LINEAR LOGIC AND LOCAL COMPUTATIONS

Ugo Solitro — Silvio Valentini

Dipartimento di Scienze dell'Informazione
Università di Milano

10 November '91

Abstract. This work deals with the exponential fragment of Girard's *linear logic* [Gir87] without the contraction rule. This logical system has a natural relation with the *direct logic* [OnKo85],[KeWe]. A new sequent calculus for this logic is presented in order to remove also the weakening rule and recover its behavior *via* a special treatment of the propositional constants; the process of cut-elimination can be performed using only "local reductions. Hence a typed calculus, which admits only local rewriting rules, can be introduced in a natural manner. Its main properties — normalizability and confluence — has been investigated; moreover this calculus has been proved to satisfy a *Curry-Howard isomorphism* [How69] with respect to the logical system in question.

Preface

One of the most interesting aspects of the linear logic consists in its computational behaviour which is "resource sensitive, as clearly appears since the first work of Girard [Gir87]; since the linear logic is "constructive it seems natural to apply the formulae-as-types paradigm in order to look for a calculus of the proofs which embodies such feature. But, unlike the so called intuitionistic linear logic[1] which originate a functional calculus, the computational behaviour of the original system is unmistakebly non functional as pointed out in [Gir89a], [Gir89b] and [Sol90].

As a matter of fact in the process of proof normalization (namely the computation) it is possible to recognize two kinds of phenomena: *local* steps, which roughly correspond to minor rearrangement of the proof, and *global* steps, which involve the deletion or the duplication of relevant parts of the proof, actually subproofs.

The global steps seems to be related with the functional characters of the computation, whereas in the case of the local steps a functional representation turns out to be an unnatural coercion: indeed it is more natural to conceive interacting objects instead of the application of an operator to its arguments

On the other hand one can interpret the process of normalization as a concurrent process of computation [Gir?]: in this case the global steps are moments of synchronization, instead of local ones which represent parallel independent computations.

In this work we are interested in what we conjecture to be the largest subsystem of linear logic that has a purely local computational behaviour.

The nucleus of the calculus studied in this work was proposed for the first time during the Logic Colloquium '88 [Sol88] (see also [Sol89]) and concerned the multiplicative second order fragment of linear logic. The extension of the original calculus to the additive part was the subject of a conference [SoVa88] held by the first author in Njmegen: in this extension the local character of the calculus was not preserved and it was necessary to introduce a global/functional apparatus.

1991 *Mathematics Subject Classification.* 05.
Key words and phrases. Constructive logics, λ-calculus.

[1] This system has been treated in [Laf88], [Val91] and more recently in the report [Abr90].

Now the calculus defined on '88 has been nicely presented in [Abr90] where our calculus is essentially (*i.e. modulo* a renaming of the constants) what Abramsky call PE_2 without the exponentials.

The problem of local behaviour of the full exponential fragment and its relation with the proof net system was the subject of the thesis [Sol90] where the local behaviour was preserved imposing constraint on the computational process.

Introduction

As usual in exponential propositional linear logic, formulae (which will be denoted by $A, B, C, \dots$) are formed in a language containing a denumerable set of propositional variables (notation $P_1, P_2, \dots$), the binary multiplicative connectives $\otimes$ (*times*) and $\sqcup$ (*par*), the unary exponential connectives $!$ (*of course*) and $?$ (*why not*) and the propositional constants $\top$ and $\mathbf{0}$. We will also use $\mathbf{1}$ and $\bot$ as abbreviations of $!\top$ and $?\mathbf{0}$ respectively.

In the following we will consider sequents of the form $\vdash A_1, \dots, A_n$ which we will write $\vdash \mathbf{X}$ for short, where $\mathbf{X}$ is intended to be a finite multiset of formulae. We will use the *duality operator* $(-)^\perp$ which is a non identical involution on the propositional variables recursively extended to all formulae as follows:

$$\begin{array}{lcc} \text{(base)} & \top^\perp \equiv \mathbf{0} & \mathbf{0}^\perp \equiv \top \\ \text{(step)} & (A \otimes B)^\perp \equiv A^\perp \sqcup B^\perp & (A \sqcup B)^\perp \equiv A^\perp \otimes B^\perp \\ & (!A)^\perp \equiv ?A^\perp & (?A)^\perp \equiv !A^\perp \end{array}$$

It is immediate to show that $A^{\perp\perp} \equiv A$ holds for any formula A. Moreover $\mathbf{1}^\perp \equiv \bot$ and $\bot^\perp \equiv \mathbf{1}$. Note that the duality operator is not a logical connective and hence there is no rule about it.

