Complexity of strongly normalising λ-terms via non-idempotent intersection types

Alexis Bernadet[1,2] and Stéphane Lengrand[1,3]

[1] École Polytechnique, France
[2] École Normale Supérieur de Cachan, France
[3] CNRS, France
{lengrand,bernadet}@lix.polytechnique.fr

Abstract. We present a typing system for the λ-calculus, with non-idempotent intersection types. As it is the case in (some) systems with idempotent intersections, a λ-term is typable if and only if it is strongly normalising. Non-idempotency brings some further information into typing trees, such as a bound on the longest β-reduction sequence reducing a term to its normal form.
We actually present these results in Klop's extension of λ-calculus, where the bound that is read in the typing tree of a term is refined into an exact measure of the longest reduction sequence.
This complexity result is, for longest reduction sequences, the counterpart of de Carvalho's result for linear head-reduction sequences.

1 Introduction

Intersection types were introduced in [CD78], extending the simply-typed λ-calculus with a notion of finite polymorphism. This is achieved by a new construct $A \cap B$ in the syntax of types and new typing rules such as:

$$\frac{M : A \quad M : B}{M : A \cap B}$$

where $M : A$ denotes that a term M is of type A.
One of the motivations was to characterise strongly normalising (SN) λ-terms, namely the property that a λ-term can be typed if and only if it is strongly normalising. Variants of systems using intersection types have been studied to characterise other evaluation properties of λ-terms and served as the basis of corresponding semantics [Lei86,Ghi96,DCHM00,CS07].
This paper refines with quantitative information the property that typability characterises strong normalisation. Since strong normalisation ensures that all reduction sequences are finite, we are naturally interested in identifying the length of the longest reduction sequence. We do this with a typing system that is very sensitive to the usage of resources when λ-terms are reduced.
This system results from a long line of research inspired by Linear Logic [Gir87]. The usual logical connectives of, say, classical and intuitionistic logic, are decomposed therein into finer-grained connectives, separating a *linear* part from a part that controls how and when the structural rules of *contraction* and *weakening* are used in proofs. This can be seen as resource management when hypotheses, or more generally logical formulae, are considered as resource.
The Curry-Howard correspondence, which originated in the context of intuitionistic logic [How80], can be adapted to Linear Logic [Abr93,BBdH93], whose resource-awareness translates to a control of resources in the execution of programs (in the usual computational sense). From this, have emerged some versions of linear logic that capture polytime functions [BM03,Laf04,GR07]. Also from this has emerged a theory of λ-calculus with resource, with semantical support (such as the differential λ-calculus) [ER03,BEM10]. In this line of research,

de Carvalho [dC05,dC09] obtained interesting measures of reduction lengths in the λ-calculus by means of *non-idempotent* intersection types (as pionnered by [KW99,NM04]).
Intersections were originally introduced as idempotent, with the equation $A \cap A = A$ either as an explicit quotient or as a consequence of the system. This corresponds to the understanding of the judgement $M : A \cap B$ as follows: M can be used as data of type A or data of type B. But the meaning of $M : A \cap B$ can be strengthened in that M **will** be used **once** as data of type A and **once** as data of type B. With this understanding, $A \cap A \neq A$, and dropping idempotency of intersections is thus a natural way to study control of resources and complexity.
Using this, de Carvalho [dC09] has shown a correspondence between the size of the typing derivation tree and the number of steps taken by a Krivine machine to reduce the term. This relates to the length of linear head-reductions, but if we remain in the realm of intersection systems that characterise strong normalisation, then the more interesting measure is the length of the longest reduction sequence. In this paper we get a result similar to de Carvalho's, but with the measure corresponding to strong normalisation.
First we define a system with non-idempotent intersection types. Then we prove that if a term is typable then it is SN (soundness) and that if a term is SN then it is typable (correctness). As opposed to idempotent intersection types, the proof of correctness is very direct: we use a simple measure on typing trees (the easy proof differs from that in [Val01], in that the typing system itself does not perform β-reduction).
The proof of soundness gives us immediatly a bound of the maximum number of β-reductions. So we have an inequality result for complexity. We would like an equality result.
One of the reasons why we only have an inequality result is because, in a β-reduction, the argument of the β-redex may disappear (we call this weakening). One simple way to avoid the weakening problem without blocking computation is to use Klop's extension of λ-calculus :

$$M, N ::= \ \ldots \mid [M, N]$$

In $[M, N]$, N is a sub-term that was meant to be erased. To avoid weakenings, we can replace every term $\lambda x.M$ such that $x \notin FV(M)$ with $\lambda x.[M, x]$.
We refer the reader to [Sør97,Xi97] for a survey on different techniques based on the λI-calculus to infer normalisation properties. Intersection types in the framework of Church-Klop's calculus have been studied in e.g. [DCT07], but, to our knowledge, they were always considered idempotent. Hence, the quantitative analysis provided by non-idempotency was not carried out.
In order to obtain the complexity result we want, we still have to expect a property on typing trees: *optimality*. This property is mostly on the "interface types". It is not too restrictive because the completeness theorem produces such typing trees, but it is still quite so because we can prove that for each term, the shape of such a derivation tree is unique (*principality*).
We can then prove that if π is a optimal-principal typing tree of M then we can read off π the **exact** length of the longest reduction sequence starting from M.

2 Syntax and typing

In this section we present the calculus and the typing system that we are going to use.

2.1 Lambda Calculus with Klop's extension

As said in the introduction, the language that we are going to type is the pure λ-calculus extended with Klop's construct [Klo80]:

Definition 1 (Syntax and reduction rules).

– *Terms are defined by the following grammar*

$$M, N ::= \ x \mid \lambda x.M \mid MN \mid [M, N]$$

Unless otherwise stated, we do not consider any restriction on this syntax. Sometimes we write $N\ \overrightarrow{M_i}$ *for* $N\ M_1 \dots M_n$ *(when* $\overrightarrow{M_i}$ *is the vector of terms* $M_1 \dots M_n$*).*
The free variables $fv(M)$ *of a term* M *are defined as usual and terms are considered up to* α*-equivalence.*

– *The reduction rules are* β*-reduction and* π*-reduction (see e.g. [Klo80]):*

$$\begin{array}{ll} \beta & (\lambda x.M)\ N \longrightarrow\ M\{x := N\} \\ \pi & [M_1, N]M_2 \longrightarrow\ [M_1\ M_2, N] \end{array}$$

If S *is a rule (such as* β *or* π*), or a system of rules (like* $\beta\pi$*), we write* $M \longrightarrow_S N$ *for the congruent closure of the (system of) rule(s).*

An interesting fragment of the calculus is Church-Klop's λI [Klo80]:

Definition 2 (λI). *A term* M *is in the* λI*-fragment if any abstraction* $\lambda x.N$ *occurring in* M *is such that* $x \in fv(N)$*.*

Remark 1. The λI-fragment is stable under substitution and under $\longrightarrow_{\beta\pi}$. If $M \longrightarrow_{\beta\pi} N$ then $fv(M) = fv(N)$ and $P\{x := M\} \longrightarrow^{+}_{\beta\pi} P\{x := N\}$ provided that $x \in fv(P)$.

