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

Clément Houtmann
INRIA Saclay - Île de France and LIX/École Polytechnique, 91128 Palaiseau Cedex
Clement.Houtmann@inria.fr

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 the computational content of Christian Urban's calculus is not directly related to the ($\lambda$-calculus based) Curry-Howard correspondence. In contrast the $\overline{\lambda}\mu\tilde{\mu}$-calculus is a $\lambda$-calculus for classical sequent calculus. This short paper is a first step towards a further exploration of the computational content of superdeduction proofs, for we extend the $\overline{\lambda}\mu\tilde{\mu}$-calculus in order to obtain a proofterm langage together with a cut-elimination reduction for superdeduction. We 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, the algorithm decomposes (bottom-up) the sequent $\vdash \varphi$ using $\mathrm{LK}\backslash\{\mathrm{Cut}, \mathrm{ContrR}, \mathrm{ContrL}\}$ (non-deterministically) until it reaches a sequence of atomic sequents[1] $(\Gamma_i \vdash \Delta_i)_{1\leqslant i\leqslant n}$. During this decomposition, each application of $\exists$L and $\forall$R corresponds to a side condition $x \notin \mathscr{FV}(\Upsilon)$ for some first-order variable $x$ and for some list of formula $\Upsilon$. This particular decomposition of $\vdash \varphi$ then leads to the inference rule

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

for introducing $P$ on the right where $C$ is the conjunction of the side conditions. On the left, the algorithm similarly decomposes $\varphi \vdash$ until it reaches a sequence of atomic sequents $(\Gamma'_j \vdash \Delta'_j)_{1\leqslant j\leqslant m}$ and a conjunction of side conditions $C'$ yielding similarly the inference rule

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

[1] *i.e.* sequents containing only atomic formulæ

Steffen van Bakel, Stefano Berardi, Ulrich Berger (Eds.):
Classical Logic and Computation 2010 (Cl&C'10)
EPTCS 47, 2011, pp. 34–43, doi:10.4204/EPTCS.47.5

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

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

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

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

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

Figure 1: Classical Sequent Calculus LK

As remarked in [7], this non-deterministic algorithm may return 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 [1])**.** *If $\mathscr{R}$ is a set of proposition rewrite rules, the superdeduction system associated with $\mathscr{R}$ is obtained by adding to LK all the inferences which can be computed from the elements of $\mathscr{R}$.*

The paradigmatic example for superdeduction is the system associated with the proposition rewrite rule $A\subseteq B \ \rightarrow\ \forall x.(x\in A\Rightarrow x\in B)$ which defines the inclusion predicate $\subseteq$. This rewrite rule yields inference rules

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

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]. We 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* (another formalism which removes computational arguments from proofs by reasoning modulo rewriting on propositions [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 [13]. The reduction is proved to be strongly normalising on well-typed terms when the set of proposition rewrite rules $\mathscr{R}$ satisfies the following hypothesis.

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

[2]Formulæ such as $(\forall x.\varphi)\wedge(\exists y.\psi)$ are allowed.

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

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

(a) Grammar

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

(b) Reduction

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

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

$$\frac{c \rhd \Gamma \vdash \alpha : A, \Delta}{\Gamma \vdash \mu\alpha.c : A \mid \Delta} \qquad \frac{c \rhd \Gamma, x : A \vdash \Delta}{\Gamma \mid \tilde{\mu}x.c : A \vdash \Delta} \qquad \frac{\Gamma \vdash \pi : A \mid \Delta}{\Gamma \vdash \lambda \mathsf{x}.\pi : \forall \mathsf{x}.A \mid \Delta} \, \mathsf{x} \notin \mathscr{FV}(\Gamma, \Delta) \qquad \frac{\Gamma \mid e : A[\mathsf{t}/\mathsf{x}] \vdash \Delta}{\Gamma \mid \mathsf{t} \cdot e : \forall \mathsf{x}.A \vdash \Delta}$$

(c) Type System

Figure 2: The $\overline{\lambda}\mu\tilde{\mu}$-calculus

The computational content of Christian Urban's calculus is not directly related to the (functional) Curry-Howard correspondence whereas the $\overline{\lambda}\mu\tilde{\mu}$-calculus [4] is a $\lambda$-calculus for sequent calculus. In order to explore the computational content of superdeduction inferences, we 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 the $\overline{\lambda}\mu\tilde{\mu}$-calculus.

