Fundamenta Informaticae XX (2000) 1–31
IOS Press

Elementary complexity and geometry of interaction

Patrick Baillot *
Université Aix-Marseille II
CMI (LIM-FRE 2246)
39 rue Joliot-Curie,
13453 Marseille Cedex 13, France
baillot@cmi.univ-mrs.fr

Marco Pedicini †
Istituto per le Applicazioni del Calcolo "Mauro Picone"
Consiglio Nazionale delle Ricerche
Viale del Policlinico 137
00161 Roma, Italy
marco@iac.rm.cnr.it

Abstract. We introduce a geometry of interaction model given by an algebra of clauses equipped with resolution (following [10]) into which proofs of Elementary Linear Logic can be interpreted. In order to extend geometry of interaction computation (the so called *execution formula*) to a wider class of programs in 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.

Keywords: Elementary Linear Logic, Geometry of interaction, Proof-nets, Complexity,Semantics.

*This work was mainly done while this author was at Institut de Mathématiques de Luminy (CNRS), Marseille, and completed at Laboratory for Foundations of Computer Science, University of Edinburgh, under TMR "LINEAR" research network grant FMRX-CT98-0170. Address for correspondence: Université Aix-Marseille II, CMI (LIM-FRE 2246), 39 rue Joliot-Curie, 13453 Marseille Cedex 13, France
†Partially supported by TMR "LINEAR" research network. Address for correspondence: Istituto per le Applicazioni del Calcolo "Mauro Picone", Consiglio Nazionale delle Ricerche, Viale del Policlinico 137, 00161 Roma, Italy

1. Introduction

Geometry of interaction (GOI) was introduced by Girard ([6, 8]) as a semantics of computation which:

- on the one hand, in contrast to denotational semantics interprets explicitly the dynamics of computation and handles finite objects,
- on the other hand, expresses this dynamics by more *mathematical* means than syntactical rewriting.

Various frameworks have been used to describe GOI models, including bounded operators on Hilbert spaces ([6, 4]), partial applications ([2, 16]) and algebras of clauses ([10]). This latter point of view is the one we will adopt here. In these models, the operation corresponding to the normalization process is called EXECUTION. It is not defined on all operators and sufficient conditions have been given which ensure convergence of its computation: it has been shown in the case of second-order Linear Logic ([6, 12]) and of untyped lambda-calculus [14]) that operators coming from the syntax satisfy such conditions (a nilpotency condition, for instance, in the case of Linear Logic). Let us recall that the result of EXECUTION on operators interpreting proofs is not in general an invariant of cut-elimination, though this holds provided certain conditions on the conclusions of the proof are satisfied (it is the case for instance with the type of booleans; see [6]).

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. [12]) in order to control the size explosion of proofs during normalization. It is obtained by removing the two principles: $!A \vdash A$ (*dereliction*) and $!A \vdash !!A$ (*digging*); 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 [15]).

An ELL 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 recursive: the expression of this function is a repeated exponential whose height only depends on the depth (see [15]). This property is a consequence of the preservation of depth by normalization steps: the depth of an edge is unchanged through any normalization step (but of course the edge might be duplicated or erased).

Note that it is not known whether all elementary functions can be represented within this version of ELL. An alternative approach has been carried on by Danos and Joinet (in [3]) who described ELL as a subsystem of full linear logic defined through a syntactical constraint on (LL) proof-nets (the *stratification condition*): this way they incorporate suitably the additives (keeping the isomorphism between $!(A\&B)$ and $!A\otimes !B$) and establish a representation theorem for elementary recursive functions. However, as the treatment of additives in geometry of interaction is delicate we choose to content ourselves here with the multiplicative exponential (second order) fragment: for this fragment our presentation is equivalent to that of Danos and Joinet.

Finally, recall that the main drawback of ELL (as well as of LLL) stressed by Girard in [12] was the lack of a specific semantics of proofs (though a phase semantics has been given by Kanovitch *et al.*, see [13]). We address the problem with the help of GOI , considering the semantics of reduction as an intermediary step between syntax and denotational semantics. We are looking for a GOI model such that all its elements can be considered as interpretations of algorithms terminating within elementary time, even if they are not realized by any proof (think of incorrect proof-structures or of programs using fixed points). Recently, an other approach based on coherent spaces has been proposed by the first author ([1]).

Achievements and limits of the present work: we present here an algebra of clauses along the lines of [10] with a kind of depth-preservation property analogous to that of ELL. 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 [10], section 2.3. In addition to usual EXECUTION we define a WEAK EXECUTION which amounts to giving up the computation of certain products of the EXECUTION. WEAK EXECUTION coincides with usual EXECUTION for operators coming from proof-nets.

A size and a depth are defined for general operators respectively as the number of clauses and the maximal arity of the predicates of the terms (actually all predicates have the same arity). Our main result is then that WEAK EXECUTION always terminates (there is no need for a nilpotency 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. In other words, 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 semantic setting.

Yet this WEAK EXECUTION presents a serious drawback as it is not in general an associative operation... However 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 6 for a precise statement).

Organisation of the paper: in the next section we introduce GOI through an example. Sections 3 and 4 are devoted to a presentation of the algebra, to the definition of EXECUTION and WEAK EXECUTION and to the statement of the main theorem. This result is then proved in section 5 and we examine in section 6 the sub-associativity property. The rest of the paper is devoted to the interpretation of proof-structures in the model and to the proof that WEAK EXECUTION yields the same result as usual execution when applied to operators coming from proof-nets.

Acknowledgements: the authors wish to thank Jean-Yves Girard for important suggestions and for pointing out the crucial lemma 5.3. They are also grateful to the anonymous referees for their detailed corrections and comments.

2. Geometry of interaction: a toy-example

Before getting into the technicalities of the GOI model we introduce for ELL, let us try to illustrate the general ideas at work in GOI on a toy-example. Consider the linear lambda-term $t = (\lambda x \lambda y(x)y)\lambda zz$; it can be translated into multiplicative linear logic, yielding the proof-net R of figure 1. The purpose of GOI is to provide an algebraic setting to describe the computation within a syntactical system (beta-reduction or cut-elimination). Here, the term t beta-reduces to $I = \lambda y\ y$, the proof-net R normalizes to R'. The model should be equipped with an operation enabling to retrieve these results: we want a map from the set of proof-nets to the model and an operation $\mathtt{Ex}(.)$ in the model, such that if R is mapped to R^o then $\mathtt{Ex}(R^o)$ and R'^o are equal (or at least computationally equivalent).

Concretely, we associate weights valued in the model to certain paths of the proof-net and compute all the products of these weights yielding non-zero result. The mentioned paths are paths linking external links (those premises of a cut or conclusions) and crossing one axiom.

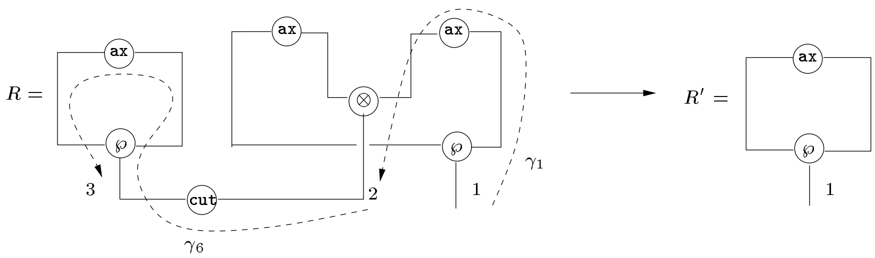

Figure 1. Example

For this system (linear lambda-calculus or multiplicative linear logic) we can use a very basic GOI model, operations on a stack built over an alphabet of size 2, $\{p, q\}$. More precisely, its elements are tuples defined by

- a source and a target among the previous external links, here numbered 1 to 3,
- an operation on stacks of the kind: "$\mathtt{pop}\, s$, then $\mathtt{push}\, t$",

written as $(source, \mathtt{pop}\, s, \mathtt{push}\, t, target)$. To this we add one extra element with no specified source and target, denoted by 0, meant to correspond to the operation with empty domain. Any invalid composition of operations (non matching consecutive target and source, or resulting operation defined nowhere) yields 0.

In our example the proof-net R is interpreted by the following set of weights:

$$\begin{array}{ll} \phi_1 = (1, \mathtt{pop}\, q, \mathtt{push}\, q, 2) & \phi_4 = (3, \mathtt{pop}\, q, \mathtt{push}\, q, 1) \\ \phi_2 = (3, \mathtt{pop}\, p, \mathtt{push}\, p, 1) & \phi_5 = (1, \mathtt{pop}\, p, \mathtt{push}\, p, 2) \\ \phi_3 = (2, \mathtt{pop}\, p, \mathtt{push}\, q, 3) & \phi_6 = (2, \mathtt{pop}\, q, \mathtt{push}\, p, 3). \end{array}$$

For instance ϕ_1 and ϕ_6 are the weights associated respectively to the paths γ_1 and γ_6 of the figure. Here are two examples of compositions: $\phi_1 \cdot \phi_6 = (1, \mathtt{pop}\, q, \mathtt{push}\, p, 3)$, $\phi_1 \cdot \phi_4 = 0$.

The *execution result* of such a system of weights is the set of non-zero products with source and target corresponding to conclusions (so in the present case they should be equal to 1). In this example we obtain two weights:

$$\begin{array}{rcl} (1, \mathtt{pop}\, p, \mathtt{push}\, q, 1) & = & \phi_5 \cdot \phi_3 \cdot \phi_4, \\ (1, \mathtt{pop}\, q, \mathtt{push}\, p, 1) & = & \phi_1 \cdot \phi_6 \cdot \phi_2. \end{array}$$

This is the interpretation of the result R' (or the lambda-term I).

3. Resolution Algebra

In this section we define a GOI model more elaborate than the one of the previous section. It will be expressive enough to interpret proofs of ELL and at the same time simple enough to enable us to establish an internal complexity bound. It is based on the algebra of clauses introduced in [10]: elements are clauses and they are composed by resolution. We recall the definitions of this general setting before describing the particular case we consider in the present 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.