Another obvious fragment is the pure λ-calculus.[4] Klop's construct is only useful here for the complexity results of section 5; we do not need it for soundness or completeness (section 3). So if one is interested only in the pure λ-calculus, it is possible to follow the proofs of these theorems while ignoring Klop's construct. As we will see later some theorems are not true for every reduction (for example, the subject expansion property), so we define the reduction $M \hookrightarrow_N M'$, where N is either a term or ϵ (a dummy placeholder) as follows:

Definition 3 (A restricted reduction relation).

$$\frac{x \in FV(M)}{(\lambda x.M)N \hookrightarrow_\epsilon M\{x := N\}} \qquad \frac{x \notin FV(M)}{(\lambda x.M)N \hookrightarrow_N M}$$

$$\frac{M_1 \hookrightarrow_N M_1'}{M_1M_2 \hookrightarrow_N M_1'M_2} \qquad \frac{M_2 \hookrightarrow_N M_2'}{M_1M_2' \hookrightarrow_N M_1M_2'}$$

$$\frac{M \hookrightarrow_N M\ x \notin FV(N)}{\lambda x.M \hookrightarrow_N \lambda x.M} \qquad \frac{}{[M_1, N]M_2 \hookrightarrow_\epsilon [M_1\ M_2, N]}$$

$$\frac{M_1 \hookrightarrow_N M_1'}{[M_1, M_2] \hookrightarrow_N [M_1', M_2]} \qquad \frac{M_2 \hookrightarrow_N M_2'}{[M_1, M_2] \hookrightarrow_N [M_1, M_2']}$$

Fig. 1. $\hookrightarrow$-reduction

Fig. 1 formalises the reduction relation that can be be intuitively described as follows: $M \hookrightarrow_N M'$ *if and only if* $M \longrightarrow_{\beta\pi} M'$ *such that:*

– *either the reduction* $M \longrightarrow_{\beta\pi} M'$ *does not erase anything in which case* $N = \epsilon$*.*
– *or the reduction* $M \longrightarrow_{\beta\pi} M'$ *erases the term* N *within* M *and the free variables of* N *are not bound by binders in* M*.*

Remark 2. If M is a λI term then every reduction is in $\hookrightarrow_\epsilon$.

[4] This motivates our choice of sticking to the reduction rules of [Klo80] rather than opting for the variant in [Bou03] where β-reduction can generate new instances of Klop's construct.

2.2 Intersection types and contexts

Definition 4 (Intersection types).
Our intersection types are defined by the following grammar :

$$\begin{array}{lll} A, B ::= & F \mid A \cap B & \\ F, G ::= & \tau \mid A \to F & \\ U, V ::= & \omega \mid A & \textit{(General types)} \end{array}$$

We consider the types up to associativity and commutativity of intersections. The notation $U \cap V$ extends the intersection construct to general types using the following equations:

$$A \cap \omega = \omega \cap A = A \qquad \omega \cap \omega = \omega$$

As opposed to [CD78,CD80] we do not have idempotency $A = A \cap A$.

Remark 3.
- It would be possible to formalise the results of this paper without using general types, but then in many definitions and proofs we would have to deal with two or three cases instead of one.
- Notice that, by construction, if $A \to B$ is a type then B is not an intersection. This limitation, which corresponds to the *strict types* of [vB92], is useful for the key property of separation (Lemma 1).

Definition 5 (Type inclusion). *We define inclusion on types as follows: $U \subseteq V$ if $V = \omega$, or $U = V$, or $U = A$ and $V = B$ with $A = B \cap C$ for some C.*

Notice that this definition of inclusion is weaker than the traditional one originating from [BCDC83]. Indeed, inclusions between domains and co-domains of function types do not induce inclusions between function types.

Remark 4. $\subseteq$ is a partial order.

Definition 6 (Contexts).
- *A context Γ is a total map from variables $(x, y, z, \ldots)$ to general types $(U, V, \ldots)$ that has finite support, i.e. such that $\{x \mid \Gamma(x) \neq \omega\}$ is finite.*
- *The expression $(x_1 : U_1, ..., x_n : U_n)$, where for all i, j, $x_i \neq x_j$, is defined as the context Γ such that:*
  - *For all i, $\Gamma(x_i) = U_i$*
  - *$\Gamma(y) = \omega$ for any other y*
- *Given two contexts Γ and Δ,*
  $\Gamma \cap \Delta$ is defined pointwise as: for all x, $(\Gamma \cap \Delta)(x) = \Gamma(x) \cap \Delta(x)$, and we write $\Gamma \subseteq \Delta$ for either $\Gamma = \Delta$ or there exists Γ' such that $\Gamma = \Gamma' \cap \Delta$

Remark 5.
- $\subseteq$ is a partial order on contexts.
- $\Gamma \subseteq \Delta$ if and only if for all x, $\Gamma(x) \subseteq \Delta(x)$

2.3 Typing system and its basic properties

Definition 7 (Typing system).
- *Fig. 2 inductively defines the derivability of typing judgements, of the form $\Gamma \vdash M : U$. Typing trees will be denoted π, $\pi' \ldots$*
- *To prove strong normalisation we will use a simple measure on typing trees: we just count the number of occurences of rule (App).*
  So we write $\Gamma \vdash^n M : U$ (resp. $\Gamma \vdash^{\leq n} M : U$, resp. $\Gamma \vdash^{<n} M : U$) if there exists a typing tree π concluding $\Gamma \vdash M : U$ and in π there are exactly (resp. less than or equal to, resp. less than) n occurences of rule (App).
- *We say that a term M is typable if there exist Γ and A such that $\Gamma \vdash M : A$*

$$\frac{}{x{:}F \vdash x{:}F} \qquad \frac{\Gamma \vdash M{:}A \quad \Delta \vdash M{:}B}{\Gamma \cap \Delta \vdash M{:}A \cap B}$$

$$\frac{\Gamma, x{:}U \vdash M{:}F \quad A \subseteq U}{\Gamma \vdash \lambda x.M{:}A{\to}F} \qquad \frac{\Gamma \vdash M{:}A{\to}B \quad \Delta \vdash N{:}A}{\Gamma \cap \Delta \vdash MN{:}B}\ (App)$$

$$\frac{\Gamma \vdash M{:}F \quad \Delta \vdash N{:}A}{\Gamma \cap \Delta \vdash [M,N]{:}F} \qquad \frac{}{\vdash M{:}\omega}$$

Fig. 2. Typing system

In the rest of this section and for the proof of subject reduction we can ignore everything about the measure: but taking it into account does not complicate the proofs.

