Resource control and intersection types: an intrinsic connection

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

[1]University of Novi Sad, Faculty of Technical Sciences, Serbia
[2]University of Lyon, École Normal Supérieure de Lyon, France
[3]Dipartimento di Informatica, Università di Torino, Italy

December 6, 2014

Abstract

In this paper we investigate the $\lambda_{\circledR}$-calculus, a λ-calculus enriched with resource control. Explicit control of resources is enabled by the presence of erasure and duplication operators, which correspond to thinning and contraction rules in the type assignment system. We introduce directly the class of $\lambda_{\circledR}$-terms and we provide a new treatment of substitution by its decomposition into atomic steps. We propose an intersection type assignment system for $\lambda_{\circledR}$-calculus which makes a clear correspondence between three roles of variables and three kinds of intersection types. Finally, we provide the characterisation of strong normalisation in $\lambda_{\circledR}$-calculus by means of an intersection type assignment system. This process uses typeability of normal forms, redex subject expansion and reducibility method.

Keywords: lambda calculus resource control intersection types strong normalisation typeability

Introduction

The notion of resource awareness and control has gained an important role both in theoretical and applicative domains: in logic and lambda calculus as well as

in programming langugages and compiler design.The idea to control the use of formulae is present in Gentzen's structural rules ([23]), whereas the idea to control the use of variables can be traced back to Church's λI-calculus (e.g. [4]). The augmented ability to control the number and order of uses of operations and objects has a wide range of applications which enables, among others, compiler optimisations and memory management that prevents memory leaking (e.g. [55]).

In this paper, we investigate the control of resources in the λ-calculus. We propose the $\lambda_{\circledR}$-calculus, a λ-calculus enriched with resource control operators. The explicit control of resources is enabled by the presence of *erasure* and *duplication* operators, which correspond to thinning and contraction rules in the type assignment system. Erasure is the operation that indicates that a variable is not present in the term anymore, whereas duplication indicates that a variable will have two occurrences in the term which receive specific names to preserve the "linearity" of the term. Indeed, in order to control all resources, in the spirit of the λI-calculus (see e.g. [4]), void lambda abstractions are not acceptable, so in order to have $\lambda x.M$ well-formed the variable x has to occur in M. But if x is not used in the term M, one must perform an *erasure* by using the expression $x \odot M$. In this way, the term M does not contain the variable x, but the term $x \odot M$ does. Similarly, a variable should not occur twice. If nevertheless, we want to have two positions for the same variable, we have to duplicate it explicitly, using fresh names. This is done by using the operator $x <^{x_1}_{x_2} M$, called *duplication* which creates two fresh variables x_1 and x_2.

Outline of the paper We first introduce the syntax and reduction rules of the $\lambda_{\circledR}$-calculus (Section 1). We then introduce intersection types into the $\lambda_{\circledR}$-calculus (Section 2). Finally, by means of intersection types, we completely caracterise strong normalisation in $\lambda_{\circledR}$ (Section 3).

Section 1 We first introduce the syntax and reduction rules of the $\lambda_{\circledR}$-calculus. Explicit control of erasure and duplication leads to decomposition 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 chose a direct approach to term calculi rather than taking a more common path through linear logic [1, 7].

Although the design of our calculus has been motivated by theoretical con-

siderations, it may have practical implications as well. Indeed, for instance in the description of compilers by rules with binders [45, 46], the implementation of substitutions of linear variables by inlining[1] 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, a precise control of erasing does not require a garbage collector and prevents memory leaking.

Section 2 Intersection types were introduced in [13, 14, 44, 48] to overcome the limitations of the simple type discipline in which the only forming operator is an arrow $\to$. The newly obtained intersection type assignment systems enable a complete characterisation of termination of term calculi [53, 21, 24]. Later on, intersection types became a powerful tool for characterising strong normalisation in different calculi [18, 34, 39, 42].

We propose an intersection type assignment system $\lambda_{\circledR}\cap$ that integrates intersection into logical rules, thus preserving syntax-directedness of the system. We assign a restricted form of intersection types to terms, namely strict types, therefore minimizing the need for pre-order on types.

Intersection types in the presence of resource control operators were firstly introduced in [26], where two systems with idempotent intersection were proposed. Later, non-idempotent intersection types for contraction and weakening are treated in [8]. In this paper, we treat a general form of intersection without any assumptions about idempotence. As a consequence, our intersection type system can be considered both as idempotent or as non-idempotent, both options having their benefits depending on the motivation.

Intersection types fit naturally with resource control. Indeed, the control allows us to consider three roles of variables: variables as placeholders (the traditional view of λ-calculus), variables to be duplicated and variables to be erased because they are irrelevant. For each kind of a variable, there is a kind of type associated to it, namely a strict type for a *placeholder*, an intersection type for a variable *to-be-duplicated*, and a specific type $\top$ for an *erased* variable.

Section 3 By the means of the introduced intersection type assignment system $\lambda_{\circledR}\cap$, we manage to completely characterise strong normalisation in $\lambda_{\circledR}$, i.e. we prove that terms in the $\lambda_{\circledR}$-calculus enjoy strong normalisation if and only if they

[1] *Inlining* is the technique which consists in copying at compile time the text of a function instead of implementing a call to that function.

are typeable in $\lambda_{\textcircled{R}}\cap$. First, we prove that all strongly normalising terms are typeable in the $\lambda_{\textcircled{R}}$-calculus by using typeability of normal forms and redex subject expansion. We then prove that terms typeable in $\lambda_{\textcircled{R}}$-calculus are strongly normalising by adapting the reducibility method for explicit resource control operators.

Main contributions The main contributions of this paper are:

(i) an improved presentation of resource control lambda calculus syntax with a direct definition of the syntax of resource control terms. Other presentations define first an unconstrainted syntax of terms with duplication and erasure which is later restricted to linear terms;

(ii) a new treatment of substitution and its decomposition into more atomic steps;

(iii) an intersection type assignment system for resource control lambda calculus which makes explicit the intrinsic correspondence between three kinds of variables and three kinds of intersection types;

(iv) a characterisation of strong normalisation in $\lambda_{\textcircled{R}}$-calculus by means of an intersection type assignment system, by using typeability of normal forms, redex subject expansion and reducibility.

Contents

1 Resource control lambda calculus $\lambda_{\textcircled{R}}$ 5
1.1 Syntax 5
1.2 Substitution 8
1.3 Operational semantics 16

2 Intersection types for $\lambda_{\textcircled{R}}$ 19
2.1 The type assignment system 19
2.2 Structural properties 24

3 Characterisation of strong normalisation in $\lambda_{\textcircled{R}}$ 31
3.1 SN $\Rightarrow$ Typeability in $\lambda_{\textcircled{R}}\cap$ 31
3.2 Typeability $\Rightarrow$ SN in $\lambda_{\textcircled{R}}\cap$ 35

4 Related work and conclusions 44

1 Resource control lambda calculus $\lambda_{\mathbb{R}}$

The *resource control* lambda calculus, $\lambda_{\mathbb{R}}$, is an extension of the λ-calculus with explicit erasure and duplication.

1.1 Syntax

Terms and lists, respectively sets, of free variables in $\lambda_{\mathbb{R}}$ are mutually recursively defined.

Definition 1.

(i) The set of $\lambda_{\mathbb{R}}$-terms, denoted by $\Lambda_{\mathbb{R}}$, is defined by inference rules given in Figure 1.

(ii) The list of free variables of a term M, denoted by $Fv[M]$, is defined by inference rules given in Figure 2.

(iii) The set of free variables of a term M, denoted by $Fv(M)$, is obtained from the list $Fv[M]$ by unordering.

$$\frac{}{x \in \Lambda_{\mathbb{R}}}\ (var)$$

$$\frac{M \in \Lambda_{\mathbb{R}}\ \ x \in Fv(M)}{\lambda x.M \in \Lambda_{\mathbb{R}}}\ (abs) \qquad \frac{M \in \Lambda_{\mathbb{R}}\ \ N \in \Lambda_{\mathbb{R}}\ \ Fv(M) \cap Fv(N) = \emptyset}{MN \in \Lambda_{\mathbb{R}}}\ (app)$$

$$\frac{M \in \Lambda_{\mathbb{R}}\ \ x \notin Fv(M)}{x \odot M \in \Lambda_{\mathbb{R}}}\ (era)$$

$$\frac{M \in \Lambda_{\mathbb{R}}\ \ x_1, x_2 \in Fv(M)\ \ x_1 \neq x_2\ \ x \notin Fv(M) \setminus \{x_1, x_2\}}{x <^{x_1}_{x_2} M \in \Lambda_{\mathbb{R}}}\ (dup)$$

Figure 1: $\Lambda_{\mathbb{R}}$: the set of $\lambda_{\mathbb{R}}$-terms

A $\lambda_{\mathbb{R}}$-term, ranged over by $M, N, P, ..., M_1, ...$, can be a variable from an enumerable set $\Lambda_{\mathbb{R}}$ (ranged over by $x, y, z, x_1, \ldots$), an abstraction, an application, an

$$\frac{}{Fv[x] = [x]} \qquad \frac{Fv[M] = [x_1, x_2, ..., x_m]}{Fv[\lambda x_i.M] = [x_1, x_2, ...x_{i-1}, x_{i+1}, ..., x_m]}$$

$$\frac{Fv[M] = [x_1, ..., x_m] \;\; Fv[N] = [y_1, ..., y_n]}{Fv[MN] = [x_1, ..., x_m, y_1, ..., y_n]} \qquad \frac{Fv[M] = [x_1, ..., x_m]}{Fv[x \odot M] = [x, x_1, ..., x_m]}$$

$$\frac{Fv[M] = [x_1, ..., x_m]}{Fv[x <^{x_i}_{x_j} M] = [x, x_1, ...x_{i-1}, x_{i+1}, ...x_{j-1}, x_{j+1}, ..., x_m]}$$

Figure 2: List of free variables of a $\lambda_{\circledR}$-term

erasure or a duplication. The duplication $x <^{x_1}_{x_2} M$ binds the variables x_1 and x_2 in M and introduces a free variable x. The erasure $x \odot M$ introduces also a free variable x. In order to avoid parentheses, we let the scope of all binders extend to the right as much as possible.

Informally, we say that a term is an expression in which every free variable occurs exactly once, and every binder binds (exactly one occurrence of) a free variable. Our notion of terms corresponds to the notion of linear terms in [30]. In that sense, only linear expressions are in the focus of our investigation. In other words, a term is well-formed in $\lambda_{\circledR}$ if and only if bound variables appear actually in the term and variables occur at most once. This assumption is not a restriction, since every pure λ-term has a corresponding $\lambda_{\circledR}$-term and vice versa, due to the embeddings given in Definition 2 and 3 and illustrated by Example 5.

Definition 2. *The mapping* $[\ ]_{rc} : \Lambda \to \Lambda_{\circledR}$ *is defined in the following way:*

$$\begin{array}{rcl} [x]_{rc} & = & x \\ [\lambda x.t]_{rc} & = & \begin{cases} \lambda x.[t]_{rc}, & x \in Fv(t) \\ \lambda x.x \odot [t]_{rc}, & x \notin Fv(t) \end{cases} \\ [MN]_{rc} & = & \begin{cases} [t]_{rc}[s]_{rc}, & Fv(t) \cap Fv(s) = \emptyset \\ x <^{x_1}_{x_2} [t[x_1/x]s[x_2/x]]_{rc}, & x \in Fv(t) \cap Fv(s) \end{cases} \end{array}$$

Reciprocally, a $\lambda_{\circledR}$-term has a corresponding λ-term.

Definition 3. *The mapping* $[\ ]_{\mathcal{R}} : \Lambda_{\mathcal{R}} \to \Lambda$ *is defined in the following way:*

$$\begin{array}{rcl} [x]_{\mathcal{R}} & = & x \\ [\lambda x.M]_{\mathcal{R}} & = & \lambda x.[M]_{\mathcal{R}} \\ [MN]_{\mathcal{R}} & = & [M]_{\mathcal{R}}\,[N]_{\mathcal{R}} \\ [x <_{x_2}^{x_1} M]_{\mathcal{R}} & = & [M]_{\mathcal{R}}[x/x_1][x/x_2] \\ [x \odot M]_{\mathcal{R}} & = & [M]_{\mathcal{R}} \end{array}$$

Proposition 4.

(i) For each pure lambda term $t \in \Lambda$ *there is a term* $M \in \Lambda_{\mathcal{R}}$ *such that* $[t]_{rc} = M$.

(ii) For each resource lambda term $M \in \Lambda_{\mathcal{R}}$ *there is a term* $t \in \Lambda$ *such that* $[M]_{\mathcal{R}} = t$.

Example 5. *Pure* λ*-terms* $\lambda x.y$ *and* $\lambda x.xx$ *are not* $\lambda_{\mathcal{R}}$*-terms, whereas* $[\lambda x.y]_{rc} = \lambda x.(x \odot y)$ *and* $[\lambda x.xx]_{rc} = \lambda x.x <_{x_2}^{x_1} (x_1x_2)$ *are both* $\lambda_{\mathcal{R}}$*-terms.*

$$\dfrac{\dfrac{\dfrac{}{y \in \Lambda_{\mathcal{R}}}\,(var) \qquad x \notin Fv(y)}{x \odot y \in \Lambda_{\mathcal{R}}}\,(era) \qquad x \in Fv(x \odot y)}{\lambda x.x \odot y \in \Lambda_{\mathcal{R}}}\,(abs)$$

$$\dfrac{\dfrac{\dfrac{\vdots}{x_1x_2 \in \Lambda_{\mathcal{R}}} \qquad x \notin Fv(x_1x_2) \setminus \{x_1,x_2\} \quad x_1,x_2 \in Fv(x_1x_2)}{x <_{x_2}^{x_1} (x_1x_2) \in \Lambda_{\mathcal{R}}}\,(dup) \qquad x \in Fv(x <_{x_2}^{x_1} (x_1x_2))}{\lambda x.x <_{x_2}^{x_1} (x_1x_2) \in \Lambda_{\mathcal{R}}}\,(abs)$$

In the sequel, we use the following abbreviations:

- $x_1 \odot \ldots x_n \odot M$ is abbreviated to $X \odot M$, when X is the list $[x_1, x_2, \ldots, x_n]$;
- $x_1 <_{z_1}^{y_1} \ldots x_n <_{z_n}^{y_n} M$ is abbreviated to $X <_Z^Y M$ if X is the list $[x_1, x_2, \ldots, x_n]$, Y is the list $[y_1, y_2, \ldots, y_n]$ and Z is the list $[z_1, z_2, \ldots, z_n]$.

Notice that X, Y and Z are lists of equal length. If $n = 0$, i.e. if X, Y and Z are the empty lists, then $X \odot M = X <_Z^Y M = M$. Note that later on due to the equivalence relation defined in Figure 7, in $X \odot M$ we can take X to be the set $\{x_1, x_2, ..., x_n\}$.

In what follows we use Barendregt's convention [4] for variables: in the same context a variable cannot be both free and bound. This applies to binders like $\lambda x.M$ which binds x in M and $x <_{x_2}^{x_1} M$ which binds x_1 and x_2 in M.

1.2 Substitution

At this point, we chose to introduce a *substitution operator* to define *substitution* in $\Lambda_{\mathbb{R}}$. Due to its interference with the linearity of terms and its slight difference with the standard substitution of the λ-calculus, the concept of substitution has to be carefully defined in the $\lambda_{\mathbb{R}}$-calculus. For that reason, in Definition 6 we first make precise the syntax of $\lambda_{\mathbb{R}}^{\boxslash}$, i.e. the language $\lambda_{\mathbb{R}}$ extended with a substitution operator, by providing mutually recursive definitions of $\lambda_{\mathbb{R}}^{\boxslash}$-terms and lists (respectively sets) of free variables (see Figures 3 and 4).

Definition 6.

(i) The set of $\lambda_{\mathbb{R}}^{\boxslash}$-terms, denoted by $\Lambda_{\mathbb{R}}^{\boxslash}$, is defined by inference rules given in Figure 3.

(ii) The list of free variables of a $\lambda_{\mathbb{R}}^{\boxslash}$-term M, denoted by $Fv^{\boxslash}[M]$, is defined by inference rules given in Figure 4.

(iii) The set of free variables of a $\lambda_{\mathbb{R}}^{\boxslash}$-term M, denoted by $Fv^{\boxslash}(M)$, is obtained from the list $Fv^{\boxslash}[M]$ by unordering.

Notice that the set $\Lambda_{\mathbb{R}}$ is a strict subset of the set $\Lambda_{\mathbb{R}}^{\boxslash}$, $\Lambda_{\mathbb{R}} \subset \Lambda_{\mathbb{R}}^{\boxslash}$, and that N in $M[N/x]$ is substitution free, therefore we can write both $Fv^{\boxslash}(N)$ and $Fv(N)$ for N in $M[N/x]$. Also, notice that if a term M is substitution free, then $Fv^{\boxslash}(M) = Fv(M)$. Barendregt's convention applies to the substitution operator as well, where $M[N/x]$ can be seen as a binder for x in M.

Definition 7.

$$\frac{}{x \in \Lambda^{\boxtimes}_{\circledR}} \ (var)$$

$$\frac{M \in \Lambda^{\boxtimes}_{\circledR} \ \ x \in Fv^{\boxtimes}(M)}{\lambda x.M \in \Lambda^{\boxtimes}_{\circledR}} \ (abs) \qquad \frac{M \in \Lambda^{\boxtimes}_{\circledR} \ \ N \in \Lambda^{\boxtimes}_{\circledR} \ \ Fv^{\boxtimes}(M) \cap Fv^{\boxtimes}(N) = \emptyset}{MN \in \Lambda^{\boxtimes}_{\circledR}} \ (app)$$

$$\frac{M \in \Lambda^{\boxtimes}_{\circledR} \ \ x \notin Fv^{\boxtimes}(M)}{x \odot M \in \Lambda^{\boxtimes}_{\circledR}} \ (era)$$

$$\frac{M \in \Lambda^{\boxtimes}_{\circledR} \ \ x \notin Fv^{\boxtimes}(M) \setminus \{x_1, x_2\} \ \ x_1, x_2 \in Fv^{\boxtimes}(M) \ \ x_1 \neq x_2}{x <^{x_1}_{x_2} M \in \Lambda^{\boxtimes}_{\circledR}} \ (dup)$$

$$\frac{M \in \Lambda^{\boxtimes}_{\circledR} \quad x \in Fv^{\boxtimes}(M) \quad N \in \Lambda_{\circledR} \quad Fv^{\boxtimes}(M) \setminus \{x\} \cap Fv(N) = \emptyset}{M[N/x] \in \Lambda^{\boxtimes}_{\circledR}} \ (sub)$$

Figure 3: $\Lambda^{\boxtimes}_{\circledR}$: the set of $\lambda^{\boxtimes}_{\circledR}$-terms

$$\frac{}{Fv^{\boxtimes}[x] = [x]} \qquad \frac{Fv^{\boxtimes}[M] = [x_1, x_2, ..., x_m]}{Fv^{\boxtimes}[\lambda x_i.M] = [x_1, x_2, ...x_{i-1}, x_{i+1}, ..., x_m]}$$

$$\frac{Fv^{\boxtimes}[M] = [x_1, ..., x_m] \ \ Fv^{\boxtimes}[N] = [y_1, ..., y_n]}{Fv^{\boxtimes}[MN] = [x_1, ..., x_m, y_1, ..., y_n]} \qquad \frac{Fv^{\boxtimes}[M] = [x_1, ..., x_m]}{Fv^{\boxtimes}[x \odot M] = [x, x_1, ..., x_m]}$$

$$\frac{Fv^{\boxtimes}[M] = [x_1, ..., x_m]}{Fv^{\boxtimes}[x <^{x_i}_{x_j} M] = [x, x_1, ...x_{i-1}, x_{i+1}, ...x_{j-1}, x_{j+1}, ..., x_m]}$$

$$\frac{Fv^{\boxtimes}[M] = [x_1, ..., x_m] \quad Fv[N] = [y_1, ..., y_n]}{Fv^{\boxtimes}[M[N/x_i]] = [x_1, x_2, ...x_{i-1}, x_{i+1}, ..., x_m, y_1, ..., y_n]}$$

Figure 4: List of free variables of a $\lambda^{\boxtimes}_{\circledR}$-term

(i) The evaluation of the substitution operator in the $\lambda^{\boxtimes}_{\mathbb{R}}$-term $M[N/x]$, denoted by $\xrightarrow{\boxtimes}$, is defined by the rules given in Figure 5. As usual, it is closed under α-equivalence and regular contexts. In the last row in Figure 5, terms N_1 and N_2 are obtained from the term N by renaming of its free variables, i.e. by substitution of all free variables of N by fresh variables, respectively.

