On paths-based criteria for polynomial time complexity in proof-nets

Matthieu Perrinel

On paths-based criteria for polynomial time complexity in proof-nets

Matthieu Perrinel

ENS de Lyon, CNRS, Inria, UCBL, Université de Lyon. Laboratoire LIP
matthieu.perrinel@ens-lyon.fr

Abstract. Several variants of linear logic have been proposed to characterize complexity classes in the proofs-as-programs correspondence. Light linear logic (LLL) ensures a polynomial bound on reduction time, and characterizes in this way the class *Ptime*. In this paper we study the complexity of linear logic proof-nets and propose two sufficient criteria, one for elementary time soundness and the other one for *Ptime* soundness, based on the study of paths inside the proof-net. These criteria offer several benefits: they provide a bound for any reduction sequence and they can be used to prove the complexity soundness of several variants of linear logic. As an example we show with our criteria a strong polytime bound for the system L^4 (light linear logic by levels).

1 Introduction

Implicit computational complexity is a research field aiming to characterize complexity classes by syntactically restricting models of computation. The main application is to achieve automated certification of a program's complexity. Due to its focus on resources management, linear logic (LL) [9] is a promising setting for this field. One of the interests of the linear logic approach is that it admits higher order types: functions are basic objects (e.g. functions can take functions as arguments and return functions).

In the linear logic approach, programs are proofs and program execution is done by the elimination of the cut rule in the proof. Proofs are either presented as sequent calculus derivations or as *proof-nets*, a graph based syntax for proofs. Programming in proof-nets is unnatural for most people. Fortunately, the proofs-as-programs correspondence states that a logical system corresponds to a type system for the λ-calculus [3]. λ-calculus is not used directly as a programming language but functional programming languages (e.g. Haskell) are based on it.

Most of the works controlling complexity in linear logic define a subsystem of LL enjoying some bounds on the length of cut-elimination sequences. Typically, the subsystem is defined in such a way that the programs are decomposed in *strata* and communication between strata is constrained. So syntax defines strata which in turn controls interaction. However a bunch of distinct subsystems have been defined in this way, corresponding to different notions of strata. How are these systems related and are there general principles underlying them?

To investigate these questions we propose here a kind of reverse approach: instead of defining strata by syntax we will define them by interaction. We believe this will contribute to establish these strata-based systems on solid ground. In a second step this could help in designing more general systems, and possibly also in analyzing the intrinsic limitations of strata-based systems.

Concretely we will consider a general language, LL, and define criteria on proofs based on interaction, expressed here by relations between subterms. These criteria entail bounds on the length of cut-elimination sequences. They can then be used either directly, to prove bounds on the complexity of a LL proof, or indirectly to prove that a LL subsystem entails complexity bounds. Concretely, the relations between subterms that we consider are defined by studying some paths in the proof, by means of context semantics [4].

Note that it is harder to control complexity in a higher order than in a first-order language. Thus, variants of LL for *Ptime* have to enforce strict control. Therefore, many polynomial time λ-terms can not be typed in those variants. For example, the λ-calculus type-system $DLAL$ [3] (obtained from the LL subsystem LLL [10]) is *Ptime extensionally complete*. It means that for any function f computable in polynomial time, there exists a λ-term computing f typable in $DLAL$. This is proved by showing that it is possible to simulate any Turing machine for a polynomial number of steps with a $DLAL$ typed term. However, $DLAL$ is not *intensionally complete*: there are λ-terms which compute in polynomial time but are not typable in $DLAL$.

Contributions In this paper, for any proof-net G, we define two relations $\twoheadrightarrow$ and $\succcurlyeq_2$ between some special elements of G called "boxes". A proof-net is said stratified if $\twoheadrightarrow$ is acyclic, and controls dependence if $\succcurlyeq_2$ is acyclic. A stratified proof-net normalizes in a number of steps bounded by an elementary function of its size. A proof-net satisfying both criteria normalizes in a number of steps bounded by a polynomial on its size. The elementary function and the polynomial only depend on the depth of the proof-net in terms of boxes and the depth of the $\twoheadrightarrow$ and $\succcurlyeq_2$ relations. Our approach has a number of benefits:

(i) strong vs weak complexity bounds;
(ii) use of the criteria to prove complexity bounds for variants of linear logic;
(iii) better understanding of existing linear logic systems for complexity.

Concerning (i), a programming language comes with a reduction strategy which determines the reduction order. For example: do we reduce the arguments before passing them to functions (call by value) or not (call by name)? Complexity bounds are sometimes proved for farfetched strategies, which are unlikely to be implemented in a real programming language. The bounds proved in this paper do not depend on the strategy (*strong complexity bounds*).

As to (ii), we show how to use our approach to study variants of linear logic: we can establish the complexity soundness of a system by showing that all its proofs satisfy our criteria. Here, we will apply this technique to L^4, for which only a weak *Ptime* bound was previously known. Note that it is relatively easy to prove that all proof-nets of a given system are stratified and control

dependence. The more difficult work is done in the proof that those properties entail complexity bounds, and this is independent of the system. Factorizing proofs of complexity bounds may ease the search and the understanding of such proofs. Actually, an important progress had already been made in this direction by Dal Lago with context semantics [4]: he provided a common method to prove complexity bounds for several systems like ELL, LLL and SLL [11]. Here we go a step further by designing higher level criteria, based on context semantics.

Concerning point (iii), we believe the present work can shed a new light on variants of linear logic for complexity. Indeed, the elaboration of such a system can be divided in two steps: first finding an abstract property implying a complexity bound and then finding a way to entail this property by syntactic means. Several extensions of LLL have been studied, like L^4 [1] and MS [13]. As those systems verify our criteria, we think those criteria illustrate a common property underlying these logics. Moreover we have found some λ-terms satisfying our criteria without being in any system of the above list. Thus, it seems we could define more expressive systems by being closer to our criteria.

Related works In the search for an expressive system for complexity properties, Dal Lago and Gaboardi have defined the type system $dlPCF$ [5] which characterizes exactly the execution time of PCF programs. Type-checking in $dlPCF$ is undecidable, but one can imagine restricting $dlPCF$ to a decidable fragment. Their framework can be seen as a top-down approach. Here we follow a bottom-up approach: we take inspiration from previous decidable type systems characterizing $Ptime$ and relax conditions losing neither soundness nor decidability.

Our main tool will be context semantics, a tool related to geometry of interaction [7]. Baillot and Pedicini used geometry of interaction to characterize elementary time [2]. Dal Lago adapted context semantics to study quantitative properties of cut-elimination [4]. From this point of view, an advantage of context semantics compared to the syntactic study of reduction is its genericity: some common results can be proved for different variants of linear logic, which allows to factor out proofs of complexity results for these various systems. We use the context semantics of Dal Lago, adapted to classical linear logic.

Here, we only give the statement of theorems, proofs can be found in [12].

