Safe Recursion on Notation into a Light Logic by Levels[*]

Luca Roversi
Dipartimento di Informatica - Università di Torino
roversi@di.unito.it

Luca Vercelli
Dipartimento di Matematica - Università di Torino
luca.vercelli@unito.it

We embed Safe Recursion on Notation (**SRN**) into Light Affine Logic by Levels (**LALL**), derived from the logic **ML**^{4}. **LALL** is an intuitionistic deductive system, with a polynomial time cut elimination strategy. The embedding allows to represent every term t of **SRN** as a family of nets $\langle \lceil t \rceil^l \rangle_{l \in \mathbb{N}}$ in **LALL**. Every net $\lceil t \rceil^l$ in the family simulates t on arguments whose bit length is bounded by the integer l. The embedding is based on two crucial features. One is the recursive type in **LALL** that encodes Scott binary numerals, *i.e.* Scott words, as nets. Scott words represent the arguments of t in place of the more standard Church binary numerals. Also, the embedding exploits the "fuzzy" borders of paragraph boxes that **LALL** inherits from **ML**^{4} to "freely" duplicate the arguments, especially the safe ones, of t. Finally, the type of $\lceil t \rceil^l$ depends on the number of composition and recursion schemes used to define t, namely the structural complexity of t. Moreover, the size of $\lceil t \rceil^l$ is a polynomial in l, whose degree depends on the structural complexity of t. So, this work makes closer both the predicative recursive theoretic principles **SRN** relies on, and the proof theoretic one, called *stratification*, at the base of Light Linear Logic.

1 Introduction

Slightly rephrasing the *incipit* of [6], comparing implicit characterizations of computational complexity classes may provide insights into their nature, while offering concepts and methods for generalizing computational complexity to computing over arbitrary structures and to higher type functions. Here, we relate two implicit characterizations of polynomial time functions (**PTIME**). One is Safe Recursion on Notation (**SRN**) [4], that we take as representative of the characterizations of **PTIME** that restrict the primitive recursion. The other one is Light Affine Linear Logic by Levels (**LALL**), a proof theoretical system we derive from Light Linear Logic by Levels (**ML**^{4}) [3] and from Intuitionistic Light Affine Logic (**ILAL**) [2]. We recall, **ML**^{4} and **ILAL** are two *Light Logics*, *i.e.* restrictions of Linear Logic that characterize some complexity class, in this case **PTIME**, under the proofs-as-programs analogy. These two logics control the complexity of the algorithms they can express by the technical notion *Stratification*, which expresses specific structural restrictions on the derivations of **ML**^{4} and **ILAL**. **SRN**, of which we recall some more aspects in Section 4, provides a *predicative analysis* of primitive recursion. It is the least set that contains the *zero* 0 (considered as a 0-ary function), the *successors* $\mathtt{s}_0(;x) = 2x, \mathtt{s}_1(;x) = 2x+1$, the *predecessor* $\mathtt{p}(;2x+i) = x$, the *projection* $\pi^{\mathsf{n};\mathsf{s}}_k(\vec{x};\vec{y}) = x_k$ if $1 \leq k \leq \mathsf{n}$, and y_k if $1 \leq k \leq \mathsf{s}$, the *conditional* $\mathtt{B}(;y,y_1,y_2) = y_1$ if y is odd, and y_2 otherwise, and which is closed under *safe composition* and *predicative recursion on notation* ((2) and (3) in Figure 1). The work [8] is the first one relating the two different traditions: it defines a map from terms of a strict fragment **BC**$^-$ of **SRN** into nets of **ILAL**. The main obstacle to a full representation of **SRN** into **ILAL** is that the duplication of nets in **ILAL**, hence of the safe arguments, is far from being free, as required instead by (3). In fact, [8] also shows that an extension **BC**$^\pm$, polynomial time complete, can be represented inside **ILAL**. However, since the primitives added to **BC**$^\pm$ are not in **SRN**, we cannot see **BC**$^\pm$ as relevant to the goal of understanding the

[*]Partially supported by MIUR PRIN CONCERTO — protocol number 2007BHXCFH

possible relation between full **SRN** and the above stratification principle, basic to **ML**[4] and **ILAL**. Since [8], no extension of the relation between **SRN** and **ILAL** has been produced, to our knowledge. Here, we show to which extent we can avoid that obstacle inside **LALL**. **LALL**, that will be formally defined in Section 2, is an intuitionistic system of nets endowed with: (i) edges labelled by indices, or levels, (ii) unconstrained weakening, to make programming with its nets somewhat more comfortable, (iii) a language of formulæ quotiented by the recursive equivalence $\mathscr{S} = \forall\alpha.(\alpha \multimap (\mathbb{B} \multimap \mathscr{S} \multimap \alpha) \multimap \alpha)$, where $\mathbb{B}$ is the type of booleans, and $\mathscr{S}$ the data type of *Scott words* [1], and (iv) a polynomial time sound cut elimination procedure (Section 3) which does not depend on the types that label the edges of a given net.

SRN embeds into **LALL** by means of the map $\lceil\cdot\rceil^{\cdot}$ (Sections 5 and 6.) The map $\lceil\cdot\rceil^{\cdot}$ has the same *natural* and *inductive* structure as the one of the map in [8] from **SRN** to **ILAL**. However, $\lceil\cdot\rceil^{\cdot}$ takes two arguments: (i) any term t of $\mathbf{SRN}^{\mathsf{n};\mathsf{s}}$, with normal arity n, the number of arguments to the left of the semicolon, and safe arity s, the number of those ones to the right, and (ii) an integer $l \geq 0$ that bounds the size of every argument of t. Then, $\lceil t\rceil^{l}$ yields a net that simulates $t(\overrightarrow{x};\overrightarrow{y})$ whenever every element in $\overrightarrow{x},\overrightarrow{y}$ *is at most as long as* l (Proposition 12). This suggests to summarize the situation we move in by:

$$\frac{\textbf{LALL}}{\textbf{SRN}} = \frac{\textbf{PTIME}\text{-uniform Boolean Circuits}}{\textbf{PTIME}\text{ Problems}} \tag{1}$$

We remark, however, that such an analogy should be read as such, and not as a formal correspondence. *I.e.*, we are not at all assuming any classical complexity theoretic perspective like the one in [12], which shows a proofs-as-programs correspondence between Boolean Circuits and nets of Multiplicative Linear Logic.

Instead, what we do reads as follows.

Let t be a term of $\mathbf{SRN}^{\mathsf{n};\mathsf{s}}$. We write $\partial_{\mathtt{C}}(t)$ and $\partial_{\mathtt{R}}(t)$ for the number of composition and recursion schemes, respectively, that are used to build t. That way, $\mathtt{cmplx}(t) = \partial_{\mathtt{C}}(t) + \partial_{\mathtt{R}}(t)$ is a naïve measure of the *static* complexity of t. Also, let p_t be the characteristic polynomial of t, whose values bound the length of the output of t. Let $\partial(p_t)$ be its degree. Then, t is represented in **LALL** by a family $\langle\lceil t\rceil^{l}\rangle_{l\in\mathbb{N}}$ of nets such that:

1. The size of every net $\lceil t\rceil^{l}$ is $O(l^{\partial(p_t)^{\mathtt{cmplx}(t)}})$, namely polynomial in l;
2. If l is at least as great as every bit length $|x_1|,\ldots,|x_{\mathsf{n}}|,|y_1|,\ldots,|y_{\mathsf{s}}|$ of the arguments, then the application of $\lceil t\rceil^{l}$ to $\lceil x_1\rceil^{l},\ldots,\lceil x_{\mathsf{n}}\rceil^{l},\lceil y_1\rceil^{l},\ldots,\lceil y_{\mathsf{s}}\rceil^{l}$ equals $\lceil t(x_1,\ldots,x_{\mathsf{n}};y_1,\ldots,y_{\mathsf{s}})\rceil^{l}$;
3. Every $\lceil t\rceil^{l}$ is a map from $(\bigotimes^{\mathsf{n}}\mathscr{S})\otimes(\bigotimes^{\mathsf{s}}(\S^{k}\mathscr{S}))$ to $\S^{k}\mathscr{S}$, where k depends on $\mathtt{cmplx}(t)$.

