Superdeduction in Lambda-bar-mu-mu-tilde

Clement Houtmann

Superdeduction in $\overline{\lambda}\mu\tilde{\mu}$

Clément Houtmann

Université Henri Poincaré Nancy 1 & LORIA*,
Campus Scientifique, BP 239
54506 Vandoeuvre-lès-Nancy Cedex, France
Clement.Houtmann@loria.fr

Abstract. Superdeduction is a method specially designed to ease the use of first-order theories in predicate logic. The theory is used to enrich the deduction system with new deduction rules in a systematic, correct and complete way. A proof-term language and a cut-elimination reduction already exist for superdeduction, both based on Christian Urban's work on classical sequent calculus. However Christian Urban's calculus is not directly related to the Curry-Howard correspondence contrarily to the $\overline{\lambda}\mu\tilde{\mu}$-calculus which relates straightaway to the λ-calculus. This short paper is my first step towards a further exploration of the computational content of superdeduction proofs, for I extend the $\overline{\lambda}\mu\tilde{\mu}$-calculus in order to obtain a proofterm langage together with a cut-elimination reduction for superdeduction. I also prove strong normalisation for this extension of the $\overline{\lambda}\mu\tilde{\mu}$-calculus.

Keywords: Classical sequent calculus, Superdeduction, $\overline{\lambda}\mu\tilde{\mu}$-calculus

1 Introduction

Superdeduction is an extension of predicate logic designed to ease the use of first-order theories by enriching a deduction system with new deduction rules computed from the theory. Once the theory is presented as a rewrite system, the translation into a set of custom (super)deduction rules is fully systematic. Superdeduction systems [1] are usually constructed on top of the classical sequent calculus LK which is described in Figure 1. New deduction rules are computed from a theory presented as a set of *proposition rewrite rules*, i.e. rewrite rules of the form $P \to \varphi$ where P is some atomic formula. Such rewrite rules actually stand for equivalences $\forall \overline{x}.(P \Leftrightarrow \varphi)$ where $\overline{x}$ represents the free variables of P. The computation of custom inferences for the proposition rewrite rule $P \to \varphi$ goes as follows. On the right, it decomposes (bottom-up) the sequent $\vdash \varphi$ using $\mathrm{LK}\backslash\{\mathrm{Cut}, \mathrm{ContrR}, \mathrm{ContrL}\}$ (indeterministically) until it reaches a sequence of atomic sequents[1] $(\Gamma_i \vdash \Delta_i)_{1 \leqslant i \leqslant n}$ and side condition C therefore yielding the

* UMR 7503 CNRS-INPL-INRIA-Nancy2-UHP
[1] *i.e.* sequents containing only atomic formulæ

$$\text{Ax}\ \frac{}{\Gamma, \varphi \vdash \varphi, \Delta} \qquad \text{Cut}\ \frac{\Gamma \vdash \varphi, \Delta \quad \Gamma, \varphi \vdash \Delta}{\Gamma \vdash \Delta}$$

$$\text{ContrR}\ \frac{\Gamma \vdash \varphi, \varphi, \Delta}{\Gamma \vdash \varphi, \Delta} \qquad \text{ContrL}\ \frac{\Gamma, \varphi, \varphi \vdash \Delta}{\Gamma, \varphi \vdash \Delta}$$

$$\bot\text{R}\ \frac{\Gamma \vdash \Delta}{\Gamma \vdash \bot, \Delta} \qquad \bot\text{L}\ \frac{}{\Gamma, \bot \vdash \Delta} \qquad \top\text{R}\ \frac{}{\Gamma \vdash \top, \Delta} \qquad \top\text{L}\ \frac{\Gamma \vdash \Delta}{\Gamma, \top \vdash \Delta}$$

$$\wedge\text{R}\ \frac{\Gamma \vdash \varphi_1, \Delta \qquad \Gamma \vdash \varphi_2, \Delta}{\Gamma \vdash \varphi_1 \wedge \varphi_2, \Delta} \qquad \wedge\text{L}\ \frac{\Gamma, \varphi_1, \varphi_2 \vdash \Delta}{\Gamma, \varphi_1 \wedge \varphi_2 \vdash \Delta}$$

$$\vee\text{R}\ \frac{\Gamma \vdash \varphi_1, \varphi_2, \Delta}{\Gamma \vdash \varphi_1 \vee \varphi_2, \Delta} \qquad \vee\text{L}\ \frac{\Gamma, \varphi_1 \vdash \Delta \qquad \Gamma, \varphi_2 \vdash \Delta}{\Gamma, \varphi_1 \vee \varphi_2 \vdash \Delta}$$

$$\Rightarrow\text{R}\ \frac{\Gamma, \varphi_1 \vdash \varphi_2, \Delta}{\Gamma \vdash \varphi_1 \Rightarrow \varphi_2, \Delta} \qquad \Rightarrow\text{L}\ \frac{\Gamma \vdash \varphi_1, \Delta \qquad \Gamma, \varphi_2 \vdash \Delta}{\Gamma, \varphi_1 \Rightarrow \varphi_2 \vdash \Delta}$$

$$\forall\text{R}\ \frac{\Gamma \vdash \varphi, \Delta}{\Gamma \vdash \forall \mathsf{x}.\varphi, \Delta}\ \mathsf{x} \notin \mathcal{FV}(\Gamma, \Delta) \qquad \forall\text{L}\ \frac{\Gamma, \varphi[\mathsf{t}/\mathsf{x}] \vdash \Delta}{\Gamma, \forall \mathsf{x}.\varphi \vdash \Delta}$$

$$\exists\text{R}\ \frac{\Gamma \vdash \varphi[\mathsf{t}/\mathsf{x}], \Delta}{\Gamma \vdash \exists \mathsf{x}.\varphi, \Delta} \qquad \exists\text{L}\ \frac{\Gamma, \varphi \vdash \Delta}{\Gamma, \exists \mathsf{x}.\varphi \vdash \Delta}\ \mathsf{x} \notin \mathcal{FV}(\Gamma, \Delta)$$

