Elementary Complexity and Geometry of Interaction (Extended Abstract)

Patrick Baillot[1] and Marco Pedicini[2]

[1] Institut de Mathématiques de Luminy (Marseille), C.N.R.S.
`baillot@iml.univ-mrs.fr`
[2] Istituto per le Applicazioni del Calcolo "Mauro Picone" (Roma), C.N.R.
`marco@iac.rm.cnr.it`

Abstract. We introduce a geometry of interaction model given by an algebra of clauses equipped with resolution (following [Gir95a]) into which proofs of Elementary Linear Logic can be interpreted. In order to extend geometry of interaction computation (Execution) to more programs of the algebra than just those coming from proofs, we define a variant of Execution (called Weak Execution). Its application to any program of clauses is shown to terminate with a bound on the number of steps which is elementary in the size of the program. We establish that Weak Execution coincides with standard Execution on programs coming from proofs.

Geometry of interaction (GOI) was introduced by Girard ([Gir88a]) as a semantics of computation which: on the one hand, contrary to denotational semantics interprets explicitly the dynamics of computation and handles finite objects; on the other hand expresses this dynamic by more *mathematical* means than syntactical rewriting.

The EXECUTION operation is the mathematical tool inside the model used to interpret the cut-elimination process. This operation is not always defined and sufficient conditions have been given which ensure termination of the computation : in the case of second-order Linear Logic ([Gir88a, Gir95a]) and of untyped lambda-calculus [MR91]), operators coming from the syntax do satisfy such conditions (a nilpotency condition for instance in the case of LL). Various frameworks have been used to describe GOI models: bounded operators on Hilbert spaces ([Gir88a, DR95]), partial applications ([Dan90, Reg92]) and clauses ([Gir95a]). This latter point of view is the one we adopt here.

Elementary Linear Logic (ELL), as Light Linear Logic (LLL), is a variant of Linear Logic in which the rules introducing exponentials have been modified (cf. [Gir95b]) in order to limit the size explosion of proofs during normalization. It is obtained by removing the two principles : $!A \vdash A$ and $!A \vdash !!A$; contraction and weakening are kept unchanged. We consider here a version of ELL without additive connectives and where introduction of the modality ! is handled through a (multi-)functorial promotion rule (called t-promotion, see [Ped96]), which offers the advantage of having simple proof-nets. A proof-net has two main parameters: its *size* (say the number of edges) and its *depth* (maximal nesting of the boxes it contains). The number of steps of its normalization is bounded by a function of the size which is elementary: the expression of this function is an exponential tower whose height only depends on the depth (see [Ped96]).

A drawback of ELL (as well as of LLL) is the lack of a specific semantics of proofs, though a semantics of provability has been given by Kanovitch et al. ([KOS97]). We address this problem from the angle of a GOI semantics.

Achievements and limits of the present work. We present here an algebra of clauses along the lines of [Gir95a] with a kind of depth-preservation property analogous to that of ELL (section 1). Execution is defined through resolution and the operators are certain sets of clauses; a comparison of these operators with Prolog programs can be found in [Gir95a], section 2.3. In addition to usual EXECUTION we define a *weak execution* (section 2) which amounts to giving up the computation of certain products of the EXECUTION (products yielding a deadlock when one restrains the depth).

A size and a depth are defined for general operators respectively as the number of clauses and the arity of the predicates of the terms (all predicates have the same arity). Our main result is then that WEAK EXECUTION always terminates (there is no need for nilpotency sufficient condition for instance) and that the depth being fixed, the number of steps of the computation is bounded by a function of the size of the program which is elementary (proved in section 2). Clearly speaking, in this setting we can bound in advance the run-time of a program provided we know its size and depth. Therefore the intrinsic elementary bound obtained in ELL by logical means has been extended to a semantical ground.

Yet this WEAK EXECUTION presents a serious drawback as it is not in general an associative operation ... Still at least one inclusion is obtained instead of the expected equality (we call this property sub-associativity): the result of *global* EXECUTION is included in the result of any *modular* EXECUTION (see section 3 for a precise statement).

In this abstract we only sketch the main proofs; complete proofs can be found in [BP99].

Acknowledgments. Authors wish to thank Jean-Yves Girard for important suggestions and for pointing out the crucial lemma 2.

1 Resolution Algebra

We need first to introduce clauses and resolution. We then recall the definition of the algebra of clauses given in [Gir95a] before describing the particular algebra we consider in this work : the *layered algebra of clauses*.

A *term language* T is built over variables and a set of symbols of functions; its elements will be denoted t, u. Let $\{P_i\}_{i\in I}$ be a set of predicate symbols given together with their arity; the *language of atoms* $\mathcal{L}$ built over T and this set of predicates is the set of $P_i(t_1, \ldots, t_n)$, where n is the arity of P_i and the t_j's belong to T.

We say that two terms (or atoms) e and e' are *comparable* when there is a substitution θ defined over the variables in e and e' such that $e\theta = e'\theta$.

If e, e' are comparable then there exists a *most general unifier* (*m.g.u.*) i.e. a substitution θ_0 such that for every unifier θ there is a θ' such that $\theta = \theta_0\theta'$. If e and e' are not comparable, we say that they are *orthogonal*: $e \perp e'$.

A *clause* ϕ in the language $\mathcal{L}$ is a sequent

$$\forall x_0, \ldots, x_d. \left(P_i(t_0, \ldots, t_m) \vdash P_j(u_0, \ldots, u_n)\right),$$

where $P_i(t_0, \ldots, t_m)$ and $P_j(u_0, \ldots, u_n)$ are atoms of L with the same variables $x_0, \ldots, x_d$. We will omit to write the quantification.

The *head* of the clause ϕ is the atom $\texttt{head}(\phi) = P_i(t_0, \ldots, t_m)$, its *tail* is the atom $\texttt{tail}(\phi) = P_j(u_0, \ldots, u_n)$.

We introduce also a formal clause 0. Let $\mathcal{C}$ denote the set of clauses over $\mathcal{L}$.

A substitution θ *acts* on a clause ϕ by : $\phi\theta = \texttt{head}(\phi)\theta \vdash \texttt{tail}(\phi)\theta$.