Remark 6. If $\Gamma \vdash^n M{:}U$ and $\Delta \vdash^m M{:}V$ then $\Gamma \cap \Delta \vdash^{n+m} M{:}U \cap V$

Lemma 1 (Separation). *If $\Gamma \vdash^n M{:}U_1 \cap U_2$ then there exist Γ_1, Γ_2, n_1 and n_2 such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, and $\Gamma_1 \vdash^{n_1} M{:}U_1$ and $\Gamma_2 \vdash^{n_2} M{:}U_2$.*

Proof. By induction on the typing tree.
- If $U_1 = \omega$ or $U_2 = \omega$ it is trivial.
- If $\dfrac{\Gamma \vdash^n M{:}C \quad \Delta \vdash^m M{:}D}{\Gamma \cap \Delta \vdash^{n+m} M{:}C \cap D}$ and $C \cap D = A_1 \cap A_2$ then there exist U_1, U_2, V_1, V_2 such that $C = U_1 \cap U_2$, $D = V_1 \cap V_2$, $A_1 = U_1 \cap V_1$ and $A_2 = U_2 \cap V_2$. By induction, there exist n_1, n_2, m_1, m_2, Γ_1, Γ_2, Δ_1, Δ_2 such that $n = n_1 + n_2$, $m = m_1 + m_2$, $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Delta = \Delta_1 \cap \Delta_2$, $\Gamma_1 \vdash^{n_1} M : U_1$, $\Gamma_2 \vdash^{n_2} M{:}U_2$, $\Delta_1 \vdash^{m_1} M{:}V_1$, $\Delta_2 \vdash^{m_2} M{:}V_2$.
  So we have $\Gamma_1 \cap \Delta_1 \vdash^{n_1+m_1} M{:}A_1$ and $\Gamma_2 \cap \Delta_2 \vdash^{n_2+m_2} M{:}A_2$.
- All the other cases are impossible, especially the application rule: if $A{\to}B$ is a type, then B is not an intersection. This is why we used this restriction in the begining.

A useful property can be inferred from the separation property:

Corollary 1 (Weakening). *If $\Gamma \vdash^n M : U$ and $U \subseteq V$ then there exists Γ' such that $\Gamma \subseteq \Gamma'$ and $\Gamma' \vdash^{\leq n} M{:}V$.*

3 Soundness and completeness of the typing system w.r.t. strong normalisation

In this section we prove that a term is typable if and only if it is strongly normalising.

3.1 Soundness

Here we prove that typed terms are strongly normalising.

Lemma 2 (Typing of substitution). *If $\Gamma, x{:}U \vdash^{n_1} M{:}V$ and $\Delta \vdash^{n_2} N{:}U$ then $\Gamma \cap \Delta \vdash^{n_1+n_2} M\{x := N\}{:}V$.*

Proof. By induction on the typing tree of M.
- For the variable rule it is trivial.

– For $\vdash^0 M:\omega$ with $(\Gamma, x:U) = ()$, $n_1 = 0$, $V = \omega$. So we have $U = \omega$, so $n_2 = 0$ and $\Delta = ()$. So we can conclude
– For $\dfrac{\Gamma, x:U, y:W \vdash^{n_1} M_1:F \quad A \subseteq W}{\Gamma, x:U \vdash^{n_1} \lambda y.M_1 : A \to F}$ with $M = \lambda y.M_1$, $V = A \to F$, $x \neq y$, $y \notin FV(N)$. From the induction hypothesis we have $\Gamma, y:W \vdash^{n_1+n_2} M_1\{x := N\}:F$ so we can conclude.
– For $\dfrac{\Gamma_1, x:U_1 \vdash^{m_1} M:A \quad \Gamma_2, x:U_2 \vdash^{m_2} M:B}{\Gamma_1 \cap \Gamma_2, x:U_1 \cap U_2 \vdash^{m_1+m_2} M:A \cap B}$ with $n_1 = m_1 + m_2$, $U = U_1 \cap U_2$, $V = A \cap B$. By the use the Lemma 1, there exists Δ_1, Δ_2, m_3, m_4, such that $n_2 = m_3 + m_4$, $\Delta = \Delta_1 \cap \Delta_2$, $\Delta_1 \vdash^{m_3} N:U_1$, $\Delta_2 \vdash^{m_4} N:U_2$. By the induction hypothesis we have $\Gamma_1 \cap \Delta_1 \vdash^{m_1+m_3} M\{x := N\}:A$ and $\Gamma_2 \cap \Delta_2 \vdash^{m_2+m_4} M\{x := N\}:B$ so we can conclude.
– For the application rule and Klop' construct rule we adapt the proof for the intersection rule.

Theorem 1 (Subject Reduction for β).
If $\Gamma \vdash^n M:A$ and $M \longrightarrow_\beta M'$ then there exists Γ' such that $\Gamma' \vdash^{<n} M':A$ and $\Gamma \subseteq \Gamma'$.

Proof. First by induction on $M \longrightarrow_\beta M'$ then by induction on A.

– If A is an intersection then we use the Lemma 1.
– For $\dfrac{}{(\lambda x.M_1)M_2 \longrightarrow_\beta M_1\{x := M_2\}}$ with A not an intersection then there exist B, Γ_1, Γ_2, n_1, n_2 such that $n = n_1+n_2+1$, $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash^{n_1} \lambda x.M_1 : B \to A$ and $\Gamma_2 \vdash^{n_2} M_2:B$. So there exists U such that $B \subseteq U$ and $\Gamma_1, x:U \vdash^{n_1} M_1:A$. So, from Corollary 1, there exists Δ such that $\Gamma_2 \subseteq \Delta$ and $\Delta \vdash^{\leq n_2} M_2:U$. So by the use of the previous lemma we can conclude.
– If $\dfrac{M \longrightarrow_\beta M'}{\lambda x.M \longrightarrow_\beta \lambda x.M'}$ with A is not an intersection, then there exist U, A_2, A_3 such that $A = A_2 \to A_3$, $A_2 \subseteq U$ $\Gamma, x : U \vdash^n M:A_3$. By the induction hyptothesis there exist m, Δ, V such that $\Gamma \subseteq \Delta$, $m < n$, $U \subseteq V$ and $\Delta, x : V \vdash^m M':A_3$. So we have $A_2 \subseteq V$, so we have $\Delta \vdash^m \lambda x.M':A_2 \to A_3$
– The other cases are straightforward.

Lemma 3 (Subject Reduction for π). *If $\Gamma \vdash^n M:A$ and $M \longrightarrow_\pi M'$ then $\Gamma \vdash^n M':A$.*

Proof. Again, by induction first on A then on $M \longrightarrow_\pi M'$. All cases are straightforward.

Corollary 2 (Soundness). *If $\Gamma \vdash M:A$ then M is SN.*

Proof. The measure is decreased by β-reduction (Theorem 1), is invariant under π-reduction (Lemma 3), and π-reduction on its own terminates. Strong normalisation follows by considering the corresponding lexicographic order.

Notice that in the particular case of $\hookrightarrow\!\!\!\!\longrightarrow$, Subject Reduction can be stated more precisely:

Theorem 2 (Subject Reduction for $\hookrightarrow\!\!\!\!\longrightarrow$).
Assume $\Gamma \vdash M:B$ and $M \hookrightarrow\!\!\!\!\longrightarrow_N M'$.
If $N \neq \epsilon$ there exist Δ and A such that $\Delta \vdash N:A$ otherwise let $\Delta = ()$.
There exists Γ' such that $\Gamma' \vdash M':B$ with $\Gamma = \Gamma' \cap \Delta$.

