Principles of Superdeduction

Paul Brauner, Clement Houtmann, Claude Kirchner

Principles of Superdeduction

Paul Brauner
INPL & LORIA

Clément Houtmann
ENS de Cachan & LORIA

Claude Kirchner
INRIA & LORIA

LORIA,* Campus Scientifique, BP 239
54506 Vandoeuvre-lès-Nancy Cedex, France

Abstract

In predicate logic, the proof that a theorem P holds in a theory $\mathcal{T}h$ is typically conducted in natural deduction or in the sequent calculus using all the information contained in the theory in a uniform way. Introduced ten years ago, Deduction modulo allows us to make use of the computational part of the theory $\mathcal{T}h$ for true computations modulo which deductions are performed.

Focusing on the sequent calculus, this paper presents and studies the dual concept where the theory is used to enrich the deduction system with new deduction rules in a systematic, correct and complete way. We call such a new deduction system "superdeduction".

We introduce a proof-term language and a cut-elimination procedure both based on Christian Urban's work on classical sequent calculus. Strong normalisation is proven under appropriate and natural hypothesis, therefore ensuring the consistency of the embedded theory and of the deduction system.

The proofs obtained in such a new system are much closer to the human intuition and practice. We consequently sketch how superdeduction along with deduction modulo can be used to ground the formal foundations of new extendible proof assistants like lemuridæ*, our prototypal implementation of superdeduction modulo.*

1. Introduction

Formal proofs are central objects in mathematics as well as informatics. For instance, security and safety assessments are backed by the common criteria setting up levels of certification where the highest ones require formal proofs to be constructed, communicated and independently verifiable or replayable. In a given context formalised by a set of axioms, one has to build "good" proofs for some propositions that describe mathematical properties (*e.g.* the four colours theorem) or safety or security properties (*e.g.* reachability properties). This proof engineering process is now well mastered with the use of proof assistants like Coq, Isabelle, PVS, HOL, Mizar and large libraries of formalised theories ease this task.

In this context one has to deal with at least two difficulties. First, the theories describing the context become huge and may consist of thousand of axioms and definitions, some of them being quite sophisticated. Second, the proof assistant needs to provide the user with the best way to understand and to guide the proof. Both concerns are currently tackled by making libraries available, by providing specific tactics, tacticals or strategies (see typically `coq.inria.fr`), by integrating decision procedures safely into the proof assistants [19], or by interfacing first-order automated theorem provers with proof assistants like [3] or like the use of Zenon in Focal [22].

Indeed these approaches raise the question of structuring the theories of interest. For instance one would like to identify the subtheory of lists or of naturals to apply specific decision procedures and of course finding a good modular structure is one of the first steps in an engineering process.

In this context, we propose a foundational framework making use of three complementary dimensions. First, as pioneered by deduction modulo, the computational axioms should be identified. Typically the definition of addition on naturals ought to be embedded into a congruence modulo which deduction is performed [12]. In this case, the deduction rules like the one of natural deduction or of the sequent calculus are not modified but they are applied modulo a congruence embedding part of the theory. Second, we are proposing a complementary approach where *new deduction rules* are inferred from part of the theory in a correct, systematic and complete way. Third, the rest of the theory will be used as the context on which all the standard and new deduction rules will act, modulo some congruence.

To sum up, a theory is split in three parts $\mathcal{T}h = \mathcal{T}h_1 \cup \mathcal{T}h_2 \cup \mathcal{T}h_3$ and instead of seeking for a proof of $\mathcal{T}h_1 \cup \mathcal{T}h_2 \cup \mathcal{T}h_3 \vdash \varphi$, we are building a proof of $\mathcal{T}h_3 \vdash^{+_{\mathcal{T}h_2}}_{\sim_{\mathcal{T}h_1}} \varphi$, *i.e.* we use the theory $\mathcal{T}h_3$ to prove φ using the extended deduction

*UMR 7503 CNRS-INPL-INRIA-Nancy2-UHP

system modulo the congruence $\sim_{\mathcal{T}h_1}$. We assume that the propositions in $\mathcal{T}h_2$ are all proposition rewrite rules, *i.e.* are of the form $\forall\overline{x}.(P \Leftrightarrow \varphi)$, where P is atomic.

To ease the presentation of the main ideas, we will not consider in this paper the case of deduction modulo even if in addition to simplicity it admits unbounded proof size speed-up [6]. We call *superdeduction* the new deduction system embedding the newly generated deduction rules, and the extended entailment relation is denoted $\vdash^{+_{\mathcal{T}h}}$ or simply $\vdash^{+}$.

Intuitively, a superdeduction rule supplants the *folding* of an atomic proposition P by its definition φ, as done by Prawitz [21], followed by as much introductions as possible of the connectives appearing in φ. For instance, the axiom

$$\textsc{Inc} \ : \ \forall X.\forall Y.(X \subseteq Y \Leftrightarrow \forall x.(x \in X \Rightarrow x \in Y))$$

is translated into a right deduction rule by first applying the rules of the classical sequent calculus to $\Gamma \vdash \forall x.(x \in X \Rightarrow x \in Y), \Delta$:

$$\forall_R \ \dfrac{\Rightarrow_R \ \dfrac{\Gamma, x \in X \vdash x \in Y, \Delta}{\Gamma \vdash x \in X \Rightarrow x \in Y, \Delta}}{\Gamma \vdash \forall x.(x \in X \Rightarrow x \in Y), \Delta} \ x \notin \mathcal{FV}(\Gamma, \Delta)$$

then by collecting the premises and the side conditions, we get the *new* deduction rule:

$$\textsc{Inc}_R \ \dfrac{\Gamma, x \in X \vdash^{+} x \in Y, \Delta}{\Gamma \vdash^{+} X \subseteq Y, \Delta} \ x \notin \mathcal{FV}(\Gamma, \Delta)$$

The left rule is similarly obtained by applying deduction rules to $\Gamma, \forall x.(x \in X \Rightarrow x \in Y) \vdash \Delta$.

$$\textsc{Inc}_L \ \dfrac{\Gamma \vdash^{+} t \in X, \Delta \qquad \Gamma, t \in Y \vdash^{+} \Delta}{\Gamma, X \subseteq Y \vdash^{+} \Delta}$$

These new deduction rules are quite natural and translate the usual mathematical reasoning *w.r.t.* this axiom. For instance, the right rule can be read as "if any element of X is an element of Y, then $X \subseteq Y$". Let us see on a simple example the difference between a proof in sequent calculus and the corresponding one in the extended deduction system. The proof that $\textsc{Inc} \vdash A \subseteq A$ is the following:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\textsc{Ax}\ \dfrac{}{\ldots, x \in A \vdash A \subseteq A, x \in A}}{\Rightarrow_R \ \dfrac{}{\ldots \vdash A \subseteq A, x \in A \Rightarrow x \in A}}}{\forall_R \ \dfrac{}{\ldots \vdash A \subseteq A, \forall x.(x \in A \Rightarrow x \in A)}} \qquad \vdots \qquad \textsc{Ax}\ \dfrac{}{\ldots, A \subseteq A \vdash A \subseteq A}}{\Rightarrow_L \ \dfrac{}{\ldots, (\forall x.(x \in A \Rightarrow x \in A)) \Rightarrow A \subseteq A \vdash A \subseteq A}}}{\wedge_L \ \dfrac{}{(A \subseteq A) \Leftrightarrow \forall x.(x \in A \Rightarrow x \in A) \vdash A \subseteq A}}}{\forall_L \ \dfrac{}{\forall Y.(A \subseteq Y) \Leftrightarrow \forall x.(x \in A \Rightarrow x \in Y) \vdash A \subseteq A}}}{\forall_L \ \dfrac{}{\textsc{Inc} \vdash A \subseteq A}}$$

In the superdeduction system, the axiom INC is used to generate a new deduction rule and the proof becomes simply:

$$\textsc{Inc}_R \ \dfrac{\textsc{Ax}\ \dfrac{}{x \in A \vdash^{+_{\textsc{Inc}}} x \in A}}{\vdash^{+_{\textsc{Inc}}} A \subseteq A}$$

These new rules are not just "macros" collapsing a sequence of introductions into a single one: they apply to a predicate, not a connector, and therefore do not solely contain purely logical information. This therefore raises non trivial questions solved in this paper, like the conditions under which the system is complete or consistent, and sufficient conditions to get cut-elimination.

There are other approaches to extend deduction systems by adding new inference rules, particularly related to logic programming. Let us mention Definitional Reflection [16], extended by McDowell and Miller in [18] with inference rules for induction. Another interesting point of view is Huang's Assertion level [17] especially motivated by the presentation of machine found proofs in natural language.

