsf{r}\ (\Gamma''' \vdash_{\lambda_{\mathsf{r}}} \cap\alpha_i \to \sigma)} \to \Lambda_{\mathsf{r}\ (\cap\Gamma'_i;\Delta;\Gamma''' \vdash_{\lambda_{\mathsf{r}}} \beta)}$, which is exactly what we need.

– Case (Cut): the derivation ends with the rule

$$\frac{\Gamma'_i \vdash t : \alpha_i \quad \Delta;\cap\alpha_i \vdash k : \sigma}{\cap\Gamma'_i;\Delta \vdash tk : \sigma}\ (Cut)$$

By IH we have that $\Gamma'_i \vdash_{\lambda_{\mathsf{r}}} \lfloor t \rfloor : \alpha$ and $\lfloor k \rfloor_k : \Lambda_{\mathsf{r}\ (\Gamma'' \vdash_{\lambda_{\mathsf{r}}} \cap\alpha_i)} \to \Lambda_{\mathsf{r}\ (\Gamma'';\Delta \vdash_{\lambda_{\mathsf{r}}} \sigma)}$. Hence, for any $M \in \Lambda^{\lambda_{\mathsf{r}}}$ such that $\Gamma'' \vdash_{\lambda_{\mathsf{r}}} M : \cap\alpha_i$, it holds $\Gamma'';\Delta \vdash_{\lambda_{\mathsf{r}}} \lfloor k \rfloor_k(M) : \sigma$. By taking $M \equiv \lfloor t \rfloor$ and $\Gamma'' \equiv \cap\Gamma'_i$, we get $\cap\Gamma'_i;\Delta \vdash_{\lambda_{\mathsf{r}}} \lfloor k \rfloor_k(\lfloor t \rfloor) : \sigma$. But $\lfloor k \rfloor_k(\lfloor t \rfloor) = \lfloor tk \rfloor$, so the proof is done.

– Case $(Weak_k)$: the derivation ends with the rule

$$\frac{\Gamma';\gamma \vdash k : \beta}{\Gamma';x : \alpha;\gamma \vdash x \odot k : \beta}\ (Weak_k)$$

By IH we have that $\lfloor k \rfloor_k$ is the function with the scope $\lfloor k \rfloor_k : \Lambda_{\mathsf{r}\ (\Gamma'' \vdash_{\lambda_{\mathsf{r}}} \gamma)} \to \Lambda_{\mathsf{r}\ (\Gamma';\Gamma'' \vdash_{\lambda_{\mathsf{r}}} \beta)}$, meaning that for each $M \in \Lambda^{\mathsf{r}}$ such that $\Gamma'' \vdash_{\lambda_{\mathsf{r}}} M : \gamma$ holds $\Gamma';\Gamma'' \vdash_{\lambda_{\mathsf{r}}} \lfloor k \rfloor_k(M) : \beta$. Now, we can apply $(Weak)$ rule:

$$\frac{\Gamma';\Gamma'' \vdash \lfloor k \rfloor_k(M) : \beta}{\Gamma';\Gamma'';x : \alpha \vdash x \odot \lfloor k \rfloor_k(M) : \beta}\ (Weak)$$

Since $x \odot \lfloor k \rfloor_k(M) = \lfloor x \odot k \rfloor_k(M)$, this means that $\lfloor x \odot k \rfloor_k : \Lambda_{\mathsf{r}\ (\Gamma'' \vdash_{\lambda_{\mathsf{r}}} \gamma)} \to \Lambda_{\mathsf{r}\ (\Gamma';\Gamma'';x:\alpha \vdash_{\lambda_{\mathsf{r}}} \beta)}$, which is exactly what we wanted to get.

– Case $(Cont_k)$: similar to the case $(Weak_k)$, relying on the rule $(Cont)$ in λ_{r}. □

For the given encoding $\lfloor\ \rfloor$, we show that each $\lambda^{\mathsf{Gtz}}_{\mathsf{r}}$-reduction step can be simulated by λ_{r} -reduction or identity. In order to do so, we prove the following lemmas. The proofs of Lemma 22 and Lemma 23 use Regnier's σ reductions, investigated in [30].