Proof. By investigating the proof of Theorem 1.

In the even more particular case of the λI-fragment, where $\hookrightarrow\!\!\!\!\longrightarrow_\epsilon = \longrightarrow_{\beta\pi}$, Subject Reduction does not modify the context.

3.2 Completeness

We now prove that strongly normalising terms can be typed.

Lemma 4 (Typing of substitution). *If $\Gamma \vdash M\{x := N\} : B$ then there exist Γ_1, Γ_2, U such that $\Gamma = \Gamma_1 \cap \Gamma_2$ and $\Gamma_1, x : U \vdash M : B$ and $\Gamma_2 \vdash N : U$.*

Proof. By induction on the typing tree for M. If $x \notin fv(M)$ we take $U = \omega$ and $\Gamma_2 = ()$.

Theorem 3 (Subject Expansion).
Assume $\Gamma \vdash M' : B$ and $M \hookrightarrow_N M'$.
Assume $\Delta \vdash N : A$ if $N \neq \epsilon$, otherwise let $\Delta = ()$.
We have $\Gamma \cap \Delta \vdash M : B$.

Proof. First by induction on B then by induction on $M \hookrightarrow_N M'$. The cases are exactly those of Subject Reduction.

Remark 7. This is not true with general β-reduction. For example, $(\lambda z.a)(\lambda y.yy)$ is typable but $(\lambda z.(\lambda x.a)(zz))(\lambda y.yy)$ is not (it is not SN).

Lemma 5 (Shape of $\hookrightarrow$-normal forms). *If a term cannot be reduced by $\hookrightarrow$ then it is of one of the following forms:*
- $\lambda x.M$
- $[M, N]$
- $xM_1 \dots M_n$

Proof. Straightforward.

Theorem 4 (Completeness). *If M is SN then there exist F and Γ such that $\Gamma \vdash M : F$.*

Proof. First by induction on the length of the longest $\longrightarrow_{\beta\pi}$ -reduction sequence starting from M then by induction on the size of M.
- If there exists M' such that $M \hookrightarrow_N M'$ then we use the induction hypothesis on M' (and if $N \neq \epsilon$ we use it on N too, which is a strict sub-term of M). Then we conclude with Theorem 3.
- If not then M is of one of these forms:
  - If $M = \lambda x.N$ then we apply the induction hypothesis on N.
  - If $M = [M_1, M_2]$ then we apply the induction hypothesis on M_1 and M_2
  - If $M = xM_1 \dots M_n$ then we apply the induction hypothesis on $M_1, \dots, M_n$ to get $\Gamma_1 \vdash M_1 : F_1, \dots, \Gamma_n \vdash M_n : F_n$, so we pick a fresh atomic type τ and we get:

$$(x : F_1 \to \dots \to F_n \to \tau) \cap \Gamma_1 \cap \dots \cap \Gamma_n \vdash xM_1 \dots M_n : \tau$$

In the particular case of the λI-fragment, typing is preserved by arbitrary expansions, with no modification of context. Since normal forms can be typed, any weakly normalising term can be typed, and are thus strongly normalising. Equivalence between weak normalisation and strong normalisation in λI is a well-known theorem that finds here a simple proof.

3.3 Corollaries

Corollary 3 (Characterisation of Strong Normalisation). *M is typable if and only if M is SN.*

With the characterisation of strong normalisation and the subject expansion theorem we have the following corollary.

Corollary 4. *If $M \hookrightarrow_N M'$ and N is SN and M' is SN then M is SN.*

This is a useful result, used for instance in many proofs of strong normalisation (e.g. by reducibility candidates) for λ-terms that are typed in various systems. It can also be seen as a generalisation the theorem that in λI, weak normalisation is equivalent to strong normalisation.

4 Optimal and principal typing

As we showed in the previous section, the typing system of Fig. 2 above characterises strongly normalising terms. Even better, the measure defined on the typing tree of a term gives a bound on the length of longest reduction sequence. But the typing system is too coarse to improve on that bound, so in order to get a better result about complexity (as established in section 5) it needs to be refined.

4.1 Optimal typing

In this section we first notice that the typing trees produced by the proof of completeness all satisfy a particular property that we call the *optimal* property. This property involves the following notions:

Definition 8 (Subsumption and forgotten types).
- *If π is a typing tree, we say that π* uses subsumption *if it features an occurrence of the abstraction rule where the condition $A \subseteq U$ is neither $A \subseteq A$ nor $A \subseteq \omega$.*
- *If π is a typing tree with no subsumption then we say that a type A is* forgotten *in π if:*
  - *π features an occurence of the abstraction rule where the condition is $A \subseteq \omega$,*
  - *or π features an occurence of the typing rule for Klop's construct $[M, N]$ where A is the type of N.*

  The multiset of forgotten types in π is written $\mathsf{forg}(\pi)$.

The optimal property also involves refining the grammar of types:

Definition 9 (Refined intersection types). *A^+, A^- and A^{--} are defined by the following grammar:*

$$\begin{array}{lll} A^+, B^+ & ::= & \tau \mid A^{--} \to B^+ \\ A^{--}, B^{--} & ::= & A^- \mid A^{--} \cap B^{--} \\ A^-, B^- & ::= & \tau \mid A^+ \to B^- \end{array}$$

The degree *of a type of the form A^+ is the number of arrows in negative positions:*

$$\begin{array}{|lll|} \hline \delta^+(\tau) & := & 0 \\ \delta^+(A^{--} \to B^+) & := & \delta^-(A^{--}) + \delta^+(B^+) + 1 \\ \hline \delta^-(A^{--} \cap B^{--}) & := & \delta^-(A^{--}) + \delta^-(B^{--}) \\ \hline \delta^-(\tau) & := & 0 \\ \delta^-(A^+ \to B^-) & := & \delta^+(A^+) + \delta^-(B^-) \\ \hline \end{array}$$

We can finally define the optimal property:

Definition 10 (Optimal typing). *A typing tree π concluding $\Gamma \vdash M : A$ is optimal if*
- *There is no subsumption in π*
- *A is of the form A^+*
- *For every $(x : B) \in \Gamma$, B is of the form B^{--}*
- *For every forgotten type B in π, B is of the form B^+.*

We write $\Gamma \vdash_{\mathsf{opt}} M : A^+$ if there exists such π.
The degree *of such a typing tree is defined as*

$$\delta(\pi) = \delta^+(A^+) + \Sigma_{x\,:\,B^{--} \in \Gamma} \delta^-(B^{--}) + \Sigma_{C^+ \in \mathsf{forg}(\pi)} \delta^+(C^+)$$

In this definition, A^+ is an output type, A^- is a basic input type (i.e. for a variable to be used once), and A^{--} is the type of a variable that can be used several times. The intuition behind this asymmetric grammar can be found in linear logic:

Remark 8. A simple type T can be translated as a type T^* of linear logic [Gir87] as follows:

$$\boxed{\begin{array}{ll} \tau^* & := \tau \\ (T \to S)^* & := !S^* \multimap T^* \end{array}}$$

It can also be translated as T^+ and T^- as follow :

$$\boxed{\begin{array}{ll} \tau^+ & := \tau \\ (T \to S)^+ & := !T^- \multimap S^+ \end{array}} \quad \boxed{\begin{array}{ll} \tau^- & := \tau \\ (T \to S)^- & := T^+ \multimap S^- \end{array}}$$

And we have in linear logic : $T^- \vdash T^*$ and $T^* \vdash T^+$

Now we can establish Subject Expansion for optimal trees:

Theorem 5 (Subject Expansion, optimal case).
Assume $\Gamma \vdash_{\mathsf{opt}} M' : B$ *and* $M \hookrightarrow_N M'$.
Assume $\Delta \vdash_{\mathsf{opt}} N : A$ *if* $N \neq \epsilon$, *otherwise let* $\Delta = ()$.
We have $\Gamma \cap \Delta \vdash_{\mathsf{opt}} M : B$.