(ii) $\xrightarrow{\boxtimes}\!\!\!\!\rightarrow$ is the reflexive, transitive closure of $\xrightarrow{\boxtimes}$.

$$
\begin{array}{rcl}
x[N/x] & \xrightarrow{\boxtimes} & N \\
(\lambda y.M)[N/x] & \xrightarrow{\boxtimes} & \lambda y.M[N/x],\ \ x \neq y \\
(MP)[N/x] & \xrightarrow{\boxtimes} & M[N/x]P,\ \ x \in Fv^{\boxtimes}(M) \\
(MP)[N/x] & \xrightarrow{\boxtimes} & MP[N/x],\ \ x \in Fv^{\boxtimes}(P) \\
(y \odot M)[N/x] & \xrightarrow{\boxtimes} & y \odot M[N/x],\ \ x \neq y \\
(x \odot M)[N/x] & \xrightarrow{\boxtimes} & Fv(N) \odot M \\
(y <^{y_1}_{y_2} M)[N/x] & \xrightarrow{\boxtimes} & y <^{y_1}_{y_2} M[N/x],\ \ x \neq y \\
(x <^{x_1}_{x_2} M)[N/x] & \xrightarrow{\boxtimes} & Fv[N] <^{Fv[N_1]}_{Fv[N_2]} M[N_1/x_1][N_2/x_2]
\end{array}
$$

Figure 5: Evaluation of the substitution operator in the $\lambda^{\boxtimes}_{\mathbb{R}}$-calculus

For a full understanding of the role of $\lambda^{\boxtimes}_{\mathbb{R}}$, we would like to stress two facts:

- $\xrightarrow{\boxtimes}$ is the operational definition of the substitution in $\Lambda_{\mathbb{R}}$.
- $\xrightarrow{\boxtimes}$ is used with a higher priority than the reductions of $\lambda_{\mathbb{R}}$ given in Figure 6 (because it is used to define substitution in $\Lambda_{\mathbb{R}}$).

To summarise, we have added a new operator to the syntax of $\lambda_{\mathbb{R}}$ called *substitution operator* and denoted by $[\ /\ ]$, and defined the evaluation of the substitution operator, which brings us to $\lambda^{\boxtimes}_{\mathbb{R}}$-calculus.

We prove the following safety property.

Proposition 8.

(i) If $Q \xrightarrow{\boxtimes}\!\!\!\!\rightarrow R$ and $Q \in \Lambda^{\boxtimes}_{\mathbb{R}}$, then $R \in \Lambda^{\boxtimes}_{\mathbb{R}}$.

(ii) If $Q \xrightarrow{\boxtimes}\!\!\!\!\!\rightarrow R$ *then* $Fv^{\boxtimes}(Q) = Fv^{\boxtimes}(R)$.

Proof. These properties are preserved by context. Therefore we can restrict our proof to the case where Q is the instance of the left-hand side of a rule in Figure 5 and consider only one-step reduction $\xrightarrow{\boxtimes}$. We consider only two paradigmatic rules.

- $(M\,P)[N/x] \xrightarrow{\boxtimes} M[N/x]\,P$ with $x \in Fv^{\boxtimes}(M)$.
  - We know that $x \in Fv^{\boxtimes}(M)$. Then $(M\ P)[N/x] \in \Lambda_{\textcircled{R}}^{\boxtimes}$ means that $M \in \Lambda_{\textcircled{R}}^{\boxtimes}$, $P \in \Lambda_{\textcircled{R}}^{\boxtimes}$, $Fv^{\boxtimes}(M) \cap Fv^{\boxtimes}(P) = \emptyset$, $N \in \Lambda_{\textcircled{R}}$ and $(Fv^{\boxtimes}(M) \cup Fv^{\boxtimes}(P)) \setminus \{x\} \cap Fv(N) = \emptyset$. On the other hand, $M[N/x]\ P \in \Lambda_{\textcircled{R}}^{\boxtimes}$ means $M \in \Lambda_{\textcircled{R}}^{\boxtimes}$, $N \in \Lambda_{\textcircled{R}}$, $P \in \Lambda_{\textcircled{R}}^{\boxtimes}$ and $Fv^{\boxtimes}(M[N/x]) \cap Fv^{\boxtimes}(P) = \emptyset$. Since $Fv^{\boxtimes}(M) \cap Fv^{\boxtimes}(P) = \emptyset$ and $((Fv^{\boxtimes}(M) \cup Fv^{\boxtimes}(P)) \setminus \{x\}) \cap Fv(N) = \emptyset$, this implies $Fv^{\boxtimes}(M[N/x]) \cap Fv^{\boxtimes}(P) = \emptyset$, hence the condition on free variables for $M[N/x]\,P$ is fulfilled.
  - $Fv^{\boxtimes}((M\,P)[N/x]) = Fv^{\boxtimes}(M\,P) \setminus \{x\} \cup Fv(N) =$
    $(Fv^{\boxtimes}(M) \cup Fv^{\boxtimes}(P)) \setminus \{x\} \cup Fv(N) =$
    $(Fv^{\boxtimes}(M) \cup Fv(N)) \setminus \{x\} \cup Fv^{\boxtimes}(P) = Fv^{\boxtimes}(M[N/x]\,P)$.
- $(x \odot M)[N/x] \xrightarrow{\boxtimes} Fv(N) \odot M$.
  - $(x \odot M)[N/x] \in \Lambda_{\textcircled{R}}^{\boxtimes}$ means $M \in \Lambda_{\textcircled{R}}^{\boxtimes}$, $x \notin Fv^{\boxtimes}(M)$, $N \in \Lambda_{\textcircled{R}}$ and $Fv^{\boxtimes}(M) \cap Fv(N) = \emptyset$. On the other hand, $Fv(N) \odot M \in \Lambda_{\textcircled{R}}^{\boxtimes}$ means $M \in \Lambda_{\textcircled{R}}^{\boxtimes}$ and $Fv(N) \cap Fv^{\boxtimes}(M) = \emptyset$.
  - $Fv^{\boxtimes}((x \odot M)[N/x]) = Fv^{\boxtimes}(M) \cup Fv(N) = \bigcup_{y \in Fv(N)} \{y\} \cup Fv^{\boxtimes}(M) =$
    $Fv^{\boxtimes}(Fv(N) \odot M)$.

□

Figure 5 defines the evaluation of substitution in $\Lambda_{\textcircled{R}}$. Indeed, the reduction $\xrightarrow{\boxtimes}\!\!\!\!\!\rightarrow$ terminates (Proposition 10) and when it terminates it yields actually a term in $\Lambda_{\textcircled{R}}$, i.e. there is no more substitution operator in the resulting term (Proposition 14). Therefore, there is no need for defining evaluation of $M[N/x]$ in case of $M \equiv Q[P/y]$, because Propositions 10 and 14 guarantee that $Q[P/y]$ will be

evaluated to some $Q' \in \Lambda_{\circledR}$, thus $Q[P/y][N/x] \xrightarrow{\boxslash}\!\!\!\!\!\rightarrow Q'[N/x] \xrightarrow{\boxslash}\!\!\!\!\!\rightarrow Q''$, for some $Q'' \in \Lambda_{\circledR}$.

In order to prove normalisation in Proposition 10, we introduce the following measure.

Definition 9. *The measure $\|\cdot\|_{\boxslash}$ on $\lambda^{\boxslash}_{\circledR}$-terms is defined as follows:*

$$\begin{array}{rcl}
\|x\|_{\boxslash} & = & 1 \\
\|\lambda x.M\|_{\boxslash} & = & \|M\|_{\boxslash} + 1 \\
\|M\,N\|_{\boxslash} & = & \|M\|_{\boxslash} + \|N\|_{\boxslash} + 1 \\
\|x \odot M\|_{\boxslash} & = & \|M\|_{\boxslash} + 1 \\
\|x <^{y}_{z} M\|_{\boxslash} & = & \|M\|_{\boxslash} + 1 \\
\|M[N/x]\|_{\boxslash} & = & \|M\|_{\boxslash}.
\end{array}$$

Proposition 10. *The reduction $\xrightarrow{\boxslash}\!\!\!\!\!\rightarrow$ terminates.*

Proof. The proof of the termination of the relation $\xrightarrow{\boxslash}\!\!\!\!\!\rightarrow$ is based on the measure $\|\cdot\|_{\boxslash}$ defined in Definition 9. We associate with each term M a multiset $\mathcal{M}ul(M)$ of natural numbers as follows:

$$\begin{array}{rcl}
\mathcal{M}ul(x) & = & \{\!\{\ \}\!\} \\
\mathcal{M}ul(\lambda y.M) & = & \mathcal{M}ul(M) \\
\mathcal{M}ul(M\,P) & = & \mathcal{M}ul(M) \cup \mathcal{M}ul(P) \\
\mathcal{M}ul(x \odot M) & = & \mathcal{M}ul(M) \\
\mathcal{M}ul(x <^{y}_{z} M) & = & \mathcal{M}ul(M) \\
\mathcal{M}ul(M[N/x]) & = & \{\!\{\|M\|_{\boxslash}\}\!\} \cup \mathcal{M}ul(M)
\end{array}$$

Notice that if a term P does not contain any substitution, then $\mathcal{M}ul(P) = \{\!\{\ \}\!\}$. The multiset order is defined for instance in [3] and is denoted by $\gg$. The rules in

Figure 5 yield the following inequalities.

$$\begin{array}{rcl}
\{\{\|x\|_{\boxtimes}\} & \gg & \mathcal{M}ul(N) \\
\{\{\|M\|_{\boxtimes}+1\}\} \cup \mathcal{M}ul(M) & \gg & \{\{\|M\|_{\boxtimes}\}\} \cup \mathcal{M}ul(M) \\
\{\{\|M\|_{\boxtimes}+\|P\|_{\boxtimes}+1\}\} \cup \mathcal{M}ul(M) \cup \mathcal{M}ul(P) & \gg & \{\{\|M\|_{\boxtimes}\}\} \cup \mathcal{M}ul(M) \cup \mathcal{M}ul(P) \\
\{\{\|M\|_{\boxtimes}+\|P\|_{\boxtimes}+1\}\} \cup \mathcal{M}ul(M) \cup \mathcal{M}ul(P) & \gg & \{\{\|P\|_{\boxtimes}\}\} \cup \mathcal{M}ul(M) \cup \mathcal{M}ul(P) \\
\{\{\|M\|_{\boxtimes}+1\}\} \cup \mathcal{M}ul(M) & \gg & \{\{\|M\|_{\boxtimes}\}\} \cup \mathcal{M}ul(M) \\
\{\{\|M\|_{\boxtimes}+1\}\} \cup \mathcal{M}ul(M) & \gg & \mathcal{M}ul(M) \\
\{\{\|M\|_{\boxtimes}+1\}\} \cup \mathcal{M}ul(M) & \gg & \{\{\|M\|_{\boxtimes}\}\} \cup \mathcal{M}ul(M) \\
\{\{\|M\|_{\boxtimes}+1\}\} \cup \mathcal{M}ul(M) & \gg & \{\{\|M\|_{\boxtimes},\|M\|_{\boxtimes}\}\} \cup \mathcal{M}ul(M) \cup \mathcal{M}ul(M)
\end{array}$$

Two inequalities require discussion. The first comes from $x[N/x] \xrightarrow{\boxtimes} N$ and is satisfied because N is substitution free, therefore $\mathcal{M}ul(N) = \{\{\ \}\}$. The second comes from $(x <^{x_1}_{x_2} M)[N/x] \xrightarrow{\boxtimes} Fv(N) <^{Fv(N_1)}_{Fv(N_2)} M[N_1/x_1][N_2/x_2]$ and is satisfied because $\|x <^{x_1}_{x_2} M\|_{\boxtimes} = \|M\|_{\boxtimes}+1$ is larger than $\|M\|_{\boxtimes}$ and than any $\|P\|_{\boxtimes}$ for P subterm of M.

This shows that $\xrightarrow{\boxtimes}\!\!\!\!\rightarrow$ is well-founded, hence that $\xrightarrow{\boxtimes}\!\!\!\!\rightarrow$ terminates. □

Proposition 11. *The reduction* $\xrightarrow{\boxtimes}\!\!\!\!\rightarrow$ *is confluent.*

Proof. There is no superposition between the left-hand sides of the rules of Figure 5, therefore there is no critical pair. Hence, the rewrite system is locally confluent. According to Proposition 10 it terminates, hence it is confluent by Newman's Lemma [3]. □

Definition 12 ($\xrightarrow{\boxtimes}$ Normal forms)**.** *Starting from M and reducing by* $\xrightarrow{\boxtimes}\!\!\!\!\rightarrow$*, the irreducible term we obtain is called the* $\xrightarrow{\boxtimes}$-normal form *of M and denoted by* $M\downarrow^{\boxtimes}$.

Every $\lambda^{\boxtimes}_{\circledR}$-term has a unique normal form, the existence is guaranteed by Proposition 10, whereas the uniqueness is a consequence of confluence (Proposition 11).

Proposition 13. *If* $Q \in \Lambda_{\circledR}$ *then* $Q[N/x]\downarrow^{\boxtimes} \in \Lambda_{\circledR}$.

Proof. Let us look at all the terms of the form $Q[N/x]$ and their evaluation by the rules in Figure 5.

- Q is a *variable*. Due to rule (sub) in Figure 3, $x \in Fv^{\boxtimes}(Q)$, hence Q must be x. Therefore, all the cases when Q is a variable are exhausted.
- Q is an *abstraction*, then one rule is enough.
- Q is an *application* MP, then either $x \in Fv^{\boxtimes}(M)$ or $x \in Fv^{\boxtimes}(P)$, hence the two rules exhaust this case.
- Q is an *erasure* $y \odot M$, then either $y = x$ or $y \neq x$ and the two cases are considered.
- Q is a *duplication* $x <^{x_1}_{x_2} M$, then again either $y = x$ or $y \neq x$ and the two cases are considered.

□

Proposition 14. *If* $M \in \Lambda^{\boxtimes}_{\circledR}$ *then* $M \downarrow^{\boxtimes} \in \Lambda_{\circledR}$.

Proof. By induction on the number of substitutions in M, Proposition 13 being the base case. □

The substitution of n different variables in the same term is denoted by

$$M[N_1/x_1]...[N_n/x_n] \downarrow^{\boxtimes} .$$

These substitutions are actually performed in "parallel" since we prove that they commute in the following proposition.

Proposition 15. *If* $M \in \Lambda_{\circledR}$ *and* $x_i \in Fv(M)$ *for* $i \in \{1,...,n\}$, $n \geq 1$ *with* $x_i \neq x_j$ *for* $i \neq j$, *then*

$$M[N_1/x_1]...[N_n/x_n] \downarrow^{\boxtimes} = M[N_{p(1)}/x_{p(1)}]...[N_{p(n)}/x_{p(n)}] \downarrow^{\boxtimes},$$

where $(p(1),...,p(n))$ *is a permutation of* $(1,...,n)$.

Proof. We prove the proposition by induction on the structure of M,

- For $M = x_1$ the statement holds since the only permutation is the identity, namely, $p(1) = 1$, therefore $x_1[N_1/x_1] \downarrow^{\boxtimes} = N_1 = x_1[N_{p(1)}/x_{p(1)}] \downarrow^{\boxtimes}$.
- If $M = \lambda y.Q$ then this works by induction. Notice that $y \neq x_i$, for $i \in \{1,...,n\}$.

- If $M = QR$ then we distinguish two cases:
  - some of $\{x_1,...,x_n\}$ belong to $Fv(Q)$, whereas the others belong to $Fv(R)$. Without loss of generality we can assume that for some k such that $1 \leq k < n$, $\{x_1,...,x_k\} \in Fv(Q)$ and $\{x_{k+1},...,x_n\} \in Fv(R)$. Then $(QR)[N_1/x_1]...[N_n/x_n]$ reduces to $Q[N_1/x_1]...[N_k/x_k] \downarrow^{\boxtimes} R[N_{k+1}/x_{k+1}]...[N_n/x_n] \downarrow^{\boxtimes}$, and the result follows by two applications of induction hypothesis.
  - If $M = QR$ and $\{x_1,...,x_n\}$ all belong to either $Fv(Q)$ or to $Fv(R)$, the result follows by induction.
- If $M = y \odot Q$ with $y \neq x_i$ for $i \in \{1,...,n\}$, then the result follows by induction.
- If $M = x_j \odot Q$ then $(x_j \odot Q)[N_1/x_1]...[N_j/x_j]...[N_n/x_n]$ reduces to $Fv(N_j) \odot Q[N_1/x_1]...[N_{j-1}/x_{j-1}][N_{j+1}/x_{j+1}]...[N_n/x_n] \downarrow^{\boxtimes}$.
  On the other hand, given an arbitrary permutation p, let us call k the index such that $p(k) = j$. Then, $(x_j \odot Q)[N_{p(1)}/x_{p(1)}]...[N_{p(k)}/x_{p(k)}]...[N_{p(n)}/x_{p(n)}]$ reduces to $Fv(N_{p(k)}) \odot Q[N_{p(1)}/x_{p(1)}]...[N_{p(k)-1}/x_{p(k)-1}][N_{p(k)+1}/x_{p(k)+1}]...[N_{p(n)}/x_{p(n)}] \downarrow^{\boxtimes}$.
  Since $N_j = N_{p(k)}$ then $Fv(N_j) = Fv(N_{p(k)})$ and the result follows by induction.
- If $M = y <_{y_2}^{y_1} Q$ where $y \neq x_i$ for $i \in \{1,...,n\}$, then the result follows by induction.
- If $M = x_j <_{x''_j}^{x'_j} Q$ then $(x_j <_{x''_j}^{x'_j} Q)[N_1/x_1]...[N_j/x_j]...[N_n/x_n]$ reduces to
  $Fv(N_j) <_{Fv(N''_j)}^{Fv(N'_j)} Q[N_1/x_1]...[N'_j/x'_j][N''_j/x''_j]...[N_n/x_n] \equiv M_1$.
  On the other hand, given an arbitrary permutation p, let us call k the index such that $p(k) = j$. We have that
  $(x_{p(k)} <_{x''_{p(k)}}^{x'_{p(k)}} Q)[N_{p(1)}/x_{p(1)}]...[N_{p(k)}/x_{p(k)}]...[N_{p(n)}/x_{p(n)}]$ reduces to
  $Fv(N_k) <_{Fv(N''_k)}^{Fv(N'_k)} Q[N_{p(1)}/x_{p(1)}]...[N'_{p(k)}/x'_{p(k)}][N''_{p(k)}/x''_{p(k)}]...[N_{p(n)}/x_{p(n)}] \equiv M_2$. By induction hypothesis (recall that $j = p(k)$),
  $Q[N_1/x_1]...[N'_j/x'_j][N''_j/x''_j]...[N_n/x_n]$ and
  $Q[N_{p(1)}/x_{p(1)}]...[N'_{p(k)}/x'_{p(k)}][N''_{p(k)}/x''_{p(k)}]...[N_{p(n)}/x_{p(n)}]$
  have the same normal forms, therefore $M_1 \downarrow^{\boxtimes} = M_2 \downarrow^{\boxtimes}$.