A *substitution* θ is a partial application from the set of variables to T with finite domain. If $\mathtt{dom}(\theta) = \{x_1,\ldots,x_n\}$ and if $t_i = \theta(x_i)$ for $1 \leq i \leq n$ we will denote θ by $\langle t_1/x_1,\ldots,t_n/x_n\rangle$. The application θ is extended into two applications $T \to T$ and $\mathcal{L} \to \mathcal{L}$ (also denoted by θ) in the following way: the image of an *expression* (term or atom) e by the substitution θ is obtained by replacing in e each occurrence of a variable x in $\mathtt{dom}(\theta)$ by $\theta(x)$; this image will be denoted by $e\theta$. The composition of two such applications θ_1, θ_2 will be denoted by $\theta_1\theta_2$: $e(\theta_1\theta_2) = (e\theta_1)\theta_2$. A *renaming* of the variables of an expression e is an injection θ from the set of variables of e to the set of variables; the *renamed expression* is $e\theta$.

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 $e\theta = e'\theta$. In such a case θ is called a *unifier* of e and e'.

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 of the language $\mathcal{L}$ is a sequent

$$P_i(t_0,\ldots,t_m) \vdash P_j(u_0,\ldots,u_n),$$

where $P_i(t_0,\ldots,t_m)$ and $P_j(u_0,\ldots,u_n)$ are atoms of $\mathcal{L}$ with the same variables $x_0,\ldots,x_d$.

The *head* of the clause ϕ is the atom

$$\mathtt{head}(\phi) := P_i(t_0,\ldots,t_m),$$

its *tail* is the atom

$$\mathtt{tail}(\phi) := P_j(u_0,\ldots,u_n).$$

We then consider the set of clauses up to the following equivalence relation:

$\phi \equiv \phi'$ if there exists a renaming θ of the variables of ϕ such that $\mathtt{head}(\phi)\theta = \mathtt{head}(\phi')$ and $\mathtt{tail}(\phi)\theta = \mathtt{tail}(\phi')$.

From now we will mean by "clause" an equivalence class w.r.t the equivalence relation $\equiv$ (this way clauses are implicitly universally quantified). To this set of classes we add a formal clause 0 and we denote the resulting set by $\mathcal{C}$.

Definition 3.1. (Resolution)
Given two clauses ϕ and ϕ' we can assume they have disjoint variables (by choosing appropriate representatives). If $\mathtt{tail}(\phi)$ is comparable with $\mathtt{head}(\phi')$ and θ is their m.g.u. we define the *resolution* of the two clauses as the equivalence class of the clause

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

this class doesn't depend on the previous choice of the representatives.

Otherwise, if $\mathtt{tail}(\phi)$ and $\mathtt{head}(\phi')$ 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.

More generally, as unification is used here to perform resolution between equivalence classes of clauses we will from now on consider that the expressions we compare (terms, atoms) have disjoint variables; when it is not the case we rename suitably their variables and unify the two new expressions. For instance, if p is a unary symbol of function the two terms px and x are said to be unifiable as the renamed terms px and y admit a unifier: $\theta = \langle px/y \rangle$.

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. $\texttt{head}(\phi) \perp \texttt{tail}(\phi)$). Notice that this holds only because we consider clauses where head and tail share the same variables.

Definition 3.2. (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 $\mathbb{C}$, the set of complex numbers, 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, denoted by 1:

$$1 = \sum_{i \in I} P_i(x_0, \ldots, x_n) \vdash P_i(x_0, \ldots, x_n),$$

- 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 [10].

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 $\mathbb{N}$, then the combinations can be considered as multisets and we write $U \subseteq V$ if for all i, $\alpha_i \leq \beta_i$. We write $\phi_i \in U$ if its coefficient α_i is nonzero. A *projection* of the algebra is a combination U such that $U^\star = U$ and $U^2 = U$.

Definition 3.3. (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. Then the *execution* of the loop (U, σ) is the element

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

and the *result* of the execution is given by

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

Remark 3.4. Another way to write the execution is directly as a sum of clauses:

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

To understand the meaning of execution w.r.t. normalization of proof-nets, recall the example of section 2. The wiring U corresponds to the paths linking external links of the proof-net and the wiring σ corresponds to the connections given by the cuts: this analogy will be made precise in section 7 where we give the interpretation of proof-structures in this algebra. Computing the execution then amounts to determining the paths of the proof-structure with non-zero weight. In the result of the execution we only keep the weights of those paths which are maximal and start and end in conclusions (not cut premises), that is to say that we restrict to the visible part of the computation and forget about the intermediary steps...

Now we specify the particular language we are going to consider.

Definition 3.5. Let the *language of unary terms* T be the language built over a set of unary symbols of function $\{p, q, r, s\}$; hence 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 3.6. Notice that due to the particular form of terms, 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 the following cases can occur: $t \perp u$ or $t \le u$ or $t \ge u$, where $t \le u$ means that u is the unchanged term.

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

$$P_i(t_0[x_0], \ldots, t_d[x_d]) \vdash P_j(u_0[x_0], \ldots, u_d[x_d]),$$

where $\mathtt{head}(\phi)$ and $\mathtt{tail}(\phi)$ belong to $T^d \cdot m$. Notice that t_k and u_k ($k \in \{0, \ldots, d\}$) 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 it is the algebra we consider.

4. Weak Execution

In order to express a bound on the lengths of computation, we need first to define some measures on clauses and combinations of clauses.

[1] The choice of $d+1$ is done to keep the same notations when we interpret proof-structures, see section 7.

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

Given a clause $\phi = P(t_0, \ldots, t_d) \vdash P'(u_0, \ldots, u_d)$ 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

$$||w|| := \sup_{1 \leq i \leq n} (||\phi_i||).$$

We also define the *cardinality* of w: $N(w) := \#\{\phi_i \mid 1 \leq i \leq n\}$.

Example: consider in $\mathcal{C}^0$ the clause $\phi = P(x) \vdash P(rx)$ and let w_n be the word $(\phi, \ldots, \phi)$ 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 4.1. (Acyclicity)
An *acyclic clause* is a clause

$$\phi = P(t_0, \ldots, t_d) \vdash P'(u_0, \ldots, u_d)$$

such that $P \neq P'$, 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* is a word $(\phi_1, \ldots, \phi_n)$ such that every sub-product ψ is either an acyclic clause or a projection.

A word is *strictly acyclic* if and only if all its sub-products are acyclic clauses.

Example: consider the clauses $\phi_1 = P(sx_0, x_1) \vdash P(rx_0, x_1)$, $\phi_2 = P(rrx_0, rrx_1) \vdash P(sx_0, rx_1)$ and $\phi_3 = P(sx_0, rx_1) \vdash P(rx_0, sx_1)$ in $\mathcal{C}^1$. 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 sub-product $\phi_1 \cdot \phi_2 = P(srx_0, rrx_1) \vdash P(sx_0, rx_1)$ is not an acyclic clause. Note also that an acyclic clause is a null-square but that a null-square isn't necessarily acyclic (*e.g.* $P(x_0, rx_1) \vdash P(rx_0, sx_1)$).

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 combinations (U, σ); theorem 4.6 will establish the fact that this definition always makes sense (i.e. the sum is finite).

Definition 4.2. (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:

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

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

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

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

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

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 4.3. Note that in cases where $\mathtt{Ex}_{\sigma}(U)$ makes sense ((U, σ) is a convergent loop), 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 clause product of an acyclic word (Proposition 4.4). In the case of a strictly acyclic word this implies that the length of the word cannot exceed a certain bound (depending on its cardinality and width) without yielding zero as result. This bound will be expressed as an exponential tower of height d (Proposition 4.5).

Proposition 4.4. Given an acyclic word $w = (\phi_1, \ldots, \phi_n)$ with non-null product, we have the following inequality: $||\phi_1 \cdot \ldots \cdot \phi_n|| \leq L(||w||N(w), d)$, where L is defined by

$$L(a, d) := \begin{cases} 2_d^{4a(d+1)^2} & d \geq 1 \\ 8a & d = 0 \end{cases}$$

and the repeated exponential is defined by: $2^m_0 = m$ and $2^m_{N+1} = 2^{2^m_N}$

Note that $||\phi_1 \cdot \ldots \cdot \phi_n||$ 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.

This proposition will be proved in section 5. The result relies of course on the fact that w is acyclic. Otherwise given a fixed width (of word) and cardinality, one might exhibit non acyclic words whose products are of arbitrary large width: see for instance the first example given above, 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.

Proposition 4.5. Given two integers $a \geq 1$ and s we define

$$B(a, s) := 2_{s+1}^{9 \cdot a(s+1)^2}.$$

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

This result is a consequence of the proposition 4.4 and its proof will be given in section 5.1. We now give 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 4.6. Given a loop (U, σ), let $N = \#\sigma U$ and $k = 1 + \max\{||\phi|| \,/\, \phi \in \sigma U\}$. We have

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

where A' is the set given in definition 4.2.

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

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

where A is the set given in definition 4.2.

Proof: [Proof of theorem 4.6] 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 $n > B(kN, d)$ then $n > B(||w'||N(w'), d)$ and we know by proposition 4.5 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 $n \leq B(kN, d)$. □

5. Proof of Proposition 4.4

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 new 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 k in $\{0, \ldots, d-1\}$, 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 of $\mathcal{C}^k$:

$$[\phi]_{(0,k)} := P(t_0, \ldots, t_k) \vdash P'(u_0, \ldots, u_k).$$

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

Proof: [Proof of proposition 4.4] We prove the proposition by means of an intermediate inequality, namely we will prove by induction on d the following one:

$$||\phi_1 \cdot \ldots \cdot \phi_n|| \leq L'(||w||N(w), d) \tag{5.1}$$

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

$$\begin{cases} L'(a,0) = 2a \\ L'(a,s+1) = 2a2^{4(s+1)L'(a,s)} \end{cases} \tag{5.2}$$

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 $\mathcal{C}^0$. If $P(t)$ and $Q(u)$ are atoms in $T^0 \cdot m$ then we say $P(t) \leq Q(u)$ if $P = Q$ and $t \leq u$.

Notice that if $\phi \cdot \psi \neq 0$ then

$$\mathtt{tail}(\psi) \leq \mathtt{tail}(\phi\psi) \text{ and } \mathtt{head}(\phi) \leq \mathtt{head}(\phi\psi).$$

Lemma 5.1. 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 $|\mathtt{head}(\phi\psi)| \leq |\mathtt{head}(\psi)| + |\mathtt{head}(\phi)|$.