Definition 1 (Resolution). *Given two clauses ϕ and ϕ' we can assume they have disjoint variables (by choosing appropriate instantiations). If $\texttt{tail}(\phi)$ is comparable with $\texttt{head}(\phi')$ and θ is their m.g.u. we define the* resolution *of the two clauses as the clause*

$$\phi \cdot \phi' = \texttt{head}(\phi)\theta \vdash \texttt{tail}(\phi')\theta$$

Otherwise, if $\texttt{tail}(\phi)$ and $\texttt{head}(\phi')$ are not comparable: $\phi \cdot \phi' = 0$.

We fix by convention that the resolution of the clause zero with any other clause is zero; this implies that resolution is associative.

A clause ϕ is said to be a *projection* (resp. a *null-square*) if $\phi^2 = \phi$ (resp. $\phi^2 = 0$), which is equivalent to $\texttt{head}(\phi) = \texttt{tail}(\phi)$ (resp. $\texttt{head}(\phi) \perp \texttt{tail}(\phi)$).

Definition 2 (Resolution Algebra). *Let $\lambda^\star(\mathcal{L})$ be the set of all finite formal linear combinations $\sum \alpha_i\phi_i$ where the scalars α_i belong to $\mathbb{C}$ and the clauses ϕ_i to $\mathcal{C}$. The set $\lambda^\star(\mathcal{L})$ is equipped with*

- *a structure of complex vector space,*
- *a structure of complex algebra, the multiplication being extended by bilinearity from resolution: $\sum \alpha_i\phi_i \sum \beta_j\phi'_j = \sum \alpha_i\beta_j(\phi_i \cdot \phi'_j)$,*
- *a unit w.r.t. multiplication : $\sum_{i \in I} P_i(x_0, \ldots, x_n) \vdash P_i(x_0, \ldots, x_n)$,*
- *an anti-involution defined by $(\sum \alpha_i\phi_i)^\star = \sum \overline{\alpha_i}{\phi_i}^\star$ where $\phi^\star := \texttt{tail}(\phi) \vdash \texttt{head}(\phi)$.*

A norm can be introduced in order to get a $\mathbb{C}^\star$-algebra, see [Gir95a].

Another way to write a combination of clauses is as $\sum \alpha(\phi)\phi$, where the sum is taken over $\mathcal{C}$ and α is an application from the set of clauses $\mathcal{C}$ to $\mathbb{C}$ such that $\alpha^{-1}(\mathbb{C}\backslash\{0\})$ is finite. We will use this notation when it is more convenient.

If $U = \sum \alpha_i\phi_i$ and $V = \sum \beta_i\phi_i$ are two elements with coefficients in $\mathbb{N}$, we write $U \subseteq V$ if for all i, $\alpha_i \leq \beta_i$.

Definition 3 (Execution Formula). *A* wiring *is a finite sum of clauses $\sum \phi_i$ such that for $i \neq j$: $\texttt{head}(\phi_i) \perp \texttt{head}(\phi_j)$ and $\texttt{tail}(\phi_i) \perp \texttt{tail}(\phi_j)$.*

A loop *is a pair of wirings (U, σ) such that σ is hermitian (i.e. $\sigma^\star = \sigma$),* *A loop* converges *when σU is nilpotent, i.e. when $(\sigma U)^n = 0$ for some n. The* execution *of the loop (U, σ) is then the element*

$$\mathtt{Ex}_\sigma(U) := U(1 - \sigma U)^{-1} = U \sum_{k=0}^{n} (\sigma U)^k$$

and the result of the execution is given by

$$\mathtt{Result}_\sigma(U) := (1 - \sigma^2)\mathtt{Ex}_\sigma(U)(1 - \sigma^2).$$

Remark 1. Another way to write the execution is directly as a sum of clauses:

$$\mathtt{Ex}_\sigma(U) = \sum_{\substack{\phi_0 \in U \\ \phi_i \in \sigma U, 1 \leq i \leq k \\ k \leq n}} \phi_0 \cdot \phi_1 \cdot \dots \cdot \phi_k$$

Now we specify the particular language we are going to consider. The terms of T are built over a set of unary symbols of function $\{p, q, r, s\}$; therefore such terms have exactly one free variable, and $t[x]$ will denote a term with free variable x. The *length* $|t|$ of t is the number of symbols of function appearing in it.

Remark 2. Notice that as T is defined over unary symbols of function, if two terms t and u are unifiable then their m.g.u. θ leaves at least one of the two terms unchanged (up to renaming of its variable). For any pair of terms (t, u), only one of the following cases can occur: $t \perp u$ or $t \leq u$ or $t \geq u$, where $t \leq u$ means that u is the unchanged term.

We will consider a family of symbols of predicate $\{P_i\}_{i \in \{1, \dots, m\}}$ of same arity[1] $(d + 1)$. Let $T^d \cdot m$ denote the set of atoms defined this way.

The set $\mathcal{C}^d$ is the set of clauses:

$$\phi = \forall x_0, \dots, x_d. \left(P_i(t_0[x_0], \dots, t_d[x_d]) \vdash P_j(u_0[x_0], \dots, u_d[x_d])\right),$$

where $\mathtt{head}(\phi)$ and $\mathtt{tail}(\phi)$ belong to $T^d \cdot m$. Notice that t_k and u_k $(0 \leq k \leq d)$ are required to have the same free variable.

We call *layered algebra* the algebra of clauses defined over $\mathcal{C}^d$ and we denote it by $\lambda^\star(T^d \cdot m)$. From now on this is the algebra we consider.

2 Weak Execution

A *word* of clauses w is a finite sequence of clauses $w = (\phi_1, \dots, \phi_n)$ with $\phi_i \in \mathcal{C}^d$, and the *product clause* is $\phi_1 \cdot \phi_2 \cdots \phi_n$. A *sub-product* of the word w is the product clause of a word $(\phi_i, \dots, \phi_j)$ for some $i \leq j \leq n$.

Given a clause $\phi = P(t_0, \dots, t_d) \vdash P'(u_0, \dots, u_d)$ its *width* is defined as $||\phi|| := \sup\{|t_k|, |u_k| \,/\, 0 \leq k \leq d\}$. The width of a word w is simply given by $||w|| := \sup_{1 \leq i \leq n}(||\phi_i||)$. The *cardinality* of w is: $N(w) := \#\{\phi_i \mid 1 \leq i \leq n\}$.

[1] The choice of $d + 1$ is done to keep the same notations when we interpret proof-structures, see section 4.