□

Finally, we can formally define *substitution* in $\Lambda_{\mathbb{R}}$ and *simultaneous substitution* in $\Lambda_{\mathbb{R}}$ via $\lambda^{\boxtimes}_{\mathbb{R}}$-normal forms.

Definition 16 (Substitution in $\Lambda_{\mathbb{R}}$)**.** *If $M \in \Lambda_{\mathbb{R}}$ and $N \in \Lambda_{\mathbb{R}}$ then*

$$M[\![N/\!\!/x]\!] \triangleq M[N/x]\downarrow^{\boxtimes}.$$

Notice that $M[\![N/\!\!/x]\!]$ is well-defined, since $M[\![N/\!\!/x]\!] \in \Lambda_{\mathbb{R}}$, due to Proposition 14. Moreover, Proposition 15 allows us to give simply a meaning to simultaneous substitution.

Definition 17 (Simultaneous substitution in $\Lambda_{\mathbb{R}}$)**.** *Simultaneous substitution in $\Lambda_{\mathbb{R}}$ is defined as follows:*

$$M[\![N_1/\!\!/x_1,\ldots,N_p/\!\!/x_p]\!] = M[\![N_1/\!\!/x_1]\!]\ldots[\![N_p/\!\!/x_p]\!].$$

provided that $Fv(N_i)\cap Fv(N_j) = \emptyset$ for $i \neq j$.

1.3 Operational semantics

The operational semantics of $\lambda_{\mathbb{R}}$ is defined by a *reduction relation* $\rightarrow$, given by the set of reduction rules in Figure 6. In the $\lambda_{\mathbb{R}}$-calculus, one works modulo the *structural equivalence* $\equiv_{\lambda_{\mathbb{R}}}$, defined as the smallest equivalence that satisfies the axioms given in Figure 7 and is closed under α-conversion. The reduction relation $\rightarrow$ is closed under $\equiv_{\lambda_{\mathbb{R}}}$ and contexts. Its reflexive, transitive closure will be denoted by $\twoheadrightarrow$. As usual, a term is a *redex* if it has the form of a term on the left-hand side of a rule in Figure 6, whereas its *contractum* is the term on the right-hand side of the same rule.

The reduction rules are divided into four groups. The main computational step is β-reduction. The group of (γ) reductions perform propagation of duplications into the expression. Similarly, (ω) reductions extract erasures out of expressions. This discipline allows us to optimise the computation by delaying duplication of terms on the one hand, and by performing erasure of terms as soon as possible on the other. Finally, the rules in the $(\gamma\omega)$ group explain the interaction between the explicit resource operators that are of different nature. Notice that in the rule $(\gamma\omega_2)$ the substitution in $\Lambda_{\mathbb{R}}$ is actually a syntactic variable replacement, i.e., renaming.[2]

[2]We decided to use the same notation in order to introduce less different notations.

$$
\begin{array}{rrcl}
(\beta) & (\lambda x.M)N & \to & M[\![N/\!/x]\!] \\
(\gamma_1) & x <^{x_1}_{x_2} (\lambda y.M) & \to & \lambda y.x <^{x_1}_{x_2} M \\
(\gamma_2) & x <^{x_1}_{x_2} (MN) & \to & (x <^{x_1}_{x_2} M)N, \text{ if } x_1, x_2 \notin Fv(N) \\
(\gamma_3) & x <^{x_1}_{x_2} (MN) & \to & M(x <^{x_1}_{x_2} N), \text{ if } x_1, x_2 \notin Fv(M) \\
(\omega_1) & \lambda x.(y \odot M) & \to & y \odot (\lambda x.M),\ x \neq y \\
(\omega_2) & (x \odot M)N & \to & x \odot (MN) \\
(\omega_3) & M(x \odot N) & \to & x \odot (MN) \\
(\gamma\omega_1) & x <^{x_1}_{x_2} (y \odot M) & \to & y \odot (x <^{x_1}_{x_2} M),\ y \neq x_1, x_2 \\
(\gamma\omega_2) & x <^{x_1}_{x_2} (x_1 \odot M) & \to & M[\![x/\!/x_2]\!]
\end{array}
$$

Figure 6: Reduction rules

$$
\begin{array}{rrcl}
(\varepsilon_1) & x \odot (y \odot M) & \equiv_{\lambda_{\circledR}} & y \odot (x \odot M) \\
(\varepsilon_2) & x <^{x_1}_{x_2} M & \equiv_{\lambda_{\circledR}} & x <^{x_2}_{x_1} M \\
(\varepsilon_3) & x <^{y}_{z} (y <^{u}_{v} M) & \equiv_{\lambda_{\circledR}} & x <^{y}_{u} (y <^{z}_{v} M) \\
(\varepsilon_4) & x <^{x_1}_{x_2} (y <^{y_1}_{y_2} M) & \equiv_{\lambda_{\circledR}} & y <^{y_1}_{y_2} (x <^{x_1}_{x_2} M),\ x \neq y_1, y_2,\ y \neq x_1, x_2
\end{array}
$$

Figure 7: Structural equivalence

Proposition 18 (Soundness of $\twoheadrightarrow$)**.**

- *For all terms M and N such that $M \to N$, if $M \in \Lambda_{\circledR}$, then $N \in \Lambda_{\circledR}$.*
- *For all terms M and N such that $M \twoheadrightarrow N$, if $M \in \Lambda_{\circledR}$, then $N \in \Lambda_{\circledR}$.*

In particular, in the case of (β)-reduction if $(\lambda x.M)N \in \Lambda_{\circledR}$, then

$$M[\![N/\!/x]\!] = M[N/x]\downarrow^{\square} \in \Lambda_{\circledR}$$

by Proposition 14.

No variable is lost during the computation, which is stated by the following proposition.

Proposition 19 (Preservation of free variables by $\twoheadrightarrow$)**.**
If $M \twoheadrightarrow N$ then $Fv(M) = Fv(N)$.

Proof. The proof is by case analysis on the reduction rules and uses Proposition 8 (ii). □

First, let us observe the structure of the $\lambda_{\mathbb{R}}$-normal forms, given by the following abstract syntax. As usually, a term is a normal for if it does not have any redex as subterm.

Definition 20 (Set of Normal Forms)**.** *The set $\mathcal{NF}$ of normals forms is generated by the following abstract syntax:*

$$\begin{array}{rcl} M_{nf} & ::= & \lambda x.M_{nf} \,|\, \lambda x.x \odot M_{nf} \,|\, xM^1_{nf} \ldots M^n_{nf} \,|\, x <^{x_1}_{x_2} M_{nf} \\ & & \qquad \textit{in the last case } M_{nf} \equiv P_{nf}Q_{nf},\ x_1 \in Fv(P_{nf}),\ x_2 \in Fv(Q_{nf}) \\ E_{nf} & ::= & x \odot M_{nf} \,|\, x \odot E_{nf} \end{array}$$

where $n \geq 0$. It is necessary to distinguish normal forms E_{nf} separately because the term $\lambda x.y \odot M_{nf}$ is not a normal form, since $\lambda x.y \odot M_{nf} \to_{\omega_1} y \odot \lambda x.M_{nf}$. Also, in the last case the term $x <^{x_1}_{x_2} P_{nf}Q_{nf}$, where $x_1 \in Fv(P_{nf}), x_2 \in Fv(Q_{nf})$ is not necessarily a normal form since $P_{nf}Q_{nf}$ can be a redex, in turn $M_{nf} \equiv P_{nf}Q_{nf}$ guarantees that the application is a normal form.

Next we define the set of strongly normalising terms $\mathcal{SN}$.

Definition 21 (Strongly normalising terms)**.** *The set of strongly normalising terms $\mathcal{SN}$ is defined as follows:*

$$\frac{M \in \mathcal{NF}}{M \in \mathcal{SN}} \qquad \frac{\forall N \in \Lambda_{\mathbb{R}} \,.\, M \twoheadrightarrow N \;\Rightarrow\; N \in \mathcal{SN}}{M \in \mathcal{SN}}$$

Lemma 22. *Every term has one of the following forms, where $n \geq 0$:*

$$\begin{array}{llll} (\mathsf{Abs}) & \lambda x.N, & (\mathsf{AbsApp}) & (\lambda x.N)PT_1 \ldots T_n \\ (\mathsf{Var}) & xT_1 \ldots T_n & (\mathsf{DupApp}) & (x <^{x_1}_{x_2} N)T_1 \ldots T_n \\ (\mathsf{Era}) & x \odot N & (\mathsf{EraApp}) & (x \odot N)PT_1 \ldots T_n \end{array}$$

Proof. These terms are well-formed according to Definition 1 (we did not explicitly write the conditions, since we work with linear terms). The proof is by induction on the structure of the term $M \in \Lambda_{\mathbb{R}}$.

- If M is a variable, this case is covered by Var for $n = 0$.
- If M is an abstraction $\lambda x.Q$, then by induction Q has one of the given forms, hence $\lambda x.Q$ is covered by Abs.
- If M is an application then M is of the form $M \equiv QP_1 \ldots P_n$, for $n \geq 1$ and Q is not an application. We proceed by subinduction on the structure of Q. Accordingly, Q is one of the following:

– Q is a variable, then we have the case Var, with $n \geq 1$;

– Q is an abstraction, then we have the case AbsApp;

– Q is an erasure, then we have the case EraApp;

– Q is a duplication, then we have the case DupApp, with $n \geq 1$.

- If M is an erasure $x \odot Q$, then by induction Q has one of the given forms, hence $x \odot Q$ is covered by Era.

- If M is a duplication $x <^{x_1}_{x_2} Q$, then by induction Q has one of the given forms, hence $x <^{x_1}_{x_2} Q$ is covered by DupApp for $n = 0$.

□

2 Intersection types for $\lambda_{\circledR}$

In this section we introduce an intersection type assignment $\lambda_{\circledR}\cap$ system which assigns *strict types* to $\lambda_{\circledR}$-terms. Strict types were proposed in [53] and used in [20] for characterisation of strong normalisation in λ^{Gtz}-calculus.

The syntax of types is defined as follows:

$$\begin{array}{lllll} \text{Strict types} & \sigma & ::= & p \mid \alpha \to \sigma \\ \text{Types} & \alpha & ::= & \cap_i^n \sigma_i \end{array}$$

where p ranges over a denumerable set of type atoms and

$$\cap_i^n \sigma_i = \begin{cases} \sigma_1 \cap \ldots \cap \sigma_n & \text{for } n > 0 \\ \top & \text{for } n = 0 \end{cases}$$

$\top$ being the *neutral element* for the intersection operator, i.e. $\sigma \cap \top = \sigma$.

We denote types by $\alpha, \beta, \gamma \ldots$, strict types by $\sigma, \tau, \upsilon \ldots$ and the set of all types by Types. We assume that the intersection operator is commutative and associative. We also assume that intersection has priority over arrow. Hence, we will omit parenthesis in expressions like $(\cap_i^n \tau_i) \to \sigma$.

2.1 The type assignment system

Definition 23. *(i) A* basic type assignment (declaration) *is an expression of the form* $x : \alpha$*, where* x *is a term variable and* α *is a type.*

(ii) Consider a finite set $Dom(\Gamma)$ *of variables. A* basis *is a function*

$$\Gamma : Dom(\Gamma) \to \mathsf{Types}.$$

A basis extension *of* Γ *is a function* $\Gamma, x : \alpha : Dom(\Gamma) \cup \{x\} \to \mathsf{Types}$*:*

$$y \mapsto \begin{cases} \Gamma(y) & \text{if } y \in Dom(\Gamma) \\ \alpha & \text{if } y = x \end{cases}$$

(iii) Given Γ *and* Δ *such that* $Dom(\Gamma) = Dom(\Delta)$*, the* bases intersection *of* Γ *and* Δ *is the function* $\Gamma \sqcap \Delta : Dom(\Gamma) \to \mathsf{Types}$*, such that:*

$$\Gamma \sqcap \Delta(x) = \Gamma(x) \cap \Delta(x).$$

(iv) $\Gamma^\top$ *is the constant function* $\Gamma^\top : Dom(\Gamma) \to \{\top\}$.

In what follows we assume that the bases intersection has priority over the basis extension, hence the parenthesis in $\Gamma, (\Delta_1 \sqcap \ldots \sqcap \Delta_n)$ will be omitted. It is easy to show that $\Gamma^\top \sqcap \Delta = \Delta$ for arbitrary bases Γ and Δ that can be intersected, hence $\Gamma^\top$ is the neutral element for the intersection of bases of domain $Dom(\Gamma)$.

$$\frac{}{x : \sigma \vdash x : \sigma}\ (Ax)$$

$$\frac{\Gamma, x : \alpha \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \alpha \to \sigma}\ (\to_I) \qquad \frac{\Gamma \vdash M : \cap_i^n \tau_i \to \sigma \quad \Delta_0 \vdash N : \tau_0 \ \ldots \ \Delta_n \vdash N : \tau_n}{\Gamma, \Delta_0^\top \sqcap \Delta_1 \sqcap \ldots \sqcap \Delta_n \vdash MN : \sigma}\ (\to_E)$$

$$\frac{\Gamma, x : \alpha, y : \beta \vdash M : \sigma}{\Gamma, z : \alpha \cap \beta \vdash z <_y^x M : \sigma}\ (Cont) \qquad \frac{\Gamma \vdash M : \sigma}{\Gamma, x : \top \vdash x \odot M : \sigma}\ (Thin)$$

Figure 8: $\lambda_{\circledR}\cap$: $\lambda_{\circledR}$-calculus with intersection types

The type assignment system $\lambda_{\circledR}\cap$ is given in Figure 8. It is syntax directed and the rules are context-splitting. The axiom (Ax) ensures that void λ-abstraction cannot be typed, i.e. in a typeable term each free variable appears at least once. The context-splitting rule $(\to_E)$ ensures that in a typeable term each free variable appears not more than once.

Assume that we implement these properties in the type system with (Ax), $(\to_E)$ and $(\to_I)$, then the combinators $K = \lambda xy.x$ and $W^{-1} = \lambda xy.xyy$ would not

be typeable. This motivates and justifies the introduction of the operators of erasure and duplication and the corresponding typing rules $(Thin)$ and $(Cont)$, which further maintain the explicit control of resources and enable the typing of K and W^{-1}, namely of their corresponding $\lambda_{\circledR}$-terms $\lambda xy.y \odot x$ and $\lambda xy.y <^{y_1}_{y_2} xy_1y_2$, respectively. Let us mention that on the logical side, structural rules of *thinning* and *contraction* are present in Gentzen's original formulation of *LJ*, Intuitionistic Sequent Calculus, but not in *NJ*, Intuitionistic Natural Deduction [22, 23]. Here instead, the presence of the typing rules $(Thin)$ and $(Cont)$ completely maintains the explicit control of resources in $\lambda_{\circledR}$.

In the proposed system, intersection types occur only in two inference rules. In the rule $(Cont)$ the intersection type is created, this being *the only* place where this happens. This is justified because it corresponds to the duplication of a variable. In other words, the control of the duplication of variables entails the control of the introduction of intersections in building the type of the term in question. In the rule $(\rightarrow_E)$, intersection appears on the right hand side of the turnstyle $\vdash$ which corresponds to the usage of the intersection type after it has been created by the rule $(Cont)$ or by the rule $(Thin)$ if $n = 0$.

The role of Δ_0 in the rule $(\rightarrow_E)$ should be emphasized. It is needed only when $n = 0$ to ensure that N has a type, i.e. that N is strongly normalising as would be seen below. Then, in the conclusion of the rule, the types of the free variables of N can be forgotten, hence all the free variables of N receive the type $\top$. All the free variables of the term must occur in the environment Γ (see Lemma 28), therefore useless variables occur with the type $\top$. When $n > 0$, Δ_0 can be any of the other environments and the type of N the associated type. Since $\Delta^{\top}$ is a neutral element for $\sqcap$, when $n > 0$, $\Delta^{\top}$ disappears in the conclusion of the rule. The case $n = 0$ resembles the rules $(drop)$ and/or *(K-cup)* in [38] and was used to present the two cases, $n = 0$ and $n \neq 0$ in a uniform way. In the rule $(Thin)$ the choice of the type of x is $\top$, since this corresponds to a variable which does not occur anywhere in M. The remaining rules, namely (Ax) and $(\rightarrow_I)$ are traditional, i.e. they are the same as in the simply typed λ-calculus. Notice however that the type of the variable in (Ax) is a strict type.

Roles of the variables

In the syntax of $\lambda_{\circledR}$, there are three kinds of variables according to the way they are introduced, namely as a placeholder (associated with the typing rule *(Ax)*), as the result of a duplication (associated with the typing rule *(Cont)*) or as the result of an erasure (associated with the typing rule *(Thin)*). Each kind of variable

receives a specific type:

- variables as placeholders have a strict type,
- variables resulting from a duplication have an intersection type,
- variables resulting from an erasure have the type $\top$.

In order to emphasize the sensitivity of the system $\lambda_{\circledR\cap}$ w.r.t. the role of a variable in a term, we provide the following examples in which variables change their role during the computation process. Our goal is to show that when the role of a variable changes, its type in the type derivation changes as well, so that the correspondence between particular roles and types is preserved.

