Abstract

We consider reducibility sets defined not by induction on types but by induction on sequents as a tool to prove strong normalisation of systems with explicit substitution. To illustrate this point, we give a proof of strong normalisation (SN) for simply-typed call-by-name $\overline{\lambda}\mu\tilde{\mu}$-calculus enriched with operators of explicit unary substitutions. The $\overline{\lambda}\mu\tilde{\mu}$-calculus, defined by Curien & Herbelin, is a variant of $\lambda\mu$-calculus with a *let* operator that exhibits symmetries such as terms/contexts and call-by-name/call-by-value reduction.

Introduction

Explicit substitution in λ-calculus

The traditional theory of λ-calculus relies on β-reduction, that is the capture by a function of its argument followed by the process of substituting this argument to the places where it is used. The latter is a global process whose computational complexity may be linear in the size of the function (at least for a tree representation of λ-terms).

We obtain a λ-calculus with explicit substitution when the process of substitution is made explicit in the calculus. This is achieved by adding a new operator to the syntax, say $t[x\leftarrow u]$, and by making explicit the rules of propagation of the substitution in the process of β-reduction. This has for effect to turn β-reduction from a global reduction system (expressible as a higher-order rewriting system) to a local one (expressible as a first-order rewriting system). Let's remark the underlying rewriting system is necessarily overlapping since both reductions of $((\lambda x.t)u)[y\leftarrow v]$ into $t[x\leftarrow u][y\leftarrow v]$ and into $((\lambda x.t)[y\leftarrow v])(u[y\leftarrow v])$ are necessary to simulate arbitrary usual β-reduction.

The traditional representation of bindings in λ-terms uses names and bindings are considered up to α-conversion, i.e. up to non-clashing renaming of bound variable names. Such a representation hides a sometimes mandatory process of renaming (consider for instance $(\lambda z.(zz))\lambda x.\lambda y.(xy)$). The problematic rule is the following propagation rule: $(\lambda x.t)[y\leftarrow u] \longrightarrow \lambda x.(t[y\leftarrow u])$ (with $x \neq y$) for which the binder λx may capture a free variable in u. From a typing point of view, this propagation rule hides an implicit usage of the weakening rule (the context of binding variables for u is now extended by x) and we need to make weakening explicit to get a truly local rewriting system.

On computers, evaluation of programs/functions applied to arguments has traditionally been implemented using indices pointing into a vector of delayed (i.e. explicit) substitutions, called environments (in higher-order functional programming community) or stacks (in the imperative programming community which usually considers only first-order functions). A model for such an evaluation mechanism is sometimes called abstract machine (at least in the functional programming community).

Designed in 1986 by Curien [6], the $\lambda\sigma$-calculus can be seen as the first λ-calculus allowing to talk about delayed/explicit substitution and abstract machines. Since then, a lot of variants of the $\lambda\sigma$-calculus have been developed. The simplest of them, in the sense it is up to α-conversion and uses the minimal apparatus for substitution, is the λx-calculus deeply studied in Bloo & Rose [3] (but described by Lins [23] in an independent context).

Various naming strategies are possible in order to make explicit the weakening rule in λx-calculus. Besides the simple renaming of x into a variable not occurring in u (notations as above), two strategies based on numbers have specially been studied. The first uses so-called de Bruijn levels (see Lescanne and Rouyer-Degli $\lambda\chi$-calculus [22]) and the second uses so-called de Bruijn indices (see λs- and λt-calculi by Kamareddine & Ríos [19] and $\lambda\upsilon$ by Lescanne [21][1]).

A good survey on the current state of the study of λ-calculus with substitution made explicit can be found in the introduction of [10].

Cut rule and explicit substitution

Along the lines of the Curry-Howard correspondence, the proofs of implicational Gentzen's natural deduction [13] are λ-terms and Prawitz' process of normalisation for natural deduction [27] is β-reduction. Explicitation of substitution can be transfered to natural deduction and it happens that substitution is nothing but a cut rule.

The cut rule is central in this other kind of formal systems: Gentzen's sequent calculi LJ and LK [13]. In contrast with the prominent rôle of β-reduction in λ-calculus and natural deduction, the traditional approach of normalisation in sequent calculus precisely relies on the progressive elimination of cuts through application of local rewriting steps.

To our knowledge, the first strong normalisation proof for sequent calculus LK comes from Dragalin [12]. The reduction system he considers is a non confluent rewriting system but without overlapping between distinct redexes. Reading it along the lines of the correspondence with λ-calculus in [16, 7], it lacks the full generality of the propagation rule $(tu)[x{\leftarrow}v] \longrightarrow (t[x{\leftarrow}v])(u[x{\leftarrow}v])$, a rule necessary to simulate β-reduction. Therefore, strong normalisation of, say λx-calculus, cannot be inferred (at least directly) from Dragalin's proof.

Another approach of strong normalisation for sequent calculus is inherited from Girard's strong normalisation of proof nets [14]. It has allowed for various confluent [8] and non confluent [30] strong normalisation results for LK, all based on a global reduction system related to β-reduction.

What about strong normalisation of a reduction system including fine-grained Dragalin's cut-elimination but extended to allow simulation of β-reduction too? This would first unify both approaches for strong normalisation of LK but since natural deduction is compositionally embeddable in sequent calculus (see [13] or, for a more formal account, [7]), it would also provide for strong normalisation for, say λx-calculus. This is the purpose of this paper.

The $\overline{\lambda}\mu\tilde{\mu}$-calculus

The $\overline{\lambda}\mu\tilde{\mu}$-calculus defined by Curien & Herbelin [7] is a variant of Parigot's $\lambda\mu$-calculus [24] with a *let* operator that provides with a term notation for proofs of LK. It embeds also various λ-calculi in their call-by-name (CBN) or call-by-value (CBV) variants, such as $\lambda\mu$-calculus and usual λ-calculus.

[1]The only differences in the latter three calculi is the representation of indices (primitive numbers or recursively defined ones) and consequently the granularity of the lifting process and variables substitution process.

We focus in the paper on a CBN variant of simply-typed $\overline{\lambda}\mu\tilde{\mu}$-calculus equipped with a fine-grained reduction system allowing simulation of β. This version is very close from LK equipped with the same kind of reduction system. Therefore proving strong normalisation of this variant of $\overline{\lambda}\mu\tilde{\mu}$-calculus is not far from proving strong normalisation for CBN LK (which, according to [7], corresponds to the system LK^t in [8]).

In the first three sections of the paper, we present $\overline{\lambda}\mu\tilde{\mu}$-calculus, its CBN version with explicit substitution and its typing system. We don't go in the detail of renaming and only consider a version up to α-conversion of variables expressed by names.

Strong normalisation proofs

The traditional approach to strong normalisation (SN) in λ-calculi with explicit substitution is through the property of preservation of strong normalisation (PSN). This property states that if a term without explicit substitution strongly normalises then it strongly normalises when the substitution (and renaming if any) is made explicit, that is when the process of substitution (and the process of weakening if any) is turned into a fine-grained reduction process allowing for possible interleaving with other reductions.

For all systems of the λ-cube[2] with names and enriched by a unary explicit substitution (as in λx-calculus), Bloo [2] has inferred SN from PSN. Regarding the calculi with de Bruijn indices, let us mention that SN from PSN has been proved for λs-calculus by Kamareddine & Ríos [20]. Some proofs have also been proposed using an embedding into proof nets (assuming reduction in proof nets is proved strongly normalising). This works for David & Guillaume λ_l-calculus (see Di Cosmo *et al* [11]) but fails to manage some pathological interferences between weakening and substitution for the λx-calculus style of reduction. Notice that David & Guillaume λ_l-calculus [9] is precisely designed in order to forbid these interferences[3].

Proofs by reducibility

Our own approach is to develop a direct proof by reducibility[4]. It turned out this approach was already followed by Bonelli [5] to prove strong normalisation for Church-style system F with explicit substitution. Bonelli's proof is based on a standard definition of reducibility sets, with the extra condition of closure by reverse propagation of substitution. In contrast, our definition of reducibility sets is by induction not on types but on sequents, what we find well-suited to deal with explicit substitutions.

Unfortunately, our definition of reducibility sets is not powerful enough to cope with the interferences between renaming and substitutions mentioned above. This problem takes place in the weakening lemma and the only kind of proof we found

[2]The λ-cube contains all intermediate systems from simply-typed λ-calculus to the Coquand's Calculus of Constructions

[3]The problem concerns subterms of the form $u[x := t]$ where u was originally known not to contain x but the information has been lost when u, as a substituend, has traversed some $\lambda x.v$. In λ_l-calculus the information is preserved and $[x := t]$ is erased instead of propagated as in λx-calculus.

[4]Reducibility has been introduced by Tait [29] to prove the normalisation of Gödel's system T. It has been reused in various frameworks, most noticeable is probably its extension for system F by Girard [15].

is a combinatory proof, similar to what is done in PSN proofs, but contrasting a lot with the regularity of the definition of the reducibility sets (the same occurs in Bonelli's proof where the problem is covered by a so-called perpetuality proposition, independent of the general pattern of proof by reducibility). Notice that the proof is not by contradiction as usual in PSN proofs, but based on a direct induction.

Weakening lemma, and an associated strengthening lemma, are proved in the fifth section. Then we can complete the strong normalisation proof. This is done in the sixth section.

For the sake of completeness, let us mention a peculiar definition of reducibility by Ritter [28] to prove PSN from SN for a variant of $\lambda\sigma$-calculus [6]. There, reducibility is defined neither on types, nor on sequents but on the well-foundedness of the strong normalisation property of terms without explicit substitution.

Simple proofs

In fact, it turns out that PSN directly implies SN for a large class of typed λ-calculi, including for instance Curry-style system F. For a calculus up to α-conversion, just raise all subterms of the form $t[x := u]$ into $(\lambda x.t\ u)$ to get a term to which PSN applies. Since the transformation preserves typing, the original term (a reduct of the transformed one) is strongly normalising. For a calculus with explicit renaming, the same idea seems to apply by reasoning on a canonical representative (to which the transformation is applicable) of each α-conversion class.

On another side, Bloo & Rose (see [4], or, for a more general result [2]), proved that for a large subset of interesting Combinatory Reduction Systems (CRS), the extension with explicit substitution (up to α-conversion) satisfies the PSN property. This includes λ-calculus and $\overline{\lambda}\mu\tilde{\mu}$-calculus which therefore are SN as soon as their version without explicit substitution is.

A first version of this paper appeared in [18]. As pointed out by an anonymous referee, the simulation of Parigot's $\lambda\mu$-calculus with explicit substitution into $\overline{\lambda}\mu\tilde{\mu}$-calculus described in [18] is wrong. We found no direct simulation that works and then removed the section from the current paper.

1 The $\overline{\lambda}\mu\tilde{\mu}$-Calculus

The syntax relies on three syntactic categories: terms, contexts and commands. They are respectively written using lowercase v, lowercase e and lowercase c. Values are a subset of terms. They are written with a capital V. Applicative contexts are a subset of contexts. They are written with a capital E.

A set $\mathcal{X}$ of term variables is given. Their elements are written with lowercase x, y, z... A set $\mathcal{A}$ of continuation variables is given. Their elements are written with lowercase Greek α, β, γ... The syntax of $\overline{\lambda}\mu\tilde{\mu}$-calculus is:

$$\begin{aligned} c &::= \langle v|e\rangle \\ v &::= V \mid \mu\alpha.c \\ V &::= x \mid \lambda x.v \\ e &::= E \mid \tilde{\mu}x.c \\ E &::= \alpha \parallel v \cdot e \end{aligned}$$

The symbols λ and μ are those of Parigot's $\lambda\mu$-calculus [24]. The construction $\tilde{\mu}x.c$ builds a context. It must be read as *let* $x = [\ ]$ *in* c where $[\ ]$ is the hole of the context. The construction $v \cdot e$ also builds a context. It must be understood as $[\ ]v$ taking the place of the hole of e. The construction $\langle v|e\rangle$ is a cut. It can be seen as putting v in the hole of e.

Read $\langle v|\alpha\rangle$ as ordinary $[\alpha]v$ of $\lambda\mu$-calculus and $\langle v|v_1 \cdots v_n \cdot \alpha\rangle$ as $[\alpha]vv_1 \ldots v_n$ and finally $\langle v|v_1 \cdots v_n \cdot \tilde{\mu}x.c\rangle$ as *let* $x = vv_1 \ldots v_n$ *in* c.

Free and bound variables are defined as usual. We write $x \in v$ to express that x occurs free in v. Similarly for the various other combinations with x or α, and v, E, e or c. We reason up to α-conversion of bound term and context variables. The notations $[x/c]v$ and $[\alpha/c]e$ denote here meta-substitution.

Evaluation rules are:

$$\begin{array}{lrcl} & \langle \lambda x.v|v' \cdot e\rangle & \longrightarrow & \langle v'|\tilde{\mu}x.\langle v|e\rangle\rangle \\ (\mu) & \langle \mu\alpha.c|e\rangle & \longrightarrow & [\alpha/c]e \\ (\tilde{\mu}) & \langle v|\tilde{\mu}x.c\rangle & \longrightarrow & [x/c]v \end{array}$$

The last two rules form a critical pair. Call-by-value consists in giving priority to the (μ)-redexes, while call-by-name gives priority to the $(\tilde{\mu})$-redexes. We refer to [7] for a deeper account on $\overline{\lambda}\mu\tilde{\mu}$-calculus.