Lemma 21. *If* $M \to_{\lambda_{\mathsf{r}}} M'$, *then* $\lfloor k \rfloor_k(M) \to_{\lambda_{\mathsf{r}}} \lfloor k \rfloor_k(M')$:

Lemma 22. $\lfloor k \rfloor_k((\lambda x{:}P)N) \to_{\lambda_{\mathsf{r}}} (\lambda x{:}\lfloor k \rfloor_k(P))N$:

Lemma 23. *If* $M \in \Lambda^{\mathsf{r}}$ *and* $k; k' \in \Lambda^{\mathsf{Gtz}}_{\mathsf{r};C}$, *then* $\lfloor k' \rfloor_k \circ \lfloor k \rfloor_k(M) \to_{\lambda_{\mathsf{r}}} \lfloor k@k' \rfloor_k(M)$:

Lemma 24. *(i) If* $x \in Fv(k)$, *then* $(\lfloor k \rfloor_k(M))[N{=}x] = \lfloor k \rfloor_k(M[N{=}x])$:
(ii) If $x; y \in Fv(k)$, *then* $z <^x_y (\lfloor k \rfloor_k(M)) \to_{\lambda_{\mathsf{r}}} \lfloor k \rfloor_k(z <^x_y M)$:
(iii) $\lfloor k \rfloor_k(x \odot M) \to_{\lambda_{\mathsf{r}}} x \odot \lfloor k \rfloor_k(M)$:

Now we can prove that the reduction rules of $\lambda^{\mathsf{Gtz}}_{\mathsf{r}}$ can be simulated by the reduction rules or identities in λ_{r} -calculus.

Theorem 25 (Simulation of $\lambda^{\mathsf{Gtz}}_{\mathsf{r}}$-reduction by λ_{r} -reduction).

(i) If term $M \to M'$, *then* $\lfloor M \rfloor \to_{\lambda_{\mathsf{r}}} \lfloor M' \rfloor$.
(ii) If context $k \to k'$ *by* γ_6 *or* ω_6 *reduction, then* $\lfloor k \rfloor_k(M) \equiv \lfloor k' \rfloor_k(M)$, *for any* $M \in \Lambda^{\mathsf{r}}$.
(iii) If context $k \to k'$ *by some other reduction, then* $\lfloor k \rfloor_k(M) \to_{\lambda_{\mathsf{r}}} \lfloor k' \rfloor_k(M)$, *for any* $M \in \Lambda^{\mathsf{r}}$.

The previous proposition shows that each $\lambda^{\mathsf{Gtz}}_{\mathsf{r}}$-reduction step is interpreted either by a λ_{r} -reduction or by an identity. If one wants to prove that there is no infinite sequence of $\lambda^{\mathsf{Gtz}}_{\mathsf{r}}$-reductions one has to prove that there cannot exist an infinite sequence of $\lambda^{\mathsf{Gtz}}_{\mathsf{r}}$-reductions which are all interpreted as identities. To prove this, one shows that if a term is reduced with such a $\lambda^{\mathsf{Gtz}}_{\mathsf{r}}$-reduction, it is reduced for another order that forbids infinite decreasing chains. This order is itself composed of several orders, free of infinite decreasing chains (Definition 29).

Definition 26. *The functions* $\mathcal{S}$; $\| \ \|_C$; $\| \ \|_W : (\Lambda^{\mathsf{Gtz}}_{\mathsf{r}} \cup \Lambda_{\mathsf{r}}) \to \mathbb{N}$ *are defined in Figure 8.*

$\mathcal{S}(x) = 1$	$\|x\|_C = 0$	$\|x\|_W = 1$
$\mathcal{S}(\lambda x{:}t) = 1 + \mathcal{S}(t)$	$\|\lambda x{:}t\|_C = \|t\|_C$	$\|\lambda x{:}t\|_W = 1 + \|t\|_W$
$\mathcal{S}(x \odot e) = 1 + \mathcal{S}(e)$	$\|x \odot e\|_C = \|e\|_C$	$\|x \odot e\|_W = 0$
$\mathcal{S}(x <^y_z e) = 1 + \mathcal{S}(e)$	$\|x <^y_z e\|_C = \|e\|_C + \mathcal{S}(e)$	$\|x <^y_z e\|_W = 1 + \|e\|_W$
$\mathcal{S}(tk) = \mathcal{S}(t) + \mathcal{S}(k)$	$\|tk\|_C = \|t\|_C + \|k\|_C$	$\|tk\|_W = 1 + \|t\|_W + \|k\|_W$
$\mathcal{S}(\widehat{x}{:}t) = 1 + \mathcal{S}(t)$	$\|\widehat{x}{:}t\|_C = \|t\|_C$	$\|\widehat{x}{:}t\|_W = 1 + \|t\|_W$
$\mathcal{S}(t :: k) = \mathcal{S}(t) + \mathcal{S}(k)$	$\|t :: k\|_C = \|t\|_C + \|k\|_C$	$\|t :: k\|_W = 1 + \|t\|_W + \|k\|_W$