Fig. 1. Classical Sequent Calculus LK

inference rule for introducing P on the right

$$\frac{(\Gamma, \Gamma_i \vdash \Delta_i, \Delta)_{1 \leqslant i \leqslant n}}{\Gamma \vdash P, \Delta}\ C \quad .$$

On the left, it similarly decomposes $\varphi \vdash$ until it reaches a sequence of atomic sequents $(\Gamma'_j \vdash \Delta'_j)_{1 \leqslant j \leqslant m}$ and side condition C' yielding similarly the inference rule

$$\frac{(\Gamma, \Gamma'_j \vdash \Delta'_j, \Delta)_{1 \leqslant j \leqslant m}}{\Gamma, P \vdash \Delta}\ C' \quad .$$

As remarked in [7], this indeterministic process may yield several inference rules for introducing P respectively on the right or on the left. One must add all the possible inference rules in order to obtain a complete superdeduction system.

Definition 1 (Superdeduction systems) *If $\mathcal{R}$ is a set of proposition rewrite rules, the superdeduction system associated to $\mathcal{R}$ is the system obtained by adding to LK all the inferences which can be computed from the elements of $\mathcal{R}$.*

The paradigmatic example for superdeduction is the system associated to the proposition rewrite rule $A \subseteq B \quad \rightarrow \quad \forall \mathsf{x}.(\mathsf{x} \in A \Rightarrow \mathsf{x} \in B)$. This rewrite rule yields inference rules

$$\frac{\Gamma, \mathsf{x} \in A \vdash \mathsf{x} \in B, \Delta}{\Gamma \vdash A \subseteq B, \Delta}\ \mathsf{x} \notin \mathcal{FV}(\Gamma, \Delta) \quad \text{and} \quad \frac{\Gamma, \mathsf{t} \in B \vdash \Delta \qquad \Gamma \vdash \mathsf{t} \in A, \Delta}{\Gamma, A \subseteq B \vdash \Delta} \quad .$$

As demonstrated in [1], superdeduction systems are always sound *w.r.t.* predicate logic. Completeness is ensured whenever right-hand sides of proposition rewrite rules do not alternate quantifiers[2]. Cut-elimination is more difficult to obtain: several counterexamples are displayed in [8]. I have proved in [7] that whenever right-hand sides of proposition rewrite rules do not contain universal quantifiers and existential quantifiers at the same time[3], cut-elimination in superdeduction is equivalent to cut-elimination in deduction modulo [6].

In the original paper introducing superdeduction [1], a proof-term language and a cut-elimination reduction are defined for superdeduction, both based on Christian Urban's work on classical sequent calculus [12]. The reduction is proved to be strongly normalising on well-typed terms when the set of proposition rewrite rules $\mathcal{R}$ satisfies the following hypothesis.

Hypothesis 1 *The rewriting relation associated to $\mathcal{R}$ is weakly normalising and confluent and no first-order function symbol appear in the left-hand sides of proposition rewrite rules of $\mathcal{R}$.*

Christian Urban's calculus is not directly related to the Curry-Howard correspondence contrarily to the $\overline{\lambda}\mu\tilde{\mu}$-calculus which relates straightaway to the λ-calculus. In order to explore the computational content of superdeduction inferences, I will define in section 2 an extension of the $\overline{\lambda}\mu\tilde{\mu}$-calculus for superdeduction systems and prove the same strong normalisation result using Hypothesis 1. But before doing so, let us recall the definition of $\overline{\lambda}\mu\tilde{\mu}$.

The $\overline{\lambda}\mu\tilde{\mu}$-calculus is defined as follows. In order to avoid confusion between first-order variables and $\overline{\lambda}\mu\tilde{\mu}$ variables, I will use sans-serif symbols for first-order variables ($\mathsf{x}, \mathsf{y} \dots$) and first-order terms ($\mathsf{t}, \mathsf{u} \dots$). Commands, terms and environments are respectively defined by the grammar in Figure 2(a). The type system is described in Figure 2(c). Reduction rules are depicted in Figure 2(b). I have added a constant environment f in order to realise falsity. I also have added constructions $\lambda\mathsf{x}.\pi$ and $\mathsf{t} \cdot e$ in order to realise universal quantifications respectively on the right and on the left. Implication, universal quantification and falsity are sufficient to express all the connectives in LK. The typing rules

$$\text{FocusR}\ \frac{\Gamma \vdash \pi : A \mid \Delta}{\langle \pi | \alpha \rangle \rhd \Gamma \vdash \alpha : A, \Delta} \qquad \text{and} \qquad \text{FocusL}\ \frac{\Gamma \mid e : A \vdash \Delta}{\langle x | e \rangle \rhd \Gamma, x : A \vdash \Delta}$$

are admissible in the type system of Figure 2(c). Replacing the Cut rule by FocusR and FocusL yields a type system that I will call *cut-free* $\overline{\lambda}\mu\tilde{\mu}$. It is obviously not equivalent to the original type system in Figure 2(c). The reduction relation defined in Figure 2(b) is strongly normalising on well-typed terms as demonstrated in [4].

