Archive for Mathematical Logic manuscript No.
(will be inserted by the editor)

An elementary proof of strong normalization for intersection types

Silvio Valentini

Dipartimento di Matematica Pura ed Applicata, Università di Padova, via G. Belzoni n.7, I–35131 Padova, Italy, e-mail: silvio@math.unipd.it

The date of receipt and acceptance will be inserted by the editor

Abstract We provide a new and elementary proof of strong normalization for the lambda calculus of intersection types. It uses no strong method, like for instance Tait-Girard *reducibility predicates*, but just simple induction on type complexity and derivation length and thus it is obviously formalizable within first order arithmetic.

To obtain this result, we introduce a new system for intersection types whose rules are directly inspired by the reduction relation.

Finally, we show that not only the set of strongly normalizing terms of pure lambda calculus can be characterized in this system, but also that a straightforward modification of its rules allows to characterize the set of weakly normalizing terms.

Key words Lambda calculus – intersection types – normalization

1 Introduction

It is well known that the pure lambda calculus Λ (see [1]) formalizes the notion of computable function without any reference to the concepts of domain and co-domain, contrary to what happens with the set theoretic or the categorical approach. The main advantage of this approach is the possibility of coding any recursive function within a very simple formalism. Indeed, a lambda term is built inductively, starting from the variables, by means of the lambda abstraction and a *free* form of application, i.e. we have

Mathematics Subject Classification: 03B40

the following term formation rules:

$$\mathsf{Term} := \mathsf{Var} \mid (\lambda\ \mathsf{Var}.\mathsf{Term}) \mid \mathsf{Term}(\mathsf{Term})$$

where Var is a countable set whose elements are called variables.

Not only the syntax of the objects of Λ is simple, but also the notion of computation for this very abstract notion of function becomes the simple notion of β*-reduction* (notation $\rightsquigarrow_\beta$). This is the relation between lambda terms obtained by closing under the term construction operations the relation of β*-contraction*, that is

$$(\lambda x.c)(a) \rightsquigarrow c[x := a]$$

The computation of the value of a lambda term is then defined as a *reduction process*, i.e. successive steps of β-reduction, until a *normal form* of the term is possibly reached, i.e. a form where no β-contraction can be applied. Given a lambda term c, there are in general many different reduction processes, according to the choice of the β-contraction to be expanded within c; hence, it is well possible that only some of the reduction processes eventually terminate into a normal form. Moreover, since it is possible to have a code within Λ for any recursive function, there is no possibility to know if a reduction process for c will eventually terminate, because of the halting problem.

On the other hand, in the usual mathematical practice - both in the set theoretic and in the categorical approach - and in many concrete algorithms, functions are intended to operate over objects of a certain type in order to produce objects of some other type. Following this idea, the rule of application should be no longer completely free; in fact a function should be applicable only to arguments of the correct type. Thus it will be no longer possible to build all the terms of Λ. However, a main advantage of this approach is the possibility to prove more properties on the terms which can be built because of the greater quantity of information. For instance, one of the main problems on the terms of Λ is to determine whether all the reduction processes for a certain term will eventually terminate, i.e. the *strong normalization* problem, or if there exists at least one reduction process which will eventually terminate, i.e. the *weak normalization* problem. In the case of the lambda-calculi where functions and their arguments have a type there are suitable tools to deal with these problems.

In order to keep the good aspects of both the sides, a possible strategy is to find suitable typing systems for the terms of Λ. Many typing systems are possible and the choice among them depends on the particular problem that one has to solve. For instance, provided one wants to solve the strong normalization problem, a possibility is to use *simply typed* lambda calculus

$\Lambda_{\rightarrow}$; its rules of type formation are the following:

$$\mathsf{Type} := \mathsf{Const} \mid \mathsf{Type} \rightarrow \mathsf{Type}$$

were Const is a set whose elements are called *basic types*.

The intended meaning is that a type $\sigma \rightarrow \tau$ of $\Lambda_{\rightarrow}$ denotes a set of functions from elements of the set denoted by the type σ into elements of the set denoted by the type τ. Thus, in order to build the elements of these types, we use the following rules[1]:

$$\text{(variable)} \qquad \Gamma, x : \sigma \vdash x : \sigma$$

$$\text{(lambda abstraction)} \qquad \frac{\Gamma, x : \sigma \vdash c : \tau}{\Gamma \vdash \lambda x.c : \sigma \rightarrow \tau}$$

$$\text{(application)} \qquad \frac{\Gamma \vdash c : \tau \rightarrow \sigma \qquad \Gamma \vdash a : \tau}{\Gamma \vdash c(a) : \sigma}$$

where Γ is a commutative list of assumptions of the form $x : \sigma$ for some type σ such that no variable appears more than once and $\Gamma \vdash c : \tau$ means that c is an element of type τ in the *base* Γ.

It is well known (see for instance [4] or here, section 4) that all the terms of $\Lambda_{\rightarrow}$ are strongly normalizing. Hence, the terms of $\Lambda_{\rightarrow}$ form a subset of the set of strongly normalizing terms of Λ. But, not all the strongly normalizing terms of Λ have a type in $\Lambda_{\rightarrow}$; for instance, consider the term $\lambda x.x(x)$: it is in normal form, and hence it is trivially strongly normalizing, but it cannot have a type within $\Lambda_{\rightarrow}$ because of the instance of *self-application*. It is clear that a real solution of the strong normalization problem would be a typing system which allows to build all the strongly normalizing terms of Λ, and only them.

Surprisingly, this typing system exists and can be obtained from $\Lambda_{\rightarrow}$ by adding just one type. The abstract syntax of the types of the calculus $\Lambda_{\wedge}$ of *intersection types* is the following:

$$\mathsf{Type} := \mathsf{Const} \mid \mathsf{Type} \rightarrow \mathsf{Type} \mid \mathsf{Type} \wedge \mathsf{Type}$$

The intended meaning of the new type $\sigma \wedge \tau$ of $\Lambda_{\wedge}$ is that $\sigma \wedge \tau$ denotes the intersection of the two sets denoted by the type σ and τ respectively. Thus, in order to build the elements for these new types, we add the following rules to the previous ones:

$$\text{(intersection introduction)} \qquad \frac{\Gamma \vdash c : \sigma \qquad \Gamma \vdash c : \tau}{\Gamma \vdash c : \sigma \wedge \tau}$$

$$\text{(intersection elimination)} \qquad \frac{\Gamma \vdash c : \sigma \wedge \tau}{\Gamma \vdash c : \sigma} \qquad \frac{\Gamma \vdash c : \sigma \wedge \tau}{\Gamma \vdash c : \tau}$$

[1] To be more precise we should speak here of typing system *a la* Curry versus a typing system *a la* Church where all the variables within a term and the sub-terms themselves are typed.