Example : Consider in $\mathcal{C}^0$ the clause $\phi = P(x) \vdash P(rx)$ and let w_n be the word $(\phi, \dots, \phi)$ of length n. In that case we have $||w_n|| = ||\phi|| = 1$, $N(w_n) = 1$ and the product of w_n is the clause $P(x) \vdash P(r^n x)$.

Definition 4 (Acyclicity). *A clause $\phi = P(t_0, \dots, t_d) \vdash P'(u_0, \dots, u_d)$ is an* acyclic clause *if $P \neq P'$, or ($P = P'$ and there exists $k \leq d$ such that for every $i < k$ we have $u_i = t_i$ and $u_k \perp t_k$).*

An acyclic word *(resp.* strictly acyclic word*) is a word $(\phi_1, \dots, \phi_n)$ such that every sub-product ψ is either an acyclic clause or a projection (resp. an acyclic clause).*

Example : Consider in $\mathcal{C}^1$ the clauses $\phi_1 = P(sx_0, x_1) \vdash P(rx_0, x_1)$, $\phi_2 = P(rx_0, rrx_1) \vdash P(rx_0, sx_1)$ and $\phi_3 = P(x_0, sx_1) \vdash P(rx_0, rsx_1)$. Each of them is an acyclic clause. The word $w = (\phi_1, \phi_2, \phi_3)$ has a non-null product but is not acyclic since its subproduct $\phi_2 \cdot \phi_3 = P(rx_0, rrx_1) \vdash P(rrx_0, rsx_1)$ is not an acyclic clause (though it is a null-square).

We now introduce a restricted form of execution over strictly acyclic words of clauses. Contrarily to usual execution we define it not only for converging loops but for any pair of combination (U, σ); theorem 1 will establish the fact that this definition always makes sense (the sum is finite).

Definition 5 (Weak Execution). *Given a pair of combinations (U, σ) denote $U = \sum \alpha(\phi)\phi$ and $\sigma U = \sum \gamma(\phi)\phi$. Its* weak execution *is defined as:*

$$\mathtt{Ex}^{\dagger}_{\sigma}(U) = \sum_{(\phi_0, \phi_1 \dots, \phi_n) \in A} \alpha(\phi_0) (\prod_{i=1}^{n} \gamma(\phi_i))\, \phi_0 \cdot \phi_1 \cdot \dots \cdot \phi_n$$

$$\text{where } A := \left\{ (\phi_0, \phi_1 \dots, \phi_n) \,\middle|\, \begin{array}{l} \alpha(\phi_0) \neq 0, \gamma(\phi_i) \neq 0 \text{ when } i \neq 0, \text{ and} \\ \text{the word } (\phi_0, \dots, \phi_n) \text{ is strictly acyclic} \end{array} \right\}.$$

As a particular case, given a loop (U, σ) its weak execution *is*

$$\mathtt{Ex}^{\dagger}_{\sigma}(U) = \sum_{(\phi_0, \phi_1 \dots, \phi_n) \in A'} \phi_0 \cdot \phi_1 \cdot \dots \cdot \phi_n$$

$$\text{where } A' := \left\{ (\phi_0, \phi_1 \dots, \phi_n) \,\middle|\, \begin{array}{l} \phi_0 \in U, \phi_i \in \sigma U \text{ when } i \neq 0, \text{ and} \\ \text{the word } (\phi_0, \dots, \phi_n) \text{ is strictly acyclic} \end{array} \right\}.$$

The result of the weak execution is in that case defined as

$$\mathtt{Result}^{\dagger}_{\sigma}(U) := (1 - \sigma^2)\mathtt{Ex}^{\dagger}_{\sigma}(U)(1 - \sigma^2).$$

Remark 3. Note that in cases where $\mathtt{Ex}_{\sigma}(U)$ makes sense ((U, σ) is a loop and σU is nilpotent), we have that $\mathtt{Ex}^{\dagger}_{\sigma}(U) \subseteq \mathtt{Ex}_{\sigma}(U)$.

Our first goal is to show that we can bound the width of the product clause of an acyclic word (proposition 1). In the case of a strictly acyclic word this implies that the length of the word cannot exceed a certain bound (depending on the number and the width of the clauses) without yielding zero as result. This bound will be expressed as an exponential tower of height d (proposition 2).

Proposition 1. *Given an acyclic word $w = (\phi_1, \ldots \phi_n)$ with non-null product, we have the following inequality*[2]*: $||\phi_1 \cdot \cdots \cdot \phi_n|| \leq L(||w||, N(w), d)$, where L is defined by*

$$L(a,b,d) := 2^{2^{\cdot^{\cdot^{\cdot^{2^{4ab(d+1)^2}}}}}}$$

and the height of the exponential tower is d (for d=0 we get the exponent $8ab$).

This proposition will be proved further. The result relies of course on the fact that w is strictly acyclic. Otherwise given a fixed width (of word) and cardinality one might exhibit non acyclic words whose products are of arbitrary big width: see for instance the first example given where for any n, $||w_n|| = 1$, $N(w_n) = 1$ and the product of w_n is $P(x) \vdash P(r^n x)$ whose width is n.

Definition 6. *Given three integers $l, N \geq 1$ and s we define*

$$B(l,N,s) := 2^{2^{\cdot^{\cdot^{\cdot^{2^{9 \cdot lN(s+1)^2}}}}}}$$

where the height of the exponential tower is $s+1$.

Proposition 2. *Given a strictly acyclic word w with non-null product and such that $||w|| \geq 1$, its length is bounded by $B(||w||, N(w), d)$.*

We give now the main result: weak execution always terminates and can be computed in an *elementary* number of resolution steps. We state it first for a loop and then give the result for an arbitrary pair of combinations:

Theorem 1. *Let (U, σ) be a loop and let us fix the variables $N = \#\sigma U$ and $k = 1 + \max\{||\phi|| \,/\, \phi \in \sigma U\}$. We have*

$$\mathtt{Ex}^{\dagger}_{\sigma}(U) = \sum_{\substack{(\phi_0, \ldots, \phi_n) \in A' \\ n \leq B(k,N,d)}} \phi_0 \cdot \phi_1 \cdot \cdots \cdot \phi_n$$

where A' is the set given in definition 5.