[2] Formulæ such as $(\forall \mathsf{x}.\varphi) \wedge (\exists \mathsf{y}.\psi)$ are allowed.
[3] Formulæ such as $(\forall \mathsf{x}.\varphi) \wedge (\exists \mathsf{y}.\psi)$ are *not* allowed.

$$\begin{array}{rcll} c & ::= & \langle \pi | e \rangle & \text{(commands)} \\ \pi & ::= & x \mid \lambda x.\pi \mid \mu\alpha.c \mid \lambda \mathtt{x}.\pi & \text{(terms)} \\ e & ::= & \alpha \mid \pi \cdot e \mid \tilde{\mu} x.c \mid \mathtt{t} \cdot e \mid \mathtt{f} & \text{(environments)} \end{array}$$

(a) Grammar

$$\begin{array}{rcl} \langle \lambda x.\pi | \pi' \cdot e \rangle & \to & \langle \pi[\pi'/x] | e \rangle \\ \langle \mu\alpha.c | e \rangle & \to & c[e/\alpha] \\ \langle \pi | \tilde{\mu} x.c \rangle & \to & c[\pi/x] \\ \langle \lambda \mathtt{x}.\pi | \mathtt{t} \cdot e \rangle & \to & \langle \pi[\mathtt{t}/\mathtt{x}] | e \rangle \end{array}$$

(b) Reduction

$$\frac{}{\Gamma, x : A \vdash x : A \mid \Delta} \qquad \frac{}{\Gamma \mid \alpha : A \vdash \alpha : A, \Delta} \qquad \frac{\Gamma, x : A \vdash \pi : B \mid \Delta}{\Gamma \vdash \lambda x.\pi : A \Rightarrow B \mid \Delta}$$

$$\frac{\Gamma \vdash \pi : A \mid \Delta \qquad \Gamma \mid e : B \vdash \Delta}{\Gamma \mid \pi \cdot e : A \Rightarrow B \vdash \Delta} \qquad \text{Cut } \frac{\Gamma \vdash \pi : A \mid \Delta \qquad \Gamma \mid e : A \vdash \Delta}{\langle \pi | e \rangle \rhd \Gamma \vdash \Delta}$$

$$\frac{}{\Gamma \mid \mathtt{f} : \bot \vdash \Delta} \qquad \frac{c \rhd \Gamma \vdash \alpha : A, \Delta}{\Gamma \vdash \mu\alpha.c : A \mid \Delta} \qquad \frac{c \rhd \Gamma, x : A \vdash \Delta}{\Gamma \mid \tilde{\mu} x.c : A \vdash \Delta}$$

$$\frac{\Gamma \vdash \pi : A \mid \Delta}{\Gamma \vdash \lambda \mathtt{x}.\pi : \forall \mathtt{x}.A \mid \Delta} \; \mathtt{x} \notin \mathcal{FV}(\Gamma, \Delta) \qquad \frac{\Gamma \mid e : A[\mathtt{t}/\mathtt{x}] \vdash \Delta}{\Gamma \mid \mathtt{t} \cdot e : \forall \mathtt{x}.A \vdash \Delta}$$

(c) Type System

Fig. 2. The $\overline{\lambda}\mu\tilde{\mu}$-calculus

Notations. Sequences $(a_i)_{1 \leqslant i \leqslant n}$ may be denoted $(a_i)_i$ or just $\bar{a}$ when the upper bound n can be retrieved from the context (or is irrelevant). I may even combine both notations: $(\bar{a}_i)_i$ represents a sequence of sequences $((a_{j,i})_{1 \leqslant j \leqslant m_i})_{1 \leqslant i \leqslant n}$. Finally if $\Gamma = (A_i)_i$ and $\bar{x} = (x_i)_i$ are respectively a sequence of n formulæ and a sequence of n variables, then $\bar{x} : \Gamma$ denotes the (typed) context $x_1 : A_1, x_2 : A_2 \ldots$

2 Extending $\overline{\lambda}\mu\tilde{\mu}$

In the LICS'07 paper introducing superdeduction [1], Christian Urban's calculus is presented as a better choice than $\overline{\lambda}\mu\tilde{\mu}$ for a basis of a proofterm language for superdeduction. In this section, I demonstrate that $\overline{\lambda}\mu\tilde{\mu}$ can be extended for superdeduction as easily as Christian Urban's calculus. Such an extension is my first step towards a computational interpretation of superdeduction, since the $\overline{\lambda}\mu\tilde{\mu}$-calculus relates directly to the Curry-Howard correspondence. An imprecision of the LICS'07 paper is also corrected in the process. The extension of $\overline{\lambda}\mu\tilde{\mu}$ that I will present corrects this mistake. The imprecision concerns first-order quantifications. Indeed a superdeduction inference represents an open derivation which may contain several quantifiers destructions. The structure of these destructions is essential to the definition of the underlying cut-elimination mechanisms. For instance a sequence $\forall\exists$ on the right corresponds to the creation of an eigenvariable, say $\mathtt{x}$, followed by an instanciation by some first-order term, say $\mathtt{t}$, which may contain $\mathtt{x}$ as a free variable. A sequence $\exists\forall$ on the right corresponds to an instanciation by some first-order term, say $\mathtt{t}$, followed by the

creation of an eigenvariable, say $\mathtt{x}$. In this latter case, $\mathtt{t}$ is not allowed to contain $\mathtt{x}$ as a free variable. This distinction is completely erased in the syntax of the LICS'07 extension. It results in an imprecision of the scope of eigenvariables in extended proofterms: the scope is not explicit in the syntax. In my extension of $\overline{\lambda}\mu\tilde{\mu}$, I will correct the syntactical imprecision by introducing a notion of *trace* which represents the correct syntax for a precise syntactical representation of the scopes of eigenvariables in extended proofterms. Then I will present a correct cut-elimination procedure by introducing a notion of *interpretation* for the constructs of the extended $\overline{\lambda}\mu\tilde{\mu}$ relating such constructs to $\overline{\lambda}\mu\tilde{\mu}$ proofterms in a correct way. At the end of the section, a pathological example is depicted to illustrate the imprecision of the LICS'07 extension and the correction of the present extension.