We recall here the rules of the fragment of multiplicative classical linear logic with the weakening rule (MWLL) we are concerned:[2]

$$\begin{array}{ll} \text{(logical axioms)} & \vdash A, A^\perp \\ \text{(cut)} & \dfrac{\vdash \mathbf{X}, A \qquad \vdash \mathbf{Y}, A^\perp}{\vdash \mathbf{X}, \mathbf{Y}} \\ \text{(multiplicative)} & \dfrac{\vdash \mathbf{X}, A, B}{\vdash \mathbf{X}, A \sqcup B} \qquad \dfrac{\vdash \mathbf{X}, A \qquad \vdash \mathbf{Y}, B}{\vdash \mathbf{X}, \mathbf{Y}, A \otimes B} \\ \text{(exponential)} & \dfrac{\vdash \mathbf{X}}{\vdash \mathbf{X}, ?A} \qquad \dfrac{\vdash \mathbf{X}, A}{\vdash \mathbf{X}, ?A} \qquad \dfrac{\vdash ?\mathbf{X}, A}{\vdash ?\mathbf{X}, !A} \end{array}$$

where $?\mathbf{X} = \{?C : C \in \mathbf{X}\}$;

$$\text{(constant)} \qquad \vdash \mathbf{X}, \top \qquad \vdash \mathbf{1} \qquad \frac{\vdash \mathbf{X}}{\vdash \mathbf{X}, \bot}.$$

Inspecting the proof of the CUT-elimination theorem for linear logic in [Gir87] one can readily convince himself that the CUT-rule is redundant in MWLL.[3]

1. The sequent calculus DLL

The rules of the calculus DLL, which we are now going to define, differ from the previous ones mainly on the treatment of the constants. In fact the logical axioms, the CUT-rule and the multiplicative rules

[2] In linear logic the constants $\mathbf{1}$ and $\bot$ are primitive, but can be defined as shown above.

[3] Here and in the following we will omit the details of the proofs which are standard in the literature.

are exactly as before, whereas the exponential rules lack of the first one (*weakening*) and the last rule is modified in the following way:

(*!-introduction*) $$\frac{\vdash \mathsf{T}, ?\mathbf{X}, A}{\vdash \mathsf{T}, ?\mathbf{X}, !A}$$

where T is a shorthand for a finite number of occurrences of $\top$.

The real novelty of the system rests in the rules about constants:

(constants axioms) $$\vdash \top, P_i \qquad \vdash \top, \top$$

(constant rules) $$\frac{\vdash \mathbf{X}, \top, \top}{\vdash \mathbf{X}, \top} \qquad \frac{\vdash \mathbf{X}, \top \qquad \vdash \mathbf{Y}, \top}{\vdash \mathbf{X}, \mathbf{Y}, \top} \qquad \frac{\vdash \mathbf{X}, \mathbf{1} \qquad \vdash \mathbf{Y}}{\vdash \mathbf{X}, \mathbf{Y}}$$

The justification of the slight change in the rule of *!-introduction* is that $\top$ and $?\top$ are logically equivalent formulae in MWLL. In a similar way, being $\top \otimes \top$ and $\top \sqcup \top$ both logically equivalent to $\top$, the new *constant rules* are admittable.

The new constant axioms are instances of the standard axiom $\vdash \mathbf{X}, \top$; on the other hand the new axioms are powerful enough to recover it as will be shown in the proof of theorem 1.3. The first step is to prove, by induction on *structural complexity*,[4] the following

1.1 Lemma. $\vdash_{\mathrm{DLL}} \top, A$ *for each formula* A.

Instead of giving a formal proof we just observe that the basic cases are axioms of DLL and the constant rules has been chosen just in order to perform the inductive steps.

1.2 Corollary. *If* $\vdash_{\mathrm{DLL}} \mathbf{X}, \mathbf{0}$ *then* $\vdash_{\mathrm{DLL}} \mathbf{X}, A$ *for each formula* A.

It is now easy to show that DLL is equivalent to MWLL.

1.3 Theorem. $\vdash_{\mathrm{DLL}} \mathbf{X}$ *iff* $\vdash_{\mathrm{MWLL}} \mathbf{X}$.

Proof. Both directions of the proof are by induction on the length of the derivation. Here we will illustrate only some representative cases.

For instance, in the *(Only if)* direction, a possible derivation in MWLL of the DLL rule:

$$\frac{\vdash \mathbf{X}, \mathbf{1} \qquad \vdash \mathbf{Y}}{\vdash \mathbf{X}, \mathbf{Y}} \qquad \text{is} \qquad \frac{\vdash \mathbf{X}, \mathbf{1} \qquad \dfrac{\vdash \mathbf{Y}}{\vdash \mathbf{Y}, \bot}}{\vdash \mathbf{X}, \mathbf{Y}}$$

On the *(If)* direction, the MWLL-*axiom* $\vdash A_1, \ldots, A_n, \top$ can be proved in DLL using lemma 1.1 and its corollary 1.2:

$$\dfrac{\dfrac{\dfrac{\vdash \mathbf{0}, \top \qquad \vdash A_1, \top}{\vdash A_1, \mathbf{0}, \top}}{\cdots}}{\vdash A_1, A_2, \top}$$
$$\cdots$$

and so on.