Superdeduction is based on previous works on supernatural deduction, a deduction system introduced by Benjamin Wack in [27] and providing a logical interpretation of the rewriting calculus [7, 8].

In this context, our contributions are the following:

- We define in a systematic way the extension of the classical sequent calculus by new deduction rules inferred from the axioms of the theory that are proposition rewrite rules; We prove that this is correct and complete taking into account permutability problems; This is described in the next section.
- Building on Urban's proof-term language for the sequent calculus [23], we propose a simple and expressive calculus that we show to provide a Curry-Howard-de Bruijn correspondence for superdeduction; Assuming the proposition rewrite system used to extend deduction to be weakly normalising and confluent, we show in Section 3 that the calculus is strongly normalising and therefore that the theory is consistent and that the superdeduction system has the cut-elimination property.
- Last, we investigate in Section 4 the consequence of these results for the foundation of a new generation of proof assistants for which we have a first downloadable prototype, lemuridæ (`rho.loria.fr`).

The proof of the main results together with detailled experimentations can be found in [5].

2. Super sequent calculus

As mentioned in the introduction and as for deduction modulo, we focus our attention to formulæ of the form $\forall\overline{x}.(P \Leftrightarrow \varphi)$ where P is atomic:

Definition 2.1 (Propositions rewrite rule)**.** *We denote* $\mathsf{R} : P \to \varphi$ *the axiom* $\forall \overline{x}.(P \Leftrightarrow \varphi)$ *where* R *is its name,* P *is an atomic proposition,* φ *some proposition and* $\overline{x}$ *their free variables.*

Notice that P may contain first-order terms and therefore that such an axiom is not just a definition. For instance, $isZero(succ(n)) \to \bot$ is a proposition rewrite rule.

Let us recall the classical sequent calculus:

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

$$\bot_L\ \frac{}{\Gamma, \bot \vdash \Delta} \qquad \top_R\ \frac{}{\Gamma \vdash \top, \Delta}$$

$$\wedge_L\ \frac{\Gamma, \varphi_1, \varphi_2 \vdash \Delta}{\Gamma, \varphi_1 \wedge \varphi_2 \vdash \Delta} \qquad \wedge_R\ \frac{\Gamma \vdash \varphi_1, \Delta \qquad \Gamma \vdash \varphi_2, \Delta}{\Gamma \vdash \varphi_1 \wedge \varphi_2, \Delta}$$

$$\vee_L\ \frac{\Gamma, \varphi_1 \vdash \Delta \qquad \Gamma, \varphi_2 \vdash \Delta}{\Gamma, \varphi_1 \vee \varphi_2 \vdash \Delta} \qquad \vee_R\ \frac{\Gamma \vdash \varphi_1, \varphi_2, \Delta}{\Gamma \vdash \varphi_1 \vee \varphi_2, \Delta}$$

$$\Rightarrow_R\ \frac{\Gamma, \varphi_1 \vdash \varphi_2, \Delta}{\Gamma \vdash \varphi_1 \Rightarrow \varphi_2, \Delta} \qquad \Rightarrow_L\ \frac{\Gamma \vdash \varphi_1, \Delta \qquad \Gamma, \varphi_2 \vdash \Delta}{\Gamma, \varphi_1 \Rightarrow \varphi_2 \vdash \Delta}$$

$$\forall_R\ \frac{\Gamma \vdash \varphi, \Delta}{\Gamma \vdash \forall x.\varphi, \Delta}\ x \notin \mathcal{FV}(\Gamma, \Delta) \qquad \forall_L\ \frac{\Gamma, \varphi[t/x] \vdash \Delta}{\Gamma, \forall x.\varphi \vdash \Delta}$$

$$\exists_R\ \frac{\Gamma \vdash \varphi[t/x], \Delta}{\Gamma \vdash \exists x.\varphi, \Delta} \qquad \exists_L\ \frac{\Gamma, \varphi \vdash \Delta}{\Gamma, \exists x.\varphi \vdash \Delta}\ x \notin \mathcal{FV}(\Gamma, \Delta)$$

$$\textsc{Contr}_R\ \frac{\Gamma \vdash \varphi, \varphi, \Delta}{\Gamma \vdash \varphi, \Delta} \qquad \textsc{Contr}_L\ \frac{\Gamma, \varphi, \varphi \vdash \Delta}{\Gamma, \varphi \vdash \Delta}$$

For the classical sequent calculus, let us now describe how the computation of the superdeduction new inference rules is performed.

Definition 2.2 (Super sequent calculus rules computation)**.** *Let* $\mathcal{C}alc$ *be a set of rules composed by the subset of the sequent calculus deduction rules formed of* Ax, $\bot_L$, $\top_R$, $\vee_L$, $\vee_R$, $\wedge_L$, $\wedge_R$, $\Rightarrow_L$, $\Rightarrow_R$, $\forall_L$, $\forall_R$, $\exists_L$ *and* $\exists_R$*, as well as of the two following rules* $\top_L$ *and* $\bot_R$*:*

$$\top_L\ \frac{\Gamma \vdash \Delta}{\Gamma, \top \vdash \Delta} \qquad \bot_R\ \frac{\Gamma \vdash \Delta}{\Gamma \vdash \bot, \Delta}$$

Let $\mathsf{R} : P \to \varphi$ *be a proposition rewrite rule.*

1. *To get the right rule associated with* R*, initialise the procedure with the sequent* $\Gamma \vdash \varphi, \Delta$*. Next, apply the rules of* $\mathcal{C}alc$ *until no more open leaves remain on which they can be applied. Then, collect the premises, the side conditions and the conclusion and replace* φ *by* P *to obtain the right rule* R_R*.*

2. *To get the left rule* R_L *associated with* R*, initialise the procedure with the sequent* $\Gamma, \varphi \vdash \Delta$*. apply the rules of* $\mathcal{C}alc$ *and get the new left rule the same way as for the right one.*

Definition 2.3 (Super sequent calculus)**.** *Given a proposition rewrite system* $\mathcal{R}$*, the super sequent calculus associated with* $\mathcal{R}$ *is formed of the rules of classical sequent calculus and the rules built upon* $\mathcal{R}$*. The sequents in such a system are written* $\Gamma \vdash^{+_\mathcal{R}} \Delta$*.*

To ensure good properties of the system, we need to put some restrictions on the axioms though. Although the deduction rules of the classical sequent calculus propositional fragment may be applied in any order to reach axioms, the application order of rules concerning quantifiers is significant. Let us consider the following cases:

$$\textsc{Ax}\ \frac{}{P(x_0) \vdash P(x_0)}$$
$$\forall_L\ \frac{P(x_0) \vdash P(x_0)}{\forall x.P(x) \vdash P(x_0)}$$
$$\forall_R\ \frac{\forall x.P(x) \vdash P(x_0)}{\forall x.P(x) \vdash \forall x.P(x)} \qquad \forall_L\ \frac{P(t) \vdash \forall x.P(x)}{\forall x.P(x) \vdash \forall x.P(x)}$$

The left-hand side proof succeeds because the early application of the $\forall_R$ rule provides the appropriate term for instantiating the variable of the proposition present in the context. On the other hand, the second proof cannot be completed since the $\forall_R$ side condition requires the quantified variable to be substituted for a fresh one. Such a situation may occur when building the super sequent calculus custom rules and therefore may break its completeness *w.r.t.* classical predicate logic. This common permutability problem of automated proof search appears here since superdeduction systems are in fact embedding a part of compiled automated deduction. Thereby we apply an idea inspired by focusing techniques [1], namely replacing every subformula of φ leading to a permutability problem by a fresh predicate symbol parameterised by the free variables of the subformula. To formalise this, let us recall the polarity notion:

Definition 2.4 (Polarity of a subformula)**.** *The polarity* $pol_\varphi(\psi)$ *of* ψ *in* φ *where* ψ *is an occurrence of a subformula of* φ *is a boolean defined as follows:*

- *if* $\varphi = \psi$*, then* $pol_\varphi(\psi) = 1$*;*
- *if* $\varphi = \varphi_1 \wedge \varphi_2$ *or* $\varphi_1 \vee \varphi_2$*, then* $pol_\varphi(\psi) = pol_{\varphi_1}(\psi)$ *if* ψ *is a subformula of* φ_1*,* $pol_{\varphi_2}(\psi)$ *otherwise;*
- *if* $\varphi = \forall x.\varphi_1$ *or* $\exists x.\varphi_1$*, then* $pol_\varphi(\psi) = pol_{\varphi_1}(\psi)$*;*
- *if* $\varphi = \varphi_1 \Rightarrow \varphi_2$*, then* $pol_\varphi(\psi) = \neg pol_{\varphi_1}(\psi)$ *if* ψ *is a subformula of* φ_1*,* $pol_{\varphi_2}(\psi)$ *otherwise.*