Motivations for the study of $\Lambda_\wedge$ are widely discussed in the literature (see for instance [3] or [5]). In particular, it is possible to prove that the terms which are typable in this typed lambda calculus are precisely the strongly normalizing terms of Λ [7]. However, the proofs of the strong normalization theorem for the terms of $\Lambda_\wedge$, which is possible to find in the literature [5], use a version of the Tait-Girard's reducibility predicates [8,4], that is, an argument which is not elementary.

Here, we propose a new proof of the strong normalization theorem for $\Lambda_\wedge$ which is completely elementary. Indeed it uses no strong method but just simple induction on type complexity and derivation length and thus it is obviously formalizable within first order arithmetic. To obtain this result, in the next section we introduce a new system $\Lambda^s_\wedge$ for intersection types, whose rules are directly inspired by the reduction relation, and we state a strong normalization theorem for the terms which are typable in this system. Then, in the following sections, we show that all the terms of $\Lambda_\wedge$ can be typed also within $\Lambda^s_\wedge$ and hence that they are obviously strong normalizing. Finally, we show that not only the set of strongly normalizing terms of pure lambda calculus can be characterized in $\Lambda^s_\wedge$, but also that a straightforward modification of its rules allows to characterize the set of weakly normalizing terms.

2 A new system for intersection types

In this section we will tailor to $\Lambda_\wedge$ a technique to prove the strong normalization theorem which was already used for the simple typed lambda calculus in [9] and for more complex typed lambda calculi of first and second order in [2]. The first step is the introduction of a new typing system $\Lambda^s_\wedge$. In order to distinguish the judgements of the new typing system from those of $\Lambda_\wedge$ we will use the symbol $\vdash_s$ instead of $\vdash$ (in the following rules we will write C to mean a basic type).

$$(\lambda\text{-introduction})\ \frac{\Gamma \vdash_s c[x := a](a_1)\dots(a_n) : C \quad \Gamma \vdash_s a : \sigma}{\Gamma \vdash_s (\lambda x.c)(a)(a_1)\dots(a_n) : C}$$

$$(\text{abstraction})\ \frac{\Gamma, x : \sigma \vdash_s b(x) : \tau}{\Gamma \vdash_s b : \sigma \to \tau} \quad x \notin \mathsf{FV}(b) \text{ and } x \notin \Gamma$$

$$(\text{intersection})\ \frac{\Gamma \vdash_s c : \sigma \qquad \Gamma \vdash_s c : \tau}{\Gamma \vdash_s c : \sigma \wedge \tau}$$

The treatment of the variables in a base is more complex: we require that a variable is introduced only when the term obtained by applying it to suitable terms is of basic type. Thus, in order to deal with variables, we propose the

following rules:

(basic) $\quad \Gamma, x : C \vdash_s x : C$

(impl-left) $\dfrac{\Gamma \vdash_s a : \sigma_1 \qquad \Gamma, y : \sigma_2 \vdash_s y(a_1) \ldots (a_n) : C}{\Gamma, x : \sigma_1 \to \sigma_2 \vdash_s x(a)(a_1) \ldots (a_n) : C}$
$y \notin \mathsf{FV}(a_1) \cup \ldots \cup \mathsf{FV}(a_n)$ and $y \notin \Gamma$

(inters-left) $\dfrac{\Gamma, x : \sigma_1, x : \sigma_2 \vdash_s x(a_1) \ldots (a_n) : C}{\Gamma, x : \sigma_1 \wedge \sigma_2 \vdash_s x(a_1) \ldots (a_n) : C}$

It may be useful to note that in this calculus we admit the possibility for a variable to appear in a base more than once, as an hypothetical element of different types; we think that this fact should not appear too strange within the framework of intersection types: we are just assuming that this variable is an element of all these types; for instance in the *inters-left* rule we just state that from the assumption that an element is in the intersection of two types we can infer all what we could infer from the assumption that it is an element of both the two types. However, we have to take care of this possibility in the *abstraction* rule where we want to discharge at once all the assumptions of the abstracted variable within the base: this is the reason of the second side condition, i.e. $x \notin \Gamma$, for the application of this rule; its meaning is that the variable x does not appear in any assumption in Γ, i.e., for no type σ, $x : \sigma \in \Gamma$.

Since the previous rules are not standard, let us show a simple example of derivation within $\Lambda^s_\wedge$. A typical term which can be typed within the intersection types, and which is not in a more usual typing system because of the instance of self-application, is $\lambda x.x(x)$. Here we will exhibit its derivation by using the rules of $\Lambda^s_\wedge$ (in order to keep the proof as simple as possible we suppose that A and B are basic types). Consider first the following derivation:

$$\dfrac{\dfrac{x : A \wedge (A \to B) \vdash_s x : A}{x : A, x : A \to B \vdash_s x : A} \qquad \dfrac{\dfrac{\dfrac{x : A, z : A \vdash_s z : A \quad x : A, z : A, y : B \vdash_s y : B}{x : A, x : A \to B, z : A \vdash_s x(z) : B}\ \text{impl-l}}{x : A \wedge (A \to B), z : A \vdash_s x(z) : B}\ \text{inter-l}}{x : A \wedge (A \to B) \vdash_s x : A \to B}\ \text{abstr.}}{x : A \wedge (A \to B) \vdash_s x : A \wedge (A \to B)}\ \text{inters.}$$

Now we obtain:

$$\dfrac{\dfrac{\dfrac{x : A \vdash_s x : A \quad x : A, y : B \vdash_s y : B}{x : A, x : A \to B \vdash_s x(x) : B}\ \text{impl-l}}{x : A \wedge (A \to B) \vdash_s x(x) : B} \qquad \vdots \; x : A \wedge (A \to B) \vdash_s x : A \wedge (A \to B)}{x : A \wedge (A \to B) \vdash_s (\lambda x.x(x))(x) : B}\ \lambda\text{-intro}$$
$$\dfrac{x : A \wedge (A \to B) \vdash_s (\lambda x.x(x))(x) : B}{\vdash_s \lambda x.x(x) : (A \wedge (A \to B)) \to B}\ \text{abstraction}$$

The main property of $\Lambda^s_\wedge$ is the straightforward proof of the strong normalization theorem.

Theorem 1 *Let $\Gamma \vdash_s c : \sigma$. Then c is strongly normalizing.*

Proof The proof is immediate by induction on the length of the derivation of $\Gamma \vdash_s c : \sigma$.