Finally the weakening rule of MWLL can be derived in DLL, again using lemma 1.1:

$$\frac{\vdash \mathbf{X} \qquad \dfrac{\vdash ?A, \top}{\vdash ?A, \mathbf{1}}}{\vdash \mathbf{X}, ?A}$$

[4] By *structural complexity* of a formula A we mean simply the number of logical sign (propositional variables, constants and connective) which are in A; this number will be usually denoted by $\delta(A)$. Formally, given a formula A we recursively define $\delta(A)$ in the usual way: $\delta(P_i) = \delta(\top) = \delta(\mathbf{0}) \stackrel{\text{def}}{=} 1$, $\delta(?A) = \delta(!A) \stackrel{\text{def}}{=} \delta(A) + 1$, $\delta(A \otimes B) = \delta(A \sqcup B) \stackrel{\text{def}}{=} \delta(A) + \delta(B) + 1$; obviously it turns out that $\delta(A) = \delta(A^{\perp})$.

□

It is interesting to note that one of the main differences of our calculus with respect to the standard presentation of linear logic is that we never use weakenings, not only for the formulae whose main connective is *?*, but also in the hidden form of the standard axiom on $\top$ or in the rule of $\bot$-introduction. For this reason in the proof-net calculus corresponding to DLL some of the boxes that Girard uses are not needed.[5]. On the other hand the box corresponding to the rule of *!-introduction* is in our opinion not eliminable and this is why we can not deal with the contraction rule in the style used in our approach.

2. Relationship with Direct Logic

In this section we are going to spell out the relation between DLL and the *propositional direct logic*.

The formulae and the rules of a standard sequent calculus for *propositional direct logic* (let us write DL) are those of the classical propositional calculus with the exception of the *contraction* rule; thus here is a set of rules suitable for DL (see [Bel90]):

(axioms) $$A \vdash A$$

(not) $$\frac{\mathbf{X} \vdash \mathbf{Y}, A}{\neg A, \mathbf{X} \vdash \mathbf{Y}} \qquad \frac{A, \mathbf{X} \vdash \mathbf{Y}}{\mathbf{X} \vdash \mathbf{Y}, \neg A}$$

(and) $$\frac{A, B, \mathbf{X} \vdash \mathbf{Y}}{A \wedge B, \mathbf{X} \vdash \mathbf{Y}} \qquad \frac{\mathbf{X}' \vdash \mathbf{Y}', A \qquad \mathbf{X}'' \vdash \mathbf{Y}'', B}{\mathbf{X}', \mathbf{X}'' \vdash \mathbf{Y}', \mathbf{Y}'', A \wedge B}$$

(or) $$\frac{A, \mathbf{X}' \vdash \mathbf{Y}' \qquad B, \mathbf{X}'' \vdash \mathbf{Y}''}{A \vee B, \mathbf{X}', \mathbf{X}'' \vdash \mathbf{Y}', \mathbf{Y}''} \qquad \frac{\mathbf{X} \vdash \mathbf{Y}, A, B}{\mathbf{X} \vdash \mathbf{Y}, A \vee B}$$

(imply) $$\frac{\mathbf{X}' \vdash \mathbf{Y}', A \qquad B, \mathbf{X}'' \vdash \mathbf{Y}''}{A \to B, \mathbf{X}', \mathbf{X}'' \vdash \mathbf{Y}', \mathbf{Y}''} \qquad \frac{A, \mathbf{X} \vdash \mathbf{Y}, B}{\mathbf{X} \vdash \mathbf{Y}, A \to B}$$

(weakening) $$\frac{\mathbf{X} \vdash \mathbf{Y}}{A, \mathbf{X} \vdash \mathbf{Y}} \qquad \frac{\mathbf{X} \vdash \mathbf{Y}}{\mathbf{X} \vdash \mathbf{Y}, A}$$

In order to obtain an interpretation of direct logic in linear logic observe that in DL *a priori* any formula could have been introduced using the *weakening* rule; on the contrary in a linear logic framework the weakening rule introduces only formulae whose principal connective is *?*. Hence the most obvious translation just puts "enough" *?*, as described below.

2.1 Interpretation.

$$\begin{aligned} \phi(P_i) &= ?P_i & \phi(\neg P_i) &= ?P_i{}^{\perp} \\ \phi(A \wedge B) &= ?(\phi(A) \otimes \phi(B)) & \phi(\neg(A \wedge B)) &= ?(\phi(\neg A) \sqcup \phi(\neg B)) \\ \phi(A \vee B) &= ?(\phi(A) \sqcup \phi(B)) & \phi(\neg(A \vee B)) &= ?(\phi(\neg A) \otimes \phi(\neg B)) \\ \phi(A \to B) &= ?(\phi(\neg A) \sqcup \phi(B)) & \phi(\neg(A \to B)) &= ?(\phi(A) \otimes \phi(\neg B)) \\ \phi(\neg\neg A) &= \phi(A) \end{aligned}$$

By this translation we can state the following result:

2.2 Theorem (Interpretation of the Direct Logic).

$$A_1, \ldots, A_n \vdash_{\mathrm{DL}} B_1, \ldots, B_m \text{ iff } \vdash_{\mathrm{MWLL}} \phi(\neg A_1), \ldots, \phi(\neg A_n), \phi(B_1), \ldots, \phi(B_m).$$

Proof.