Definition 2.5 (Set of permutability problems)**.** *A formula* ψ *is in the set* $PP(\varphi)$ *of* φ *permutability problems if there exists* φ' *a subformula of* φ *such that* ψ *is an occurrence of a subformula of* φ' *and one of these propositions holds:*

- $\varphi' = \forall x.\varphi'_1$*,* $\psi = \forall x.\psi'_1$ *and* $pol_{\varphi'}(\psi) = 0$
- $\varphi' = \exists x.\varphi'_1$*,* $\psi = \exists x.\psi'_1$ *and* $pol_{\varphi'}(\psi) = 0$
- $\varphi' = \forall x.\varphi'_1$*,* $\psi = \exists x.\psi'_1$ *and* $pol_{\varphi'}(\psi) = 1$
- $\varphi' = \exists x.\varphi'_1$*,* $\psi = \forall x.\psi'_1$ *and* $pol_{\varphi'}(\psi) = 1$

This allows us to define the most appropriate generalisation of a proposition rewrite rule $\mathsf{R} : P \to \varphi$:

Definition 2.6 (Set of delayed proposition rewrite rules)**.** $Dl(\mathsf{R} : P \to \varphi) =$
$\{P \to C\big[Q_1(\overline{x_1}), \ldots, Q_n(\overline{x_n})\big]\} \bigcup_{i=1\ldots n} Dl\big(Q_i \to \varphi_i\big)$
such that:

- *C is the largest context in φ with no formula in $PP(\varphi)$ such that $\varphi = C[\varphi_1 \ldots \varphi_n]$;*
- *$\forall i \in \{1 \ldots n\}$, $\overline{x_i}$ is the vector of φ_i free variables;*
- *$Q_1 \ldots Q_n$ are fresh predicate symbols.*

As an example, let us consider the proposition rewrite rule defining the natural numbers as the set of terms verifying the inductive predicate $\in_{\mathbb{N}} : \mathbb{N}(n) \to$

$$\forall P.(0 \in P \Rightarrow \forall m.(m \in P \Rightarrow S(m) \in P) \Rightarrow n \in P)$$

This axiom can be found in [15] which introduces an axiomatisation of constructive arithmetic with rewrite rules only. It uses a simple second-order encoding by expressing quantification over propositions by quantification over classes; $x \in P$ should therefore be read as $P(x)$. The delayed set $Dl(\in_{\mathbb{N}})$ of proposition rewrite rules derived from the rules above is:

$$\begin{array}{rcl} \in_{\mathbb{N}} : \mathbb{N}(n) & \to & \forall P.(0 \in P \Rightarrow H(P) \Rightarrow n \in P) \\ hered : H(P) & \to & \forall m.(m \in P \Rightarrow S(m) \in P) \end{array}$$

Let us notice that the proposition $H(P)$ revealed by the elimination of permutability problems expresses heredity, a well-known notion. Focusing on parts of the propositions which raise some non-trivial choice at some phase on the proof has been naturally done by mathematicians. Then we obtain the following deduction rules for the natural numbers definition:

$$\in_{\mathbb{N}_L} \frac{\Gamma \vdash^+ 0 \in P, \Delta \quad \Gamma \vdash^+ H(P), \Delta \quad \Gamma, n \in P \vdash^+ \Delta}{\Gamma, \mathbb{N}(n) \vdash^+ \Delta}$$

$$\in_{\mathbb{N}_R} \frac{0 \in P, H(P) \vdash^+ n \in P, \Delta}{\Gamma \vdash^+ \mathbb{N}(n), \Delta}\ P \notin \mathcal{FV}(\Gamma, \Delta)$$

The left rule translates exactly the usual induction rule. The $hered$ proposition rewrite rule generates new deduction rules too:

$$hered_L \frac{\Gamma \vdash^+ m \in P, \Delta \qquad \Gamma, S(m) \in P \vdash^+ \Delta}{\Gamma, H(P) \vdash^+ \Delta}$$

$$hered_R \frac{\Gamma, m \in P \vdash^+ S(m) \in P, \Delta}{\Gamma \vdash^+ H(P), \Delta}\ m \notin \mathcal{FV}(\Gamma, \Delta)$$

Once again, the right rule corresponds to the usual semantics of heredity.

Main properties of the super sequent calculus associated with a delayed set of axioms are its soundness and completeness *w.r.t.* classical predicate logic.

Theorem 2.1 (Soundness and completeness of super sequent calculus)**.** *Given $\mathcal{T}h$ an axiomatic theory made of axioms of the form $\forall \overline{x}.(P \Leftrightarrow \varphi)$ with P atomic and $\mathcal{R}$ the associated proposition rewrite rules, every proof of $\Gamma \vdash_{Dl(\mathcal{R})} \Delta$ in super sequent calculus can be translated into a proof of $\Gamma, \mathcal{T}h \vdash \Delta$ in sequent calculus (soundness) and conversely (completeness).*

The proof is given in [5].

3. Curry-Howard-de Bruijn correspondence and cut-elimination

The relation between classical logic and computation has been lately explored in various ways. Recent attempts to find a Curry-Howard-de Bruijn correspondence for classical sequent calculus are concentrated in Herbelin's and in Urban's works [10, 24, 23]. They ground two families of proof-term languages for classical sequent calculus.

On one hand, the $\bar{\lambda}\mu\tilde{\mu}$-calculus presented in [10] and inspired from the $\lambda\mu$-calculus [20], proposes to *focus* on one formula in each sequent. Thus the type of any well-typed proof-term is the formula focused in the corresponding sequent. Two cut-elimination strategies are exhibited corresponding respectively to call-by-value and call-by-name which are proven to be De Morgan dual.

On the other hand Urban's proof-term language presented in [24] is constructed from the opposite point of view. Instead of switching step by step from the λ-calculus for intuitionistic natural deduction to the $\lambda\mu$-calculus for classical natural deduction and finally to the $\bar{\lambda}\mu\tilde{\mu}$-calculus for classical sequent calculus, the starting point is the classical sequent calculus for which constructors are written, representing exactly each deduction rule. A large part of the work consists then in comparing the various possibilities for a cut-elimination procedure, such as Gentzen's original procedure. In particular Urban proposes two cut-elimination procedures and proves their strong normalisation, amongst other appropriate properties. The first is exposed in [24] while the second, presented as *Gentzen-like*, is introduced in [23]. The implicational fragment of Urban's proof-term language is named $\mathcal{X}$ and reconsidered in [25] in an untyped setting. Its expressive power is demonstrated in particular by interpretations for the λ-calculus, the $\lambda\mu$-calculus and Bloo and Rose's calculus $\lambda\mathbf{x}$.

To build an appropriate proof-term language for superdeduction, what is the best language to start from? To explain our choice, we need to take into account that the computation of the new deduction rules may contain changes of focus such as:

$$\Rightarrow_R \frac{\vee_L \dfrac{\varphi_1, \varphi_3 \vdash \varphi_4 \qquad \varphi_2, \varphi_3 \vdash \varphi_4}{\varphi_1 \vee \varphi_2, \varphi_3 \vdash \varphi_4}}{\varphi_1 \vee \varphi_2 \vdash \varphi_3 \Rightarrow \varphi_4}$$

expressed in $\bar{\lambda}\mu\tilde{\mu}$-calculus with the following underlined focuses:

$$\Rightarrow_R \dfrac{\text{FOCUS}\ \dfrac{\vee_L \dfrac{\varphi_3, \underline{\varphi_1} \vdash \varphi_4 \qquad \varphi_3, \underline{\varphi_2} \vdash \varphi_4}{\varphi_3, \underline{\varphi_1 \vee \varphi_2} \vdash \varphi_4}}{\varphi_1 \vee \varphi_2, \varphi_3 \vdash \underline{\varphi_4}}}{\varphi_1 \vee \varphi_2 \vdash \underline{\varphi_3 \Rightarrow \varphi_4}}$$

These hidden steps in sequent calculus are explicit in $\bar{\lambda}\mu\tilde{\mu}$-calculus: constructors μ and $\tilde{\mu}$ play indeed the role of *focusers*. Since the construction of new inference rules in superdeduction obviously contains focus steps which are hidden in the final deduction rule, they should be hidden in the new proof-term we are seeking for. This important difference between $\bar{\lambda}\mu\tilde{\mu}$-calculus and Urban's proof-term language conduces us to choose Urban's framework as the base of our proof-term language for superdeduction. However slight changes are made to the original language to make it correspond with our version of classical sequent calculus (in Section 2). Our specific version of Urban's calculus is depicted in the following subsection.