More generally, (U, σ) being simply a pair of combinations let us denote $U = \sum \alpha(\phi)\phi$, $\sigma U = \sum \gamma(\phi)\phi$. We define then as before $N = \#\{\phi, \gamma(\phi) \neq 0\}$ and $k = 1 + \max\{||\phi|| \,/\, \gamma(\phi) \neq 0\}$. We have:

[2] Note that $||\phi_1 \cdot \cdots \cdot \phi_n||$ should not be confused with $||(\phi_1, \ldots, \phi_n)||$, the former being the width of a clause and the latter the width of the word.

$$\mathtt{Ex}^{\dagger}_{\sigma}(U) = \sum_{\substack{(\phi_0,\ldots,\phi_n)\in A\\ n\leq B(k,N,d)}} \alpha(\phi_0)(\prod_{i=1}^{n}\gamma(\phi_i))\ \phi_0\cdot\phi_1\cdot\dots\cdot\phi_n$$

where A is the set given in definition 5.

Proof (Proof of theorem 1). Let $w = (\phi_0,\ldots,\phi_n)$ be a word in the set A with $n \geq 1$. Let us denote by w' the word $(\phi_1,\ldots,\phi_n)$ on σU. We have $N(w') \leq N$ and $||w'|| \leq k$. Now, if $n > B(k,N,d)$ then $n > B(||w'||, N(w'), d)$ and we know by proposition 2 that w' (and consequently w) has a null product.

Therefore the sum in $\mathtt{Ex}^{\dagger}_{\sigma}(U)$ can be restricted to the words of A such that $n \leq B(k,N,d)$. □

Let us introduce a few more notations on clauses and words of clauses. To each predicate symbol P_i of our set we associate d predicate symbols, one for each arity $k+1$ in $\{1,\ldots,d\}$; we will denote them all by P_i as anyway in atoms the arity of the predicate will be made explicit by the number of terms. For $0 \leq k \leq d-1$ we denote by $T^k\cdot m$ the language built from T and the family of predicates of arity $k+1$ and $\mathcal{C}^k$ is defined as before from $T^k\cdot m$.

Given a clause $\phi = P(t_0,\ldots,t_d) \vdash P'(u_0,\ldots,u_d)$ of $\mathcal{C}^d$ and $0 \leq k \leq d-1$, its k-th *layer* is the clause of $\mathcal{C}^0$ $[\phi]_k := P(t_k) \vdash P'(u_k)$ and its k-th *truncation* is the clause $[\phi]_{(0,k)} := P(t_0,\ldots,t_k) \vdash P'(u_0,\ldots,u_k)$ of $\mathcal{C}^k$.

The k-th layer of a word $w = (\phi_1,\ldots,\phi_n)$ is $[w]_k = ([\phi_1]_k,\ldots,[\phi_n]_k)$; similarly its k-th truncation is $[w]_{(0,k)} = ([\phi_1]_{(0,k)},\ldots,[\phi_n]_{(0,k)})$.

We define width of atoms by: $|P(t_0,\ldots,t_k)| = \sup\{|t_i|/0 \leq i \leq k\}$.

Proof (Proposition 1). We prove the proposition by means of an intermediate inequality, namely we will prove by induction on d the following one:

$$||\phi_1\cdot\dots\cdot\phi_n|| \leq L'(||w||, N(w), d) \tag{1}$$

where $L'(a,b,s)$ is defined inductively by:

$$\begin{cases} L'(a,b,0) = 2ab \\ L'(a,b,s+1) = 2ab2^{4(s+1)L'(a,b,s)} \end{cases} \tag{2}$$

Then the announced result will be obtained as a consequence. Next lemmas give the result for $d = 0$. Until it is differently specified we consider clauses in $\mathcal{C}^0$.

Lemma 1. *Given two clauses ϕ and ψ,*

1. *if* $\mathtt{tail}(\phi) \geq \mathtt{head}(\psi)$ *then* $|\mathtt{head}(\phi\psi)| = |\mathtt{head}(\phi)|$,
2. *if* $\mathtt{tail}(\phi) \leq \mathtt{head}(\psi)$ *then* $|\mathtt{head}(\phi\psi)| \leq |\mathtt{head}(\psi)| + |\mathtt{head}(\phi)|$.

Remark 4. Given a word $w = (\phi_1,\ldots,\phi_n)$ with non-null product, let us denote $\{j_1,\ldots,j_m\} = \{j \geq 2 \mid \mathtt{tail}(\phi_1\cdot\dots\cdot\phi_{j-1}) < \mathtt{head}(\phi_j)\}$. By induction over the integer m we deduce from the previous lemma the following inequality:

$|\mathtt{head}(\phi_1\cdot\dots\cdot\phi_n)| \leq |\mathtt{head}(\phi_1)| + \sum_{i=1}^{m}|\mathtt{head}(\phi_{j_i})|$; analogously for the $\mathtt{tail}(\phi_1\cdot\dots\cdot\phi_n)$.

Lemma 2. *An acyclic word $w = (\phi_1, \ldots, \phi_n)$ with non null product denoted by $\psi := \phi_1 \cdot \cdots \cdot \phi_n$, satisfies*

$$||\psi|| \leq ||w||(N(w) + 1).$$

Proof. In order to get contradiction assume $||\psi|| > ||w||(N(w)+1)$. In that case either $|\mathtt{head}(\psi)| > ||w||(N(w) + 1)$ or $|\mathtt{tail}(\psi)| > ||w||(N(w) + 1)$. Suppose for instance that we are in the first situation (the second case is handled in a completely symmetric way). By remark 4, using the same notations we have that $|\mathtt{head}(\phi_1 \cdot \cdots \cdot \phi_n)| \leq (m+1)||w||$; then we have $m \geq N(w) + 1$, so there exist $i_1 < i_2$ such that $\phi := \phi_{j_{i_1}} = \phi_{j_{i_2}}$.

We claim that the sub-product $\phi_{j_{i_1}} \cdot \cdots \cdot \phi_{j_{i_2}-1}$ gives a cyclic clause, hence the contradiction with the acyclicity of w. Indeed: let us denote $\Pi' := \phi_1 \cdot \cdots \cdot \phi_{j_{i_1}-1}$ and $\Pi'' := \phi_{j_{i_1}+1} \cdot \cdots \cdot \phi_{j_{i_2}-1}$; then we have $\mathtt{tail}(\Pi' \cdot \phi \cdot \Pi'') < \mathtt{head}(\phi)$. So as $\mathtt{tail}(\phi \cdot \Pi'') \leq \mathtt{tail}(\Pi' \cdot \phi \cdot \Pi'')$, we get $\mathtt{tail}(\phi \cdot \Pi'') < \mathtt{head}(\phi)$. Moreover, from $\mathtt{head}(\phi) \leq \mathtt{head}(\phi \cdot \Pi'')$, we deduce $\mathtt{tail}(\phi \cdot \Pi'') < \mathtt{head}(\phi \cdot \Pi'')$ and we are done. □

This lemma ends the base case of induction ($d = 0$) since

$$||w||(N(w) + 1) \leq 2||w||N(w) = L'(||w||, N(w), 0), \quad \text{as } N(w) \geq 1.$$

In order to get the step of induction, we need a few intermediary results about products of clauses.