First, let us consider any derivation in LK, potentially unfinished, *i.e.* with open leaves. Since such a derivation is a tree, there exists a natural partial order on its inferences: an inference precedes another if the former is placed under the latter. Such a partial order can easily be extended into a total order (in an indeterministic way). Considering only instances of ∀R, ∀L, ∃R and ∃L, such a total order yields a list L of such instances. Each instance of ∀R or ∃L corresponds to the use of an eigenvariable, say $\mathtt{x}$, that I denote $\mathtt{x}?$. Each instance of ∀L or ∃R corresponds to the instanciation of some first-order variable by a first-order term, say $\mathtt{t}$, that I denote $\mathtt{t}!$. The list L becomes a list whose elements are either of the form $\mathtt{x}?$ or of the form $\mathtt{t}!$. Such a list is called a *trace* for the derivation.

Let us consider a proposition rewrite rule $r : P \to \varphi$ leading to the superdeduction inferences

$$\frac{(\Gamma, \Gamma_i \vdash \Delta_i, \Delta)_i}{\Gamma \vdash P, \Delta}\, C \qquad \text{and} \qquad \frac{(\Gamma, \Gamma'_j \vdash \Delta'_j, \Delta)_j}{\Gamma, P \vdash \Delta}\, C' \quad .$$

Let us consider the first one. Since it is derived from inferences of LK, there exists a derivation of $\vdash \varphi$ with open leaves $(\Gamma_i \vdash \Delta_i)_i$ in LK [8, Property 6.1.3]. Let L be a trace for this derivation. Then the superdeduction inference introducing P on the right is turned into the typing rule

$$\mathrm{rR}\ \frac{(c_i \rhd \Gamma, \overline{x}_i : \Gamma_i \vdash \overline{\alpha}_i : \Delta_i, \Delta)_i}{\Gamma \vdash r(L, (\mu_i(\overline{x}_i, \overline{\alpha}_i).c_i)_i) : P \mid \Delta}\, C$$

Similarly we obtain a corresponding trace L' for the superdeduction inference introducing P on the left which is turned into the typing rule

$$\mathrm{rL}\ \frac{(c'_j \rhd \Gamma, \overline{y}_j : \Gamma'_j \vdash \overline{\beta}_j : \Delta'_j, \Delta)_j}{\Gamma \mid r(L', (\tilde{\mu}_j(\overline{y}_j, \overline{\beta}_j).c'_j)_j) : P \vdash \Delta}\, C' \quad .$$

For example, the inference rules for $\subseteq$ are turned into

$$\frac{c \rhd \Gamma, x : \mathtt{x} \in A \vdash \alpha : \mathtt{x} \in B, \Delta}{\Gamma \vdash r(\mathtt{x}?, \mu(x, \alpha).c) : A \subseteq B, \Delta}\, \mathtt{x} \notin \mathcal{FV}(\Gamma, \Delta)$$

and

$$\frac{c_1 \rhd \Gamma, x : \mathtt{t} \in B \vdash \Delta \qquad c_2 \rhd \Gamma \vdash \alpha : \mathtt{t} \in A, \Delta}{\Gamma, r(\mathtt{t!}, \tilde{\mu}_1(x).c_1, \tilde{\mu}_2(\alpha).c_2) : A \subseteq B \vdash \Delta} \ .$$

We must now define how cuts of the form

$$\langle\ r(L, (\mu_i(\overline{x}_i, \overline{\alpha}_i).c_i)_i)\ \mid\ r(L', (\tilde{\mu}_j(\overline{y}_j, \overline{\beta}_j).c'_j)_j)\ \rangle$$

are reduced. Such reductions are computed using *open* $\overline{\lambda}\mu\tilde{\mu}$, a type system for derivations with open leaves in $\overline{\lambda}\mu\tilde{\mu}$. An open leaf is represented by a *variable command* (symbols $X, Y \ldots$). The types of such variables have the same shape as the types of usual commands in $\overline{\lambda}\mu\tilde{\mu}$-calculus: full sequents $\Gamma \vdash \Delta$. Therefore typing in open $\overline{\lambda}\mu\tilde{\mu}$ is performed in a context Θ which contains a list of typed variable commands of the form $X \rhd \Gamma \vdash \Delta$. As usual, variable commands are allowed to appear only once in such contexts. Typing judgements are denoted

$$\begin{array}{lll} & \Theta \Vdash c \rhd \Gamma \vdash \Delta & \text{when typing a command;} \\ & \Theta \Vdash \Gamma \vdash \pi : A \mid \Delta & \text{when typing a term} \\ \text{and} & \Theta \Vdash \Gamma \mid e : A \vdash \Delta & \text{when typing an environment.} \end{array}$$

Open $\overline{\lambda}\mu\tilde{\mu}$ is obtained by trivially extending cut-free $\overline{\lambda}\mu\tilde{\mu}$ to such judgements and by adding the typing rule

$$\text{Open}\ \frac{}{\Theta; X \rhd S \Vdash X \rhd S} \ .$$

For example, Figure 3 contains a derivation of

$$\begin{array}{l} X \rhd x : C \vdash \alpha : D\ ;\ Y \rhd \vdash \alpha : D, \beta : B \Vdash \\ \qquad \langle \lambda y.\mu\alpha.\langle y | (\mu\beta.Y) \cdot (\tilde{\mu}x.X) \rangle | \gamma \rangle \ \rhd \quad (\vdash \gamma : (B \Rightarrow C) \Rightarrow D) \ . \end{array}$$