Fig. 8. Definitions of $\mathcal{S}(e); \|e\|_C; \|e\|_W$

Lemma 27. *For all* $e; e' : \Lambda_{\mathsf{r}}$ *:*

(i) If $e \to_{\gamma_6} e'$, *then* $\|e\|_C > \|e'\|_C$.
(ii) If $e \to_{\omega_6} e'$, *then* $\|e\|_C = \|e'\|_C$.

Lemma 28. *For all* $e; e' \in \Lambda_{\mathsf{r}}$ *: If* $e \to_{\omega_6} e'$, *then* $\|e\|_W > \|e'\|_W$.

Now we can define the following orders based on the previously introduced mapping and norms.

Definition 29. *We define the following strict orders and equivalencies on* $\Lambda^{\mathsf{Gtz}}_{\mathsf{r}}\cap$*:*

(i) $t > _{\lambda_{\mathsf{r}}} t'$ *iff* $\lfloor t\rfloor \to^{+}_{\lambda_{\mathsf{r}}} \lfloor t'\rfloor$; $t =_{\lambda_{\mathsf{r}}} t'$ *iff* $\lfloor t\rfloor \equiv \lfloor t'\rfloor$;
$k > _{\lambda_{\mathsf{r}}} k'$ *iff* $\lfloor k\rfloor_k(M) \to^{+}_{\lambda_{\mathsf{r}}} \lfloor k'\rfloor(M)$ *for every* λ_{r} *term* M;
$k =_{\lambda_{\mathsf{r}}} k'$ *iff* $\lfloor k\rfloor_k(M) \equiv \lfloor k'\rfloor_k(M)$ *for every* λ_{r} *term* M;
(ii) $e > _c e'$ *iff* $\|e\|_C > \|e'\|_C$; $e =_c e'$ *iff* $\|e\|_C = \|e'\|_C$;
(iii) $e > _w e'$ *iff* $\|e\|_W > \|e'\|_W$; $e =_w e'$ *iff* $\|e\|_W = \|e'\|_W$;

A lexicographic product of two orders $>_1$ and $>_2$ is usually defined as follows ([2]):

$$a > _1 \times_{lex} > _2 b \;\Leftrightarrow\; a > _1 b \;\; or \;\; (a =_1 b \; and \; a > _2 b):$$

Definition 30. *We define the relation* $\gg$ *on* $\Lambda^{\mathsf{Gtz}}_{\mathsf{r}}$ *as the lexicographic product:*

$$\gg \;=\; > _{\lambda_{\mathsf{r}}} \;\times_{lex}\; > _c \;\times_{lex}\; > _w :$$

The following propositions proves that the reduction relation on the set of typed $\lambda^{\mathsf{Gtz}}_{\mathsf{r}}$-expressions is included in the given lexicographic product $\gg$.

Proposition 31. *For each* $e \in \Lambda^{\mathsf{Gtz}}_{\mathsf{r}}$*: if* $e \to e'$*, then* $e \gg e'$*.*

Proof. The proof is by case analysis on the kind of reduction and the structure of $\gg$.
If $e \to e'$ by β, σ, π, μ, γ_1, γ_2, γ_3, γ_4 γ_5, $\gamma\omega_1$, $\gamma\omega_2$, ω_1, ω_2, ω_3 ω_4 or ω_5 reduction, then $e > _{\lambda_{\mathsf{r}}} e'$ by Proposition 25.
If $e \to e'$ by γ_6, then $e =_{\lambda_{\mathsf{r}}} e'$ by Proposition 25, and $e > _c e'$ by Lemma 27.
Finally, if $e \to e'$ by ω_6, then $e =_{\lambda_{\mathsf{r}}} e'$ by Proposition 25, $e =_c e'$ by Lemma 27 and $e > _w e'$ by Lemma 28. □

SN of $\to$ is another terminology for the well-foundness of the relation $\to$ and it is well-known that a relation included in a well-founded relation is well-founded and that the lexicographic product of well-founded relations is well-founded.