**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, we will use sans-serif symbols for first-order variables ($\mathsf{x}, \mathsf{y} \ldots$) and first-order terms ($\mathsf{t}, \mathsf{u} \ldots$). 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). We have added a constant environment $\mathsf{f}$ in order to realise falsity. We 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 we 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 [12].

**Notations.** Sequences $(a_i)_{1 \leqslant i \leqslant n}$ may be denoted $(a_i)_i$ or just $\overline{a}$ when the upper bound $n$ can be retrieved from the context (or is irrelevant). Both notations may even be combined: $(\overline{a_i})_i$ 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 $\overline{x} = (x_i)_i$ are respectively a sequence of $n$ formulæ and a sequence of $n$ variables, then $\overline{x} : \Gamma$ denotes the (typed) context $x_1 : A_1, x_2 : A_2 \ldots$

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

In the paper introducing superdeduction [1], Christian Urban's calculus is presented as a better choice than the $\overline{\lambda}\mu\tilde{\mu}$-calculus for a basis of a proofterm language for superdeduction. In this section, we refute this claim and demonstrate that the $\overline{\lambda}\mu\tilde{\mu}$-calculus is as suitable as Christian Urban's calculus. Such an extension is a first step towards a Curry-Howard based computational interpretation of superdeduction, since the $\overline{\lambda}\mu\tilde{\mu}$-calculus relates directly to the $\lambda$-calculus. An inaccuracy of the original paper [1] is also corrected in the process. The extension of the $\overline{\lambda}\mu\tilde{\mu}$-calculus that we will present corrects this mistake. The imprecision concerns first-order quantifications. Indeed a superdeduction inference represents an open derivation which may contain several quantifier destructions. The structure organizing 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 $x$, followed by an instantiation by some first-order term, say $t$, which may contain $x$ as a free variable. A sequence $\exists\forall$ on the right corresponds to an instantiation by some first-order term, say $t$, followed by the creation of an eigenvariable, say $x$. In this latter case, $t$ is not allowed to contain $x$ as a free variable. This distinction is completely erased in the syntax of the original extension [1]. It results in an imprecision of the scope of eigenvariables in extended proofterms: the scope is not explicit in the syntax. In our extension of the $\overline{\lambda}\mu\tilde{\mu}$-calculus, this syntactical imprecision is corrected 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 we present a correct cut-elimination procedure by introducing a notion of *interpretation* for the constructs of the extended $\overline{\lambda}\mu\tilde{\mu}$-calculus 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 original extension [1] and the correction of the present extension.

First, let us consider any derivation in LK, potentially unfinished, *i.e.* with leaves that remain unproven. 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 a non-deterministic way). Considering only instances of $\forall$R, $\forall$L, $\exists$R and $\exists$L, such a total order returns a list $L$ of such instances. Each instance of $\forall$R or $\exists$L corresponds to the use of an eigenvariable, say $x$. Such a use will be denoted $x?$. Each instance of $\forall$L or $\exists$R corresponds to the instantiation of some first-order variable by a first-order term, say $t$. Such a use will be denoted $t!$. The list $L$ becomes a list whose elements are either of the form $x?$ or of the form $t!$. 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

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

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

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

Here variables $\overline{x}_i$ and $\overline{\alpha}_i$ are bound in $c_i$ for each $i$. Similarly we obtain a corresponding trace $L'$ for the

superdeduction inference introducing $P$ on the left which is turned into the typing rule

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

Here variables $\overline{y}_j$ and $\overline{\beta}_j$ are bound in $c'_j$ for each $j$. For example, the inference rules for $\subseteq$ are turned into

$$\frac{c \rhd \Gamma, x : \mathtt{x} \in A \vdash \alpha : \mathtt{x} \in B, \Delta}{\Gamma \vdash r(\mathtt{x}?, \mu(x, \alpha).c) : A \subseteq B, \Delta}\ \mathtt{x} \notin \mathscr{FV}(\Gamma, \Delta) \quad \text{and} \quad \frac{c_1 \rhd \Gamma, x : \mathtt{t} \in B \vdash \Delta \qquad c_2 \rhd \Gamma \vdash \alpha : \mathtt{t} \in A, \Delta}{\Gamma, r(\mathtt{t}!, \tilde{\mu}_1(x).c_1, \tilde{\mu}_2(\alpha).c_2) : A \subseteq B \vdash \Delta} \quad .$$

If $\mathscr{R}$ is a set of proposition rewrite rules, the type system resulting of extending the type system of Figure 2(c) with the typing rules for $\mathscr{R}$ is denoted $\overline{\lambda}\mu\tilde{\mu}_{\mathscr{R}}$.