The first two points suggest the analogy (1). Specifically, the first point expresses a uniformity condition on the nets in the family, since it states that their dimension are bounded only by the length of the inputs. The second point says that $\lceil t\rceil^{l}$ soundly simulates t on *every* input of length *smaller or equal* to l. Finally, the third point is a natural property we can expect as soon as we try to compositionally and naturally represent first order algebraic terms, that operate on a given domain, into a higher order language. It is a static description of the behavior of t in terms of types of **LALL**, a kind of information we cannot have by, for example, representing **SRN** as circuit families.

We see the use of $\mathscr{S}$ as a first fundamental choice to write $\lceil\cdot\rceil^{\cdot}$. The reason is twofold. One reason is a kind of obvious, since $\mathscr{S}$ supports the representation of successors, predecessor, projection and conditional as constant time operations, unlike $\mathscr{C} = \forall\alpha.!(\alpha \multimap \alpha) \multimap !(\alpha \multimap \alpha) \multimap \S(\alpha \multimap \alpha)$, which is generally used to represent *Church words* in Light Logics. The other reason, instead, brings a certain

$$\circ[t' u_1 \ldots u_{\mathsf{n}'} v_1 \ldots v_{\mathsf{s}'}](\vec{x};\vec{y}) = t'(u_1(\vec{x};),\ldots,u_{\mathsf{n}'}(\vec{x};);v_1(\vec{x};\vec{y}),\ldots,v_{\mathsf{s}'}(\vec{x};\vec{y})) \tag{2}$$

$$\begin{cases} \mathtt{r}[u_\varepsilon,u_0,u_1](0,\vec{x};\vec{y}) = u_\varepsilon(\vec{x};\vec{y}) & (3) \\ \mathtt{r}[u_\varepsilon,u_0,u_1](2z+i,\vec{x};\vec{y}) = u_i(z,\vec{x};\vec{y},\mathtt{r}[u_\varepsilon,u_0,u_1](z,\vec{x};\vec{y})) & i \in \{0,1\} \end{cases}$$

Figure 1: **SRN**: predicative recursion on notation and safe composition.

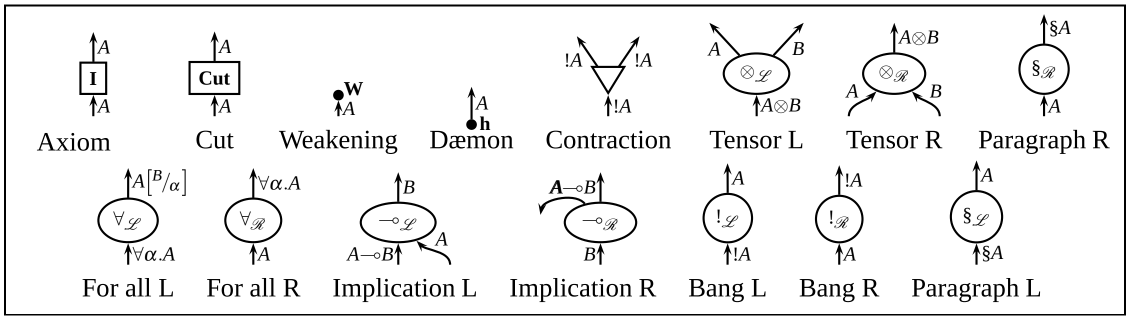


Figure 2: The nodes in the proof nets of **LALL**.

degree of novelty with it because we exploit a crucial property of **LALL**, and of Light Logics, *which had hardly been used so far*. The crucial property is that the polynomial time cut elimination of **LALL** depends only on the structure of any given net Π, while the logical complexity of the formulæ in Π does not affect it. So, we are free to add fixpoints formulæ, like $\mathscr{S}$ is, which adds a huge expressivity to the logic.

A second step to get $\lceil\cdot\rceil^{\cdot}$, for every l, we exploit what we like to call the *fuzzy* borders of paragraph boxes of **LALL** to write the net $\nabla\mathtt{S}^k_l$. The net $\nabla\mathtt{S}^k_l$ duplicates a Scott word at most as long as l, starting from a premise of type $\S^k\mathscr{S}$ and concluding with the type $\S^k\mathscr{S}\otimes\S^k\mathscr{S}$, for any k. We remark that in **ILAL**, where the border of paragraph boxes is "rigid", we could only write a net, analogous to $\nabla\mathtt{S}^k_l$, concluding with type $\S^k(\mathscr{S}\otimes\mathscr{S})$ which would generally impede to get the right type for $\lceil t\rceil^l$. By the way, this is why [8] shows how to embed **BC**$^-$ but not **SRN** into **ILAL**. Indeed, in **BC**$^-$, composition and safe recursion schemes allow *linear* safe arguments only, i.e. the safe arguments are never duplicated.

To conclude, we recall what *stratification* means. It is a structural property underpinning the **PTIME**-sound cut elimination of Girard's Light Linear Logic (**LLL**) [5], and its variants **ILAL**, **ML**^{4}, and **LALL**. A net Π is *stratified* if the number of boxes around every node keeps being constant in every net we reach from Π by cut elimination. This work should be a step further towards studying how the *stratification* is compatible with the predicative analysis of **PTIME**-sound computations that **SRN** embodies.

Acknowledgments. We want to thank the anonymous referees whose questions helped us to better address the points subject of this work.

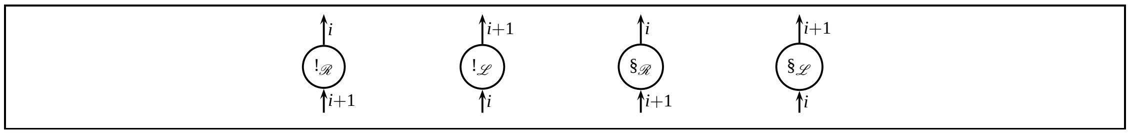


Figure 3: Constraints on the indexing. The nodes we omit have the same index on all of their incident edges.

2 Light Affine Logic by Levels (LALL)

The language of formulæ. First, for any fixed countable set $\mathscr{V}$ of propositional variables, the set $\mathscr{F}$ of formulæ is generated by the following grammar:

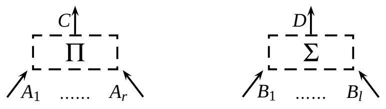
$$F ::= \mathscr{S} \mid \alpha \mid F \otimes F \mid F \multimap F \mid \forall\alpha.F \mid\ !F \mid \S F \qquad \alpha \in \mathscr{V}$$

where $\mathscr{S}$ is a propositional constant. Second, we define the quotient $\mathscr{F}_{\mathscr{S}}$ of $\mathscr{F}$ by assuming:

$$\mathscr{S} = \forall\alpha.(\alpha \multimap (\mathbb{B} \multimap \mathscr{S} \multimap \alpha) \multimap \alpha) \tag{4}$$

among the elements of $\mathscr{F}$. Namely, (4) says that $\mathscr{S}$ represents Scott words [1]. The formulæ we shall effectively use are the equivalence classes in $\mathscr{F}_{\mathscr{S}}$. Every time we label an edge of a net of **LALL** by $\mathscr{S}$, we can also label that edge by any “unfolding” of $\mathscr{S}$ that obeys (4). $A\left[{}^{B}/_{\alpha}\right]$ is the substitution of every free occurrence of α in A with B.