2 Linear Logic

Linear logic [9] can be thought of as a refinement of System F [8] which focuses on the duplication of arguments. In LL, $A \Rightarrow B$ is decomposed into $!A \multimap B$. $!A$ means "infinitely many proofs of A" and $A \multimap B$ means "using one proof of A, I can prove B". In fact, $A \multimap B$ is a notation of $A^{\perp} \parr B$. We can view $(\_)^{\perp}$ as a negation and $\parr$ as a disjunction. The conjunction is written $\otimes$. Finally $\forall$ and $\exists$ allow us to quantify over the set of formulae. Compared to full linear logic, we use neither additives, nor constants and we add a modality: §, introduced by Girard for the expressive power of LLL. Precisely, formulae of LL are defined

inductively as follows (X ranges over a countable set of variables).

$$\mathcal{F} = X \mid X^\perp \mid \mathcal{F} \otimes \mathcal{F} \mid \mathcal{F} \parr \mathcal{F} \mid \forall X\mathcal{F} \mid \exists X\mathcal{F} \mid !\mathcal{F} \mid ?\mathcal{F} \mid \S\mathcal{F}$$

As examples, let us notice that $\forall X.X \multimap X$ is provable (for any formula X, using one proof of X, we get a proof of X). On the contrary, $\forall X.X \multimap (X \otimes X)$ is not provable because, in the general case, we need two proofs of X to prove $X \otimes X$.

We define inductively $(\_)^\perp$ on $\mathcal{F}$, which can be viewed as a negation: $(X)^\perp = X^\perp$, $(X^\perp)^\perp = X$, $(A \otimes B)^\perp = A^\perp \parr B^\perp$, $(A \parr B)^\perp = A^\perp \otimes B^\perp$, $(\forall X.A)^\perp = \exists X.A^\perp$, $(\exists X.A)^\perp = \forall X.A^\perp$, $(!A)^\perp = ?(A^\perp)$, $(?A)^\perp = !(A^\perp)$ and $(\S A)^\perp = \S(A^\perp)$.

Definition 1. *A* proof-net *is a graph-like structure defined inductively by the graphs of Figure 1 (G and H being proof-nets). Edges are labelled by formulae.*

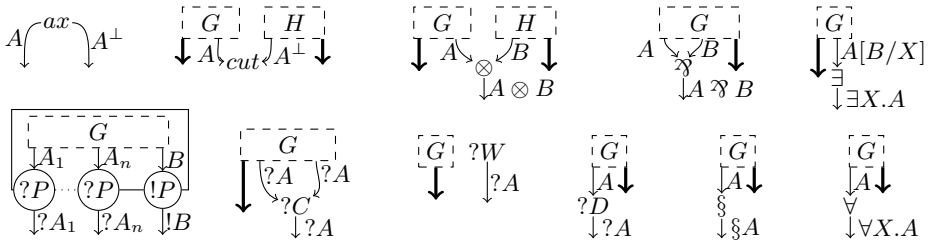

Fig. 1. Construction of proof-nets. For the $\forall$ rule, we require X not to be free in the formulae labelling the other conclusions of G

Readers familiar with linear logic may notice the absence of the digging principle ($!A \multimap !!A$). We removed it from this article to simplify the definitions. The way we deal with digging is described in [12].

The set of edges is written E_G. The rectangle in the left-most proof-net on the second row of Figure 1 is called a *box*. Formally a box is a subset of the nodes of the proof-net. We say that an edge (l, m) belongs to box B if l is in B. The number of boxes containing an edge e is its *depth* written $\partial(e)$. ∂_G is the maximum depth of an edge of G. The set of boxes of G is B_G. Let us call B the box in Figure 1. The node labelled $!P$ is the *principal door* of B, its outgoing edge is written $\sigma(B)$. The $?P$ nodes are the *auxiliary doors* of box B. $D_G(B)$ is the set of doors of B. The doors of box B do not belong to box B.

Lists are written in the form $[a_1; \ldots; a_n]$, $l_1@l_2$ represents the concatenation of l_1 and l_2, and . represents "push" ($[a_1; \ldots; a_n].b = [a_1; \ldots; a_n; b]$). If X is a set, $|X|$ is the cardinal of X. If $b, h, n \in \mathbb{N}$, we define b_h^n by $b_0^n = n$ and $b_{h+1}^n = b^{b_h^n}$.

The λ-terms correspond, through the proofs-as-programs paradigm, to proof-nets. Intuitively, proof-nets are λ-terms where applications and abstractions are respectively replaced by $\otimes$ and $\parr$ and with additional information on duplication. Cut-elimination is a relation, described in Figure 2, on proof-nets which corresponds to β-reduction. Proof-nets are stable under cut-elimination.

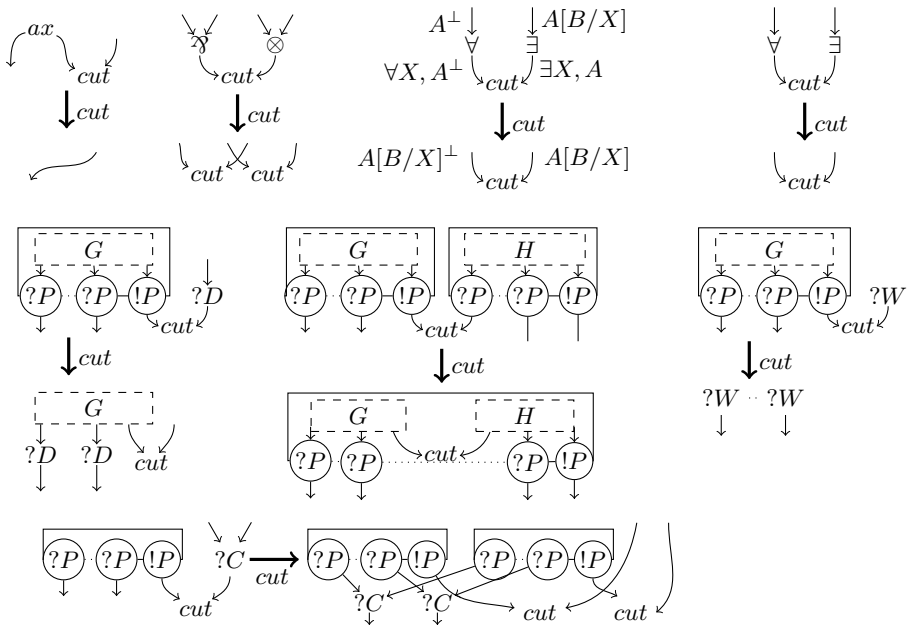

Fig. 2. Rules of cut-elimination. In the ∀/∃ rule, the substitution of the variable X by B takes place on the whole net.

3 Context Semantics

Let us first give an informal explanation. The usual way to prove strong bounds on a rewriting system is to assign a weight T_G to each term G such that, if G reduces to H, $T_G > T_H$. In LL, the $!P/?C$ step makes the design of such a weight hard: a whole box is duplicated, increasing the number of edges, cuts,... Let us suppose that G reduces to H, an element (box, edge or node) x' of H is said to be a *residue* of an element x of G if x' "comes" from x. In Figure 3, e, e_1, e_2, e_3 and e_4 are residues of e. A duplicate of $x \in G$ is a residue of x which has at most 1 residue (it can not be copied). In Figure 3, the duplicates of e are e_1, e_3 and e_4. Then, $\sum_{e \in E_G} |\{\text{duplicates of } e\}|$ does not increase, even during $!P/?C$ steps. We will define a weight based on this sum.