[5]This point of view can be found in [SoVa89], [Sol90]

From left to right, the proof is by induction on the length of the derivation. All the inductive steps are almost straightforward whereas the basic case — that is the case of an axiom $A \vdash_{\mathrm{DL}} A$ — requires a secondary induction on the structural complexity $\delta(A)$ of the formula A.

Conversely, note that $\phi(\neg A_1), \ldots, \phi(\neg A_n), \phi(B_1), \ldots, \phi(B_m)$ embodies only $?$, $\otimes$ and $\sqcup$ among the connectives; thus a cut-free derivation of

$$\vdash_{\mathrm{MWLL}} \phi(\neg A_1), \ldots, \phi(\neg A_n), \phi(B_1), \ldots, \phi(B_m)$$

must use only rules which have an immediate correspondent in direct logic; hence we can obtain

$$\vdash_{\mathrm{DL}} \neg A_1, \ldots, \neg A_n, B_1, \ldots, B_m$$

and the result follows possibly using some CUT's. □

This theorem assures that all the results we will obtain for DLL, and in particular the definition of the typed calculus (see sect.4), have a straightforward counterpart in direct logic.

3. The cut elimination theorem

The proof of the CUT-elimination theorem for DLL is fairly standard; moreover, lacking the *contraction* rule, it is much simpler than in the general linear logic setting. Nonetheless, considering the relations with the typed calculus which we are going to discuss in the next sections, it is useful to talk shortly about this subject.

The proof goes on by induction on the structural complexity of the CUT-formula (principal induction), on (the length of) its $?$-thread, and on (the length of) its logical thread. As usual by *thread* we mean the segment of the derivation between the introduction of a formula and the examined occurrence; the *$?$-thread* is simply the thread for a formula of the form $?A$ and 0 otherwise. The *logical thread* is exactly the thread on the non exponential formulae; as regard to the exponential ones we state that the rule of $!$-introduction "introduces" all the formulae in its conclusion. Hence the $!$-introduction rule halts all the logical threads, but none of the $?$-threads. Finally we state that both rules on $\top$ cut off the logical thread of the formula $\top$ (this fact is natural if one recall that the first rule introduces a masked form of $\top \sqcup \top$ and the second one a form of $\top \otimes \top$).

The proof of the CUT-elimination theorem can be thought as the definition of a procedure whose aim is to replace one CUT with other CUT's (possibly none) of lower complexity; without loss of generality the procedure may be defined only on derivations containing exactly one CUT which is also the final rule.

The procedure is a loop that works in two stages and halts when no CUT appears in the derivation. In the first stage it applies the *commutative reductions* in such a way that the logical and $?$-threads of the CUT-formulae will eventually become 0; in the second stage the procedure applies the appropriate CUT-*reduction* in order to reduce the complexity of its final CUT.

For instance, the commutative reduction for $\otimes$ is:

$$\dfrac{\vdash \mathbf{X}, C \qquad \dfrac{\vdash \mathbf{Y}, A, C^{\perp} \qquad \vdash \mathbf{Z}, B}{\vdash \mathbf{Y}, \mathbf{Z}, A \otimes B, C^{\perp}}}{\vdash \mathbf{X}, \mathbf{Y}, \mathbf{Z}, A \otimes B} \quad \longmapsto \quad \dfrac{\dfrac{\vdash \mathbf{X}, C \qquad \vdash \mathbf{Y}, A, C^{\perp}}{\vdash \mathbf{X}, \mathbf{Y}, A} \qquad \vdash \mathbf{Z}, B}{\vdash \mathbf{X}, \mathbf{Y}, \mathbf{Z}, A \otimes B}$$

and clearly reduces the logical thread of $C^{\perp}$.

Now suppose we have reduced the logical thread of $!A^{\perp}$; then the commutative reduction concerning $!$ is:

$$\dfrac{\dfrac{\vdash \mathsf{T}', ?\mathbf{X}, ?A, B}{\vdash \mathsf{T}', ?\mathbf{X}, ?A, !B} \qquad \vdash \mathsf{T}'', ?\mathbf{Y}, !A^{\perp}}{\vdash \mathsf{T}', ?\mathbf{X}, !B, \mathsf{T}'', ?\mathbf{Y}} \quad \longmapsto \quad \dfrac{\vdash \mathsf{T}', ?\mathbf{X}, ?A, B \qquad \vdash \mathsf{T}'', ?\mathbf{Y}, !A^{\perp}}{\dfrac{\vdash \mathsf{T}', ?\mathbf{X}, B, \mathsf{T}'', ?\mathbf{Y}}{\vdash \mathsf{T}', ?\mathbf{X}, !B, \mathsf{T}'', ?\mathbf{Y}}}$$

which reduces the $?$-thread of $?A$.