2 Call-by-name $\overline{\lambda}\mu\tilde{\mu}$-Calculus with Explicit Substitution

For call-by-name $\overline{\lambda}\mu\tilde{\mu}$-calculus, we can simplify the first rule into[5]

$$\langle \lambda x.v|v' \cdot e\rangle \quad \longrightarrow \quad \langle [x/v]v'|e\rangle$$

This is the rule we adopt in the sequel but the main novelty is that we enrich the syntax with an operator of unary explicit substitution. Since the rule $(\tilde{\mu})$ has priority, the substitution may carry any kind of term but only applicative contexts. We use names based on the Greek letter τ to denote unary substitutions. For the same reason, distinction between terms and values becomes useless. The syntax is

$$\begin{array}{l} c ::= \langle v|e\rangle \mid c\tau \\ v ::= x \parallel \lambda x.v \mid \mu\alpha.c \mid v\tau \\ e ::= E \mid \tilde{\mu}x.c \mid e\tau \\ E ::= \alpha \mid v \cdot e \parallel E\tau \\ \tau ::= [x{\leftarrow}v] \parallel [\alpha{\leftarrow}E] \end{array}$$

The source $Dom(\tau)$ of τ is x if $\tau = [x{\leftarrow}v]$ and α if $\tau = [\alpha{\leftarrow}E]$. The substituend $S(\tau)$ is v in the first case and E in the second.

The evaluation rules given in previous section are now turned into their variant with explicit substitution. Moreover, we syntactically enforce the priority of the rule $(\tilde{\mu})$ by now expecting an applicative context E in the (μ) rule:

[5]That is we force the reduction of the $(\tilde{\mu})$ redex produced by the rule

$$\begin{array}{lrcl} & \langle \lambda x.v | v' \cdot e \rangle & \longrightarrow & \langle v[x \leftarrow v'] | e \rangle \\ (\mu) & \langle \mu\alpha.c | E \rangle & \longrightarrow & c[\alpha \leftarrow E] \\ (\tilde{\mu}) & \langle v | \tilde{\mu}x.c \rangle & \longrightarrow & c[x \leftarrow v] \end{array}$$

We complete these rules with the following set of propagation rules for substitution

$$\begin{array}{lccl} \langle v|e\rangle\tau & \longrightarrow & \langle v\tau|e\tau\rangle & \\ & & & \\ x[x \leftarrow v] & \longrightarrow & v & \\ x\tau & \longrightarrow & x & (\text{if } x \neq Dom(\tau)) \\ (\lambda x.v)\tau & \longrightarrow & \lambda x.(v\tau) & (\text{for } x \text{ chosen s.t. } x \neq Dom(\tau) \text{ and } x \notin S(\tau)) \\ (\mu\alpha.c)\tau & \longrightarrow & \mu\alpha.(c\tau) & (\text{for } \alpha \text{ chosen s.t. } \alpha \neq Dom(\tau) \text{ and } \alpha \notin S(\tau)) \\ & & & \\ \alpha[\alpha \leftarrow E] & \longrightarrow & E & \\ \alpha\tau & \longrightarrow & \alpha & (\text{if } \alpha \neq Dom(\tau)) \\ (v \cdot e)\tau & \longrightarrow & (v\tau)\cdot(e\tau) & \\ (\tilde{\mu}x.c)\tau & \longrightarrow & \tilde{\mu}x.(c\tau) & (\text{for } x \text{ chosen s.t. } x \neq Dom(\tau) \text{ and } x \notin S(\tau)) \end{array}$$

We add also two simplification rules[6]:

$$\begin{array}{lcll} \mu\alpha.\langle v|\alpha\rangle & \longrightarrow & v & (\text{if } \alpha \notin v) \\ \tilde{\mu}x.\langle x|e\rangle & \longrightarrow & e & (\text{if } x \notin e) \end{array}$$

A generalized substitution σ is a sequence of unary substitutions:

$$\sigma \quad ::= \quad [\,] \parallel \sigma\tau$$

The notation $\sigma\sigma'$ denotes the concatenation of σ and σ'. We denote by $c\sigma$ the term c where the unary substitutions in σ are successively applied to c as follows: $c(\sigma\tau) = (c\sigma)\tau$ and $c[\,] = c$. Similarly for $v\sigma$ and $e\sigma$. We say σ reduces to σ', and write $\sigma \longrightarrow \sigma'$ when exactly one of its component is replaced by a reduct of it. This means it is inductively defined by the following cases:

$$\frac{\sigma \longrightarrow \sigma'}{\tau\sigma \longrightarrow \tau\sigma'} \qquad \frac{v \longrightarrow v'}{[x \leftarrow v]\sigma \longrightarrow [x \leftarrow v']\sigma} \qquad \frac{E \longrightarrow E'}{[\alpha \leftarrow E]\sigma \longrightarrow [\alpha \leftarrow E']\sigma}$$

We write $\stackrel{*}{\longrightarrow}$ for the reflexive-transitive closure of $\longrightarrow$. A term (resp context, command, substitution) is normal if all instances of $\langle v|e\rangle$ in it have the form $\langle \lambda x.v'|\alpha\rangle$, $\langle x|v'\cdot e'\rangle$ or $\langle x|\alpha\rangle$ and if it contains no explicit substitution. This is equivalent to saying that no rule of reduction applies. A term (resp context, command, substitution) is strongly normalising either because it is normal or because all its reducts are strongly normalising. We write $\mathcal{SN}$ for the set of all strongly normalising terms (resp contexts, commands, substitutions). That a term t is $\mathcal{SN}$ is then inductively defined. This allows us to reason in the sequel by induction on the proof of $t \in \mathcal{SN}$[7].

[6] They have no computational meaning in the sense that they do not correspond to some cut-elimination

[7] This is classically equivalent to the definition of $t \in \mathcal{SN}$ as the absence of infinite sequence of reduction starting from t and to a reasoning by induction on the maximal length of such a set of finite sequences

We now consider (weak-)head-reduction of applicative contexts. We say $E \downarrow_h E'$ in the following cases:

$$\frac{}{(v \cdot e)\tau \downarrow_h (v\tau) \cdot (e\tau)} \qquad \frac{Dom(\tau) \neq \alpha}{\alpha\tau \downarrow_h \alpha} \qquad \frac{E \downarrow_h E'}{\alpha[\alpha{\leftarrow}E] \downarrow_h E'}$$

$$\frac{}{\alpha \downarrow_h \alpha} \qquad \frac{}{v \cdot e \downarrow_h v \cdot e} \qquad \frac{E \downarrow_h E' \quad E'\tau \downarrow_h E''}{E\tau \downarrow_h E''}$$

The relation $E \downarrow_h E'$ is functional: for any E, there is a unique applicative context E' such that $E \downarrow_h E'$. We write $E' = \mathit{hnf}(E)$. We have the following lemmas:

Lemma 1 $E \longrightarrow E'$ implies $\mathit{hnf}(E) \stackrel{*}{\longrightarrow} \mathit{hnf}(E')$.

Proof By induction on the proof of $E \downarrow_h \mathit{hnf}(E)$.

Lemma 2 Let $E \in \mathcal{SN}$ and $\tau \in \mathcal{SN}$.
If $E \downarrow_h v \cdot e$ with $v\tau \in \mathcal{SN}$ and $e\tau \in \mathcal{SN}$ then $E\tau \in \mathcal{SN}$.
If $E \downarrow_h \alpha$ then $E\tau \in \mathcal{SN}$.

Proof By induction on the proofs of $E \in \mathcal{SN}$, $\tau \in \mathcal{SN}$ and on the proofs of $E \downarrow_h v \cdot e$ or $E \downarrow_h \alpha$. If the reduction occurs in τ then we use induction hypothesis on the strong normalisation of τ. If the reduction occurs in E, i.e. $E\tau \longrightarrow E'\tau$ then, by lemma 1, either $E' \downarrow_h v' \cdot e'$ with $v \stackrel{*}{\longrightarrow} v'$, $e \stackrel{*}{\longrightarrow} e'$ and therefore $v'\tau, e'\tau \in \mathcal{SN}$, or $E' \downarrow_h \alpha$. In both cases, we then apply the induction hypothesis on the strong normalisation of E. If $E = v \cdot e$ and $E\tau \longrightarrow v\tau \cdot e\tau$ then any further reduction occurs either in $v\tau$ or $e\tau$ and we are done by hypothesis on the strong normalisability of $v\tau$ and $e\tau$. If $E = \alpha$, $\tau = [\alpha{\leftarrow}E']$ and $E\tau \longrightarrow E'$ then $E' \in \mathcal{SN}$ since $\tau \in \mathcal{SN}$. Finally if $E = \alpha$, $Dom(\tau) \neq \alpha$ and $E\tau \longrightarrow \alpha$ then no more reduction can occur.

3 Simply–typed Call–by–name $\overline{\lambda}\mu\tilde{\mu}$–Calculus with Explicit Substitution

We now give a typing system for call-by-name $\overline{\lambda}\mu\tilde{\mu}$-calculus with explicit substitution. Types are built from atoms and $\rightarrow$. Antecedents (succedents) are sets of term (context) variables of a given type. Variables are supposed distinct in antecedents and succedents. We write Γ (Δ) for sets of term (context) variables of a given type. The union of Γ and Γ' is written Γ, Γ' and similarly for succedents. The union is assumed to be disjoint.

There are five kinds of judgments corresponding to the five kinds of syntactic constructions.

$$\begin{array}{l} c : (\Gamma \vdash \Delta) \\ \Gamma \vdash v : A \,|\, \Delta \\ \Gamma \,|\, e : A \vdash \Delta \\ \Gamma \,;\, E : A \vdash \Delta \\ \tau : (\Gamma \vdash \Delta) \Rightarrow (\Gamma' \vdash \Delta') \end{array}$$

A judgment relates a command, a term, a context, an applicative context or a substitution and a pure typing information. We call sequent this pure typing part. That is, the five judgments above can respectively be read as a command c typed by the sequent $\Gamma \vdash \Delta$, a term v typed by the sequent $\Gamma \vdash A \,|\, \Delta$, a context e typed by the sequent $\Gamma \,|\, A \vdash \Delta$, an applicative context E typed by the sequent $\Gamma\,;\, A \vdash \Delta$ and a unary substitution τ typed by the sequent $(\Gamma \vdash \Delta) \Rightarrow (\Gamma' \vdash \Delta')$.

For judgments $(\Gamma \vdash v : A \,|\, \Delta)$, $(\Gamma \,|\, e : A \vdash \Delta)$ and $(\Gamma\,;\, E : A \vdash \Delta)$, we say A is the active formula. There is no explicit weakening rule (weakening is taken into account in the axiom rules) and no explicit contraction rule (contraction is implicit in the cut rule and in the rule building applicative contexts). The rules which do not involve explicit substitution are the following (see [7] for more details):

$$\frac{\Gamma \vdash v : A \,|\, \Delta \qquad \Gamma \,|\, e : A \vdash \Delta}{\langle v | e \rangle : (\Gamma \vdash \Delta)}$$

$$\frac{}{\Gamma\,;\, \alpha : A \vdash \Delta, \alpha : A} \qquad \frac{}{\Gamma, x : A \vdash x : A \,|\, \Delta}$$

$$\frac{c : (\Gamma, x : A \vdash \Delta)}{\Gamma \,|\, \tilde{\mu}x.c : A \vdash \Delta} \qquad \frac{c : (\Gamma \vdash \beta : B, \Delta)}{\Gamma \vdash \mu\beta.c : B \,|\, \Delta}$$

$$\frac{\Gamma \vdash v : A \,|\, \Delta \qquad \Gamma \,|\, e : B \vdash \Delta}{\Gamma\,;\, v \cdot e : A \rightarrow B \vdash \Delta} \qquad \frac{\Gamma, x : A \vdash v : B \,|\, \Delta}{\Gamma \vdash \lambda x.v : A \rightarrow B \,|\, \Delta}$$

$$\frac{\Gamma\,;\, E : A \vdash \Delta}{\Gamma \,|\, E : A \vdash \Delta}$$

In addition, we have the following rules which type explicit substitutions:

$$\frac{\Gamma \vdash v : A \,|\, \Delta}{[x \leftarrow v] : (\Gamma, x : A \vdash \Delta) \Rightarrow (\Gamma \vdash \Delta)} \qquad \frac{\Gamma\,;\, E : A \vdash \Delta}{[\alpha \leftarrow E] : (\Gamma \vdash \Delta, \alpha : A) \Rightarrow (\Gamma \vdash \Delta)}$$

$$\frac{\Gamma \,|\, e : A \vdash \Delta \quad \tau : (\Gamma \vdash \Delta) \Rightarrow (\Gamma' \vdash \Delta')}{\Gamma' \,|\, e\tau : A \vdash \Delta'} \qquad \frac{\Gamma \vdash v : A \,|\, \Delta \quad \tau : (\Gamma \vdash \Delta) \Rightarrow (\Gamma' \vdash \Delta')}{\Gamma' \vdash v\tau : A \,|\, \Delta'}$$

$$\frac{\Gamma\,;\, E : A \vdash \Delta \quad \tau : (\Gamma \vdash \Delta) \Rightarrow (\Gamma' \vdash \Delta')}{\Gamma'\,;\, E\tau : A \vdash \Delta'}$$

$$\frac{c : (\Gamma \vdash \Delta) \qquad \tau : (\Gamma \vdash \Delta) \Rightarrow (\Gamma' \vdash \Delta')}{c\tau : (\Gamma' \vdash \Delta')}$$

4 Reducibility Sets

The reducibility method comes from Tait [29] who used it to prove the normalisation of Gödel's system T. It has been extended to system F by Girard [15] and to second-order $\lambda\mu$-calculus (classical system F) by Parigot [25].