Example 24. *A variable as a "placeholder" becomes an "erased" variable: this is the case with the variable z in $(\lambda x.x \odot y)z$, because*

$$(\lambda x.x \odot y)z \to_\beta (x \odot y)[z/\!\!/x] \triangleq (x \odot y)[z/x]\downarrow^{\boxslash} = z \odot y.$$

Since $z : \top, y : \sigma \vdash z \odot y : \sigma$, we want to show that $z : \top, y : \sigma \vdash (\lambda x.x \odot y)z : \sigma$. Indeed:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{y : \sigma \vdash y : \sigma}\,(Ax)}{x : \top, y : \sigma \vdash x \odot y : \sigma}\,(Weak)}{y : \sigma \vdash \lambda x.x \odot y : \top \to \sigma}\,(\to_I) \qquad \dfrac{}{z : \tau \vdash z : \tau}\,(Ax)}{z : \top, y : \sigma \vdash (\lambda x.x \odot y)z : \sigma}\,(\to_E).$$

In the rule $(\to_E)$, we have $n = 0$, $\Delta_0 = z : \tau$ and $\Delta_0^\top = z : \top$. Thus, in the previous derivation, the variable z changed its type from a strict type to $\top$, in accordance with the change of its role in the bigger term.

Example 25. *A variable as a "placeholder" becomes a "duplicated" variable: this is the case with the variable v in $(\lambda x.x <_z^y yz)v$, because*

$$\begin{aligned}(\lambda x.x <_z^y yz)v \to_\beta (x <_z^y yz)[v/\!\!/x] \triangleq (x <_z^y yz)[v/x]\downarrow^{\boxslash} = \\ = Fv[v] <_{Fv[v_2]}^{Fv[v_1]} (yz)[v_1/y][v_2/z]\downarrow^{\boxslash} = v <_{v_2}^{v_1} v_1 v_2.\end{aligned}$$

Since $v : (\tau \to \sigma) \cap \tau \vdash v <_{v_2}^{v_1} v_1 v_2 : \sigma$, we want to show that $v : (\tau \to \sigma) \cap \tau \vdash (\lambda x.x <_z^y yz)v : \sigma$.

Indeed:

$$\dfrac{\dfrac{\vdots}{\vdash \lambda x.x <_z^y yz : ((\tau\to\sigma)\cap\tau)\to\sigma}(\to_I) \quad \dfrac{}{v:\tau\vdash v:\tau}(Ax) \quad \dfrac{}{v:\tau\to\sigma\vdash v:\tau\to\sigma}(Ax)}{v:(\tau\to\sigma)\cap\tau\vdash(\lambda x.x <_z^y yz)v:\sigma}(\to_E).$$

In the rule $(\to_E)$, we have $n=2$, therefore $\Delta_0\vdash N:\tau_0$ can be one of the two existing typing judgements, for instance $v:\tau\vdash v:\tau$. In this case $\Delta_0^\top$ disappears in the conclusion, because
$\Delta_0^\top\sqcap\Delta_1\sqcap\Delta_2 = v:\top\sqcap v:\tau\to\sigma\sqcap v:\tau = v:\top\cap(\tau\to\sigma)\cap\tau = v:(\tau\to\sigma)\cap\tau$. *Again, we see that the type of the variable v changed from strict type to (intersection) type.*

Example 26. *A "duplicated" variable becomes an "erased" variable: this is the case with the variable z in $(\lambda x.x\odot y)(z <_v^u uv)$, because*

$$\begin{aligned}(\lambda x.x\odot y)(z<_v^u uv) \to_\beta (x\odot y)\llbracket z<_v^u uv/\!\!/x\rrbracket &\triangleq (x\odot y)[z<_v^u uv/x]\downarrow^{\not\square} = \\ &= Fv(z<_v^u uv)\odot y = z\odot y.\end{aligned}$$

Like in the previous examples, both $z:\top, y:\sigma\vdash z\odot y:\sigma$ and $z:\top, y:\sigma\vdash(\lambda x.x\odot y)(z<_v^u uv):\sigma$ can be shown.

Example 27. *An "erased" variable becomes a "duplicated" variable: this is the case with the variable u in $(\lambda x.x <_z^y yz)(u\odot v)$, because*

$$\begin{aligned}(\lambda x.x<_z^y yz)(u\odot v) \to_\beta\ & (x<_z^y yz)\llbracket u\odot v/\!\!/x\rrbracket \\ \triangleq\ & (x<_z^y yz)[u\odot v/x]\downarrow^{\not\square} \\ =\ & Fv[u\odot v]<^{Fv[u_1\odot v_1]}_{Fv[u_2\odot v_2]}(yz)[u_1\odot v_1/y][u_2\odot v_2/z]\downarrow^{\not\square} \\ =\ & u<_{u_2}^{u_1} v<_{v_2}^{v_1}(u_1\odot v_1)(u_2\odot v_2).\end{aligned}$$

The situation here is slightly different. Fresh variables u_1 and u_2 are obtained from u using the substitution in $\Lambda_{\circledR}$. The variable u is introduced by thinning, so its type is $\top$. Substitution in $\Lambda_{\circledR}$ does not change the types, therefore both u_1 and u_2 have the type $\top$. Finally, u in the resulting term is obtained by contracting u_1 and u_2, therefore its type is $\top\cap\top=\top$. Thus we have an interesting situation - the role of the variable u changes from "to be erased" to "to be duplicated", but its type remains $\top$.

However, this paradox (if any) is only apparent, as well as the change of the role. Unlike the previous three examples, in which we obtained normal forms, in

this case the computation can continue:

$$\begin{array}{lll} u <^{u_1}_{u_2} v <^{v_1}_{v_2} (u_1 \odot v_1)(u_2 \odot v_2) & \to_{(\omega_2+\varepsilon_4)} & v <^{v_1}_{v_2} u <^{u_1}_{u_2} u_1 \odot v_1(u_2 \odot v_2) \\ & \to_{\gamma\omega_2} & v <^{v_1}_{v_2} v_1((u_2 \odot v_2))[\![u/\!/u_2]\!] \\ & = & v <^{v_1}_{v_2} v_1(u \odot v_2). \end{array}$$

So, we see that the actual role of the variable u in the obtained normal form, is "to be erased", as indicated by its type $\top$.

To conclude the analysis, we point out the following key points:

- The type assignment system $\lambda_{\circledR}\cap$ is constructed in such way that the type of a variable always indicates its actual role in the term. Due to this, we claim that the system $\lambda_{\circledR}\cap$ fits naturally to the resource control calculus $\lambda_{\circledR}$.
- Switching between roles is not reversible: once a variable is meant to be erased, it cannot be turned back to some other role. Moreover, the information about its former role cannot be reconstructed from the type.

A note about idempotence and identity rule

Recall that the typing tree of a term is dictated by the syntax: $\to$ is introduced by $(\to_I)$, $\cap$ is introduced by $(Cont)$ and $\top$ is introduced by $(Thin)$. In this context it would not pertain to remove an intersection by idempotence or identity rule. This is why they are not considered here.

2.2 Structural properties

Lemma 28 (Domain correspondence for $\lambda_{\circledR}\cap$)**.** *Let* $\Gamma \vdash M : \sigma$ *be a typing judgment. Then* $x \in Dom(\Gamma)$ *if and only if* $x \in Fv(M)$.

Proof. The rules of Figure 8 belong to three categories.

1. *The rules that introduce a variable*. These rules are *(Ax)*, $(Cont)$ and $(Thin)$. One sees that the variable is introduced in the environment if and only it is introduced in the term as a free variable.
2. *The rules that remove variables*. These rules are $(\to_I)$ and $(Cont)$. One sees that the variables are removed from the environment if and only if they are removed from the term as a free variable.

3. *The rule that neither introduces nor removes a variable.* This rule is $(\to_E)$.

Notice that $(Cont)$ introduces and removes variables. □

The Generation Lemma makes somewhat more precise the Domain Correspondence Lemma.

Lemma 29 (Generation lemma for $\lambda_{\circledR}\cap$)**.**

(i) $\Gamma \vdash \lambda x.M : \tau$ *iff there exist* α *and* σ *such that* $\tau \equiv \alpha \to \sigma$ *and* $\Gamma, x : \alpha \vdash M : \sigma$.

(ii) $\Gamma \vdash MN : \sigma$ *iff and there exist* Δ_i *and* τ_i, $i \in \{0,\ldots,n\}$ *such that* $\Gamma' \vdash M : \cap_i^n \tau_i \to \sigma$ *and for all* $i \in \{0,\ldots,n\}$, $\Delta_i \vdash N : \tau_i$ *and* $\Gamma = \Gamma', \Delta_0^\top \sqcap \Delta_1 \sqcap \ldots \sqcap \Delta_n$.

(iii) $\Gamma \vdash z <_y^x M : \sigma$ *iff there exist* Γ', α, β *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* $\Gamma = \Gamma', x : \top$ *and* $\Gamma' \vdash M : \sigma$.

Proof. The proof is straightforward since all the rules are syntax directed, and relies on Lemma 28. □

In the sequel, we prove that the proposed system satisfies the following properties: Substitution lemma for $\lambda_{\circledR}\cap$ (Proposition 35) and Subject reduction and equivalence (Proposition 36).

In order to prove the Substitution lemma we extend the type assignment system $\lambda_{\circledR}\cap$ with a new rule for typing the substitution operator, thus obtaining an auxiliary system $\lambda_{\circledR}^{\boxslash}\cap$ that assigns types to $\lambda_{\circledR}^{\boxslash}$-terms.

Definition 30. *(i) The type assignment system* $\lambda_{\circledR}^{\boxslash}\cap$ *consists of rules from Figure 8 plus the following* $(Subst)$ *rule:*

$$\frac{\Gamma, x : \cap_i^n \tau_i \vdash^{\boxslash} M : \sigma \quad \Delta_0 \vdash N : \tau_0 \quad \ldots \quad \Delta_n \vdash N : \tau_n}{\Gamma, \Delta_0^\top \sqcap \Delta_1 \sqcap \ldots \sqcap \Delta_n \vdash^{\boxslash} M[N/x] : \sigma} \ (Subst)$$

(ii) Typing judgements in the system $\lambda_{\circledR}^{\boxslash}\cap$ *are denoted by* $\Gamma \vdash^{\boxslash} M : \sigma$.

The system $\lambda^{\boxtimes}_{\circledR}\cap$ is also syntax-directed, and assigns strict types to $\lambda^{\boxtimes}_{\circledR}$-terms. Therefore, it represents *a conservative extension* of the system $\lambda_{\circledR}\cap$, meaning that if $\Gamma \vdash^{\boxtimes} M : \sigma$ and $M \in \Lambda_{\circledR}$ (i.e. M is substitution-free), then $\Gamma \vdash M : \sigma$ and the two derivations coincide.

It is easy to adapt Lemma 28 and Lemma 29 to prove the corresponding properties of the system $\lambda^{\boxtimes}_{\circledR}\cap$.

Lemma 31 (Domain correspondence for $\lambda^{\boxtimes}_{\circledR}\cap$)**.** *Let $\Gamma \vdash^{\boxtimes} M : \sigma$ be a typing judgment. Then $x \in Dom(\Gamma)$ if and only if $x \in Fv^{\boxtimes}(M)$.*

Proof. The proof is the same as the proof of Lemma 28, having in mind the definition of $Fv^{\boxtimes}(M)$ and the fact that the rule $(Subst)$ belongs to the category of rules that remove variables. □

Lemma 32 (Generation lemma for $\lambda^{\boxtimes}_{\circledR}\cap$)**.** *(i)* $\Gamma \vdash^{\boxtimes} \lambda x.M : \tau$ *iff there exist* α *and* σ *such that* $\tau \equiv \alpha \to \sigma$ *and* $\Gamma, x : \alpha \vdash^{\boxtimes} M : \sigma$.

(ii) $\Gamma \vdash^{\boxtimes} MN : \sigma$ *iff there exist* Δ_i *and* τ_i, $i = 0, \ldots, n$ *such that* $\Gamma' \vdash^{\boxtimes} M : \cap_i^n \tau_i \to \sigma$ *and for all* $i \in \{0, \ldots, n\}$, $\Delta_i \vdash^{\boxtimes} N : \tau_i$ *and* $\Gamma = \Gamma', \Delta_0^{\top} \sqcap \Delta_1 \sqcap \ldots \sqcap \Delta_n$.

(iii) $\Gamma \vdash^{\boxtimes} z <_y^x M : \sigma$ *iff there exist* Γ', α, β *such that* $\Gamma = \Gamma', z : \alpha \cap \beta$ *and* $\Gamma', x : \alpha, y : \beta \vdash^{\boxtimes} M : \sigma$.

(iv) $\Gamma \vdash^{\boxtimes} x \odot M : \sigma$ *iff* $\Gamma = \Gamma', x : \top$ *and* $\Gamma' \vdash^{\boxtimes} M : \sigma$.

(v) $\Gamma \vdash^{\boxtimes} M[N/x] : \sigma$ *iff there exist* Δ_i *and* τ_i, $i = 0, \ldots, n$ *such that* $\Gamma', x : \cap_i^n \tau_i \vdash^{\boxtimes} M : \sigma$ *and for all* $i \in \{0, \ldots, n\}$, $\Delta_i \vdash N : \tau_i$ *and* $\Gamma = \Gamma', \Delta_0^{\top} \sqcap \Delta_1 \sqcap \ldots \sqcap \Delta_n$.

Proof. The proof is straightforward since all the rules are syntax directed, and relies on Lemma 31. □

To prove Lemma 34 we will need the definition of contexts.

Definition 33 ($\lambda^{\boxtimes}_{\circledR}$-Contexts)**.**

$$\mathcal{C} ::= [\,] \mid \lambda x.\mathcal{C} \mid M\mathcal{C} \mid \mathcal{C}M \mid x \odot \mathcal{C} \mid x <_{x_2}^{x_1} \mathcal{C} \mid \mathcal{C}[N/x]$$

Lemma 34 (Type preservation under $\xrightarrow{\boxtimes}\!\!\!\!\!\rightarrow\!\!\!\!\!\rightarrow$)**.**

(i) For all $M, M' \in \Lambda^{\boxslash}_{\circledR}$, $N \in \Lambda_{\circledR}$, *if* $\Gamma \vdash^{\boxslash} M[N/x] : \sigma$ *and* $M[N/x] \xrightarrow{\boxslash}\!\!\!\!\twoheadrightarrow M'$, *then* $\Gamma \vdash^{\boxslash} M' : \sigma$.

(ii) For all $M, M' \in \Lambda^{\boxslash}_{\circledR}$, $N \in \Lambda_{\circledR}$, *if* $\Gamma \vdash^{\boxslash} \mathcal{C}[M[N/x]] : \sigma$ *and* $\mathcal{C}[M[N/x]] \xrightarrow{\boxslash}\!\!\!\!\twoheadrightarrow \mathcal{C}[M']$, *then* $\Gamma \vdash^{\boxslash} \mathcal{C}[M'] : \sigma$.

Proof. (i) The proof is by case analysis on $\xrightarrow{\boxslash}$ (Figure 5). We consider only some representative rules. The other rules are routine and their proofs are analogous to the second rule we consider.

- Rule $x[N/x] \xrightarrow{\boxslash} N$. In this case $n = 1$ and Γ is empty. Recall that $\Delta^{\top} \sqcap \Delta = \Delta$. On one hand we have

$$\dfrac{\dfrac{}{x : \tau \vdash^{\boxslash} x : \tau}\,(Ax) \qquad \Delta \vdash^{\boxslash} N : \tau \quad \Delta \vdash^{\boxslash} N : \tau}{\Delta \vdash^{\boxslash} x[N/x] : \tau}\,(Subst)$$

  and on the other hand we have

$$\Delta \vdash^{\boxslash} N : \tau$$

  by assumption.

- Rule $(MP)[N/x] \xrightarrow{\boxslash} M[N/x]P$, $x \in Fv^{\boxslash}(M)$. On one hand we have:

$$\dfrac{\dfrac{\Gamma, x : \cap_i^n \upsilon_i \vdash^{\boxslash} M : \cap_i^m \rho_i \to \sigma \quad \Theta_0 \vdash^{\boxslash} P : \rho_0 \ldots\ \Theta_m \vdash^{\boxslash} P : \rho_m}{\Gamma, x : \cap_i^n \upsilon_i, \Theta_0^{\top} \sqcap \Theta_1 \sqcap \ldots \sqcap \Theta_m \vdash^{\boxslash} M\,P : \sigma}\,{\to_E} \qquad \Delta_0 \vdash^{\boxslash} N : \tau_0 \quad \ldots \quad \Delta_n \vdash^{\boxslash} N : \tau_n}{\Gamma, \Theta_0^{\top} \sqcap \Theta_1 \sqcap \ldots \sqcap \Theta_m, \Delta_0^{\top} \sqcap \Delta_1 \sqcap \ldots \sqcap \Delta_n \vdash^{\boxslash} (M\,P)[N/x] : \sigma}\,(Subst)$$

  One the other hand we have:

$$\dfrac{\dfrac{\Gamma, x : \cap_i^n \upsilon_i \vdash^{\boxslash} M : \cap_i^m \rho_i \to \sigma \quad \Delta_0 \vdash^{\boxslash} N : \tau_0 \quad \ldots \quad \Delta_n \vdash^{\boxslash} N : \tau_n}{\Gamma, \Delta_0^{\top} \sqcap \Delta_1 \sqcap \ldots \sqcap \Delta_n \vdash^{\boxslash} M[N/x] : \cap_i^m \rho_i \to \sigma}\,(Subst) \qquad \Theta_0 \vdash^{\boxslash} P : \rho_0 \ldots\ \Theta_m \vdash^{\boxslash} P : \rho_m}{\Gamma, \Theta_0^{\top} \sqcap \Theta_1 \sqcap \ldots \sqcap \Theta_m, \Delta_0^{\top} \sqcap \Delta_1 \sqcap \ldots \sqcap \Delta, n \vdash^{\boxslash} M[N/x]\,P : \sigma}\,({\to_E})$$

- Rule $(x \odot M)[N/x] \xrightarrow{\boxslash} Fv(N) \odot M$. In this case $n = 0$. On one hand we have:

$$\dfrac{\dfrac{\Gamma \vdash^{\boxslash} M : \sigma}{\Gamma, x : \top \vdash^{\boxslash} x \odot M}\,(Thin) \qquad \Delta_0 \vdash^{\boxslash} N : \tau_0}{\Gamma, \Delta_0^{\top} \vdash^{\boxslash} (x \odot M)[N/x] : \sigma}\,(Subst)$$

On the other hand we have:

$$\dfrac{\dfrac{\Gamma \vdash^{\not\square} M : \sigma}{\vdots}\,(Thin)}{\Gamma, \Delta_0^{\top} \vdash^{\not\square} Fv(N) \odot M : \sigma}\,(Thin)$$

- Rule $(x <_{x_2}^{x_1} M)[N/x] \xrightarrow{\not\square} Fv[N] <_{Fv[N_2]}^{Fv[N_1]} M[N_1/x_1][N_2/x_2]$. In order to make the proof tree readable, we adopt the following abbreviations:

$$\begin{array}{rcl}
\tau_1 & \triangleq & \cap_i^{n_1} \tau_{1,i} \\
\tau_2 & \triangleq & \cap_i^{n_2} \tau_{2,i} \\
\Delta_1 & \triangleq & \Delta_{1,1} \sqcap \ldots \sqcap \Delta_{1,n_1} \\
\Delta_2 & \triangleq & \Delta_{2,1} \sqcap \ldots \sqcap \Delta_{2,n_2} \\
\mathfrak{L}_1 & \triangleq & \Delta_{1,1} \vdash^{\not\square} N : \tau_{1,1} \ \ldots \ \Delta_{1,n_1} \vdash^{\not\square} N : \tau_{1,n_1} \\
\mathfrak{L}_2 & \triangleq & \Delta_{2,1} \vdash^{\not\square} N : \tau_{2,1} \ldots \Delta_{2,n_2} \vdash^{\not\square} N : \tau_{2,n_2}
\end{array}$$