3 A logical calculus derived from the rules of $\Lambda^s_\wedge$

If we strip all the terms in the rules of $\Lambda^s_\wedge$, we obtain an almost standard sequent calculus PC^* for the fragment of the propositional logic containing only $\rightarrow$ and $\wedge$ formulas[2].

$$\text{(axiom}^*\text{)} \qquad \Gamma, C \vdash_l C$$

$$\rightarrow\text{-L}^* \quad \frac{\Gamma \vdash_l \sigma_1 \qquad \Gamma, \sigma_2 \vdash_l C}{\Gamma, \sigma_1 \rightarrow \sigma_2 \vdash_l C} \qquad \rightarrow\text{-R} \quad \frac{\Gamma, \sigma \vdash_l \tau}{\Gamma \vdash_l \sigma \rightarrow \tau}$$

$$\wedge\text{-L}^* \quad \frac{\Gamma, \sigma_1, \sigma_2 \vdash_l C}{\Gamma, \sigma_1 \wedge \sigma_2 \vdash_l C} \qquad \wedge\text{-R} \quad \frac{\Gamma \vdash_l \sigma \quad \Gamma \vdash_l \tau}{\Gamma \vdash_l \sigma \wedge \tau}$$

The constrain is that the *axioms* and the *left* rules can be applied only under the assumption that the *side* formula is atomic: we will prove that this constrain does not change the set of provable sequents.

It is worth noting that the theorems of PC^* do not correspond to the non-empty types of $\Lambda^s_\wedge$; consider for instance $(A \rightarrow A) \wedge (A \rightarrow (B \rightarrow A))$: this formula is provable in PC^* whereas this type is not inhabited in $\Lambda^s_\wedge$. But PC^* will be useful in the following and we think that it is an interesting system of rules from a logical point of view.

Lemma 1 PC^* *is closed under the* weakening *rule, i.e. the following rule*

$$\frac{\Gamma \vdash_l \tau}{\Gamma, \sigma \vdash_l \tau}$$

is admissible for every formula σ.

Proof Immediate, by induction on the length of the derivation of $\Gamma \vdash_l \tau$.

In the following we will use the *weakening* rule without any explicit mention.

Lemma 2 *The following axioms and rules are admissible in the propositional calculus* PC^*

$$\textit{(axiom)}\ \Gamma, \tau \vdash_l \tau \qquad \rightarrow\textit{-L}\ \frac{\Gamma \vdash_l \sigma_1 \qquad \Gamma, \sigma_2 \vdash_l \tau}{\Gamma, \sigma_1 \rightarrow \sigma_2 \vdash_l \tau} \qquad \wedge\textit{-L}\ \frac{\Gamma, \sigma_1, \sigma_2 \vdash_l \tau}{\Gamma, \sigma_1 \wedge \sigma_2 \vdash_l \tau}$$

[2] It is interesting to note that in [6] a calculus similar to PC^* for a wider fragment of predicative logic is presented, starting from a completely different perspective, with the purpose to make easier the decision procedure for the provability of a formula.

that is, PC^* *is a complete calculus for the fragment of the propositional logic containing only* $\rightarrow$ *and* $\wedge$ *formulas.*

Proof The admissibility of the axioms and the rules must be proved all together by induction on the complexity of the formula τ. The basis case, that is the case τ is an atomic formula, is valid by hypothesis. Thus, let us analyze the case $\tau \equiv \tau_1 \rightarrow \tau_2$:

1. We have to prove that
$$\Gamma, \tau_1 \rightarrow \tau_2 \vdash_l \tau_1 \rightarrow \tau_2$$
By inductive hypothesis, we can assume
$$\Gamma, \tau_1 \vdash_l \tau_1$$
and
$$\Gamma, \tau_2 \vdash_l \tau_2$$
Hence, by using $\rightarrow$-L in the case of the formula τ_2 which is simpler then τ, we can obtain
$$\Gamma, \tau_1 \rightarrow \tau_2, \tau_1 \vdash_l \tau_2$$
and hence we conclude by $\rightarrow$-R.
2. We have to prove that if
(a) $\Gamma \vdash_l \sigma_1$
(b) $\Gamma, \sigma_2 \vdash_l \tau_1 \rightarrow \tau_2$
then
$$\Gamma, \sigma_1 \rightarrow \sigma_2 \vdash_l \tau_1 \rightarrow \tau_2$$
The last rule in the derivation of (b) within PC^* must have been an instance of $\rightarrow$-R whose premise is $\Gamma, \sigma_2, \tau_1 \vdash_l \tau_2$. Thus, by using on the latter sequent and (a) an instance of $\rightarrow$-L in the case of the type τ_2 which is simpler then τ, we obtain
$$\Gamma, \sigma_1 \rightarrow \sigma_2, \tau_1 \vdash_l \tau_2$$
and hence we conclude by $\rightarrow$-R.
3. We have to prove that if
(a) $\Gamma, \sigma_1, \sigma_2 \vdash_l \tau_1 \rightarrow \tau_2$
then
$$\Gamma, \sigma_1 \wedge \sigma_2 \vdash_l \tau_1 \rightarrow \tau_2$$
The last rule in the derivation of (a) within PC^* must have been an instance of $\rightarrow$-R whose premise is $\Gamma, \sigma_1, \sigma_2, \tau_1 \vdash_l \tau_2$. Thus, by using an instance of $\wedge$-L in the case of the type τ_2 which is simpler then τ, we obtain
$$\Gamma, \sigma_1 \wedge \sigma_2, \tau_1 \vdash_l \tau_2$$
and hence we conclude by $\rightarrow$-R.

Now, we have to analyze the case $\tau \equiv \tau_1 \wedge \tau_2$:

1. We have to prove that

$$\Gamma, \tau_1 \wedge \tau_2 \vdash_l \tau_1 \wedge \tau_2$$

   By inductive hypothesis, we can assume

$$\Gamma, \tau_1, \tau_2 \vdash_l \tau_1$$

   and

$$\Gamma, \tau_1, \tau_2 \vdash_l \tau_2$$

   Hence, by using $\wedge$-L in the case of the type τ_1 which is simpler then τ, we obtain

$$\Gamma, \tau_1 \wedge \tau_2 \vdash_l \tau_1$$

   Analogously, we obtain

$$\Gamma, \tau_1 \wedge \tau_2 \vdash_l \tau_2$$

   and hence we conclude by $\wedge$-R.
2. We have to prove that if

   (a) $\Gamma \vdash_l \sigma_1$
   (b) $\Gamma, \sigma_2 \vdash_l \tau_1 \wedge \tau_2$

   then

$$\Gamma, \sigma_1 \to \sigma_2 \vdash_l \tau_1 \wedge \tau_2$$

   The last rule in the derivation of (b) within PC^* must have been an instance of $\wedge$-R whose premises are

$$\Gamma, \sigma_2 \vdash_l \tau_1$$

   and

$$\Gamma, \sigma_2 \vdash_l \tau_2$$

   Thus, by using on (a) and the first of these two sequents an instance of $\to$-L in the case of the type τ_1 which is simpler then τ, we obtain

$$\Gamma, \sigma_1 \to \sigma_2 \vdash_l \tau_1$$

   In a completely analogous way we obtain

$$\Gamma, \sigma_1 \to \sigma_2 \vdash_l \tau_2$$

   and hence we conclude by $\wedge$-R.