Let us now consider the CUT-reductions.

If the CUT-formula is introduced by a logical axiom the reduction is:

$$\frac{\vdash \mathbf{X}, A \qquad \vdash A, A^{\perp}}{\vdash \mathbf{X}, A} \qquad \longmapsto \qquad \vdash \mathbf{X}, A$$

(note that in this case it is not necessary to shorten the logical thread of A).

Suppose that a constant axiom introduces one of the CUT-formulae, so that it turns out to be a propositional variable or the constant $\top$; after having reduced both the logical thread, we have to be in the previous situation or in the following one:

$$\frac{\vdash \top, P_i \qquad \vdash \top, P_i^{\perp}}{\vdash \top, \top} \qquad \longmapsto \qquad \vdash \top, \top$$

Finally we show the CUT-reductions concerning the multiplicative connectives ...

$$\dfrac{\dfrac{\vdash \mathbf{X}, A, B}{\vdash \mathbf{X}, A \sqcup B} \qquad \dfrac{\vdash \mathbf{Y}, A^{\perp} \qquad \vdash \mathbf{Z}, B^{\perp}}{\vdash \mathbf{Y}, \mathbf{Z}, A^{\perp} \otimes B^{\perp}}}{\vdash \mathbf{X}, \mathbf{Y}, \mathbf{Z}} \quad \longmapsto \quad \dfrac{\dfrac{\vdash \mathbf{X}, A, B \qquad \vdash \mathbf{Y}, A^{\perp}}{\vdash \mathbf{X}, \mathbf{Y}, B} \qquad \vdash \mathbf{Z}, B^{\perp}}{\vdash \mathbf{X}, \mathbf{Y}, \mathbf{Z}}$$

... and the exponential ones:

$$\dfrac{\dfrac{\vdash \mathbf{X}, A}{\vdash \mathbf{X}, ?A} \qquad \dfrac{\vdash \mathsf{T}, ?\mathbf{Y}, A^{\perp}}{\vdash \mathsf{T}, ?\mathbf{Y}, !A^{\perp}}}{\vdash \mathbf{X}, \mathsf{T}, \mathbf{Y}} \quad \longmapsto \quad \dfrac{\vdash \mathbf{X}, A \qquad \vdash \mathsf{T}, ?\mathbf{Y}, A^{\perp}}{\vdash \mathbf{X}, \mathsf{T}, \mathbf{Y}}$$

Using the CUT-elimination procedure just described it follows that:

3.1 Theorem (CUT-Elimination). *Any proof in* DLL *can effectively be transformed in a* CUT*-free one with the same end-sequent.*

4. The typed calculus

The typed calculus we present in this section is defined in order to obtain an isomorphism *à la Curry-Howard* [How69] for the linear logic.

In the language of the calculus we are going to define the formulae of DLL become, as usual, the *types*; the *terms* are built on variables $(x, x_0, x_1, \dots, y, z, \dots)$ and a constant (ϕ) by means of the constructors $*, \times, \vee, \varpi\langle\rangle, f()$.

We call *judgment* (see [MaLö84]) an enriched sequent of the following shape:

$$\vdash_{\nu} a_1 \colon A_1, \dots, a_n \colon A_n ; c_1 * d_1[C_1], \dots, c_m * d_m[C_m]$$

where A_i, C_j are types and a_i, c_j, d_h are terms. Its intended meaning is: *"if $\vdash A_1, \dots, A_n$ is a sequent provable in* DLL*, the terms in the judgment are the codes of one of its derivation in which $C_j, C_j{}^{\perp} (j = 1, \dots, m)$ are the* CUT*-formulae."* In any judgement the comma and $*$ must be considered as commutative operations since they represent constructions which in DLL do not actually depend on the order.

We say *J-type* of a judgment the sequent obtained removing all the terms from it, namely the J-type of the previous judgment is $A_1, \dots, A_n$. In the sequel the types of a judgment are implicit.

We also write Γ as a shorthand for $a_1 \colon A_1, \dots, a_n \colon A_n$ and possibly $?\Gamma$ in the special case that any A_i is $?B$ or $\top$; finally we use σ as a shorthand for $c_1 * d_1[C_1], \dots, c_m * d_m[C_m]$. Thus the previous judgment becomes $\vdash_{\nu} \Gamma ; \sigma$.

Now here are the rules for the ν-calculus DLLC; note that they are just the rules of DLL enriched with terms which code the derivation.