Proof. By investigating the proof of Theorem 3, noticing that, in Lemma 4:
- if the typing tree of $\Gamma \vdash M\{x := N\} : B$ does not use subsumption, neither do those of $\Gamma_1, x : U \vdash M : B$ and $\Gamma_2 \vdash N : U$;
- the forgotten types in the typing tree of $\Gamma \vdash M\{x := N\} : B$ are exactly those in the typing trees of $\Gamma_1, x : U \vdash M : B$ and $\Gamma_2 \vdash N : U$.

The multiset of forgotten types in the proof of $\Gamma \cap \Delta \vdash_{\mathsf{opt}} M : B$ is that in the proof of $\Gamma \vdash_{\mathsf{opt}} M' : B$ (union that in the proof of $\Delta \vdash_{\mathsf{opt}} N : A$ if $N \neq \epsilon$).

From this we can derive a strengthened completeness theorem:

Theorem 6 (Completeness of optimal typing). *If* M *is SN then there exist* A^+ *and* Γ *such that* $\Gamma \vdash_{\mathsf{opt}} M : A^+$.

This raises the question of why we did not set our theory (say, the characterisation of strongly normalising terms) with optimal typing from the start. First, the optimal property is not preserved when going into sub-terms: if $\Gamma \vdash_{\mathsf{opt}} (\lambda x.M)N : A^+$, then it is not necessarily the case that $\Gamma \vdash_{\mathsf{opt}} N : B^+$ (it could be a type B that is not of the form B^+). Second the optimal property is not preserved by arbitrary reductions, as the use of subsumption for typing abstractions is necessary for Subject Reduction to hold.

Example 1.

$$\lambda x.x((\lambda y.z)x) \longrightarrow_\beta \lambda x.xz$$

However, Subject Reduction does hold for $\hookrightarrow$:

Theorem 7 (Subject Reduction, optimal case).
Assume $\Gamma \vdash_{\mathsf{opt}} M : B$ *and* $M \hookrightarrow_N M'$.
If $N \neq \epsilon$ *there exist* Δ *and* A *such that* $\Delta \vdash N : A$, *otherwise let* $\Delta = ()$.
There exists Γ' *such that* $\Gamma' \vdash_{\mathsf{opt}} M' : B$ *with* $\Gamma = \Gamma' \cap \Delta$.

Proof. By investigating the proof of Theorem 1, noticing that, in Lemma 2:
- if the typing trees of $\Gamma, x : U \vdash^{n_1} M : V$ and $\Delta \vdash^{n_2} N : U$ do not use subsumption then neither does that of $\Gamma \cap \Delta \vdash^{n_1+n_2} M\{x := N\} : V$;
- the forgotten types in the typing trees of $\Gamma, x : U \vdash^{n_1} M : V$ and $\Delta \vdash^{n_2} N : U$ are those in $\Gamma \cap \Delta \vdash^{n_1+n_2} M\{x := N\} : V$.

Again, the multiset of forgotten types in the proof of $\Gamma \vdash_{\mathsf{opt}} M : B$ is that in the proof of $\Gamma' \vdash_{\mathsf{opt}} M' : B$ (union that in the proof of $\Delta \vdash_{\mathsf{opt}} N : A$ if $N \neq \epsilon$).

Remark 9. Notice the particular case of λI, where $\beta\pi$ reductions and expansions both preserve optimal typings and multisets of forgotten types, and therefore they preserve the degree of optimal typing trees.

4.2 Principal-optimal typing trees

We now introduce the notion of principal typing, and for that we first define the commutativity and associativity of the intersection rule.

Definition 11 (AC of the intersection rule). *Let $\simeq$ be the smallest congruence on typing trees containing the two equations*

$$\frac{\Gamma \vdash M:A \quad \Delta \vdash M:B}{\Gamma \cap \Delta \vdash M:A \cap B} \simeq \frac{\Delta \vdash M:B \quad \Gamma \vdash M:A}{\Gamma \cap \Delta \vdash M:A \cap B}$$

$$\frac{\dfrac{\Gamma_1 \vdash M:A_1 \quad \Gamma_2 \vdash M:A_2}{\Gamma_1 \cap \Gamma_2 \vdash M:A_1 \cap A_2} \qquad \Gamma_3 \vdash M:A_3}{\Gamma_1 \cap \Gamma_2 \cap \Gamma_3 \vdash M:A_1 \cap A_2 \cap A_3} \simeq \frac{\Gamma_1 \vdash M:A_1 \qquad \dfrac{\Gamma_2 \vdash M:A_2 \quad \Gamma_3 \vdash M:A_3}{\Gamma_2 \cap \Gamma_3 \vdash M:A_2 \cap A_3}}{\Gamma_1 \cap \Gamma_2 \cap \Gamma_3 \vdash M:A_1 \cap A_2 \cap A_3}$$

Definition 12 (Principal typing).

- *A substitution σ mapping the atomic types $\tau_1, \ldots, \tau_n$ to the types $F_1, \ldots, F_n$ acts on types, contexts, judgements and typing trees, so we can write $A\sigma$, $\Gamma\sigma$, $\pi\sigma$,...*
- *We write $\pi \leq \pi'$ if there exists a substitution σ such that $\pi' \simeq \pi\sigma$. (In that case if π concludes $\Gamma \vdash M:A$ then π' must conclude $\Gamma\sigma \vdash M:A\sigma$.)*
- *A typing tree π concluding $\Gamma \vdash_{\mathsf{opt}} M:A^+$ is said to be* principal *if for any typing tree π' concluding $\Gamma' \vdash_{\mathsf{opt}} M:A'^+$ we have $\pi \leq \pi'$.*

Typing trees produced by the proof of the completeness theorem are principal:

Theorem 8 (Principal typing always exists). *If M is SN then there exist F, Γ and a principal typing tree π concluding $\Gamma \vdash_{\mathsf{opt}} M:F$.*

Proof. The proof follows that of Theorems 4 and 6: first by induction on the longest reduction sequence starting from M then by induction on the size of M. The novelty resides in checking principality:

- If there exists M' such that $M \hookrightarrow_N M'$, we assume another optimal typing π of M and use Theorem 7 to get an optimal typing π' for M'. By principality, π' must be an instance of the principal typing tree we have recursively constructed for M' (and similarly for N if $N \neq \epsilon$). From this we deduce that π is an instance of the one we got by Subject Expansion.
- If not then M is of one of these forms:
  - $M = \lambda x.N$ or $M = [M_1, M_2]$, in which case we call upon the induction hypothesis,
  - $M = xM_1 \ldots M_n$, in which case the induction hypothesis provide principal $\Gamma_1 \vdash_{\mathsf{opt}} M_1:A_1^+, \ldots, \Gamma_n \vdash_{\mathsf{opt}} M_n:A_n^+$, and principality is ensured by choosing a fresh atomic type τ in the type $A_1^+ \to \cdots \to A_n^+ \to \tau$ of x.