Proof structures and nets. LALL is a language of nets. Nets will be defined as particular proof structures. Given the nodes in Figure 2, we say that *an Axiom node and a Dæmon nodes are proof structures*. Moreover, given two proof structures Π and Σ:



denoted as $\Pi \triangleright A_1,\ldots,A_r \vdash C$ and $\Sigma \triangleright B_1,\ldots,B_l \vdash D$, respectively, with $r,l \geq 0$, then all the graphs inductively built from Π and Σ by the rule schemes in Figure 4 are proof structures.

If $\Pi \triangleright \Gamma \vdash A$, we say that Π *proves* the sequent $\Gamma \vdash A$. The *inputs* (resp. *outputs*) of Π are the edges labelled Γ (resp. A). The set of the nodes of Π is V_Π, and E_Π is the set of edges. The *size* $|\Pi|$ of Π is the cardinality of V_Π. The *depth* $\partial(x)$ of a node or edge $x \in V_\Pi \cup E_\Pi$ is the number of nested !-boxes containing x. The *depth* $\partial(\Pi)$ of Π is the greatest depth among the nodes of Π.

Every !-box simultaneously introduces one Bang R node and at most one Bang L node, recording this by the box border as in Figure 4.

Definition 1 (Indexing and Nets, adapted from [3]) *Let Π be a proof structure.*

1. *An* indexing *for Π is a function I from the edges of Π to $\mathbb{Z}$ that satisfies the constraints in Figure 3 and such that $I(e) = I(e')$, for every pair e,e' of inputs and output of Π.*

2. *A* net *is a proof structure that admits an indexing.*

3. *An indexing I of Π is* canonical *if Π has an edge e such that $I(e) = 0$, and $I(e') \geq 0$ for all edges e' of Π.*

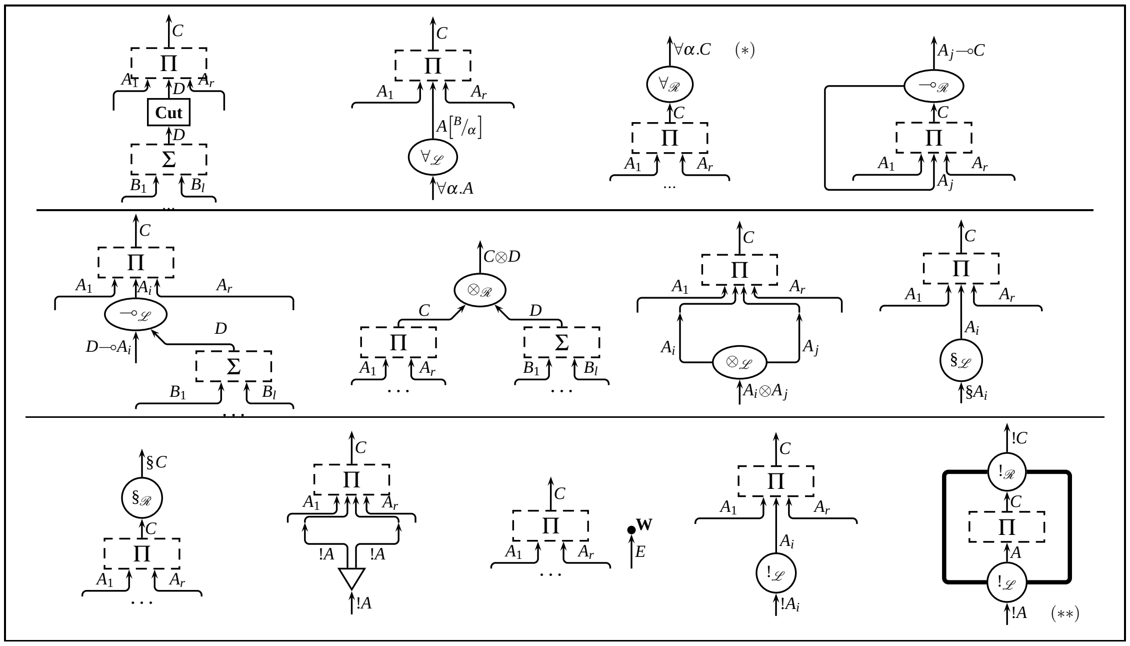


Figure 4: Inductive rule schemes to build proof structures of **LALL**. (*) α does not occur free in $A_1,\dots,A_r$. (**) A !-box, which has *at most* a single assumption.

As in [3], we can state that every net of **LALL** admits a unique canonical indexing.

The indexing tells that the nodes $!_{\mathscr{L}}$ and $\S_{\mathscr{L}}$ are not *dereliction* nodes. Remember that the dereliction rule of Linear Logic is inherently not stratified, because the cut-elimination is presence of a dereliction node may also "open" boxes. Instead, these nodes can be considered as auxiliary ports of §-boxes whose border is somewhat *fuzzy*. We mean that a §-box need not be contained in or disjoint from another box. Instead, it can "overlap" a !-box, and it can have more than one conclusion $\S_{\mathscr{R}}$. To distinguish §-boxes from the ! ones we adopt a dotted border.

Let I_0 be the canonical indexing of Π and $e \in E_\Pi$. The *level of* e is $l(e)$. It is defined as $I_0(e)$. The *level of* Π is $l(\Pi)$. It is defined as the greatest value assumed by I_0 on the edges of Π. We denote as B_Π the set of the !-boxes in Π, and it is naturally in bijection with the set of the $!_{\mathscr{R}}$ nodes in Π. Finally, for every net Π, and for $\dagger \in \{!,\S\}$, $\dagger^n\Pi$ denotes n nested $\dagger$-boxes around Π.

Cut elimination. We just recall its steps, which are standard. The *linear* cut elimination steps annihilate in the natural way a pair of linear nodes (Identity/Cut, $\multimap_{\mathscr{L}}$ / $\multimap_{\mathscr{R}}$, $\otimes_{\mathscr{L}}/\otimes_{\mathscr{R}}$, $\S_{\mathscr{L}}$ / $\S_{\mathscr{R}}$, $\forall_{\mathscr{L}}/\forall_{\mathscr{R}}$). The *exponential* cut elimination steps are of two kinds: $!_{\mathscr{L}}$ / $!_{\mathscr{R}}$ is reduced merging the two involved boxes which can be !-boxes as well as §-boxes with fuzzy borders. Instead, *contraction*/$!_{\mathscr{R}}$ duplicates the whole !-box cut with the contraction, as in **ILAL**. The *garbage collection* cut elimination steps involve the weakening or the dæmon nodes, cut with any other node. It is always possible to reduce such a cut with the help of some more weakening and dæmon nodes, as done in **ILAL** [2]. The set of cut nodes of Π is $\mathrm{cuts}(\Pi)$.