3. We have to prove that if

$$\text{(a) } \Gamma, \sigma_1, \sigma_2 \vdash_l \tau_1 \wedge \tau_2$$

then

$$\Gamma, \sigma_1 \wedge \sigma_2 \vdash_l \tau_1 \wedge \tau_2$$

But the last rule in the derivation of (a) within PC^* must have been an instance of $\wedge$-R whose premises are

$$\Gamma, \sigma_1, \sigma_2 \vdash_l \tau_1$$

and

$$\Gamma, \sigma_1, \sigma_2 \vdash_l \tau_2$$

Thus, by using an instance of $\wedge$-L in the case of the type τ_1 which is simpler then τ, we obtain

$$\Gamma, \sigma_1 \wedge \sigma_2 \vdash_l \tau_1$$

In a completely analogous way we obtain

$$\Gamma, \sigma_1 \wedge \sigma_2 \vdash_l \tau_2$$

and hence we conclude by $\wedge$-R.

We will use the structure of the proof of this theorem in the next section.

4 Embedding of $\Lambda_\wedge$ into $\Lambda^s_\wedge$

After theorem 1, in order to prove that all the terms of $\Lambda_\wedge$ are strongly normalizing, it is sufficient to show that they can be typed in $\Lambda^s_\wedge$. In the following lemmas we will show that $\Lambda^s_\wedge$ is indeed closed for the rules of $\Lambda_\wedge$.

Lemma 3 *Let τ be any type. Then $\Gamma, x : \tau \vdash_s x : \tau$.*

To prove lemma 3 we just need to decorate all the judgements that we used in the proof of lemma 2 with suitable terms; in fact, the axiom case is what we are looking for. Anyhow, it is necessary to state before the following lemma of closure under the *weakening* rule of $\Lambda^s_\wedge$.

Lemma 4 *Suppose z is any variable and σ is any type. Then, if $\Gamma \vdash_s c : \tau$ then $\Gamma, z : \sigma \vdash_s c : \tau$.*

Proof The proof is immediate by induction on the length of the derivation of $\Gamma \vdash_s c : \tau$.

It may be useful to note that in this lemma we do not assume that the variable z is fresh.

We are now ready to prove that the following rules are admissible in $\Lambda^s_\wedge$:

$$\Gamma, x : \tau \vdash_s x : \tau$$

$$\frac{\Gamma \vdash_s a : \sigma_1 \qquad \Gamma, y : \sigma_2 \vdash_s y(a_1) \dots (a_n) : \tau}{\Gamma, x : \sigma_1 \to \sigma_2 \vdash_s x(a)(a_1) \dots (a_n) : \tau}$$

$$\frac{\Gamma, x : \sigma_1, x : \sigma_2 \vdash_s x(a_1) \dots (a_n) : \tau}{\Gamma, x : \sigma_1 \wedge \sigma_2 \vdash_s x(a_1) \dots (a_n) : \tau}$$

Also in this case the admissibility of the axioms and the rules must be proved all together by induction on the complexity of the type τ. Most of the cases are straightforward rephrasing of the proof of lemma 2 and hence we will show here only some of them. The basis case, i.e. τ is an atomic type, is valid by hypothesis. Thus, let us analyze the case $\tau \equiv \tau_1 \to \tau_2$:

– We have to prove that

$$\Gamma, x : \tau_1 \to \tau_2 \vdash_s x : \tau_1 \to \tau_2$$

By inductive hypothesis, we can assume

$$\Gamma, y_1 : \tau_1 \vdash_s y_1 : \tau_1$$

and

$$\Gamma, y_1 : \tau_1, y_2 : \tau_2 \vdash_s y_2 : \tau_2$$

for some fresh variables y_1 and y_2. Hence we can obtain

$$\Gamma, x : \tau_1 \to \tau_2, y_1 : \tau_1 \vdash_s x(y_1) : \tau_2$$

and conclude by *abstraction.*

– We have to prove that if

(a) $\Gamma \vdash_s a : \sigma_1$
(b) $\Gamma, y : \sigma_2 \vdash_s y(a_1) \dots (a_n) : \tau_1 \to \tau_2$

then

$$\Gamma, x : \sigma_1 \to \sigma_2 \vdash_s x(a)(a_1) \dots (a_n) : \tau_1 \to \tau_2$$

The last rule in the derivation of (b) within $\Lambda^s_\wedge$ must have been an instance of *abstraction* whose premise is

$$\Gamma, y : \sigma_2, z : \tau_1 \vdash_s y(a_1) \dots (a_n)(z) : \tau_2$$

for some fresh variable z. Thus, by inductive hypothesis, we obtain

$$\Gamma, x : \sigma_1 \to \sigma_2, z : \tau_1 \vdash_s x(a)(a_1) \dots (a_n)(z) : \tau_2$$

and hence we conclude by *abstraction.*

Now, we have to analyze the case $\tau \equiv \tau_1 \wedge \tau_2$:

– We have to prove that

$$\Gamma, x : \tau_1 \wedge \tau_2 \vdash_s x : \tau_1 \wedge \tau_2$$

By inductive hypothesis, we can assume

$$\Gamma, x : \tau_1, x : \tau_2 \vdash_l x : \tau_1$$

and

$$\Gamma, x : \tau_1, x : \tau_2 \vdash_s x : \tau_2$$

Hence we obtain

$$\Gamma, x : \tau_1 \wedge \tau_2 \vdash_s x : \tau_1$$

and

$$\Gamma, x : \tau_1 \wedge \tau_2 \vdash_s x : \tau_2$$

and we conclude by *intersection*.

– We have to prove that if

(a) $\Gamma \vdash_s a : \sigma_1$
(b) $\Gamma, y : \sigma_2 \vdash_s y(a_1) \dots (a_n) : \tau_1 \wedge \tau_2$

then

$$\Gamma, x : \sigma_1 \to \sigma_2 \vdash_s x(a)(a_1) \dots (a_n) : \tau_1 \wedge \tau_2$$

The last rule in the derivation of (b) must have been an instance of *intersection* whose premises are

$$\Gamma, y : \sigma_2 \vdash_s y(a_1) \dots (a_n) : \tau_1$$

and

$$\Gamma, y : \sigma_2 \vdash_l y(a_1) \dots (a_n) : \tau_2$$

Thus, by inductive hypothesis, we obtain

$$\Gamma, x : \sigma_1 \to \sigma_2 \vdash_s x(a)(a_1) \dots (a_n) : \tau_1$$

and

$$\Gamma, x : \sigma_1 \to \sigma_2 \vdash_l x(a)(a_1) \dots (a_n) : \tau_2$$

and hence we conclude by *intersection*.