Proof: In the first case $\mathtt{head}(\phi\psi) = \mathtt{head}(\phi)$, in the second one $\mathtt{head}(\phi\psi) = \mathtt{head}(\phi)\theta$ where

$$\theta := m.g.u.(\mathtt{tail}(\phi), \mathtt{head}(\psi)) = \langle u[x]/x_\phi, x/x_\psi \rangle,$$

and so $|u[x]| \leq |\mathtt{head}(\psi)|$. □

Remark 5.2. Given a word $w = (\phi_1, \ldots, \phi_n)$ with non-null product, let us denote

$$\{j_1, \ldots, j_m\} = \{j \geq 2 \mid \mathtt{tail}(\phi_1 \cdot \ldots \cdot \phi_{j-1}) < \mathtt{head}(\phi_j)\}.$$

By induction over the integer m we deduce from the previous lemma the following inequality:

$$|\mathtt{head}(\phi_1 \cdot \ldots \cdot \phi_n)| \leq |\mathtt{head}(\phi_1)| + \sum_{i=1}^{m} |\mathtt{head}(\phi_{j_i})|;$$

analogously for $\mathtt{tail}(\phi_1 \cdot \ldots \cdot \phi_n)$.

Lemma 5.3. An acyclic word $w = (\phi_1, \ldots, \phi_n)$ with non-null product $\psi = \phi_1 \cdot \ldots \cdot \phi_n$, 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 5.2, using the same notations we have that $|\mathtt{head}(\phi_1 \cdot \ldots \cdot \phi_n)| \leq (m + 1)||w||$; 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 $\phi_{j_{i_1}} \cdot \ldots \cdot \phi_{j_{i_2}-1}$ gives a cyclic clause, hence the contradiction with the acyclicity of w. Indeed: let us denote $\Pi' = \phi_1 \cdot \ldots \cdot \phi_{j_{i_1}-1}$ and $\Pi'' = \phi_{j_{i_1}+1} \cdot \ldots \cdot \phi_{j_{i_2}-1}$; 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

$$||w||(N(w)+1) \leq 2||w||N(w) = L'(||w||N(w), 0),$$

as $N(w) \geq 1$.

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

Lemma 5.4. 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 only and

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

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

$$\mathtt{tail}(\phi_i)\sigma_i = \mathtt{head}(\phi_{i+1})\sigma_{i+1} \text{ when } 1 \leq i \leq n-1 \tag{5.3}$$

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

$$(\sigma_1, \ldots, \sigma_n) = (\sigma_1^0\theta, \ldots, \sigma_n^0\theta).$$

Proof: We start by proving the existential part of the first claim by induction over n. If $n = 1$, $\sigma_1 = \langle x/x \rangle$ satisfies the property.

Now assume the property is satisfied for $n \geq 1$ and let us prove it for $n+1$:
there exists a family $(\sigma_1^0, \ldots, \sigma_n^0)$ such that

$$\phi_1{\cdot}\ldots{\cdot}\phi_n = \mathtt{head}(\phi_1)\sigma_1^0 \vdash \mathtt{tail}(\phi_n)\sigma_n^0,$$

- first case: $\mathtt{tail}(\phi_n)\sigma_n^0 \geq \mathtt{head}(\phi_{n+1})$.

  There exists σ_{n+1}^0 such that $\mathtt{tail}(\phi_n)\sigma_n^0 = \mathtt{head}(\phi_{n+1})\sigma_{n+1}^0$ and we have:

  $$\phi_1{\cdot}\ldots{\cdot}\phi_{n+1} = \mathtt{head}(\phi_1)\sigma_1^0 \vdash \mathtt{tail}(\phi_{n+1})\sigma_{n+1}^0.$$

  Moreover the second condition (5.3) is satisfied by $(\sigma_1^0, \ldots, \sigma_{n+1}^0)$ with respect to $\phi_1, \ldots, \phi_{n+1}$.

- second case: $\mathtt{tail}(\phi_n)\sigma_n^0 < \mathtt{head}(\phi_{n+1})$.

  There exists τ such that $\mathtt{tail}(\phi_n)\sigma_n^0\tau = \mathtt{head}(\phi_{n+1})$ and we have:

$$\begin{array}{rcl}\phi_1\cdot\ldots\cdot\phi_{n+1} & = & (\phi_1\cdot\ldots\cdot\phi_n)\cdot\phi_{n+1}\\ & = & (\mathtt{head}(\phi_1)\sigma_1^0 \vdash \mathtt{tail}(\phi_n)\sigma_n^0)\cdot\phi_{n+1}\\ & = & \mathtt{head}(\phi_1)\sigma_1^0\tau \vdash \mathtt{tail}(\phi_{n+1}).\end{array}$$

  and the second condition (5.3) is satisfied by $(\sigma_1^0\tau, \ldots, \sigma_n^0\tau, \langle x/x\rangle)$.

Now we check the uniqueness part of the first claim: assume $(\sigma_1^0, \ldots, \sigma_n^0)$ and $(\tau_1^0, \ldots, \tau_n^0)$ both satisfy condition (5.3) and

$$\begin{array}{rcl}\phi_1\cdot\ldots\cdot\phi_n & = & \mathtt{head}(\phi_1)\sigma_1^0 \vdash \mathtt{tail}(\phi_n)\sigma_n^0\\ & = & \mathtt{head}(\phi_1)\tau_1^0 \vdash \mathtt{tail}(\phi_n)\tau_n^0.\end{array}$$

Then $\mathtt{head}(\phi_1)\sigma_1^0 = \mathtt{head}(\phi_1)\tau_1^0$ implies $\sigma_1^0 = \tau_1^0$ and consequently $\mathtt{tail}(\phi_1)\sigma_1^0 = \mathtt{tail}(\phi_1)\tau_1^0$. Using (5.3) for σ_1^0 and τ_1^0 ($i = 1$) we get $\mathtt{head}(\phi_2)\sigma_2^0 = \mathtt{head}(\phi_2)\tau_2^0$ and again $\sigma_2^0 = \tau_2^0$. Applying this method inductively we eventually conclude with $(\sigma_1^0, \ldots, \sigma_n^0) = (\tau_1^0, \ldots, \tau_n^0)$.

As to the second statement, we prove it again by induction over n. The base case is trivial. Assume given $n \geq 1$ the result is true for any $(\phi_1, \ldots, \phi_n)$ and $(\sigma_1, \ldots, \sigma_n)$ satisfying the hypothesis, and consider the case of $(\phi_1, \ldots, \phi_{n+1})$ and of a family $(\sigma_1, \ldots, \sigma_{n+1})$ satisfying (5.3).

Let $(\sigma_1^0, \ldots, \sigma_n^0)$ be the family of substitutions induced by the product of the word $(\phi_1, \ldots, \phi_n)$ according to the first statement. As (5.3) ($1 \leq i \leq n$) is satisfied by $(\phi_1, \ldots, \phi_n)$ and $(\sigma_1, \ldots, \sigma_n)$, by induction hypothesis there exists θ such that for $1 \leq i \leq n$: $\sigma_i = \sigma_i^0\theta$.

Now we consider the two cases we distinguished previously:

- first case: $\mathtt{tail}(\phi_n)\sigma_n^0 \geq \mathtt{head}(\phi_{n+1})$

  There exists σ_{n+1}^0 such that $\mathtt{tail}(\phi_n)\sigma_n^0 = \mathtt{head}(\phi_{n+1})\sigma_{n+1}^0$ and the family induced by the product of $(\phi_1, \ldots, \phi_{n+1})$ is $(\sigma_1^0, \ldots, \sigma_{n+1}^0)$.

  Now,

$$\begin{array}{rcl}\mathtt{head}(\phi_{n+1})\sigma_{n+1} & = & \mathtt{tail}(\phi_n)\sigma_n\\ & = & \mathtt{tail}(\phi_n)\sigma_n^0\theta\\ & = & \mathtt{head}(\phi_{n+1})\sigma_{n+1}^0\theta\end{array}$$

  and so $\sigma_{n+1} = \sigma_{n+1}^0\theta$, $(\sigma_1, \ldots, \sigma_{n+1}) = (\sigma_1^0\theta, \ldots, \sigma_{n+1}^0\theta)$.

- second case: $\mathtt{tail}(\phi_n)\sigma_n^0 < \mathtt{head}(\phi_{n+1})$

  There exists τ such that $\mathtt{tail}(\phi_n)\sigma_n^0\tau = \mathtt{head}(\phi_{n+1})$, and the family induced by the product of $(\phi_1, \ldots, \phi_{n+1})$ is

$$(\sigma_1^0\tau, \ldots, \sigma_n^0\tau, \langle x/x\rangle) = ({\sigma'}_1^0, \ldots, {\sigma'}_{n+1}^0).$$

We have:

$$\begin{aligned} \mathtt{head}(\phi_{n+1})\sigma_{n+1} &= \mathtt{tail}(\phi_n)\sigma_n \\ &= \mathtt{tail}(\phi_n)\sigma_n^0\theta \end{aligned}$$

and

$$\mathtt{head}(\phi_{n+1})\sigma_{n+1} = (\mathtt{tail}(\phi_n)\sigma_n^0\tau)\sigma_{n+1},$$

therefore $\theta = \tau\sigma_{n+1}$ and

$$\begin{aligned} (\sigma_1, \dots, \sigma_n, \sigma_{n+1}) &= (\sigma_1^0\theta, \dots, \sigma_n^0\theta, \sigma_{n+1}) \\ &= (\sigma_1^0\tau\sigma_{n+1}, \dots, \sigma_n^0\tau\sigma_{n+1}, \sigma_{n+1}) \\ &= ((\sigma_1^0\tau)\sigma_{n+1}, \dots, (\sigma_n^0\tau)\sigma_{n+1}, \langle x/x\rangle\sigma_{n+1}) \\ &= ({\sigma'}_1^0\theta', \dots, {\sigma'}_{n+1}^0\theta') \end{aligned}$$

where $\theta' = \sigma_{n+1}$.

So in both cases the result is true for $(\phi_1, \dots, \phi_{n+1})$ and $(\sigma_1, \dots, \sigma_{n+1})$, the induction step is proved and we can conclude by induction. □

Remark 5.5. Note that for $2 \le i \le n-1$ we have $\mathtt{tail}(\phi_i)\sigma_i^0 = \mathtt{head}(\phi_{i+1})\sigma_{i+1}^0$ and that this term is equal to $\mathtt{tail}(\phi_1 \cdot \ldots \cdot \phi_i)$ if $\mathtt{tail}(\phi_1 \cdot \ldots \cdot \phi_i) \ge \mathtt{head}(\phi_{i+1} \cdot \ldots \cdot \phi_n)$, or to $\mathtt{head}(\phi_{i+1} \cdot \ldots \cdot \phi_n)$ otherwise.