(The prefix $X \rhd x : C \vdash \alpha : D\ ;\ Y \rhd \vdash \alpha : D, \beta : B \Vdash$ is omitted for readability.)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{Open}\ \dfrac{}{Y \rhd \vdash \beta : B, \alpha : D}}{\vdash \mu\beta.Y : B \mid \alpha : D} \qquad \text{Open}\ \dfrac{}{X \rhd x : C \vdash \alpha : D}}{\mid \tilde{\mu}x.X : C \vdash \alpha : D}}{\mid (\mu\beta.Y) \cdot (\tilde{\mu}x.X) : (B \Rightarrow C) \vdash \alpha : D}}{\langle y | (\mu\beta.Y) \cdot (\tilde{\mu}x.X) \rangle \rhd y : (B \Rightarrow C) \vdash \alpha : D}}{y : (B \Rightarrow C) \vdash \mu\alpha.\langle y | (\mu\beta.Y) \cdot (\tilde{\mu}x.X) \rangle : D \mid}}{\vdash \lambda y.\mu\alpha.\langle y | (\mu\beta.Y) \cdot (\tilde{\mu}x.X) \rangle : (B \Rightarrow C) \Rightarrow D \mid}}{\langle \lambda y.\mu\alpha.\langle y | (\mu\beta.Y) \cdot (\tilde{\mu}x.X) \rangle | \gamma \rangle \rhd \vdash \gamma : (B \Rightarrow C) \Rightarrow D}$$

Fig. 3. Typing in open $\overline{\lambda}\mu\tilde{\mu}$

The reduction in Figure 2(b) is extended to open $\overline{\lambda}\mu\tilde{\mu}$ by simply defining how subtitutions behave on command variables ($X[t/x]$, $X[e/\alpha]$ or $X[\mathtt{t}/\mathtt{x}]$): they are

turned into *delayed substitutions*, *i.e.* syntactic constructions, denoted $X\{t/x\}$, $X\{e/\alpha\}$ or $X\{\mathtt{t}/\mathtt{x}\}$, which will be turned back into primitive substitutions once X is instanciated.

A typing derivation in open $\overline{\lambda}\mu\tilde{\mu}$ obviously corresponds to a derivation in LK (with open leaves) and whenever L is a trace for the latter derivation, I will say that it is a trace for the typed command, term or environment. Let us reconsider our extended terms

$$r(L, (\mu_i(\overline{x}_i, \overline{\alpha}_i).c_i)_i) \qquad \text{and} \qquad r(L', (\tilde{\mu}_j(\overline{x}_j, \overline{\alpha}_j).c'_j)_j)$$

and their respective typing rules rR and rL. Since such rules *logically* come from decomposing φ respectively on the right and on the left, the sets

$$\left\{ \pi \quad / \quad \begin{array}{l} (X_i \rhd (\overline{x}_i : \Gamma_i \vdash \overline{\alpha}_i : \Delta_i))_i \Vdash \ \vdash \pi : \varphi \text{ well-typed in open } \overline{\lambda}\mu\tilde{\mu} \\ \text{and } L \text{ is a trace for } \pi \end{array} \right\}$$

and

$$\left\{ e \quad / \quad \begin{array}{l} (Y_j \rhd (\overline{y}_j : \Gamma'_j \vdash \overline{\beta}_j : \Delta'_j))_j \Vdash e : \varphi \vdash \text{ well-typed in open } \overline{\lambda}\mu\tilde{\mu} \\ \text{and } L' \text{ is a trace for } e \end{array} \right\}$$

are both non-empty. Therefore we choose indeterministically one π and one e in these sets. This term and this environment are called the respective *interpretations* of $r(L, (\mu_i(\overline{x}_i, \overline{\alpha}_i).c_i)_i)$ and $r(L', (\tilde{\mu}_j(\overline{x}_j, \overline{\alpha}_j).c'_j)_j)$. The command $\langle \pi | e \rangle$ reduces (using the reduction in Figure 2(b)) to normal forms. Cut-elimination on extended terms is defined by adding to the reduction relation defined in Figure 2(b) the rule

$$\langle (\mu r(L, (\mu_i(\overline{x}_i, \overline{\alpha}_i).c_i)_i) | r(L', (\tilde{\mu}_j(\overline{y}_j, \overline{\beta}_j).c'_j)_j) \rangle \quad \to \quad c[(c_i/X_i)_i, (c'_j/Y_j)_j]$$

for each normal form c of $\langle \pi | e \rangle$ (where delayed substitutions $\{\cdot/\cdot\}$ are replaced by primitive substitutions $[\cdot/\cdot]$).

Let us reconsider the inclusion example. The term $\lambda\mathtt{x}.\lambda x.\mu\alpha.X$ is a potential interpretation of $r(\mathtt{x}?, \mu(x, \alpha).c)$. The environment $\mathtt{t} \cdot (\mu\beta.Y) \cdot (\tilde{\mu}y.Z)$ is a potential interpretation of $r(\mathtt{t}!, \tilde{\mu}_1(y).c_1, \tilde{\mu}_2(\beta).c_2)$. The cut

$$\langle \lambda\mathtt{x}.\lambda x.\mu\alpha.X | \mathtt{t} \cdot (\mu\beta.Y) \cdot (\tilde{\mu}y.Z) \rangle$$

has two normal forms, namely

$$X\{\mathtt{t}/\mathtt{x}\}\{(\mu\beta.Y)/x\}\{\tilde{\mu}y.Z/\alpha\} \qquad \text{and} \qquad Z\{\mu\alpha.X\{\mathtt{t}/\mathtt{x}\}\{(\mu\beta.Y)/x\}/y\} \ .$$

Therefore a cut

$$\langle r(\mathtt{x}?, \mu(x, \alpha).c) | r(\mathtt{t}!, \tilde{\mu}_1(y).c_1, \tilde{\mu}_2(\beta).c_2) \rangle$$

reduces to

$$c[\mathtt{t}/\mathtt{x}][(\mu\beta.c_2)/x][\tilde{\mu}y.c_1/\alpha] \qquad \text{and} \qquad c_1[\mu\alpha.c[\mathtt{t}/\mathtt{x}][(\mu\beta.c_2)/x]/y] \ .$$

If $\mathcal{R}$ is a set of proposition rewrite rule, I will denote $\overline{\lambda}\mu\tilde{\mu}_{\mathcal{R}}$ the type system resulting of extending the type system of Figure 2(c) with the typing rules for $\mathcal{R}$. I will denote $\to_{\overline{\lambda}\mu\tilde{\mu}_{\mathcal{R}}}$ the reduction relation of Figure 2(b) extended by the reduction rules for $\mathcal{R}$.