Proposition 2 (Cut-elimination) *Every* **LALL** *net reduces to a cut-free net.*

A direct proof would be very long; anyway, such a proof directly follows from the proof of the namesake propositions in **ILAL** and **ML**[4]. Please notice that the presence of fixpoints (i.e. the recursive type $\mathscr{S}$) does not affect the proof in any way, because the cut-elimination independently by the formulæ labelling the edges of a net. This is not true in full Linear Logic.

3 Polynomial time soundness of LALL

We adapt [3] to prove the cut elimination **PTIME**-soundness in presence of unconstrained Weakening, which we introduce for easy of programming since it is handy to erase nets structure. Let us fix a proof net Π to reduce. We define an ordering over cuts(Π) that determines which cuts to reduce first.

A graph theoretic path in any proof net Π is *exponential* if it contains a, possibly empty, sequence of consecutive contractions and stops at a $!_{\mathscr{L}}$ node.

Let $B, C \in B_{\Pi}$. Let $B \prec_1^L C$ if the roots of B and C lie at the same level, and the root of B is in cut with an exponential path that enters an auxiliary port of C. $\preceq^L$ is the reflexive and transitive closure of $\prec_1^L$. One can show that $\preceq^L$ is a partial order, *upward arborescent*: for every C there is at most one B such that $B \prec_1^L C$.

Let $c, c' \in$ cuts(Π). We write $c \leq c'$ iff one of the following conditions holds. (i) c' is connected to a weakening or a dæmon, and c is not. (ii) The condition (i) is false but $l(c) < l(c')$ holds. (iii) The conditions (i) and (ii) are false, so $l(c) = l(c')$. In this case, $c \leq c'$ iff: (a) either c' is connected to a contraction, and c is not, or (b) c, c' are connected to a contraction on one side, to the boxes B, B', respectively, on the other, and $B \preceq^L B'$.

Definition 3 (Canonical normalization) *A sequence of normalization steps that starts from a given proof net* Π *is* canonical *whenever smaller cuts relatively to* $\leq$ *are eliminated before higher ones.*

Theorem 4 (Polynomial bound for LALL) *Let* Π *be an* **LALL** *proof net of size* s*, level* l*, and depth* d*. Then, every canonical reduction is at most* $(l+1)s^{(d+2)^l}$ *steps long.*

The proof strategy coincides with the one in [3], with the following adaptation: the reduction of the garbage collection steps is always delayed till the end.

4 Preliminary notions about SRN

We recall from Section 1 that $\mathbf{SRN}^{\mathsf{n};\mathsf{s}}$ is the subset of **SRN** whose terms have normal arity n, and safe arity s. If not otherwise stated, $\overrightarrow{t}^m = t_1, \ldots, t_m$ will always denote sequences of $m \geq 0$ terms of **SRN**. Moreover, we write $|\overrightarrow{t}^m| \leq l$, for some $l > 0$, meaning that the size of every term t_i is not greater than l. Now, from [4], we recall that, for every t in $\mathbf{SRN}^{\mathsf{n};\mathsf{s}}$, and $\overrightarrow{x} = x_1, \ldots, x_{\mathsf{n}}$, $\overrightarrow{y} = y_1, \ldots, y_{\mathsf{s}}$:

$$|t(\overrightarrow{x};\overrightarrow{y})| \leq p_t(|x_1|, \ldots, |x_{\mathsf{n}}|) + \max\{|y_1|, \ldots, |y_{\mathsf{s}}|\} \tag{5}$$

where p_t is the *characteristic polynomial of* t which is non-decreasing and depends on t. We notice that if u is a subterm of t, then $\partial(p_u) \leq \partial(p_t)$. At last, we define the *composition degree* $\partial_{\mathsf{C}}(t)$ and the *recursion degree* $\partial_{\mathsf{R}}(t)$ of t, as the functions that count resp. the number of safe composition and recursion schemes inside t.

Definition 5 (The Term Bounding Function $\boldsymbol{tb}.(\cdot)$) *Let* t *in* $\mathbf{SRN}^{\mathsf{n};\mathsf{s}}$ *and* $l \geq 0$*. We define* $tb.(\cdot)$*, that takes* t *and* l *as arguments, as* $tb_t(l) = p_t(l, \ldots, l) + l$*.*

Fact 6 ($\boldsymbol{tb}.(\cdot)$ Bounds the Output Length of $t \in$ SRN) *For every* t *in* $\mathbf{SRN}^{\mathsf{n};\mathsf{s}}$*,* $l \geq 0$*, and sequences* $\overrightarrow{x}, \overrightarrow{y}$ *such that* $|\overrightarrow{x}|, |\overrightarrow{y}| \leq l$*, we have* $|t(\overrightarrow{x};\overrightarrow{y})| \leq tb_t(l)$*.*

$$
\begin{array}{rcll}
x^A \in V & \Rightarrow & x \in \Lambda_V^A & \\
m \geq 1,\, M \in \Lambda_V^A, x_1 \in \Lambda_V^{A_1}, \ldots, x_m \in \Lambda_V^{A_m} & \Rightarrow & (\lambda \otimes_{i=1}^m x_i^{A_i}.M) \in & \\
 & & \in \Lambda_V^{A_1 \otimes \ldots \otimes A_m \multimap A} & (6)\\
M \in \Lambda_U^{A' \multimap A}, N \in \Lambda_W^{A'}, U \cup W \subseteq V, U \cap W = \emptyset & \Rightarrow & (MN) \in \Lambda_V^A & \\
m \geq 2, \big(1 \leq i \neq j \leq m \Rightarrow N_i \in \Lambda_{W_i}^{A_i}, & & & \\
W_i \cap W_j = \emptyset, W_i \subseteq V\big) & \Rightarrow & (\otimes_{i=1}^m N_i) \in \Lambda_V^{A_1 \otimes \ldots \otimes A_m} & (7)\\
M \in \Lambda_V^A & \Rightarrow & \Lambda\alpha.M \in \Lambda_V^{\forall \alpha.A} & (8)\\
M \in \Lambda_V^{\forall \alpha.A} & \Rightarrow & M\{A'\} \in \Lambda_V^{A\left[{}^{A'}/_{\alpha}\right]} & (9)
\end{array}
$$

Figure 5: Typed II order affine λ-terms.

Definition 7 (The Net Bounding Function *nb*$_\cdot(\cdot)$) *Let t in* $\mathbf{SRN}^{\mathsf{n;s}}$ *and* $l \geq 0$. *We define* $nb_\cdot(\cdot)$, *that takes t and l as arguments, as* $nb_t(l) = tb_t(tb_t(\ldots tb_t(l)\ldots))$, *with* $\partial_R(t) + \partial_C(t)$ *occurrences of* $tb_t(\cdot)$.

Fact 8 (*nb*$_\cdot(\cdot)$ is a Polynomial) *For every fixed t in* $\mathbf{SRN}^{\mathsf{n;s}}$, $nb_t(l)$ *is a polynomial in the free variable l, whose degree is* $\partial(p_t)^{\partial_C(t)+\partial_R(t)}$.