Remark 10. The shape of an optimal typing tree is unique but it is not syntax-directed, so we cannot use this unicity to have an algorithm to directly compute the typing tree (other than "executing" the term).

One can also notice that in λI, there is a direct link between the measure read off from a principal optimal typing and its degree.

Lemma 6 (Degree of λI's normal forms). *If π is a principal typing tree of $\Gamma \vdash^n_{\mathsf{opt}} M:A$, where M is a λI-term in $\beta\pi$-normal form, then $\delta(\pi) = n$. It is also the number of applications in the term M.*

Proof. Again, by inspecting the proof of completeness: every time we type $M = xM_1 \ldots M_n$, using $\Gamma_1 \vdash_{\mathsf{opt}} M_1:A_1^+, \ldots, \Gamma_n \vdash_{\mathsf{opt}} M_n:A_n^+$, we add as many arrows by constructing the type $A_1^+ \to \cdots \to A_n^+ \to \tau$ as we use new occurrences of rule (App).

5 Complexity

In this section we derive two complexity results, one for each fragment of our calculus: Church-Klop's λI-calculus and the pure λ-calculus.

5.1 Complexity result for Church-Klop's λI

In this section every term is assumed to be in the λI-fragment of the calculus. Remember that in that fragment, $\hookrightarrow_\epsilon$ is the same as $\longrightarrow_{\beta\pi}$.

We first identify a smaller reduction relation $\overset{\beta\mathsf{small}}{\hookrightarrow}$ (within λI) that will always decrease the measure of optimal trees exactly by one.

Definition 13 (Small-reduction). *Small-reduction, written $\overset{\beta\mathsf{small}}{\hookrightarrow}$, is defined in Fig. 3.*

$$\frac{}{(\lambda x.M)\ N\ \overrightarrow{N_i} \overset{\beta\mathsf{small}}{\hookrightarrow} M\{x := N\}\ \overrightarrow{N_i}}$$

$$\frac{N \overset{\beta\mathsf{small}}{\hookrightarrow} N' \qquad x \notin fv(M)}{(\lambda x.[M,x])\ N\ \overrightarrow{N_i} \overset{\beta\mathsf{small}}{\hookrightarrow} (\lambda x.[M,x])\ N'\ \overrightarrow{N_i}}$$

$$\frac{N \overset{\beta\mathsf{small}}{\hookrightarrow} N'}{x\ \overrightarrow{N_i}\ N\ \overrightarrow{M_j} \overset{\beta\mathsf{small}}{\hookrightarrow} x\ \overrightarrow{N_i}\ N'\ \overrightarrow{M_j}} \qquad \frac{M \overset{\beta\mathsf{small}}{\hookrightarrow} M'}{\lambda x.M \overset{\beta\mathsf{small}}{\hookrightarrow} \lambda x.M'}$$

$$\frac{M \overset{\beta\mathsf{small}}{\hookrightarrow} M'}{[M,N] \overset{\beta\mathsf{small}}{\hookrightarrow} [M',N]} \qquad \frac{N \overset{\beta\mathsf{small}}{\hookrightarrow} N'}{[M,N] \overset{\beta\mathsf{small}}{\hookrightarrow} [M,N']}$$

Fig. 3. Small-reduction

Remark 11. If a term can be reduced by $\longrightarrow_\beta$ and not by $\longrightarrow_\pi$ then it can be reduced by $\overset{\beta\mathsf{small}}{\hookrightarrow}$.[5]

Lemma 7 (Small reduction decreases the measure by 1).
If $\Gamma \vdash^n_{\mathsf{opt}} M : A$ and $M \overset{\beta\mathsf{small}}{\hookrightarrow} M'$ then $\Gamma \vdash^{n-1}_{\mathsf{opt}} M' : A$.

Proof. The typing tree is the one produced by the proof of Subject Reduction. Checking that that tree is also optimal with measure $n-1$ is done by induction on M (or equivalently by induction on the derivation of $M \overset{\beta\mathsf{small}}{\hookrightarrow} M'$ then by induction on n for those rules that feature a series of n applications).
The optimal property of typing is preserved in inductive steps, i.e. while going into the term M and until the base case of $\overset{\beta\mathsf{small}}{\hookrightarrow}$ is found (the first rule):

- In the first rule of $\overset{\beta\mathsf{small}}{\hookrightarrow}$, notice that only one application is removed only because we are in the λI-fragment.
- In the second rule, x has a type of the form B^-, i.e. of the form $C_1^+ \to \cdots \to C_s^+ \to \tau$ (with $s \geq n$), so the typing of N is optimal. We can then use the induction hypothesis to conclude.

[5] We could call $\overset{\beta\mathsf{small}}{\hookrightarrow}$ a strategy, but it does not necessarily determine a unique redex to reduce.

– In the third rule, the type of x is a forgotten type, so by the optimal property is must be of the form A^+, and by construction N has the very same type. This makes its typing optimal and we can then use the induction hypothesis to conclude.
– In the fourth rule, the typing of M is optimal if that of $\lambda x.M$ is.
– In the fifth rule, the typing of M is optimal if that of $[M, N]$ is (the two terms have the same type in the same context).
– In the sixth rule, the type of N is a forgotten type, so by optimality is has to be of the form A^+, so again the typing of N is optimal.

Theorem 9 (Complexity result).
If $\Gamma \vdash^n_{\mathsf{opt}} M : A$ with a principal typing tree of degree n' then there exists a $\beta\pi$-normal form M' such that

$$M \longrightarrow^*_\pi (\longrightarrow_\beta \longrightarrow^*_\pi)^{n-n'} M'$$

This reduction sequence from M to M' is of maximal length.[6]

Proof. By induction on n. We reduce by $\xrightarrow{\beta\mathsf{small}}$ and $\longrightarrow_\pi$ until hitting the normal form M', also typed by $\Gamma \vdash^{n'}_{\mathsf{opt}} M' : A$ with some principal typing tree of measure n'. By Lemma 6, n' is both the number of applications in M' and the degree of its principal typing tree. That degree is not changed by expansions, so it is also the degree of the principal typing tree of M which we started with.