Lemma 3. *Let us consider a word $w = (\phi_1, \ldots, \phi_n)$ with non null product; the product of w induces a unique substitution family $(\sigma_1^0, \ldots, \sigma_n^0)$ such that σ_i^0 is defined on the variable of ϕ_i and*

$$\begin{aligned} \phi_1 \cdot \cdots \cdot \phi_n &= \mathtt{head}(\phi_1)\sigma_1^0 \vdash \mathtt{tail}(\phi_n)\sigma_n^0, \\ \textit{and} \quad \mathtt{tail}(\phi_i)\sigma_i^0 &= \mathtt{head}(\phi_{i+1})\sigma_{i+1}^0 \textit{ when } 1 \leq i \leq n-1. \end{aligned}$$

Moreover, every substitution family $(\sigma_1, \ldots, \sigma_n)$ such that σ_i is defined on the variable of ϕ_i and satisfying:

$$\mathtt{tail}(\phi_i)\sigma_i = \mathtt{head}(\phi_{i+1})\sigma_{i+1} \textit{ when } 1 \leq i \leq n-1 \tag{3}$$

can be obtained from $(\sigma_1^0, \ldots, \sigma_n^0)$ by means of a substitution θ such that $(\sigma_1, \ldots, \sigma_n) = (\sigma_1^0\theta, \ldots, \sigma_n^0\theta)$.

These properties are proved by induction over the length n of the word.

Remark 5. Note that for $2 \leq i \leq n-1$ we have $\mathtt{tail}(\phi_i\sigma_i^0) = \mathtt{head}(\phi_{i+1}\sigma_{i+1}^0)$ and that this term is equal either to $\mathtt{tail}(\phi_1 \cdot \cdots \cdot \phi_i)$ if $\mathtt{tail}(\phi_1 \cdot \cdots \cdot \phi_i) \geq \mathtt{head}(\phi_{i+1} \cdot \cdots \cdot \phi_n)$, or to $\mathtt{head}(\phi_{i+1} \cdot \cdots \cdot \phi_n)$ otherwise.

Lemma 4. *Let us consider a word $w = (\phi_1, \ldots, \phi_n)$ with non null product; the product of w induces a word $w' = (\phi_1', \ldots, \phi_n')$ such that $\phi_i' = \phi_i\sigma_i^0$ $(1 \leq i \leq n)$ with σ_i^0 as in lemma 3. Then for every i and h, we have that the subproduct $\phi_i \cdot \cdots \cdot \phi_{i+h}$ is a projection if and only if the corresponding sub-product of w', $\phi_i' \cdot \cdots \cdot \phi_{i+h}'$ is a projection.*

If $\phi_i \cdot \cdots \cdot \phi_{i+h}$ is a null-square then $\phi_i' \cdot \cdots \cdot \phi_{i+h}'$ is a null-square.

We establish now the induction step of inequality (1). Assume the inequality is true for any acyclic word in $\mathcal{C}^k$ with $k \leq d$, and take a word $w = (\phi_1, \ldots, \phi_n)$ over $\mathcal{C}^{(d+1)}$. Consider for every layer $[w]_k$ the induced family of substitutions: $(\sigma_1^k, \ldots, \sigma_n^k)$. Let w' be the word obtained by applying in w the substitution family to every layer $k \leq d$ and by freezing variables by means of newly introduced symbols of constants a_k: i.e. $[\phi_i']_k = [\phi_i]_k \sigma_i^k \langle a_k / x_k \rangle$ for $0 \leq k \leq d$ and $[\phi_i']_{d+1} = [\phi_i]_{d+1}$.

Notice that in w' variables remain only in the last layer $d+1$, so we can consider w' built over clauses of $\mathcal{C}^{(0)}$ with the first d layers constituting the predicate (we enlarge our set of predicates).

Let us show that w' is an acyclic word: we take a sub-word $(\phi_i', \ldots, \phi_{i+h}')$ and its product $\phi' := \phi_i' \cdot \cdots \cdot \phi_{i+h}'$; we denote the corresponding sub-product in w by $\phi := \phi_i \cdot \cdots \cdot \phi_{i+h}$. By lemma 4, if the layer $[\phi]_k$ is a projection then $[\phi']_k$ is a projection too and if $[\phi]_k$ is a null-square then $[\phi']_k$ is a null-square. Combined with the fact that ϕ is an acyclic clause (definition 4), this implies that ϕ' is an acyclic clause.

So w' is an acyclic word in $\mathcal{C}^0$ and by establishing $N(w')$ and $||w'||$ we obtain the following inequality:

$$||\phi_1' \cdot \cdots \cdot \phi_n'|| \leq L'(||w'||, N(w'), 0). \tag{4}$$

As the width of a word doesn't depend upon predicates appearing in its clauses and terms in w' are equal to terms in the last layer of w, we have $||w'|| \leq ||w||$. By definition $N(w')$ is the number of distinct clauses in w'; in order to find it we can calculate the number of all possible instances of terms in w'.

Remark 5 tells us that $\mathtt{tail}([\phi_i']_k)$ is equal to $\mathtt{tail}([\phi_1 \cdot \cdots \cdot \phi_i]_k \langle a_k / x_k \rangle$ or $\mathtt{head}([\phi_{i+1} \cdot \cdots \cdot \phi_n]_k \langle a_k / x_k \rangle$ (similarly for $\mathtt{head}([\phi_i']_k)$). Moreover we have

$$||[\phi_1 \cdot \cdots \cdot \phi_i]_k|| \leq ||[\phi_1 \cdot \cdots \cdot \phi_i]_{(0,k)}||.$$

Since by induction hypothesis: $||[\phi_1 \cdot \cdots \cdot \phi_i]_{(0,k)}|| \leq L'(||[w]_{(0,k)}||, N([w]_{(0,k)}), k)$, we can apply the inequalities $N([w]_{(0,k)}) \leq N(w)$ and $||[w]_{(0,k)}|| \leq ||w||$, and we get

$$||[\phi_1 \cdot \cdots \cdot \phi_i]_{(0,k)}|| \leq L'(||w||, N(w), k).$$
$$\text{Similarly} \quad ||[\phi_{i+1} \cdot \cdots \cdot \phi_n]_{(0,k)}|| \leq L'(||w||, N(w), k).$$

Finally, we obtain: $||[\phi_i']_k|| \leq L'(||w||, N(w), k)$, and the number of all possible unary terms built in our language[3] of length at most $L'(||w||, N(w), k)$ is bounded by $4^{L'(||w||, N(w), k)+1}$.