Also the remaining rule which we stated only for a basic type, i.e. the λ*-introduction* rule, can be generalized to any type.

Theorem 2 *Suppose* $\Gamma \vdash_s c[x := a](a_1) \dots (a_n) : \tau$ *and* $\Gamma \vdash_s a : \sigma$. *Then* $\Gamma \vdash_s (\lambda x.c)(a)(a_1) \dots (a_n) : \tau$.

Proof The proof is by induction on the complexity of the type τ.

– $\tau \equiv C$: the result follows directly by λ-*introduction*.
– $\tau \equiv \tau_1 \to \tau_2$: the last rule in the derivation of

$$\Gamma \vdash_s c[x := a](a_1) \dots (a_n) : \tau_1 \to \tau_2$$

must have been an instance of the *abstraction* rule whose premise is

$$\Gamma, y : \tau_1 \vdash_s c[x := a](a_1) \dots (a_n)(y) : \tau_2$$

for some fresh variable y. Then, by inductive hypothesis,

$$\Gamma, y : \tau_1 \vdash_s (\lambda x.c)(a)(a_1) \dots (a_n)(y) : \tau_2$$

and hence

$$\Gamma \vdash_s (\lambda x.c)(a)(a_1) \dots (a_n) : \tau_1 \to \tau_2$$

by *abstraction*.
– $\tau \equiv \tau_1 \wedge \tau_2$: the last rule in the derivation of

$$\Gamma \vdash_s c[x := a](a_1) \dots (a_n) : \tau_1 \wedge \tau_2$$

must have been an instance of *intersection* whose premises are

$$\Gamma \vdash_s c[x := a](a_1) \dots (a_n) : \tau_1$$

and

$$\Gamma \vdash_s c[x := a](a_1) \dots (a_n) : \tau_2$$

Then, by inductive hypothesis,

$$\Gamma \vdash_s (\lambda x.c)(a)(a_1) \dots (a_n) : \tau_1$$

and

$$\Gamma \vdash_s (\lambda x.c)(a)(a_1) \dots (a_n) : \tau_2$$

and hence

$$\Gamma \vdash_s (\lambda x.c)(a)(a_1) \dots (a_n) : \tau_1 \wedge \tau_2$$

by *intersection*.

The next step is to prove the closure of $\Lambda^s_\wedge$ under the *application* rule. Also in this case we need a preliminary lemma.

Lemma 5 *$\Lambda^s_\wedge$ is closed under substitution, i.e. if*

$$\Gamma, x : \sigma_1, \dots, x : \sigma_n \vdash_s b : \tau,$$

where $x : \sigma_1$, …, $x : \sigma_n$ are all the assumptions of the variable x within the base, and, for any $i = 1, \dots, n$, $\Gamma \vdash_s a : \sigma_i$, then

$$\Gamma \vdash_s b[x := a] : \tau.$$

Proof The proof is by principle induction on the complexity $\mu(\sigma_1, .., \sigma_n)$ of the sequence $\sigma_1, .., \sigma_n$ of the types of the substituted variable and secondary induction on the length of the derivation of the judgment

$$\Gamma, x : \sigma_1, \ldots, x : \sigma_n \vdash_s b : \tau$$

The inductive definition of the complexity measure μ is the following:

$$\mu(\Phi) = \begin{cases} 1 & \text{if } \Phi = C \\ \mu(\sigma) + \mu(\tau) + 1 & \text{if } \Phi = \sigma \to \tau \\ \mu(\sigma) + \mu(\tau) + 1 & \text{if } \Phi = \sigma \wedge \tau \\ \mu(\sigma_1 \wedge \ldots \wedge \sigma_n) & \text{if } \Phi = \sigma_1, \ldots, \sigma_n \end{cases}$$

The intended meaning is to give a standard measure on types and consider a sequence of types like the intersection of all of them.

Most of the cases do not use at all the main inductive hypothesis and they work in a straightforward way by secondary inductive hypothesis. Let us consider here only some cases (in order to simplify the notation we will write $\overline{x} : \overline{\sigma}$ as a shorthand for $x : \sigma_1, \ldots, x : \sigma_n$).

– (λ-introduction)

$$\frac{\Gamma, \overline{x} : \overline{\sigma} \vdash_s c[y := d](a_1) \ldots (a_n) : C \quad \Gamma, \overline{x} : \overline{\sigma} \vdash_s d : \delta}{\Gamma, \overline{x} : \overline{\sigma} \vdash_s (\lambda y.c)(d)(a_1) \ldots (a_n) : C}$$

By secondary inductive hypothesis, we obtain both

$$\Gamma \vdash_s c[y := d][x := a](a_1[x := a])..(a_n[x := a]) : C$$

and

$$\Gamma \vdash_s d[x := a] : \delta$$

But y is an abstracted variable and hence we can assume that it does not appear in a; hence the first judgement is

$$\Gamma \vdash_s c[x := a][y := d[x := a]](a_1[x := a]) \ldots (a_n[x := a]) : C$$

Thus, we obtain

$$\Gamma \vdash_s (\lambda y.c[x := a])(d[x := a])(a_1[x := a]) \ldots (a_n[x := a]) : C$$

by λ-*introduction*.

– (abstraction)

$$\frac{\Gamma, \overline{x} : \overline{\sigma}, y : \tau_1 \vdash_s b(y) : \tau_2}{\Gamma, \overline{x} : \overline{\sigma} \vdash_s b : \tau_1 \to \tau_2} \quad y \notin \mathsf{FV}(b) \text{ and } y \notin \Gamma, \overline{x} : \overline{\sigma}$$

By secondary inductive hypothesis, we obtain

$$\Gamma, y : \tau_1 \vdash_s b(y)[x := a] : \tau_2$$

Now, the side condition on the *abstraction* rule yields that $x \neq y$ and thus $b(y)[x := a] \equiv b[x := a](y)$; hence we conclude by an instance of the *abstraction* rule.

– (basic rule on $y \neq x$)

$$\Gamma, \overline{x} : \overline{\sigma}, y : C \vdash_s y : C$$

In this case we have just to take $\Gamma, y : C \vdash_s y : C$, which is an axiom.