Lemma 5.6. Let us consider a word $w = (\phi_1, \dots, \phi_n)$ with non-null product; the product of w induces a word $w' = (\phi'_1, \dots, \phi'_n)$ such that $\phi'_i = \phi_i\sigma_i^0$ $(1 \le i \le n)$ with σ_i^0 as in lemma 5.4. Then for every i and h, we have that the sub-product $\phi_i \cdot \ldots \cdot \phi_{i+h}$ is a projection if and only if the corresponding sub-product of w', $\phi'_i \cdot \ldots \cdot \phi'_{i+h}$ is a projection.

If $\phi_i \cdot \ldots \cdot \phi_{i+h}$ is a null-square then $\phi'_i \cdot \ldots \cdot \phi'_{i+h}$ is a null-square.

Proof: Easily checked using lemma 5.4. The family of substitutions $(\sigma_i^0, \dots, \sigma_{i+h}^0)$ satisfies condition (5.3) with respect to the word $(\phi_i, \dots, \phi_{i+h})$. So by lemma 5.4 there exists a family $(\tau_i^0, \dots, \tau_{i+h}^0)$ such that product $\phi_i \cdot \ldots \cdot \phi_{i+h} = \mathtt{head}(\phi_i)\tau_i^0 \vdash \mathtt{tail}(\phi_{i+h})\tau_{i+h}^0$ and a substitution θ such that $\sigma_j^0 = \tau_j^0\theta$ for $j = i, \dots, i+h$. Then we have $\phi'_i \cdot \ldots \cdot \phi'_{i+h} = \mathtt{head}(\phi_i)\tau_i^0\theta \vdash \mathtt{tail}(\phi_{i+h})\tau_{i+h}^0\theta$, and $\mathtt{head}(\phi_i)\tau_i^0\theta = \mathtt{tail}(\phi_{i+h})\tau_{i+h}^0\theta$ holds if and only if $\mathtt{head}(\phi_i)\tau_i^0 = \mathtt{tail}(\phi_{i+h})\tau_{i+h}^0$.

The second part of the proposition is obtained by the fact that:

$$\mathtt{head}(\phi_i)\tau_i^0 \perp \mathtt{tail}(\phi_{i+h})\tau_{i+h}^0 \text{ implies } \mathtt{head}(\phi_i)\tau_i^0\theta \perp \mathtt{tail}(\phi_{i+h})\tau_{i+h}^0\theta.$$

□

We establish now the induction step of inequality (5.1). Assume the inequality is true for any acyclic word in $\mathcal{C}^k$ with $k \le d$, and take a word $w = (\phi_1, \dots, \phi_n)$ over $\mathcal{C}^{(d+1)}$. Consider for every layer $[w]_k$ the induced family of substitutions: $(\sigma_1^k, \dots, \sigma_n^k)$. 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:

$$\begin{cases} [\phi'_i]_k &= [\phi_i]_k \sigma_i^k \langle a_k / x_k \rangle \qquad 0 \leq k \leq d, \\ [\phi'_i]_{d+1} &= [\phi_i]_{d+1}. \end{cases} \tag{5.4}$$

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 $(\phi'_i, \dots, \phi'_{i+h})$ and its product $\phi' := \phi'_i \cdot \ldots \cdot \phi'_{i+h}$; we denote the corresponding sub-product in w by $\phi := \phi_i \cdot \ldots \cdot \phi_{i+h}$. By lemma 5.6, 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 4.1), this implies that ϕ' is an acyclic clause.

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

$$||\phi'_1 \cdot \ldots \cdot \phi'_n|| \leq L'(||w'||N(w'), 0). \tag{5.5}$$

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 bound it we can calculate the number of all possible instances of terms in w'. Remark 5.5 tells us that $\mathtt{tail}([\phi'_i]_k)$ is equal to $\mathtt{tail}([\phi_1 \cdot \ldots \cdot \phi_i]_k \langle a_k/x_k \rangle$ or $\mathtt{head}([\phi_{i+1} \cdot \ldots \cdot \phi_n]_k \langle a_k/x_k \rangle$ (similarly for $\mathtt{head}([\phi'_i]_k)$). Moreover we have

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

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

$$\begin{aligned} ||[\phi_1 \cdot \ldots \cdot \phi_i]_{(0,k)}|| &\leq L'(||w||N(w), k), \\ ||[\phi_{i+1} \cdot \ldots \cdot \phi_n]_{(0,k)}|| &\leq L'(||w||N(w), k). \end{aligned}$$

Finally, we obtain

$$||[\phi'_i]_k|| \leq L'(||w||N(w), k).$$

Besides 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 $\sum_{k=0}^{l} 4^k \leq 4^{l+1}$. Therefore the number of terms in our language 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 $N(w)$; at 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 (5.5) we have

$$\begin{aligned} ||\phi'_1\cdot\ldots\cdot\phi'_n|| &\leq L'(||w'||N(w'),0) = 2||w'||N(w') \\ &\leq 2||w||N(w)\prod_{k=0}^{d} 2^{4(L'(||w||N(w),k)+1)} \\ &\leq 2||w||N(w)2^{4(d+1)L'(||w||N(w),d)} = L'(||w||N(w),d+1) \end{aligned} \tag{5.6}$$

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

$$||[\phi_1\cdot\ldots\cdot\phi_n]_{d+1}|| \leq L'(||w||N(w),d+1),$$

and by induction hypothesis we have:

$$||[\phi_1\cdot\ldots\cdot\phi_n]_{(0,d)}|| \leq L'(||w||N(w),d).$$

Since $L'(||w||N(w),d) \leq L'(||w||N(w),d+1)$, we get:

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

This ends our proof for the induction step and inequality (5.1) is established. $\square$

Let us conclude this section with the bound of the quantity $L'(||w||N(w),d)$ to a more readable expression, so that it will explicitly appear as an exponential tower of height d:

$$L'(a,d) = 2a\cdot 2^{\,d(8a)\cdot 2^{\,(d-1)(8a)\cdot 2^{\cdot^{\cdot^{\cdot^{2^{8a}}}}}}}$$

using inequalities $\alpha \leq 2^{\alpha}$ and $x+y \leq xy$ whenever $x \geq 2$ and $y \geq 2$, we have

$$\begin{aligned} L'(a,d) &\leq 2_d^{2a+8a\sum_{j=1}^{d} j} \\ &\leq 2_d^{2a(1+2d(d+1))} \\ &\leq 2_d^{4a(d+1)^2} \end{aligned} \tag{5.7}$$

5.1. Consequence of Proposition 4.4

Proof: [Proof of Proposition 4.5] Let us take $w = (\phi_1,\ldots,\phi_n)$ a strictly acyclic word. Let $(\sigma_1^k,\ldots,\sigma_n^k)$ be the family of substitutions induced by the product $[w]_k$, for any $k = 0,\ldots,d$. We consider the word $w' := (\psi_1,\ldots,\psi_n)$ such that $[\psi_i]_k := [\phi_i]_k\sigma_i^k$ for $k = 0,\ldots,d$ and $i = 1,\ldots,n$. By the same argument we already used in the proof of proposition 4.4, from lemma 5.6 we get that as w is strictly acyclic the word w' is also strictly acyclic.

By proposition 4.4 we have $||[w']_k|| \leq L(||w||N(w),k)$. We proceed as before:

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

$$
\begin{aligned}
&\leq \; N(w)2^{4\sum_{k=0}^{d}(L(||w||N(w),k)+1)}\\
&\leq \; N(w)2^{4(d+1)L(||w||N(w),d)}\\
&\qquad \text{(we applied } L(a,k)+2\leq L(a,d) \text{ when } k\leq d-1\text{)},\\
&\leq \; N(w)2^{4(d+1)\cdot 2_d^{4||w||N(w)(d+1)^2}}\\
&\leq \; N(w)2_{d+1}^{4||w||N(w)(d+1)^2+4(d+1)}\\
&\leq \; 2_{d+1}^{4||w||N(w)(d+1)^2+4(d+1)+N(w)}\\
&\leq \; 2_{d+1}^{9||w||N(w)(d+1)^2}
\end{aligned} \tag{5.8}
$$

the three last inequalities being obtained by using the same basic inequalities as in the computation in the proof of proposition 4.4.

Assume now the length n of w is such that $n > B(||w||N(w), d)$. As the length of w' is the same as the length of w, there exist $1 \leq i < j \leq n$ such that $\psi_i = \psi_j$. Let us now consider the sub-product $\pi = \psi_i \cdot \ldots \cdot \psi_{j-1}$, then $\texttt{head}(\pi) = \texttt{head}(\psi_i)$ and $\texttt{tail}(\pi) = \texttt{tail}(\psi_{j-1})$. But $\texttt{tail}(\psi_{j-1}) = \texttt{head}(\psi_j) = \texttt{head}(\psi_i)$ therefore $\texttt{head}(\pi) = \texttt{tail}(\pi)$. This means that for every layer k, $[\pi]_k$ is a projection, which contradicts the strict acyclicity of w'. So finally the length of w is bounded by $B(||w||N(w), d)$. $\square$

6. Sub-associativity of weak Execution

Let Δ, Δ', Γ be a partition of a set of indexes $\Delta \cup \Delta' \cup \Gamma$,[2] such that indexes in Δ can be assembled in pairs, noted dually as $(B, B^{\perp})$, and the ones in Δ' as well.

We consider the algebra $\lambda^{\star}(\Delta, \Delta', \Gamma)$ built, as in section 3, using the language of terms T defined over the set of unary symbols of function $\{p, q, r, s\}$ and the family of predicates of arity $(d+1)$: $\{P_A\}_{A\in\Delta\cup\Delta'\cup\Gamma}$.

Let

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

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

Proposition 6.1. (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:

$$\texttt{Result}^{\dagger}_{\sigma+\tau}(U) \subseteq \texttt{Result}^{\dagger}_{\tau}(\texttt{Result}^{\dagger}_{\sigma}(U)).$$

Remark 6.2. 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.

[2]in the sequel these indexes will vary over formulas of a sequent $\vdash \Gamma, \Delta, \Delta'$ of ELL, where formulas in Δ and Δ' are cut-formulas.