Context semantics determines, among the paths in a proof-net G, which paths are preserved by cut-elimination (such paths are called *persistent* in the literature [7]). Computing those paths is somehow like reducing the proof-net, and the persistent paths starting at the principal door of a box correspond to the duplicates of this box. In context semantics, persistent paths are captured by tokens (*contexts*) travelling across the proof-net according to some rules. The following definitions introduce the components of the tokens.

A *signature* is a list of `l` and `r`. A signature corresponds to a list of choices of premises of $?C$ nodes, to designate a particular duplicate of a box. The signature

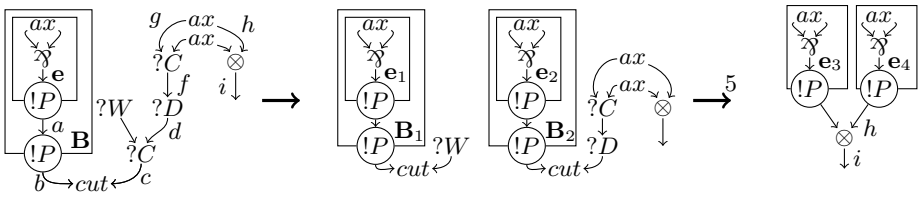

Fig. 3. Cut-elimination of a proof-net.

$t.\mathtt{r}$ means: "I choose the right premise, and in the next $?C$ nodes I will use t to make my choices". The set of signatures is written S.

A *potential* is a list of signatures: a signature corresponds to the duplication of one box, but an element is copied whenever any of the boxes containing it is cut with a $?C$ node. The set of potentials is written P. A potential is meant to represent duplicates. The duplicates of e in Figure 3, e_1, e_3 and e_4, will be respectively represented by potentials $[[\mathtt{l}];[]]$, $[[\mathtt{r}];[\mathtt{r}]]$ and $[[\mathtt{r}];[\mathtt{l}]]$. The *canonical potentials*, will characterize exactly the potentials corresponding to a duplicate.

A *trace element* is one of the following characters: $⅋_l, ⅋_r, \otimes_l, \otimes_r, \forall, \exists, \S, !_t, ?_t$ with t a signature. A trace element means "I have crossed a node with this label, from that premise to its conclusion". A *trace* is a non-empty list of trace elements. The set of traces is T. A trace is a memory of the path followed, up to cut-eliminations. We define duals of trace elements: $⅋_l^\perp = \otimes_l$, $!_t^\perp = ?_t, \ldots$ and extend the notion to traces by $([a_1;\cdots;a_k])^\perp = [a_1^\perp;\cdots;a_k^\perp]$.

A *polarity* is either $+$ or $-$. It will tell us in which way we are crossing the arrows. We define $+^\perp = -$ and $-^\perp = +$.

A *context* is a tuple (e, P, T, p) with $e \in E_G$, $P \in \mathsf{P}$, $T \in \mathsf{T}$ and p a polarity. It can be seen as a state of a token that will travel around the net. It is located on edge e (more precisely its duplicate corresponding to P) with orientation p and carries information T about its past travel.

The nodes define two relations $\rightsquigarrow$ and $\hookrightarrow$ on contexts. The rules are presented in Figure 4. Observe that these rules are deterministic. For any rule $(e, P, T, p) \rightsquigarrow (g, Q, U, q)$ presented in Figure 4, we also define the dual rule $(g, Q, U^\perp, q^\perp) \rightsquigarrow (e, P, T^\perp, p^\perp)$. We define $\mapsto$ as the union of $\rightsquigarrow$ and $\hookrightarrow$. In other words, $\mapsto$ is the smallest relation on contexts including every instance of $\rightsquigarrow$ rules in Figure 4 together with every instance of their duals and every instance of the $\hookrightarrow$ rule.

For every sequence $(e_1, P_1, T_1, p_1) \rightsquigarrow (e_2, P_2, T_2, p_2) \rightsquigarrow \cdots \rightsquigarrow (e_n, P_n, T_n, p_n)$, the sequence of directed edges $(e_1, p_1), \cdots, (e_n, p_n)$ is a path (i.e the head of e_i is the same node as the tail of e_{i+1}). The $\hookrightarrow$ relation breaks this property as it is non-local, in the sense that it deals with two non-adjacent edges. It is the main relation that distinguishes Dal Lago's context semantics from geometry of interaction. The trace keeps track of the history of previously crossed nodes to enforce path persistence: the $\mapsto$ paths are preserved by cut-elimination. The study of *paths*, sequences of the shape $C_1 \mapsto C_2 \mapsto \ldots$, will give us information on complexity.

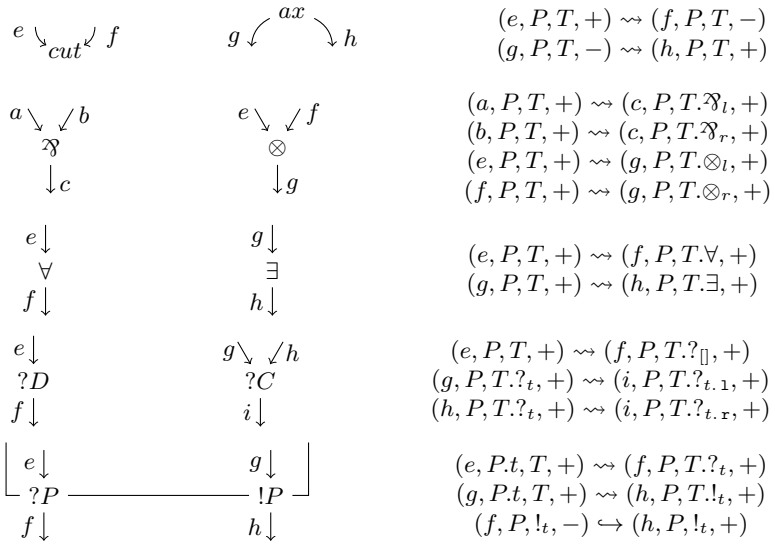

Fig. 4. Exponential rules of the context semantics

As an example, the path in the first proof-net of Figure 3 $(e, [[\mathtt{r}]; [\mathtt{l}]], [\parr_r], +) \mapsto (a, [[\mathtt{r}]], [\parr_r; !_{[\mathtt{l}]}], +) \mapsto (b, [], [\parr_r; !_{[\mathtt{l}]}; !_{[\mathtt{r}]}], +) \mapsto (c, [], [\parr_r; !_{[\mathtt{l}]}; !_{[\mathtt{r}]}], -) \mapsto (d, [], [\parr_r; !_{[\mathtt{l}]}; !_{[]}], -) \mapsto (f, [], [\parr_r; !_{[\mathtt{l}]}], -) \mapsto (g, [], [\parr_r; !_{[]}], -) \mapsto (h, [], [\parr_r; !_{[]}], +) \mapsto (i, [], [\parr_r; !_{[]}; \otimes_r], +)$ becomes the path $(e_4, [[]], [\parr_r], +) \mapsto (h, [], [\parr_r; !_{[]}], +) \mapsto (i, [], [\parr_r; !_{[]}; \otimes_r], +)$ in the third proof-net of Figure 3.