– (impl-left on $y \neq x$)

$$\frac{\Gamma, \overline{x} : \overline{\sigma} \vdash_s c : \tau_1 \quad \Gamma, \overline{x} : \overline{\sigma}, z : \tau_2 \vdash_s z(a_1)..(a_n) : C}{\Gamma, \overline{x} : \overline{\sigma}, y : \tau_1 \to \tau_2 \vdash_s y(a)(a_1)..(a_n) : C}$$

where $z \notin \mathsf{FV}(a_1) \cup \ldots \cup \mathsf{FV}(a_n)$ and $z \notin \Gamma, \overline{x} : \overline{\sigma}$.
By secondary inductive hypothesis, we obtain both

$$\Gamma \vdash_s c[x := a] : \tau_1$$

and

$$\Gamma, z : \tau_2 \vdash_s z(a_1) \ldots (a_n)[x := a] : C$$

Now, the side condition on the *impl-left* rule yields that $x \neq z$ and thus $z(a_1) \ldots (a_n)[x := a] \equiv z(a_1[x := a]) \ldots (a_n[x := a])$. Hence we conclude by an instance of the *impl-left* rule.

More complex are the cases when the last rule used in the proof of $\Gamma, \overline{x} : \overline{\sigma} \vdash_s b : \tau$ is an instance of an *axiom* or an instance of one of the *left* rules and the substituted variable is the *main* variable in this rule.

– Suppose $\Gamma, \overline{x} : \overline{\sigma}, x : C \vdash_s x : C$ is an instance of an *axiom*. Then we have to prove $\Gamma \vdash_s a : C$. But, in this case, this is one of the assumptions.

– Suppose the last rule in the derivation of $\Gamma, \overline{x} : \overline{\sigma} \vdash_s b : \tau$ is

$$\frac{\Gamma, \overline{x} : \overline{\sigma} \vdash_s d : \tau_1 \qquad \Gamma, \overline{x} : \overline{\sigma}, y : \tau_2 \vdash_s y(a_1) \ldots (a_n) : C}{\Gamma, \overline{x} : \overline{\sigma}, x : \tau_1 \to \tau_2 \vdash_s x(d)(a_1) \ldots (a_n) : C}$$

where $y \notin \mathsf{FV}(a_1) \cup \ldots \cup \mathsf{FV}(a_n)$ and $y \notin \Gamma, \overline{x} : \overline{\sigma}$. By secondary inductive hypothesis, we obtain

$$\text{(a)} \qquad \Gamma \vdash_s d[x := a] : \tau_1$$

and

$$\text{(b)} \qquad \Gamma, y : \tau_2 \vdash_s y(a_1) \ldots (a_n)[x := a] : C$$

and, since the side condition yields $y \neq x$, we get

$$y(a_1)..(a_n)[x := a] \equiv y(a_1[x := a])..(a_n[x := a])$$

Let us consider now the judgement $\Gamma \vdash_s a : \tau_1 \to \tau_2$; the last rule used in its derivation must have been an instance of the *abstraction* rule whose premise is $\Gamma, z : \tau_1 \vdash_s a(z) : \tau_2$ for some fresh variable z which does not appear in Γ. Hence, by principle inductive hypothesis,

since $\mu(\tau_1) < \mu(\overline{\sigma}, \tau_1 \to \tau_2)$, we obtain $\Gamma \vdash_s a(d[x := a]) : \tau_2$ by substituting the term $d[x := a]$ in (a) for z. Hence, again by principle inductive hypothesis since $\mu(\tau_2) < \mu(\overline{\sigma}, \tau_1 \to \tau_2)$ and $y \notin \Gamma$, we obtain

$$\Gamma \vdash_s a(d[x := a])(a_1[x := a]) \dots (a_n[x := a]) : C$$

by substituting $\Gamma \vdash_s a(d[x := a]) : \tau_2$ for y in (b) since, for any $i = 1, \dots, n$, $y \notin \mathsf{FV}(a_i)$.

– Suppose the last rule in the derivation of $\Gamma, \overline{x} : \overline{\sigma} \vdash_s b : \tau$ is

$$\frac{\Gamma, \overline{x} : \overline{\sigma}, x : \tau_1, x : \tau_2 \vdash_s x(a_1) \dots (a_n) : C}{\Gamma, \overline{x} : \overline{\sigma}, x : \tau_1 \wedge \tau_2 \vdash_s x(a_1) \dots (a_n) : C}$$

Let us consider the judgement $\Gamma \vdash_s a : \tau_1 \wedge \tau_2$; the last rule used in its derivation must have been an instance of the *intersection* rule whose premises are $\Gamma \vdash_s a : \tau_1$ and $\Gamma \vdash_s a : \tau_2$. Hence, by secondary inductive hypothesis, since $\mu(\overline{\sigma}, \tau_1 \wedge \tau_2) = \mu(\overline{\sigma}, \tau_1, \tau_2)$ and the derivation of $\Gamma, \overline{x} : \overline{\sigma}, x : \tau_1, x : \tau_2 \vdash_s x(a_1) \dots (a_n) : C$ is shorter than the derivation of $\Gamma, \overline{x} : \overline{\sigma}, x : \tau_1 \wedge \tau_2 \vdash_s x(a_1) \dots (a_n) : C$, we can substitute

(a) $\Gamma \vdash_s \overline{a} : \overline{\sigma}$
(b) $\Gamma \vdash_s a : \tau_1$
(c) $\Gamma \vdash_s a : \tau_2$

in

$$\Gamma, \overline{x} : \overline{\sigma}, x : \tau_1, x : \tau_2 \vdash_s x(a_1) \dots (a_n) : C$$

and obtain

$$\Gamma \vdash_s a(a_1[x := a]) \dots (a_n[x := a]) : C$$

Since the substitution is performed in a non-standard way, it may be useful to illustrate how it works by means of an example.

Suppose $\Gamma \vdash_s a : A \wedge (A \to B)$ and that we want to substitute the term a for the variable x, which does not appear in Γ, within the following derivation:

$$\dfrac{\dfrac{\Gamma, x : A \vdash_s x : A \quad \Gamma, x : A, y : B \vdash_s y : B}{\Gamma, x : A, x : A \to B \vdash_s x(x) : B}\ \text{impl-left}}{\Gamma, x : A \wedge (A \to B) \vdash_s x(x) : B}\ \text{inter-left}$$

Then, we first analyze the proof of $\Gamma \vdash_s a : A \wedge (A \to B)$ and we obtain that the last step has to have been an instance of the *intersection* rule whose premises are (a) $\Gamma \vdash_s a : A$ and (b) $\Gamma \vdash_s a : A \to B$. By using (a), we substitute the variable x both in the axiom $\Gamma, x : A \vdash_s x : A$, and we get (c) $\Gamma \vdash_s a : A$, and in the axiom $\Gamma, x : A, y : B \vdash_s y : B$ and we get (d) $\Gamma, y : B \vdash_s y : B$. Now, let us analyze the second premise

$\Gamma \vdash_s a : A \to B$; it must have been derived by using an instance of *abstraction* rule whose premise is $\Gamma, z : A \vdash_s a(z) : B$ for some fresh variable z. Hence, we substitute the term a derived in (c) for the variable z and we obtain $\Gamma \vdash_s a(a) : B$. Finally, we substitute the term $a(a)$ in this judgement for the variable y in (d) and we obtain $\Gamma \vdash_s a(a) : B$.