Theorem 1 (Subject Reduction) *For all $\mathcal{R}$, typability in $\overline{\lambda}\mu\tilde{\mu}_{\mathcal{R}}$ is preserved by reduction through $\rightarrow_{\overline{\lambda}\mu\tilde{\mu}_{\mathcal{R}}}$.*

Proof. The only case worth considering is a reduction from

$$\langle r(L, (\mu_i(\overline{x}_i, \overline{\alpha}_i).c_i)_i) | r(L', (\tilde{\mu}_j(\overline{y}_j, \overline{\beta}_j).c'_j)_j) \rangle$$

to $c[(c_i/X_i)_i, (c'_j/Y_j)_j]$. By definition, c is a normal form of $\langle \pi | e \rangle$ where π and e are the respective interpretations of $r(L, (\mu_i(\overline{x}_i, \overline{\alpha}_i).c_i)_i)$ and $r(L', (\tilde{\mu}_j(\overline{y}_j, \overline{\beta}_j).c'_j)_j)$. $(X_i \rhd (\overline{x}_i : \Gamma_i \vdash \overline{\alpha}_i : \Delta_i))_i \Vdash \vdash \pi : \varphi$ and $(Y_j \rhd (\overline{y}_j : \Gamma'_j \vdash \overline{\beta}_j : \Delta'_j))_j \Vdash e : \varphi \vdash$ are well-typed in open $\overline{\lambda}\mu\tilde{\mu}$. Therefore by subject reduction in open $\overline{\lambda}\mu\tilde{\mu}$

$$(X_i \rhd (\overline{x}_i : \Gamma_i \vdash \overline{\alpha}_i : \Delta_i))_i \,;\, (Y_j \rhd (\overline{y}_j : \Gamma'_j \vdash \overline{\beta}_j : \Delta'_j))_j \Vdash c \rhd \vdash$$

is also well-typed in open $\overline{\lambda}\mu\tilde{\mu}$. This demonstrates that whatever the type of $\langle r(L, (\mu_i(\overline{x}_i, \overline{\alpha}_i).c_i)_i) | r(L', (\tilde{\mu}_j(\overline{y}_j, \overline{\beta}_j).c'_j)_j) \rangle$ is, the command $c[(c_i/X_i)_i, (c'_j/Y_j)_j]$ has the same type.

Theorem 2 (Strong Normalisation) *For all $\mathcal{R}$* **satisfying hypothesis 1**, *$\rightarrow_{\overline{\lambda}\mu\tilde{\mu}_{\mathcal{R}}}$ is strongly normalising on commands, terms and environments that are well-typed in $\overline{\lambda}\mu\tilde{\mu}_{\mathcal{R}}$.*

Proof. Hypothesis 1 implies that any formula φ has a unique normal form for $\mathcal{R}$ that we denote $\varphi\downarrow_p$. Let us denote $\rightarrow_e$ the rewrite relation defined by replacing extended terms for superdeduction by their interpretations.

$$\begin{array}{lll} r(L, (\mu_i(\overline{x}_i, \overline{\alpha}_i).c_i)_i) & \rightarrow_e & \pi \\ r(L', (\tilde{\mu}_j(\overline{y}_j, \overline{\beta}_j).c'_j)_j) & \rightarrow_e & e \\ \multicolumn{3}{c}{\ldots} \end{array}$$

Such a rewrite relation is strongly normalising and confluent, therefore yielding for any extended command c, term π or environment e a normal form denoted $c\downarrow_e$, $\pi\downarrow_e$ of $e\downarrow_e$. Such normal forms are raw $\overline{\lambda}\mu\tilde{\mu}$ commands, terms or environments. Strong normalisation of our extended cut-elimination reduction comes from the facts that **1.** $c \rhd \Gamma \vdash \Delta$ well-typed in our extended type system implies that $c\downarrow_e \rhd (\Gamma)\downarrow_p \vdash (\Delta)\downarrow_p$ well-typed in $\overline{\lambda}\mu\tilde{\mu}$; **2.** $\Gamma \vdash \pi : A \mid \Delta$ well-typed in our extended type system implies that $(\Gamma)\downarrow_p \vdash \pi\downarrow_e : A\downarrow_p \mid (\Delta)\downarrow_p$ well-typed in $\overline{\lambda}\mu\tilde{\mu}$; **3.** $\Gamma \mid e : A \vdash \Delta$ well-typed in our extended type system implies that $(\Gamma)\downarrow_p \mid e\downarrow_e : A\downarrow_p \vdash (\Delta)\downarrow_p$ well-typed in $\overline{\lambda}\mu\tilde{\mu}$; **4.** $c \rightarrow c'$ implies $c\downarrow_e \rightarrow^+ c'\downarrow_e$ **5.** $\pi \rightarrow \pi'$ implies $\pi\downarrow_e \rightarrow^+ \pi'\downarrow_e$ **6.** $e \rightarrow e'$ implies $e\downarrow_e \rightarrow^+ e'\downarrow_e$. The hypothesis on first-order function symbols (see Hypothesis 1) is crucial in establishing points 1 to 3: indeed for any formula φ and any first-order substitution σ, it must be the case that $(\varphi\downarrow_p)\sigma = (\varphi\sigma)\downarrow_p$. These six points demonstrate that through $\downarrow_e$ and $\downarrow_p$, the $\overline{\lambda}\mu\tilde{\mu}$-calculus simulates our extended calculus. Strong normalisation of $\overline{\lambda}\mu\tilde{\mu}$ therefore implies strong normalisation of our extended reduction.