We give an example of a pair (U, σ) whose weak execution lacks compositional modularity; we consider two combinations of clauses:

$$U = \sum_{i=1}^{4} \psi_i \quad \text{where} \quad \begin{cases} \psi_1 & = P_A(sx_0) \vdash P_B(rx_0) \\ \psi_2 & = P_{B^\perp}(rx_0) \vdash P_C(sx_0) \\ \psi_3 & = P_{C^\perp}(ssx_0) \vdash P_B(rx_0) \\ \psi_4 & = P_{B^\perp}(rx_0) \vdash P_A(rx_0) \end{cases}$$

and

$$\sigma = \sigma_1 + \sigma_2, \quad \text{where} \quad \begin{cases} \sigma_1 = & \sigma_{\Delta;\Delta',A} \\ \sigma_2 = & \sigma_{\Delta';\Delta,A} \quad \text{with } \Delta = B, B^\perp \text{ and } \Delta' = C, C^\perp. \end{cases}$$

Then we have that in the computation of the weak execution, the word

$$(\phi_1, \phi_2, \phi_3, \phi_4)$$

where $\phi_1 = \psi_1$, $\phi_2 = \sigma_1 \cdot \psi_2$, $\phi_3 = \sigma_2 \cdot \psi_3$, $\psi_4 = \sigma_1 \cdot \psi_4$ is not included. Indeed, though its product

$$\phi_1 \cdot \phi_2 \cdot \phi_3 \cdot \phi_4 = P_A(ssx_0) \vdash P_A(rx_0)$$

is non-null and acyclic, there is a cyclic sub-product:

$$\phi_2 \cdot \phi_3 = P_B(rsx_0) \vdash P_B(rx_0).$$

So we have: $\texttt{Result}^\dagger_\sigma(U) = 0$, but $\texttt{Result}^\dagger_{\sigma_2}(\texttt{Result}^\dagger_{\sigma_1}(U)) = \{P_A(ssx_0) \vdash P_A(rx_0)\}$.

Proof: Let us first describe the result of the execution with this particular choice of hermitian wiring $\sigma_{\Delta,\Delta';\Gamma}$:

$$\begin{aligned} \texttt{Result}^\dagger_{\sigma_{\Delta,\Delta';\Gamma}}(U) &= (1 - \sigma^2_{\Delta,\Delta';\Gamma})\texttt{Ex}^\dagger_{\sigma_{\Delta,\Delta';\Gamma}}(U)(1 - \sigma^2_{\Delta,\Delta';\Gamma}) \\ &= \left\{ \Pi \in \texttt{Ex}^\dagger_{\sigma_{\Delta,\Delta';\Gamma}}(U) \;\middle|\; \begin{array}{l} \Pi \text{ is of the form} \\ P_A(t_0, \ldots, t_d) \vdash P_B(u_0, \ldots, u_d) \\ \text{where } A, B \in \Gamma \end{array} \right\}. \end{aligned}$$

Let Π be an element of $\texttt{Result}^\dagger_{\sigma+\tau}$, then it can be written as a product $\psi_0 \cdot \psi_1 \cdot \ldots \cdot \psi_n$ where the word $(\psi_0, \psi_1, \ldots, \psi_n)$ is a strictly acyclic word with:

- ψ_0 belongs to U,
- ψ_i belongs to $(\sigma + \tau)U$ for $1 \leq i \leq n$, which means that ψ_i belongs either to σU or to τU.
- the term $\texttt{head}(\psi_0)$ starts with a predicate P_A where $A \in \Gamma$; the term $\texttt{tail}(\psi_n)$ with a predicate P_B where $B \in \Gamma$.

Let $i_1, \ldots i_m$ denote the indexes of the clauses ψ_i such that ψ_i belongs to τU (or equivalently: the term $\mathtt{head}(\psi_i)$ starts with a predicate P_A where $A \in \Delta'$). For j belonging to $\{1, \ldots, m\}$ we introduce T_j and ψ'_{i_j} the respective elements of τ and U such that $\psi_{i_j} = T_j \cdot \psi'_{i_j}$.

Now, consider $1 \leq j \leq (m-1)$: by the fact that $(\psi_0, \psi_1, \ldots, \psi_n)$ is strictly acyclic, we know that the word $(\psi'_{i_j}, \psi_{i_j+1}, \ldots, \psi_{i_{j+1}-1})$ is strictly acyclic too and it has a non-null product, so the product $\psi'_{i_j} \cdot \psi_{i_j+1} \cdot \ldots \cdot \psi_{i_{j+1}-1}$ belongs to $\mathtt{Ex}^{\dagger}_{\sigma}(U)$.

Moreover $\psi'_{i_j} \cdot \psi_{i_j+1} \cdot \ldots \cdot \psi_{i_{j+1}-1}$ is of the shape $P_A(.) \vdash P_B(.)$ where $A, B \in \Delta'$, which insures by what precedes that it belongs to $\mathtt{Result}^{\dagger}_{\sigma}(U)$.

Similarly:

- the product $\psi_0 \cdot \ldots \cdot \psi_{i_1-1}$ belongs to $\mathtt{Ex}^{\dagger}_{\sigma}(U)$ and is of the shape $P_A(.) \vdash P_B(.)$ where $A \in \Gamma$ and $B \in \Delta'$, so it belongs to $\mathtt{Result}^{\dagger}_{\sigma}(U)$;
- the product $\psi'_{i_m} \cdot \ldots \cdot \psi_n$ belongs to $\mathtt{Ex}^{\dagger}_{\sigma}(U)$ and is of the shape $P_A(.) \vdash P_B(.)$ where $A \in \Delta'$ and $B \in \Gamma$, so it belongs to $\mathtt{Result}^{\dagger}_{\sigma}(U)$.

To simplify the notations we set $i_0 = 0$ and $i_{m+1} = n+1$. Let us then denote by Π_j the product $\psi_{i_j} \cdot \ldots \cdot \psi_{i_{j+1}-1}$ for $0 \leq j \leq m$.

We have shown that Π_0 belongs to $\mathtt{Result}^{\dagger}_{\sigma}(U)$ and Π_j belongs to $\tau\mathtt{Result}^{\dagger}_{\sigma}(U)$ for $1 \leq j \leq m$. By associativity of resolution we have:

$$\begin{aligned} \Pi &= \psi_0 \cdot \psi_1 \cdot \ldots \cdot \psi_n \\ &= (\psi_0 \cdot \ldots \cdot \psi_{i_1-1}) \cdot (\psi_{i_1} \cdot \ldots \cdot \psi_{i_2-1}) \cdot \ldots \cdot (\psi_{i_m} \cdot \ldots \cdot \psi_n) \\ &= \Pi_0 \cdot \Pi_1 \cdot \ldots \cdot \Pi_m \end{aligned}$$

In fact, $(\Pi_0, \Pi_1, \ldots, \Pi_m)$ has a non-null product and since the word $(\psi_0, \ldots, \psi_n)$ is strictly acyclic, we get that $(\Pi_0, \Pi_1, \ldots, \Pi_m)$ is also strictly acyclic. So its product Π belongs to $\mathtt{Ex}^{\dagger}_{\tau}(\mathtt{Result}^{\dagger}_{\sigma}(U))$.

Moreover we know that Π is of the shape $P_A(.) \vdash P_B(.)$ where $A, B \in \Gamma$, so Π belongs to $\mathtt{Result}^{\dagger}_{\tau}(\mathtt{Result}^{\dagger}_{\sigma}(U))$. □

7. Interpretation of ELL Proof-Structures

We consider second order elementary linear logic with t-promotion and without additives. Let us recall briefly the sequent calculus; the rules are given as in multiplicative exponential linear logic except for:

- dereliction which is not included in the system,
- the introduction of the !-modality (promotion): the new rule (t-promotion) acts as derelictions on the context formulas followed by the usual promotion.

- **Identity Group**

Axiom $$\frac{}{\vdash A^{\perp}, A}\ \mathbf{ax}$$ Cut $$\frac{\vdash A, \Delta \quad \vdash A^{\perp}, \Delta'}{\vdash \Delta, \Delta'}\ \mathbf{cut}$$

- **Logical Rules** *(multiplicatives and second-order quantifiers)*

Par $$\frac{\vdash A, B, \Delta}{\vdash A \wp B, \Delta}\ \wp$$

Times $$\frac{\vdash A, \Delta \quad \vdash B, \Delta'}{\vdash A \otimes B, \Delta, \Delta'}\ \otimes$$

Universal $$\frac{\vdash A, \Delta}{\vdash \forall \alpha A, \Delta}\ \forall$$

Existential $$\frac{\vdash A[C/\alpha], \Delta}{\vdash \exists \alpha A, \Delta}\ \exists$$

provided α is not free in Δ

- **Structural Rules**

Weakening $$\frac{\vdash \Delta}{\vdash ?A, \Delta}\ \textbf{we}$$

Contraction $$\frac{\vdash ?A, ?A, \Delta}{\vdash ?A, \Delta}\ \textbf{co}$$

t-Promotion $$\frac{\vdash A, \Delta}{\vdash !A, ?\Delta}\ \textbf{pro}$$

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*.

7.1. ELL Proof-Structures

They can be defined in the same way as ordinary (LL) proof-structures (introduced in [5, 9, 11]); the only difference is in the typing constraints of the exponential boxes (reflecting the multi-functoriality of t-promotion). We recall briefly this definition.

We consider labelled graphs (with pending edges, called *conclusions*) built over the nodes and typing constraints of Figure 2. We assume that each $\forall$-node binds a distinct propositional variable, its *eigenvariable* and that eigenvariables are not free in conclusions.

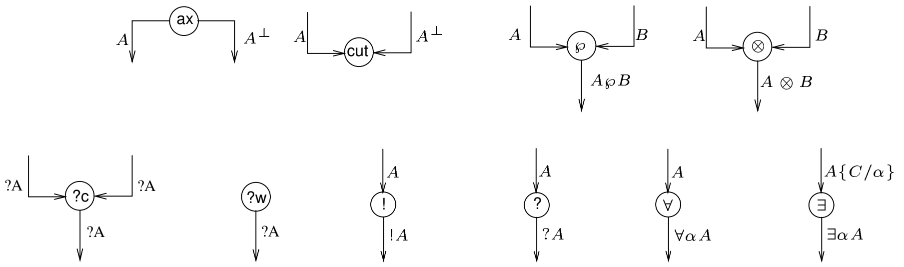

Figure 2. links in proof structures

To each $?w$-node we associate another node (distinct from a cut) of the graph (a *jump*).