Theorem 32 (Strong normalization). *Each expression in* $\Lambda^{\mathsf{Gtz}}_{\mathsf{r}}\cap$ *is SN.*

Proof. The reduction $\to$ is well-founded on $\Lambda^{\mathsf{Gtz}}_{\mathsf{r}}\cap$ as it is included (Proposition 31) in the relation $\gg$ which is well-founded as the lexicographic product of the well-founded relations $>_{\lambda_{\mathsf{r}}}$, $>_c$ and $>_w$. Relation $>_{\lambda_{\mathsf{r}}}$ is based on the interpretation $\lfloor\ \rfloor : \Lambda^{\mathsf{Gtz}}_{\mathsf{r}} \to \Lambda_{\mathsf{r}}$. By Proposition 20 typeability is preserved by the interpretation $\lfloor\ \rfloor$ and $\to_{\lambda_{\mathsf{r}}}$ is SN (i.e., well-founded) on $\Lambda_{\mathsf{r}}\cap$ (Section 3.1), hence $>_{\lambda_{\mathsf{r}}}$ is well-founded on $\Lambda^{\mathsf{Gtz}}_{\mathsf{r}}\cap$. Similarly, $>_c$ and $>_w$ are well-founded, as they are based on interpretations into the well-founded relation $>$ on the set $\mathbb{N}$ of natural numbers. □

4 SN ⇒ Typability in both systems

4.1 SN ⇒ Typability in $\lambda_{\mathsf{r}} \cap$

We want to prove that if a λ_{r}-term is SN, then it is typable in the system $\lambda_{\mathsf{r}} \cap$. We proceed in two steps: 1) we show that all λ_{r}-normal forms are typable and 2) we prove The head subject expansion property. First, let us observe the structure of the λ_{r}-normal forms, given by the following abstract syntax:

$$M_{nf} ::= x \,|\, \lambda x{:}M_{nf} \,|\, \lambda x{:}x \odot M_{nf} \,|\, xM^1_{nf} :: M^n_{nf} \,|\, x <^{x_1}_{x_2} M_{nf}N_{nf};\ \text{if}\ x_1 \in Fv(M_{nf}); x_2 \in Fv(N_{nf})$$
$$W_{nf} ::= x \odot M_{nf} \,|\, x \odot W_{nf}$$

Proposition 33. *λ_{r}-normal forms are typable in the system $\lambda_{\mathsf{r}} \cap$.*

Proposition 34 (Inverse substitution lemma). *Let $\Gamma \vdash M[N{=}x] : \alpha$ and N typable. Then, there are Δ_i and β_i; $i \in I$ such that $\Delta_i \vdash N : \beta_i$; $\forall i$ and $\Gamma'; x : \cap\beta_i \vdash M : \alpha$, where $\Gamma = \Gamma'; \cap\Delta_i$.*

Proof. By induction on the structure of M. □

Proposition 35 (Head subject expansion). *For every λ_{r}-term M: if $M \to M'$, M is contracted redex and $\Gamma \vdash M' : \alpha$, then $\Gamma \vdash M : \alpha$, provided that if $M \equiv (\lambda x{:}N)P \to_\beta N[P{=}x] \equiv M'$, P is typable.*

Proof. By the case study according to the applied reduction. □

Theorem 36 (SN ⇒ typability). *All strongly normalising λ_{r}-terms are typable in the $\lambda_{\mathsf{r}} \cap$ system.*

Proof. The proof is by induction on the length of the longest reduction path out of a strongly normalising term M, with a subinduction on the size of M.