5 Preliminary useful nets in LALL

We introduce a first set of nets useful to define the embedding from **SRN** to **LALL**. However, whenever neither boxes, nor contractions are used in a given net Π, whose conclusion has type A, to save space, we represent Π by means of a λ-term. The term belongs to the set Λ_V^A of polymorphic typed *affine* λ-terms with variables in V, patterns, tuples, and type $A \in \mathscr{F}_{\mathscr{S}}$. Figure 5 defines Λ_V^A. (6) introduces λ-abstractions on a tuple pattern, while (7) introduces tuples. The application is left-associative. We shall drop useless parenthesis to avoid cluttering the terms. For any A and V, the terms in Λ_V^A rewrite under the standard β-reduction, extended with the following two rules: (i) $(\lambda \otimes_{i=1}^m x_i.M)(\otimes_{i=1}^m N_i) \rightarrow_\beta M\left[{}^{N_1}/_{x_1} \cdots {}^{N_m}/_{x_m}\right]$, where $M\left[{}^{N_1}/_{x_1} \cdots {}^{N_m}/_{x_m}\right]$ stands for the simultaneous substitution of N_i for x_i, with $1 \leq i \leq m$, and (ii) $(\Lambda\alpha.M)\{B\} \rightarrow_\beta M\left[{}^{B}/_{\alpha}\right]$.

Booleans. The type of booleans is $\mathbb{B} = \forall\gamma.\gamma \multimap \gamma \multimap \gamma$ whose representative nets are:

$$
\begin{array}{ll}
\mathtt{F} = \Lambda\gamma.\lambda x^\gamma y^\gamma.x \rhd\ \vdash \mathbb{B} & \text{(True)}\\
\mathtt{T} = \Lambda\gamma.\lambda x^\gamma y^\gamma.y \rhd\ \vdash \mathbb{B} & \text{(False)}\\
\nabla\mathtt{B}[b] = b\,\{\mathbb{B}\otimes\mathbb{B}\}\,(\mathtt{T}\otimes\mathtt{T})\,(\mathtt{F}\otimes\mathtt{F})\ \rhd \mathbb{B} \vdash \mathbb{B}\otimes\mathbb{B} & \text{(Duplication)}
\end{array}
$$

The net $\nabla\mathtt{B}[b]$ duplicates any boolean we may plug into b by a Cut node.

Church words or, simply, words. The type of words is $\mathscr{C} = \forall\alpha.!(\alpha \multimap \alpha) \multimap !(\alpha \multimap \alpha) \multimap \S(\alpha \multimap \alpha)$. Figure 6(a) introduces the successor $\mathtt{SuccC}_0[w]$, where w identifies the lowermost dangling edge. It should be trivial to recover $\mathtt{SuccC}_1[w]$ from $\mathtt{SuccC}_0[w]$. Figure 6(c) introduces $\varepsilon\mathtt{C}$. If w is a natural number in binary notation, $\overline{w}$ is its usual representation by a net. Figures 6(b) and 6(d) introduce nets that invert the bits inside any $\overline{w}$, plugged by Cut into the dangling input of $\mathtt{RevC}[w]$.

Scott words. Intuitively, the type $\mathscr{S}$ of Scott words describes a tuple of booleans. On Scott words

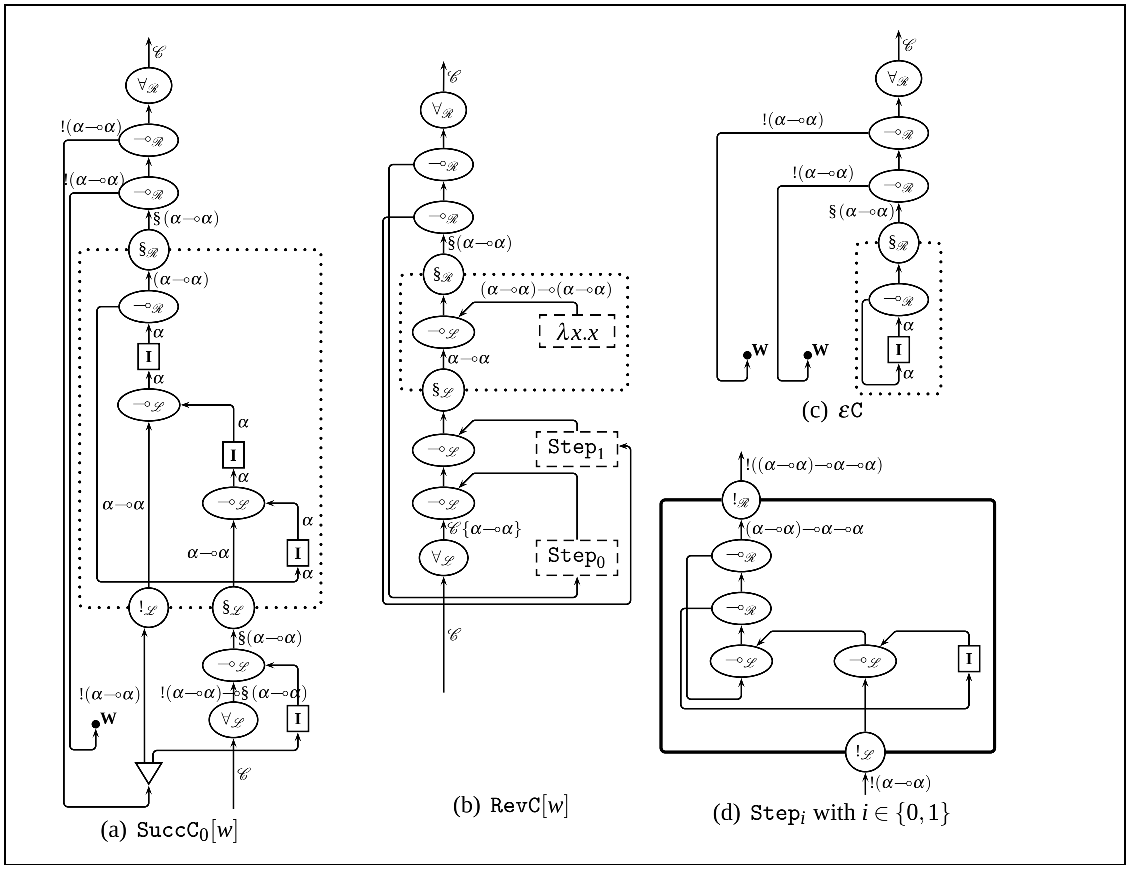

(a) $\mathtt{SuccC}_0[w]$
(b) $\mathtt{RevC}[w]$
(c) $\varepsilon\mathtt{C}$
(d) $\mathtt{Step}_i$ with $i \in \{0,1\}$

Figure 6: (Church) Words.

we have the following nets:

$$\varepsilon\mathtt{S} = \Lambda\alpha.\lambda x^{\alpha} y^{\mathbb{B} \multimap \mathscr{S} \multimap \alpha}.x \triangleright\ \vdash \mathscr{S} \qquad \text{(Empty Scott word)}$$

$$\mathtt{SuccS}_0[s] = \Lambda\alpha.\lambda x^{\alpha} y^{\mathbb{B} \multimap \mathscr{S} \multimap \alpha}.y\,\mathtt{F}\,s \triangleright \mathscr{S} \vdash \mathscr{S} \qquad \text{(Successor zero)}$$

$$\mathtt{SuccS}_1[s] = \Lambda\alpha.\lambda x^{\alpha} y^{\mathbb{B} \multimap \mathscr{S} \multimap \alpha}.y\,\mathtt{T}\,s \triangleright \mathscr{S} \vdash \mathscr{S} \qquad \text{(Successor one)}$$

$$\mathtt{PredS}[s] = s\,\{\mathscr{S}\}\,\varepsilon\mathtt{S}\,(\lambda b^{\mathbb{B}} w^{\mathscr{S}}.w) \triangleright \mathscr{S} \vdash \mathscr{S} \qquad \text{(Predecessor)}$$

$$\mathtt{CondS}[s,x,y] = \mathtt{PrepS}[s]\,\{\mathscr{S}\}\,\varepsilon\mathtt{S}\,(\lambda b^{\mathbb{B}}.b\,\{\mathscr{S}\}\,y x) \triangleright \mathscr{S},\mathscr{S},\mathscr{S} \vdash \mathscr{S} \qquad \text{(Conditional)}$$

$$\mathtt{PrepS}[s] = s\,\{\mathscr{S}\}\,\mathtt{SuccS}_0[\varepsilon\mathtt{S}]\,\big(\lambda b^{\mathbb{B}} \lambda w^{\mathscr{S}}.b\{\mathscr{S} \multimap \mathscr{S}\} (\lambda x^{\mathscr{S}}.\mathtt{SuccS}_0[x])\,(\lambda x^{\mathscr{S}}.\mathtt{SuccS}_1[x])\,w\big) \triangleright \mathscr{S} \vdash \mathscr{S} \qquad \text{(Preprocessing)}$$