Strong normalisation with explicit substitution raises several problems. In a calculus without operator of explicit substitution, a term can only interact against its arguments. In the presence of explicit substitutions, a term can also interact with a substitution applied to it. The idea of reducibility is to anticipate the possible interactions a term is liable to encounter according to its type. To apply this idea to a calculus with explicit substitution, the natural choice is to associate reducibility sets not to types but to sequents.

Call-by-name $\overline{\lambda}\mu\tilde{\mu}$-calculus is based on 5 kinds of judgments (including judgments for unary substitutions). Then, we defined 5 kinds of reducibility sets. In our framework, "interaction" is asymmetric: one side "absorbs" the other. Commands and applicative contexts only absorb substitutions. The terms $\lambda x.v$ and $\mu\alpha.c$ absorb both substitutions and applicative contexts (because it is call-by-name). And finally, a non applicative context $\tilde{\mu}x.c$ absorbs all, substitutions and terms. This suggests a definition of reducibility sets by mutual induction on the length of antecedents and succedents of sequents, on the active formula if any and on the following order on judgment: "contexts" > "terms" > "commands" (see below).

To get strong normalisation, especially to get the weakening lemma working (see lemma 12), we need a strong control on free variables. In particular, we didn't succeed to avoid to put an order on the variables declared in the sequents, both term and context variables merged together. A simple way to express this order is to use sequents not of the form, say $x_1 : A_1, x_2 : A_2, ... \vdash \alpha_1 : B_1, \alpha_2 : B_2, ...$ where no order between term variables and context variables is expressible, but of the form, say $\vdash x_1 : A_1, \alpha_1 : B_1, \alpha_2 : B_2, x_2 : A_2, ...$ But then we lose the logical meaning of right-hand-side and left-hand-side of the deduction symbol. Our choice is to parameterize sequents with an external ordered set of variables.

Since there is no closed term typed by a sequent with empty antecedent and succedent, we need an initial set of extra variables which does not occur anywhere in derivations. Let $\mathcal{V}_0$ be such a set assumed to contain at least one term variable and at least one context variable.

Let $\mathcal{V}$ be an ordered sequence of variables in $\mathcal{X} \cup \mathcal{A}$. We say that $\mathcal{V}$ is compatible with $c : (\Gamma \vdash \Delta)$ or $\Gamma \vdash v : A \,|\, \Delta$ or $\Gamma\,;\, E : A \vdash \Delta$ or $\Gamma \,|\, e : A \vdash \Delta$ if it contains the variables in $\mathcal{V}_0$ and all the variables declared in Γ and Δ. We say the pair $(x\mathcal{V}, \mathcal{V})$ is compatible with $(\Gamma, x : A \vdash \Delta) \Rightarrow (\Gamma \vdash \Delta)$ and the pair $(\alpha\mathcal{V}, \mathcal{V})$ is compatible with $(\Gamma \vdash \alpha : A, \Delta) \Rightarrow (\Gamma \vdash \Delta)$ if $\mathcal{V}$ contains the variables in $\mathcal{V}_0$ and all the variables declared in Γ and Δ. Besides being an ordered sequence, it may happen that we consider $\mathcal{V}$ as a set in which case it is the set of its variables.

We can now turn to the definition of reducibility sets.

Main definition *(Reducibility sets)*

- $v \in [\![\Gamma \vdash_{\mathcal{V}} A \,|\, \Delta]\!]$ iff
  - for all $E \in [\![\Gamma\,;\, A \vdash_{\mathcal{V}} \Delta]\!]$, $\langle v|E\rangle \in [\![\Gamma \vdash_{\mathcal{V}} \Delta]\!]$

– and for all $\tau \in [\![(\Gamma\vdash_{\mathcal{V}}\Delta) \Rightarrow (\Gamma'\vdash_{\mathcal{V}'}\Delta')]\!], v\tau \in [\![\Gamma'\vdash_{\mathcal{V}'}A \,|\, \Delta']\!]$

- $E \in [\![\Gamma \,;\, A\vdash_{\mathcal{V}}\Delta]\!]$ iff $E \in \mathcal{SN}$ and
  - either $E \downarrow_h \alpha$
  - or $E \downarrow_h v \cdot e$ and $A = B\to C$, with $v \in [\![\Gamma\vdash_{\mathcal{V}}B \,|\, \Delta]\!]$ and $e \in [\![\Gamma \,|\, C\vdash_{\mathcal{V}}\Delta]\!]$

- $e \in [\![\Gamma \,|\, A\vdash_{\mathcal{V}}\Delta]\!]$ iff
  - for all $v \in [\![\Gamma\vdash_{\mathcal{V}}A \,|\, \Delta]\!], \langle v|e\rangle \in [\![\Gamma\vdash_{\mathcal{V}}\Delta]\!]$
  - and for all $\tau \in [\![(\Gamma\vdash_{\mathcal{V}}\Delta) \Rightarrow (\Gamma'\vdash_{\mathcal{V}'}\Delta')]\!], e\tau \in [\![\Gamma' \,|\, A\vdash_{\mathcal{V}'}\Delta']\!]$

- $c \in [\![\Gamma\vdash_{\mathcal{V}}\Delta]\!]$ iff
  - either Γ and Δ are empty and $c \in \mathcal{SN}$
  - or for all $\tau \in [\![(\Gamma\vdash_{\mathcal{V}}\Delta) \Rightarrow (\Gamma'\vdash_{\mathcal{V}'}\Delta')]\!], c\tau \in [\![\Gamma'\vdash_{\mathcal{V}'}\Delta']\!]$

- $[\![(\Gamma\vdash_{\mathcal{V}}\Delta) \Rightarrow (\Gamma'\vdash_{\mathcal{V}'}\Delta')]\!]$ is characterized by the following two cases:
  - $[x\leftarrow v] \in [\![(\Gamma, x : A\vdash_{\mathcal{V}x\mathcal{V}'}\Delta) \Rightarrow (\Gamma\vdash_{\mathcal{V}\mathcal{V}'}\Delta)]\!]$ soon as $v \in [\![\Gamma\vdash_{\mathcal{V}\mathcal{V}'}A \,|\, \Delta]\!]$ *and* the free variables of v are in $\mathcal{V}'$
  - $[\alpha\leftarrow E] \in [\![(\Gamma\vdash_{\mathcal{V}\alpha\mathcal{V}'}\Delta, \alpha : A) \Rightarrow (\Gamma\vdash_{\mathcal{V}\mathcal{V}'}\Delta)]\!]$ as soon as $E \in [\![\Gamma \,;\, A\vdash_{\mathcal{V}\mathcal{V}'}\Delta]\!]$ *and* the free variables of E are in $\mathcal{V}'$

Remarks: 1) If $v \in [\![\Gamma\vdash_{\mathcal{V}}A \,|\, \Delta]\!]$ (as an example) then $\mathcal{V}$ is intended to assign an order to the variables declared in Γ and Δ and also to contain $\mathcal{V}_0$ (even if not strictly necessary in the definition). 2) No extra condition on the typability of terms, contexts or commands is required to have them belonging to a reducibility set. 3) The definition of $[\![(\Gamma\vdash_{\mathcal{V}}\Delta) \Rightarrow (\Gamma'\vdash_{\mathcal{V}'}\Delta')]\!]$ implies that Γ' is included in Γ, Δ' in Δ and that $\mathcal{V}'$ is a strict subsequence of $\mathcal{V}$. We will implicitly use this fact throughout the proofs.

We now extend the definition of $[\![(\Gamma\vdash_{\mathcal{V}}\Delta) \Rightarrow (\Gamma\vdash_{\mathcal{V}'}\Delta)]\!]$ to generalized substitutions as follows:

- $[\,] \in [\![(\Gamma\vdash_{\mathcal{V}}\Delta) \Rightarrow (\Gamma\vdash_{\mathcal{V}}\Delta)]\!]$
- if $\sigma \in [\![(\Gamma\vdash_{\mathcal{V}}\Delta) \Rightarrow (\Gamma'\vdash_{\mathcal{V}'}\Delta')]\!]$ and $\tau \in [\![(\Gamma'\vdash_{\mathcal{V}'}\Delta') \Rightarrow (\Gamma''\vdash_{\mathcal{V}''}\Delta'')]\!]$ then $\sigma\tau \in [\![(\Gamma\vdash_{\mathcal{V}}\Delta) \Rightarrow (\Gamma''\vdash_{\mathcal{V}''}\Delta'')]\!]$

We can then rephrase the definition of $[\![\Gamma\vdash_{\mathcal{V}}A \,|\, \Delta]\!]$, $[\![\Gamma \,|\, A\vdash_{\mathcal{V}}\Delta]\!]$ and $[\![\Gamma\vdash_{\mathcal{V}}\Delta]\!]$:

- $v \in [\![\Gamma\vdash_{\mathcal{V}}A \,|\, \Delta]\!]$ iff we have $\langle v\sigma|E\rangle\sigma' \in \mathcal{SN}$ for all $\sigma \in [\![(\Gamma\vdash_{\mathcal{V}}\Delta) \Rightarrow (\Gamma'\vdash_{\mathcal{V}'}\Delta')]\!]$, $E \in [\![\Gamma' \,;\, A\vdash_{\mathcal{V}'}\Delta']\!]$ and $\sigma' \in [\![(\Gamma'\vdash_{\mathcal{V}'}\Delta') \Rightarrow (\vdash_{\mathcal{V}''})]\!]$.
- $e \in [\![\Gamma \,|\, A\vdash_{\mathcal{V}}\Delta]\!]$ iff we have $\langle v|e\sigma\rangle\sigma' \in \mathcal{SN}$ for all $\sigma \in [\![(\Gamma\vdash_{\mathcal{V}}\Delta) \Rightarrow (\Gamma'\vdash_{\mathcal{V}'}\Delta')]\!]$, $v \in [\![\Gamma'\vdash_{\mathcal{V}'}A \,|\, \Delta']\!]$ and $\sigma' \in [\![(\Gamma'\vdash_{\mathcal{V}'}\Delta') \Rightarrow (\vdash_{\mathcal{V}''})]\!]$.

- $c \in [\![\Gamma \vdash_{\mathcal{V}} \Delta]\!]$ iff $c\sigma \in \mathcal{SN}$ for all $\sigma \in [\![(\Gamma \vdash_{\mathcal{V}} \Delta) \Rightarrow (\vdash_{\mathcal{V}'})]\!]$.

Here is a quick justification. For the "only if" part, reason by induction on the length on σ then on the length of σ' (if any). For the "if" part, reason by induction on the length of Γ and Δ. For $[\![\Gamma \vdash_{\mathcal{V}} \Delta]\!]$, this is quite direct. For $[\![\Gamma \vdash_{\mathcal{V}} A \mid \Delta]\!]$ and $[\![\Gamma \mid A \vdash_{\mathcal{V}} \Delta]\!]$ only the first case of the definition needs to be shown when both Γ and Δ are empty. This is direct. Otherwise, both cases must be proved. For the first one, use the induction hypothesis for $[\![\Gamma \vdash_{\mathcal{V}} \Delta]\!]$. For the second one, use the induction hypothesis on the same category of object (i.e. v or e).

We now turn to the strong normalisation proof. In all the sequel, we assume to have variables sequences $\mathcal{V}$ compatible with the sequent considered. In particular, such sequences contain $\mathcal{V}_0$. We first show that reducibility sets are closed by reduction.

Lemma 3 (Closure by reduction)

(1) If $E \in [\![\Gamma\,;\, A \vdash_{\mathcal{V}} \Delta]\!]$ and $E \longrightarrow E'$ then $E' \in [\![\Gamma\,;\, A \vdash_{\mathcal{V}} \Delta]\!]$
(2) If $c \in [\![\Gamma \vdash_{\mathcal{V}} \Delta]\!]$ and $c \longrightarrow c'$ then $c' \in [\![\Gamma \vdash_{\mathcal{V}} \Delta]\!]$
(3) If $v \in [\![\Gamma \vdash_{\mathcal{V}} A \mid \Delta]\!]$ and $v \longrightarrow v'$ then $v' \in [\![\Gamma \vdash_{\mathcal{V}} A \mid \Delta]\!]$
(4) If $e \in [\![\Gamma \mid A \vdash_{\mathcal{V}} \Delta]\!]$ and $e \longrightarrow e'$ then $e' \in [\![\Gamma \mid A \vdash_{\mathcal{V}} \Delta]\!]$
(5) If $\sigma \in [\![(\Gamma \vdash_{\mathcal{V}} \Delta) \Rightarrow (\Gamma' \vdash_{\mathcal{V}'} \Delta')]\!]$ and $\sigma \longrightarrow \sigma'$ then $\sigma' \in [\![(\Gamma \vdash_{\mathcal{V}} \Delta) \Rightarrow (\Gamma' \vdash_{\mathcal{V}'} \Delta')]\!]$

Proof We reason by induction on the length of Γ and Δ, on the active formula if any and on the following order of judgments: "contexts" > "terms" > "commands".