We are now able to bound the number of clauses $N(w')$ in w': the number of possibilities for the choice of the head and tail predicates is bounded by $N(w)$;

[3] as the number of symbols of function in our language is 4, the number of terms of length k is 4^k, and the number of terms of length at most l is $\sum_{k=0}^{l} 4^k \leq 4^{l+1}$.

at the level k the number of possibilities for the head and tail terms is bounded by $4^{2(L'(||w||,N(w),k)+1)}$. Therefore we have:

$$N(w') \leq N(w) \prod_{k=0}^{d} 4^{2(L'(||w||,N(w),k)+1)} = N(w) \prod_{k=0}^{d} 2^{4(L'(||w||,N(w),k)+1)}.$$

By substitution of quantities $N(w')$ and $||w'||$ in (4) we have

$$\begin{aligned} ||\phi'_1 \cdot \dots \cdot \phi'_n|| &\leq L'(||w'||, N(w'), 0) = 2||w'||N(w') \\ &\leq 2||w||N(w) \prod_{k=0}^{d} 2^{4(L'(||w||,N(w),k)+1)} \\ &\leq 2||w||N(w) 2^{4(d+1)L'(||w||,N(w),d)} = L'(||w||, N(w), d+1) \quad (5) \end{aligned}$$

We used the following inequality $L'(a, b, k) + 2 \leq L'(a, b, d)$ for $k \leq d - 1$.

We therefore get $||[\phi_1 \cdot \dots \cdot \phi_n]_{d+1}|| \leq L'(||w||, N(w), d + 1)$, and by induction hypothesis we have: $||[\phi_1 \cdot \dots \cdot \phi_n]_{(0,d)}|| \leq L'(||w||, N(w), d)$. Since $L'(||w||, N(w), d) \leq L'(||w||, N(w), d+1)$, we get:

$$||\phi_1 \cdot \dots \cdot \phi_n|| \leq L'(||w||, N(w), d+1).$$

This ends our proof for the induction step and the inequality (1) is established.

Using inequalities $\alpha \leq 2^\alpha$ and $x + y \leq xy$ whenever $x \geq 2$ and $y \geq 2$ one easily checks that: $L'(a, b, d) \leq L(a, b, d)$. This way we infer from inequality (1) proposition 1. □

The proof of proposition 2 uses proposition 1 and is in the same spirit (see [BP99]).

3 Sub-Associativity of Weak Execution

Let $\vdash \Gamma, \Delta, \Delta'$ be a sequent such that in Δ and Δ' formulas can be assembled dually in pairs $(B, B^\perp)$. We consider the algebra $\lambda^\star(\Delta, \Delta', \Gamma)$ built, as in section 1, using the language T and the family of predicates of arity $(d+1)$, $\{P_A\}_{A \in \Delta, \Delta', \Gamma}$. Let:

$$\sigma_{\Delta;\Delta',\Gamma} = \sum_{B \in \Delta} P_B(x_0, \dots, x_d) \vdash P_{B^\perp}(x_0, \dots, x_d)$$

We denote $\sigma_{\Delta;\Delta',\Gamma}$ by σ and $\sigma_{\Delta';\Delta,\Gamma}$ by τ, so that $\sigma + \tau = \sigma_{\Delta,\Delta';\Gamma}$.

Proposition 3 (Sub-associativity of weak execution). *Let U be a wiring of $\lambda^\star(\Delta, \Delta', \Gamma)$ and σ and τ defined as above; we have:*

$$\mathtt{Result}^\dagger_{\sigma+\tau}(U) \subseteq \mathtt{Result}^\dagger_\tau(\mathtt{Result}^\dagger_\sigma(U)).$$

Remark 6. The equality is false in general, which contrasts with usual execution and the expected modularity of a valuable computation process. Still, as far as we are dealing with loops coming from proofs, associativity is valid since we will prove in the sequel that weak execution and ordinary execution coincide on such loops.

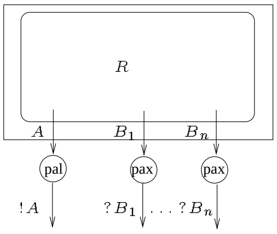
4 Interpretation of ELL Proof-Structures

We consider elementary linear logic with t-promotion and without additives and quantifiers. The sequent calculus is given in [Ped96]; the rules are as in multiplicative exponential linear logic but for dereliction which is not included and for promotion which is replaced by t-promotion: from $\vdash A, \Delta$ infer $\vdash !A, ?\Delta$. We now give the corresponding definition of *ELL proof-structures*. As usual there is a translation of proofs into proof-structures, yielding *ELL proof-nets*.

4.1 ELL Proof-Structures

Definition 7. *The* ELL proof-structures *are graphs with boxes whose edges are labelled by (multiplicative exponential) LL formulas; they are defined inductively together with their* depth *by:*

- *A proof-structure of depth 0 is a labeled graph R built over the nodes:*
  - *Axiom and cut: an* **ax** *(axiom) node has no premise and two conclusions labeled by dual formulas A and $A^{\perp}$; a* **cut** *node has two premises labeled by dual formulas (*cut formulas*) and no conclusion; we consider axioms labeled by atomic formulas to simplify the definition of the interpretation.*
  - *Multiplicative nodes: a $\otimes$ node (resp. a $⅋$ node) has two premises labeled by A and B and one conclusion $A \otimes B$ (resp. $A ⅋ B$).*
- *if $R_1, \ldots, R_n$ are proof-structures of maximal depth d then a graph R built from $R_1, \ldots R_n$ using the preceding nodes and the following exponential nodes is a proof-structure of depth d: a* $?$**c** *(contraction) node has two premises labeled by $?A$ and one conclusion labeled by $?A$; a* $?$**w** *(weakening) node has no premise and one conclusion labeled by a formula $?A$;*
- *if R is a proof-structure of depth d, the box containing R and with conclusions as on the figure is a proof-structure of depth $d+1$.*



The depth *of an edge is the number of boxes it is contained in.*

We consider in proof-structures oriented paths crossing multiplicative and exponential nodes either from a premise to the conclusion or from the conclusion to a premise, and axiom nodes (resp. cut nodes) from a conclusion (resp. premise) to the other conclusion (resp. premise). A path is *up* (resp. *down*) if it only crosses nodes from conclusion to premise (resp. premise to conclusion).