We can now prove the admissibility of the *application* rule.

Lemma 6 *If $\Gamma \vdash_s c : \sigma \to \tau$ and $\Gamma \vdash_s a : \sigma$ then $\Gamma \vdash_s c(a) : \tau$.*

Proof The last rule in the derivation of $\Gamma \vdash_s c : \sigma \to \tau$ must have been an instance of the *abstraction* rule whose premise is $\Gamma, x : \sigma \vdash_s c(x) : \tau$, where the only occurrence of the variable x in $c(x)$ is the manifested one and $x \notin \Gamma$. Then, by the previous lemma, $\Gamma \vdash_s c(x)[x := a] \equiv c(a) : \tau$.

Now we have to prove closure under lambda abstraction.

Lemma 7 *If $\Gamma, x : \sigma \vdash_s c : \tau$ and $x \notin \Gamma$ then $\Gamma \vdash_s \lambda x.c : \sigma \to \tau$.*

Proof Since $\Gamma, x : \sigma \vdash_s x : \sigma$ is provable by lemma 3 and $c[x := x] \equiv c$, $\Gamma, x : \sigma \vdash_s c : \tau$ yields $\Gamma, x : \sigma \vdash_s (\lambda x.c)(x) : \tau$ because of lemma 2. Hence we obtain $\Gamma \vdash_s \lambda x.c : \sigma \to \tau$ by *abstraction* since we assumed that $x \notin \Gamma$.

Finally, we have to deal with the rules for the intersection types, but they are immediate: the *intersection introduction* rule is present in both the systems while the admissibility of the *intersection elimination* rules follows from the fact that the last rule in the only possible derivation of $\Gamma \vdash_s c : \sigma \wedge \tau$ within $\Lambda^s_\wedge$ must have been an instance of the *intersection* rule whose premises are $\Gamma \vdash_s c : \sigma$ and $\Gamma \vdash_s c : \tau$.

5 Weak normalization and strong normalization in pure lambda calculus

We have already observed that one of the main results on $\Lambda_\wedge$ is a characterization theorem which states that a term of the pure lambda calculus Λ is strongly normalizing if and only if it can be typed within $\Lambda_\wedge$. After the proof of immersion of $\Lambda_\wedge$ into $\Lambda^s_\wedge$ of the previous section we know that the same theorem holds also for $\Lambda^s_\wedge$. Anyhow, we give here a short and direct proof of this result. We think that this proof is interesting because it helps to understand how to use directly $\Lambda^s_\wedge$ and it suggests how to obtain later a similar characterization for the weakly normalizing terms of Λ. Supposing c is a strongly normalizing lambda term, the proof that it is possible to give

it a type within $\Lambda^s_\wedge$ will be by main induction on the depth $\delta(c)$ of its *reduction tree* and secondary induction on its term complexity $|c|$. We recall here the definition of the function δ.

$$\delta(c) = \begin{cases} 0 & \text{if } c \text{ is in normal form} \\ \mathsf{max}(\delta(c_1), .., \delta(c_n)) + 1 & \text{if } c_1, .., c_n \text{ are the terms such that } c \rightsquigarrow_\beta c_i \end{cases}$$

Before the proof of the main theorem, we need the following lemma.

Lemma 8 *Let*

$$\Gamma, x : \sigma_1, \ldots, x : \sigma_n \vdash_s b : \tau,$$

where $x : \sigma_1, \ldots, x : \sigma_n$ are all the assumptions of the variable x within the base. Then

$$\Gamma, x : \sigma_1 \wedge \ldots \wedge x : \sigma_n \vdash_s b : \tau.$$

Proof The proof is immediate by induction on the length of the derivation of $\Gamma, x : \sigma_1, \ldots, x : \sigma_n \vdash_s b : \tau$

We can now prove the annunced theorem.

Theorem 3 *Let c be any strongly normalizing lambda term. Then, there are a base Γ and a type σ such that $\Gamma \vdash_s c : \sigma$.*

Proof Let us analyze the possible cases according to the shape of the term c.

- $c \equiv x$ for some variable x. This is the basic case in the induction. In this case we put $x : C \vdash_s x : C$, for some basic type C, and this judgement is an axiom in $\Lambda^s_\wedge$.
- $c \equiv x(a_1) \ldots (a_n)$. In this case, for any $i = 1, \ldots, n$, $\delta(a_i) \leq \delta(c)$ and $|a_i| < |c|$. Then, by inductive hypothesis, there exist basis Γ_i and types σ_i such that $\Gamma_i \vdash_s a_i : \sigma_i$. Then consider the following derivation (recall that $\Lambda^s_\wedge$ is closed under the *weakening* rule):

$$\dfrac{\dfrac{\dfrac{\bigcup \Gamma_i \vdash_s a_1 : \sigma_1 \qquad \bigcup \Gamma_i, y_1 : \sigma_2 \to .. \to \sigma_n \to C \vdash_s y_1(a_2)..(a_n) : C}{\bigcup \Gamma_i, x : \sigma_1 \to .. \to \sigma_n \to C \vdash_s x(a_1)..(a_n) : C} \quad \bigcup \Gamma_i \vdash_s a_2 : \sigma_2 \quad \bigcup \Gamma_i, y_2 : \sigma_3 \to .. \to \sigma_n \to C \vdash_s y_2(a_3)..(a_n) : C}{\vdots} \quad \bigcup \Gamma_i \vdash_s a_n : \sigma_n \quad \bigcup \Gamma_i, y_n : C \vdash_s y_n : C}{\bigcup \Gamma_i, y_{n-1} : \sigma_n \to C \vdash_s y_{n-1}(a_n) : C}$$

- $c \equiv \lambda x.d$. In this case $\delta(d) = \delta(c)$ and $|d| < |c|$. Then, by inductive hypothesis, there exist a base Γ and a type σ such that $\Gamma \vdash_s d : \sigma$. Observe that $\Gamma \equiv \Gamma', x : \sigma_1, \ldots, x : \sigma_n$ for some Γ' such that $x \notin \Gamma'$, and hence, by the previous lemma 8,

$$\Gamma', x : \sigma_1 \wedge \ldots \wedge \sigma_n \vdash_s d : \sigma$$

Then, by lemma 7,

$$\Gamma' \vdash_s \lambda x.d : \sigma_1 \wedge \ldots \wedge \sigma_n \to \sigma$$

since $x \notin \Gamma'$.