(1) Let $E \in [\![\Gamma\,;\, A \vdash_{\mathcal{V}} \Delta]\!]$ and assume $E \longrightarrow E'$. By definition $E \in \mathcal{SN}$ and either $E \downarrow_h \alpha$ or $E \downarrow_h v \cdot e$ with $A = B \to C$, $v \in [\![\Gamma \vdash_{\mathcal{V}} B \mid \Delta]\!]$ and $e \in [\![\Gamma \mid C \vdash_{\mathcal{V}} \Delta]\!]$. Since $E \in \mathcal{SN}$, so does E'. By lemma 1, $E' \downarrow_h \alpha$ or $E' \downarrow_h v' \cdot e'$ with $v \xrightarrow{*} v'$ and $e \xrightarrow{*} e'$. The first case directly implies $E' \in [\![\Gamma\,;\, A \vdash_{\mathcal{V}} \Delta]\!]$, while the second needs the induction hypothesis on A (for v or/and for e).

(2) Let $c \in [\![\Gamma \vdash_{\mathcal{V}} \Delta]\!]$ and assume $c \longrightarrow c'$. If Γ and Δ are empty then c is in $\mathcal{SN}$ and its reduct c' is in $\mathcal{SN} = [\![\vdash_{\mathcal{V}}]\!]$ too. Otherwise, we show that $c'\tau \in [\![\Gamma \vdash_{\mathcal{V}} \Delta]\!]$ for all $\tau \in [\![(\Gamma \vdash_{\mathcal{V}} \Delta) \Rightarrow (\Gamma' \vdash_{\mathcal{V}'} \Delta')]\!]$. By definition of $[\![\Gamma \vdash_{\mathcal{V}} \Delta]\!]$ we have $c\tau \in [\![\Gamma' \vdash_{\mathcal{V}'} \Delta']\!]$ which, by induction hypothesis on Γ and Δ, is stable by reduction and then contains $c'\tau$.

(3) Let $v \in [\![\Gamma \vdash_{\mathcal{V}} A \mid \Delta]\!]$ and assume $v \longrightarrow v'$. By reasoning as for c', we can show that for all $\tau \in [\![(\Gamma \vdash_{\mathcal{V}} \Delta) \Rightarrow (\Gamma' \vdash_{\mathcal{V}'} \Delta')]\!]$, $v'\tau \in [\![\Gamma' \vdash_{\mathcal{V}'} A \mid \Delta']\!]$. It remains to show that for all $E \in [\![\Gamma\,;\, A \vdash_{\mathcal{V}} \Delta]\!]$, $\langle v'|E\rangle \in [\![\Gamma \vdash_{\mathcal{V}} \Delta]\!]$. By definition of v we have $\langle v|E\rangle \in [\![\Gamma \vdash_{\mathcal{V}} \Delta]\!]$ and the induction hypothesis on the order of judgments ensures that its reduct $\langle v'|E\rangle$ is in $[\![\Gamma \vdash_{\mathcal{V}} \Delta]\!]$ too.

(4) We prove the case $e \in [\![\Gamma \mid A \vdash_{\mathcal{V}} \Delta]\!]$ with $e \longrightarrow e'$ as the case v and, finally, the statement (5) is an immediate consequence of (1) and (3), observing that the set of free variables cannot grow by reduction.

Then we show that reducibility sets contain variables and are included in $\mathcal{SN}$. The fact that $\mathcal{V}$ contains at least one term variable and one context variable is crucial for proving that the reducibility sets are not empty.

Lemma 4

(1) $\mathcal{V} \cap \mathcal{A} \subset [\![\Gamma\,;\,A \vdash_{\mathcal{V}} \Delta]\!] \subset \mathcal{SN}$
(2) $[\![\Gamma \vdash_{\mathcal{V}} \Delta]\!] \subset \mathcal{SN}$
(3) $\mathcal{V} \cap \mathcal{X} \subset [\![\Gamma \vdash_{\mathcal{V}} A \,|\, \Delta]\!] \subset \mathcal{SN}$
(4) $[\![\Gamma \,|\, A \vdash_{\mathcal{V}} \Delta]\!] \subset \mathcal{SN}$
(5) $[\![(\Gamma \vdash_{\mathcal{V}} \Delta) \Rightarrow (\Gamma' \vdash_{\mathcal{V}'} \Delta')]\!] \subset \mathcal{SN}$

Proof The statement (1) is a direct consequence of the definition of $[\![\Gamma\,;\,A \vdash_{\mathcal{V}} \Delta]\!]$. We prove the other statements by mutual induction on Γ and Δ and the order on judgment kinds.

Let us first consider statement (2) and recall that any $\mathcal{V}$ contains $\mathcal{V}_0$. If $c \in [\![\Gamma, x : A \vdash_{\mathcal{V}} \Delta]\!]$, then any $x_0 \in \mathcal{V}_0$ is in $[\![\Gamma \vdash_{\mathcal{V}} A \,|\, \Delta]\!]$ by induction hypothesis and it is enough to consider $c[x \leftarrow x_0] \in [\![\Gamma \vdash_{\mathcal{V}} \Delta]\!]$ to get by induction hypothesis $c \in \mathcal{SN}$. If $c \in [\![\vdash_{\mathcal{V}} \alpha : A, \Delta]\!]$ we proceed similarly considering a context variable. Finally if $c \in [\![\vdash_{\mathcal{V}}]\!]$, there is nothing to prove.

To show that any term variable $x \in \mathcal{V} \cap \mathcal{A}$ is in $[\![\Gamma \vdash_{\mathcal{V}} A \,|\, \Delta]\!]$ (first part of statement (3)), we prove that for all $\sigma \in [\![(\Gamma \vdash_{\mathcal{V}} \Delta) \Rightarrow (\Gamma' \vdash_{\mathcal{V}'} \Delta')]\!]$, $E \in [\![\Gamma'\,;\,A \vdash_{\mathcal{V}'} \Delta']\!]$, $\sigma \in [\![(\Gamma' \vdash_{\mathcal{V}'} \Delta') \Rightarrow (\vdash_{\mathcal{V}''})]\!]$, $\langle x\sigma | E \rangle \sigma' \in \mathcal{SN}$.

By definition, $E \in \mathcal{SN}$, and by induction hypothesis, all terms and contexts in σ and σ' are also in $\mathcal{SN}$. We reason by induction on the length of Γ' and Δ' and by induction on the strong normalisability of E, σ and σ'.

- If the reduction occurs in E, σ or σ' then its reduct is reducible by lemma 3 and we are precisely done by induction hypothesis on the strong normalisability.
- If $\sigma = \tau\sigma''$ with $Dom(\tau) \neq x$, $\tau \in [\![(\Gamma \vdash_{\mathcal{V}} \Delta) \Rightarrow (\Gamma'' \vdash_{\mathcal{V}''} \Delta'')]\!]$ and $\langle x\tau\sigma'' | E \rangle \sigma' \longrightarrow \langle x\sigma'' | E \rangle \sigma'$, then the latter is in $\mathcal{SN}$. This is because x is also in $[\![\Gamma'' \vdash_{\mathcal{V}''} A \,|\, \Delta'']\!]$ by induction hypothesis on the length of Γ and Δ, which implies $\langle x\sigma'' | E \rangle \sigma' \in \mathcal{SN}$ by definition.
- If $\sigma = [x \leftarrow v]\sigma''$ and $\langle x[x \leftarrow v]\sigma'' | E \rangle \sigma' \longrightarrow \langle v\sigma'' | E \rangle \sigma'$ then the latter is in $[\![\vdash_{\mathcal{V}''}]\!] = \mathcal{SN}$ by definition of $v \in [\![\Gamma' \vdash_{\mathcal{V}'} A \,|\, \Delta']\!]$.
- If $\sigma' = \tau\sigma''$ and $\langle x\sigma | E \rangle \tau\sigma'' \longrightarrow \langle x\sigma\tau | E\tau \rangle \sigma''$ then we are done by induction hypothesis on the length of Γ' and Δ'.

To show that all v in $[\![\Gamma \vdash_{\mathcal{V}} A \,|\, \Delta]\!]$ is in $\mathcal{SN}$ (second part of statement (3)), just take $\alpha_0 \in \mathcal{V}_0 \subset [\![\Gamma\,;\,A \vdash_{\mathcal{V}} \Delta]\!]$. Since $\langle v | \alpha_0 \rangle \in [\![\Gamma \vdash_{\mathcal{V}} \Delta]\!]$ which is included in $\mathcal{SN}$ by induction hypothesis, we have $v \in \mathcal{SN}$. Similarly for statement (4): to show that all e in $[\![\Gamma \vdash_{\mathcal{V}} A \,|\, \Delta]\!]$ is in $\mathcal{SN}$, just take $x_0 \in \mathcal{V}_0 \subset [\![\Gamma \vdash_{\mathcal{V}} A \,|\, \Delta]\!]$ which has just been proved. Since $\langle x_0 | e \rangle \in [\![\Gamma \vdash_{\mathcal{V}} \Delta]\!]$ which is included in $\mathcal{SN}$ by induction hypothesis, we have $e \in \mathcal{SN}$. Finally, statement (5) is a direct consequence of (1) and (3).

All kinds of reducibility sets are, by definition, closed by substitution except the ones for applicative contexts. This is the purpose of the following lemma.

Lemma 5 $E \in [\![\Gamma\,;\,A \vdash_{\mathcal{V}} \Delta]\!]$ implies that we have $E\sigma \in [\![\Gamma'\,;\,A \vdash_{\mathcal{V}'} \Delta']\!]$ for all $\sigma \in [\![(\Gamma \vdash_{\mathcal{V}} \Delta) \Rightarrow (\Gamma' \vdash_{\mathcal{V}'} \Delta')]\!]$

Proof We reason by induction on σ. For the case $[\,]$, there is nothing to prove. Otherwise, we are in the case $\sigma\tau$ with $\sigma \in [\![(\Gamma \vdash_{\mathcal{V}} \Delta) \Rightarrow (\Gamma'' \vdash_{\mathcal{V}''} \Delta'')]\!]$.

By induction hypothesis for σ, we already know $E\sigma \in \mathcal{SN}$. Then, assume $E\sigma \downarrow_h v \cdot e$ with $A = B \rightarrow C$, $v \in [\![\Gamma'' \vdash_{\mathcal{V}''} B \mid \Delta'']\!]$ and $e \in [\![\Gamma'' \mid C \vdash_{\mathcal{V}''} \Delta'']\!]$. Then $E\sigma\tau \downarrow_h v\tau \cdot e\tau$ and, by definition of the reducibility of v, e and τ, we get $v\tau \in [\![\Gamma' \vdash_{\mathcal{V}'} B \mid \Delta']\!]$ and $e\tau \in [\![\Gamma' \mid C \vdash_{\mathcal{V}'} \Delta']\!]$. By lemma 4 for v, e and τ and lemma 2 we get $E\sigma\tau \in \mathcal{SN}$ and then the conclusion.

On the opposite, assume $E\sigma \downarrow_h \alpha$. By reducibility of τ, lemma 4 and lemma 2, we have $E\sigma\tau \in \mathcal{SN}$. If $Dom(\tau) \neq \alpha$ then clearly $E\sigma\tau \downarrow_h \alpha$. If $\tau = [\alpha \leftarrow E']$ with $E' \in [\![\Gamma' \vdash_{\mathcal{V}'} A \mid \Delta']\!]$ then either $E' \downarrow_h v \cdot e$ with some conditions on v and e or $E' \downarrow_h \beta$, hence $E\sigma\tau \downarrow_h v \cdot e$ with the same conditions or $E\sigma\tau \downarrow_h \beta$, that is $E\sigma\tau \in [\![\Gamma' \vdash_{\mathcal{V}'} A \mid \Delta']\!]$ in all cases.

5 Weakening Lemma

We now want to state a weakening lemma w.r.t. reducibility sets. It turns out we need first a quite intricate combinatory proof.

In the section, we write t to denote either a command, a term or a context. We call such a t an object and we consider objects together with a set $\mathcal{O}$ of distinguished disjoint occurrences of explicit substitutions in t. By disjoint, we mean that no explicit substitution in $\mathcal{O}$ is a strict subterm of another explicit substitution in $\mathcal{O}$.

A reduction in t may either happen in the body of some substitution listed in $\mathcal{O}$, or make some of these distinguished substitutions propagate, or otherwise happen anywhere else. We write respectively $\xrightarrow{in}$, $\xrightarrow{through}$, $\xrightarrow{out}$ for these kinds of reduction. Moreover, we maintain for these reductions a trace of the residues of the substitutions listed in $\mathcal{O}$. This means we write $t, \mathcal{O} \xrightarrow{in} t', \mathcal{O}$ for the first kind of reduction (it does not change the occurrences of substitutions in $\mathcal{O}$). We write also $t, \mathcal{O} \xrightarrow{through} t', \mathcal{O}'$ if some substitution listed in $\mathcal{O}$ is propagated one step (then $\mathcal{O}'$ is obtained by replacing the occurrence of this propagated substitution by the new set of its remaining occurrences – it may now occur 0, 1 or 2 times). For the last kind of reduction we write $t, \mathcal{O} \xrightarrow{out} t', \mathcal{O}'$ where $\mathcal{O}'$ is the updated set of residues of the substitutions listed in $\mathcal{O}$.