(logical axioms) $$\vdash x\colon A, x\colon A^{\perp}$$

(constant axioms) $$\vdash \phi\colon \top, \phi\colon P_i \qquad \vdash \phi\colon \top, \phi\colon \top$$

(cut) $$\frac{\vdash \Gamma, a\colon A; \sigma \qquad \vdash \Delta, b\colon A^{\perp}; \tau}{\vdash \Gamma, \Delta; a * b[A], \sigma, \tau}$$

(multiplicative) $$\frac{\vdash \Gamma, a\colon A, b\colon B; \sigma}{\vdash \Gamma, a \vee b\colon A \sqcup B; \sigma} \qquad \frac{\vdash \Gamma, a\colon A; \sigma \qquad \vdash \Delta, b\colon B; \tau}{\vdash \Gamma, \Delta, a \times b\colon A \otimes B; \sigma, \tau}$$

(exponential) $$\frac{\vdash \Gamma, a\colon A; \sigma}{\vdash \Gamma, \varpi\langle a\rangle\colon\ ?A; \sigma} \qquad \frac{\vdash\ ?\Gamma, a\colon A; \sigma}{\vdash\ ?\Gamma, f(a)\colon\ !A; \sigma}$$

(constants rules) $$\frac{\vdash \Gamma, a\colon \top, b\colon \top; \sigma}{\vdash \Gamma, a \vee b\colon \top; \sigma} \qquad \frac{\vdash \Gamma, a\colon \top; \sigma \qquad \vdash \Delta, b\colon \top; \tau}{\vdash \Gamma, \Delta, a \times b\colon \top; \sigma, \tau}$$

$$\frac{\vdash \Gamma, a\colon \mathbf{1}; \sigma \qquad \vdash \Delta; \tau}{\vdash \Gamma, \Delta; \sigma, \tau}$$

The main novelty with respect to a standard judgement calculus is the following:

4.1 Hypothesis of Linearity. *We assume that in the binary rules the sets of variables used in the two premisses are always distinct.*

Hence, just inspecting the rules, one can easily verify that any variable occurs at most twice in a judgment.

The *Linearity Hypothesis* and its consequences state that we are in a linear setting: since a proof must be used exactly once we are compelled to use distinct variables in order to separate different proofs.

Since a derivation in DLLC is just an "enriched" derivation of DLL one can adjust the procedure described in section 3 in order to remove its CUT's. The novelty of the new procedure is that it must be specified how the terms are transformed.

First, it is not difficult to verify the following:

4.2 Lemma. *The commutative reductions have no effect on the terms of the end-judgment.*

As an illustration, the commutative reduction about $!$ becomes:

$$\frac{\dfrac{\vdash\ ?\Gamma, a\colon\ ?A, b\colon B; \sigma}{\vdash\ ?\Gamma, a\colon\ ?A, f(b)\colon\ !B; \sigma} \qquad \vdash\ ?\Delta, c\colon\ !A^{\perp}; \tau}{\vdash\ ?\Gamma, f(b)\colon\ !B, ?\Delta; a * c, \sigma, \tau} \quad \longmapsto \quad \frac{\vdash\ ?\Gamma, a\colon\ ?A, b\colon B; \sigma \qquad \vdash\ ?\Delta, c\colon\ !A^{\perp}; \tau}{\dfrac{\vdash\ ?\Gamma, b\colon\ !B, ?\Delta; a * c, \sigma, \tau}{\vdash\ ?\Gamma, f(b)\colon\ !B, ?\Delta; a * c, \sigma, \tau}}$$

This property is no longer true if we consider the CUT-reductions; for example, the axiom reduction induces:

$$\frac{\vdash \Gamma, a\colon A; \sigma \qquad \vdash x\colon A, x\colon A^{\perp}}{\vdash \Gamma, x\colon A; \sigma, a * x} \qquad \longmapsto \qquad \vdash \Gamma, x[a/x]\colon A; \sigma$$

So, with a slight generalization for the logical axiom case, the CUT-reductions suggest a collection of *ν-reductions* which act directly on the judgments:

(logical axioms) $$\Gamma; \sigma, a * x \quad \overset{\nu}{\longmapsto} \quad \Gamma[a/x]; \sigma[a/x]$$

(non log. ax.s) $$\Gamma; \sigma, \phi * \phi \quad \overset{\nu}{\longmapsto} \quad \Gamma; \sigma$$

(mult.) $$\Gamma; \sigma, (a \times b) * (c \vee d) \quad \overset{\nu}{\longmapsto} \quad \Gamma; \sigma, a * c, b * d$$

(exponentials) $$\Gamma; \sigma, \varpi\langle a\rangle * f(b) \quad \overset{\nu}{\longmapsto} \quad \Gamma; \sigma, a * b$$

We will use $\overset{*}{\longmapsto}$ and $\overset{\nu^*}{\longmapsto}$ as a notations for the transitive and reflexive closure of the reduction relation in DLL and in DLLC, respectively.

4.3 Proposition (Correctness of the ν-reductions). *The ν-reductions are correct with respect to* DLLC*, i.e. if the judgment in the left-hand side is derivable in* DLLC*, then also the judgment in the right-hand side is.*

Proof. Consider a derivation of a judgment to which it is possible to apply one of the ν-reductions. We can assume that the CUT under consideration is the last rule applied and that the CUT-formulae are introduced just above it; indeed, using the commutative reductions, we can always obtain such a derivation.