A *box* of such a graph is a subgraph such that exactly one pending edge is premise of a !-node and the others (possibly none) are premises of ?-nodes (see Figure 3). The $!A$ (resp. $?B_i$) edge is called the *principal port* (resp. *auxiliary port*) of the box. Such a graph is an (ELL) proof-structure if any ! or ?-node (*box node*) is associated to a box and if given two distinct boxes, either they are disjoint or one is included in the other.

A node *depends on an eigenvariable* α if α is free in its conclusion, or if it is a $\exists$-node and α is free in the instantiating formula C of its premise.

The *depth* of a node (resp. an edge, a box) is the number of boxes it is contained in. The depth of the proof-structure is the maximal depth of its nodes.

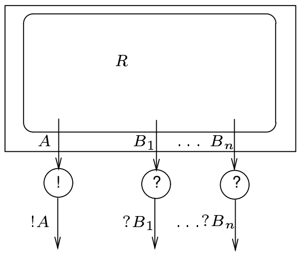

Figure 3. a box in a proof structure

Definition 7.1. A *switching graph* of a proof-structure R is defined in the following way:

- for each $\wp$- and $?c$-node we choose one premise and erase the other one;
- for each $?w$-node we add an edge between the node and its jump;
- for each $\forall$-node we either keep its premise or erase it and add an edge to a node depending on its eigenvariable;
- each box of depth 0 is erased and its conclusions are connected altogether.

A proof-structure is called a *proof-net* if all its switching graphs are connected and acyclic and if every proof-structure in a box at depth 0 is a proof-net (*correctness criterion).*

The following result is the statement of the sequentialization theorem ([9, 11]) in the case of ELL proof-structures (the proof is unchanged):

Theorem 7.2. (J.Y. Girard)
A proof-structure is *sequentializable* (i.e. comes from an ELL proof) iff it is a proof-net.

We consider in proof-structures *straight paths* (terminology of [4]) that is to say oriented paths crossing multiplicative, exponential and quantifier nodes either from a premise to the conclusion or from the conclusion to a premise, crossing axiom nodes (resp. cut nodes) from a conclusion (resp. premise) to the other conclusion (resp. premise) and not changing direction in the conclusions of the proof-structure.

By *path* we will now mean *straight path*. A path is *ascending* (resp. *descending*) 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 pairs $(B, B^{\perp})$ 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.

7.2. Interpretation of a proof-structure

For the sake of simplicity we will consider proof-structures with axioms labelled by atomic formulas.

Representation of a constant-depth path by a term: as they don't cross axiom or cut nodes, constant-depth paths are ascending or descending. 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 an ascending path by induction on the length of the path. In the case of a descending path the interpretation t_γ is that of the reverted ascending path (orientation will be taken into account when we introduce the clauses...).

- if γ is reduced to an edge premise of a box node or conclusion of the proof-structure, 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, contraction or quantifier node:
  - if γ_2 is the left (resp. right) premise of a multiplicative node then
  $$t_\gamma = t_{\gamma_1}\langle px_i/x_i\rangle \text{ (resp. } t_\gamma = t_{\gamma_1}\langle qx_i/x_i\rangle\text{)},$$
  - if γ_2 is the left (resp. right) premise of a contraction node then
  $$t_\gamma = t_{\gamma_1}\langle rx_i/x_i\rangle \text{ (resp. } t_\gamma = t_{\gamma_1}\langle sx_i/x_i\rangle\text{)},$$
  - if γ_2 is the premise of a quantifier node then
  $$t_\gamma = t_{\gamma_1}.$$

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:

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

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 an ascending (resp. descending) constant-depth 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 *weight* of γ is the clause $W(\gamma)$ given by:

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

(with the natural adjustment in the cases where $i = 0, j = d$ or $k = 0$. For instance if $i = 0$ and $j = d$ the head of the clause becomes: $P_A(t_{\gamma_0}, \dots, t_{\gamma_d})$).

See figure 4 for an example: we give the intermediary steps of computation of the weight of the path γ, for which we obtain $W(\gamma) = P_A(prx_0, px_1) \vdash P_A(qx_0, px_1)$.

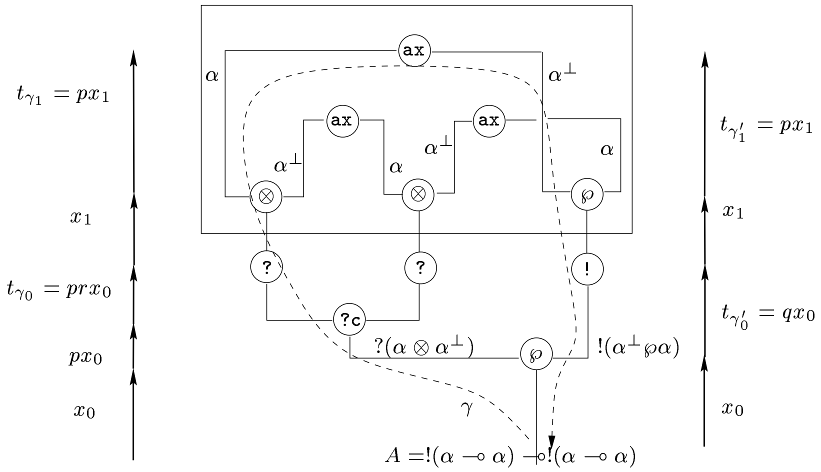

Figure 4. Computation of the weight of the path γ

Remark 7.3. Notice that if γ and γ' are two elementary paths starting in the same terminal formula and forking in a node at depth k we have: if $\texttt{head}(W(\gamma)) = P_A(t_0, \dots, t_d)$ and $\texttt{head}(W(\gamma')) = P_A(t'_0, \dots, t'_d)$ then for $i \leq k-1$ we have $t_i = t'_i$ and $t_k \perp t'_k$.

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

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

8. 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 4.3.

First we will prove a proposition (8.5) and then we will derive this result as a corollary (8.7). Let us give before a few definitions.

A *balanced path* of R is a path starting upwards in a conclusion or downwards in a cut premise, and ending downwards in a conclusion 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 $\sigma_0 = P_B(x_0, \dots, x_d) \vdash P_{B^\perp}(x_0, \dots, x_d)$ 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: $\gamma = \gamma_0; \dots; \gamma_n$ and its *weight* is given by the product

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

Definition 8.1. We say a clause $\phi = P(t_0, \dots, t_d) \vdash P'(u_0, \dots, u_d)$ 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 8.2. 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 (i.e. at depth $l \leq k$).

Proof: First, the fact that $W(\gamma)$ is cyclic implies that $\mathtt{head}(W(\gamma))$ and $\mathtt{tail}(W(\gamma))$ have the same predicate symbol and therefore that γ starts and ends in the same terminal formula A (either a cut formula or a conclusion). In order to get a contradiction assume γ doesn't cross any cut at depth lower than k. Let us write the decomposition of γ into elementary balanced paths: $\gamma = \gamma_1; \gamma_2; \dots; \gamma_n$. We denote the weights by $\phi_i = W(\gamma_i)$ and $\phi = W(\gamma)$.

- first case: A is a conclusion. We know that each γ_i with $2 \leq i \leq n$ starts in a cut formula at depth strictly greater than k and that each γ_i with $1 \leq i \leq (n-1)$ ends in a cut formula at depth strictly greater than k. Consequently for $2 \leq i \leq (n-1)$, the weight ϕ_i is of the shape:

$$\phi_i = P_{B_i^\perp}(x_0, \dots, x_k, t^i_{k+1}, \dots, t^i_d) \vdash P_{C_i}(x_0, \dots, x_k, u^i_{k+1}, \dots, u^i_d)$$

Therefore we get:

$$\begin{array}{rcl} \mathtt{head}([\phi]_{(0,k)}) & = & \mathtt{head}([\phi_1]_{(0,k)}), \\ \mathtt{tail}([\phi]_{(0,k)}) & = & \mathtt{tail}([\phi_n]_{(0,k)}). \end{array}$$

But as A is a conclusion we know by remark 7.3 (applied to γ_1 and the reverted path γ_n^{-1}) that if $\mathtt{head}([\phi_1]_{(0,k)}) = P_A(t_0, \ldots, t_k)$ and $\mathtt{tail}([\phi_n]_{(0,k)}) = P_A(u_0, \ldots, u_k)$ then:

- either for all $0 \leq i \leq k$, $t_i = u_i$,
- or there exists $j \leq k$ such that for $0 \leq i \leq (j-1)$, $t_i = u_i$ and $t_j \perp u_j$.

Both cases contradict the fact that $W(\gamma)$ is cyclic at depth k, so that we are done.

- second case: A is a cut formula. Let us call σ the corresponding cut node. By assumption we know that σ is at depth strictly greater than k, so in the same way as before we get: $[\phi]_{(0,k)} = P_A(x_0, \ldots, x_k) \vdash P_A(x_0, \ldots, x_k)$. This contradicts the hypothesis.

This last case ends the proof. □

We will need in the sequel the notion of *special cut*. 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 [16], [2]). We use a variant of the "special cut lemma" stated in [16] whose proof follows the same line.

Lemma 8.3. Let γ be a path of a proof-net R that crosses at least one cut at depth lower than k. If all the cuts crossed by γ at depth lower than k are exponential, then R has a special cut w.r.t. γ at depth lower than k.

Proof: We proceed by induction over the number of nodes of the proof-net. We use the sequentializability property of proof-nets : if R is a proof-net it can be obtained by a last rule (of course there might be several possible last rules).

- If R can be obtained by a $\wp$-rule on a proof-net R' then γ gives a path of R' satisfying the hypothesis. By induction hypothesis on R' we know that R' has a special cut w.r.t. γ at depth lower than k, which yields in R a special cut w.r.t. γ.

- If R can be obtained by a contraction, weakening or quantifier rule from a certain R' then the same argument applies.

- If R is obtained from R_1 and R_2 by a $\otimes$-rule then γ is contained in one of the R_i's and we can use the i.h. on R_i.

- If R is obtained from R_1 and R_2 by an axiom or multiplicative cut σ, then σ is at depth 0 in R and by assumption on γ we know that the path is contained in one of the R_i's.

- If R is obtained from R' by a t-promotion, then γ gives a path γ' in R' and $k \geq 1$. By assumption on γ we know that γ' crosses in R' only exponential cuts at depth lower than $k-1$ (and at least one) and by i.h. we conclude that R' has a special exponential cut σ w.r.t. γ' at depth lower than $k-1$. Then σ is a special cut of R w.r.t. γ and is at depth lower than k in R.