We now define a more general reduction $t, \mathcal{O} \xrightarrow{push} t', \mathcal{O}'$ related to the propagation of a substitution in $\mathcal{O}$. It includes all reductions in $\xrightarrow{through}$ plus the following

$$t[x \leftarrow v]\tau, \mathcal{O} \xrightarrow{push} t\tau[x \leftarrow v\tau], \mathcal{O}'$$
$$t[\alpha \leftarrow E]\tau, \mathcal{O} \xrightarrow{push} t\tau[\alpha \leftarrow E\tau], \mathcal{O}'$$

where τ is in $\mathcal{O}$ and $[x \leftarrow v]$ and $[\alpha \leftarrow E]$ are not. The set $\mathcal{O}'$ is simply obtained from $\mathcal{O}$ by taking into account the move of τ (α-conversion may be necessary to ensure there is no unexpected capture of variables).

Remark that each reduction considered until now (*in*, *through*, *out* and *push*) preserves the property to have occurrences in $\mathcal{O}$ disjoint. Then we can state some commutation lemmas (we write $\xrightarrow{in(n)}$, $\xrightarrow{through(n)}$, etc to denote n steps of the corresponding reduction).

Lemma 6 If $c, \mathcal{O} \stackrel{push(n)}{\longrightarrow} c_1, \mathcal{O}_1$ and $c, \mathcal{O} \stackrel{in}{\longrightarrow} c_2, \mathcal{O}_2$ then there is $c_3, \mathcal{O}_3$ such that $c_2, \mathcal{O}_2 \stackrel{push(n)}{\longrightarrow} c_3, \mathcal{O}_3$ and $c_1 \stackrel{*}{\longrightarrow} c_3$.

$$\begin{array}{ccc} c, \mathcal{O} & \stackrel{push(n)}{\longrightarrow} & c_1, \mathcal{O}_1 \\ {\scriptstyle in} \downarrow & & \downarrow * \\ c_2, \mathcal{O}_2 & \stackrel{push(n)}{\longrightarrow} & c_3, \mathcal{O}_3 \end{array}$$

Proof By induction on the proof of $c, \mathcal{O} \stackrel{push(n)}{\longrightarrow} c_1, \mathcal{O}_1$. Consider $c_4, \mathcal{O}_4$ such that $c, \mathcal{O} \stackrel{push}{\longrightarrow} c_4, \mathcal{O}_4 \stackrel{push(n-1)}{\longrightarrow} c_1, \mathcal{O}_1$. Since occurrences in $\mathcal{O}$ are disjoint, either the same τ in $\mathcal{O}$ is reduced or disjoint τ and τ'. If they are disjoint, then clearly the diagram commutes, i.e. there is $c_5, \mathcal{O}_5$ such that $c_4, \mathcal{O}_4 \stackrel{in}{\longrightarrow} c_5, \mathcal{O}_5$ and $c_2, \mathcal{O}_2 \stackrel{push}{\longrightarrow} c_5, \mathcal{O}_5$ and we are done by induction hypothesis.

If both reductions are located at the same occurrence, then the reduction *push* may duplicate or erase the subterm τ reduced *in*. If it is erased, then there is nothing more to prove. If it is duplicated, then we apply the induction hypothesis twice. Otherwise, we apply the induction hypothesis once.

Lemma 7 If $c, \mathcal{O} \stackrel{push(n)}{\longrightarrow} c_1, \mathcal{O}_1$ for $n > 0$ and $c, \mathcal{O} \stackrel{through}{\longrightarrow} c_2, \mathcal{O}_2$ then either we have $c_2, \mathcal{O}_2 \stackrel{push(n-1)}{\longrightarrow} c_1, \mathcal{O}_1$ or there is $c_3, \mathcal{O}_3$ such that $c_2, \mathcal{O}_2 \stackrel{push(n)}{\longrightarrow} c_3, \mathcal{O}_3$ and $c_1, \mathcal{O}_1 \stackrel{through}{\longrightarrow} c_3, \mathcal{O}_3$.

$$\begin{array}{ccc} c, \mathcal{O} & \stackrel{push(n)}{\longrightarrow} & c_1, \mathcal{O}_1 \\ {\scriptstyle through} \downarrow & & || \\ c_2, \mathcal{O}_2 & \stackrel{push(n-1)}{\longrightarrow} & c_1, \mathcal{O}_1 \end{array} \quad \text{or} \quad \begin{array}{ccc} c, \mathcal{O} & \stackrel{push(n)}{\longrightarrow} & c_1, \mathcal{O}_1 \\ {\scriptstyle through} \downarrow & & \downarrow {\scriptstyle through} \\ c_2, \mathcal{O}_2 & \stackrel{push(n)}{\longrightarrow} & c_3, \mathcal{O}_3 \end{array}$$

Proof We reason by induction on n. Since occurrences in $\mathcal{O}$ are disjoint then either both reductions $\stackrel{push}{\longrightarrow}$ and $\stackrel{through}{\longrightarrow}$ commute or they are the same. If they are the same, then we fall in the first case otherwise we apply the induction hypothesis. Finally, when $n = 0$, we fall in the second case.

Lemma 8 If $c, \mathcal{O} \stackrel{push}{\longrightarrow} c_1, \mathcal{O}_1$ and $c, \mathcal{O} \stackrel{out}{\longrightarrow} c_2, \mathcal{O}_2$ then there are natural numbers n and n' and $c_3, \mathcal{O}_3$ such that $c_2, \mathcal{O}_2 \stackrel{push(n')}{\longrightarrow} c_3, \mathcal{O}_3$ and $c_1, \mathcal{O}_1 \stackrel{through(n);out}{\longrightarrow} c_3, \mathcal{O}_3$.

$$\begin{array}{ccc} c, \mathcal{O} & \stackrel{push}{\longrightarrow} & c_1, \mathcal{O}_1 \\ {\scriptstyle out} \downarrow & & \downarrow {\scriptstyle through(n);out} \\ c_2, \mathcal{O}_2 & \stackrel{push(n')}{\longrightarrow} & c_3, \mathcal{O}_3 \end{array}$$

Proof If the *out* reduction does not overlap with any reduction in $\stackrel{push}{\longrightarrow}$, then the diagram clearly commutes. Otherwise, the *out* reduction either propagates τ' not in $\mathcal{O}$ in some subterm $t\tau'\tau$ of c or is one of the following

$$\begin{array}{l} \langle \lambda x.v | v' \cdot e \rangle \tau \longrightarrow \langle v[x{\leftarrow}v'] | e \rangle \tau \\ \langle \mu\alpha.c | E \rangle \tau \longrightarrow c[\alpha{\leftarrow}E]\tau \end{array}$$

$$\begin{array}{lcl}\langle v|\tilde{\mu}x.c\rangle\tau & \longrightarrow & c[x\leftarrow v]\tau\\ (\mu\alpha.\langle v|\alpha\rangle)\tau & \longrightarrow & v\tau\\ (\tilde{\mu}x.\langle x|e\rangle)\tau & \longrightarrow & e\tau\end{array}$$

with τ in $\mathcal{O}$ and the *push* reduction propagates τ.

Let's for instance consider the case of a β-redex. Its residue in c_1 is $\langle(\lambda x.v)\tau|(v'\cdot e)\tau\rangle$ and $\langle(\lambda x.v)\tau|(v'\cdot e)\tau\rangle \xrightarrow{through(2)} \langle\lambda x.(v\tau)|v'\tau\cdot e\tau\rangle \xrightarrow{out} \langle v\tau[x\leftarrow v'\tau]\|e\tau\rangle$ while the corresponding subterm of c_2 satisfied $\langle v[x\leftarrow v']\|e\rangle\tau \xrightarrow{push} \langle v[x\leftarrow v']\tau|e\tau\rangle \xrightarrow{push} \langle v\tau[x\leftarrow v'\tau]\|e\tau\rangle$. The other cases are similar.

Lemma 9 If $c,\mathcal{O} \xrightarrow{push(n)} c_1,\mathcal{O}_1$ and $c,\mathcal{O} \xrightarrow{out} c_2,\mathcal{O}_2$ then there is $c_3,\mathcal{O}_3$ such that $c_2,\mathcal{O}_2 \xrightarrow{push(n')} c_3,\mathcal{O}_3$ and $c_1,\mathcal{O}_1 \xrightarrow{(p)} c_3,\mathcal{O}_3$ with $p>0$.

$$\begin{array}{ccc} c,\mathcal{O} & \xrightarrow{push(n)} & c_1,\mathcal{O}_1\\ out\downarrow & & \downarrow p>0\\ c_2,\mathcal{O}_2 & \xrightarrow{push(n')} & c_3,\mathcal{O}_3\end{array}$$

Proof We reason by induction on n. If $n=0$, the result is immediate. Otherwise, assume $c,\mathcal{O} \xrightarrow{push} c_4,\mathcal{O}_4 \xrightarrow{push(n-1)} c_1,\mathcal{O}_1$. By lemma 8, there is $c_5,\mathcal{O}_5$ and $c_6,\mathcal{O}_6$ such that $c_4,\mathcal{O}_4 \xrightarrow{through(q)} c_5,\mathcal{O}_5 \xrightarrow{out} c_6,\mathcal{O}_6$ and $c_2,\mathcal{O}_2 \xrightarrow{push(q')} c_6,\mathcal{O}_6$. By applying lemma 7 several times, we finally get the existence of $c_7,\mathcal{O}_7$ such that $c_5,\mathcal{O}_5 \xrightarrow{push(n')} c_7,\mathcal{O}_7$ and $c_1,\mathcal{O}_1 \xrightarrow{through(n'')} c_7,\mathcal{O}_7$ with $n'+n''\leq n-1$. By applying induction hypothesis to $c_5,\mathcal{O}_5 \xrightarrow{push(n')} c_7,\mathcal{O}_7$, we finally get $p>0$ and $c_3,\mathcal{O}_3$ such that $c_1,\mathcal{O}_1 \xrightarrow{through(n'')} c_7,\mathcal{O}_7 \xrightarrow{(p)} c_3,\mathcal{O}_3$, that is $c_1,\mathcal{O}_1 \xrightarrow{(p')} c_3,\mathcal{O}_3$ for some $p'>0$, and $c_2,\mathcal{O}_2 \xrightarrow{push} c_3,\mathcal{O}_3$.

Lemma 10 If $c,\mathcal{O} \xrightarrow{push(n)} c_1,\mathcal{O}_1$ with $c_1\in\mathcal{SN}$ and all substitutions listed in $\mathcal{O}$ are in $\mathcal{SN}$ then $c\in\mathcal{SN}$

Proof By induction on the proof of $c_1\in\mathcal{SN}$, then on n and finally on the multiset of proofs of strong normalisation of the substitutions listed in $\mathcal{O}$.

Assume $c,\mathcal{O} \longrightarrow c_2,\mathcal{O}_2$. The reduction is either *in*, *through* or *out*.

If the reduction is *in*, then, by lemma 6, there is $c_3,\mathcal{O}_3$ such that $c_2,\mathcal{O}_2 \xrightarrow{push(n)} c_3,\mathcal{O}_3$ and $c_1,\mathcal{O}_1 \xrightarrow{(p)} c_3,\mathcal{O}_3$. If $p>0$ then we are done by induction hypothesis on $c_1\in\mathcal{SN}$. Otherwise $p=0$ and we are done by induction hypothesis on the multiset of proofs of strong normalisation of the substitutions listed in $\mathcal{O}$ (which decreases for the substitution reduced).

If the reduction is *through*, then, by lemma 7, we have either $c_2,\mathcal{O}_2 \xrightarrow{push(n-1)} c_1,\mathcal{O}_1$ or the existence of some $c_3,\mathcal{O}_3$ such that $c_2,\mathcal{O}_2 \xrightarrow{push(n)} c_3,\mathcal{O}_3$ and $c_1,\mathcal{O}_1 \xrightarrow{through} c_3,\mathcal{O}_3$. In the first case, we are done by induction hypothesis on n and in the second case by induction on the strong normalisation of c_1.

Finally, if the reduction is *out*, then, by lemma 9, there is $c_3, \mathcal{O}_3$ such that $c_1, \mathcal{O}_1 \stackrel{push(n')}{\longrightarrow} c_3, \mathcal{O}_3$ and $c_1, \mathcal{O}_1 \stackrel{(p)}{\longrightarrow} c_3, \mathcal{O}_3$ with $p > 0$. Then we are done by induction hypothesis on $c_1 \in \mathcal{SN}$.

We are now able to state a strengthening lemma and a weakening lemma w.r.t. reducibility sets.