Since N_1 and N_2 are obtained from N only by renaming the free variables with fresh variables of the same type, for each derivation $\Delta_{1,i} \vdash^{\not\square} N : \tau_{1,i}$ where $i \in \{1, \ldots, n_1\}$ we have $\Delta'_{1,i} \vdash^{\not\square} N_1 : \tau_{1,i}$, where $\Delta_{1,i}$ and $\Delta'_{1,i}$ differ only in variables names. Analogously, for each derivation $\Delta_{1,j} \vdash^{\not\square} N : \tau_{1,j}$ where $i \in \{1, \ldots, n_2\}$ we have $\Delta''_{1,j} \vdash^{\not\square} N_2 : \tau_{1,j}$, where $\Delta_{1,j}$ and $\Delta''_{1,j}$ differ only in variables names. Now, we also adopt the following abbreviations:

$$\begin{array}{rcl}
\Delta'_1 & \triangleq & \Delta'_{1,1} \sqcap \ldots \sqcap \Delta'_{1,n_1} \\
\Delta''_2 & \triangleq & \Delta''_{2,1} \sqcap \ldots \sqcap \Delta''_{2,n_2} \\
\mathfrak{L}'_1 & \triangleq & \Delta'_{1,1} \vdash^{\not\square} N_1 : \tau_{1,1} \ \ldots \ \Delta'_{1,n_1} \vdash^{\not\square} N_1 : \tau_{1,n_1} \\
\mathfrak{L}''_2 & \triangleq & \Delta''_{2,1} \vdash^{\not\square} N_2 : \tau_{2,1} \ldots \Delta''_{2,n_2} \vdash^{\not\square} N_2 : \tau_{2,n_2}
\end{array}$$

Moreover, we do not consider the environment Δ_0 since it is useless here. Now, on one hand we have:

$$\dfrac{\dfrac{\Gamma, x_1 : \tau_1, x_2 : \tau_2 \vdash^{\not\square} M : \sigma}{\Gamma, x : \tau_1 \cap \tau_2 \vdash^{\not\square} x <_{x_2}^{x_1} M : \sigma}\,(Cont) \qquad \mathfrak{L}_1 \quad \mathfrak{L}_2}{\Gamma, \Delta_1 \sqcap \Delta_2 \vdash^{\not\square} (x <_{x_2}^{x_1} M)[N/x] : \sigma}\,(Subst)$$

On the other hand we have

$$\dfrac{\dfrac{\dfrac{\Gamma, x_1:\tau_1, x_2:\tau_2 \vdash^{\boxtimes} M:\sigma \quad \mathfrak{L}'_1}{\Gamma, \Delta'_1, x_2:\tau_2 \vdash^{\boxtimes} M[N_1/x_1]:\sigma}\,(Subst) \qquad \mathfrak{L}''_2}{\Gamma, \Delta'_1, \Delta''_2 \vdash^{\boxtimes} M[N_1/x_1][N_2/x_2]:\sigma}\,(Subst)}{\vdots}\,(Cont)}{\Gamma, \Delta_1 \sqcap \Delta_2 \vdash^{\boxtimes} Fv[N] <^{Fv[N_1]}_{Fv[N_2]} M[N_1/x_1][N_2/x_2]:\sigma}\,(Cont)$$

(ii) We will denote by $Q \equiv \mathcal{C}[M[N/x]]$ and $Q' \equiv \mathcal{C}[M']$. If $Q \xrightarrow{\quad\boxtimes\quad}\!\!\!\!\twoheadrightarrow Q'$ this means that $M[N/x] \xrightarrow{\quad\boxtimes\quad}\!\!\!\!\twoheadrightarrow M'$. We prove the statement by induction on the structure of a context containing a redex. We provide the proof for the basic case $\mathcal{C} = [\,]$ and three additional cases $\mathcal{C} = \lambda x.\mathcal{C}'$, $\mathcal{C} = x \odot \mathcal{C}'$ and $\mathcal{C} = \mathcal{C}'[P/y]$, the proof being similar for the remaining context kinds.

- Case $\mathcal{C} = [\,]$. This is the first part of this lemma (i).
- Case $\mathcal{C} = \lambda x.\mathcal{C}'$. Then $Q = \lambda x.\mathcal{C}'[M[N/x]]$ and $Q' = \lambda x.\mathcal{C}'[M']$. By assumption $\Gamma \vdash^{\boxtimes} Q:\sigma$, i.e. $\Gamma \vdash^{\boxtimes} \lambda x.\mathcal{C}'[M[N/x]]:\sigma$. Using Generation lemma for $\lambda^{\boxtimes}_{\circledR}\cap$ (Lemma 32(i)) we obtain that there exist α and τ such that $\sigma = \alpha \to \tau$ and $\Gamma, x:\alpha \vdash^{\boxtimes} \mathcal{C}'[M[N/x]]:\tau$. Since $M[N/x] \xrightarrow{\quad\boxtimes\quad}\!\!\!\!\twoheadrightarrow M'$ by IH we have that $\Gamma, x:\alpha \vdash^{\boxtimes} \mathcal{C}'[M']:\tau$. Using rule $(\to_I)$ we can conclude that $\Gamma \vdash^{\boxtimes} \lambda.\mathcal{C}'[M']:\alpha \to \tau = \sigma$.
- Case $\mathcal{C} = x \odot \mathcal{C}'$. Then $Q = x \odot \mathcal{C}'[M[N/x]]$ and $Q' = x \odot \mathcal{C}'[M']$. By assumption $\Gamma \vdash^{\boxtimes} Q:\sigma$, i.e. $\Gamma \vdash^{\boxtimes} x \odot \mathcal{C}'[M[N/x]]:\sigma$. Using Generation lemma for $\lambda^{\boxtimes}_{\circledR}\cap$ (Lemma 32(iv)) we obtain that $\Gamma = \Gamma', x:\top$ and $\Gamma' \vdash^{\boxtimes} \mathcal{C}'[M[N/x]]:\sigma$. Since $M[N/x] \xrightarrow{\quad\boxtimes\quad}\!\!\!\!\twoheadrightarrow M'$ by IH we have that $\Gamma' \vdash^{\boxtimes} \mathcal{C}'[M']:\sigma$. Using rule $(Thin)$ we can conclude that $\Gamma \vdash^{\boxtimes} x \odot \mathcal{C}'[M']:\sigma$.
- Case $\mathcal{C} = \mathcal{C}'[P/y]$. Then $Q = \mathcal{C}'[P/y][M[N/x]]$ and $Q' = \mathcal{C}'[P/y][M']$. By assumption $\Gamma \vdash^{\boxtimes} Q:\sigma$, i.e. $\Gamma \vdash^{\boxtimes} \mathcal{C}'[P/y][M[N/x]]:\sigma$. Using Generation lemma for $\lambda^{\boxtimes}_{\circledR}\cap$ (Lemma 32(v)) we obtain that there exist Δ_i and $\tau_i, i = 0,\ldots,n$ such that $\Gamma', y:\cap_i^n\tau_i \vdash^{\boxtimes} \mathcal{C}'[M[N/x]]:\sigma$ and for all $i \in \{0,\ldots,n\}$, $\Delta_i \vdash^{\boxtimes} P:\tau_i$ and $\Gamma = \Gamma', \Delta_0^{\top} \sqcap \Delta_1 \sqcap \ldots \sqcap \Delta_n$. Since $M[N/x] \xrightarrow{\quad\boxtimes\quad}\!\!\!\!\twoheadrightarrow M'$ by IH

we have that $\Gamma', y : \cap_i^n \tau_i \vdash^{\not\square} \mathcal{C}'[M'] : \sigma$. Using rule $(Subst)$ we can conclude that $\Gamma \vdash^{\not\square} \mathcal{C}'[P/y][M'] : \sigma$.

□

Lemma 35 (Substitution lemma for $\lambda_{\circledR}\cap$)**.** *If* $\Gamma, x : \cap_i^n \tau_i \vdash M : \sigma$ *and* $\Delta_i \vdash N : \tau_i$, *for all* $i \in \{0, \ldots, n\}$, *then* $\Gamma, \Delta_0^\top \sqcap \Delta_1 \sqcap \ldots \sqcap \Delta_n \vdash M\llbracket N/\!\!/x \rrbracket : \sigma$.

Proof. From assumptions $\Gamma, x : \cap_i^n \tau_i \vdash M : \sigma$ and $\Delta_i \vdash N : \tau_i$, for all $i \in \{0, \ldots, n\}$, we get that $\Gamma, x : \cap_i^n \tau_i \vdash^{\not\square} M : \sigma$ and for all $i \in \{0, \ldots, n\}$, $\Delta_i \vdash^{\not\square} N : \tau_i$. Applying $(Subst)$ rule we get $\Gamma, \Delta_0^\top \sqcap \Delta_1 \sqcap \ldots \sqcap \Delta_n \vdash^{\not\square} M[N/x] : \sigma$. Now, using termination and confluence of $\xrightarrow{\not\square}$ reduction (Proposition 10 and Proposition 11) and preservation of type under the $\xrightarrow{\not\square}$ reduction (Lemma 34) we obtain that the unique normal form $M\llbracket N/\!\!/x \rrbracket$ exists and that $\Gamma, \Delta_0^\top \sqcap \Delta_1 \sqcap \ldots \sqcap \Delta_n \vdash^{\not\square} M\llbracket N/\!\!/x \rrbracket : \sigma$. Since $M\llbracket N/\!\!/x \rrbracket \in \Lambda_{\circledR}$ (Proposition 14), having that $\lambda_{\circledR}^{\not\square}\cap$ is conservative extension of $\lambda_{\circledR}\cap$, we finally get that $\Gamma, \Delta_0^\top \sqcap \Delta_1 \sqcap \ldots \sqcap \Delta_n \vdash M\llbracket N/\!\!/x \rrbracket : \sigma$. □

Proposition 36 (Type preservation under reduction and equivalence in $\lambda_{\circledR}\cap$)**.** *For every* $\lambda_{\circledR}$*-term* M*: if* $\Gamma \vdash M : \sigma$ *and* $M \twoheadrightarrow M'$ *or* $M \equiv_{\lambda_{\circledR}} M'$, *then* $\Gamma \vdash M' : \sigma$.

Proof. The proof is done by case analysis on the applied reduction. Since the property is stable by context, we can without loss of generality assume that the reduction takes place at the outermost position of the term. Here we just show several cases. We will use GL as an abbreviation for Generation lemma (Lemma 29).

- Case (β): Let $\Gamma \vdash (\lambda x.M)N : \sigma$. We want to show that $\Gamma \vdash M\llbracket N/\!\!/x \rrbracket : \sigma$. From $\Gamma \vdash (\lambda x.M)N : \sigma$ and from GL(ii) it follows that $\Gamma = \Gamma', \Delta_0^\top \sqcap \Delta_1 \sqcap \ldots \sqcap \Delta_n$, and that there is a type $\cap_i^n \tau_i$ such that for all $i = 0, \ldots, n$, $\Delta_i \vdash N : \tau_i$, and $\Gamma' \vdash \lambda x.M : \cap_i^n \tau_i \to \sigma$. Further, by GL(i) we have that $\Gamma', x : \cap_i^n \tau_i \vdash M : \sigma$. Now, all the assumptions of Substitution lemma 35 hold, yielding $\Gamma', \Delta_0^\top \sqcap \Delta_1 \sqcap \ldots \sqcap \Delta_n \vdash M\llbracket N/\!\!/x \rrbracket : \sigma$ which is exactly what we need, since $\Gamma = \Delta_0^\top \sqcap \Gamma', \Delta_1 \sqcap \ldots \sqcap \Delta_n$.

- Case $(\gamma\omega_2)$: Let $\Gamma \vdash x <_{x_2}^{x_1} x_1 \odot M : \sigma$. We are showing that $\Gamma \vdash M\llbracket x/\!\!/x_2 \rrbracket : \sigma$. From the first sequent by GL(iii) we have that $\Gamma = \Gamma', x : \alpha \cap \beta$ and $\Gamma', x_1 : \alpha, x_2 : \beta \vdash x_1 \odot M : \sigma$. Further, by GL(iv) we conclude that $\alpha \equiv \top$, $x : \top \cap \beta \equiv \beta$ and $\Gamma', x_2 : \beta \vdash M : \sigma$. Since $\beta = \cap_i^n \tau_i$ for some $n \geq 0$, by applying Substitution lemma 35 to $\Gamma', x_2 : \beta \vdash M : \sigma$ and $x : \tau_i \vdash x : \tau_i$, $i = 0, \ldots, n$ we get $\Gamma \vdash M\llbracket x/\!\!/x_2 \rrbracket : \sigma$.

- The other rules are easy since they do not essentially change the structure of the term.

□

Due to this property, equivalent (by $\equiv_{\lambda_{\circledR}}$) terms have the same type.

3 Characterisation of strong normalisation in $\lambda_{\circledR}$

3.1 SN $\Rightarrow$ Typeability in $\lambda_{\circledR}\cap$

We want to prove that if a $\lambda_{\circledR}$-term is strongly normalising (SN), then it is typeable in the system $\lambda_{\circledR}\cap$. We proceed in two steps:

1. we show that all $\lambda_{\circledR}$-normal forms are typeable and
2. we prove the redex subject expansion.

Proposition 37. *$\lambda_{\circledR}$-normal forms are typeable in the system $\lambda_{\circledR}\cap$.*

Proof. By induction on the structure of M_{nf} and E_{nf}, given in Definition 20. The basic case is a variable, namely $xM^1_{nf}\dots M^n_{nf}$, where $n=0$. It is typeable by (Ax). Cases involving duplication and erasure operators are easy, because the associated type assignment rules $(Cont)$ and $(Thin)$ preserve the type of a term. If $M_{nf}=\lambda x.x\odot N_{nf}$, then by the induction hypothesis $\Gamma\vdash N_{nf}:\sigma$, hence $\Gamma,x:\top\vdash x\odot N_{nf}:\sigma$ and $\Gamma\vdash\lambda x.x\odot N_{nf}:\top\to\sigma$. Further, we discuss the case $xM^1_{nf}\dots M^n_{nf}$, where $n\geq 1$. In this case, $M^1_{nf},\dots,M^n_{nf}$ are typeable by the induction hypothesis, say $\Gamma^i_j\vdash M^i_{nf}:\sigma^i_j$, $i\in\{1,\dots,n\}$, $j\in\{1,\dots,m_i\}$. Then, since x is a fresh variable, taking $x:\cap^{m_1}_j\sigma^1_j\to(\cap^{m_2}_j\sigma^2_j\to\dots(\cap^{m_n}_j\sigma^n_j\to\tau)\dots)$ and applying $(\to_E)$ rule n times, we obtain $\Gamma\vdash xM^1_{nf}\dots M^n_{nf}:\tau$, where $\Gamma=x:\cap^{m_1}_j\sigma^1_j\to(\cap^{m_2}_j\sigma^2_j\to\dots(\cap^{m_n}_j\sigma^n_j\to\tau)\dots),\Gamma^{1\top}_0\sqcap\Gamma^1_1\sqcap\dots\sqcap\Gamma^1_{m_1},\dots,\Gamma^{n\top}_0\sqcap\Gamma^n_1\sqcap\dots\sqcap\Gamma^n_{m_n}$. □

Lemma 38. *For all $M,M'\in\Lambda^{\boxtimes}_{\circledR}$ and $N\in\Lambda_{\circledR}$, if $\Gamma\vdash^{\boxtimes}M':\sigma$, $M[N/x]\xrightarrow{\boxtimes}M'$, and N is typeable, then $\Gamma\vdash^{\boxtimes}M[N/x]:\sigma$.*

Proof. The proof is by case analysis on the applied $\xrightarrow{\boxtimes}$ reduction. We consider only some interesting rules.

- Rule $(x \odot M)[N/x] \xrightarrow{\boxtimes} Fv(N) \odot M$.
  Let $Fv(N) = \{x_1, ..., x_m\}$. By assumption N is typeable, thus $\Delta_0 \vdash N : \tau_0$ for some $\Delta_0 = \{x_1 : \tau_1, ..., x_m : \tau_m\}$. If $\Gamma \vdash^{\boxtimes} Fv(N) \odot M : \sigma$, then by applying m times the Generation Lemma 32(iv), we get $\Gamma' \vdash^{\boxtimes} M : \sigma$, where $\Gamma = \Gamma', \Delta_0^{\top}$. On the other hand

$$\dfrac{\dfrac{\Gamma' \vdash^{\boxtimes} M : \sigma}{\Gamma', x : \top \vdash^{\boxtimes} x \odot M : \sigma}\,(Thin) \qquad \Delta_0 \vdash N : \tau_0}{\Gamma', \Delta_0^{\top} \vdash^{\boxtimes} (x \odot M)[N/x] : \sigma.}\,(Subst)$$

  Notice that the rule $(Subst)$ can be applied because $\top = \cap_i^n \tau_i$ for $n = 0$.