We want to capture the potentials which correspond to duplicates of a box. The definition can be difficult to understand. To give the reader a grasp of it, we will introduce the notion of $\mapsto$-copy progressively, proceeding by successive refinements. For the sake of simplicity, we will start with the case of depth 0.

- First, we could say that t corresponds to a duplicate of box B iff t corresponds to a sequence of choices of residues along a cut-elimination sequence, and the box residue we chose either will not be part of a *cut*, or the cut will open it. The $\mapsto$-paths are exactly the paths preserved by cut-elimination. So, we could make the first attempt: "t corresponds to a duplicate of B iff $(\sigma(B), [], [!_t], +) \mapsto^* (e, P, T, p) \not\mapsto$". However, this definition would allow potentials which refuse choices, corresponding to residues which have several residues themselves (e.g. [] for B in Figure 3). Indeed $(\sigma(B), [], [!_{[]}], +) \mapsto^* (c, [], [!_{[]}], -) \not\mapsto$. The duplicates of B in this figure are B_1 and B_2, which correspond to $[\mathtt{l}]$ and $[\mathtt{r}]$. [] corresponds to B, which can be copied.
- Thus, our second try would be "(B, t) corresponds to a duplicate iff $\exists n \in \mathbb{N}, \forall u \in \mathsf{S}, (\sigma(B), [], [!_{u.t}], +) \mapsto^n \not\mapsto$". However, this definition would allow potentials which make too many choices. For example $(B, [\mathtt{l}; \mathtt{r}])$ in Figure 3 satisfies this definition. Indeed $(\sigma(B), [], [!_{[\mathtt{l};\mathtt{r}]}], +) \mapsto^* (d, [], [!_{[\mathtt{l}]}], -) \not\mapsto$. So we

add the condition that the left-most trace element in the last context must be $!_{[]}$, the signature must be used entirely.

Such a signature will be called a *copy* of $(B, [])$. If $B_k \subset \cdots \subset B_1$, duplicates of B_k will be represented by $[t_1; \cdots ; t_k]$ with $[t_1; \cdots ; t_{k-1}]$ corresponding to a duplicate of B_{k-1} and t_k a copy of $(B_k, [t_1; \cdots ; t_{k-1}])$.

Definition 2. *Let $\to$ be a relation on contexts, a $\to$-copy of (B, P) (with $B \in B_G$ and $P \in \mathsf{P}$) is a signature t such that there exists $n \in \mathbb{N}$ such that*

$$\forall u \in \mathsf{S}, (\sigma(B), P, [!_{u.t}], +) \to^n (e, Q, [!_u], -) \not\to$$

The set of $\to$-copies of (B, P) is denoted $C_\to(B, P)$. Intuitively, $C_\mapsto(B, P)$ represents the duplicates of B, given the duplicates of the outer boxes.

Definition 3. *Let $\to$ be a binary relation on contexts and $e \in E_G$ such that $e \in B_{\partial(e)} \subset ... \subset B_1$. The set $L_\to(e)$ of canonical potentials for e is the set of potentials $[s_1; ...; s_{\partial(e)}]$ such that $\forall i \leq \partial(e), s_i \in C_\to(B_i, [s_1; \cdots ; s_{i-1}])$.*

So, a $\to$-canonical potential for e is the choice, for all box B_i containing e, of a $\to$-copy of B_i. In particular, in the case of $\to = \mapsto$, $L_\mapsto(e)$ corresponds to all duplicates of e. For example, in Figure 3 $L_\mapsto(e) = \{[[\mathtt{l}]; []]; [[\mathtt{r}]; [\mathtt{l}]]; [[\mathtt{r}]; [\mathtt{r}]]\}$. We define $L_\to(B) = L_\to(\sigma(B))$.

The next theorem is due to Dal Lago [4]. The intuition behind it is that each cut-elimination step either erases a node or copies a box. Thus, if we know the number of duplicates of each edge, we can bound the number of cut-elimination steps. This result allows to prove strong complexity bounds for several systems.

Theorem 1 (Dal Lago's weight theorem). *For every proof-net G, the length of any cut-elimination sequence beginning by G is bounded by:*

$$T_G = \sum_{e \in E_G} |L_\mapsto(e)| + 2 \cdot \sum_{B \in B_G} \left(|D_G(B)| \sum_{P \in L_\mapsto(B)} \sum_{t \in C_\mapsto(B,P)} |t| \right)$$

Moreover, G is acyclic*: there exists no path of the form $(e, P, [!_s], b) \mapsto^+ (e, P, [!_t], b)$.*

4 Stratification

4.1 Motivations

Stratification designates a restriction of a framework, which forbids the identification of two objects belonging to two morally different "strata". Russell's paradox in naive set theory relies on the identification of two formulae which belong morally to different strata. The non-terminating λ-term $\Omega = (\lambda x.(x)x)\lambda y.(y)y$ depends on the identification of a function and its argument. In recursion theory, to create from the elementary sequences $\theta_m(n) = 2^n_m$ (tower of exponential of height m in n), the non elementary sequence $n \mapsto 2^n_n$, we also need to identify n

and m which seem to belong to different strata. Stratification restrictions have been applied to those frameworks (naive set theory, linear logic, lambda calculus and recursion theory) to entail coherence or complexity properties [1].

ELL [10] can be seen as linear logic deprived of the dereliction ($!X \multimap X$) and digging ($!X \multimap !!X$) principles. This is a stratification restriction on linear logic. The stratum of an edge is the depth of the edge in terms of box inclusion. One can observe in the rules of cut-elimination that without the $?D$ dereliction node (here, we did not present the node of digging), the depth of an edge never changes during cut-elimination, so during cut-elimination, a residue of e can be cut with a residue of f only if e and f have same depth. Any proof-net of ELL reduces to its normal form in a number of steps bounded by an elementary function of its size [6]. In [1], Baillot and Mazza present an analysis of the concept of stratification and L^3, a generalization of ELL. Their stratification condition is enforced by a labelling of edges and also entails elementary time.

Here, we present an even more general stratification condition. This generalization is not given by a LL subsystem but by a criterion on proof-nets. Then, to prove that a system is elementary time sound, we only have to prove that all the proof-nets of the system satisfy the criterion. ELL and L^3 satisfy the criterion.

4.2 Preliminary intuition: stratification on λ-calculus

Our definition of stratification is based on context semantics paths and may be difficult to grasp at first read. To motivate the criterion, we first state a criterion on λ-calculus, the formal system whose terms are generated by $\Lambda = x \mid \lambda x.\Lambda \mid (\Lambda)\Lambda$.

We define *hole-terms* as λ-terms with a variable $\circ$ appearing free exactly once. If $t \in \Lambda$, $h[t]$ denotes $h[t/\circ]$ and, if t is not a variable, t is said a *subterm* of $h[t]$. For $t \in \Lambda$, $|t|$ is the size of t.

If $g[u] \to_\beta v$, the *residues* of u are the copies by β-reduction of u where free variables may have been substituted. Let us consider $t = (\lambda x.\lambda y.(x)(x)y)\lambda z.z \to_\beta \lambda y.(\lambda z.z)(\lambda z.z)y = t'$, then the residues of $\lambda z.z$ are the two occurrences of $\lambda z.z$ in t'. The residue of $(x)y$ is $(\lambda z.z)y$. Finally t and $\lambda x.\lambda y.(x)(x)y$ have no residue.