Lemma 11 (Strengthening)

(1) $\tau \in [\![(\Gamma, x : A \vdash_{x\mathcal{V}} \Delta) \Rightarrow (\Gamma', x : A \vdash_{x\mathcal{V}'} \Delta')]\!]$ and $Dom(\tau) \neq x$ implies

$$\tau \in [\![(\Gamma \vdash_{\mathcal{V}} \Delta) \Rightarrow (\Gamma' \vdash_{\mathcal{V}'} \Delta')]\!]$$

(2) $\tau \in [\![(\Gamma, \alpha : A \vdash_{\alpha\mathcal{V}} \Delta) \Rightarrow (\Gamma', \alpha : A \vdash_{\alpha\mathcal{V}'} \Delta')]\!]$ and $Dom(\tau) \neq \alpha$ implies

$$\tau \in [\![(\Gamma \vdash_{\mathcal{V}} \Delta) \Rightarrow (\Gamma' \vdash_{\mathcal{V}'} \Delta')]\!]$$

Proof We prove (1). Either $\tau = [y \leftarrow v]$ with $\Gamma = \Gamma', y : B$, $\Delta = \Delta'$, $\mathcal{V}' = \mathcal{V}'' y \mathcal{V}'''$ and $v \in [\![\Gamma', x : A \vdash_{x\mathcal{V}'} B \mid \Delta']\!]$ such that the free variables of v are in $\mathcal{V}'''$ or $\tau = [\alpha \leftarrow E]$ with $\Delta = \Delta', \alpha : B$, $\Gamma = \Gamma'$, $\mathcal{V}' = \mathcal{V}'' \alpha \mathcal{V}'''$ and $E \in [\![\Gamma', x : A\,;\, B \vdash_{x\mathcal{V}'} \Delta']\!]$ such that the free variables of E are in $\mathcal{V}'''$. Let x_0 be a term variable in $\mathcal{V}_0$. By lemma 4, $x_0 \in [\![\Gamma'\,;\, A \vdash_{\mathcal{V}'} \Delta']\!]$ and, since $x_0 \in \mathcal{V}'$ by compatibility of $\mathcal{V}'$, the substitution $[x \leftarrow x_0]$ is in $[\![(\Gamma', x : A \vdash_{x\mathcal{V}'} \Delta') \Rightarrow (\Gamma' \vdash_{\mathcal{V}'} \Delta')]\!]$. Then, by definition of reducibility sets, $v[x \leftarrow x_0] \in [\![\Gamma' \vdash_{\mathcal{V}'} B \mid \Delta']\!]$ and $E[x \leftarrow x_0] \in [\![\Gamma'\,;\, B \vdash_{\mathcal{V}'} \Delta']\!]$ respectively. Finally, by lemma 3 and the fact that x does not occur in v and E, we get $v \in [\![\Gamma' \vdash_{\mathcal{V}'} B \mid \Delta']\!]$ and $E \in [\![\Gamma'\,;\, B \vdash_{\mathcal{V}'} \Delta']\!]$ respectively. Then $\tau \in [\![(\Gamma \vdash_{\mathcal{V}} \Delta) \Rightarrow (\Gamma' \vdash_{\mathcal{V}'} \Delta')]\!]$. Statement (2) is proved similarly.

Weakening makes the sequent grow. Then it cannot be taken into account in the definition of reducibility and must be proven apart.

Lemma 12 (Weakening) If $\tau \in [\![(\Gamma \vdash_{\mathcal{V}} \Delta) \Rightarrow (\Gamma' \vdash_{\mathcal{V}'} \Delta')]\!]$ then

(1) $x \notin \mathcal{V}$ implies $\tau \in [\![(\Gamma, x : A \vdash_{x\mathcal{V}} \Delta) \Rightarrow (\Gamma', x : A \vdash_{x\mathcal{V}'} \Delta')]\!]$

(2) $\alpha \notin \mathcal{V}$ implies $\tau \in [\![(\Gamma \vdash_{\alpha\mathcal{V}} \alpha : A, \Delta) \Rightarrow (\Gamma' \vdash_{\alpha\mathcal{V}'} \alpha : A, \Delta')]\!]$

Proof We initially reason by induction on the size of Γ and Δ. The statements amount to proving that $v \in [\![\Gamma' \vdash_{\mathcal{V}'} B \mid \Delta']\!]$ implies $v \in [\![\Gamma', x : A \vdash_{x\mathcal{V}'} B \mid \Delta']\!]$ and $v \in [\![\Gamma' \vdash_{\alpha\mathcal{V}'} B \mid \alpha : A, \Delta']\!]$ and the same for $E \in [\![\Gamma' \mid B \vdash_{\mathcal{V}'} \Delta']\!]$.

Consider first the latter. If $E \downarrow_h \beta$ and $E \in \mathcal{SN}$ then this directly implies $E \in [\![\Gamma', x : A \mid B \vdash_{x\mathcal{V}'} \Delta']\!]$ and $E \in [\![\Gamma' \mid B \vdash_{\alpha\mathcal{V}'} \alpha : A, \Delta']\!]$.

If $E \downarrow_h v \cdot e$ and $E \in \mathcal{SN}$ then we need to show the property for v (that we already have to) but also for e (what we need to do too). However the two cases are similar and we focus on v. We even just focus on the case $x \notin \mathcal{V}$, since the case $\alpha \notin \mathcal{V}$ works similarly. Then, we have to prove $v \in [\![\Gamma', x : A \vdash_{x\mathcal{V}'} B \mid \Delta']\!]$ and we do it by induction on Γ' and Δ'.

To prove the latter, we need to prove $v\tau \in [\![\Gamma'' \vdash_{\mathcal{V}''} B \mid \Delta'']\!]$ for all unary substitution $\tau \in [\![(\Gamma', x : A \vdash_{x\mathcal{V}'} \Delta') \Rightarrow (\Gamma'' \vdash_{\mathcal{V}''} \Delta'')]\!]$ and $\langle v | E \rangle \in [\![\Gamma', x : A \vdash_{x\mathcal{V}'} \Delta']\!]$ for all $E \in [\![\Gamma', x : A\,;\, B \vdash_{x\mathcal{V}} \Delta']\!]$.

If $Dom(\tau) \neq x$ with $\Gamma'' = \Gamma''', x : A$ then $\tau \in [\![(\Gamma' \vdash_{\mathcal{V}'} \Delta') \Rightarrow (\Gamma''' \vdash_{\mathcal{V}''} \Delta'')]\!]$ by lemma 11. Then we get $v\tau \in [\![\Gamma''' \vdash_{\mathcal{V}''} B \mid \Delta'']\!]$ by reducibility of v and $v \in \tau[\![\Gamma''', x : A \vdash_{x\mathcal{V}''} B \mid \Delta'']\!]$ by induction hypothesis on Γ' and Δ'.

If $\tau = [x \leftarrow v']$ with $v' \in [\![\Gamma' \vdash_{\mathcal{V}'} A \mid \Delta']\!]$ then $v' \in \mathcal{SN}$ by lemma 4. By reducibility of v, we have also $\langle v\sigma | E \rangle \sigma' \in \mathcal{SN}$ for all $\sigma \in [\![(\Gamma' \vdash_{\mathcal{V}'} \Delta') \Rightarrow (\Gamma'' \vdash_{\mathcal{V}''} \Delta'')]\!]$, $E \in [\![\Gamma'' \,;\, B \vdash_{\mathcal{V}''} \Delta'']\!]$ and $\sigma' \in [\![(\Gamma'' \vdash_{\mathcal{V}''} \Delta'') \Rightarrow (\vdash_{\mathcal{V}'''})]\!]$. Since $\langle v\tau\sigma | E \rangle \sigma' \xrightarrow{push} \langle v\sigma | E \rangle \sigma'$ regarding τ, we obtain from lemma 10 that $\langle v\tau\sigma | E \rangle \sigma' \in \mathcal{SN}$, which implies that $v\tau \in [\![\Gamma'', x : A \vdash_{x\mathcal{V}''} B \mid \Delta'']\!]$.

We now prove $\langle v | E \rangle \in [\![\Gamma', x : A \vdash_{x\mathcal{V}'} \Delta']\!]$ for $E \in [\![\Gamma', x : A \,;\, B \vdash_{x\mathcal{V}'} \Delta']\!]$. To this aim, we consider $\sigma \in [\![(\Gamma', x : A \vdash_{x\mathcal{V}'} \Delta') \Rightarrow (\vdash_{\mathcal{V}'''})]\!]$ and prove $\langle v | E \rangle \sigma \in \mathcal{SN}$. Since x is declared in $\Gamma', x : A$, σ can be decomposed into some substitution $\sigma'[x \leftarrow v']\sigma''$ with $\sigma' \in [\![(\Gamma', x : A \vdash_{x\mathcal{V}'} \Delta') \Rightarrow (\Gamma'', x : A \vdash_{\mathcal{V}''} \Delta'')]\!]$, $v' \in [\![\Gamma'' \vdash_{\mathcal{V}''} A \mid \Delta'']\!]$ and $\sigma'' \in [\![(\Gamma'' \vdash_{\mathcal{V}''} \Delta'') \Rightarrow (\vdash_{\mathcal{V}'''})]\!]$. We have $v' \in \mathcal{SN}$ by lemma 4 and then, by lemma 11, $\sigma' \in [\![(\Gamma' \vdash_{\mathcal{V}'} \Delta') \Rightarrow (\Gamma'' \vdash_{\mathcal{V}''} \Delta'')]\!]$. By iterated application of the initial induction hypothesis, we obtain $[x \leftarrow v'] \in [\![(\Gamma', x : A \vdash_{x\mathcal{V}'} \Delta') \Rightarrow (\Gamma' \vdash_{\mathcal{V}'} \Delta')]\!]$. By reducibility of E we get $E[x \leftarrow v'] \in [\![\Gamma' \,;\, B \vdash_{\mathcal{V}'} \Delta']\!]$. By reducibility of v, we get $\langle v | E[x \leftarrow v'] \rangle \in [\![\Gamma' \vdash_{\mathcal{V}'} \Delta']\!]$, and finally, $\langle v | E[x \leftarrow v'] \rangle \sigma' \sigma'' \in \mathcal{SN}$. Since $\langle v | E \rangle \sigma \xrightarrow{push} \langle v | E[x \leftarrow v'] \rangle \sigma' \sigma''$, we conclude by lemma 10 that $\langle v | E \rangle \sigma \in \mathcal{SN}$.

The case e works similarly.

6 Strong Normalisation

Lemma 13 (Abstraction) $v \in [\![\Gamma, x : A \vdash_{x\mathcal{V}} B \mid \Delta]\!]$ implies $\lambda x.v \in [\![\Gamma \vdash_{\mathcal{V}} A \rightarrow B \mid \Delta]\!]$

Proof We strengthen the result and show by induction on the length of Γ' and Δ' that $\lambda x.(v\sigma) \in [\![\Gamma' \vdash_{\mathcal{V}'} A \rightarrow B \mid \Delta']\!]$ for all $\sigma \in [\![(\Gamma, x : A \vdash_{x\mathcal{V}} \Delta) \Rightarrow (\Gamma', x : A \vdash_{x\mathcal{V}'} \Delta')]\!]$. Otherwise said, we should prove that for all $\sigma' \in [\![(\Gamma' \vdash_{\mathcal{V}'} \Delta') \Rightarrow (\Gamma'' \vdash_{\mathcal{V}''} \Delta'')]\!], E \in [\![\Gamma'' \,;\, A \rightarrow B \vdash_{\mathcal{V}''} \Delta'']\!], \sigma'' \in [\![(\Gamma'' \vdash_{\mathcal{V}''} \Delta'') \Rightarrow (\vdash_{\mathcal{V}'''})]\!]$, we have $\langle (\lambda x.(v\sigma))\sigma' | E \rangle \sigma'' \in \mathcal{SN}$.

We first reason by induction on the length of Γ'' and Δ''. Then, since we have $v\sigma, E, \sigma', \sigma'' \in \mathcal{SN}$ by lemma 4, we reason by induction on the proofs of strong normalisability for $v\sigma$, E, σ' and σ''.