Since no focus is made on a particular formula of a sequent $\Gamma \vdash \Delta$ in Urban's calculus, a proof-term M will always annotate the full sequent. Such typing judgements will be denoted $M \rhd \Gamma \vdash \Delta$.

3.1. Urban's calculus

Urban's proof-term language for classical sequent calculus makes no use of the first-class objects of the λ-calculus such as *abstractions* or *variables*. Variables are replaced by *names* and *conames*. Let X and A be respectively the set of *names* and the set of *conames*. Symbols $x, y, \ldots$ will range over X while symbols $a, b, \ldots$ will range over A. Symbols $\mathsf{x}, \mathsf{y}, \ldots$ will range over the set of first-order variables. Left-contexts and right-contexts are sets containing respectively pairs $x : \varphi$ and pairs $a : \varphi$. Symbol Γ will range over left-contexts and symbol Δ will range over the right-contexts. Moreover, contexts cannot contain more than one occurrence of a name or coname. We will never omit the 'first-order' in 'first-order term' in order to avoid confusion with 'terms' (*i.e.* proof-terms). The set of terms is defined as follows.

$$\begin{aligned} M, N ::= {} & \mathsf{Ax}(x,a) \mid \mathsf{Cut}(\widehat{a}M, \widehat{x}N) \mid \mathsf{False}_L(x) \mid \mathsf{True}_R(a) \\ & \mid \mathsf{And}_R(\widehat{a}M, \widehat{b}N, c) \mid \mathsf{And}_L(\widehat{x}\widehat{y}M, z) \mid \mathsf{Or}_R(\widehat{a}\widehat{b}M, c) \\ & \mid \mathsf{Imp}_R(\widehat{x}\widehat{a}M, b) \mid \mathsf{Imp}_L(\widehat{x}M, \widehat{a}N, y) \mid \mathsf{Or}_L(\widehat{x}M, \widehat{y}N, z) \\ & \mid \mathsf{Exists}_R(\widehat{a}M, t, b) \mid \mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}M, y) \\ & \mid \mathsf{Forall}_R(\widehat{a}\widehat{\mathsf{x}}M, b) \mid \mathsf{Forall}_L(\widehat{x}M, t, y) \end{aligned}$$

Names and conames are not called *variables* and *covariables* such as in $\bar{\lambda}\mu\tilde{\mu}$-calculus since they do not represent places where terms might be inserted. They still may appear bound: the symbol «$\widehat{\ }$» is the unique binder of the calculus and thus we can compute the sets of free and bound names, conames and first-order variables in any term. We consequently adopt Barendregt's convention on names, conames and first-order variables: in a term or in a statement a name, a coname or a first-order variable is never both bound and free in the same context.

The type system is the following:

$$\text{Ax}\ \frac{}{\mathsf{Ax}(x,a) \rhd \Gamma, x : \varphi \vdash a : \varphi, \Delta}$$

$$\text{Cut}\ \frac{M \rhd \Gamma \vdash a : \varphi, \Delta \qquad N \rhd \Gamma, x : \varphi \vdash \Delta}{\mathsf{Cut}(\widehat{a}M, \widehat{x}N) \rhd \Gamma \vdash \Delta}$$

$$\bot_L\ \frac{}{\mathsf{False}_L(x) \rhd \Gamma, x : \bot \vdash \Delta}$$

$$\top_R\ \frac{}{\mathsf{True}_R(a) \rhd \Gamma \vdash a : \top, \Delta}$$

$$\wedge_R\ \frac{M \rhd \Gamma \vdash a : \varphi_1, \Delta \qquad N \rhd \Gamma \vdash b : \varphi_2, \Delta}{\mathsf{And}_R(\widehat{a}M, \widehat{b}N, c) \rhd \Gamma \vdash c : \varphi_1 \wedge \varphi_2, \Delta}$$

$$\wedge_L\ \frac{M \rhd \Gamma, x : \varphi_1, y : \varphi_2 \vdash \Delta}{\mathsf{And}_L(\widehat{x}\widehat{y}M, z) \rhd \Gamma, z : \varphi_1 \wedge \varphi_2 \vdash \Delta}$$

$$\vee_R\ \frac{M \rhd \Gamma \vdash a : \varphi_1, b : \varphi_2, \Delta}{\mathsf{Or}_R(\widehat{a}\widehat{b}M, c) \rhd \Gamma \vdash c : \varphi_1 \vee \varphi_2, \Delta}$$

$$\vee_L\ \frac{M \rhd \Gamma, x : \varphi_1 \vdash \Delta \qquad N \rhd \Gamma, y : \varphi_2 \vdash \Delta}{\mathsf{Or}_L(\widehat{x}M, \widehat{y}N, z) \rhd \Gamma, z : \varphi_1 \vee \varphi_2 \vdash \Delta}$$

$$\Rightarrow_R\ \frac{M \rhd \Gamma, x : \varphi_1 \vdash a : \varphi_2, \Delta}{\mathsf{Imp}_R(\widehat{x}\widehat{a}M, b) \rhd \Gamma \vdash b : \varphi_1 \Rightarrow \varphi_2, \Delta}$$

$$\Rightarrow_L\ \frac{M \rhd \Gamma, x : \varphi_2 \vdash \Delta \qquad N \rhd \Gamma \vdash a : \varphi_1, \Delta}{\mathsf{Imp}_L(\widehat{x}M, \widehat{a}N, y) \rhd \Gamma, y : \varphi_1 \Rightarrow \varphi_2 \vdash \Delta}$$

$$\exists_R\ \frac{M \rhd \Gamma \vdash a : \varphi[\mathsf{x} := t], \Delta}{\mathsf{Exists}_R(\widehat{a}M, t, b) \rhd \Gamma \vdash b : \exists \mathsf{x}.\varphi, \Delta}$$

$$\exists_L\ \frac{M \rhd \Gamma, x : \varphi \vdash \Delta}{\mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}M, y) \rhd \Gamma, y : \exists \mathsf{x}.\varphi \vdash \Delta}\ \mathsf{x} \notin \mathcal{FV}(\Gamma, \Delta)$$

$$\forall_R\ \frac{M \rhd \Gamma \vdash a : \varphi, \Delta}{\mathsf{Forall}_R(\widehat{a}\widehat{\mathsf{x}}M, b) \rhd \Gamma \vdash b : \forall \mathsf{x}.\varphi, \Delta}\ \mathsf{x} \notin \mathcal{FV}(\Gamma, \Delta)$$

$$\forall_L\ \frac{M \rhd \Gamma, x : \varphi[\mathsf{x} := t] \vdash \Delta}{\mathsf{Forall}_L(\widehat{x}M, t, y) \rhd \Gamma, y : \forall \mathsf{x}.\varphi \vdash \Delta}$$

The differences with Urban's type system is the use of

$$\vee_R\ \frac{\Gamma \vdash \varphi_1, \varphi_2, \Delta}{\Gamma \vdash \varphi_1 \vee \varphi_2, \Delta} \quad \text{instead of} \quad \vee_R\text{-}i\ \frac{\Gamma \vdash \varphi_i, \Delta}{\Gamma \vdash \varphi_1 \vee \varphi_2, \Delta}$$

for $i \in \{1, 2\}$ and similarly for $\wedge$.

A comma in a conclusion stands for the set union and a comma in a premise stands for the *disjoint* set union. This allows our type inference rules to contain *implicit* contraction.

A term M *introduces* the name z if it is of the form $\mathsf{Ax}(z,a)$, $\mathsf{False}_L(z)$, $\mathsf{And}_L(\widehat{x}\widehat{y}M, z), \mathsf{Or}_L(\widehat{x}M, \widehat{y}N, z)$, $\mathsf{Imp}_L(\widehat{x}M, \widehat{a}N, z)$, $\mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}M, z)$, $\mathsf{Forall}_L(\widehat{x}M, t, z)$, and it *introduces* the coname c is it is of the form

$\mathsf{Ax}(x,c)$, $\mathsf{True}_R(c)$, $\mathsf{And}_R(\widehat{a}M,\widehat{b}N,c)$, $\mathsf{Or}_R(\widehat{a}\widehat{b}M,c)$, $\mathsf{Imp}_R(\widehat{x}\widehat{a}M,c)$, $\mathsf{Exists}_R(\widehat{a}M,t,c)$, $\mathsf{Forall}_R(\widehat{a}\widehat{\mathrm{x}}M,c)$. A term M *freshly* introduces a name or a coname if it introduces it, but none of its proper subterms. It means that the corresponding formula is introduced at the top-level of the proof, but not implicitly contracted and consequently introduced in some subproof.