Let u, v be subterms of t, we capture "u belongs to a higher stratum than v" by the following relation $u \twoheadrightarrow v$.

Definition 4. *Let $t \in \Lambda$ and u, v be subterms of t, $u \twoheadrightarrow v$ if $t \to_\beta^* t'$ and there exists a subterm of t' of shape $(v')h[u']$ with u', v' residues of u, v.*

A λ-term is *stratified* if $\twoheadrightarrow$ is acyclic. Thus, Ω is not stratified because $\Omega \to_\beta (\lambda y.(y)y)\lambda y.(y)y$ so $\lambda y.(y)y \twoheadrightarrow \lambda y.(y)y$. For any stratified term t, we define S_t as the depth of $\twoheadrightarrow$ (i.e. the maximum n such that $\exists (t_i), t_0 \twoheadrightarrow \cdots \twoheadrightarrow t_n$).

Theorem 2. *If $t \in \Lambda$ is stratified, then there is a normalization sequence for t of length $\leq |t|^{|t|}_{3.S_t}$.*

Proof (sketch). Let t be a stratified λ-term. The strategy is to reduce first the set R_0 of subterms of t at stratum 0. By definition, these subterms have no residue inside the right part of an application. So there are never copied, they

only have one duplicate. Thus we can reduce all the elements of R_0 in at most $|R_0|$ steps, obtaining a term t' in which elements of R_0 have no residue. We have $|t'| \leq |t|^{2^{|R_0|}} \leq |t|^{2^{|t|}}$.

If u' and v' are subterms of t' such that $u' \twoheadrightarrow v'$ and u', v' are residues of u, v, then $u \twoheadrightarrow v$. Thus, $S_{t'} < S_t$. In at most S_t such rounds of reduction, we reach a normal form.

4.3 Stratification on proof-nets

We define a relation $\twoheadrightarrow$ between boxes of proof nets. It loosely correspond to the relation on Λ. In terms of context semantics paths, $B \twoheadrightarrow C$ means that there is a path beginning by the principal door of B which enters C by its principal door.

$$B \twoheadrightarrow C \Leftrightarrow \exists P, Q \in \mathsf{P}, t \in \mathsf{S}, T \in \mathsf{T}, (\sigma(B), P, [!_t], +) \rightsquigarrow^* (\sigma(C), Q, T, -)$$

To illustrate the correspondence with the criterion on Λ, we can observe that in the proof-net of Figure 5, $(\sigma(B_u), [], [!_{\mathtt{e}}], +) \mapsto^5 (\sigma(B_v), [], [!_{\mathtt{e}}; \otimes_l; ?_{\mathtt{e}}], -)$ so $B_u \twoheadrightarrow B_v$. Let u, v be the λ-terms corresponding to B_u and B_v, then the whole proof-net corresponds to $(\lambda x.h[(x)u])v$ which reduces to $h[(v)u]$ so $u \twoheadrightarrow v$.

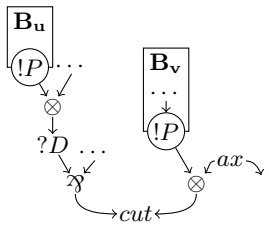

Fig. 5. This proof-net corresponds to $(\lambda x.h[(x)u])v$ for some h.

Definition 5. *A proof-net G is stratified if $\twoheadrightarrow$ is acyclic.*

For example, in Figure 6, $(\sigma(B), [], [!_{[1]}], +) \rightsquigarrow^* (\sigma(C), [], [!_{[]}; \otimes_l; !_{[\mathtt{r}]}], -)$ so $B \twoheadrightarrow C$. This is the only relation in $\twoheadrightarrow$ so the proof-net is stratified.

The weak bounds for ELL and L^3 were proved using a stratum by stratum strategy, for a specific notion of stratum. They prove that reducing the cuts at strata $\leq i$ does not increase too much the size of the proof-net at stratum $i+1$. Similarly, we will bound the number of copies of a box when we only reduce cuts in the strata $\leq i+1$ by the maximum number of copies of a box when reducing only cuts in strata $\leq i$. Thus, we need a notion of copy corresponding to reduction of cuts only in strata $\leq i$. This means we need a relation $\mapsto_i$ on contexts which simulates cut-elimination restricted to strata $\leq i$.

First, we define the *stratum* of a box B, written $S(B)$ as the depth of B in terms of $\twoheadrightarrow$, i.e. $\max\{s \mid \exists (B_i)_{i \leq s}, B_0 \twoheadrightarrow B_1 \ldots \twoheadrightarrow B_s\}$. Then, the stratum of a

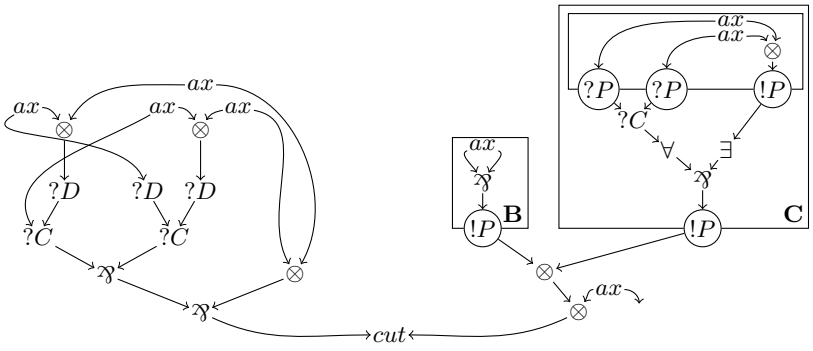

Fig. 6. This proof-net corresponds to $(\lambda\langle f,g\rangle.\langle (f)g,(g)f\rangle)\langle \lambda x.x, \lambda y.\langle y,y\rangle\rangle$

context C is the stratum of the box from which the context comes from: more formally, if there exists $B \in B_G$, $P \in \mathsf{P}$ and $t \in \mathsf{S}$ such that $(\sigma(B), P, [!_t], +) \rightsquigarrow^* C$, $S(C)$ is defined as $S(B)$. There are contexts such that $S(C)$ is undefined (if the left-most trace element is not a $!_u$, for example). $S(C)$ is not ambiguous because $\rightsquigarrow$ is injective and $(\sigma(B), P, [!_t], +)$ has no antecedent by $\rightsquigarrow$ (because the trace of a context is non-empty).

Let $s \in \mathbb{N}$, we define $C \mapsto_s D$ by: $C \mapsto_s D$ iff $C \mapsto D$ and $S(D) \leq s$. Thus, if B is a box of stratum $> s$, then for every P, t, $(\sigma(B), P, [!_t], +) \not\mapsto$ so $C_{\mapsto_s}(B, P) = \{[]\}$. This is the expected behaviour: we forbid the reduction steps involving a box of stratum $> s$, so B will never be duplicated. For matters of readability, we will often write $L_s(x)$ for $L_{\mapsto_s}(x)$ and $C_s(x, P)$ for $C_{\mapsto_s}(x, P)$. For any $s \in \mathbb{N}$, $\mapsto_s \subseteq \mapsto$ so $\mapsto_s$-copies of a box B are suffixes of $\mapsto$-copies of B.