- If the reduction occurs in $v\sigma$, E, σ', or σ'' then its reduct is reducible by lemma 3 and we are done by induction on SN.
- If $\sigma' = [\ ]$, $E = v' \cdot e$ and $\langle \lambda x.(v\sigma) | v' \cdot e \rangle \sigma'' \longrightarrow \langle v\sigma[x \leftarrow v'] | e \rangle \sigma''$ then $v' \in [\![\Gamma' \vdash_{\mathcal{V}'} A \mid \Delta']\!]$ and $e \in [\![\Gamma' \mid B \vdash_{\mathcal{V}'} \Delta']\!]$. We deduce $v\sigma[x \leftarrow v'] \in [\![\Gamma' \vdash_{\mathcal{V}'} B \mid \Delta']\!]$ by definition of $v \in [\![\Gamma, x : A \vdash_{x\mathcal{V}} B \mid \Delta]\!]$ and $\langle v\sigma[x \leftarrow v'] | e \rangle \sigma'' \in \mathcal{SN}$ by definition of $e \in [\![\Gamma' \mid B \vdash_{\mathcal{V}'} \Delta']\!]$.
- If $\sigma' = \tau\sigma'''$ with $\tau \in [\![(\Gamma' \vdash_{\mathcal{V}'} \Delta') \Rightarrow (\Gamma''' \vdash_{\mathcal{V}'''} \Delta''')]\!]$ and $\langle (\lambda x.(v\sigma))\tau\sigma''' | E \rangle \sigma'' \longrightarrow \langle (\lambda x.(v\sigma\tau))\sigma''' | E \rangle \sigma''$ then we get $\tau \in [\![(\Gamma', x : A \vdash_{x\mathcal{V}'} \Delta') \Rightarrow (\Gamma''', x : A \vdash_{x\mathcal{V}'''} \Delta''')]\!]$ by lemma 12 (the definition of $v \in [\![\Gamma, x : A \vdash_{x\mathcal{V}} B \mid \Delta]\!]$ ensures $x \notin \mathcal{V}'$). From this we get $\sigma\tau \in [\![(\Gamma, x : A \vdash_{x\mathcal{V}} \Delta) \Rightarrow (\Gamma''', x : A \vdash_{x\mathcal{V}'''} \Delta''')]\!]$ and then $\lambda x.(v\sigma\tau) \in [\![\Gamma''' \vdash_{\mathcal{V}'''} A \rightarrow B \mid \Delta''']\!]$ by induction hypothesis on the length of Γ' and Δ'. Finally, $\langle (\lambda x.(v\sigma\tau))\sigma''' | E \rangle \sigma'' \in \mathcal{SN}$ by definition of $[\![\Gamma''' \vdash_{\mathcal{V}'''} A \rightarrow B \mid \Delta''']\!]$.

- If $\sigma'' = \tau\sigma'''$ with $\tau \in [\![(\Gamma''\vdash_{\mathcal{V}''}\Delta'') \Rightarrow (\Gamma'''\vdash_{\mathcal{V}'''}\Delta''')]\!]$ and $\langle(\lambda x.(v\sigma))\sigma'|E\rangle\tau\sigma''' \longrightarrow \langle(\lambda x.(v\sigma))\sigma'\tau|E\tau\rangle\sigma'''$ then $\sigma'\tau \in [\![(\Gamma'\vdash_{\mathcal{V}'}\Delta') \Rightarrow (\Gamma'''\vdash_{\mathcal{V}'''}\Delta''')]\!]$. Then, by definition of $E \in [\![\Gamma'' \,;\, A{\to}B\vdash_{\mathcal{V}''}\Delta'']\!]$, we get $E\tau \in [\![\Gamma''' \,;\, A{\to}B\vdash_{\mathcal{V}'''}\Delta''']\!]$ and we are done by induction hypothesis on the length of Γ'' and Δ''.

Lemma 14 Let $c = \langle v|e\rangle$ such that $v \in [\![\Gamma\vdash_{\mathcal{V}}B \mid \Delta]\!]$ and $e \in [\![\Gamma \mid B\vdash_{\mathcal{V}}\Delta]\!]$. Then we have $\mu\alpha.(c\sigma) \in [\![\Gamma'\vdash_{\mathcal{V}'}A \mid \Delta']\!]$ for all $\sigma \in [\![(\Gamma\vdash_{\mathcal{V}}\Delta) \Rightarrow (\Gamma'\vdash_{\alpha\mathcal{V}'}\alpha : A, \Delta')]\!]$.

Proof Let $c = \langle v|e\rangle$. By definition of $e \in [\![\Gamma \mid B\vdash_{\mathcal{V}}\Delta]\!]$ we have $c \in [\![\Gamma\vdash_{\mathcal{V}}\Delta]\!]$. We prove on the length of Γ' and Δ' that for all $\sigma \in [\![(\Gamma\vdash_{\mathcal{V}}\Delta) \Rightarrow (\Gamma'\vdash_{\alpha\mathcal{V}'}\alpha : A, \Delta')]\!]$ we have $\mu\alpha.(c\sigma) \in [\![\Gamma\vdash_{\mathcal{V}}A \mid \Delta]\!]$. Actually, we prove that for all $\sigma' \in [\![(\Gamma'\vdash_{\mathcal{V}'}\Delta') \Rightarrow (\Gamma''\vdash_{\mathcal{V}''}\Delta'')]\!]$, $E \in [\![\Gamma'' \,;\, A\vdash_{\mathcal{V}''}\Delta'']\!]$, $\sigma'' \in [\![(\Gamma''\vdash_{\mathcal{V}''}\Delta'') \Rightarrow (\vdash_{\mathcal{V}'''})]\!]$ we have $\langle(\mu\alpha.(c\sigma))\sigma'|E\rangle\sigma'' \in \mathcal{SN}$.

As for abstraction we first reason by induction on the length of Γ'' and Δ''. Then, since we have $c\sigma, E, \sigma', \sigma'' \in \mathcal{SN}$ by lemma 4, we reason by induction on the proofs of strong normalisability for $c\sigma$, E, σ' and σ''.

- If the reduction occurs in $c\sigma$, E, σ', or σ'' then its reduct is reducible by lemma 3 and we are done by the induction hypothesis for SN.

- If $\sigma' = [\,]$ and $\langle\mu\alpha.(c\sigma)|E\rangle\sigma'' \longrightarrow c\sigma[\alpha \leftarrow E]\sigma''$ then the unary substitution $[\alpha{\leftarrow}E] \in [\![(\Gamma\vdash_{\alpha\mathcal{V}}\alpha : A, \Delta) \Rightarrow (\Gamma\vdash_{\mathcal{V}}\Delta)]\!]$ and $c\sigma[\alpha \leftarrow E']\sigma'' \in \mathcal{SN}$ by definition of $c \in [\![\Gamma\vdash_{\alpha\mathcal{V}}\alpha : A, \Delta]\!]$.

- If $\sigma' = \tau\sigma'''$ with $\tau \in [\![(\Gamma'\vdash_{\mathcal{V}'}\Delta') \Rightarrow (\Gamma'''\vdash_{\mathcal{V}'''}\Delta''')]\!]$ and $\langle(\mu\alpha.(c\sigma))\tau\sigma'''|E\rangle\sigma'' \longrightarrow \langle(\mu\alpha.(c\sigma\tau))\sigma'''|E\rangle\sigma''$ then we get $\tau \in [\![(\Gamma'\vdash_{\alpha\mathcal{V}'}\alpha : A, \Delta') \Rightarrow (\Gamma'''\vdash_{\alpha\mathcal{V}'''}\alpha : A, \Delta''')]\!]$ by lemma 12. From this, we get $\sigma\tau \in [\![(\Gamma\vdash_{\mathcal{V}}\Delta) \Rightarrow (\Gamma'''\vdash_{\alpha\mathcal{V}'''}\alpha : A''', \Delta)]\!]$, then $\mu\alpha.(c\sigma\tau) \in [\![\Gamma'''\vdash_{\mathcal{V}'''}A \mid \Delta''']\!]$ by induction hypothesis on the length of Γ' and Δ'. Finally, $\langle(\mu\alpha.(c\sigma\tau))\sigma'''|E\rangle\sigma'' \in \mathcal{SN}$ by definition of $[\![\Gamma'''\vdash_{\mathcal{V}'''}A \mid \Delta''']\!]$.

- If $\sigma'' = \tau\sigma'''$ and $\langle(\mu\alpha.(c\sigma))\sigma'|E\rangle\tau\sigma''' \longrightarrow \langle(\mu\alpha.(c\sigma))\sigma'\tau|E\tau\rangle\sigma'''$ then $\sigma'\tau \in [\![(\Gamma'\vdash_{\mathcal{V}'}\Delta') \Rightarrow (\Gamma'''\vdash_{\mathcal{V}'''}\Delta''')]\!]$ and $E\tau \in [\![\Gamma''' \,;\, A\vdash_{\mathcal{V}'''}\Delta''']\!]$ by definition of $E \in [\![\Gamma'' \,;\, A\vdash_{\mathcal{V}''}\Delta'']\!]$ and we are done by induction hypothesis on the length of Γ'' and Δ''.

- If $e = \alpha$ with $\alpha \notin v$, $\sigma = [\,]$, $v \in [\![\Gamma'\vdash_{\alpha\mathcal{V}'}B \mid \alpha : A, \Delta']\!]$ and $\langle(\mu\alpha.\langle v|\alpha\rangle)\sigma'|E\rangle\sigma'' \longrightarrow \langle v\sigma'|E\rangle\sigma''$ then $v \in [\![\Gamma'\vdash_{\mathcal{V}'}B \mid \Delta']\!]$. This is by lemma 3 and the fact that $v[\alpha{\leftarrow}\beta] \xrightarrow{*} v$ and $v[\alpha{\leftarrow}\beta] \in [\![\Gamma\vdash_{\mathcal{V}}B \mid \Delta]\!]$ for any variable β obtained in $[\![\Gamma \,;\, B\vdash_{\mathcal{V}}\Delta]\!]$ by lemma 4. We then deduce $\langle v\sigma'|E\rangle\sigma'' \in \mathcal{SN}$ by definition of $v \in [\![\Gamma'\vdash_{\mathcal{V}'}B \mid \Delta']\!]$.

Lemma 15 Let $c = \langle v|e\rangle$ such that $v \in [\![\Gamma\vdash_{\mathcal{V}}B \mid \Delta]\!]$ and $e \in [\![\Gamma \mid B\vdash_{\mathcal{V}}\Delta]\!]$. Then, we have $\tilde{\mu}x.(c\sigma) \in [\![\Gamma'\vdash_{\mathcal{V}'}A \mid \Delta']\!]$ for all $\sigma \in [\![(\Gamma\vdash_{\mathcal{V}}\Delta) \Rightarrow (\Gamma', x : A\vdash_{x\mathcal{V}'}\Delta')]\!]$.

Proof By definition of $e \in [\![\Gamma \mid B\vdash_{\mathcal{V}}\Delta]\!]$ we have $c\sigma \in [\![\Gamma\vdash_{\mathcal{V}}\Delta]\!]$. The proof is similar to the proof of lemma 14 except that we prove $\langle v'|(\tilde{\mu}x.(c\sigma))\sigma'\rangle\sigma'' \in \mathcal{SN}$ for all $\sigma \in [\![(\Gamma\vdash_{\mathcal{V}}\Delta) \Rightarrow (\Gamma', x : A\vdash_{x\mathcal{V}'}\Delta')]\!]$, $\sigma' \in [\![(\Gamma'\vdash_{\mathcal{V}'}\Delta') \Rightarrow (\Gamma''\vdash_{\mathcal{V}''}\Delta'')]\!]$, $v' \in [\![\Gamma''\vdash_{\mathcal{V}'}A \mid \Delta'']\!]$ and $\sigma'' \in [\![(\Gamma''\vdash_{\mathcal{V}''}\Delta'') \Rightarrow (\vdash_{\mathcal{V}'''})]\!]$.

Lemma 16 (Adequacy)

$\mathcal{V}$ compatible with $\Gamma \vdash v : A \,|\, \Delta$ implies $v \in [\![\Gamma \vdash_{\mathcal{V}} A \,|\, \Delta]\!]$,
$\mathcal{V}$ compatible with $\Gamma\,;\, E : A \vdash \Delta$ implies $E \in [\![\Gamma\,;\, A \vdash_{\mathcal{V}} \Delta]\!]$,
$\mathcal{V}$ compatible with $\Gamma \,|\, e : A \vdash \Delta$ implies $e \in [\![\Gamma \,|\, A \vdash_{\mathcal{V}} \Delta]\!]$,
$\mathcal{V}$ compatible with $c : (\Gamma \vdash \Delta)$ implies $c \in [\![\Gamma \vdash_{\mathcal{V}} \Delta]\!]$,
$(\mathcal{V}, \mathcal{V}')$ compatible with $\tau : (\Gamma \vdash \Delta) \Rightarrow (\Gamma' \vdash \Delta')$ implies $\tau \in [\![(\Gamma \vdash_{\mathcal{V}} \Delta) \Rightarrow (\Gamma' \vdash_{\mathcal{V}'} \Delta')]\!]$.

Proof Let $\mathcal{V}$ be compatible with Γ and Δ. We reason by mutual induction on the derivations of c, v, e and E. In all cases (but the cases x, α, $v \cdot e$ and τ), we use the induction hypothesis with $x\mathcal{V}$ or $\alpha\mathcal{V}$ where x or α is the only variable occurring in the upper sequent and not in the lower one.

- Terms

  For x, $\lambda x.v$ and $\mu\alpha.c$, we use lemma 4, lemma 13 and lemma 14 respectively. Finally, if $v = v\tau$ with $\Gamma' \vdash v : A \,|\, \Delta'$ and $\tau : (\Gamma' \vdash \Delta') \Rightarrow (\Gamma \vdash \Delta)$, we get the result by the induction hypotheses $v \in [\![\Gamma' \vdash_{\mathcal{V}'} A \,|\, \Delta']\!]$ and $\tau \in [\![(\Gamma' \vdash_{\mathcal{V}'} \Delta') \Rightarrow (\Gamma \vdash_{\mathcal{V}} \Delta)]\!]$ which directly implies $v\tau \in [\![\Gamma \vdash_{\mathcal{V}} A \,|\, \Delta]\!]$.

- Applicative contexts.