In [24], a (non-confluent) cut-elimination procedure denoted $\xrightarrow{\mathsf{cut}}$ is presented and proven to be strongly normalising on well-typed terms. It is *complete* in the sense that irreducible terms are cut-free.

3.2. Proof-terms for superdeduction

During the computation of the deduction rules for some proposition rewrite rule, the procedure computes an *open* derivation where two kinds of information still need to be provided: (1) premises that remain to be proved and (2) first-order terms written at a metalevel by rules $\exists_R$ and $\forall_L$ that still remain to be instantiated.

In order to represent these, we use a formal notion of *open-terms*: terms that contains (1) open leaves that represent premises that remain to be proved and are denoted $\Box$, and (2) placeholders for first-order terms that represent uninstantiated first-order terms and are denoted by α, β, ... Substitutions over placeholder-terms are written $[\alpha := t, \dots]$ and are defined over first-order terms, formulæ, sequents, and terms. The syntax of open-terms is then:

$$C, D ::= \Box \rhd \Gamma \vdash \Delta \mid \mathsf{Ax}(x,a) \mid \mathsf{Cut}(\widehat{a}C, \widehat{x}D) \mid \dots$$
$$\mid \mathsf{Exists}_R(\widehat{a}C,\alpha,b) \mid \mathsf{Exists}_L(\widehat{x}\widehat{\mathrm{x}}C,y)$$
$$\mid \mathsf{Forall}_R(\widehat{a}\widehat{\mathrm{x}}C,b) \mid \mathsf{Forall}_L(\widehat{x}C,\alpha,y)$$

Urban's cut-elimination procedure is extended to open-terms in the obvious way. Typing is also extended to open-terms by adding the rule $\dfrac{(\Box \rhd \Gamma \vdash \Delta) \rhd \Gamma \vdash \Delta}{}$ to the type system. These leaves will be denoted for short $\dfrac{}{\Box \rhd \Gamma \vdash \Delta}$. Type inference derivation for open-terms are called open type inference derivations. Their *open leaves* are the later leaves, *i.e.* the *open leaves* of the open-term. For some open-term C, its number of occurrences of $\Box$ is denoted n_C. Then for some placeholder-term substitution $\sigma = [\alpha_1 := t_1, \dots, \alpha_p := t_p]$ where all placeholder-terms appearing in C are substituted by σ (we say that σ *covers* C) and for $M_1, \dots, M_{n_C}$ some terms, we define the term $\sigma C[M_1, \dots, M_{n_C}]$ as follows.

- if C is a term and $n_C = 0$ then trivially $\sigma C[] \triangleq \sigma C$;
- if $C = \Box \rhd \Gamma \vdash \Delta$ and $n_C = 1$ then $\sigma C[M] \triangleq M$;
- if $C = \mathsf{And}_R(\widehat{a}C_1, \widehat{b}C_2, c)[M_1, \dots, M_{n_C}]$ then $\sigma C[M_1, \dots, M_{n_C}] \triangleq \mathsf{And}_R(\widehat{a}\sigma C_1[M_1, \dots, M_{n_{C_1}}], \widehat{b}\sigma C_2[M_{n_{C_1}+1}, \dots, M_{n_C}], c)$;
- if $C = \mathsf{Exists}_L(\widehat{x}\widehat{\mathrm{x}}C_1, y)$, then $\sigma C[M_1, \dots, M_{n_C}] \triangleq \mathsf{Exists}_L(\widehat{x}\widehat{\mathrm{x}}\sigma C_1[M_1, \dots, M_{n_C}], y)$;
- if $C = \mathsf{Exists}_R(\widehat{a}C_1, \alpha, b)$, then $\sigma C[M_1, \dots, M_{n_C}] \triangleq \mathsf{Exists}_R(\widehat{a}\sigma C_1[M_1, \dots, M_{n_C}], \sigma\alpha, b)$;
- the other remaining cases are similar.

First, we can prove the following simple result.

Lemma 3.1. *For some well-typed open-term $C \rhd \Gamma \vdash \Delta$ whose open leaves are $\Box \rhd \Gamma_i \vdash \Delta_i$ for $1 \leqslant i \leqslant n_C$, for some σ covering C, if for all $1 \leqslant i \leqslant n_C$, $M_i \rhd \sigma\Gamma_i \vdash \sigma\Delta_i$ is a well-typed term, then $\sigma C[M_1, \dots, M_{n_C}] \rhd \sigma\Gamma \vdash \sigma\Delta$ is a well-typed term.*

Proof. We proceed by induction on the context C. □

Let us define now the extended terms and reduction rules associated with the proposition rewrite rule $\mathsf{R} : P \to \varphi$. For some formula φ, for x and a some name and coname, the open-terms denoted $\langle\!|\vdash a : \varphi \,|\!\rangle$ and $\langle\!| \, x : \varphi \vdash |\!\rangle$ are defined in Figure 1. The definition is non-deterministic just as the definition of new deduction rules in super sequent calculus systems. We may pick any of the possibilities just as we do for the computation of new deduction rules. We prove the following lemma.

Lemma 3.2. *Let $\mathsf{R} : P \to \varphi$ be some proposition rewrite rule and let C be $\langle\!|\vdash a : \varphi \,|\!\rangle$. Then, if $\Gamma \vdash a : P, \Delta$ is the conclusion of some instance of R_R, then $C \rhd \Gamma \vdash a : \varphi, \Delta$ is well-typed, and there exists σ covering C such that the premises of C substituted by σ are the premises of this instance of R_R.*

Proof. By construction, an instance of R_R can be transformed into a decomposition of the logical connectors of φ, and thus into some open type inference of $C \rhd \Gamma \vdash a : \varphi, \Delta$, by construction of C. The substitution σ substitutes for the placeholder-terms in this open type inference derivation the terms that are used in this instance of R_R. We obtain thus that the sequents in the premises of C substituted by σ are the premises of this instance of R_R. □

An analogous version of lemma 3.2 can be proven for the introduction of P on the left. So we propose the type inference rules presented in Figure 2 for introducting P on the left and on the right. We obtain the extended proof-terms for a super sequent calculus system. Proofs substitutions are extended in the obvious way on proof-terms.

We define the extended cut-elimination associated with $\xrightarrow{\mathsf{cut}}$, denoted $\xrightarrow{\mathsf{excut}}$ as follows. For each proposition rewrite rule $\mathsf{R} : P \to \varphi$, for each reduction

$$\mathsf{Cut}(\widehat{a}\langle\!|\vdash a : \varphi \,|\!\rangle, \widehat{x}\langle\!| \, x : \varphi \vdash |\!\rangle) \xrightarrow{\mathsf{cut}}{}^{+} C$$

where C is a normal form for $\xrightarrow{\mathsf{cut}}$, we add to $\xrightarrow{\mathsf{cut}}$ the rewrite rule depicted in Figure 3. The cut-elimination $\xrightarrow{\mathsf{excut}}$ is *complete*: any instance of a cut is a redex and thus a normal

$$
\begin{aligned}
\langle\!|\ \Gamma \vdash \Delta\ |\!\rangle &\triangleq \Box \rhd \Gamma \vdash \Delta \quad \text{if } \Gamma \text{ and } \Delta \text{ only contain atomic formulæ}\\
\langle\!|\ \Gamma, x : \varphi \vdash a : \varphi, \Delta\ |\!\rangle &\triangleq \mathsf{Ax}(x, a)\\
\langle\!|\ \Gamma \vdash a : \varphi_1 \Rightarrow \varphi_2, \Delta\ |\!\rangle &\triangleq \mathsf{Imp}_R(\widehat{x}\widehat{b}\langle\!|\ \Gamma, x : \varphi_1 \vdash b : \varphi_2, \Delta\ |\!\rangle, a)\\
\langle\!|\ \Gamma, x : \varphi_1 \Rightarrow \varphi_2 \vdash \Delta\ |\!\rangle &\triangleq \mathsf{Imp}_L(\widehat{y}\langle\!|\ \Gamma, y : \varphi_2 \vdash \Delta\ |\!\rangle, \widehat{a}\langle\!|\ \Gamma \vdash a : \varphi_1, \Delta\ |\!\rangle, x)\\
&\cdots\\
\langle\!|\ \Gamma \vdash a : \forall \mathsf{x}.\varphi, \Delta\ |\!\rangle &\triangleq \mathsf{Forall}_R(\widehat{b}\widehat{\mathsf{x}}\langle\!|\ \Gamma \vdash b : \varphi, \Delta\ |\!\rangle, a) \quad \text{if } \mathsf{x} \notin \mathcal{FV}(\Gamma, \Delta)\\
\langle\!|\ \Gamma, x : \forall \mathsf{x}.\varphi \vdash \Delta\ |\!\rangle &\triangleq \mathsf{Forall}_L(\widehat{y}\langle\!|\ \Gamma, y : \varphi[\mathsf{x} := \alpha] \vdash \Delta\ |\!\rangle, \alpha, x) \quad \alpha \text{ is fresh}
\end{aligned}
$$