Lemma 1. *For any $s \in \mathbb{N}$, any copy $t \in C_{\mapsto}(B, P)$ there is a unique $t^{/s} \in C_s(B, P)$ such that $t^{/s}$ is a suffix of t. For any $s \in \mathbb{N}$ and for any $P = [t_1; \ldots; t_{\partial(B)}] \in L_{\mapsto}(B)$, there is a unique $P^{/s} = [t'_1; \ldots; t'_{\partial(B)}] \in L_s(B)$ such that for all $1 \leq i \leq \partial(B)$, t'_i is a suffix of t_i.*

By Theorem 1, proof-nets are acyclic. So no $\mapsto$ path may go through two contexts of the shape $(e, Q, [!_u], p)$ and $(e, Q, [!_v], p)$. In fact, we can prove the following refinement. Let us assume $e \in B_{\partial(e)} \subset \cdots \subset B_1$, then no $\mapsto$ path beginning by $(\sigma(B), P, [!_t], +)$ may go through two contexts of the shape $(e, [q_1; \cdots; q_{\partial(e)}], [!_u], p)$ and $(e, [r_1; \cdots; r_{\partial(e)}], [!_v], p)$ where $q_i = r_i$ whenever $B \twoheadrightarrow B_i$. This gives us the following "strong acyclicity" lemma.

Lemma 2 (strong acyclicity). *Let us suppose that G is stratified and $e \in E_G$. If $(e, P, [!_t], p) \mapsto_s^+ (e, Q, [!_u], p)$ then $P^{/s-1} \neq Q^{/s-1}$.*

Theorem 3. *If a proof-net G is stratified, then the length of its longest reduction sequence is bounded by $2^{3.n}_{3.S_G+1}$ with $n = |E_G|$.*

Proof (sketch). Let us consider a $\mapsto_s$ path beginning by $(\sigma(B), P, [!_t], +)$. Lemma 2 bounds the number of times we can go through the same $?C$ node with a $[!_u]$ trace

by $\max_{(C,Q)} |C_{s-1}(C,Q)|^{\partial_G}$. So the length of any $\mapsto_s$-copy of (B,P) will be inferior to $|E_G|\cdot\max_{(C,Q)} |C_{s-1}(C,Q)|^{\partial_G}$. We get $|C_s(B,P)| \leq 2^{|E_G|\cdot\max_{(C,Q)} |C_{s-1}(C,Q)|^{\partial_G}}$. This gives an elementary bound on $|C_s(B,P)|$, by induction on s. Then, the result folows by Theorem 1.

5 Dependence Control

5.1 Motivations

Though stratification gives us a bound on the length of the reduction, elementary time is not considered as a reasonable bound. Figure 7 illustrates how the complexity arises, despite stratification. On this proof-net, the box A duplicates the box B and each copy of B duplicates C. If we extended this chain to n boxes, it will normalize in time 2^n. In [13], this situation is called a chain of *spindles*. We call "dependence control condition" any restriction on linear logic which aims to limit chains of spindles. The solution chosen by Girard [10] was to limit the number of $?P$-doors of each !-boxes to 1. To keep some expressivity, he introduced a new modality § with §-boxes which can have an arbitrary number of $?P$-doors.

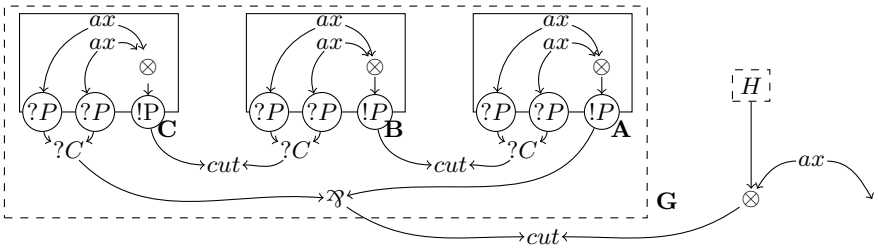

Fig. 7. If H is in normal form, this proof-net reduces in 32 cut-elimination steps

However, this solution forbids many proof-nets whose complexity is polynomial. The complexity explodes in Figure 7 because two copies of a box B merge with the same box A. A box with several auxiliary doors is harmful only if two of its auxiliary edges are contracted. Besides, we study the complexity of functions, not stand-alone proof-nets. We say that a proof-net G is in *polynomial time* if there is a polynomial P_G such that whenever G is cut with a proof-net H in normal form, the resulting proof-net normalizes in time $P_G(|E_H|)$. G is fixed and P_G depends on G. Thus, the sub-proof-net G of Figure 7 normalizes in constant time.

In fact, what really leads to an exponential blowup is when the length of such a chain of spindles depends on the input, as in Figure 8. If we replace the sub proof-net H (which represents 3) by a proof-net H' representing n, the resulting proof-net normalizes in time 2^n.

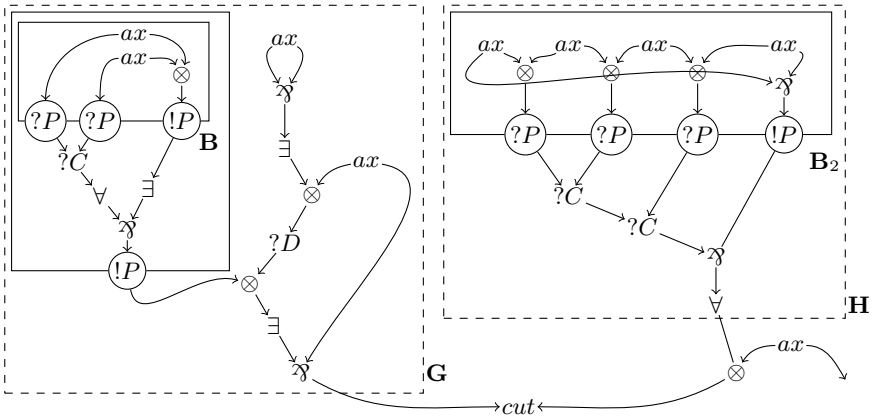

Fig. 8. The sub proof-net G is not polynomial time

5.2 A Dependence Control Criterion

We will define a relation $B \succcurlyeq_2 C$ on boxes meaning that at least 2 residues of B have their principal doors cut with an auxiliary door of C (spindle from B to C). We say that a proof-net *controls dependence* if $\succcurlyeq_2$ is acyclic. Then, if a proof-net G is stratified and controls dependence, the length of a chain of spindles in reducts of G is bounded by the number of boxes of G.

Definition 6.

$$B \succcurlyeq_2 C \Leftrightarrow \exists t \neq u \in \mathcal{S}, \begin{cases} (\sigma(B), P, [!_t], +) \mapsto^+ (\sigma(C), Q, [!_{\mathsf{e}}], -) \\ (\sigma(B), P, [!_u], +) \mapsto^+ (\sigma(C), Q, [!_{\mathsf{e}}], -) \end{cases}$$

If G controls dependence and $B \in B_G$, we define the *nest of* B (written $N(B)$) as the depth of B in terms of the $\succcurlyeq_2$ relation. N_G refers to $\max_{B \in B_G} N(B)$.