– $c \equiv (\lambda x.d)(a)(a_1)\ldots(a_n)$. Then, $\delta(d[x := a](a_1)\ldots(a_n)) < \delta(c)$ and $\delta(a) \leq \delta(c)$ and $|a| < |c|$. Then, by main inductive hypothesis, there exist a base Γ_1 and a type σ_1, such that

$$\Gamma_1 \vdash_s d[x := a](a_1)\ldots(a_n) : \sigma_1,$$

and, by secondary inductive hypothesis, there exist a base Γ_2 and a type σ_2 such that $\Gamma_1 \vdash_s a : \sigma_2$. Then, by closure under weakening and lemma 2, we obtain

$$\Gamma_1, \Gamma_2 \vdash_s (\lambda x.d)(a)(a_1)\ldots(a_n) : \sigma_1 \to \sigma_2$$

It is interesting to note that a straightforward modification of the rules of $\Lambda^s_\wedge$ allows to give a type exactly to the weakly normalizing terms of Λ. To this aim, it is sufficient to consider the typed lambda calculus $\Lambda^w_\wedge$ obtained from $\Lambda^s_\wedge$ by modifying the *λ-introduction* rule as follows (in order to distinguish the judgements of $\Lambda^w_\wedge$ from those of $\Lambda^s_\wedge$ we write $\vdash_w$ instead of $\vdash_s$):

$$(\lambda_w\text{-introduction}) \qquad \frac{\Gamma \vdash_w c[x := a](a_1)\ldots(a_n) : C}{\Gamma \vdash_w (\lambda x.c)(a)(a_1)\ldots(a_n) : C} \quad a \in \Lambda$$

The following theorem is immediate.

Theorem 4 *Let $\Gamma \vdash_w a : \sigma$ then a is weakly normalizing.*

Proof The proof is immediate by induction on the length of the derivation of $\Gamma \vdash_w a : \sigma$.

Now, we want to prove that also the other implication holds. To this aim, let us recall that a reduction strategy is an algorithm which maps lambda terms into lambda terms and respects the transitive closure of β-reduction. Then, it is possible to show that if a lambda term is weakly normalizing then it can be normalized by using the following reduction strategy L (see [1]).

Definition 1 *The* leftmost outermost reduction strategy *is the map on the set of lambda terms recursively defined as follows:*

$$\begin{array}{ll} \mathsf{L}(x(a_1)\ldots(a_n)) & = x(\mathsf{L}(a_1))\ldots(\mathsf{L}(a_n)) \quad n \geq 0 \\ \mathsf{L}(\lambda x.c) & = \lambda x.\mathsf{L}(c) \\ \mathsf{L}((\lambda x.c)(a)(a_1)\ldots(a_n)) & = \mathsf{L}(c[x := a](a_1)\ldots(a_n)) \quad n \geq 0 \end{array}$$

Now, we can prove that any weakly normalizing term c can be typed within $\Lambda^w_\wedge$. The proof of this theorem is completely similar to the proof of the previous theorem 3, but we need to change the induction paramiter since the term c is no longer supposed to be strongly normalizing.

Theorem 5 *Let c be a weakly normalizing lambda term. Then, there are a base Γ and a type σ such that $\Gamma \vdash_w c : \sigma$.*

Proof Since c is weakly normalizing, it can be normalized by using the reduction strategy L. Thus, induction on the number of steps of the reduction strategy L is a correct proof method.

- Suppose the number of steps in the normalization of c by using L is 0. Then c must be a variable x. Thus, we put $x : C \vdash_w x : C$, for some basic type C, and this judgement is an axiom of $\Lambda^w_\wedge$.
- Suppose that the number of steps of application of L in the normalization of c is $k > 0$. Then one of the following cases apply.
  1. $c \equiv x(a_1)\dots(a_n)$. In this case, for any $i = 1, \dots, n$, the number of applications of the reduction strategy L which are necessary in order to normalize a_i is lower than k. Then, by inductive hypothesis, there exist bases Γ_i and types σ_i such that $\Gamma_i \vdash_w a_i : \sigma_i$. Now we can continue as in the similar case in proof of theorem 3 since also $\Lambda^w_\wedge$ is closed under *weakening*.
  2. $c \equiv \lambda x.d$. Then, the number of applications of the reduction strategy L necessary in order to normalize the term d is lower than k and hence, by inductive hypothesis, there exist a base Γ and a type σ such that $\Gamma \vdash_w d : \sigma$. Hence we can conclude as in the similar case in the proof of theorem 3, since we can prove for $\Lambda^w_\wedge$ a version of lemma 7.
  3. $c \equiv (\lambda x.d)(a)(a_1)\dots(a_n)$. Then, the number of applications of the reduction strategy L necessary in order to normalize the term $d[x := a](a_1)\dots(a_n)$ is lower than k. Thus, there exist a base Γ and a type σ, such that $\Gamma \vdash_w d[x := a](a_1)\dots(a_n) : \sigma$. Hence, by closure under weakening and a version for $\Lambda^w_\wedge$ of lemma 2, we obtain $\Gamma \vdash_w (\lambda x.d)(a)(a_1)\dots(a_n) : \sigma$

Acknowledgments. I want to thank Mario Coppo for his careful reading of the first version of this work and for his precious suggestions which made this work to look in the way it does. Moreover, I want to thank Michele Bugliesi for some very useful discussions on the best presentation for the rules of $\Lambda^s_\wedge$.

References

[1] H. Barendregt, The Lambda Calculus, its Syntax and Semantics, North–Holland, Amsterdam, 1984.

[2] Capretta, V., Valentini, S., A general method to prove the normalization theorem for first and second order typed λ-calculi, to appear in Mathematical Structure in Computer Science.
[3] Dezani, M., Giovannetti, E., De Liguoro, U., Intersection types, λ-models and Böhm trees, to appear.
[4] Girard, J., Lafont, Y., Taylor, P., Proofs and Types, Cambridge University Press, Cambridge, 1989.
[5] Krivine, J.L., Lambda-Calculus, Types and Models, Masson, Paris, Ellis Horwood, Hemel Hempstead, 1993.
[6] Miller, D., Nadathur, G., Pfenning, F., Scedrov, A., Uniform proofs as a foundation for logic programming, Annals of Pure and Applied Logic, 51 (1991), pp. 125-157.
[7] Pottinger, G., A type assignment for strongly normalizable λ-terms, in "To H.B. Curry, Essay on Combinatory Logic, Lambda Calculus and Formalism", Academic Press, New York, 1980, pp. 561-577.
[8] Tait, W.W., Intensional interpretation of functionals of finite type I, J. of Symbolic Logic, 32, 1967, pp. 198-212.
[9] Valentini, S., A note on a straightforward proof of normal form theorem for simply typed λ-calculi, Boll. Un. Mat. It., 8-A, 1994, pp. 207-213.