The end of this section is dedicated to a pathological example for superdeduction: the proposition rewrite rule

$$r \ : \ P \to (\exists \mathsf{x}_1.\forall \mathsf{x}_2.A(\mathsf{x}_1,\mathsf{x}_2)) \vee (\exists \mathsf{y}_1.\forall \mathsf{y}_2.B(\mathsf{y}_1,\mathsf{y}_2))$$

whose *most general* superdeduction rules are

$$\frac{\Gamma \vdash A(\mathsf{t},\mathsf{x}_2), B(\mathsf{u},\mathsf{y}_2), \Delta}{\Gamma \vdash P, \Delta} \begin{cases} \mathsf{x}_2 \notin \mathcal{FV}(\Gamma,\Delta,\mathsf{u}) \\ \mathsf{y}_2 \notin \mathcal{FV}(\Gamma,\Delta) \end{cases}$$

and

$$\frac{\Gamma \vdash A(\mathsf{t},\mathsf{x}_2), B(\mathsf{u},\mathsf{y}_2), \Delta}{\Gamma \vdash P, \Delta} \begin{cases} \mathsf{x}_2 \notin \mathcal{FV}(\Gamma,\Delta) \\ \mathsf{y}_2 \notin \mathcal{FV}(\Gamma,\Delta,\mathsf{t}) \end{cases} .$$

The LICS'07 proofterm extension transforms these two inferences into a unique proofterm $rR(\lambda \mathsf{x}_2.\lambda \mathsf{y}_2.(\lambda\alpha.\lambda\beta.m),\mathsf{t},\mathsf{u},\gamma)$. It is obviously inaccurate with respect to the scope of x_2 and y_2: in the proofterm there is no mention that either t is not in the scope of y_2 or u is not in the scope of x_2. This fact is not reflected in the pure syntax but in the typing rules

$$\frac{m \rhd \Gamma \vdash \alpha : A(\mathsf{t},\mathsf{x}_2), \beta : B(\mathsf{u},\mathsf{y}_2), \Delta}{rR(\lambda \mathsf{x}_2.\lambda \mathsf{y}_2.(\lambda\alpha.\lambda\beta.m),\mathsf{t},\mathsf{u},\gamma) \rhd \Gamma \vdash \gamma : P, \Delta} \begin{cases} \mathsf{x}_2 \notin \mathcal{FV}(\Gamma,\Delta,\mathsf{u}) \\ \mathsf{y}_2 \notin \mathcal{FV}(\Gamma,\Delta) \end{cases}$$

and

$$\frac{m \rhd \Gamma \vdash \alpha : A(\mathsf{t},\mathsf{x}_2), \beta : B(\mathsf{u},\mathsf{y}_2), \Delta}{rR(\lambda \mathsf{x}_2.\lambda \mathsf{y}_2.(\lambda\alpha.\lambda\beta.m),\mathsf{t},\mathsf{u},\gamma) \rhd \Gamma \vdash \gamma : P, \Delta} \begin{cases} \mathsf{x}_2 \notin \mathcal{FV}(\Gamma,\Delta) \\ \mathsf{y}_2 \notin \mathcal{FV}(\Gamma,\Delta,\mathsf{t}) \end{cases} .$$

Let us see now this mistake is corrected in our extension of $\overline{\lambda}\mu\tilde{\mu}$. Traces for the superdeduction inferences are respectively $\mathsf{u}!\,\mathsf{y}_2?\,\mathsf{t}!\,\mathsf{x}_2?$ and $\mathsf{t}!\,\mathsf{x}_2?\,\mathsf{u}!\,\mathsf{y}_2?$. These traces clearly specify whether t is not in the scope of y_2 or u is not in the scope of x_2. My extension of $\overline{\lambda}\mu\tilde{\mu}$ translates these superdeduction inferences into the typing rule

$$\frac{c \rhd \Gamma \vdash \alpha : A(\mathsf{t},\mathsf{x}_2), \beta : B(\mathsf{u},\mathsf{y}_2), \Delta}{\Gamma \vdash r(\mathsf{u}!\,\mathsf{y}_2?\,\mathsf{t}!\,\mathsf{x}_2?, \mu(\alpha,\beta).m) : P \mid \Delta} \begin{cases} \mathsf{x}_2 \notin \mathcal{FV}(\Gamma,\Delta,\mathsf{u}) \\ \mathsf{y}_2 \notin \mathcal{FV}(\Gamma,\Delta) \end{cases}$$

and

$$\frac{c \rhd \Gamma \vdash \alpha : A(\mathsf{t},\mathsf{x}_2), \beta : B(\mathsf{u},\mathsf{y}_2), \Delta}{\Gamma \vdash r(\mathsf{t}!\,\mathsf{x}_2?\,\mathsf{u}!\,\mathsf{y}_2?, \mu(\alpha,\beta).c) : P \mid \Delta} \begin{cases} \mathsf{x}_2 \notin \mathcal{FV}(\Gamma,\Delta) \\ \mathsf{y}_2 \notin \mathcal{FV}(\Gamma,\Delta,\mathsf{t}) \end{cases} .$$

The proofterms (and the typing rules) reflect the scope of the eigenvariables. The *interpretation* of $r(\mathsf{u}!\,\mathsf{y}_2?\,\mathsf{t}!\,\mathsf{x}_2?, \mu(\alpha,\beta).m)$ is by definition a term well-typed in $\overline{\lambda}\mu\tilde{\mu}$ whose trace is $\mathsf{u}!\,\mathsf{y}_2?\,\mathsf{t}!\,\mathsf{x}_2?$ and the *interpretation* of $r(\mathsf{t}!\,\mathsf{x}_2?\,\mathsf{u}!\,\mathsf{y}_2?, \mu(\alpha,\beta).c)$ is by definition a term well-typed in $\overline{\lambda}\mu\tilde{\mu}$ whose trace is $\mathsf{t}!\,\mathsf{x}_2?\,\mathsf{u}!\,\mathsf{y}_2?$. This trace restriction implies that $r(\mathsf{u}!\,\mathsf{y}_2?\,\mathsf{t}!\,\mathsf{x}_2?, \mu(\alpha,\beta).m)$ and $r(\mathsf{t}!\,\mathsf{x}_2?\,\mathsf{u}!\,\mathsf{y}_2?, \mu(\alpha,\beta).c)$ behave differently with respect to cut-elimination.