Theorem 4. *If G is stratified and controls dependence, let $n = |E_G|$, $N = N_G + 1$, $S = S_G + 1$ and $\partial = \partial_G + 1$, the maximal reduction length of G is bounded by*

$$n^{3+16N \cdot \partial^{2 \cdot N \cdot S}}$$

We can represent binary words in linear logic by the proof-nets of conclusion $\mathbf{B} = \forall X.!(X \multimap X) \multimap !(X \multimap X) \multimap !(X \multimap X)$. Let us notice that the number of boxes in a cut-free binary words (or any other inductive data-type in Church encoding) is fixed. Let us suppose that G is a proof-net of type $\mathbf{B} \multimap A$ (A being a formula) and for any cut-free binary word l, G cut with l is stratified and controls dependence. Then, there exists a polynomial P such that for all normal proof-net l representing a binary word of length n, the application of G to l normalizes in at most $P(n)$ cut-elimination steps.

We can notice that the degree of the polynomial rises very fast. During the proof we used rough bounds. Otherwise, the statement of the bound would have

been too complex. The bound is so high because, given S_G and N_G, we must consider the worst possible case (for example that there are boxes of nest N_G in each stratum). Given the exact $\twoheadrightarrow$ and $\succcurlyeq_2$ relations on a stratified proof-net G controlling dependence, one can statically infer tighter bounds (if we are not in the worst possible case) by following the proofs of Lemmas 24 and 25 in [12].

6 Applications

L^4 (Light Linear Logic by Levels) is a system introduced by Baillot and Mazza [1] which generalizes LLL. L^4 is defined as the set of proof-nets for which we can label each edge e with an integer $l(e)$ verifying the rules of Figure 9, and whose boxes have at most one auxiliary door. We define l_G as $\max\{l(e) \mid e \in E_G\}$. Baillot and Mazza proved a weak polynomial bound for L^4 proof-nets for a particular strategy [1], but no strong bound[1]. Obtaining a strong polynomial bound is important to define a type system for λ-calculus based on L^4, because it is unclear whether the particular strategy on proof-nets of [1] could be converted into a β-reduction strategy.

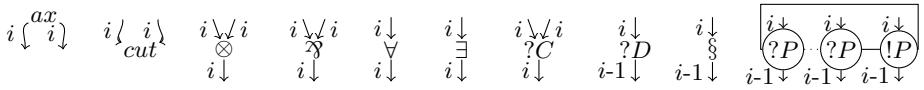

Fig. 9. Relations between levels of neighbour edges in L^4

In L^4, the level of a box is stable by cut-elimination. This property has an equivalent in the context semantics presentation: the sum of the depth of the edge and the number of exponential trace elements is stable. Let $T \in \mathsf{T}$, the number of !, ? and § trace elements in T is denoted $\|T\|$. Notice that the following Lemma only holds for the $\rightsquigarrow$ relation, not for the $\mapsto$ relation. This makes the reasonings on L^4 more complex and partly explains why it was difficult to prove a strong bound for L^4.

Lemma 3. *If G is a L^4 proof-net and $(e, P, T, p) \rightsquigarrow_G^* (f, Q, U, q)$, then*

$$l(e) + \|T\| = l(f) + \|U\|$$

Lemma 4. *If G is a L^4 proof-net and $B \twoheadrightarrow B'$ then $l(\sigma(B)) > l(\sigma(B'))$.*

Thanks to Lemma 4, we can prove that L^4 proof-nets are stratified with strata of boxes being bounded by the maximum level of edges. It gives us an elementary bound on *cut* elimination. To prove a polynomial bound, one can prove that L^4 proof-nets control dependence. In fact, in L^4, the $\succcurlyeq_2$ relation is empty so acyclic.

[1] In fact, a proof of a strong bound is claimed in [13], but it contains flaws which do not seem to be easily patchable. See Appendix A and [12] for more details.

Theorem 5. *Let G be a L^4 proof-net, with $n = |E_G|$, of maximal level l, then the length of the longest reduction path is inferior to*

$$n^{3+16\partial_G^{2l_G+2}}$$

In L^4, binary words are represented using the Church encoding with the type: $\mathbf{B}_4 = \forall X.!(X \multimap X) \multimap !(X \multimap X) \multimap \S(X \multimap X)$. In Theorem 5, the polynomial only depends on the level and depth of the proof-net. Let G be a L^4 proof-net representing a function on binary words, i.e. the only conclusion of G has type $\mathbf{B}_4 \multimap A$ for some A. Then, there exists a polynomial P such that for all normal proof-net H representing a binary word of length n, the application of G to H normalizes in at most $P(n)$ cut-elimination steps.

Other systems The framework MS [13] is a set of subsystems of ELL where ! connectives are indexed by integers. ELL is stratified so all proof-nets of MS are stratified. In [13], Vercelli characterizes the "most general" *Ptime* sound subsystems of MS. Those systems allow !-boxes with several auxiliary doors. In those systems, if $B \succcurlyeq_2 B'$, $\sigma(B)$ and $\sigma(B')$ are labelled by formulae of respective shape $!_n A$ and $=!_{n'} A'$, then $n < n'$. The dependence control follows immediately. The strong polynomial bound, however, was already proved in [13].

We also prove a strong bound for L_0^4, a refinement of L^4 [1]. L_0^4 does not enjoy stratification but we can derive a strong bound for L_0^4 from the strong bound for L^4. No polynomial bound was previously proved for this system.

Comparison with L^3. Our criteria allowed to show strong polynomial bounds for systems for which only weak bounds were known. In addition, it shows that the stratification constraints of L^3 could be relaxed. Indeed, we found stratified proof-nets controlling dependence corresponding to the λ-term $((\underline{2})\lambda n.((n)S)\underline{1})\underline{0}$ (with $\underline{n}$ being the representation of $n \in \mathbb{N}$ in Church numerals) and the λ-term with pair[2] $(\lambda\langle f, g\rangle.\langle (f)g, (g)f\rangle)\langle \lambda x.x, \lambda y.\langle y, y\rangle\rangle$ (Figure 6). It seems that there are no L^3 proof-net corresponding to those terms. However, those examples are contrived, and it is still not clear how much expressive power can be gained in practice by relaxing the stratification conditions of L^3.

Comparison with MS. For any *Ptime* sound MS system S, the length of chains of spindles is bounded by an integer k_S. Let us fix S, we can extend the chain of spindles of G in Figure 7 to $k_S + 1$ spindles, so that G is still constant time but not in S. Our criteria are more general. First, we do not fix *a priori* a limit on the length of chains of spindles, but only forbid cycles. So G, even with an extended chain of spindles, satisfies our criteria. Let $t = \underline{k}(\lambda\langle x, y, z\rangle.\langle x, ((+)x)y, y\rangle)$ and $u = \lambda\langle x, y, z\rangle.\langle z, z, z\rangle$, then $(t)(u)(t)\cdots(u)t$ is stratified and controls dependence, whatever the length of the chain of applications, whereas in MS the maximum length of such a chain is bounded. Moreover, our bound is still valid in presence of dereliction ($?D$) and digging (dealing with the latter is more complex and is only presented in the long version [12]).