- Rule $(x <_{x_2}^{x_1} M)[N/x] \xrightarrow{\boxtimes} Fv[N] <_{Fv[N_2]}^{Fv[N_1]} M[N_1/x_1][N_2/x_2]$.
  Let $Fv[N] = [y_1, ..., y_m]$. Then, since N_1 and N_2 are obtained from N by renaming the free variables, we have that $Fv[N_1] = [y'_1, ..., y'_m]$ and $Fv[N_2] = [y''_1, ..., y''_m]$. From the assumption $\Gamma \vdash^{\boxtimes} Fv[N] <_{Fv[N_2]}^{Fv[N_1]} M[N_1/x_1][N_2/x_2] : \sigma$, by m applications of Lemma $32(iii)$, we obtain that $\Gamma = \Gamma', y_1 : \tau_1 \cap \rho_1, ..., y_m : \tau_m \cap \rho_m$ and that $\Gamma', \Delta', \Delta'' \vdash^{\boxtimes} M[N_1/x_1][N_2/x_2] : \sigma$, where $\Delta' = \{y'_1 : \tau_1, ..., y'_m : \tau_m\}$ and $\Delta'' = \{y''_1 : \rho_1, ..., y''_m : \rho_m\}$. Now, by two applications of Lemma $32(v)$, we get that $\Delta' = {\Delta'_0}^{\top} \sqcap \Delta'_1 ... \sqcap \Delta'_{n_1}$, $\Delta'' = {\Delta''_0}^{\top} \sqcap \Delta''_1 ... \sqcap \Delta''_{n_2}$, where $\Delta'_i = \{y'_1 : \tau_{1,i}, ..., y'_m : \tau_{m,i}\}$ for $i \in \{0, ..., n_1\}$, $\Delta''_j = \{y''_1 : \rho_{1,j}, ..., y''_m : \rho_{m,j}\}$ for $j \in \{0, ..., n_2\}$, $\Delta'_i \vdash^{\boxtimes} N_1 : \cap_k^m \tau_{k,i}$, $\Delta''_j \vdash^{\boxtimes} N_2 : \cap_k^m \rho_{k,j}$, and finally $\Gamma', x_1 : \cap_i^{n_1} \tau_i, x_2 : \cap_j^{n_2} \rho_j \vdash^{\boxtimes} M : \sigma$ (we used the following abbreviations: $\cap_k^m \tau_{k,i} \equiv \tau_i$, $\cap_k^m \rho_{k,j} \equiv \rho_j$). Now, since N_1 and N_2 are obtained from N by renaming, for each derivation of the type of N_1 (respectively N_2) we can write an analogous derivation of the type of N, i.e. $\Delta_i \vdash^{\boxtimes} N : \tau_i$ for $i \in \{0, ..., n_1\}$ and $\Delta_j \vdash^{\boxtimes} N : \rho_j$ for $j \in \{0, ..., n_2\}$, where Δ_i differ from Δ'_i (and respectively Δ_j from Δ''_j) only by the domain ($Dom(\Delta_i) = Dom(\Delta_j) = \{y_1, ..., y_m\}$). If we adopt abbreviations $\mathfrak{L}_1$ for the array of the first n_1 derivations, and $\mathfrak{L}_2$ for the array of the latter n_2 derivations, we have:

$$\dfrac{\dfrac{\Gamma', , x_1 : \cap_i^{n_1} \tau_i, x_2 : \cap_j^{n_2} \rho_j \vdash^{\boxtimes} M : \sigma}{\Gamma', x : (\cap_i^{n_1} \tau_i) \cap (\cap_j^{n_2} \rho_j) \vdash^{\boxtimes} x <_{x_2}^{x_1} M : \sigma}\,(Cont) \qquad \mathfrak{L}_1 \quad \mathfrak{L}_2}{\Gamma \vdash^{\boxtimes} (x <_{x_2}^{x_1} M)[N/x] : \sigma.}\,(Subst)$$

  The left hand side of the latter assignment holds because $\Gamma', \Delta_0^{\top} \sqcap \Delta_1 \sqcap ... \sqcap \Delta_{n_1+n_2} = \Gamma', y_1 : \top \cap (\cap_i^{n_1} \tau_{1,i}) \cap (\cap_j^{n_2} \rho_{1,j}), ..., y_m : \top \cap (\cap_i^{n_1} \tau_{m,i}) \cap (\cap_j^{n_2} \rho_{m,j}) = \Gamma', y_1 : \tau_1 \cap \rho_1, ..., y_m : \tau_m \cap \rho_m = \Gamma$.

□

Proposition 39 (Redex subject expansion)**.**

(i) If $\Gamma \vdash M[\![N/\!/x]\!] : \sigma$ *and* N *is typeable, then* $\Gamma \vdash (\lambda x.M)N : \sigma$.

(ii) Let M *be a* $\lambda_{\circledR}$*-redex other than a* β*-redex and* $M \to M'$*. If* $\Gamma \vdash M' : \sigma$*, then* $\Gamma \vdash M : \sigma$.

Proof. (i) From $\Gamma \vdash M[\![N/\!/x]\!] : \sigma$ we have that $\Gamma \vdash^{\boxslash} M[N/x] : \sigma$ using Lemma 38 multiple times, since $M[\![N/\!/x]\!] = M[N/x]\downarrow^{\boxslash}$, i.e. $M[N/x] \xrightarrow{\boxslash}\!\!\!\!\twoheadrightarrow M[\![N/\!/x]\!]$. From $\Gamma \vdash^{\boxslash} M[N/x] : \sigma$ by Lemma 32(v) (Generation lemma) it follows that there exist Δ_i and τ_i, $i = 0,\ldots,n$ such that $\Gamma', x : \cap_i^n \tau_i \vdash^{\boxslash} M : \sigma$ and for all $i \in \{0,\ldots,n\}$, $\Delta_i \vdash N : \tau_i$ and $\Gamma = \Gamma', \Delta_0^{\top} \sqcap \Delta_1 \sqcap \ldots \sqcap \Delta_n$. Now:

$$\dfrac{\dfrac{\Gamma', x : \cap_i^n \tau_i \vdash^{\boxslash} M : \sigma}{\Gamma' \vdash^{\boxslash} \lambda x.M : \cap_i^n \tau_i \to \sigma}\,(\to_I) \quad \Delta_0 \vdash^{\boxslash} N : \tau_0 \quad \ldots \quad \Delta_n \vdash^{\boxslash} N : \tau_n}{\Gamma \vdash^{\boxslash} (\lambda x.M)N : \sigma}\,(\to_E)$$

Since $M, N \in \Lambda_{\circledR}$ we have that $\Gamma \vdash (\lambda x.M)N : \sigma$.

(ii) By case analysis according to the applied reduction, similar to the proof of Proposition 36. □

Theorem 40 (SN $\Rightarrow$ typeability)**.** *All strongly normalising* $\lambda_{\circledR}$*-terms are typeable in the* $\lambda_{\circledR}\cap$ *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 structure of M.

- If M is a normal form, then M is typeable by Proposition 37.
- If M is a $\lambda_{\circledR}$-redex, i.e. $M \to M'$, then let M' be its contractum. M' is also strongly normalising, hence by IH it is typeable. Then M is typeable, by Proposition 39. Notice that, if $M \equiv (\lambda x.N)P \to_\beta N[\![P/\!/x]\!] \equiv M'$, then, by IH, P is typeable, since the length of the longest reduction path out of P is smaller than that of M.
- Next, suppose that M itself is neither a redex nor a normal form. Then, according to Lemma 22, M has of one of the following forms:

- $\lambda x.N$ (where $N \neq y \odot P$ and $y \neq x$, since in this case M would be a redex and previous case would apply),
- $xT_1 \dots T_n$,
- $x \odot N$,
- $(\lambda x.N)PT_1 \dots T_n$,
- $(x \odot N)PT_1 \dots T_n$,
- $(x <_{x_2}^{x_1} N)T_1 \dots T_n$,

where $N, P, T_1, \dots, T_n$, are *not* all normal forms. We can classify these forms into the following two categories:

1) Terms with internal redexes: $\lambda x.N$, $xT_1 \dots T_n$, $x \odot N$ and $(x <_{x_2}^{x_1} N)T_1 \dots T_n$ when duplication cannot be propagated further into N, i.e. $N \equiv PQ$, $x_1 \in Fv(P)$, $x_2 \in Fv(Q)$. In all these cases, we proceed by subinduction on the structure of M, since the length of the longest reduction path out of a subterm that contains a redex is equal to the length of the longest reduction path out of M.
2) Terms with a leftmost redex: $(\lambda x.N)PT_1 \dots T_n$, $(x \odot N)PT_1 \dots T_n$ and $(x <_{x_2}^{x_1} N)T_1 \dots T_n$ when duplication can be propagated further into N. In these cases, by applying the leftmost reduction, we obtain a term with smaller length of the longest reduction path, therefore we can proceed using induction.

In all the cases, after the application of induction (respectively subinduction) hypothesis in order to conclude typeability of subterms of M, it is easy to build the type of M. We will prove some illustrative cases from both categories, the rest being similar.

– $M \equiv \lambda x.N$. Then, the only way to reduce M is to reduce N and the number of reductions in N is equal to the number of reductions in M. Since M is SN, N is also *SN*. Since N is a subterm of M, N is typeable by subinduction and $\lambda x.N$ is typeable by $(\rightarrow_I)$.
– $M \equiv xT_1 \dots T_n$. Then $T_1, \dots, T_n$ must be SN by subinduction, hence typeable. Then we build the type for M by multiple application of the rule $(\rightarrow_E)$, as in Proposition 37.

– $M \equiv (x <^{x_1}_{x_2} PQ)T_1 \dots T_n$ with $x_1 \in Fv(P)$, $x_2 \in Fv(Q)$. Again, each of $P, Q, T_1, \dots, T_n$ must be SN by subinduction, hence typeable. We first use the rule $(Cont)$ to type $x <^{x_1}_{x_2} PQ$ and then we use the rule $(\to_E)$, as in Proposition 37 to type M.

– $M \equiv (\lambda x.N)PT_1 \dots T_n$. Then $M \to M'$ where $M' \equiv N\llbracket P/\!\!/x \rrbracket T_1 \dots T_n$. M' is also SN, hence typeable by induction hypothesis, since the longest reduction path out of M' is smaller than the one out of M. This implies that $N\llbracket P/\!\!/x \rrbracket, T_1, \dots, T_n$ are also SN and hence typeable by sub induction. Then we build the type for M by multiple application of the rule $(\to_E)$, as in Proposition 37. The cases $M \equiv (x \odot N)PT_1 \dots T_n$ and $M \equiv (x <^{x_1}_{x_2} N)T_1 \dots T_n$ are analogous.

□

3.2 Typeability ⇒ SN in $\lambda_{\circledR}\cap$

In various type assignment systems, the *reducibility method* can be used to prove many reduction properties of typeable terms. It was first introduced by Tait [51] for proving the strong normalisation of simply typed λ-calculus, and developed further to prove *strong normalisation* of various calculi in [52, 28, 37, 24, 27], *confluence* (the Church-Rosser property) of βη-reduction in [36, 50, 40, 41, 27] and to characterise certain classes of λ-terms such as strongly normalising, normalising, head normalising, and weak head normalising terms (and their persistent versions) by their typeability in various intersection type systems in [21, 17, 15, 16].

The main idea of the reducibility method is to interpret types by suitable sets of lambda terms which satisfy some realisability properties and prove the soundness of type assignment with respect to these interpretations. A consequence of soundness is that every typeable term belongs to the interpretation of its type, hence satisfying a desired reduction property.

In the sequel, we adapt the reducibility method in order to prove that terms typeable in $\lambda_{\circledR}\cap$ are strongly normalising.

Definition 41. *For $\mathcal{M}, \mathcal{N} \subseteq \Lambda_{\circledR}$, we define $\mathcal{M} \longrightarrow \mathcal{N} \subseteq \Lambda_{\circledR}$ as*

$$\mathcal{M} \longrightarrow \mathcal{N} = \{M \in \Lambda_{\circledR} \mid \forall N \in \mathcal{M} \quad MN \in \mathcal{N}\}.$$

Definition 42. *The type interpretation $[\![-]\!] : \mathsf{Types} \to 2^{\Lambda_{\circledR}}$ is defined by:*

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

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

(I3) $[\![\cap_i^n \sigma_i]\!] = \begin{cases} \cap_i^n [\![\sigma_i]\!] & \text{for } n > 0 \\ \mathcal{SN} & \text{for } n = 0. \end{cases}$

Next, we introduce the notions of *variable property*, β-*expansion property*, ω-*expansion property*, γ-*reduction property*, *thinning property* and *contraction property.* The variable property and the β-expansion property correspond to the saturation property given in [5].

Definition 43.

- *A set* $\mathcal{X} \subseteq \Lambda_{\circledR}$ *satisfies the* variable property, *notation* $\mathsf{VAR}(\mathcal{X})$, *if* $\mathcal{X}$ *contains all the terms of the form* $xM_1 \ldots M_n$, *where* $n \geq 0$ *and* $M_i \in \mathcal{SN}$, $i = 1, \ldots, n$.

- *A set* $\mathcal{X} \subseteq \Lambda_{\circledR}$ *satisfies the* β-expansion property, *notation* $\mathsf{EXP}_\beta(\mathcal{X})$ *if*

$$\frac{M_1 \in \mathcal{SN} \ \ldots \ M_n \in \mathcal{SN} \ N \in \mathcal{SN} \qquad M[\![N/\!/x]\!]M_1 \ldots M_n \in \mathcal{X}}{(\lambda x.M)\,N M_1 \ldots M_n \in \mathcal{X}.}\ \mathsf{EXP}_\beta(\mathcal{X})$$

- *A set* $\mathcal{X} \subseteq \Lambda_{\circledR}$ *satisfies the* ω-expansion property, *notation* $\mathsf{EXP}_\omega(\mathcal{X})$ *if*

$$\frac{M_1 \in \mathcal{SN} \ \ldots \ M_n \in \mathcal{SN} \ N \in \mathcal{SN} \qquad x \odot (MN)M_1 \ldots M_n \in \mathcal{X}}{(x \odot M)\,N M_1 \ldots M_n \in \mathcal{X}.}\ \mathsf{EXP}_\omega(\mathcal{X})$$

- *A set* $\mathcal{X} \subseteq \Lambda_{\circledR}$ *satisfies the* γ-reduction property, *notation* $\mathsf{RED}_\gamma(\mathcal{X})$ *if*

$$\frac{M_1 \in \mathcal{SN} \ \ldots \ M_n \in \mathcal{SN} \ N \in \mathcal{SN} \qquad x <_{x_2}^{x_1} (MN)M_1 \ldots M_n \in \mathcal{X}}{(x <_{x_2}^{x_1} M)\,N M_1 \ldots M_n \in \mathcal{X}.}\ \mathsf{RED}_\gamma(\mathcal{X})$$

- *A set* $\mathcal{X} \subseteq \Lambda_{\circledR}$ *satisfies the* thinning property, *notation* $\mathsf{THIN}(\mathcal{X})$ *if:*

$$\frac{M \in \mathcal{X}}{x \odot M \in \mathcal{X}.}\ \mathsf{THIN}(\mathcal{X})$$

- *A set* $\mathcal{X} \subseteq \Lambda_{\circledR}$ *satisfies the* contraction property, *notation* $\mathsf{CONT}(\mathcal{X})$ *if:*

$$\frac{M \in \mathcal{X}}{x <_z^y M \in \mathcal{X}.}\ \mathsf{CONT}(\mathcal{X})$$

Remark. In Definition 43 it is not necessary to explicitly write the conditions about free variables since we work with $\lambda_{\circledR}$-terms.

Definition 44 ($\circledR$-Saturated set)**.** *A set $\mathcal{X} \subseteq \Lambda_{\circledR}$ is called* $\circledR$-saturated*, if $\mathcal{X} \subseteq \mathcal{SN}$ and $\mathcal{X}$ satisfies the variable, β-expansion, ω-expansion, γ-reduction, thinning and contraction properties.*

Proposition 45. *Let $\mathcal{M}, \mathcal{N} \subseteq \Lambda_{\circledR}$.*

(i) $\mathcal{SN}$ is $\circledR$-saturated.

(ii) If $\mathcal{M}$ and $\mathcal{N}$ are $\circledR$-saturated, then $\mathcal{M} \longrightarrow \mathcal{N}$ is $\circledR$-saturated.

(iii) If $\mathcal{M}$ and $\mathcal{N}$ are $\circledR$-saturated, then $\mathcal{M} \cap \mathcal{N}$ is $\circledR$-saturated.

(iv) For all types $\varphi \in Types$, $[\![\varphi]\!]$ is $\circledR$-saturated.

Proof. (i)

- $\mathcal{SN} \subseteq \mathcal{SN}$ and $\mathsf{VAR}(\mathcal{SN})$ trivially hold.
- $\mathsf{EXP}_{\beta}(\mathcal{SN})$. Suppose that $M[\![N/\!/x]\!]M_1 \dots M_n \in \mathcal{SN}$, $M_1, \dots, M_n \in \mathcal{SN}$ and $N \in \mathcal{SN}$. We know that $M[\![N/\!/x]\!] \in \mathcal{SN}$ as a subterm of a term in $\mathcal{SN}$ and $N \in \mathcal{SN}$, hence $M \in \mathcal{SN}$. By assumption, $M_1, \dots, M_n \in \mathcal{SN}$, so all reductions inside of these terms terminate. Starting from $(\lambda x.M)NM_1 \dots M_n$, we can either contract the head redex and obtain $M[\![N/\!/x]\!]M_1 \dots M_n$ which is SN by assumption, so we are done, or we can contract redexes inside $M, N, M_1, \dots, M_n$, which are all SN by assumption. All these reduction paths are finite. Consider a term obtained after finitely many reduction steps

  $$(\lambda x.M)NM_1 \dots M_n \to \dots \to (\lambda x.M')N'M'_1 \dots M'_n$$

  where $M \twoheadrightarrow M'$, $N \twoheadrightarrow N'$, $M_1 \twoheadrightarrow M'_1, \dots, M_n \twoheadrightarrow M'_n$. After contracting the head redex of $(\lambda x.M')N'M'_1 \dots M'_n$ to $M'[\![N'/\!/x]\!]M'_1 \dots M'_n$, we actually obtain a reduct of $M[\![N/\!/x]\!]M_1 \dots M_n \in \mathcal{SN}$. Hence, $(\lambda x.M)NM_1 \dots M_n \in \mathcal{SN}$.
- $\mathsf{EXP}_{\omega}(\mathcal{SN})$. Suppose that $x \odot (MN)M_1 \dots M_n \in \mathcal{SN}$, $M_1, \dots, M_n \in \mathcal{SN}$.Since $x \odot (MN)$ is a subterm of a term in $\mathcal{SN}$, we know that $MN \in \mathcal{SN}$ and consequently $M, N \in \mathcal{SN}$. By assumption, $M_1, \dots, M_n \in \mathcal{SN}$, so the reductions inside of these terms terminate. Starting from $(x \odot M)NM_1 \dots M_n$, we can either contract the head redex and obtain $x \odot (MN)M_1 \dots M_n$ which is SN by assumption, so we are done, or we can contract redexes inside

$M, N, M_1, \ldots, M_n$, which are all SN by assumption. All these reduction paths are finite. Consider a term obtained after finitely many reduction steps

$$(x \odot M)NM_1 \ldots M_n \to \ldots \to (x \odot M')N'M'_1 \ldots M'_n$$

where $M \twoheadrightarrow M'$, $M_1 \twoheadrightarrow M'_1, \ldots, M_n \twoheadrightarrow M'_n$. After contracting the head redex of $(x \odot M')N'M'_1 \ldots M'_n$ to $x \odot (M'N')M'_1 \ldots M'_n$, we obtain a reduct of $x \odot (MN)M_1 \ldots M_n \in \mathcal{SN}$. Hence, $(x \odot M)NM_1 \ldots M_n \in \mathcal{SN}$.

- $\mathsf{RED}_\gamma(\mathcal{SN})$. This is trivial, since by reducing a SN term we again obtain a SN term.

- $\mathsf{THIN}(\mathcal{SN})$. Suppose that $M \in \mathcal{SN}$ and $x \notin Fv(M)$. Then trivially $x \odot M \in \mathcal{SN}$, since no new redexes are formed.

- $\mathsf{CONT}(\mathcal{SN})$. Suppose that $M \in \mathcal{SN}$, $y \neq z$, $y, z \in Fv(M)$, $x \notin Fv(M) \setminus \{y, z\}$. We prove that $x <_z^y M \in \mathcal{SN}$ by induction on the structure of M.