The *length* of a path is the number of edges it goes through. If γ_1 is a path ending upwards (resp. downwards) with an edge conclusion (resp. premise) of a node N and γ_2 starts upwards (resp. downwards) with an edge premise (resp. conclusion) of N, we denote by $\gamma_1;\gamma_2$ their concatenation.

An *elementary path* of R is a path going upwards from a conclusion or a cut node to an axiom and then downwards to a conclusion or a cut node; we denote their set by $\mathcal{P}_e(R)$. A *constant-depth path* of R is a path of R which doesn't cross any box node, axiom node or cut node and starting upwards with a premise of box node or downwards with a conclusion of box node. The depth of such a path is the number of boxes of R it is contained in.

A proof-structure R gives a multiset Γ of conclusion formulas and a multiset Δ of cut formulas (associated dually in couples $(B, B^{\perp})$ by cut nodes). The language we consider is $T^d \cdot m$ where d is the depth of the proof-structure R and m is the cardinality of Γ, Δ. Predicates are indexed by formulas in Γ, Δ. The wiring part U_R of the loop interpreting R will be obtained by interpreting each elementary path of R by a clause.

4.2 Interpretation of a Proof-Structure

Representation of a constant-depth path by a term. As they don't cross axiom or cut nodes, constant-depth paths are up or down. We only consider constant-depth paths which don't visit any weakening node; this is enough to give the interpretation of proof-structures.

We associate to such a path γ of depth i a term $t_\gamma[x_i]$; we define this interpretation below in the case of a path oriented up by induction on the length of the path. In the case of a down path the interpretation t_γ is that of the reverted up path (orientation will be taken into account when we introduce the clauses...).

- if γ is reduced to an edge premise of a box node, then $t_\gamma = x_i$,
- otherwise we can write $\gamma = \gamma_1;\gamma_2$ where γ_2 is reduced to an edge premise of a multiplicative or a contraction node:
  - if γ_2 is the left (resp. right) premise of a multiplicative node then $$t_\gamma = t_{\gamma_1}[px_i/x_i] \text{ (resp. } t_\gamma = t_{\gamma_1}[qx_i/x_i]).$$
  - if γ_2 is the left (resp. right) premise of a contraction node then $$t_\gamma = t_{\gamma_1}[rx_i/x_i] \text{ (resp. } t_\gamma = t_{\gamma_1}[sx_i/x_i]).$$

Representation of an elementary path by a clause. If γ is an elementary path of the proof-structure R of depth d, it can be decomposed as:

$$\gamma = \gamma_i;\gamma_{i+1};\ldots;\gamma_j;\gamma'_j;\gamma'_{j-1};\ldots;\gamma'_k$$

where $0 \leq i, j \leq d$ and the path γ_l (resp. γ'_l) for $i \leq l \leq j$ (resp. $k \leq l \leq j$) is a constant-depth up path (resp. down path) of depth l.

Let A (resp. A') be the beginning (resp. ending) conclusion or cut formula. Their respective depths (i.e. the depths of their edges) are i and k. The clause $W(\gamma)$ interpreting the path γ (the *weight* of the path) is given by:

$$P_A(x_0,..,x_{i-1},t_{\gamma_i},..,t_{\gamma_j},x_{j+1},..,x_d) \vdash P_{A'}(x_0,..,x_{k-1},t_{\gamma'_k},..,t_{\gamma'_j},x_{j+1},..,x_d)$$

The ELL proof-structure R is interpreted by the loop (U_R, σ_R) with:

$$U_R = \sum_{\gamma \in \mathcal{P}_e(R)} W(\gamma)$$
$$\sigma_R = \sum_{B \in \Delta} P_B(x_0, \ldots, x_d) \vdash P_{B^\perp}(x_0, \ldots, x_d).$$

5 Weak Execution of Proof-Nets

In this section we prove that for every proof net R the associated loop (U, σ) satisfies: $\mathtt{Result}_\sigma(U) \subseteq \mathtt{Result}^\dagger_\sigma(U)$. The equality $\mathtt{Result}_\sigma(U) = \mathtt{Result}^\dagger_\sigma(U)$ follows then by remark 3.

First we will prove a proposition (4) and then we will derive this result as a corollary (1). Let us give before a few definitions.

A *balanced path* of R is a path starting upwards in a conclusion of R or downwards in a cut premise, ending downwards in a conclusion of R or in a cut premise. An *elementary balanced path* γ of R is a balanced path crossing at most one cut node, so that :

- if γ crosses no cut node it is an elementary path and its weight is given in the previous section;
- if it crosses a cut node from the premise B to the premise $B^\perp$ then it can be decomposed in the path just crossing the cut with weight $\sigma_0 = P_B(x_0, \ldots, x_d) \vdash P_{B^\perp}(x_0, \ldots, x_d)$ and in an elementary path γ_0 with weight $W(\gamma_0)$, so its weight is $W(\gamma) = \sigma_0 \cdot W(\gamma_0)$.

Any balanced path γ can be written as a concatenation of elementary balanced paths: $\gamma = \gamma_0; \ldots; \gamma_n$ and its weight is given by the product

$$W(\gamma) = W(\gamma_0) \cdot \cdots \cdot W(\gamma_n).$$

Definition 8. *We say a clause $\phi = P(t_0, \ldots, t_d) \vdash P'(u_0, \ldots, u_d)$ is* cyclic at depth $k \leq d$ *if:* **(1)** *$P = P'$,* **(2)** *for all $i < k$, $t_i = u_i$,* **(3)** *$t_k \neq u_k$ and t_k and u_k are comparable.*

We say the clause is cyclic at depth $+\infty$ if it is a projection.