Figure 1. Definition of $\langle\!|\ \_\ |\!\rangle$

$$
\mathsf{R}_R\ \frac{\left(\ M_i \rhd \Gamma, x_1^i : A_1^i, \ldots, x_{p_i}^i : A_{p_i}^i \vdash a_1^i : B_1^i, \ldots, a_{q_i}^i : B_{q_i}^i, \Delta\ \right)_{1 \leqslant i \leqslant n}}{\mathsf{R}_R\left(\widehat{\mathsf{x}_1} \ldots \widehat{\mathsf{x}_p}, \left(\widehat{x_1^i} \ldots \widehat{x_{p_i}^i}\,\widehat{a_1^i} \ldots \widehat{a_{q_i}^i} M_i\right)_{1 \leqslant i \leqslant n}, \alpha_1, \ldots, \alpha_q, a\right) \rhd \Gamma \vdash a : P, \Delta}\ \mathcal{C}
$$

n is the number of open leaves of $\langle\!|\vdash a : \varphi\ |\!\rangle$. $\mathcal{C}$ is the side condition of the corresponding rule in the super sequent calculus. The $\mathsf{x}_1, \ldots, \mathsf{x}_p$ are the variables concerned by $\mathcal{C}$ and by lemma 3.2, they are the bound first-order variables of $\langle\!|\vdash a : \varphi\ |\!\rangle$. The $\alpha_1, \ldots, \alpha_q$ are the placeholder-terms appearing in this later open-term. They are to be instantiated by first-order terms when using this type inference rule.

$$
\mathsf{R}_L\ \frac{\left(\ N_j \rhd \Gamma, y_1^j : C_1^j, \ldots, y_{r_j}^j : C_{r_j}^j \vdash b_1^j : D_1^j, \ldots, b_{s_j}^j : D_{s_j}^j, \Delta\ \right)_{1 \leqslant j \leqslant m}}{\mathsf{R}_L\left(\widehat{\mathsf{y}_1} \ldots \widehat{\mathsf{y}_r}, \left(\widehat{y_1^j} \ldots \widehat{y_{r_j}^j}\,\widehat{b_1^j} \ldots \widehat{b_{s_j}^j} N_j\right)_{1 \leqslant j \leqslant m}, \beta_1, \ldots, \beta_s, x\right) \rhd \Gamma, x : P \vdash \Delta}\ \mathcal{C}'
$$

m is the number of open leaves of $\langle\!|\ x : \varphi \vdash |\!\rangle$. $\mathcal{C}'$ is the side condition of the corresponding rule in the super sequent calculus. The $\mathsf{y}_1, \ldots, \mathsf{y}_r$ are the variables concerned by $\mathcal{C}'$ and by the version of lemma 3.2 for introducing P on the left, they are the bound first-order variables of $\langle\!|\ x : \varphi \vdash |\!\rangle$. The $\beta_1, \ldots, \beta_s$ are the placeholder-terms appearing in this later open-term. They are to be instantiated by first-order terms when using this type inference rule. By duality it is expected that $p = s$ and $q = r$.

Figure 2. Type inference rules for a proposition rewrite rule $\mathsf{R} : P \to \varphi$

$$
\begin{aligned}
\sigma\mathsf{Cut}\Bigg(&\widehat{a}\mathsf{R}_R\left(\widehat{\mathsf{x}_1} \ldots \widehat{\mathsf{x}_p}, \left(\widehat{x_1^i} \ldots \widehat{x_{p_i}^i}\,\widehat{a_1^i} \ldots \widehat{a_{q_i}^i} M_i\right)_{1 \leqslant i \leqslant n}, \alpha_1 \ldots \alpha_q, a\right),\\
&\widehat{x}\mathsf{R}_L\left(\widehat{\mathsf{y}_1} \ldots \widehat{\mathsf{y}_r}, \left(\widehat{y_1^j} \ldots \widehat{y_{r_j}^j}\,\widehat{b_1^j} \ldots \widehat{b_{s_j}^j} N_j\right)_{1 \leqslant j \leqslant m}, \beta_1 \ldots \beta_s, x\right)\Bigg) \xrightarrow{\mathsf{excut}} \sigma C[M_1, \ldots, N_m]
\end{aligned}
$$

if $\mathsf{R}_R(\ldots)$ and $\mathsf{R}_L(\ldots)$ freshly introduce a and x.
Here σ substitutes for each placeholder-term a first-order term. However these terms are *meta* just as the symbol t in the eighth and ninth rule of our type system.

Figure 3. Extended cut-elimination rule

form for $\xrightarrow{\mathsf{excut}}$ is cut-free. Subject reduction is implied by lemmas 3.1 and 3.2.

Lemma 3.3 (Subject Reduction)**.** *If $M \xrightarrow{\mathsf{excut}^*} M'$ and $M \rhd \Gamma \vdash \Delta$ is well-typed, then $M' \rhd \Gamma \vdash \Delta$ is well-typed.*

We define a rewrite system denoted $\xrightarrow{\mathsf{prop}}$ on propositions by turning each proposition rewrite rule into a rewrite rule in the standard way (see for example [12]). We define a rewrite system denoted $\xrightarrow{\mathsf{term}}$ on extended proof-terms as follows. It contains for each $\mathsf{R} : P \to \varphi$ the rewrite rules written in Figure 4.

As $\xrightarrow{\mathsf{term}}$ is orthogonal, it is confluent. Besides if $\xrightarrow{\mathsf{term}}$ is confluent and weakly normalising, then the unique normal form of an extended term M is denoted $M\downarrow^{\mathsf{t}}$. Similarly if $\xrightarrow{\mathsf{prop}}$ is confluent and weakly normalising, then the unique normal form of a formula φ is denoted $\varphi\downarrow^{\mathsf{p}}$. This notation is extended to contexts and sequents. It is also extended to open-terms, since they also contain sequents through the

$$\sigma\mathsf{R}_R\left(\widehat{\mathsf{x}_1}\dots\widehat{\mathsf{x}_p},\left(\widehat{x_1^i}\dots\widehat{x_{p_i}^i}\,\widehat{a_1^i}\dots\widehat{a_{q_i}^i}M_i\right)_{1\leqslant i\leqslant n},\alpha_1\dots\alpha_q,a\right)\xrightarrow{\mathsf{term}}\sigma\langle\!\Vdash a:\varphi\Vdash\!\rangle[M_1,\dots,M_n]$$

where σ is a substitution over placeholder-terms covering $\langle\!\Vdash a:\varphi\Vdash\!\rangle$. Here the bound names and conames of this later open-term are supposed different from the free and bound names and conames of $\mathsf{R}_R(\dots)$.

$$\sigma\mathsf{R}_L\left(\widehat{\mathsf{y}_1}\dots\widehat{\mathsf{y}_r},\left(\widehat{y_1^j}\dots\widehat{y_{r_j}^j}\,\widehat{b_1^j}\dots\widehat{b_{s_j}^j}N_j\right)_{j\in\{1,\dots,m\}},\beta_1\dots\beta_s,x\right)\xrightarrow{\mathsf{term}}\sigma\langle\!\Vdash x:\varphi\vdash\!\Vdash\!\rangle[N_1,\dots,N_m]$$

where σ is a substitution over placeholder-terms covering $\langle\!\Vdash x:\varphi\vdash\!\Vdash\!\rangle$. Here the bound names and conames of this later open-term are supposed different from the free and bound names and conames of $\mathsf{R}_L(\dots)$.

Figure 4. Rewrite system on extended terms

$\square\rhd\Gamma\vdash\Delta$ constructor.

Let us prove now that $\xrightarrow{\mathsf{excut}}$ is strongly normalising on well-typed extended terms under the following hypothesis.

Hypothesis 3.1. *For a set of proposition rewrite rules $\mathcal{R}$, the rewrite relation $\xrightarrow{\mathsf{prop}}$ associated with $\mathcal{R}$ is weakly normalising and confluent and for each of its rules $\mathsf{R}:P\to\varphi$ P contains only first-order variables (no function symbol or constant) and $\mathcal{FV}(\varphi)\subseteq\mathcal{FV}(P)$.*

The second hypothesis restricts the use of first-order constants and functions in particular to avoid counterexamples such as the presentation of Russel's paradox from [14] for which the set of proposition rewrite rules terminates but the cut-elimination does not.