  - $M = yz$. Then $x <_z^y M = x <_z^y (yz)$ which is a normal form.
  - $M = y \odot z$. Then $x <_z^y M = x <_z^y (y \odot z) \to_{\gamma\omega_2} z[x/\!/z] = x \in \mathcal{SN}$.
  - $M = \lambda w.N$. Then $N \in \mathcal{SN}$ and $x <_z^y M = x <_z^y (\lambda w.N) \to_{\gamma_1} \lambda w.x <_z^y N \in \mathcal{SN}$, since $x <_z^y N \in \mathcal{SN}$ by IH.
  - $M = PQ$. Then $P, Q \in \mathcal{SN}$ and if $y, z \notin Fv(Q)$, $x <_z^y M = x <_z^y (PQ) \to_{\gamma_2} (x <_z^y P)Q \in \mathcal{SN}$, since by IH $x <_z^y P \in \mathcal{SN}$.
    The case of $\to_{\gamma_3}$ reduction when $y, z \notin Fv(P)$ is analogous.
  - $M = w \odot N$. Then $x <_z^y M = x <_z^y (w \odot N) \to_{\gamma\omega_1} w \odot (x <_z^y N)$. By IH $x <_z^y N \in \mathcal{SN}$ and $w \odot (x <_z^y N)$ does not introduce any new redexes.
  - $M = y \odot N$. Then $x <_z^y M = x <_z^y (y \odot N) \to_{\gamma\omega_2} N[x/\!/z] \in \mathcal{SN}$, since $N \in \mathcal{SN}$ by IH.
  - $M = y <_v^u N$. Then the only possible reduction is inside the term N which is strongly normalising as a subterm of the strongly normalising term $M = y <_v^u N$.
  - $M = x_1 <_{z_1}^{y_1} N$. Analogous to the previous case.

(ii)

- $\mathcal{M} \longrightarrow \mathcal{N} \subseteq \mathcal{SN}$. Suppose that $M \in \mathcal{M} \longrightarrow \mathcal{N}$. Then, for all $N \in \mathcal{M}$, $MN \in \mathcal{N}$. Since $\mathcal{M}$ is ®-saturated, $\mathsf{VAR}(\mathcal{M})$ holds so $x \in \mathcal{M}$ and $Mx \in \mathcal{N} \subseteq \mathcal{SN}$. From here we can deduce that $M \in \mathcal{SN}$.

- $\mathsf{VAR}(\mathcal{M} \longrightarrow \mathcal{N})$. Suppose that x is a variable and $M_1, \ldots, M_n \in \mathcal{SN}, n \geq 0$, such that $x \cap Fv(M_1) \cap \ldots \cap Fv(M_n) = \emptyset$. We need to show that $xM_1 \ldots M_n \in \mathcal{M} \longrightarrow \mathcal{N}$, i.e. $\forall N \in \mathcal{M}$, $xM_1 \ldots M_n N \in \mathcal{N}$. This holds since by assumption $\mathcal{M} \subseteq \mathcal{SN}$ and $\mathcal{N}$ is ®-saturated, i.e. $\mathsf{VAR}(\mathcal{N})$ holds.

- $\mathsf{EXP}_\beta(\mathcal{M} \longrightarrow \mathcal{N})$. Suppose that $M[\![N/\!/x]\!]M_1 \ldots M_n \in \mathcal{M} \longrightarrow \mathcal{N}$, $M_1, \ldots, M_n \in \mathcal{SN}$ and $N \in \mathcal{SN}$. This means that for all $P \in \mathcal{M}$, $M[\![N/\!/x]\!]M_1 \ldots M_n P \in \mathcal{N}$. But $\mathcal{N}$ is ®-saturated, so $\mathsf{EXP}_\beta(\mathcal{N})$ holds and we have that for all $P \in \mathcal{N}$, $(\lambda x.M)NM_1 \ldots M_n P \in \mathcal{N}$. This means that $(\lambda x.M)NM_1 \ldots M_n \in \mathcal{M} \longrightarrow \mathcal{N}$.

- $\mathsf{EXP}_\omega(\mathcal{M} \longrightarrow \mathcal{N})$. Analogous to $\mathsf{EXP}_\beta(\mathcal{M} \longrightarrow \mathcal{N})$.

- $\mathsf{RED}_\gamma(\mathcal{M} \longrightarrow \mathcal{N})$. Suppose that $x <^{x_1}_{x_2} (MN) \in \mathcal{M} \longrightarrow \mathcal{N}$. This means that for all $P \in \mathcal{M}, x <^{x_1}_{x_2} (MN)P \in \mathcal{N}$. But $\mathcal{N}$ is ®-saturated, i.e. $\mathsf{RED}_\gamma(\mathcal{N})$ holds, hence $(x <^{x_1}_{x_2} M)NP \in \mathcal{N}$. This means that $(x <^{x_1}_{x_2} M)N \in \mathcal{M} \longrightarrow \mathcal{N}$.

- $\mathsf{THIN}(\mathcal{M} \longrightarrow \mathcal{N})$. Suppose that $M \in \mathcal{M} \longrightarrow \mathcal{N}$ and $x \notin Fv(M)$. This means that for all $N \in \mathcal{M}, MN \in \mathcal{N}$. But $\mathcal{N}$ is ®-saturated, i.e. $\mathsf{THIN}(\mathcal{N})$ holds, hence $x \odot (MN) \in \mathcal{N}$. Also $\mathsf{EXP}_\omega(\mathcal{N})$ holds so we obtain for all $N \in \mathcal{M}, (x \odot M)N \in \mathcal{N}$, i.e. $x \odot M \in \mathcal{M} \longrightarrow \mathcal{N}$.

- $\mathsf{CONT}(\mathcal{M} \longrightarrow \mathcal{N})$. Let $M \in \mathcal{M} \longrightarrow \mathcal{N}$. We want to prove that $x <^y_z M \in \mathcal{M} \longrightarrow \mathcal{N}$ for $y \neq z$, $y, z \in Fv(M)$ and $x \notin Fv(M)$. Let P be any term in $\mathcal{M}$. We have to prove that $(x <^y_z M)P \in \mathcal{N}$. Since $M \in \mathcal{M} \longrightarrow \mathcal{N}$, we know that $MP \in \mathcal{N}$. By assumption $\mathcal{N}$ is ®-saturated so $x <^y_z (MP) \in \mathcal{N}$. Using $\mathsf{RED}_\gamma(\mathcal{N})$ we obtain $(x <^y_z M)P \in \mathcal{N}$. Therefore $x <^y_z M \in \mathcal{M} \longrightarrow \mathcal{N}$.

(iii)

- $\mathcal{M} \cap \mathcal{N} \subseteq \mathcal{SN}$ is straightforward, since $\mathcal{M}, \mathcal{N} \subseteq \mathcal{SN}$ by assumption.

- $\mathsf{VAR}(\mathcal{M} \cap \mathcal{N})$. Since $\mathsf{VAR}(\mathcal{M})$ and $\mathsf{VAR}(\mathcal{N})$ hold, we have that $\forall M_1, \ldots, M_n \in \mathcal{SN}$, $n \geq 0$: $xM_1 \ldots M_n \in \mathcal{M}$ and $xM_1 \ldots M_n \in \mathcal{N}$. We deduce that $\forall M_1, \ldots, M_n \in \mathcal{SN}, n \geq 0$: $xM_1 \ldots M_n \in \mathcal{M} \cap \mathcal{N}$, i.e. $\mathsf{VAR}(\mathcal{M} \cap \mathcal{N})$ holds.

- $\mathsf{EXP}_\beta(\mathcal{M} \cap \mathcal{N})$ is straightforward.

- $\mathsf{EXP}_\omega(\mathcal{M} \cap \mathcal{N})$ is straightforward.

- $\mathsf{RED}_\gamma(\mathcal{M}\cap\mathcal{N})$. Suppose that $x <^{x_1}_{x_2} (MN) \in \mathcal{M}\cap\mathcal{N}$. Since both $\mathcal{M}$ and $\mathcal{N}$ are ®-saturated $\mathsf{RED}_\gamma(\mathcal{M})$ and $\mathsf{RED}_\gamma(\mathcal{N})$ hold, hence $(x <^{x_1}_{x_2} M)N \in \mathcal{M}$ and $(x <^{x_1}_{x_2} M)N \in \mathcal{M}$, i.e. $(x <^{x_1}_{x_2} M)N \in \mathcal{M}\cap\mathcal{N}$.

- $\mathsf{THIN}(\mathcal{M}\cap\mathcal{N})$. Let $M \in \mathcal{M}\cap\mathcal{N}$ and $x \notin Fv(M)$. Then $M \in \mathcal{M}$ and $M \in \mathcal{N}$. Since both $\mathcal{M}$ and $\mathcal{N}$ are ®-saturated $\mathsf{THIN}(\mathcal{M})$ and $\mathsf{THIN}(\mathcal{N})$ hold, hence $x \odot M \in \mathcal{M}$ and $x \odot M \in \mathcal{N}$, i.e. $x \odot M \in \mathcal{M}\cap\mathcal{N}$.

- $\mathsf{CONT}(\mathcal{M}\cap\mathcal{N})$. Suppose that $M \in \mathcal{M}\cap\mathcal{N}$, $y \neq z$, $y,z \in Fv(M)$, $x \notin Fv(M)\setminus\{y,z\}$. Since both $\mathcal{M}$ and $\mathcal{N}$ are ®-saturated $\mathsf{CONT}(\mathcal{M})$ and $\mathsf{CONT}(\mathcal{N})$ hold, hence $x <^y_z M \in \mathcal{M}$ and $x <^y_z M \in \mathcal{N}$, i.e. $x <^y_z M \in \mathcal{M}\cap\mathcal{N}$.

(iv) By induction on the construction of $\varphi \in \mathsf{Types}$.

- If $\varphi \equiv p$, p a type atom, then $[\![\varphi]\!] = \mathcal{SN}$, so it is ®-saturated using (i).

- If $\varphi \equiv \alpha \to \sigma$, then $[\![\varphi]\!] = [\![\alpha]\!] \longrightarrow [\![\sigma]\!]$. Since $[\![\alpha]\!]$ and $[\![\sigma]\!]$ are ®-saturated by assumption, we can use (ii).

- If $\varphi \equiv \cap_i^n \sigma_i$, then we distinguish two cases:
  - for $n > 0$, $[\![\varphi]\!] = [\![\cap_i^n \sigma_i]\!] = \cap_i^n [\![\sigma_i]\!]$ and for all $i = 1,\ldots,n, [\![\sigma_i]\!]$ are ®-saturated by assumption, so we can use (iii).
  - for $n = 0$, $\varphi \equiv \cap_i^0 \sigma_i$, then $[\![\varphi]\!] = \mathcal{SN}$ and we can use (i).

□

We further define a *valuation of terms* $[\![-]\!]_\rho : \Lambda_® \to \Lambda_®$ and the *semantic satisfiability relation* $\models$ connecting the type interpretation with the term valuation.

Definition 46. *Let* $\rho : \mathtt{var} \to \Lambda_®$ *be a valuation of term variables in* $\Lambda_®$. *For* $M \in \Lambda_®$, *with* $Fv(M) = \{x_1,\ldots,x_n\}$ *the* term valuation $[\![-]\!]_\rho : \Lambda_® \to \Lambda_®$ *is defined as follows:*

$$[\![M]\!]_\rho = M[\rho(x_1)/\!/x_1,\ldots,\rho(x_n)/\!/x_n]$$

providing that $x \neq y \;\Rightarrow\; Fv(\rho(x)) \cap Fv(\rho(y)) = \emptyset$.

Notation: $\rho(N/x)$ is the valuation defined as: $\rho(N/x)(y) = \rho(y)$ and $\rho(N/x)(x) = N$ for $x \neq y$.

Lemma 47.

(i) $[\![x]\!]_\rho = \rho(x)$;

(ii) $[\![MN]\!]_\rho = [\![M]\!]_\rho [\![N]\!]_\rho$;

(iii) $[\![\lambda x.M]\!]_\rho N \to_\beta [\![M]\!]_\rho [N/\!/x]$ *and* $[\![M]\!]_\rho [N/\!/x] = [\![M]\!]_{\rho(N/x)}$;

(iv) $[\![x \odot M]\!]_\rho = Fv(\rho(x)) \odot [\![M]\!]_\rho$;

(v) $[\![z <_y^x M]\!]_\rho = Fv[N] <_{Fv[N_2]}^{Fv[N_1]} [\![M]\!]_{\rho(N_1/x,N_2/y)}$
where $N = \rho(z)$ *and* N_1, N_2 *are obtained from* N *by renaming its free variables.*

Proof.

(i) $[\![x]\!]_\rho = x[\rho(x)/\!/x] = x[\rho(x)/x] \downarrow^{\boxslash} = \rho(x)$, since $x[\rho(x)/x] \xrightarrow{\boxslash} \rho(x)$.

(ii) Without loss of generality, we can assume that $Fv(M) = \{x_1, \ldots, x_i\}$ and $Fv(N) = \{x_{i+1}, \ldots, x_n\}$. Then
$[\![MN]\!]_\rho = (MN)[\rho(x_1)/\!/x_1, \ldots, \rho(x_n)/\!/x_n] =$
$M[\rho(x_1)/\!/x_1, \ldots, \rho(x_i)/\!/x_i] N[\rho(x_{i+1})/\!/x_{i+1}, \ldots, \rho(x_n)/\!/x_n] = [\![M]\!]_\rho [\![N]\!]_\rho$.

(iii) If $Fv(\lambda x.M) = \{x_1, \ldots, x_n\}$, then
$[\![\lambda x.M]\!]_\rho N = (\lambda x.M)[\rho(x_1)/\!/x_1, \ldots, \rho(x_n)/\!/x_n] N =$
$(\lambda x.M[\rho(x_1)/\!/x_1, \ldots, \rho(x_n)/\!/x_n]) N \to (M[\rho(x_1)/\!/x_1, \ldots, \rho(x_n)/\!/x_n])[N/\!/x] =$
$[\![M]\!]_\rho [N/\!/x]$.
$[\![M]\!]_{\rho(N/x)} = M[\rho(N/x)(x_1)/\!/x_1, \ldots, \rho(N/x)(x_n)/\!/x_n, \rho(N/x)(x)/\!/x] =$
$M[\rho(x_1)/\!/x_1, \ldots, \rho(x_n)/\!/x_n][N/\!/x] = [\![M]\!]_\rho [N/\!/x]$.

(iv) If $Fv(M) = \{x_1, \ldots, x_n\}$, then $Fv(x \odot M) = \{x, x_1, \ldots, x_n\}$ and
$[\![x \odot M]\!]_\rho = (x \odot M)[\rho(x)/\!/x, \rho(x_1)/\!/x_1, \ldots, \rho(x_n)/\!/x_n] =$
$Fv(\rho(x)) \odot M[\rho(x_1)/\!/x_1, \ldots, \rho(x_n)/\!/x_n] = Fv(\rho(x)) \odot [\![M]\!]_\rho$ since
$(x \odot M)[\rho(x)/x][\rho(x_1)/x_1] \ldots [\rho(x_n)/x_n] \xrightarrow{\boxslash}$
$(Fv(\rho(x)) \odot M)[\rho(x_1)/x_1] \ldots [\rho(x_n)/x_n] \xrightarrow{\boxslash\!\!\to}$
$Fv(\rho(x)) \odot M[\rho(x_1)/x_1] \ldots [\rho(x_n)/x_n]$.

(v) If $Fv(M) = \{x, y, x_1, \ldots, x_n\}$, then $Fv(z <_y^x M) = \{z, x_1, \ldots, x_n\}$ and
$[\![z <_y^x M]\!]_\rho = (z <_y^x M)[\rho(z)/\!/z, \rho(x_1)/\!/x_1, \ldots, \rho(x_n)/\!/x_n] =$
$(z <_y^x M)[N/\!/z][\rho(x_1)/\!/x_1, \ldots, \rho(x_n)/\!/x_n] =$
$Fv[N] <_{Fv[N_2]}^{Fv[N_1]} M[N_1/\!/x][N_2/\!/y][\rho(x_1)/\!/x_1, \ldots, \rho(x_n)/\!/x_n]$
since $(z <_y^x M)[N/z][\rho(x_1)/x_1] \ldots [\rho(x_n)/x_n] \xrightarrow{\boxslash}$

$(Fv[N] <^{Fv[N_1]}_{Fv[N_2]} M[N_1/x][N_2/y])[\rho(x_1)/x_1]\dots[\rho(x_n)/x_n] \xrightarrow{\not\square}$
$Fv[N] <^{Fv[N_1]}_{Fv[N_2]} M[N_1/x][N_2/y][\rho(x_1)/x_1]\dots[\rho(x_n)/x_n].$

On the other hand, denoting by $\rho' = \rho(N_1/x, N_2/y)$ we obtain

$Fv[N] <^{Fv[N_1]}_{Fv[N_2]} [\![M]\!]_{\rho(N_1/x,N_2/y)} =$
$Fv[N] <^{Fv[N_1]}_{Fv[N_2]} M[\rho'(x)/\!/x, \rho'(y)/\!/y, \rho'(x_1)/\!/x_1, \dots, \rho'(x_n)/\!/x_n] =$
$Fv[N] <^{Fv[N_1]}_{Fv[N_2]} M[N_1/\!/x, N_2/\!/y, \rho(x_1)/\!/x_1, \dots, \rho(x_n)/\!/x_n]$

□

Definition 48.

(i) $\rho \models M : \sigma \iff [\![M]\!]_\rho \in [\![\sigma]\!]$;

(ii) $\rho \models \Gamma \iff (\forall (x:\alpha) \in \Gamma) \quad \rho(x) \in [\![\alpha]\!]$;

(iii) $\Gamma \models M : \sigma \iff (\forall \rho, \rho \models \Gamma \Rightarrow \rho \models M : \sigma)$.

Lemma 49. *Let* $\Gamma \vDash M : \sigma$ *and* $\Delta \vDash M : \tau$*, then*

$$\rho \vDash \Gamma \sqcap \Delta \text{ if and only if } \rho \vDash \Gamma \text{ and } \rho \vDash \Delta.$$

Proof. The proof is a straightforward consequence of the definition of bases intersection $\sqcap$. □

Proposition 50 (Soundness of $\lambda_{\circledR\cap}$)**.** *If* $\Gamma \vdash M : \sigma$*, then* $\Gamma \models M : \sigma$.

Proof. By induction on the derivation of $\Gamma \vdash M : \sigma$.

- The last rule applied is (Ax), i.e.

$$\frac{}{x : \sigma \vdash x : \sigma}\ (Ax)$$

We have to prove $x : \sigma \models x : \sigma$. i.e. $(\forall \rho)\ \rho(x) \in [\![\sigma]\!] \Rightarrow [\![x]\!]_\rho \in [\![\sigma]\!]$. This is trivial since according to Lemma 47(i) $[\![x]\!]_\rho = \rho(x)$.

- The last rule applied is $(\to_I)$, i.e.

$$\frac{\Gamma, x : \alpha \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \alpha \to \sigma}\ (\to_I)$$

By the IH $\Gamma, x:\alpha \models M:\sigma$ (*). Suppose that $\rho \models \Gamma$ and we want to show that $\rho \models \lambda x.M : \alpha \to \sigma$. We have to show that

$$[\![\lambda x.M]\!]_\rho \in [\![\alpha \to \sigma]\!] = [\![\alpha]\!] \longrightarrow [\![\sigma]\!] \quad \text{i.e. } \forall N \in [\![\alpha]\!].\ [\![\lambda x.M]\!]_\rho N \in [\![\sigma]\!].$$

Suppose that $N \in [\![\alpha]\!]$. We have that $\rho(N/x) \models \Gamma, x:\alpha$ (**) since $\rho \models \Gamma$, $x \notin \Gamma$ and $\rho(N/x)(x) = N \in [\![\alpha]\!]$. From (*) and (**) we conclude that $\rho(N/x) \models M:\sigma$, hence we can conclude that $[\![M]\!]_{\rho(N/x)} \in [\![\sigma]\!]$. Using Lemma 47(iii) we get $[\![\lambda x.M]\!]_\rho N \to_\beta [\![M]\!]_\rho[N/\!\!/x] = [\![M]\!]_{\rho(N/x)}$. Since $[\![M]\!]_{\rho(N/x)} \in [\![\sigma]\!]$ and $[\![\sigma]\!]$ is ®-saturated, we obtain $[\![\lambda x.M]\!]_\rho N \in [\![\sigma]\!]$.