- If none of the previous cases applies, then all conclusions of R are conclusions of boxes at depth 0. Let us call $\mathbb{B}_1, \ldots, \mathbb{B}_n$ the boxes of R at depth 0. Consider the relation $\mathcal{R}$ between the $\mathbb{B}_i$'s defined by : $(\mathbb{B}_i \mathcal{R} \mathbb{B}_j)$ if there exists a cut between the principal port of $\mathbb{B}_j$ and an edge hereditary conclusion of an auxiliary port of $\mathbb{B}_i$. The correctness of R implies that the transitive closure $\mathcal{R}^\star$ of $\mathcal{R}$ is antisymmetric; therefore there is a box $\mathbb{B}_{i_0}$ such that there exists no cut below the auxiliary ports of $\mathbb{B}_{i_0}$. Let us denote by σ the cut on the principal port of $\mathbb{B}_{i_0}$. The proof-net R is obtained by the cut σ between a proof-net R' and $\mathbb{B}_{i_0}$. Now, if γ doesn't cross σ then it is contained in R' or $\mathbb{B}_{i_0}$ and as usual we can conclude by induction hypothesis. Otherwise if γ crosses σ, then by the property of $\mathbb{B}_{i_0}$ the cut σ is special w.r.t. γ and we are done.

$\square$

Lemma 8.4. 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, axiom or quantifier cut or a special 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.

Proof: Let $d(\sigma)$ denote the depth of σ, then $d(\sigma) \leq k$ by hypothesis. As $W(\gamma)$ is cyclic the path γ starts and ends in the same terminal formula X. We consider each case of cut σ:

- σ is an axiom cut, then let γ' be the path obtained by removing from γ the edges conclusions of the erased axiom; hence, we have $W(\gamma') = W(\gamma)$ and $W(\gamma')$ is cyclic at depth k;

- σ is a quantifier cut then γ' is obtained by removing from γ all the edges premises of the cut σ. Then $W(\gamma')$ is equal to $W(\gamma)$ up to a possible change of the predicate, so that the result ($W(\gamma)$ cyclic implies $W(\gamma')$ cyclic) follows immediately.

- σ is a multiplicative cut, let $B_1 \otimes B_2$ and $B_1^\perp \wp B_2^\perp$ be its premises. Let us decompose γ into maximal balanced subpaths
$$\gamma = \gamma_1; \gamma_2; \cdots; \gamma_n$$
such that:
  - γ_1 starts in X (upwards if it is a conclusion, downwards if it is a cut premise), ends in a premise of σ downwards and crosses σ at most once in the beginning if X is premise of σ,
  - when $1 < i < n$, γ_i starts and ends in a premise of σ downwards, and crosses σ once,
  - γ_n starts in σ downwards and ends in X crossing σ once.

  We denote by ϕ_i the clause $W(\gamma_i)$. Now, let γ'_i ($1 \leq i \leq n$) be the path of R' defined in the following way (we call it *residual path* of γ_i in R'):
  - for $2 \leq i \leq n$, $\gamma'_i = \delta_i; \gamma''_i$ where γ''_i is obtained from γ_i by removing the edges labelled by $B_1 \otimes B_2$ or $B_1^\perp \wp B_2^\perp$ and δ_i is the path of length 2 crossing the cut $(B_j, B_j^\perp)$ of R' (from B_j to $B_j^\perp$ if γ''_i starts in $B_j^\perp$, from $B_j^\perp$ to B_j if γ''_i starts in B_j);
  - if γ_1 starts in a premise of σ (i.e. X is premise of σ) then γ'_1 is obtained from γ_1 in the same way; otherwise γ'_1 is simply obtained by removing the edges labelled by $B_1 \otimes B_2$ or $B_1^\perp \wp B_2^\perp$.

We examine the composition of γ_i and γ_{i+1} for $1 \leq i \leq (n-1)$. As $W(\gamma) \neq 0$, we know that $\texttt{tail}(\phi_i)$ unifies with $\texttt{head}(\phi_{i+1})$; this implies in particular that they share the same predicate $P_{B_1 \otimes B_2}$ or $P_{B_1^\perp \wp B_2^\perp}$, let us assume for instance it is $P_{B_1 \otimes B_2}$. We have:

$$\begin{array}{rcl} \texttt{tail}([\phi_i]_{d(\sigma)}) & = & P_{B_1 \otimes B_2}(cu_{d(\sigma)}) \\ \texttt{head}([\phi_{i+1}]_{d(\sigma)}) & = & P_{B_1 \otimes B_2}(c' t_{d(\sigma)}), \end{array}$$

where $c = p$ (resp. $c = q$) if γ_i passes through B_1 (resp. B_2) before ending in $B_1 \otimes B_2$; $c' = p$ (resp. $c' = q$) if γ_{i+1} passes through $B_1^\perp$ (resp. $B_2^\perp$) after $B_1^\perp \wp B_2^\perp$. Now, as $\texttt{tail}(\phi_i)$ and $\texttt{head}(\phi_{i+1})$ are unifiable we get $c = c'$ and one can check that $\gamma'_1; \gamma'_2; \ldots; \gamma'_n$ is indeed a path in R' which we denote by γ'.

Let us denote by ϕ'_i the clause $W(\gamma'_i)$, then we have :

$$\begin{array}{rcl} \texttt{tail}([\phi'_i]_{d(\sigma)}) & = & P_{B_j}(u_{d(\sigma)}) \\ \texttt{head}([\phi'_{i+1}]_{d(\sigma)}) & = & P_{B_j}(t_{d(\sigma)}) \end{array}$$

where $j = 1$ if $c = c' = p$, $j = 2$ if $c = c' = q$. The term of $\texttt{tail}([\phi'_i]_h)$ (resp. $\texttt{head}([\phi'_{i+1}]_h)$) where $h \neq d(\sigma)$ is equal to that of $\texttt{tail}([\phi_i]_h)$ (resp. $\texttt{head}([\phi_{i+1}]_h)$). As a result we have :

- if X is not $B_1 \otimes B_2$ or $B_1^\perp \wp B_2^\perp$ then

  $$\begin{array}{rcl} \texttt{head}(\phi'_1) & = & \texttt{head}(\phi_1) \\ \texttt{tail}(\phi'_n) & = & \texttt{tail}(\phi_n); \end{array}$$

  we get :

  $$\phi'_1 \cdot \phi'_2 \cdots \phi'_n = \phi_1 \cdot \phi_2 \cdots \phi_n,$$

  so $W(\gamma') = W(\gamma)$ and thus γ' has a weight non-null and cyclic at depth k.
- if X is one of the premises of σ, say $X = B_1 \otimes B_2$ for instance : as $W(\gamma)$ is cyclic at depth k we know that $\texttt{tail}([\phi_n]_{(0,k)})$ is unifiable with $\texttt{head}([\phi_1]_{(0,k)})$, so if :

  $$\begin{array}{rcl} \texttt{tail}([\phi_n]_{d(\sigma)}) & = & P_{B_1 \otimes B_2}(cu_{d(\sigma)}) \\ \texttt{head}([\phi_1]_{d(\sigma)}) & = & P_{B_1 \otimes B_2}(c' t_{d(\sigma)}), \end{array}$$

  by the same argument as before we get $c = c'$ and :

  $$\begin{array}{rcl} \texttt{tail}([\phi'_n]_{d(\sigma)}) & = & P_{B_j}(u_{d(\sigma)}) \\ \texttt{head}([\phi'_1]_{d(\sigma)}) & = & P_{B_j}(t_{d(\sigma)}), \end{array}$$

  where $j = 1$ or $j = 2$. Thus $[W(\gamma)]_{d(\sigma)}$ is of the shape $P_{B_1 \otimes B_2}(cv_{d(\sigma)}) \vdash P_{B_1 \otimes B_2}(cw_{d(\sigma)})$ and $[W(\gamma')]_{d(\sigma)} = P_{B_j}(v_{d(\sigma)}) \vdash P_{B_j}(w_{d(\sigma)})$. For $k < d(\sigma)$, the atoms $\texttt{tail}([W(\gamma')]_k)$ and $\texttt{tail}([W(\gamma)]_k)$ (resp. $\texttt{head}([W(\gamma')]_k)$ and $\texttt{head}([W(\gamma)]_k)$) have the same term. Therefore $W(\gamma')$ is non-null and cyclic at same depth k as $W(\gamma)$.

- σ is a special (exponential) cut w.r.t. γ, let $?A^{\perp}$ and $!A$ be its premises. Notice that σ cannot be a weakening cut since γ crosses σ and no balanced path crosses any edge conclusion of a weakening node. So σ is either a contraction cut or a commutative cut. As before we decompose γ into maximal balanced subpaths
$$\gamma = \gamma_1; \gamma_2; \cdots ; \gamma_n$$
such that:

  - γ_1 starts in X, ends in a premise of σ and crosses σ at most once in the beginning if X is premise of σ,
  - when $1 < i < n$, γ_i starts and ends in a premise of σ downwards, and crosses σ once,
  - γ_n starts in σ downwards and ends in X crossing σ once.

  We have two cases: the case of an exponential cut of contraction type and the case of one of commutative type.

  - we consider the case of a special cut σ of contraction type. As σ is a special cut w.r.t. γ, if γ_i starts in $?A^{\perp}$ then γ_i ends in $!A$ and is contained in the box of principal port $!A$ but for its initial crossing of σ (see figure 5). Therefore if γ_i starts in $?A^{\perp}$ we have: $[\phi_i]_{d(\sigma)} = P_{?A^{\perp}}(x_{d(\sigma)}) \vdash P_{!A}(x_{d(\sigma)})$. Now if γ_j starts in $!A$ then γ_j ends in $?A^{\perp}$. This case can be handled as the multiplicative cut case:
    * if X is not a premise of σ then γ has a residual path γ' in R' of same weight;
    * if X is a premise of σ, say for instance $X = !A$. Then for $h < d(\sigma)$, $[W(\gamma)]_h$ and $[W(\gamma')]_h$ have the same terms and the layer $[W(\gamma)]_{d(\sigma)}$ is of the shape $P_{!A}(cv_{d(\sigma)}) \vdash P_{!A}(c'w_{d(\sigma)})$ with c, c' belonging to $\{r, s\}$. As $W(\gamma)$ is cyclic at depth k we have $c = c'$. We get $[W(\gamma')]_{d(\sigma)} = P_{!A}(v_{d(\sigma)}) \vdash P_{!A}(w_{d(\sigma)})$ and so finally $W(\gamma')$ is cyclic at depth k.