Now we can apply the suitable CUT-reduction to the derivation and a routine verification will tell us that the new end-judgment is exactly the right-hand side of the ν-reduction. □

An immediate consequence of the foregoing is:

4.4 Theorem (Elimination of cut's). *A derivation of a judgment α can be effectively transformed into a* CUT*-free one of a judgment of the same J-type of α.*

5. More about the ν-Calculus.

The proof of proposition 4.3 suggests a strict relation between the processes of reduction in DLL and in DLLC that we want to spell out. First we recall that, given a derivation π of a sequent $\mathbf{X}$, it is possible to transform it just substituting the rules of DLL with the corresponding rules of DLLC; in this way we can associate to π a judgment $\nu(\pi)$ (of J-type $\mathbf{X}$) which is the end judgment of the transformed derivation.

5.1 Theorem (Isomorphism). *The map $\nu(-)$ is natural, i.e. if $\pi \overset{*}{\longmapsto} \pi'$ using both commutative and* CUT*-reductions then $\nu(\pi) \overset{\nu^*}{\longmapsto} \nu(\pi')$ using only ν-reductions.*

Proof. Since the commutative reductions when applied to a derivation in DLLC have no effect on the end-judgment (Lemma 4.2) one has simply to verify that the effect of any CUT-reduction can be obtained by one ν-reduction. □

Adapting to our framework the standard definition of normal form, the previous theorem shows that the cut-elimination theorem 4.4 is the *weak normal form theorem* for the ν-calculus. Formally:

5.2 Definition (Normal Form).

(1) *A judgment is said to be in* **normal form** *if it embodies no cuts.*
(2) *A judgment α' is said* **a normal form** *of the judgment α if α' is in normal form and $\alpha \overset{\nu^*}{\longmapsto} \alpha'$.*

We now continue the investigation of the properties of the calculus with the following:

5.3 Theorem (Confluence). *Let α be any judgment and suppose that:*

$$\alpha \overset{\nu}{\longmapsto} \beta_1 \qquad \textit{and} \qquad \alpha \overset{\nu}{\longmapsto} \beta_2$$

then there is a judgment δ such that:

$$\beta_1 \overset{\nu^*}{\longmapsto} \delta \qquad \textit{and} \qquad \beta_2 \overset{\nu^*}{\longmapsto} \delta$$

Furthermore the last two reductions can be performed in at most one step.

Proof. Out of all the cases — most of which are trivial — we are going to present the following one.

Suppose that

$$\alpha \equiv \Gamma; \sigma, a * x, b * y.$$

Since a variable appears at most twice in a judgment, the case in which $x = y$ is immediate. Hence let us suppose $x \neq y$.

First note that it is not possible that x occurs in b and y in a at the same time: for x in b means that at some stage of the derivation we have obtained $\Gamma', x\colon A^{\perp}, b\colon B; \sigma'$ and y in a means that the judgment $\Gamma'', a\colon A, y\colon B^{\perp}; \sigma''$ is derivable; without loss of generality we can suppose that the CUT on A is applied at this step, then the linearity hypothesis prevents the application of the cut on B.

This fact stated, suppose that:

$$\alpha \overset{\nu}{\longmapsto} \Gamma[a/x]; \sigma[a/x], b[a/x] * y$$

and

$$\alpha \overset{\nu}{\longmapsto} \Gamma[b/y]; \sigma[b/y], a[b/y] * x.$$

Applying a suitable one-step-reduction we can obtain respectively:

$$\delta_1 \equiv \Gamma[a/x][b[a/x]/y]; \sigma[a/x][b[a/x]/y]$$
$$\delta_2 \equiv \Gamma[b/y][a[b/y]/x]; \sigma[b/y][a[b/y]/x].$$

We must prove that $\delta_1 = \delta_2$. Since obviously x cannot appear in a and y cannot appear in b we have to consider only the following two cases:

(1) if *y does not occur in a*, we obtain:

$$\delta_2 = \Gamma[b/y][a/x]; \sigma[b/y][a/x] = \delta_1.$$

(2) if *y does occur in a, so that x cannot occur in b*, we proceed like in the previous case.

□

A standard consequence of the confluence theorem is the following:

5.4 Corollary. *Any judgment has a unique normal form.*

We conclude the analysis of the main properties of the ν-calculus stating the strong normalizability theorem.

5.5 Theorem (Strong Normalizability). *Any sequence of reductions of a derivable judgment ends in the normal form after a finite number of steps.*

Proof. The proof is by induction on a suitable measure of the complexity of a judgment which depends essentially on the CUT's of its derivation.