- The last rule applied is $(\to_E)$, i.e.

$$\frac{\Gamma \vdash M : \cap_i^n \tau_i \to \sigma \quad \Delta_0 \vdash N:\tau_0 \ \ldots\ \Delta_n \vdash N:\tau_n}{\Gamma, \Delta_0^\top \sqcap \Delta_1 \sqcap \ldots \sqcap \Delta_n \vdash MN : \sigma}\ (\to_E)$$

Let ρ be any valuation. Assuming that $\Gamma \vdash M : \cap_i^n \tau_i \to \sigma, \Delta_0 \vdash N:\tau_0, \ldots, \Delta_n \vdash N:\tau_n$, we have to prove that if $\rho \vDash \Gamma, \Delta_0^\top \sqcap \Delta_1 \sqcap \ldots \sqcap \Delta_n$, then $\rho \vDash MN:\sigma$, i.e. $[\![MN]\!]_\rho \in [\![\sigma]\!]$.

By IH, $\Gamma \models M : \cap_i^n \tau_i \to \sigma$ and $\Delta_0 \models N:\tau_0, \ldots, \Delta_n \models N:\tau_n$. Assume that $\rho \models \Gamma, \Delta_0^\top \sqcap \Delta_1 \sqcap \ldots \sqcap \Delta_n$. This means that $\rho \models \Gamma$ and $\rho \models \Delta_0^\top \sqcap \Delta_1 \sqcap \ldots \sqcap \Delta_n$. From $\rho \models \Gamma$ we deduce by Definition 48 (iii) $\rho \models M : \cap_i^n \tau_i \to \sigma$ and by Definition 48 (i) $[\![M]\!]_\rho \in [\![\cap_i^n \tau_i \to \sigma]\!]$. By Definition 46 $[\![M]\!]_\rho \in \bigcap_i^n [\![\tau_i]\!] \longrightarrow [\![\sigma]\!]$ (*). Using Lemma 49 $\rho \models \Delta_0^\top \sqcap \Delta_1 \sqcap \ldots \sqcap \Delta_n$ implies $(\rho \models \Delta_0^\top) \wedge (\bigwedge_{i=1}^n \rho \models \Delta_i)$, hence by Definition 48 (i) and (iii) we get $([\![N]\!]_\rho \in [\![\top]\!]) \wedge \bigwedge_{i=1}^n ([\![N]\!]_\rho \in [\![\tau_i]\!])$, i.e. $[\![N]\!]_\rho \in \mathcal{SN} \cap \cap_i^n [\![\tau_i]\!] = \cap_i^n [\![\tau_i]\!]$ (**), since $[\![\tau_i]\!] \subseteq \mathcal{SN}$ by Proposition 45(iv). From (*) and (**), using Definition 41 of $\longrightarrow$, we can conclude that $[\![M]\!]_\rho [\![N]\!]_\rho \in [\![\sigma]\!]$. Using Lemma 47(ii) we can conclude that $[\![MN]\!]_\rho = [\![M]\!]_\rho [\![N]\!]_\rho \in [\![\sigma]\!]$ and by Definition 48 (i) $\rho \vDash MN:\sigma$.

- The last rule applied is $(Thin)$, i.e.,

$$\frac{\Gamma \vdash M:\sigma}{\Gamma, x:\top \vdash x \odot M : \sigma}\ (Thin)$$

By the IH $\Gamma \models M:\sigma$. Suppose that $\rho \models \Gamma, x:\top \Leftrightarrow \rho \models \Gamma$ and $\rho \models x:\top$. From $\rho \models \Gamma$ we obtain $[\![M]\!]_\rho \in [\![\sigma]\!]$. Using multiple times the thinning property $\mathsf{THIN}([\![\sigma]\!])$ and Lemma 47(iv) we obtain $Fv(\rho(x)) \odot [\![M]\!]_\rho = [\![x \odot M]\!]_\rho \in [\![\sigma]\!]$, since $Fv(\rho(x)) \cap Fv([\![M]\!]_\rho) = \emptyset$.

- The last rule applied is $(Cont)$, i.e.,

$$\frac{\Gamma, x:\alpha, y:\beta \vdash M:\sigma}{\Gamma, z:\alpha\cap\beta \vdash z <_y^x M:\sigma}\ (Cont)$$

By the IH $\Gamma, x:\alpha, y:\beta \models M:\sigma$. Suppose that $\rho \models \Gamma, z:\alpha\cap\beta$.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 valuation ρ' such that $\rho' = \rho(N_1/x, N_2/y)$, where N_1 and N_2 are obtained by renaming the free variables of N. Then $\rho' \models \Gamma, x:\alpha, y:\beta$ since $x, y \notin Dom(\Gamma)$, $N_1 \in [\![\alpha]\!]$ and $N_2 \in [\![\beta]\!]$. By the IH $[\![M]\!]_{\rho'} = [\![M]\!]_{\rho(N_1/x,N_2/y)} \in [\![\sigma]\!]$. Using the contraction property $\mathsf{CONT}([\![\sigma]\!])$ and Lemma 47(v) we have that $Fv(N) <^{Fv(N_1)}_{Fv(N_2)} [\![M]\!]_{\rho(N_1/x,N_2/y)} = [\![z <_y^x M]\!]_\rho \in [\![\sigma]\!]$.

□

Theorem 51 ($\mathcal{SN}$ for $\lambda_{\circledR}\cap$)**.** *If $\Gamma \vdash M:\sigma$, then M is strongly normalising, i.e. $M \in \mathcal{SN}$.*

Proof. Suppose $\Gamma \vdash M:\sigma$. By Proposition 50 $\Gamma \models M:\sigma$. According to Definition 48(iii), this means that $(\forall\rho)\ \rho \models \Gamma \quad \Rightarrow \quad \rho \models M:\sigma$. We can choose a particular $\rho_0(x) = x$ for all $x \in \mathtt{var}$. By Proposition 45(iv), $[\![\sigma]\!]$ is ®-saturated for each type σ, hence $[\![x]\!]_{\rho_0} = x \in [\![\sigma]\!]$ (variable condition for $n = 0$). Therefore, $\rho_0 \models \Gamma$ and we can conclude that $[\![M]\!]_{\rho_0} \in [\![\sigma]\!]$. On the other hand, $M = [\![M]\!]_{\rho_0}$ and $[\![\sigma]\!] \subseteq \mathcal{SN}$ (Proposition 45), hence $M \in \mathcal{SN}$. □

Finally, we can give a characterisation of strong normalisation in $\lambda_{\circledR}$-calculus.

Theorem 52. *In $\lambda_{\circledR}$-calculus, the term M is strongly normalising if and only if it is typeable in $\lambda_{\circledR}\cap$.*

Proof. Immediate consequence of Theorems 51 and 40. □

4 Related work and conclusions

The idea to control the use of variables can be traced back to Church's λI-calculus [4] and Klop's extension of λ-calculus [35]. Currently, there are several different lines of research in resource aware term calculi.

Van Oostrom [54] and later Kesner and Lengrand [30], applying ideas from linear logic [29], proposed to extend λ-calculus with explicit substitution [30] with operators to control the use of variables (resources). Their linear λlxr-calculus is an extension of the λx-calculus [9, 47] with operators for linear substitution, erasure and duplication, preserving at the same time confluence and full composition of explicit substitutions. The simply typed version of this calculus corresponds to the intuitionistic fragment of linear logic proof-nets, according to Curry-Howard correspondence, and it enjoys strong normalisation and subject reduction. Generalising this approach, Kesner and Renaud [31, 32] developed the *prismoid of resources*, a system of eight calculi parametric over the explicit and implicit treatment of substitution, erasure and duplication.

On the other hand, process calculi and their relation to λ-calculus by Boudol [10] initialised investigations in resource aware non-deterministic λ-calculus with multiplicities and a generalised notion of application [11]. The theory was connected to linear logic via differential λ-calculus by Ehrhard and Regnier in [19] and typed with non-idempotent intersection types by Pagani and Ronchi Della Rocha in [43]. An account of this approach is given in [2].

Resource control in sequent calculus corresponding to classical logic was proposed by Žunić in [56]. Resource control in sequent λ-calculus was investigated in [25].

Intersection types in the presence of resource control were first introduced in [26]. Later on non-idempotent intersection types for λlxr-calculus were introduced by Bernadet and Lengrand in [8]. Their proof of strong normalisation takes advantage of intersection types being non-idempotent.

Our contribution extends the work of [26], accordingly we follow the notation of [56] and [26], along the lines of [54]. We have proposed an intersection type assignment system for the resource control lambda calculus $\lambda_{\circledR}$, which gives a complete characterisation of strongly normalising terms of the $\lambda_{\circledR}$-calculus. The proofs do not rely on any assumption about idempotence, hence they can be applied both to idempotent and non-idempotent intersection types.

This paper expands the range of the intersection type techniques and combines different methods in the strict type environment. It should be noticed that the strict control on the way variables are introduced determines the way terms are typed in a given environment. Basically, in a given environment no irrelevant intersection types are introduced. The flexibility on the choice of a type for a term, as it is used in rule $(\rightarrow_E)$ in Figure 8, comes essentially from the choice one has in invoking the axiom.

The presented calculus is a good candidate to investigate the computational

content of substructural logics [49] in natural deduction style and relation to substructural type systems [55]. The motivation for these logics comes from philosophy (Relevant Logics), linguistics (Lambek Calculus), computing (Linear Logic). Since the basic idea of resource control is to explicitly handle structural rules, the control operators could be used to handle the absence of (some) structural rules in substructural logics such as thinning, weakening, contraction, commutativity, associativity. This would be an interesting direction for further research. Another direction involves the investigation of the use of intersection types, being a powerful means for building models of lambda calculus [6, 16], in constructing models for substructural type systems. Finally, one may wonder how the strict control on the duplication and the erasure of variables influences the type reconstruction of terms [12, 33].

Acknowledgements: We would like to thank anonymous referees of a previous version of this paper for their careful reading and many valuable comments, which helped us to improve the paper. We would also like to thank Dragiša Žunić for participating in the earlier stages of the work. This work is partially supported by the Serbian Ministry of Science - project ON174026 and by a bilateral project between Serbia and France within the "Pavle Savić" framework.

References

[1] S. Abramsky. Computational interpretations of linear logic. *Theoretical Computer Science*, 111(1&2):3–57, 1993.

[2] Sandra Alves, Maribel Fernández, Mário Florido, and Ian Mackie. Linearity: A roadmap. *Journal of Logic and Computation*, 24(3):513–529, 2014.

[3] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, UK, 1998.

[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. *Journal of Symbolic 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 Marc Bezem and Jan Friso Groote, editors, *1st International Conference on Typed Lambda Calculus, TLCA '93*, volume 664 of *Lecture Notes in Computer Science*, pages 75–90. Springer, 1993.

[8] Alexis Bernadet and Stéphane Lengrand. Non-idempotent intersection types and strong normalisation. *Logical Methods in Computer Science*, 9(4), 2013.

[9] 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.

[10] G. Boudol. The lambda-calculus with multiplicities (abstract). In E. Best, editor, *4th International Conference on Concurrency Theory, CONCUR '93*, volume 715 of *Lecture Notes in Computer Science*, pages 1–6. Springer, 1993.

[11] G. Boudol, P.-L. Curien, and C. Lavatelli. A semantics for lambda calculi with resources. *Mathematical Structures in Computer Science*, 9(4):437–482, 1999.

[12] G. Boudol and P. Zimmer. On type inference in the intersection type discipline. *Electronic Notes in Theoretical Computer Science*, 136:23–42, 2005.

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

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

[15] M. Dezani-Ciancaglini and S. Ghilezan. Two behavioural lambda models. In H. Geuvers and F. Wiedijk, editors, *Types for Proofs and Programs*, volume 2646 of *Lecture Notes in Computer Science*, pages 127–147. Springer, 2003.

[16] M. Dezani-Ciancaglini, S. Ghilezan, and S. Likavec. Behavioural Inverse Limit Models. *Theoretical Computer Science*, 316(1–3):49–74, 2004.

[17] M. Dezani-Ciancaglini, F. Honsell, and Y. Motohama. Compositional characterization of λ-terms using intersection types. In *25th International Symposium on Mathematical Foundations of Computer Science, MFCS '00*, volume 1893 of *Lecture Notes in Computer Science*, pages 304–314. Springer, 2000.

[18] D. J. Dougherty, S. Ghilezan, and P. Lescanne. Characterizing strong normalization in the Curien-Herbelin symmetric lambda calculus: extending the Coppo-Dezani heritage. *Theoretical Computer Science*, 398:114–128, 2008.

[19] T. Ehrhard and L. Regnier. The differential lambda-calculus. *Theoretical Computer Science*, 309(1-3):1–41, 2003.

[20] José Espírito Santo, Jelena Ivetic, and Silvia Likavec. Characterising strongly normalising intuitionistic terms. *Fundamenta Informaticae*, 121(1-4):83–120, 2012.

[21] J. Gallier. Typing untyped λ-terms, or reducibility strikes again! *Annals of Pure and Applied Logic*, 91:231–270, 1998.

[22] G. Gentzen. Untersuchungen über das logische schließen. I. *Mathematische Zeitschrift*, 39:176–210, 1934.

[23] G. Gentzen. Unterschungen über das logische Schliessen, Math Z. 39 (1935), 176–210. In M.E. Szabo, editor, *Collected papers of Gerhard Gentzen*, pages 68–131. North-Holland, 1969.

[24] S. Ghilezan. Strong normalization and typability with intersection types. *Notre Dame Journal of Formal Logic*, 37(1):44–52, 1996.

[25] S. Ghilezan, J. Ivetić, P. Lescanne, and D. Žunić. Intuitionistic sequent-style calculus with explicit structural rules. In Nick Bezhanishvili, Sebastian Löbner, Kerstin Schwabe, and Luca Spada, editors, *8th International Tbilisi Symposium on Language, Logic and Computation*, volume 6618 of *Lecture Notes in Computer Science*, pages 101–124. Springer, 2011.

[26] Silvia Ghilezan, Jelena Ivetić, Pierre Lescanne, and Silvia Likavec. Intersection types for the resource control lambda calculi. In Antonio Cerone and Pekka Pihlajasaari, editors, *8th International Colloquium on Theoretical Aspects of Computing, ICTAC '11*, volume 6916 of *Lecture Notes in Computer Science*, pages 116–134. Springer, 2011.

[27] Silvia Ghilezan and Silvia Likavec. Reducibility: A Ubiquitous Method in Lambda Calculus with Intersection Types. In Steffen van Bakel, editor, *ITRS '02*, volume 70 of *Electronic Notes in Theoretical Computer Science*, pages 106–123, 2002.

[28] J.-Y. Girard. Une extension de l'interprétation de Gödel à l'analyse, et son application à l'elimination des coupures dans l'analyse et la théorie des types. In J. E. Fenstad, editor, *2nd Scandinavian Logic Symposium*, pages 63–92. North-Holland, 1971.

[29] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[30] D. Kesner and S. Lengrand. Resource operators for lambda-calculus. *Information and Computation*, 205(4):419–473, 2007.

[31] D. Kesner and F. Renaud. The prismoid of resources. In R. Královič and D. Niwiński, editors, *34th International Symposium on Mathematical Foundations of Computer Science, MFCS '09*, volume 5734 of *Lecture Notes in Computer Science*, pages 464–476. Springer, 2009.

[32] D. Kesner and F. Renaud. A prismoid framework for languages with resources. *Theoretical Computer Science*, 412(37):4867–4892, 2011.

[33] A. J. Kfoury and J. B. Wells. Principality and type inference for intersection types using expansion variables. *Theoretical Computer Science*, 311(1-3):1–70, 2004.

[34] 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 *Lecture Notes in Computer Science*, pages 257–272. Springer, 2007.

[35] Jan Willem Klop. *Combinatory reduction systems*. PhD thesis, 1980.

[36] G. Koletsos. Church-Rosser theorem for typed functionals. *Journal of Symbolic Logic*, 50:782–790, 1985.

[37] J.-L. Krivine. *Lambda-calcul types et modèles*. Masson, Paris, 1990.

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

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

[40] J. C. Mitchell. Type systems for programming languages. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, Volume B, pages 415–431. Elsevier, Amsterdam, 1990.

[41] J. C. Mitchell. *Foundation for Programmimg Languages*. MIT Press, Boston, 1996.

[42] P. M. Neergaard. Theoretical pearls: A bargain for intersection types: a simple strong normalization proof. *Journal of Functional Programming*, 15(5):669–677, 2005.

[43] M. Pagani and S. Ronchi Della 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 *Lecture Notes in Computer Science*, pages 358–373. Springer, 2010.

[44] 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.

[45] K H. Rose. CRSX - Combinatory Reduction Systems with Extensions. In Manfred 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.

[46] K. H. Rose. Implementation Tricks That Make CRSX Tick. Talk at IFIP 1.6 workshop, 6th International Conference on Rewriting, Deduction, and Programming, RDP '11, 2011.

[47] Kristoffer Rose, Roel Bloo, and Frédéric Lang. On explicit substitution with names. *Journal of Automated Reasoning*, pages 1–26, 2011.

[48] 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 *Lecture Notes in Computer Science*, pages 398–410. Springer, 1978.

[49] P. Schroeder-Heister and K. Došen. *Substructural Logics*. Oxford University Press, UK, 1993.

[50] R. Statman. Logical relations and the typed λ-calculus. *Information and Control*, 65:85–97, 1985.

[51] W. W. Tait. Intensional interpretations of functionals of finite type I. *Journal of Symbolic Logic*, 32:198–212, 1967.

[52] W. W. Tait. A realizability interpretation of the theory of species. In R. Parikh, editor, *Logic Colloquium*, volume 453 of *Lecture Notes in Mathematics*, pages 240–251. Springer, 1975.

[53] S. van Bakel. Complete restrictions of the intersection type discipline. *Theoretical Computer Science*, 102(1):135–163, 1992.

[54] V. van Oostrom. Net-calculus. Course notes, Utrecht University, 2001.

[55] David Walker. Substructural type systems. In Benjamin Pierce, editor, *Advanced Topics in Types and Programming Languages*, pages 3–44. MIT Press, Cambridge, 2005.

[56] D. Žunić. *Computing with sequents and diagrams in classical logic - calculi* ${}^{*}\mathcal{X}$, ${}^{d}\mathcal{X}$ *and* ${}^{\copyright}\mathcal{X}$. Phd thesis, École Normale Supérieure de Lyon, 2007.