We need three intermediary lemmas:

Lemma 5. *Let R be a proof-net and γ be a balanced path of R such that $W(\gamma)$ is non-null and cyclic at depth k. Then γ crosses at least one cut in R at depth lower than k.*

A *special cut w.r.t. a path γ* is an exponential cut σ such that γ crosses σ but doesn't cross any cut below the auxiliary ports of the box associated to the !-premise of σ (special cuts have been introduced by Regnier and Danos in [Reg92], [Dan90]). We use a variant of the "special cut lemma" stated in [Reg92]:

Lemma 6. *Let γ be a path of a proof-net R. If γ crosses only exponential cuts at depth lower than k and at least one, then R has a special exponential cut w.r.t. γ at depth lower than k.*

Lemma 7. *Let R be a proof-net and γ be a balanced path of R such that $W(\gamma)$ is cyclic at depth k. Assume σ is a cut of R at depth lower than k and crossed by γ which is either a multiplicative or axiom cut or a special exponential cut w.r.t. γ. Let R' be the proof-net obtained from R by reducing σ. Then R' has a balanced path γ' such that $W(\gamma')$ is non-null and cyclic at depth k.*

Proposition 4. *Given a proof net R and γ a balanced path of non-null weight, the clause $W(\gamma)$ associated to γ is acyclic.*

Proof. In order to get contradiction assume the proof-net R has a balanced path φ of non-null weight cyclic at depth k. By lemma 5 this implies that φ crosses at least one cut in R at depth lower than k. The idea is then to reduce progressively all the cuts at depth lower than k crossed by φ in such a way that at each step we keep in the corresponding proof-net a path satisfying the hypothesis. Now in order to do so we need to consider a particular strategy of reduction:

- if there is a multiplicative cut at depth lower than k crossed by the path, then we reduce it,
- otherwise, if all cuts crossed by the path at depth lower than k are exponential then we choose a special cut w.r.t. the path and reduce it.

We build a sequence (R_i, δ_i) of pairs of a proof-net and a path in it satisfying the property : $W(\delta_i)$ is non-null and cyclic at depth lower than k. Put $R_0 = R$ and $\delta_0 = \varphi$. Now assume the sequence has been defined up to rank $i \geq 0$. By lemma 5, δ_i crosses at least one cut in R_i at depth lower than k. If it crosses a multiplicative or axiom cut σ at depth lower than k take for R_{i+1} the proof-net obtained from R by reducing σ ; then by lemma 7 we know that R_{i+1} has a path satisfying the hypothesis which we take as δ_{i+1}. Otherwise lemma 6 ensures that δ_i has a special exponential cut σ at depth lower than k and this is the cut we choose.

This way we build an infinite sequence (R_i, δ_i) of pairs of a proof-net and a path in it with these properties. This sequence contradicts the strong normalization property of ELL. ☐

From this proposition, we derive the two following corollaries:

Corollary 1. *Let (U, σ) be the loop associated to a proof-net R; we have:*

$$\mathtt{Result}^{\dagger}_{\sigma}(U) = \mathtt{Result}_{\sigma}(U).$$

Corollary 2. *Let $(U, \sigma + \tau)$ be the loop associated to a proof-net R, then:*

$$\mathtt{Result}^{\dagger}_{\sigma}(\mathtt{Result}^{\dagger}_{\tau}(U)) = \mathtt{Result}^{\dagger}_{\sigma+\tau}(U) = \mathtt{Result}_{\sigma+\tau}(U).$$

Conclusion and Perspectives: Broadly speaking, our aim is to define a setting as large – and as simple – as possible for elementarily bounded computations. Weak Execution satisfies the complexity requirement with respect to programs of clauses of our algebra but (partially) fails to fulfill the modularity requirement. We are looking for a sufficient condition on programs which would ensure this modularity/associativity property for a larger class of programs than those coming from proof-nets. One direction under exploration (suggested in [Gir95b]) is that of an untyped calculus whose computations would be performed in the algebra through Weak Execution.

References

[BP99] P. Baillot and M. Pedicini. Elementary complexity and geometry of interaction. Prepublication, Quaderni dell'Istituto per le Applicazioni del Calcolo "M.Picone" di Roma, IAC-CNR n. 21-998, January 1999.

[Dan90] V. Danos. *La Logique Linéaire appliquée à l'étude de divers processus de normalisation (principalement du λ-calcul).* PhD thesis, Université Paris VII, Juin 1990.

[DR95] V. Danos and L. Regnier. Proof nets and the Hilbert space. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, pages 307–328. Cambridge University Press, 1995. London Mathematical Society Lecture Note Series 222, Proceedings of the 1993 Workshop on Linear Logic, Cornell University, Ithaca.

[Gir88a] J.-Y. Girard. Geometry of interaction I: an interpretation of system F. In Ferro and al, editors, *Proceedings of A.S.L. Meetings*, Padova, 1988. North-Holland.

[Gir88b] J.-Y. Girard. Geometry of interaction II: Deadlock free algorithms. In Martin-Löf and Mints, editors, *Proceedings of COLOG'88*, volume 417 of *Lecture Notes in Computer Science*, pages 76–93. Springer Verlag, 1988.

[Gir95a] J.-Y. Girard. Geometry of interaction 3: the general case. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, pages 329–389. Cambridge University Press, 1995. London Mathematical Society Lecture Note Series 222, Proceedings of the 1993 Workshop on Linear Logic, Cornell University, Ithaca.

[Gir95b] J.-Y. Girard. Light linear logic. In D. Leivant, editor, *Proceedings of the International Workshop on Logic and Computational Complexity (LCC'94)*, volume 960 of *LNCS*, pages 145–176, Berlin, GER, October 1995. Springer.

[KOS97] M. I. Kanovich, M. Okada, and A. Scedrov. Phase semantics for light linear logic (extended abstract). In *Mathematical foundations of programming semantics (Pittsburgh, PA, 1997)*, volume 6 of *Electron. Notes Theor. Comput. Sci.* Elsevier, Amsterdam, 1997.

[MR91] P. Malacaria and L. Regnier. Some results on the interpretation of λ-calculus in operator algebras. In *Proceedings of the Sixth Annual Symposium on Logic in Computer Science*, pages 63–72. IEEE Computer Society Press, 1991.

[Ped96] M. Pedicini. Remarks on elementary linear logic. In *A Special Issue on the "Linear Logic 96, Tokyo Meeting"*, volume 3 of *Electronic Notes in Theoretical Computer Science*, Amsterdam, The Netherlands, 1996. Elsevier.

[Reg92] L. Regnier. *Lambda-Calcul et réseaux.* PhD thesis, Université Paris VII, 1992.