[2] The pairs are here represented by using the connective $\otimes$

7 Conclusion

We defined two criteria on proof-nets which imply bounds on the complexity of cut-elimination. These are then used to prove strong bounds for systems for which only weak bounds were known. A major advantage of our approach is that, once our general lemmas are established, proving bounds for various systems is quite simple. There are *Ptime* proof-nets which do not verify our criteria, however the expressive power of those criteria is still unclear. In future work, we plan to define more expressive systems based on these.

Let us comment on decidability issues. One can compute all the $\mapsto$ paths in a proof-net, for example by reducing the proof-net. So, stratification and dependence control of a proof-net are decidable. However, as we are interested by the complexity of functions, the interesting problem of certifying complexity of a proof-net G is "Is there any cut-free proof-net H such that G cut with H is not stratified or does not control dependence?". This problem seems undecidable. As a future work, we want to design a decidable type system, inspired by our criteria, capturing *Ptime*.

References

1. P. Baillot and D. Mazza. Linear logic by levels and bounded time complexity. *Theoretical Computer Science*, 411(2), 2010.
2. P. Baillot and M. Pedicini. Elementary complexity and geometry of interaction. *Fundamenta Informaticae*, 45(1-2), 2001.
3. P. Baillot and K. Terui. Light types for polynomial time computation in lambda calculus. *Information and Computation*, 207(1), 2009.
4. U Dal Lago. Context semantics, linear logic, and computational complexity. *ACM Transactions on Computational Logic*, 10(4), 2009.
5. U. Dal Lago and M. Gaboardi. Linear dependent types and relative completeness. In *Logic in Computer Science, 2011*. IEEE, 2011.
6. V. Danos and J.B. Joinet. Linear logic and elementary time. *Information and Computation*, 183(1), 2003.
7. V. Danos and L. Regnier. Proof-nets and the Hilbert space. *London Mathematical Society Lecture Note Series*, 1995.
8. J.Y. Girard. Une extension de l'interpretation de gödel a l'analyse, et son application a l'elimination des coupures dans l'analyse et la theorie des types. *Studies in Logic and the Foundations of Mathematics*, 63, 1971.
9. J.Y. Girard. Linear logic. *Theoretical computer science*, 50(1), 1987.
10. J.Y. Girard. Light linear logic. *Information and Computation*, 143(2), 1998.
11. Y. Lafont. Soft linear logic and polynomial time. *Theoretical Computer Science*, 318(1-2), 2004.
12. M. Perrinel. On paths-based criteria for polynomial time complexity in proof-nets (long version). `http://arxiv.org/abs/1201.2956`, 2013.
13. L. Vercelli. *On the Complexity of Stratified Logics*. PhD thesis, Scuola di Dottorato in Scienze e Alta Tecnologia, Università degli Studi di Torino – Italy, 2010.

A Discussion on previous work on L^4 strong bound

In Section 8.2.2 of [13], Vercelli claims a proof of strong polynomial bounds for some subsystems of $MS^\dagger$. L^4 is one of these systems. However, the proof of the strong bound contains some flaws. Indeed, the $\mapsto$ relation used in this section has no rule to leave a box by its principal door. Moreover, the weight T_G used differs from the weight used by Dal Lago [4] and us. In the following, T_G designs the weight defined by Vercelli and we will show that the lemma 8.2.15 - which corresponds to the "Dal Lago's weight theorem" - is false. Indeed, in Figure 10, $G \to_{cut} H$ but $T_G = 0 + 2 + 2 + 10 = 14$: 0 for box B door because no maximal CS-path begin by $\sigma(B)$, 2 for both boxes at depth 0, 1 for each node which is neither an axiom nor a door. And $T_H = 2.4.2 + 2 + 2 + 10 = 22$: 2.4.2 for B door because each of the 4 B copies has length 2, 2 for the box at depth 0, 1 for each node which is neither an axiom nor a door. So $T_G < T_H$.

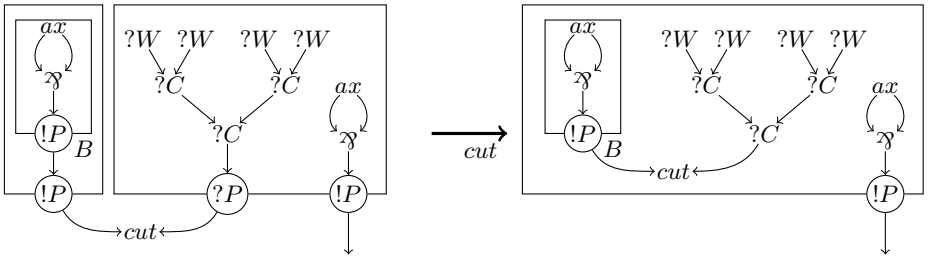

Fig. 10. The proof-net G reduces to H, but $T_G < T_H$

If we want the lemma 8.2.15 to hold, we could allow the contexts to leave boxes by their principal door (as in our $\mapsto$ relation). Then there would be a problem in the way the $?D$ is handled. Indeed, crossing a $?D$ node upwards adds a signature on the potential without entering a box. Thus, the lemma 8.2.15 would still fail, as shown on Figure 11: $(\sigma(B), [[\mathtt{1}]; []], [!_t], +) \rightsquigarrow (a, [[\mathtt{1}]], [!_t; !_{[]}], +) \rightsquigarrow (b, [[\mathtt{1}]], [!_t; !_{[]}], -) \rightsquigarrow (c, [[\mathtt{1}]; []], [!_t], -) \rightsquigarrow^2 (d, [[\mathtt{1}]], [!_t; !_{[]}], +) \rightsquigarrow (e, [[\mathtt{1}]], [!_t; !_{[]}], -) \not\mapsto$.

If we fix this problem by taking our $\mapsto$ relation, then lemma 8.2.17 would fail. Indeed, crossing a $?D$ node changes the number of exponential stack element in the stack without changing the length of the potential. If we fixed it by replacing the length of the potential by the level of the edge in the enunciation of lemma 8.2.17 then the lemma would fail on the $\hookrightarrow$ steps because the doors of a same box may have different levels. So the correct form of the lemma is:

If G is a L^4 proof-net and $(e, P, T, p) \rightsquigarrow_G^* (f, Q, U, q)$, then

$$l(e) + \|T\| = l(f) + \|U\|$$

This is exactly our lemma 3. However, proving that this weaker lemma is enough is far from trivial.

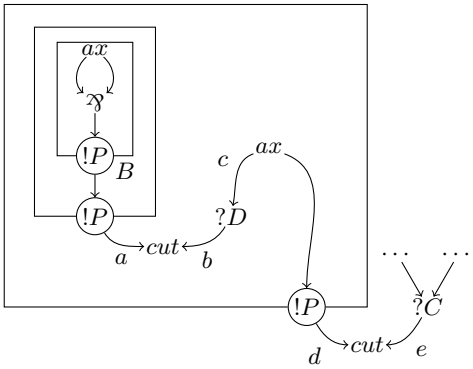

Fig. 11. The $\mapsto$-path beginning by $(\sigma(B), [[\mathtt{1}]; []], [!_t], +)$ does not cross the contraction node.