We remark that $\mathtt{SuccS}_0[s]$ adds to s the least significant bit $\mathtt{T}$, which stands for the digit 0, and $\mathtt{SuccS}_1[s]$ adds $\mathtt{F}$, instead, which stands for 1. $\mathtt{PredS}[s]$ shifts s to its right deleting the *least significant* bit. So:

Remark 9 *A Scott word is in fact a stack of bits, the least significant bit being on the top of the stack.*

Moreover, $\mathtt{CondS}[s,x,y]$ branches a computation, depending on the value of s. It yields x if the least significant bit of s is 0, *or* if $s = \varepsilon\mathtt{S}$, while it yields y if the least significant bit of s is 1. The preprocessing avoids to return $\varepsilon\mathtt{S}$: if $s = \varepsilon\mathtt{S}$, then s becomes $\mathtt{SuccS}_0[\varepsilon\mathtt{S}]$. Also, the three assumptions of type $\mathscr{S}$ in $\mathscr{S},\mathscr{S},\mathscr{S} \vdash \mathscr{S}$ specify the type of s, x, and y, respectively.

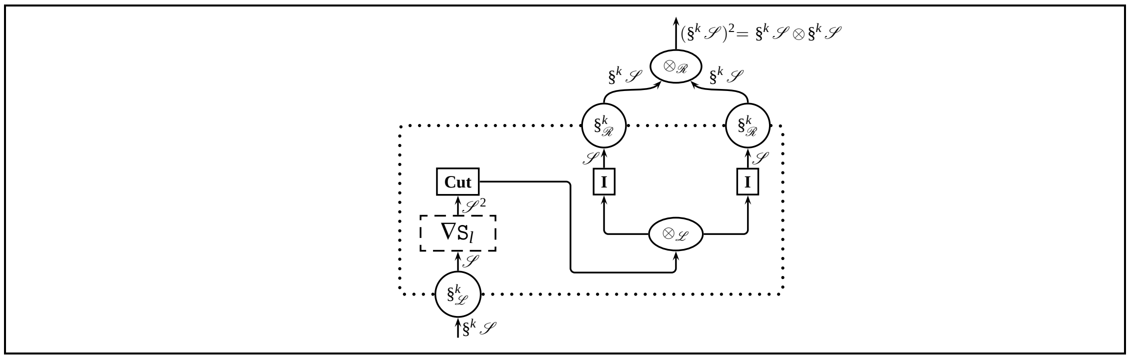


Figure 7: The generalized duplication $\nabla\mathtt{S}^k_l[s]$ of Scott words.

Fact 10 (Relation between naturals, Scott words, and words-as-terms) *Every sequence $(d_1,\ldots,d_l)$ with $d_1,\ldots,d_l \in \{0,1\}$ and $l \geq 0$, identifies uniquely a number $n = 2^{l-1}\cdot d_l + \cdots + 2^0 \cdot d_1 \in \mathbb{N}$. So, both the term of* **SRN** $\mathtt{s}_{d_l}(;\ldots\mathtt{s}_{d_1}(;0)\ldots)$ *and the Scott word $[n]$ identify n, too. We say that the sequence, as well as the Scott number and the* **SRN** *term*, represent *n.*

We underline that an infinite number of sequences, and of terms, represent the same n.

Scott words to words. For any $l \geq 0$, $\mathtt{StoC}_l[s] \rhd \mathscr{S} \vdash \mathscr{C}$ is inductively defined on l:

$$\mathtt{StoC}_0[s] = \varepsilon\mathtt{C}$$
$$\mathtt{StoC}_l[s] = s\,\{\mathscr{C}\}\,\varepsilon\mathtt{C}\,(\lambda x^{\mathbb{B}}y^{\mathscr{S}}.x\,(\lambda z^{\mathscr{C}}.\mathtt{SuccC}_0[z])\,(\lambda z^{\mathscr{C}}.\mathtt{SuccC}_1[z])\,\mathtt{StoC}_{l-1}[s])$$

The net $\mathtt{StoC}_l[s]$ normalizes to the word $\overline{w}$, for any Scott word *at most as long as l*.

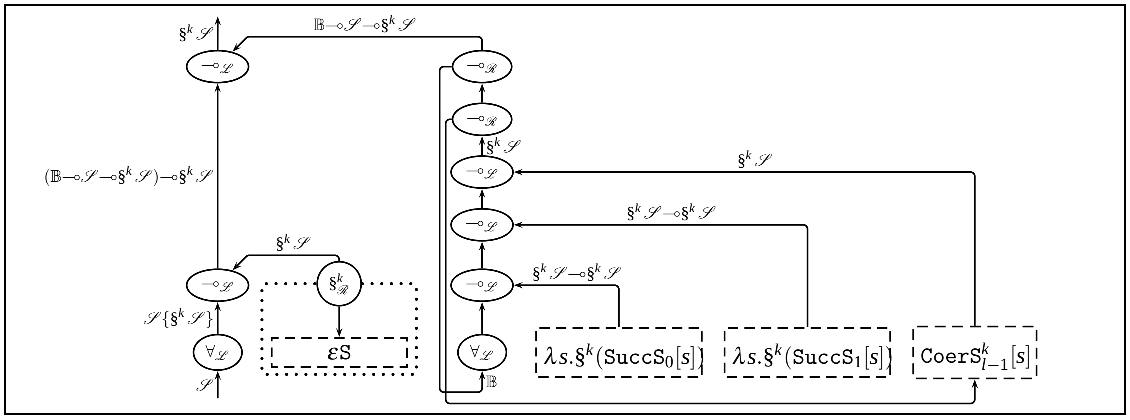
Duplicating Scott words. For any $l \geq 0$, the net $\nabla\mathtt{S}_l[s]$ is inductively defined on l:

$$\nabla\mathtt{S}_0[s] = \varepsilon\mathtt{S}\otimes\varepsilon\mathtt{S}$$
$$\nabla\mathtt{S}_l[s] = s\,\{\mathscr{S}\otimes\mathscr{S}\}\,(\varepsilon\mathtt{S}\otimes\varepsilon\mathtt{S})\,\big(\lambda b^{\mathbb{B}}s^{\mathscr{S}}.b\{\mathscr{S}^2 \multimap \mathscr{S}^2\}\,(\lambda x^{\mathscr{S}}\otimes y^{\mathscr{S}}.\mathtt{SuccS}_0[x]\otimes\mathtt{SuccS}_0[y])$$
$$(\lambda x^{\mathscr{S}}\otimes y^{\mathscr{S}}.\mathtt{SuccS}_1[x]\otimes\mathtt{SuccS}_1[y])\,\nabla\mathtt{S}_{l-1}[s]\big)$$

such that $\nabla\mathtt{S}_l[s] \rhd \mathscr{S} \vdash \mathscr{S}^2$, where $\mathscr{S}^2 = \mathscr{S}\otimes\mathscr{S}$. The net $\nabla\mathtt{S}_l[s]$ builds two copies of any Scott word *at most as long as l*. The generalization $\nabla\mathtt{S}^k_l[s] \rhd \S^k\mathscr{S} \vdash \S^k\mathscr{S}\otimes\S^k\mathscr{S}$ of $\nabla\mathtt{S}_l[s]$ duplicates a given Scott word *at most as long as l* which lies inside $k \geq 0$ paragraph boxes. Specifically, $\nabla\mathtt{S}^0_l[s]$ is $\nabla\mathtt{S}_l[s]$, while $\nabla\mathtt{S}^k_l[s]$ is in Figure 7, with $k > 0$, which is the only net that exploits the *fuzzy borders* of paragraph boxes. By induction on k, $|\nabla\mathtt{S}^k_l[s]| = 19 + 89l + 3k \in O(l)$.