  Consider the case α with $\Gamma\,;\, \alpha : A \vdash \alpha : A, \Delta$. Then we immediately have $\alpha \in [\![\Gamma\,;\, A \vdash_{\mathcal{V}} \alpha : A, \Delta]\!]$ by definition of $[\![\Gamma\,;\, A \vdash_{\mathcal{V}} \alpha : A, \Delta]\!]$.

  Consider the case $v \cdot e$ with $\Gamma \vdash v : B \,|\, \Delta$ and $\Gamma \,|\, e : C \vdash \Delta$. By induction hypothesis $v \in [\![\Gamma \vdash_{\mathcal{V}} B \,|\, \Delta]\!]$ and $e \in [\![\Gamma \,|\, C \vdash_{\mathcal{V}} \Delta]\!]$. By lemma 4, we get $v \cdot e \in \mathcal{SN}$ and since $v \cdot e \downarrow_h v \cdot e$, we are done.

  As for the case $E\tau$, it is handled by lemma 5.

- Arbitrary contexts

  Assume first $e = E$ with $\Gamma\,;\, E : A \vdash \Delta$ and therefore $E \in [\![\Gamma\,;\, A \vdash_{\mathcal{V}} \Delta]\!]$. We need to show $\forall \sigma \in [\![(\Gamma \vdash_{\mathcal{V}} \Delta) \Rightarrow (\Gamma' \vdash_{\mathcal{V}'} \Delta')]\!], v \in [\![\Gamma' \vdash_{\mathcal{V}'} A \,|\, \Delta']\!], \sigma' \in [\![(\Gamma \vdash_{\mathcal{V}} \Delta) \Rightarrow (\vdash_{\mathcal{V}''})]\!]$ and $\langle v | E\sigma \rangle \sigma' \in \mathcal{SN}$. By lemma 5 we obtain $E\sigma \in [\![\Gamma'\,;\, A \vdash_{\mathcal{V}'} \Delta']\!]$. By definition of $v \in [\![\Gamma' \vdash_{\mathcal{V}'} A \,|\, \Delta']\!]$ we then directly get $\langle v | E\sigma \rangle \sigma' \in \mathcal{SN}$.

  If $e = \tilde{\mu} x.c$ then we use lemma 15.

  Assume $e = e'\tau$ with $\Gamma' \,|\, e' : A \vdash \Delta'$ and $\tau : (\Gamma' \vdash \Delta') \Rightarrow (\Gamma \vdash \Delta)$. By induction hypothesis $e' \in [\![\Gamma' \,|\, A \vdash_{\mathcal{V}'} \Delta']\!]$ and $\tau \in [\![(\Gamma' \vdash_{\mathcal{V}'} \Delta') \Rightarrow (\Gamma \vdash_{\mathcal{V}} \Delta)]\!]$which implies $e'\tau \in [\![\Gamma \,|\, A \vdash_{\mathcal{V}} \Delta]\!]$.

- Commands

  Assume $c = \langle v | e \rangle$ with $\Gamma \vdash v : A \,|\, \Delta$ and $\Gamma \,|\, e : A \vdash \Delta$ and therefore $v \in [\![\Gamma \vdash_{\mathcal{V}} A \,|\, \Delta]\!]$ and $e \in [\![\Gamma \,|\, A \vdash_{\mathcal{V}} \Delta]\!]$. By definition of $e \in [\![\Gamma \,|\, A \vdash_{\mathcal{V}} \Delta]\!]$, we directly have $c \in [\![\Gamma \vdash_{\mathcal{V}} \Delta]\!]$.

  Assume $c = c'\tau$ with $c' : (\Gamma' \vdash \Delta')$ and $\tau : (\Gamma' \vdash \Delta') \Rightarrow (\Gamma \vdash \Delta)$. By induction hypothesis we have $c' \in [\![\Gamma' \vdash_{\mathcal{V}'} \Delta']\!]$ and $\tau \in [\![(\Gamma' \vdash_{\mathcal{V}'} \Delta') \Rightarrow (\Gamma \vdash_{\mathcal{V}} \Delta)]\!]$ which directly implies $c'\tau \in [\![\Gamma \vdash_{\mathcal{V}} \Delta]\!]$.

- Unary substitutions

  This directly comes by induction from the definition of $[\![(\Gamma \vdash_{\mathcal{V}} \Delta) \Rightarrow (\Gamma' \vdash_{\mathcal{V}'} \Delta')]\!]$.

Lemma 16 and lemma 4 allow us to state the strong normalisation property.

Theorem 17 Simply-typed $\overline{\lambda}\mu\tilde{\mu}$-calculus satisfies the strong normalisation property for the reduction considered.

7 Perspectives

The $\overline{\lambda}\mu\tilde{\mu}$-calculus can be extended to the difference type "–" (the dual of implication) in order to get a fully dual system (see [7]). This is achieved by adding the dual of $\lambda x.v$, let's write it $\beta\lambda.e$, in the categories of applicative contexts and the dual of $v \cdot e$, let's write it $e \cdot v$, in the category of values. Proving strong normalisation for this extended calculus, would imply, by duality of CBN and CBV in $\overline{\lambda}\mu\tilde{\mu}$-calculus, strong normalisation for the simply-typed CBV $\overline{\lambda}\mu\tilde{\mu}$-calculus. As a corollary we would get strong normalisation for a formulation of call-by-value simply-typed $\lambda\mu$-calculus.

The proof pattern used for the strong normalisation of Barbanera & Berardi's symmetrical λ-calculus [1] can without any doubt be adapted for $\overline{\lambda}\mu\tilde{\mu}$-calculus as it has been by Urban & Bierman to get strong non-deterministic (big step) cut-elimination of LK [30]. Then we would get strong normalisation for $\overline{\lambda}\mu\tilde{\mu}$-calculus

Simply-typed call-by-name and call-by-value $\overline{\lambda}\mu\tilde{\mu}$-calculi are strongly related with the sequent calculi LK^t and LK^q defined in Danos *et al* [8]. We believe our proof can be adapted to get a strong cut-elimination result for LK for a fine-grained set of cut-elimination rules (as in Dragalin [12] but with extra permutation of cuts allowing simulation of β-reduction). This would provide a direct proof for what can be proved combining SN for Proof Nets [14], embeddings of LK^t and LK^q in Proof Nets [8], PSN for the fine-grained reduction [4], and SN as described in the last paragraph of introduction.

Permutative conversions (also known as commutative cuts) for disjunction or *let* operator are "natural" in $\overline{\lambda}$-calculus with sums (see [17]). Then we can expect that "the canonical extension" of the proof to sums will provide strong normalisation for natural deduction with sums.

We have used variables to manage binders in $\overline{\lambda}\mu\tilde{\mu}$-calculus. The proof could be turned into a strong normalisation proof for a formulation with de Bruijn indices instead[8], as it was the case in [17] to get a proof of strong normalisation for the variant of $\overline{\lambda}\mu$-calculus considered there[9].

Acknowledgements

I'm very grateful to an anonymous referee for his careful reading and for pointing out that the simulation of Parigot's $\lambda\mu$-calculus with explicit substitution into $\overline{\lambda}\mu\tilde{\mu}$-calculus described in [18] is wrong.

[8]The formulation of $\overline{\lambda}\mu\tilde{\mu}$-calculus with de Bruijn indices was a good guide to understand possible variable capture in weakening and strengthening lemmas!

[9]The notion of reduction considered in this calculus is more restricted than ours: the (equivalent form of the) rule $\langle v|e\rangle\tau \longrightarrow \langle v\tau|e\tau\rangle$ is not considered there and full β-reduction cannot be simulated.

References

[1] F. Barbanera and S. Berardi. A symmetric lambda-calculus for 'classical' proof extraction, In *Proceedings of 2nd International Symposium on Theoretical Aspects of Computer Software (TACS'94)*, volume 789 of LNCS, 495–515. Springer Verlag, 1994.

[2] R. Bloo. *Preservation of Termination for Explicit Substitution.* PhD thesis, Eindhoven University of Technology, 1997.

[3] R. Bloo and K. H. Rose. Preservation of Strong Normalisation in Named Lambda-Calculi with Explicit Substitution and Garbage Collection. *Computer Science in the Netherlands (CSN)*, 62–72, 1995.

[4] R. Bloo and K. H. Rose. Combinatory reduction systems with explicit substitution that preserve strong normalization. In *Rewriting Techniques and Applications, 7th International Conference (RTA'96)*, volume 1103 of LNCS, 169–183. Springer Verlag, 1996.

[5] E. Bonelli. Perpetuality in a named lambda calculus with explicit substitutions. to appear in *Mathematical Structures in Computer Science*, 2000.

[6] P.-L. Curien. Categorical Combinators. *Information and Control* 69:188–254, 1986.

[7] P.-L. Curien and H. Herbelin. The duality of computation. In *Proceedings of the 5^{th} ACM SIGPLAN International Conference on Functional Programming (ICFP'00)*, 233–243. ACM Press, 2000.

[8] V. Danos, J.-B. Joinet, H. Schellinx. A New Deconstructive Logic: Linear Logic. *Journal of Symbolic Logic* 62(3):755–807, 1997.

[9] R. David and B. Guillaume. The λ_l-calculus. In *Proceedings of the 2^{nd} International Workshop on Explicit Substitutions, Theory and Applications to Programs and Proofs (WESTAPP'99)*, 1999.

[10] R. Di Cosmo and D. Kesner. Strong Normalization of Explicit Substitutions via Cut Elimination in Proof Nets. In *Proceedings of 12^{th} Annual IEEE Symposium on Logic in Computer Science (LICS'97)*, 35–46. IEEE Computer Society Press, 1997.

[11] R. Di Cosmo, D. Kesner, and E. Polonovski. Proof Nets and Explicit Substitutions. *Foundations of Software Science and Computation Structures (FOSSACS)*, 2000.

[12] A. G. Dragalin. *Mathematical Intuitionism: Introduction to Proof Theory.* Translations of mathematical monographs 67, Providence, R.I., AMS, 1988 (original edition in Russian from 1979).

[13] G. Gentzen. Investigations into logical deduction (1935). E.g. in *Gentzen collected works*, 68ff, Ed M. E. Szabo. North Holland, 1969.

[14] J.-Y. Girard. Linear Logic. *Theoretical Computer Science*, 50, 1987.

[15] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proof and Types.* Cambridge University Press, 1989.

[16] H. Herbelin. A lambda-calculus structure isomorphic to sequent calculus structure. In *Proceedings of Computer Science Logic (CSL'94)*, volume 933 of LNCS, 61–75. Springer Verlag, 1994.

[17] H. Herbelin. *Séquents qu'on calcule.* Thèse de Doctorat, Université Paris 7, 1995.

[18] H. Herbelin. Explicit Substitutions and Reducibility. In *Proceedings of the 3^{nd} International Workshop on Explicit Substitutions, Theory and Applications to Programs and Proofs (WESTAPP'00)*, 2000.

[19] F. Kamareddine and A. Ríos. A λ-calculus à la de Bruijn with explicit substitutions. In *Proceedings of the International Symposium on Programming Language Implementation and Logic Programming (PLILP'95)*, volume 982 of LNCS. Springer Verlag, 1995.

[20] F. Kamareddine and A. Ríos. The λs-calculus, its typed and its extended versions. Technical Report TR-95-13, Computer Science department, University of Glasgow, 1995.

[21] P. Lescanne. From $\lambda\sigma$ to $\lambda\upsilon$: a journey through calculi of explicit substitutions. In *Proceedings of the 21st Annual Symposium on Principles of Programming Languages (POPL'94)*, 60–109. ACM Press, 1994.

[22] P. Lescanne and J. Rouyer-Degli. Explicit Substitutions with de Bruijn Levels In *Proceedings of the 6th International Conference on Rewriting Techniques and Applications (RTA'95)*, volume 914 of LNCS, 294–308. Springer Verlag, 1995.

[23] R. Lins. A new formula for the execution of categorical combinators. In *Proceedings of the 8th International Conference on Automated Deduction (CADE'86)*, volume 230 of LNCS, 89–98. Springer Verlag, 1986.

[24] M. Parigot. $\lambda\mu$-calculus: An algorithmic interpretation of classical natural deduction. In *Proceedings of the International Conference on Logic Programming and Automated Reasoning (LPAR'92)*, volume 624 of LNCS. Springer Verlag, 1992.

[25] M. Parigot. Strong normalization for second order classical natural deduction. In *Proceedings of 8^{th} Annual IEEE Symposium on Logic in Computer Science (LICS'93)*, 39–46. IEEE Computer Society Press, 1993.

[26] G. D. Plotkin. Call-by-name, call-by-value and the lambda-calculus. *Theoretical Computer Science* 1:125–159, 1975.

[27] D. Prawitz. *Natural Deduction, a Proof-Theoretical Study.* Almqvist & Wiksell, 1965.

[28] E. Ritter. Characterising Explicit Substitutions which Preserve Termination. In *Proceedings of Typed Lambda Calculus and Applications (TLCA'99)*, volume 1581 of LNCS. Springer Verlag, 1999.

[29] W. W. Tait. Intensional interpretation of functionals of finite types I. *Journal of Symbolic Logic* 32:198–212, 1967.

[30] C. Urban and G. Bierman. Strong normalisation of cut-elimination in classical logic. In *Proceedings of Typed Lambda Calculus and Applications (TLCA'99)*, volume 1581 of LNCS. Springer Verlag, 1999.