- If M is a normal form, then M is typable by Proposition 33.
- If M is itself a redex, let M' be the term obtained by contracting the redex M. M' is also strongly normalising, hence by IH it is typable. Then M is typable, by Proposition 35. Notice that, if $M \equiv (\lambda x{:}N)P \to_\beta N[P{=}x] \equiv M'$, then, by IH, P is typable, since the length of the longest reduction path out of P is smaller than that of M, and the size of P is smaller than the size of M.
- Next, suppose that M is not itself a redex nor a normal form. Then M is of one of the following forms: $\lambda x{:}N$, $\lambda x{:}x \odot N$, $xM_1 :: M_n$, $x \odot N$, or $x <^{x_1}_{x_2} NP$, $x_1 \in Fv(N)$; $x_2 \in Fv(P)$ (where $M_1; :: ; M_n$, and NP are *not* normal forms). $M_1; :: ; M_n$ and NP are typable by IH, as subterms of M. Then, it is easy to build the typing for M. For instance, let us consider the case $x <^{x_1}_{x_2} NP$ with $x_1 \in Fv(N)$; $x_2 \in Fv(P)$. By induction NP is typable, hence N is typable with say $\Gamma; x_1 : \beta \vdash N : \cap\alpha_i \to \sigma$ and P is typable with say $\Delta_i; x_2 : \gamma_i \vdash P : \alpha_i$. Then using the rule $(E \to)$ we obtain $\Gamma; \cap\Delta_i; x_1 : \beta; x_2 : \cap\gamma_i \vdash NP : \sigma$. Finally, the rule $(Cont)$ yields $\Gamma; \cap\Delta_i; x : \beta \cap (\cap\gamma_i) \vdash x <^{x_1}_{x_2} NP : \sigma$. □

4.2 SN ⇒ Typability in $\lambda^{Gtz}_{r}\cap$

Finally, we want to prove that if a λ^{Gtz}_{r}-term is SN, then it is typable in the system $\lambda^{Gtz}_{r}\cap$. We follow the procedure used in Section 4.1. The proofs are similar to the ones in Section 4.1 and omitted due to the lack of space.

The abstract syntax of λ^{Gtz}_{r}-normal forms is the following:

$$\begin{array}{l} t_{nf} ::= x \,|\, \lambda x{:}t_{nf} \,|\, \lambda x{:}x \odot t_{nf} \,|\, x(t_{nf} :: k_{nf}) \,|\, x <^{y}_{z} y(t_{nf} :: k_{nf}) \\ k_{nf} ::= \widehat{x}{:}t_{nf} \,|\, \widehat{x}{:}x \odot t_{nf} \,|\, t_{nf} :: k_{nf} \,|\, x <^{y}_{z} (t_{nf} :: k_{nf});\ y \in Fv(t_{nf}); z \in Fv(k_{nf}) \\ w_{nf} ::= x \odot e_{nf} \,|\, x \odot w_{nf} \end{array}$$

We use e_{nf} for any λ^{Gtz}_{r}-expression in the normal form.

Proposition 37. *λ^{Gtz}_{r}-normal forms are typable in the system $\lambda^{Gtz}_{r}\cap$.*

The following two lemmas explain the behavior of the meta operators $[=]$ and @ during expansion.

Lemma 38 (Inverse substitution lemma).

(i) Let $\Gamma \vdash t[u{=}x] : \alpha$ and u typable. Then, there exist $\cap\Delta_i$ and $\cap\beta_i$; $i \in I$ such that $\Delta_i \vdash u : \beta_i$; $\forall i$ and $\Gamma'; x : \cap\beta_i \vdash t : \alpha$, where $\Gamma = \Gamma'; \cap\Delta_i$.

(ii) Let $\Gamma; \gamma \vdash k[u{=}x] : \alpha$ and u typable. Then, there exist $\cap\Delta_i$ and $\cap\beta_i$; $i \in I$ such that $\Delta_i \vdash u : \beta_i$; $\forall i$ and $\Gamma'; x : \cap\beta_i; \gamma \vdash k : \alpha$, where $\Gamma = \Gamma'; \cap\Delta_i$.

Lemma 39 (Inverse append lemma). *If $\Gamma; \alpha \vdash k@k' : \sigma$, then $\Gamma = \Gamma'; \Gamma''$ and there is a type $\cap\beta_i$ such that $\Gamma'; \alpha \vdash k : \beta_i$; $\forall i$ and $\Gamma''; \cap\beta_i \vdash k' : \sigma$.*

Now we prove that the type of a term is preserved during the expansion.