Coercing Scott words. For any $k,l \geq 0$, we define $\mathtt{CoerS}^k_l[s] \rhd \mathscr{S} \vdash \S^k\mathscr{S}$ by cases on k, and by induction on l. If $k = 0$, then $\mathtt{CoerS}^0_l[s]$ is the node Axiom. Otherwise, the net is in Figure 8, where, for $i \in \{0,1\}$, $\lambda s.\S^k(\mathtt{SuccS}_i[s]) \rhd\ \vdash \S^k\mathscr{S} \multimap \S^k\mathscr{S}$ *denotes* the net that we build by: (i) enclosing $\mathtt{SuccS}_i[s]$ into k paragraph boxes to get $\S^k(\mathtt{SuccS}_i[s]) \rhd \S^k\mathscr{S} \vdash \S^k\mathscr{S}$, and (ii) adding an Implication R to $\S^k(\mathtt{SuccS}_i[s])$ so to close it and get its type $\S^k\mathscr{S} \multimap \S^k\mathscr{S}$. The net $\mathtt{CoerS}^k_l[s]$ reconstructs a given Scott word *at most as long as l* into an identical Scott word inside k paragraph boxes. We can show that $|\mathtt{CoerS}^k_l[s]| = 43l + 3kl \in O(l)$.

Lifting. Let $\Pi \rhd \overrightarrow{\mathscr{S}}^{\mathsf{n}}, \overrightarrow{\S^k\mathscr{S}}^{\mathsf{s}} \vdash \S^k\mathscr{S}$ for some $\mathsf{n},\mathsf{s},k \geq 0$. For every $k' \geq 0$ we can build $\mathtt{Lift}_{k'}[\Pi] \rhd \overrightarrow{\mathscr{S}}^{\mathsf{n}}, \overrightarrow{\S^{k+k'}\mathscr{S}}^{\mathsf{s}} \vdash \S^{k+k'}\mathscr{S}$ by: (i) enclosing Π into k' paragraph boxes, getting Π', and (ii) plugging the



$\S^k\mathscr{S}$
$\mathbb{B}\multimap\mathscr{S}\multimap\S^k\mathscr{S}$
$\multimap_{\mathscr{L}}$
$\multimap_{\mathscr{R}}$
$(\mathbb{B}\multimap\mathscr{S}\multimap\S^k\mathscr{S})\multimap\S^k\mathscr{S}$
$\S^k\mathscr{S}\multimap\S^k\mathscr{S}$
$\S^k_{\mathscr{R}}$
$\mathscr{S}\{\S^k\mathscr{S}\}$
$\forall_{\mathscr{L}}$
εS
$\lambda s.\S^k(\texttt{SuccS}_0[s])$
$\lambda s.\S^k(\texttt{SuccS}_1[s])$
$\texttt{CoerS}^k_{l-1}[s]$
$\mathscr{S}$
$\mathbb{B}$

Figure 8: The coerce net $\texttt{CoerS}^k_l[s]$ on Scott words.

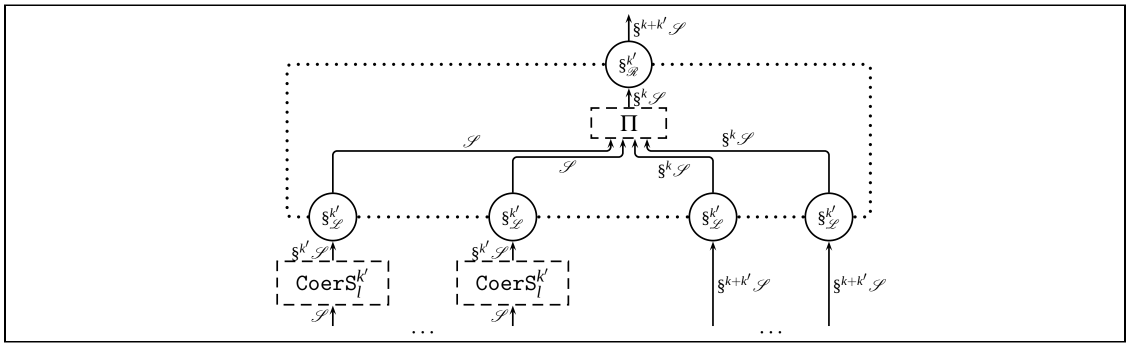


Figure 9: The *Lifting* $\texttt{Lift}_{k'}[\Pi]$ of a proof net Π.

conclusion of $\texttt{CoerS}^{k'}_l[s]$, using Cut, into every of the n premises with type $\S^{k'}\mathscr{S}$ of Π'. The net $\texttt{Lift}_{k'}[\Pi]$ is Π deepened inside k' paragraph boxes. The final net is in Figure 9.

Notice that $|\texttt{Lift}_{k'}[\Pi]| = |\Pi| + k'(\mathsf{n}+\mathsf{s}+1) + \mathsf{n}|\texttt{CoerS}^{k'}_l[s]| \in O(|\Pi|+l)$.

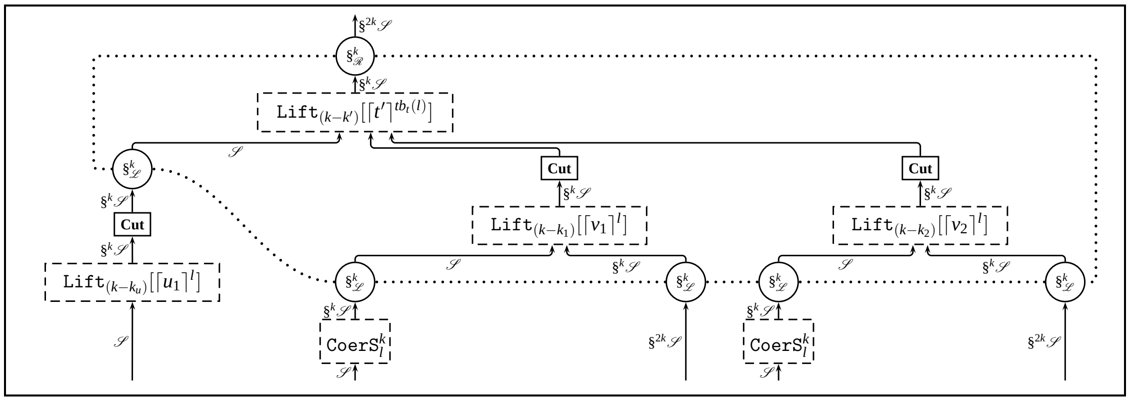
Contracting the premises of a net. Let $\Pi \triangleright \overrightarrow{A}, \S^k\mathscr{S}, \S^k\mathscr{S}, \overrightarrow{A}' \vdash A$ for some $l,k \geq 0$. We can build $\nabla^k_l[\Pi] \triangleright \overrightarrow{A}, \S^k\mathscr{S}, \overrightarrow{A}' \vdash A$ by: (i) writing Π' which is Π with a new Tensor L between the two outlined premises of type $\S^k\mathscr{S}$, and (ii) plugging the conclusion of $\nabla\texttt{S}^k_l[s] \triangleright \S^k\mathscr{S} \vdash \S^k\mathscr{S} \otimes \S^k\mathscr{S}$, by a Cut, into the premise of the new Tensor L in Π'. Notice that $|\nabla^k_l[\Pi]| = |\Pi| + |\nabla\texttt{S}^k_l[s]| + 2 \in O(l)$.