Consider the judgment $\Gamma; \sigma, a * b[A]$; recalling the definition of the structural complexity of a formula, we say that the *degree of the* CUT on A is

$$\delta(a * b) \overset{\text{def}}{=} \delta(A) = \delta(A^{\perp}).$$

Finally we can define the *complexity* $\kappa(\alpha)$ *of a judgment* α as the sum of all the degrees of its CUT's. We agree that if a judgment contains no CUT its complexity is 0.

An inspection of the ν-reductions immediately shows that the complexity of the judgment decreases after every reduction; for instance, in the case of multiplicative reduction:

$$\Gamma; \sigma, (a \times b) * (c \vee d) \quad \overset{\nu}{\longmapsto} \quad \Gamma; \sigma, a * c, b * d$$

the complexity diminishes from $k + (m + n + 1)$ to $k + m + n$.

The thesis immediately follows since the number of reduction steps is clearly bounded by the complexity of the judgment. □

Conclusions

In this work we have introduced a new formal system, DLL, which is logically equivalent to the subsystem MWLL of linear logic; as we have seen it is possible to adopt the "formulae-as-types point of view and define a calculus. In other words we have proved that the global rule called weakening can be reduced to local rule and the related calculus is purely local.

It is now time to indicate which questions are still to explore.

First of all we remark that the two mentioned logical system — DLL and MWLL — are not equivalent computationally: namely if a derivation α in MWLL reduces to β and α' is the translation of α in DLL, in general α' does not reduces to the translation of β. The computations of DLL appears to be "lazy", *i.e.* to the deletion of a subproof in MWLL it corresponds something less: we could say that in DLL it will be erased what is strictly inevitable. But the exact relation between the two systems is still unclear.

About a possible extension to the full exponential fragment, we have expressed the opinion that DLL is the largest logical system for local computation. Unfortunately there is no proof of that, even if a close analysis of the CUT-elimination support this opinion. Beside we mention the fact that also the distinction between *global* and *local* computation is not perfectly clear up to now. What one ought to introduce a new mechanism able to produce global computations in a natural manner, and hence to recover functionality; hopefully one should avoid *ad hoc* apparata like in [SoVa88b] and [Abr90].

It seems also possible to deal with the additive part of linear logic in a way similar to what is presented in this work: at the moment there is some problem from the pure logic point of view.

Finally a most intriguing question is about the pure calculus: how is it possible to define a calculus without types with the same expressive power of the λ-calculus? We have already isolated some basic concepts about the syntax and the computations and this could be the subject of a next paper.

References

[Abr90] Samson Abramsky, *Comptational Interpretations of Linear Logic*, report (1990).

[Bel91] Gianluigi Bellin, *Mechanizing proof theory: resource-aware logics and proof transformations to extract implicit information*, Ph.DThesis, Stanford University, 1990.

[Gir87] Jean-Yves Girard, *Linear logic*, Theoretical Computer Science **50(1)** (1987).

[Gir88a] ♠, *Multiplicatives*, Rendiconti del Seminario Matematico dell'Università e Politecnico di Torino (1988).

[Gir89] ♠, *Towards a geometry of interaction*, Contemporary mathematics **92** (1989), American Mathematical Society.

[GLT89] ♠ Girard, Y. Lafont and P. Taylor, *Proofs and types*.

[How80] W.A. Howard, *The formulae-as-types notion of construction*, (manuscript reprinted), H.B. Curry: Essays on combinatory logic, lambda calculus and formalism, Hyndley and Seldin editors, Academic Press, London, 1980.

[KeWe84] Jo Keijzen and R. Weyhrauch, *A decidable fragment of predicate calculus*, Theoretical Computer Science **32(3)** (1984).

[Laf88] Yves Lafont, *The linear abstract machine*, Theoretical Computer Science **59** (1988).

[ML84] Per Martin-Löf, *Intuitionistic Type Theory*, Bibliopolis, Napoli, 1984.

[OnKo85] Hiroakira Ono and Y. Komori, *Logics without the contraction rule*, Journal of Symbolic Logic **50(1)** (1985).

[Sol88] Sergio Solitro, *A typed calculus for a fragment of linear logic*, Abstract of a communication at the Logic Colloquium '88., Journal of Symbolic Logic **55** (1990), 429, (Paper published in [Sol89]).

[Sol89] ♠, *A typed calculus for a fragment of linear logic*, Theoretical Computer Science **68** (1989).

[Sol90] ♠, *La logica lineare come sistema di tipi per un calcolo non funzionale*, Ph.D Thesis, Università di Milano e Università di Torino, 1990.

[SoVa88] Sergio Solitro and Silvio Valentini, *A typed calculus for the additive linear logic*, Conference of the Workshop on typed λ-calculi held in Nijmegen (NL) (1988).

[SoVa89] ♠, *Toward typed lambda-calculi for linear logic*, Proceedings of the Third Italian Conference on Theoretical Computer Science, Modena 1989, 1989.

[Val91] Valentini, *The judgement calculus for Intuitionistic Linear Logic: proof theory and semantics* , Zeit. für math. Logik und Grund. der Math. **37** (1991).