Proposition 40 (Head subject expansion). *For every λ^{Gtz}_{r}-term t: if $t \to t'$, t is contracted redex and $\Gamma \vdash t' : \alpha$, then $\Gamma \vdash t : \alpha$.*

Theorem 41 (SN ⇒ typability). *All strongly normalising λ^{Gtz}_{r} terms are typable in the $\lambda^{Gtz}_{r}\cap$ system.*

5 Conclusions

In this paper, we have proposed intersection type assignment systems for λ_{r}-calculus (λ_{cw} of [23]) and λ^{Gtz}_{r}-calculus of [18]. The two intersection type systems proposed here, for resource control lambda and sequent lambda calculus, give a complete characterisation of strongly normalising terms for both calculi. The strong normalisation of typeable resource lambda terms is proved directly by appropriate modification of the reducibility method, whereas the same property for resource sequent lambda terms is proved by well-founded lexicographic order based on suitable embedding into the former calculus. Although the obtained results are not surprising, this paper expands the range of the intersection type techniques and combines different methods in the strict types environment. Unlike the approach of introducing non-idempotent intersection into the calculus with some kind of resource management [27], our intersection is

idempotent. As a consequence, our type assignment system corresponds to full intuitionistic logic, while non-idempotent intersection type assignment systems correspond to intuitionistic linear logic.

Resource control lambda and sequent lambda calculi are good candidates to investigate the computational content of substructural logics ([34]) both in natural deduction and sequent calculus. The motivation for these logics comes from philosophy (Relevant Logics), linguistics (Lambek Calculus) to computing (Linear Logic). The basic idea of resource control is to explicitly handle structural rules, so the absence of (some) structural rules in substructural logics such as weakening, contraction, commutativity, associativity can possibly be handled by resource control operators, which is in the domain of further research. Another direction will involve the investigation of the use of intersection types, being a powerful means for building models of lambda calculus ([6, 12]), in constructing models for sequent lambda calculi.

Acknowledgements: We would like to thank the anonymous referees for careful reading and many valuable comments, which helped us improve the final version of the paper. We would also like to thank Dragiša Žunić for participating in the earlier stages of the work.

References

1. S. Abramsky. Computational interpretations of linear logic. *Theor. Comput Sci.*, 111(1&2):3–57, 1993.
2. F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, UK, 1998.
3. F. Barbanera and S. Berardi. A symmetric lambda calculus for classical program extraction. *Inform. Comput.*, 125(2):103–117, 1996.
4. H. P. Barendregt. *The Lambda Calculus: its Syntax and Semantics*. North-Holland, Amsterdam, revised edition, 1984.
5. H. P. Barendregt. Lambda calculi with types. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, pages 117–309. Oxford University Press, UK, 1992.
6. H. P. Barendregt, M. Coppo, and M. Dezani-Ciancaglini. A filter lambda model and the completeness of type assignment. *J. Symb. Logic*, 48(4):931–940 (1984), 1983.
7. N. Benton, G. Bierman, V. de Paiva, and M. Hyland. A term calculus for intuitionistic linear logic. In *1st International Conference on Typed Lambda Calculus, TLCA '93*, volume 664 of *LNCS*, pages 75–90. Springer, 1993.
8. R. Bloo and K. H. Rose. Preservation of strong normalisation in named lambda calculi with explicit substitution and garbage collection. In *Computer Science in the Netherlands, CSN '95*, pages 62–72, 1995.
9. M. Coppo and M. Dezani-Ciancaglini. A new type-assignment for lambda terms. *Archiv für Mathematische Logik*, 19:139–156, 1978.
10. M. Coppo and M. Dezani-Ciancaglini. An extension of the basic functionality theory for the λ-calculus. *Notre Dame J. Formal Logic*, 21(4):685–693, 1980.
11. P.-L. Curien and H. Herbelin. The duality of computation. In *5th International Conference on Functional Programming, ICFP'00*, pages 233–243. ACM Press, 2000.
12. M. Dezani-Ciancaglini, S. Ghilezan, and S. Likavec. Behavioural Inverse Limit Models. *Theor. Comput Sci.*, 316(1–3):49–74, 2004.