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[4] in the $\overline{\lambda}\mu\tilde{\mu}$-calculus type system. 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 $\Theta$ 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 extending cut-free $\overline{\lambda}\mu\tilde{\mu}$ to such judgements and by adding the typing rule

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

For example, Figure 3 contains a derivation of

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

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

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). If $K$ is a typed command, term or environment, then a trace for K is a trace for the derivation corresponding to $K$. Let us reconsider our extended terms

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

[4] *i.e.* leaves that remain unproven

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

Figure 3: Typing in open $\overline{\lambda}\mu\tilde{\mu}$

and their respective typing rules rR and rL. The sets

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

and

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

are both non-empty: Indeed by construction of the superdeduction inference rules, we know that there exists a derivation in LK of $\vdash\varphi$ (resp. $\varphi\vdash$) from premisses $(\Gamma_i\vdash\Delta_i)_i$ (resp. $(\Gamma'_j\vdash\Delta'_j)_j$) such that $L$ (resp. $L'$) is a trace for this derivation. Therefore by logical completeness of (open) $\overline{\lambda}\mu\tilde{\mu}$, there exists at least one term in $\widetilde{\mathrm{rR}}$ (resp. one environment in $\widetilde{\mathrm{rL}}$). Each term $\pi\in\widetilde{\mathrm{rR}}$ intuitively represents $r(L,(\mu_i(\overline{x}_i,\overline{\alpha}_i).X_i)_i)$ in open $\overline{\lambda}\mu\tilde{\mu}$. Each environment $e\in\widetilde{\mathrm{rL}}$ intuitively represents $r(L',(\tilde{\mu}_j(\overline{x}_j,\overline{\alpha}_j).c'_j)_j)$ in open $\overline{\lambda}\mu\tilde{\mu}$. Therefore whenever $\pi$ and $e$ are respectively in $\widetilde{\mathrm{rR}}$ and $\widetilde{\mathrm{rL}}$, any normal form $\langle\pi|e\rangle$ can be chosen as a direct reduct of

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

We suppose that for each typing rule rR (resp. rL) one specific $\pi\in\widetilde{\mathrm{rR}}$ (resp. one specific $e\in\widetilde{\mathrm{rL}}$) is distinguished. This term (resp. this environment) is called the *interpretation* of $r(L,(\mu_i(\overline{x}_i,\overline{\alpha}_i).X_i)_i)$ (resp. $r(L',(\tilde{\mu}_j(\overline{x}_j,\overline{\alpha}_j).Y_j)_j)$). Then for each normal form $c$ of $\langle\pi|e\rangle$, the rule

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

is added to the cut-elimination reduction (delayed substitutions $\{\cdot/\cdot\}$ are replaced in $c$ by primitive substitutions $[\cdot/\cdot]$).

Let us reconsider the inclusion example. The term $\pi=\lambda\mathsf{x}.\lambda x.\mu\alpha.X$ is a potential interpretation of $r(\mathsf{x}?,\mu(x,\alpha).c)$. Indeed

$$X\rhd x:\mathsf{x}\in A\vdash\mathsf{x}\in B\Vdash\vdash\pi:\forall\mathsf{x}.\mathsf{x}\in A\Rightarrow\mathsf{x}\in B\mid$$

is well-typed in open $\overline{\lambda}\mu\tilde{\mu}$ as demonstrated in Figure 4(a) and $\mathsf{x}?$ is a trace for $\pi$. The environment $e=\mathsf{t}\cdot(\mu\beta.Y)\cdot(\tilde{\mu}y.Z)$ is a potential interpretation of $r(\mathsf{t}!,\tilde{\mu}_1(y).c_1,\tilde{\mu}_2(\beta).c_2)$. Indeed