5.2 Complexity result for pure λ-calculus

In this section we derive a similar result for the pure λ-calculus. For this we reduce the problem of pure λ to that of λI (treated above).
In order to make the distinction very clear about what terms are in pure λ and what terms are in λI, we use two different notational styles: $t, u, v, \ldots$ for pure λ-terms and $T, U, V, \ldots$ for λI-terms.
We want to exhibit in pure λ-calculus the longest reduction sequences, and show that their lengths are exactly those that can be predicted in λI.
For the longest reduction sequences we simply use the perpetual strategy from [vRSSX99], shown in Fig. 4.

$$\frac{x \in fv(t) \text{ or } t' \text{ is a } \beta\text{-normal form}}{(\lambda x.t)\ t'\ \overrightarrow{t_j} \rightsquigarrow t\{x := t'\}\ \overrightarrow{t_j}} \qquad \frac{t' \rightsquigarrow t'' \qquad x \notin fv(t)}{(\lambda x.t)\ t'\ \overrightarrow{t_j} \rightsquigarrow (\lambda x.t)\ t''\ \overrightarrow{t_j}}$$

$$\frac{t \rightsquigarrow t'}{x\ \overrightarrow{t_j}\ t\ \overrightarrow{p_j} \rightsquigarrow x\ \overrightarrow{t_j}\ t'\ \overrightarrow{p_j}} \qquad \frac{t \rightsquigarrow t'}{\lambda x.t \rightsquigarrow \lambda x.t'}$$

Fig. 4. A perpetual reduction strategy for λ

Remark 12. $\rightsquigarrow \subseteq \longrightarrow_\beta$
If t is not a β-normal form, then there is a λ-term t' such that $t \rightsquigarrow t'$.

Although we do not need it here, it is worth mentioning that $\rightsquigarrow$ defines a perpetual strategy w.r.t. β-reduction, i.e. if M is not β-strongly normalising and $M \rightsquigarrow M'$, then neither is M' [vRSSX99]. In that sense it can be seen as the worst strategy (the least efficient). We show here that it is the worst in a stronger sense: it maximises the lengths of reduction sequences. For that we show that the length of a reduction sequence produced by that strategy matches that of the longest reduction sequence in λI. This requires encoding the syntax of the pure λ-calculus into λI, as shown in Fig. 5 (from [Len05,Len06]).

[6] As Subject reduction implies that any other reduction sequence has a length less than or equal to $n - n'$.

$\mathsf{i}(x) := x$		$\mathsf{i}(\lambda x.M) := \lambda x.\mathsf{i}(M)$	if $x \in fv(M)$
$\mathsf{i}(M\ N) := \mathsf{i}(M)\ \mathsf{i}(N)$		$\mathsf{i}(\lambda x.M) := \lambda x.[\mathsf{i}(M), x]$	if $x \notin fv(M)$

Fig. 5. Encoding from λ to λI

Lemma 8 ([Len05,Len06]). *For any λ-terms t and u,*
- $fv(i(t)) = fv(t)$
- $i(t)\{x := i(u)\} = i(t\{x := u\})$

But this encoding will not allow the simulation of $\rightsquigarrow$ by $\longrightarrow_{\beta\pi}$ in λI. To allow the simulation we need to generalise the i-encoding into a larger encoding that needs to be non-deterministic (i.e. to be a relation rather than a function).

Definition 14 (Relation between λ & λI [Len05,Len06]). *The relation $\mathcal{G}$ between λ-terms & λI-terms is given by the rules of Fig. 6 and (non-deterministically) generalises the i encoding.*

$$\frac{}{((\lambda x.t)\ t'\ \overrightarrow{t_j})\ \mathcal{G}\ \mathsf{i}((\lambda x.t)\ t'\ \overrightarrow{t_j})} \qquad \frac{t'\ \mathcal{G}\ T' \qquad x \notin fv(t)}{((\lambda x.t)\ t'\ \overrightarrow{t_j})\ \mathcal{G}\ (\mathsf{i}(\lambda x.t)\ T'\ \overrightarrow{\mathsf{i}(t_j)})}$$

$$\frac{\forall j \qquad t_j\ \mathcal{G}\ T_j}{(x\ \overrightarrow{t_j})\ \mathcal{G}\ (x\ \overrightarrow{T_j})} \qquad \frac{t\ \mathcal{G}\ T \quad x \in fv(T)}{\lambda x.t\ \mathcal{G}\ \lambda x.T} \qquad \frac{t\ \mathcal{G}\ T \qquad N \text{ is a normal form for } \beta\pi}{t\ \mathcal{G}\ [T, N]}$$

Fig. 6. Relation between λ & λI

Lemma 9 ([Len05,Len06]).
1. *If t is a β-normal form and $t\ \mathcal{G}\ T$, then T is a $\beta\pi$-normal form.*
2. *For any λ-term t, $t\ \mathcal{G}\ i(t)$.*

In [Len06,KL07] it is shown that the perpetual strategy can be simulated in λI through the i-encoding:

Theorem 10 (Strong simulation of $\rightsquigarrow$ in λI [Len06,KL07]).
If $t\ \mathcal{G}\ T$ and $t \rightsquigarrow t'$ then there exists T' in λI such that $t'\ \mathcal{G}\ T'$ and $T \longrightarrow^{+}_{\beta\pi} T'$.

By inspecting the proof of the simulation in [Len06,KL07], one notices that $T \longrightarrow^{+}_{\beta\pi} T'$ **is in fact** $T(\longrightarrow^{*}_{\pi} \xrightarrow{\beta\mathsf{small}} \longrightarrow^{*}_{\pi})T'$, which decreases the measure read off an optimal typing tree exactly by one.
Now given Lemma 9, this means that the perpetual strategy from [vRSSX99] generates, from a given term t, a reduction sequence to its normal form t' of the same length as a reduction sequence, in λI, from $\mathsf{i}(t)$ to its normal form T' (with $t'\ \mathcal{G}\ T'$). This length can be predicted in the measure read off an optimal typing tree for $\mathsf{i}(t)$; and it so happens that it is the same as the measure read off an optimal typing tree for t:

Lemma 10 (Preservation of optimal typing by i). *Let t be a pure λ-term. If $\Gamma \vdash^{n}_{\mathsf{opt}} t : A$ then $\Gamma \vdash^{n}_{\mathsf{opt}} i(t) : A$. If the typing is principal then it remains principal and the degree is not changed.*

Proof. By induction on the derivation tree we prove that if $\Gamma \vdash^{n} t : A$ with no subsumption then $\Gamma \vdash^{n} \mathsf{i}(t) : A$ with no subsumption and with the same forgotten types.

Theorem 11 (Complexity result for λ).
If $\Gamma \vdash^{n}_{\mathsf{opt}} t : A$ with a principal typing tree of degree n' then there exists a β-normal form t' such that

$$t \longrightarrow_{\beta}^{n-n'} t'$$

This reduction sequence from t to t' is of maximal length.[7]

[7] As Subject reduction implies that any other reduction sequence has a length less than or equal to $n - n'$.