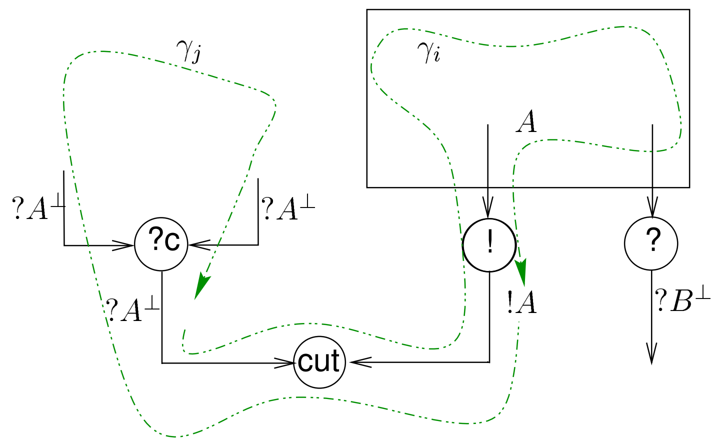

Figure 5. case of a contraction cut

– The last case is the exponential cut between an auxiliary port and a principal port: in this case the cut node σ is replaced in R' by a cut node σ' at depth $d(\sigma) + 1$, therefore weights are modified by reduction exclusively in the predicate part, so that the result ($W(\gamma)$ cyclic implies $W(\gamma')$ cyclic) follows immediately.

□

Proposition 8.5. Given a proof net R and γ a balanced path of non-null weight, the clause $W(\gamma)$ associated to γ is acyclic.

Remark 8.6. In [16], Regnier shows that in a GOI model for pure nets the weight $W(\gamma)$ of a path γ satisfies: $W(\gamma)^2 = 0$ (it follows from his theorems 3.3.1, part 3 and 2.2.1, part 4); it can be easily extended to MELL proof-nets. The property we give here (for ELL proof-nets) is stronger since all acyclic clauses are null-square but the converse is not true (see the example following definition 4.1).

Proof: In order to get contradiction we assume the proof-net R has a balanced path φ of non-null weight cyclic at depth k. By lemma 8.2 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, axiom or quantifier 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 k. Put $R_0 = R$ and $\delta_0 = \varphi$. Now assume the sequence has been defined up to rank $i \geq 0$. By lemma 8.2, δ_i crosses at least one cut in R_i at depth lower than k. If it crosses a multiplicative, axiom or quantifier cut σ at depth lower than k take for R_{i+1} the proof-net obtained from R by reducing σ; then by lemma 8.4 we know that R_{i+1} has a path satisfying the hypothesis which we take as δ_{i+1}. Otherwise lemma 8.3 ensures that δ_i has a special 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 8.7. Let (U, σ) be the loop associated to a proof-net R; we have:

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

Proof: As we know that $\mathtt{Result}^{\dagger}_{\sigma}(U) \subseteq \mathtt{Result}_{\sigma}(U)$ we only need to prove that the reverse inclusion holds. Let us recall (see [4]) that the result of the execution of a loop coming from a proof net is given by

$$\mathtt{Result}_{\sigma}(U) = \sum_{\gamma \in \mathcal{P}_c(R)} W(\gamma),$$

where $\mathcal{P}_c(R)$ is the set of balanced paths from conclusion to conclusion. Consider a path $\gamma \in \mathcal{P}_c(R)$ such that $W(\gamma) \neq 0$ and its decomposition into elementary balanced paths $\gamma = \gamma_0; \gamma_1; \cdots; \gamma_n$. We show that the associated word $(W(\gamma_0), W(\gamma_1), \cdots, W(\gamma_n))$ is strictly acyclic: for any sub-product we have

$$W(\gamma_i) \cdot \ldots \cdot W(\gamma_{i+h}) = W(\gamma_i; \cdots; \gamma_{i+h}) \neq 0$$

and $\gamma_i; \cdots; \gamma_{i+h}$ is a balanced path. By proposition 8.5 the clause $W(\gamma_i; \cdots; \gamma_{i+h})$ is acyclic. Thus the clause $W(\gamma) = W(\gamma_0) \cdot W(\gamma_1) \cdot \ldots \cdot W(\gamma_n)$ belongs to $\mathtt{Result}^{\dagger}_{\sigma}(U)$. □

Corollary 8.8. Let $(U, \sigma + \tau)$ be the loop associated to a proof-net R, then:

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

Proof: As by sub-associativity we know that

$$\mathtt{Result}^{\dagger}_{\sigma+\tau}(U) \subseteq \mathtt{Result}^{\dagger}_{\sigma}(\mathtt{Result}^{\dagger}_{\tau}(U)),$$

we only need to prove the reverse inclusion.

Let $\psi = \psi_0 \cdot \cdots \cdot \psi_m$ be an element of $\mathtt{Result}^{\dagger}_{\sigma}(\mathtt{Result}^{\dagger}_{\tau}(U))$ with for all $1 \leq i \leq m$, $\psi_i \in \sigma\mathtt{Result}^{\dagger}_{\tau}(U)$ and $\psi_0 \in \mathtt{Result}^{\dagger}_{\tau}(U)$. So there exist $\phi_0 \in U$ and ϕ_i's in τU or in σU such that

$$\psi_0 \cdot \ldots \cdot \psi_m = \phi_0 \cdot \ldots \cdot \phi_n.$$

Now the product $\phi_0 \cdot \ldots \cdot \phi_n$ belongs to $\mathtt{Result}_{\sigma+\tau}(U)$ since: $\phi_0 \in U$, the ϕ_i's belong to $(\sigma + \tau)U$, and $\mathtt{head}(\phi_0)$ and $\mathtt{tail}(\phi_n)$ are of the form $P_A(.)$ and $P_B(.)$ with A and B conclusions of R. By corollary 8.7, we conclude that ψ belongs to $\mathtt{Result}^{\dagger}_{\sigma+\tau}(U)$. □

Conclusion and perspectives

Broadly speaking our aim is to define a setting as large – and as simple – as possible for elementarily bounded computations. The operation we introduced, WEAK EXECUTION, does indeed satisfy the complexity requirement for pairs of combinations (U, σ) of our algebra; furthermore the establishing of this elementary bound doesn't need any assumption on (U, σ) (typically U and σ are not required to be wirings). This yields a large possibility as to the choice of the pairs we wish to consider as proper "programs" of our model. However, WEAK EXECUTION (partially) fails to fulfil the modularity requirement: this is the point which should guide us in the search for conditions on pairs to require to ensure good properties of the computation. Hence the work can be pursued in two directions: the first possibility is to search for a sufficient condition on pairs which guarantees this modularity/associativity property for a larger class of operators than those coming from proof-nets. One could then look for an untyped calculus whose computations could be performed in this algebra. A second option would be to come back to usual EXECUTION and use the techniques developed in this work to try to establish a result of the following kind: there exists a function $B(N, d)$ expressed as an exponential tower of height (depending on) d such that for any pair (U, σ) (satisfying certain conditions ...) if σU is nilpotent then its degree of nilpotency is bounded by $B(N, d)$ (where d is the depth of σU and N is given by its size).

Finally let us stress the fact that one goal of this work is to provide a basis for the study of the system LLL (Light Linear Logic) which, at the price of a more delicate syntax, offers a polynomial complexity bound. We believe that the study of ELL and of Elementary GOI is a worthwhile step in this more practically interesting direction.

References

[1] Baillot, P.: *Stratified coherent spaces: a denotational semantics for Light Linear Logic*, Technical Report EDI-INF-RR-0025, Division of Informatics, University of Edinburgh, august 2000. Extended abstract presented at the 2nd international Workshop on Implicit Computational Complexity, Santa Barbara, june 2000.

[2] Danos, V.: *La Logique Linéaire appliquée à l'étude de divers processus de normalisation (principalement du λ-calcul)*, Ph.D. Thesis, Université Paris VII, Juin 1990.

[3] Danos, V., Joinet, J.-B.: Elementary recursive functions and linear logic, 1999. Presented at the international workshop Implicit Computational Complexity'99

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

[5] Girard, J.-Y.: Linear Logic, *Theoretical Computer Science*, **50**, 1987, 1–102.

[6] Girard, J.-Y.: Geometry of Interaction I: an Interpretation of System F, *Proceedings of A.S.L. Meetings* (Ferro, al, Eds.), North-Holland, Padova, 1988.

[7] Girard, J.-Y.: Geometry of interaction II: Deadlock Free Algorithms, in: *Proceedings of COLOG'88* (Martin-Löf, Mints, Eds.), vol. 417 of *Lecture Notes in Computer Science*, Springer Verlag, 1988, 76–93.

[8] Girard, J.-Y.: Towards a Geometry of Interaction, *Categories in Computer Science and Logic* (J. W. Gray, A. Scedrov, Eds.), American Mathematical Society, 1989, 69–108. Proceedings of the AMS-IMS-SIAM Joint Summer Research Conference, June 14-20, 1987, Boulder, Colorado; Contemporary Mathematics Volume 92

[9] Girard, J.-Y.: Quantifiers in Linear Logic II, *Nuovi problemi della logica e della filosofia della scienza, Volume II* (G. Corsi, G. Sambin, Eds.), CLUEB, Bologna(Italy), 1991. Proceedings of the conference with the same name, Viareggio, 8-13 gennaio 1990

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

[11] Girard, J.-Y.: Proof-nets: the parallel syntax for proof theory, in: *Logic and algebra (Pontignano, 1994)*, vol. 180 of *Lecture Notes in Pure and Appl. Math.*, Dekker, New York, 1996, 97–124.

[12] Girard, J.-Y.: Light Linear Logic, *Information and Computation*, **143**, 1998, 175–204.

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

[14] Malacaria, P., Regnier, L.: Some Results on the Interpretation of λ-calculus in Operator Algebras, *Proceedings of the Sixth Annual Symposium on Logic in Computer Science*, IEEE Computer Society Press, 1991, 63–72.

[15] Pedicini, M.: Remarks on Elementary Linear Logic, *A Special Issue on the "Linear Logic 96, Tokyo Meeting"*, 3, Elsevier, Amsterdam, The Netherlands, 1996.

[16] Regnier, L.: *Lambda-Calcul et réseaux*, Ph.D. Thesis, Université Paris VII, 1992.