$$Y\rhd\vdash\beta:\mathsf{t}\in A,Z\rhd y:\mathsf{t}\in B\Vdash\mid e:\forall\mathsf{x}.\mathsf{x}\in A\Rightarrow\mathsf{x}\in B\vdash$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{Open}\ \dfrac{}{X \rhd x : \mathsf{x} \in A \vdash \mathsf{x} \in B \Vdash X \rhd x : \mathsf{x} \in A \vdash \alpha : \mathsf{x} \in B}}{X \rhd x : \mathsf{x} \in A \vdash \mathsf{x} \in B \Vdash x : \mathsf{x} \in A \vdash \mu\alpha.X : \mathsf{x} \in B \mid}}{X \rhd x : \mathsf{x} \in A \vdash \mathsf{x} \in B \Vdash x : \mathsf{x} \in A \vdash \mu\alpha.X : \mathsf{x} \in B \mid}}{X \rhd x : \mathsf{x} \in A \vdash \mathsf{x} \in B \Vdash\vdash \lambda x.\mu\alpha.X : \mathsf{x} \in A \Rightarrow \mathsf{x} \in B \mid}}{X \rhd x : \mathsf{x} \in A \vdash \mathsf{x} \in B \Vdash\vdash \lambda\mathsf{x}.\lambda x.\mu\alpha.X : \forall\mathsf{x}.\mathsf{x} \in A \Rightarrow \mathsf{x} \in B \mid}$$

(a) Typing $\lambda\mathsf{x}.\lambda x.\mu\alpha.X$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{Open}\ \dfrac{}{Y \rhd \vdash \beta : \mathsf{t} \in A; Z \rhd y : \mathsf{t} \in B \Vdash Y \rhd \vdash \beta : \mathsf{t} \in A}}{Y \rhd \vdash \beta : \mathsf{t} \in A; Z \rhd y : \mathsf{t} \in B \Vdash\vdash \mu\beta.Y : \mathsf{t} \in A \mid}}{\vdots}}{\dfrac{\text{Open}\ \dfrac{}{Y \rhd \vdash \beta : \mathsf{t} \in A; Z \rhd y : \mathsf{t} \in B \Vdash Z \rhd y : \mathsf{t} \in B \vdash}}{Y \rhd \vdash \beta : \mathsf{t} \in A; Z \rhd y : \mathsf{t} \in B \Vdash\vdash \mid \tilde{\mu}y.Z : \mathsf{t} \in B \vdash}}}{Y \rhd \vdash \beta : \mathsf{t} \in A; Z \rhd y : \mathsf{t} \in B \Vdash\vdash \mid (\mu\beta.Y) \cdot (\tilde{\mu}y.Z) : \mathsf{t} \in A \Rightarrow \mathsf{t} \in B \vdash}}{Y \rhd \vdash \beta : \mathsf{t} \in A, Z \rhd y : \mathsf{t} \in B \Vdash\vdash \mid \mathsf{t} \cdot (\mu\beta.Y) \cdot (\tilde{\mu}y.Z) : \forall\mathsf{x}.\mathsf{x} \in A \Rightarrow \mathsf{x} \in B \vdash}$$

(b) Typing $\mathsf{t} \cdot (\mu\beta.Y) \cdot (\tilde{\mu}y.Z)$

Figure 4: Typing interpretations for inclusion

is well-typed in open $\overline{\lambda}\mu\tilde{\mu}$ as demonstrated in Figure 4(b) and t! is a trace for $e$. The cut

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

has two normal forms, namely

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

Therefore a cut

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

reduces to

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

If $\mathscr{R}$ is a set of proposition rewrite rules, the reduction relation of Figure 2(b) extended by the reduction rules for $\mathscr{R}$ will be denoted $\rightarrow_{\overline{\lambda}\mu\tilde{\mu}_{\mathscr{R}}}$.