Now we can state the main result:

Theorem 3.1 (Strong Normalisation)**.** *If the set of proposition rewrite rules satisfies hypothesis 3.1, then $\xrightarrow{\mathsf{excut}}$ is strongly normalising on well-typed extended terms.*

Proof. First we prove that since $\xrightarrow{\mathsf{prop}}$ is weakly normalising and confluent, then so is $\xrightarrow{\mathsf{term}}$ and moreover if $M\rhd\Gamma\vdash\Delta$ is some well-typed term, then $M\downarrow^{\mathsf{t}}\rhd\Gamma\downarrow^{\mathsf{p}}\vdash\Delta\downarrow^{\mathsf{p}}$ is also a well-typed term. The second step is to prove that if $M\xrightarrow{\mathsf{excut}}M'$, then $M\downarrow^{\mathsf{t}}\xrightarrow{\mathsf{cut}}{}^{+}M'\downarrow^{\mathsf{t}}$. Finally strong normalisation follows from strong normalisation of $\xrightarrow{\mathsf{cut}}$ on Urban's well-typed terms. $\square$

It is interesting to notice that since hypothesis 3.1 implies the cut-admissibility in the super sequent calculus system, and since this system is sound and complete *w.r.t.* predicate logic, it implies the consistency of the corresponding first-order theory.

3.3. Simple examples

The first example we consider is known as *Crabbe's counterexample* and consists in $\mathsf{R}:A\to B\wedge(A\Rightarrow\bot)$. The open-terms associated with it are:

$\langle\!\Vdash a:B\wedge(A\Rightarrow\bot)\Vdash\!\rangle=\mathsf{And}_R(\widehat{b}M_1,\widehat{c}\mathsf{Imp}_R(\widehat{x}\widehat{b'}M_2,c),a)$ and $\langle\!\Vdash x:B\wedge(A\Rightarrow\bot)\vdash\!\Vdash\!\rangle=$ $\mathsf{And}_L(\widehat{y}\widehat{z}\mathsf{Imp}_L(\widehat{y'}\mathsf{False}_L(y'),\widehat{a}M,z),x)$. The reduction

$$\begin{array}{rl}&\mathsf{Cut}(\widehat{a}\mathsf{And}_R(\widehat{b}M_1,\widehat{c}\mathsf{Imp}_R(\widehat{x}\widehat{b'}M_2,c),a),\\&\quad\widehat{x}\mathsf{And}_L(\widehat{y}\widehat{z}\mathsf{Imp}_L(\widehat{y'}\mathsf{False}_L(y'),\widehat{a}M,z),x))\\\xrightarrow{\mathsf{cut}}{}^{*}&\mathsf{Cut}(\widehat{b}M_1,\widehat{y}\mathsf{Cut}(\widehat{a}M,\widehat{x}M_2))\end{array}$$

is replaced by

$$\begin{array}{rl}&\mathsf{Cut}(\widehat{a}\mathsf{R}_R(\widehat{b}M_1,\widehat{x}\widehat{b'}M_2,a),\widehat{x}\mathsf{R}_L(\widehat{y}\widehat{a}M,x))\\\to&\mathsf{Cut}(\widehat{b}M_1,\widehat{y}\mathsf{Cut}(\widehat{a}M,\widehat{x}M_2))\end{array}$$

with *ad hoc* conditions on freshly introduced variables. Let us define the terms δ and Δ respectively as $\mathsf{R}_L(\widehat{y}\widehat{a}\mathsf{Ax}(x,a),x)$ and $\mathsf{R}_R(\widehat{b}\mathsf{Ax}(z,b),\widehat{x}\widehat{b'}\delta,c)$. The following reduction does not terminate:

$$\begin{array}{rl}&\mathsf{Cut}(\widehat{c}\Delta,\widehat{x}\delta)=\mathsf{Cut}(\widehat{c}\Delta,\widehat{x}\mathsf{R}_L(\widehat{y}\widehat{a}\mathsf{Ax}(x,a),x))\\&(\mathsf{R}_L(\widehat{y}\widehat{a}\mathsf{Ax}(x,a),x)\text{ does not freshly introduce }x)\\\to&\mathsf{R}_L(\widehat{y}\widehat{a}\mathsf{Ax}(x,a),x)[x:=\widehat{c}\Delta]\\=&\mathsf{Cut}(\widehat{c}\Delta,\widehat{x}\mathsf{R}_L(\widehat{y}\widehat{a}\mathsf{Ax}(x,a)[x:=\widehat{c}\Delta],x))\\=&\mathsf{Cut}(\widehat{c}\Delta,\widehat{x}\mathsf{R}_L(\widehat{y}\widehat{a}\Delta[c\mapsto a],a))\\=_\alpha&\mathsf{Cut}(\widehat{c}\Delta,\widehat{x}\mathsf{R}_L(\widehat{y}\widehat{c}\Delta,a))\\=&\mathsf{Cut}(\widehat{c}\mathsf{R}_R(\widehat{b}\mathsf{Ax}(z,b),\widehat{x}\widehat{b'}\delta,c),\widehat{x}\mathsf{R}_L(\widehat{y}\widehat{c}\Delta,a))\\\to&\mathsf{Cut}(\widehat{c}\mathsf{Cut}(\widehat{b}\mathsf{Ax}(z,b),\widehat{y}\Delta),\widehat{x}\delta)\\&(\Delta\text{ does not freshly introduces }y)\\\to&\mathsf{Cut}(\widehat{c}\Delta[y:=\widehat{b}\mathsf{Ax}(z,b)],\widehat{x}\delta)\\=&\mathsf{Cut}(\widehat{c}\Delta,\widehat{x}\delta)\to\dots\end{array}$$

This proposition rewrite rules thus breaks cut-elimination. It obviously does not satisfy hypothesis 3.1.

Another interesting cut-reduction is the following. Let us consider the proposition rewrite rule named INC in Section 1. First of all and by completeness of superdeduction, there exists a proof denoted π_1 of $\vdash^{+\text{INC}}$ INC. Besides we already constructed a proof, denoted π_2, in raw classical sequent calculus of INC $\vdash A\subseteq A$. It is interesting to notice

that cut-elimination applied to:

$$\textsc{Cut}\ \frac{\dfrac{\pi_1}{\vdash^{+\textsc{Inc}}\textsc{Inc}} \qquad \dfrac{\pi_2}{\textsc{Inc}\vdash A\subseteq A}}{\vdash^{+\textsc{Inc}} A\subseteq A}$$

gives the proof:

$$\textsc{Inc}_R\ \frac{\textsc{Ax}\ \dfrac{}{x\in A\vdash^{+\textsc{Inc}} x\in A}}{\vdash^{+\textsc{Inc}} A\subseteq A}$$

4. A foundation for new proof assistants

The first strong argument in favour of proof assistants based on superdeduction is the representation of proofs. Indeed, existing proof assistants such as Coq, Isabelle or PVS are based on the proof planning paradigm, where proofs are represented by a succession of applications of tactics and of tacticals. Coq also builds a proof-term, in particular to bring the proof check down to a micro kernel. In these approaches, the witness of the proof is bound to convince the user that the proof is correct but not to actually *explain* it, as usual mathematical proofs often also do. Even if the proof-terms of Coq are displayed as trees or under the form of natural language text, the main steps of the proof are drown in a multitude of usually not expressed logical arguments due to both the underlying calculus and the presence of purely computational parts, *e.g.* the proof that $2+3$ equals 5.

Deduction modulo is a first step forward addressing this later issue by internalising computational aspects of a theory inside a congruence. With the canonical rewrite system on naturals, $P(2+3)\vdash P(5)$ becomes an axiom. However a congruence defined by proposition rewrite rules whose right-hand side is not atomic does not bring the expected comfort to interactive proving: the choice of a proposition representative in the congruence introduces some nondeterminism which is neither useful nor wanted. Superdeduction solves this problem by narrowing the choice of a deduction rule to the presence in the goal of one of the extended deduction rules conclusions and goes a step further by also eliminating trivial logical arguments in a proof. Thereby, superdeduction provides a framework for naturally building but also communicating and understanding the essence of proofs.

Notice that extended deduction rules contain only atomic premises and conclusions, thus proof building in this system is like plugging in theorems, definitions and axioms together. This points out the fact that logical arguments of proofs are actually encoded by the structure of theorems, which explains why they are usually not mentionned.