13. D. J. Dougherty, S. Ghilezan, and P. Lescanne. Characterizing strong normalization in the Curien-Herbelin symmetric lambda calculus: extending the Coppo-Dezani heritage. *Theor. Comput Sci.*, 398:114–128, 2008.
14. J. Espírito Santo. Completing Herbelin's programme. In S. R. D. Rocca, editor, *9th International Conference on Typed Lambda Calculi and Applications, TLCA '07*, volume 4583 of *LNCS*, pages 118–132. Springer, 2007.
15. J. Espírito Santo, J. Ivetić, and S. Likavec. Characterising strongly normalising intuitionistic terms. *Fundamenta Informaticae*, 2011. to appear.
16. J. Gallier. Typing untyped λ-terms, or reducibility strikes again! *Ann. Pure Appl. Logic*, 91:231–270, 1998.
17. S. Ghilezan. Strong normalization and typability with intersection types. *Notre Dame J. Formal Logic*, 37(1):44–52, 1996.
18. S. Ghilezan, J. Ivetić, P. Lescanne, and D. Žunić. Intuitionistic sequent-style calculus with explicit structural rules. In *8th International Tbilisi Symposium on Language, Logic and Computation*, volume 6618 of *LNAI*, pages 101–124, 2011.
19. S. Ghilezan and S. Likavec. Computational interpretations of logics. In Z. Ognjanović, editor, *Collection of Papers, special issue Logic in Computer Science 20(12)*, pages 159–215. Mathematical Institute of Serbian Academy of Sciences and Arts, 2009.
20. H. Herbelin. A lambda calculus structure isomorphic to Gentzen-style sequent calculus structure. In L. Pacholski and J. Tiuryn, editors, *Computer Science Logic, CSL '94*, volume 933 of *LNCS*, pages 61–75. Springer, 1995.
21. W. A. Howard. The formulas-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, London, 1980.
22. D. Kesner and S. Lengrand. Resource operators for lambda-calculus. *Inform. Comput.*, 205(4):419–473, 2007.
23. D. Kesner and F. Renaud. The prismoid of resources. In R. K. c and D. Niwiński, editors, *34th International Symposium on Mathematical Foundations of Computer Science, MFCS '09*, volume 5734 of *LNCS*, pages 464–476. Springer, 2009.
24. K. Kikuchi. Simple proofs of characterizing strong normalisation for explicit substitution calculi. In F. Baader, editor, *18th International Conference on Term Rewriting and Applications, RTA'07*, volume 4533 of *LNCS*, pages 257–272. Springer, 2007.
25. R. Matthes. Characterizing strongly normalizing terms of a λ-calculus with generalized applications via intersection types. In J. R. et al., editor, *ICALP Workshops 2000*. Carleton Scientific, 2000.
26. P. M. Neergaard. Theoretical pearls: A bargain for intersection types: a simple strong normalization proof. *J. Funct. Program.*, 15(5):669–677, 2005.
27. M. Pagani and S. R. D. Rocca. Solvability in resource lambda-calculus. In C.-H. L. Ong, editor, *13th International Conference on Foundations of Software Science and Computational Structures, FOSSACS 2010*, volume 6014 of *LNCS*, pages 358–373. Springer, 2010.
28. M. Parigot. Lambda-mu-calculus: An algorithmic interpretation of classical natural deduction. In A. Voronkov, editor, *3rd International Conference on Logic Programming and Automated Reasoning, LPAR '92*, volume 624 of *LNCS*, pages 190–201. Springer, 1992.
29. 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.
30. L. Regnier. Une équivalence sur les lambda-termes. *Theor. Comput Sci.*, 126(2):281–292, 1994.
31. K. H. Rose. CRSX - Combinatory Reduction Systems with Extensions. In M. Schmidt-Schauß, editor, *22nd International Conference on Rewriting Techniques and Applications,*

RTA'11, volume 10 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 81–90. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2011.
32. K. H. Rose. Implementation Tricks That Make CRSX Tick. Talk at IFIP 1.6 workshop, RDP'2011, May 2011.
33. P. Sallé. Une extension de la théorie des types en lambda-calcul. In G. Ausiello and C. Böhm, editors, *5th International Conference on Automata, Languages and Programming, ICALP '78*, volume 62 of *LNCS*, pages 398–410. Springer, 1978.
34. P. Schroeder-Heister and K. Došen. *Substructural Logics*. Oxford University Press, UK, 1993.
35. W. W. Tait. Intensional interpretations of functionals of finite type I. *J. Symb. Logic*, 32:198–212, 1967.
36. S. van Bakel. Complete restrictions of the intersection type discipline. *Theor. Comput Sci.*, 102(1):135–163, 1992.