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

*Proof.* The only case worth considering is a reduction of some *supercut*

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

If $\pi$ and $e$ are the respective interpretations of $r(L, (\mu_i(\overline{\mathsf{x}}_i, \overline{\alpha}_i).X_i)_i)$ and $r(L', (\tilde{\mu}_j(\overline{y}_j, \overline{\beta}_j).Y_j)_j)$ and $c$ is a normal form of $\langle \pi \mid e \rangle$, then the supercut reduces to $c[(c_i/X_i)_i, (c'_j/Y_j)_j]$. By definition of the interpretations, the judgements $(X_i \rhd (\overline{\mathsf{x}}_i : \Gamma_i \vdash \overline{\alpha}_i : \Delta_i))_i \Vdash\vdash \pi : \varphi$ 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 $\overline{\lambda}\mu\tilde{\mu}$

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

is also well-typed in open $\overline{\lambda}\mu\tilde{\mu}$. Then a simple substitution lemma on command variables[5] proves that if the command

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

has a certain type, then so does the command $c[(c_i/X_i)_i,(c'_j/Y_j)_j]$. □

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

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

$$\begin{array}{ll} r(L,(\mu_i(\overline{x}_i,\overline{\alpha}_i).c_i)_i) & \rightarrow_e \ \pi[c_i/X_i] \\ r(L',(\tilde{\mu}_j(\overline{y}_j,\overline{\beta}_j).c'_j)_j) & \rightarrow_e \ e[c'_j/Y_j] \\ \multicolumn{2}{c}{\ldots} \end{array}$$

Such a rewrite relation is strongly normalising and confluent, therefore yielding for any extended command $c$, term $\pi$ 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 $\varphi$ and any first-order substitution $\sigma$, it must be the case that $(\varphi\downarrow_p)\sigma=(\varphi\sigma)\downarrow_p$. These six points (combined with Theorem 1) demonstrate that through $\downarrow_e$ and $\downarrow_p$, the $\overline{\lambda}\mu\tilde{\mu}$-calculus simulates our extended calculus: any well-typed reduction in our extended calculus induces through $\downarrow_e$ and $\downarrow_p$ a longer well-typed reduction in $\overline{\lambda}\mu\tilde{\mu}$. 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

$$r\ :\ P\rightarrow(\exists x_1.\forall x_2.A(x_1,x_2))\vee(\exists y_1.\forall y_2.B(y_1,y_2))$$

whose *most general* superdeduction rules are

$$\frac{\Gamma\vdash A(t,x_2),B(u,y_2),\Delta}{\Gamma\vdash P,\Delta}\left\{\begin{array}{l} x_2\notin\mathscr{FV}(\Gamma,\Delta,u) \\ y_2\notin\mathscr{FV}(\Gamma,\Delta)\end{array}\right. \quad\text{and}\quad \frac{\Gamma\vdash A(t,x_2),B(u,y_2),\Delta}{\Gamma\vdash P,\Delta}\left\{\begin{array}{l} x_2\notin\mathscr{FV}(\Gamma,\Delta) \\ y_2\notin\mathscr{FV}(\Gamma,\Delta,t)\end{array}\right. .$$

The original proofterm extension [1] transforms these two inferences into a unique proofterm $rR(\lambda x_2.\lambda y_2.(\lambda\alpha.\lambda\beta.m),t,u,\gamma)$. 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

$$\frac{m\rhd\Gamma\vdash\alpha:A(t,x_2),\beta:B(u,y_2),\Delta}{rR(\lambda x_2.\lambda y_2.(\lambda\alpha.\lambda\beta.m),t,u,\gamma)\rhd\Gamma\vdash\gamma:P,\Delta}\left\{\begin{array}{l} x_2\notin\mathscr{FV}(\Gamma,\Delta,u) \\ y_2\notin\mathscr{FV}(\Gamma,\Delta)\end{array}\right.$$

[5] not detailed here for simplicity

and

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

Let us see how this mistake is corrected in our extension of the $\overline{\lambda}\mu\tilde{\mu}$-calculus. Traces for the superdeduction inferences are respectively $\mathtt{u}!\mathtt{y}_2?\mathtt{t}!\mathtt{x}_2?$ and $\mathtt{t}!\mathtt{x}_2?\mathtt{u}!\mathtt{y}_2?$. These traces clearly specify that whether $\mathtt{t}$ is not in the scope of $\mathtt{y}_2$ or $\mathtt{u}$ is not in the scope of $\mathtt{x}_2$. Our extension of the $\overline{\lambda}\mu\tilde{\mu}$-calculus translates these superdeduction inferences into the typing rules

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

and

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

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

## 3 Conclusion

This extension of the $\overline{\lambda}\mu\tilde{\mu}$-calculus is a 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: $\overline{\lambda}\mu\tilde{\mu}$ syntax, typing and reduction is as suitable as Christian Urban's calculus for superdeduction. The extension presented in this short paper is almost a mechanical transcription of the original extension [1]. It relates superdeduction more closely to the $\lambda$-calculus based Curry-Howard correspondence without exploring any further the computational content of cut-elimination for superdeduction.