3 Conclusion

This extension of the $\overline{\lambda}\mu\tilde{\mu}$-calculus is my first step towards a computational interpretation of superdeduction. Indeed it refutes the idea [1] that Christian Urban's calculus is a better basis for a proofterm language for superdeduction: it is as easy to extend $\overline{\lambda}\mu\tilde{\mu}$ syntax, typing and reduction for superdeduction. However the extension presented in this short paper is a mechanical transcription of the LICS'07 extension. It does not explore any further the computational content of cut-elimination for superdeduction.

I believe that one of the key ingredients towards this goal is pattern-matching. Indeed superdeduction systems historically come from *supernatural deduction* [13], an extension of natural deduction designed to type the rewriting-calculus (*a.k.a.* ρ-calculus) [3]. Supernatural deduction turns proposition rewrite rules of the form

$$r \ : \ P \quad \rightarrow \quad \forall \bar{\mathsf{x}}.((A_1 \wedge A_2 \dots A_n) \Rightarrow C)$$

into inference rules for natural deduction

$$\frac{\Gamma, A_1 \dots A_n \vdash C}{\Gamma \vdash P} \ \bar{\mathsf{x}} \notin \mathcal{FV}(\Gamma) \qquad \text{and} \qquad \frac{\Gamma \vdash P \qquad (\Gamma \vdash A_i[\bar{\mathsf{t}}/\bar{\mathsf{x}}])_i}{\Gamma \vdash C[\bar{\mathsf{t}}/\bar{\mathsf{x}}]} \ .$$

(The first rule is an introduction rule and the second is an elimination rule.) The rewriting calculus is an extension of the λ-calculus where rewrite rules replace lambda-abstractions. The idea underlying the relation between supernatural deduction and rewriting calculus is that the proposition rewrite rule r corresponds to a specific pattern $r(\bar{\mathsf{x}}, x_1 \dots x_n)$. The introduction rule types an abstraction on this pattern (*i.e.* a rewrite rule)

$$\frac{\Gamma, x_1 : A_1 \dots x_n : A_n \vdash \pi : C}{\Gamma \vdash r(\bar{\mathsf{x}}, x_1 \dots x_n) \rightarrow \pi : P} \ \bar{\mathsf{x}} \notin \mathcal{FV}(\Gamma) \ .$$

Dually the elimination rule types an application on this pattern

$$\frac{\Gamma \vdash \pi : P \qquad (\Gamma \vdash \pi_i : A_i[\bar{\mathsf{t}}/\bar{\mathsf{x}}])_i}{\Gamma \vdash \pi \, r(\bar{\mathsf{t}}, \pi_1 \dots \pi_n) : C} \ .$$

Supernatural deduction systems (in intuitionnistic natural deduction) have later been transformed into superdeduction systems (in classical sequent calculus) in order to handle more general proposition rewrite rules. This transformation from supernatural deduction to superdeduction systems should not break the relation with pattern matching. Indeed cut-elimination in sequent calculus relates to pattern matching [2]. Recent analysis shows that the duality between patterns and terms reflects the duality between phases in focused proof systems [14]. Finally I demonstrated [7, 8] that superdeduction systems share strong similarities with focused proof systems such as LKF [9, 10], a focused sequent calculus for classical logic. Answers should naturally arise from the study of the computational content of such focused systems [11, 5].

References

1. Paul Brauner, Clément Houtmann, and Claude Kirchner. Principles of superdeduction. In *LICS*, pages 41–50, 2007.
2. Serenella Cerrito and Delia Kesner. Pattern matching as cut elimination. *Theor. Comput. Sci.*, 323(1-3):71–127, 2004.
3. Horatiu Cirstea and Claude Kirchner. The rewriting calculus –- part i and ii. *Logic Journal of the Interest Group in Pure and Applied Logics*, 9(3):427–498, May 2001.
4. Pierre-Louis Curien and Hugo Herbelin. The duality of computation. In *ICFP*, pages 233–243, 2000.
5. Pierre-Louis Curien and Guillaume Munch-Maccagnoni. The duality of computation under focus. In *IFIP TCS*, 2010. Accepted.
6. Gilles Dowek, Thérèse Hardin, and Claude Kirchner. Theorem proving modulo. *Journal of Automated Reasoning*, 31(1):33–72, Nov 2003.
7. Clément Houtmann. Axiom directed focusing. In *TYPES*, pages 169–185, 2008.
8. Clément Houtmann. *Représentation et interaction des preuves en superdéduction modulo*. PhD thesis, Université Henri Poincaré, Nancy Universités, March 2010.
9. Chuck Liang and Dale Miller. Focusing and polarization in intuitionistic logic. In *CSL*, pages 451–465, 2007.
10. Chuck Liang and Dale Miller. A unified sequent calculus for focused proofs. In *LICS*, pages 355–364. IEEE Computer Society, 2009.
11. Guillaume Munch-Maccagnoni. Focalisation and classical realisability. In *CSL*, pages 409–423, 2009.
12. Christian Urban. *Classical Logic and Computation*. PhD thesis, University of Cambridge, October 2000.
13. Benjamin Wack. *Typage et déduction dans le calcul de réécriture*. PhD thesis, Université Henri Poincaré, Nancy 1, October 2005.
14. Noam Zeilberger. Focusing and higher-order abstract syntax. In *POPL*, pages 359–369, 2008.