Another important aspect of superdeduction is its potential ability to naturally encode custom reasoning schemes. Section 2 provides the example of structural induction over Peano naturals. Another interesting case is the encoding of other logics like higher-order logic which has been expressed through proposition rewrite rules in [11]. As an example, the proposition rewrite rule $\epsilon(\alpha(\dot\forall,x))\ \rightarrow\ \forall y.\epsilon(\alpha(x,y))$ is translated into the following deduction rules which mimic the deduction rules of higher-order logic.

$$\frac{\Gamma\vdash^+\epsilon(\alpha(x,y)),\Delta}{\Gamma\vdash^+\epsilon(\alpha(\dot\forall,x)),\Delta}\ (y\notin\mathcal{FV}(\Gamma)) \qquad \frac{\Gamma,\epsilon(\alpha(x,t))\vdash^+\Delta}{\Gamma,\epsilon(\alpha(\dot\forall,x))\vdash^+\Delta}$$

The interesting point is that these behaviours are not encoded inside the underlying logic but are the result of the chosen theory which is only a parameter of superdeduction.

These properties led us to develop a proof assistant based on the super sequent calculus: lemuridae. It features extended deduction rules derivation with focusing, rewriting on first-order terms, proof building with the associated superdeduction system, as well as some basic automatic tactics. It is implemented with the Tom [2] language (`tom.loria.fr`), which provides powerful (associative) rewriting capabilities and strategic programmation on top of Java. The choice of the Tom language has several beneficial consequences. First of all, the expressiveness of the language allows for clean and short code. This is in particular the case of the micro proofchecker, whose patterns faithfully translate deduction rules of sequent calculus. Thus, the proofchecker is only one hundred lines long and it is therefore more realistic to convince everyone that it is actually sound.

The other main contribution of Tom to lemuridae is the expression of tacticals by strategies. The Tom strategy language is directly inspired from early research on Elan [26] and ρ-calculus and allows to compose basic strategies to express complex programs using strategies combinators. In this formalism, a naive proof search tactical is simply expressed by $topdown(elim)$, where $topdown$ is a "call-by-name" strategy and $elim$ has the usual semantics of the corresponding command.

5 Conclusion

We have introduced superdeduction, a new systematic way of extending deduction systems with rules derived from an axiomatic theory. First, we have presented its application to classical sequent calculus along with its properties. After having exhibited a proof-term language associated with this deduction system, we have shown its strong normalisation under non-trivial hypothesis, therefore ensuring the consistency of instances of the system, as well as of a large class of theories. Finally, we have pointed out the benefits of superdeduction in the frame of interactive proof building by presenting our current implementation of superdeduction modulo.

The very promising results obtained when using lemuridæ, first in term of proof discovery agility and second in the close relationship between human constructed proofs and superdeduction ones, are all very encouraging and trigger the further development of the concepts and implementation. Indeed, as seen in section 4, the behaviour of superdeduction systems with propositions considered modulo a congruence is important to finalize. This will for instance allow building proofs modulo the symmetry of equality. Another promising point is program extraction from lemuridae proof-terms along with a computational interpretation of extended deduction rules. We anticipate the extracted programs to have modular structures inherited from the superdeduction proof.

The links between superdeduction and deduction modulo are being worked on [4], but we already can transpose theories expressed by proposition rewrite rules for deduction modulo to super sequent calculus systems. This is in particular the case of Peano's arithmetic [15], but also of Zermelo-Frænkel axiomatisation of set theory [13]. Finally, let us stress out the recent encoding of pure types systems in $\lambda\Pi$-calculus modulo [9]. Adapted to super sequent calculus, this will reinforce the legitimacy of superdeduction as a foundation for proof assistants.

Acknowledgements: Many thanks to Horatiu Cirstea for his detailed comments on previous version this work, to Benjamin Wack for inspiring discussions and his seminal work on this topics, to Dan Dougherty for helpful discussions, to the Modulo meetings and the Protheo team for many interactions.

References

[1] J.-M. Andreoli. Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation*, 2(3):297–347, 1992. 3

[2] E. Balland, P. Brauner, R. Kopetz, P.-E. Moreau, and A. Reilles. Tom: Piggybacking rewriting on java. In *Proceedings of RTA*, 2007. To appear. 9

[3] M. Bezem, D. Hendriks, and H. de Nivelle. Automated proof construction in type theory using resolution. *Journal of Automated Reasoning*, 29(3-4):253–275, 2002. 1

[4] P. Brauner, G. Dowek, and B. Wack. Normalization in supernatural deduction and in deduction modulo. Available on the author's webpage, 2007. 10

[5] P. Brauner, C. Houtmann, and C. Kirchner. Superdeduction at work. Technical report, INRIA Lorraine, 2007. Available at http://hal.inria.fr/inria-00141672. 2, 4

[6] G. Burel. Unbounded proof-length speed-up in deduction modulo. Technical report, INRIA Lorraine, 2007. Available at http://hal.inria.fr/inria-00138195. 2

[7] H. Cirstea and C. 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. 2

[8] H. Cirstea, L. Liquori, and B. Wack. Rewriting calculus with fixpoints: Untyped and first-order systems. In *Proceedings of TYPES*, volume 3085 of *LNCS*. Springer, 2003. 2

[9] D. Cousineau and G. Dowek. Embedding PTS in the $\lambda\Pi$-calculus modulo. In *Proceedings of TLCA*, 2007. 10

[10] P.-L. Curien and H. Herbelin. The duality of computation. In *Proceedings of ICFP '00*, pages 233–243, New York, NY, USA, 2000. ACM Press. 4

[11] G. Dowek. Proof normalization for a first-order formulation of higher-order logic. In E. Gunter and A. Felty, editors, *Proceedings of TPHOL*, volume 1275 of *LNCS*, pages 105–119. Springer, 1997. 9

[12] G. Dowek, T. Hardin, and C. Kirchner. Theorem proving modulo. *Journal of Automated Reasoning*, 31(1):33–72, Nov 2003. 1, 7

[13] G. Dowek and A. Miquel. Cut elimination for Zermelo's set theory. Available on author's web page. 10

[14] G. Dowek and B. Werner. Proof normalization modulo. *Journal of Symbolic Logic*, 68(4):1289–1316, 2003. 8

[15] G. Dowek and B. Werner. Arithmetic as a theory modulo. In J. Giesl, editor, *Proceedings of RTA'05*, volume 3467 of *LNCS*, pages 423–437. Springer, 2005. 4, 10

[16] L. Hallnäs and P. Schroeder-Heister. A proof-theoretic approach to logic programming. II. programs as definitions. *Journal of Logic and Computation*, 1(5):635–660, 1991. 2

[17] X. Huang. Reconstruction proofs at the assertion level. In *Proceedings of CADE*, volume 814 of *LNCS*, pages 738–752. Springer, 1994. 2

[18] R. McDowell and D. Miller. Cut-elimination for a logic with definitions and induction. *Theoretical Computer Science*, 232(1-2):91–119, 2000. 2

[19] Q.-H. Nguyen, C. Kirchner, and H. Kirchner. External rewriting for skeptical proof assistants. *Journal of Automated Reasoning*, 29(3-4):309–336, 2002. 1

[20] M. Parigot. Lambda-mu-calculus: An algorithmic interpretation of classical natural deduction. In A. Voronkov, editor, *Logic Programming and Automated Reasoning*, Springer, pages 190 – 201, St Petersburg, Russia, 1992. 4

[21] D. Prawitz. *Natural Deduction. A Proof-Theoretical Study*, volume 3 of *Stockholm Studies in Philosophy*. Almqvist & Wiksell, Stockholm, 1965. 2

[22] V. Prevosto. Certified mathematical hierarchies: the focal system. In *Mathematics, Algorithms, Proofs*, Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2005. 1

[23] C. Urban. Strong normalisation for a Gentzen-like cut-elimination procedure. In *Proceedings of TLCA*, pages 415–430, 2001. 2, 4

[24] C. Urban and G. M. Bierman. Strong normalisation of cut-elimination in classical logic. *Fundam. Inform.*, 45(1-2):123–155, 2001. 4, 6

[25] S. van Bakel, S. Lengrand, and P. Lescanne. The language $\mathcal{X}$: circuits, computations and classical logic. In M. Coppo, E. Lodi, and G. M. Pinna, editors, *Proceedings of ICTCS'05*, volume 3701 of *LNCS*, pages 81–96. Springer, 2005. 4

[26] E. Visser and Z.-e.-A. Benaissa. A core language for rewriting. In C. Kirchner and H. Kirchner, editors, *Proceedings of WRLA*, volume 15 of *ENTCS*. Elsevier, sep 1998. 9

[27] B. Wack. *Typage et déduction dans le calcul de réécriture*. PhD thesis, Université Henri Poincaré, Nancy 1, October 2005. 2