We believe that one of the key ingredients towards this goal is pattern-matching. Indeed superdeduction systems historically come from *supernatural deduction* [14], an extension of natural deduction designed to type the rewriting-calculus (*a.k.a.* $\rho$-calculus) [3]. Supernatural deduction turns proposition rewrite rules of the form

$$r \ : \ P \quad \rightarrow \quad \forall \overline{\mathrm{x}}.((A_1 \wedge A_2 \ldots A_n) \Rightarrow C)$$

into inference rules for natural deduction

$$\frac{\Gamma, A_1 \ldots A_n \vdash C}{\Gamma \vdash P} \, \overline{\mathrm{x}} \notin \mathscr{FV}(\Gamma) \qquad \text{and} \qquad \frac{\Gamma \vdash P \qquad (\Gamma \vdash A_i[\overline{\mathrm{t}}/\overline{\mathrm{x}}])_i}{\Gamma \vdash C[\overline{\mathrm{t}}/\overline{\mathrm{x}}]} \ .$$

(The first rule is an introduction rule and the second is an elimination rule.) The rewriting calculus is an extension of the $\lambda$-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 $r(\overline{\mathrm{x}}, x_1 \ldots x_n)$. The introduction rule types an abstraction on this pattern (*i.e.* a rewrite rule)

$$\frac{\Gamma, x_1 : A_1 \ldots x_n : A_n \vdash \pi : C}{\Gamma \vdash r(\overline{\mathrm{x}}, x_1 \ldots x_n) \rightarrow \pi : P} \, \overline{\mathrm{x}} \notin \mathscr{FV}(\Gamma) \ .$$

Dually the elimination rule types an application on this pattern

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

Supernatural deduction systems (in intuitionistic 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 [15]. Finally we 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 & Claude Kirchner (2007): *Principles of Superdeduction*. In: *LICS*, pp. 41–50.

[2] Serenella Cerrito & Delia Kesner (2004): *Pattern matching as cut elimination*. *Theor. Comput. Sci.* 323(1-3), pp. 71–127.

[3] Horatiu Cirstea & Claude Kirchner (2001): *The rewriting calculus –- Part I and II*. *Logic Journal of the Interest Group in Pure and Applied Logics* 9(3), pp. 427–498.

[4] Pierre-Louis Curien & Hugo Herbelin (2000): *The duality of computation*. In: *ICFP*, pp. 233–243.

[5] Pierre-Louis Curien & Guillaume Munch-Maccagnoni (2010): *The duality of computation under focus*. In: *IFIP TCS*. Accepted.

[6] Gilles Dowek, Thérèse Hardin & Claude Kirchner (2003): *Theorem Proving Modulo*. *Journal of Automated Reasoning* 31(1), pp. 33–72.

[7] Clément Houtmann (2008): *Axiom Directed Focusing*. In: *TYPES*, pp. 169–185.

[8] Clément Houtmann (2010): *Représentation et interaction des preuves en superdéduction modulo*. Ph.D. thesis, Université Henri Poincaré, Nancy Universités.

[9] Chuck Liang & Dale Miller (2007): *Focusing and Polarization in Intuitionistic Logic*. In: *CSL*, pp. 451–465.

[10] Chuck Liang & Dale Miller (2009): *A Unified Sequent Calculus for Focused Proofs*. In: *LICS*, IEEE Computer Society, pp. 355–364.

[11] Guillaume Munch-Maccagnoni (2009): *Focalisation and Classical Realisability*. In: *CSL*, pp. 409–423.

[12] Emmanuel Polonowski (2004): *Strong Normalization of lambda-mu-mu/tilde-Calculus with Explicit Substitutions*. In: *FoSSaCS*, pp. 423–437.

[13] Christian Urban (2000): *Classical Logic and Computation*. Ph.D. thesis, University of Cambridge.

[14] Benjamin Wack (2005): *Typage et déduction dans le calcul de réécriture*. Ph.D. thesis, Université Henri Poincaré, Nancy 1.

[15] Noam Zeilberger (2008): *Focusing and higher-order abstract syntax*. In: *POPL*, pp. 359–369.