6 Conclusion

We have defined a typing system for non-idempotent intersection types. We have shown that it characterises strongly normalising terms in a more natural way than idempotent intersection types do. With some reasonable restrictions on the derivation tree we have obtained results on the maximum number of β-reductions in a reduction sequence of a λ-term (with Klop's extension).
We noticed *a posteriori* that our technology is similar to that which can be found in e.g. [KW99,NM04]. One of the concerns of this line of research is how the process of type inference compares to that of normalisation, in terms of complexity *classes* (these two problems being parameterised by the size of terms and a notion of *rank* for types).
The present paper shows how such a technology can actually provide an exact equality, specific to each λ-term and its typing tree, between the number read off the tree and the length of the longest reduction sequence. Of course this only emphasises the fact that type inference is as hard as normalisation, but type inference as a process is not a concern of this paper.
Our non-idempotent intersection type system and our results can be lifted to other calculi featuring e.g. explicit substitutions, combinators, or algebraic constructors and destructors (to handle integers, products, sums,...).
Idempotent intersection types have been used to provide model-based proofs of strong normalisation for well-known typing systems (simple types, system F, system F_ω,...,). Such model constructions (I-filters [CS07], orthogonality) can also be done with non-idempotent intersection types with no increased difficulty, and with the extra advantage that the strong normalisation of terms in the models is much simpler to prove. This is our next paper.

References

[Abr93] S. Abramsky. Computational interpretations of linear logic. *Theoret. Comput. Sci.*, 111:3–57, 1993.

[BBdH93] N. Benton, G. Bierman, V. de Paiva, and M. Hyland. A term calculus for intuitionistic linear logic. In J. F. G. Groote and M. Bezem, editors, *Proc. of the 1st Int. Conf. on Typed Lambda Calculus and Applications*, volume 664 of *LNCS*, pages 75–90. Springer-Verlag, 1993.

[BCDC83] H. Barendregt, M. Coppo, and M. Dezani-Ciancaglini. A filter lambda model and the completeness of type assignment. *J. of Symbolic Logic*, 48(4):931–940, 1983.

[BEM10] A. Bucciarelli, T. Ehrhard, and G. Manzonetto. Categorical models for simply typed resource calculi. *ENTCS*, 265:213–230, 2010.

[BM03] P. Baillot and V. Mogbil. Soft lambda-calculus: a language for polynomial time computation. *CoRR*, cs.LO/0312015, 2003.

[Bou03] G. Boudol. On strong normalization in the intersection type discipline. In M. Hofmann, editor, *Proc. of the 7th Int. Conf. on Typed Lambda Calculus and Applications (TLCA'03)*, volume 2701 of *LNCS*, pages 60–74. Springer-Verlag, June 2003.

[CD78] M. Coppo and M. Dezani-Ciancaglini. A new type assignment for lambda-terms. *Archiv für mathematische Logik und Grundlagenforschung*, 19:139–156, 1978.

[CD80] M. Coppo and M. Dezani-Ciancaglini. An extension of the basic functionality theory for the λ-calculus. *Notre Dame J. of Formal Logic*, 21(4):685–693, 1980.

[CS07] T. Coquand and A. Spiwack. A proof of strong normalisation using domain theory. *Logic. Methods Comput. Science*, 3(4), 2007.

[dC05] D. de Carvalho. Intersection types for light affine lambda calculus. *ENTCS*, 136:133–152, 2005.

[dC09] D. de Carvalho. Execution time of lambda-terms via denotational semantics and intersection types. *CoRR*, abs/0905.4251, 2009.

[DCHM00] M. Dezani-Ciancaglini, F. Honsell, and Y. Motohama. Compositional characterizations of lambda-terms using intersection types (extended abstract). In *MFCS: Symp. on Mathematical Foundations of Computer Science*, 2000.

[DCT07] M. Dezani-Ciancaglini and M. Tatsuta. A Behavioural Model for Klop's Calculus. In F. Corradini and C. Toffalori, editors, *Logic, Model and Computer Science*, volume 169 of *ENTCS*, pages 19–32. Elsevier, 2007.

[ER03] T. Ehrhard and L. Regnier. The differential lambda-calculus. *Theoret. Comput. Sci.*, 309(1-3):1–41, 2003.

[Ghi96] S. Ghilezan. Strong normalization and typability with intersection types. *Notre Dame J. Formal Loigc*, 37(1):44–52, 1996.

[Gir87] J.-Y. Girard. Linear logic. *Theoret. Comput. Sci.*, 50(1):1–101, 1987.

[GR07] M. Gaboardi and S. R. D. Rocca. A soft type assignment system for *lambda* -calculus. In J. Duparc and T. A. Henzinger, editors, *Proc. of the 16th Annual Conf. of the European Association for Computer Science Logic (CSL'07)*, volume 4646 of *LNCS*, pages 253–267. Springer-Verlag, September 2007.

[How80] W. A. Howard. The formulae-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism*, pages 479–490. Academic Press, 1980. Reprint of a manuscript written 1969.

[KL07] D. Kesner and S. Lengrand. Resource operators for the λ-calculus. *Inform. and Comput.*, 205:419–473, 2007.

[Klo80] J.-W. Klop. *Combinatory Reduction Systems*, volume 127 of *Mathematical Centre Tracts*. CWI, 1980. PhD Thesis.

[KW99] A. J. Kfoury and J. B. Wells. Principality and decidable type inference for finite-rank intersection types. In *Proc. of the 26th Annual ACM Symp. on Principles of Programming Languages (POPL'99)*, pages 161–174. ACM Press, ACM Press, 1999.

[Laf04] Y. Lafont. Soft linear logic and polynomial time. *Theoret. Comput. Sci.*, 318(1-2):163–180, 2004.

[Lei86] D. Leivant. Typing and computational properties of lambda expressions. *Theoretical Computer Science*, 44(1):51–68, 1986.

[Len05] S. Lengrand. Induction principles as the foundation of the theory of normalisation: Concepts and techniques. Technical report, PPS laboratory, Université Paris 7, March 2005. Available at http://hal.ccsd.cnrs.fr/ccsd-00004358

[Len06] S. Lengrand. *Normalisation & Equivalence in Proof Theory & Type Theory*. PhD thesis, Univ. Paris 7 & Univ. of St Andrews, 2006.

[NM04] P. M. Neergaard and H. G. Mairson. Types, potency, and idempotency: why nonlinearity and amnesia make a type system work. In C. Okasaki and K. Fisher, editors, *Proc. of the ACM International Conference on Functional Programming*, pages 138–149. ACM Press, September 2004.

[Sør97] M. H. B. Sørensen. Strong normalization from weak normalization in typed lambda-calculi. *Inform. and Comput.*, 37:35–71, 1997.

[Val01] S. Valentini. An elementary proof of strong normalization for intersection types. *Arch. Math. Log.*, 40(7):475–488, 2001.

[vB92] S. van Bakel. Complete restrictions of the intersection type discipline. *Theoret. Comput. Sci.*, 102(1):135–163, 1992.

[vRSSX99] F. van Raamsdonk, P. Severi, M. H. B. Sørensen, and H. Xi. Perpetual reductions in λ-calculus. *Inform. and Comput.*, 149(2):173–225, 1999.

[Xi97] H. Xi. Weak and strong beta normalisations in typed lambda-calculi. In P. de Groote, editor, *Proc. of the 3th Int. Conf. on Typed Lambda Calculus and Applications (TLCA'97)*, volume 1210 of *LNCS*, pages 390–404. Springer-Verlag, April 1997.