6 The embedding $\lceil\cdot\rceil^{\cdot}$ from SRN to LALL

The goal is to compositionally embed **SRN** into **LALL**, with a map as much analogous as possible to the natural, and inductively defined one from **BC**$^-$ into **ILAL** [8]. For any fixed n and s, the map $\lceil\cdot\rceil^{\cdot}$ takes a term t of **SRN**$^{\mathsf{n};\mathsf{s}}$ as first and $l \geq 0$ as second argument, and yields a net $\lceil t\rceil^l \triangleright \overrightarrow{\mathscr{S}}^{\mathsf{n}}, \overrightarrow{\S^k\mathscr{S}}^{\mathsf{s}} \vdash \S^k\mathscr{S}$, for some k. We define the map inductively on the first argument.

The base cases of $\lceil\cdot\rceil^{\cdot}$. Some of them are straightforward:

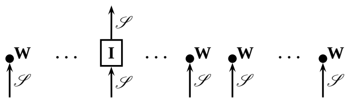
$$\begin{array}{lll}
\lceil\texttt{0}\rceil^l = \varepsilon\texttt{S} \triangleright\ \vdash \mathscr{S} & \lceil\texttt{s}_i\rceil^l = \texttt{SuccS}_i[s] \triangleright \mathscr{S} \vdash \mathscr{S} & (i \in \{0,1\}) \\
\lceil\texttt{p}\rceil^l = \texttt{PredS}[s] \triangleright \mathscr{S} \vdash \mathscr{S} & \lceil\texttt{B}\rceil^l = \texttt{CondS}[s,x,y] \triangleright \mathscr{S},\mathscr{S},\mathscr{S} \vdash \mathscr{S} &
\end{array}$$



$\S^{2k}\mathscr{S}$
$\S^k_{\mathscr{R}}$
$\S^k\mathscr{S}$
$\texttt{Lift}_{(k-k')}[\lceil t'\rceil^{tb_t(l)}]$
$\mathscr{S}$
Cut
$\S^k_{\mathscr{L}}$
$\texttt{Lift}_{(k-k_1)}[\lceil v_1\rceil^l]$
$\texttt{Lift}_{(k-k_2)}[\lceil v_2\rceil^l]$
$\texttt{Lift}_{(k-k_u)}[\lceil u_1\rceil^l]$
$\texttt{CoerS}^k_l$

Figure 10: The (partial) translation of $\circ[t', u_1, v_1, v_2]$ with missing contractions.

where, s,x,y denote the inputs of the nets they appear into. Concerning the projection, $\lceil \pi_i^{\mathsf{n;s}}\rceil^l$ is an Axiom that connects the i-th input to the conclusion, erasing all the other inputs by Weakening. An example with $1 \le i \le \mathsf{n}$, and, notice, $k = 0$ is:



The case of $\lceil\cdot\rceil^{\cdot}$ on the composition. We now focus on $t = \circ[t', u_1, \ldots, u_{\mathsf{m}}, v_1, \ldots, v_{\mathsf{r}}]$ such that t' be in $\mathbf{SRN}^{\mathsf{m;r}}$. Without loss of generality, we show how to build $\lceil t\rceil^l$ by assuming $\mathsf{m} = \mathsf{n} = \mathsf{s} = 1$, and $\mathsf{r} = 2$. By induction we have:

$$\begin{array}{cc} \lceil t'\rceil^{tb_t(l)} \triangleright \mathscr{S}, \S^{k'}\mathscr{S}, \S^{k'}\mathscr{S} \vdash \S^{k'}\mathscr{S} & \lceil u_1\rceil^l \triangleright \mathscr{S} \vdash \S^{k_u}\mathscr{S} \\ \lceil v_i\rceil^l \triangleright \mathscr{S}, \S^{k_i}\mathscr{S} \vdash \S^{k_i}\mathscr{S} & (i \in \{1,2\}) \end{array}$$

By letting $k = \max\{k', k_u, k_1, k_2\}$, we get:

$$\begin{array}{ll} \texttt{Lift}_{(k-k')}[\lceil t'\rceil^{tb_t(l)}] \triangleright \mathscr{S}, \S^k\mathscr{S}, \S^k\mathscr{S} \vdash \S^k\mathscr{S} & \texttt{Lift}_{(k-k_u)}[\lceil u_1\rceil^l] \triangleright \mathscr{S} \vdash \S^k\mathscr{S} \\ \texttt{Lift}_{(k-k_i)}[\lceil v_i\rceil^l] \triangleright \mathscr{S}, \S^k\mathscr{S} \vdash \S^k\mathscr{S} & (i \in \{1,2\}) \end{array}$$

Next, if we build Π' in Figure 10, then $\lceil t\rceil^l$ is $\nabla_l^{2k}[\nabla_l^0[\nabla_l^0[\Pi']]]$. The two occurrences of ∇_l^0 contract three "normal" inputs. One is from $\lceil u_1\rceil^l$. The other two are from $\lceil v_1\rceil^l, \lceil v_2\rceil^l$. The occurrence of ∇_l^{2k} contracts the single "safe" input of $\lceil v_1\rceil^l$ and $\lceil v_2\rceil^l$. We insist remarking the existence of $\lceil t\rceil^l$ for any $\mathsf{m},\mathsf{n},\mathsf{r},\mathsf{s}$. One can count: $|\lceil t\rceil^l| \le |\texttt{Lift}_{(k-k')}[\lceil t'\rceil^{tb_t(l)}]| + \sum_i^{\mathsf{m}} |\texttt{Lift}_{(k-k_i)}[\lceil u_i\rceil^l]| + \sum_j^{\mathsf{r}} |\texttt{Lift}_{(k-k_j)}[\lceil u_j\rceil^l]| + k(1+\mathsf{n}+\mathsf{s}'\mathsf{n}+\mathsf{s}'\mathsf{s}) + \mathsf{s}'\mathsf{n}|\texttt{CoerS}_l^k[s]|$. So, it follows $|\lceil t\rceil^l| \in O\left(|\lceil t'\rceil^{tb_t(l)}| + \sum_i^{\mathsf{m}} |\lceil u_i\rceil^l| + \sum_j^{\mathsf{r}} |\lceil v_j\rceil^l| + tb_t(l) + l\right)$.

The case of $\lceil\cdot\rceil^{\cdot}$ on the recursion. Let $t = \mathtt{r}[u_\varepsilon, u_0, u_1]$ with $u_\varepsilon \in \mathbf{SRN}^{\mathsf{n;s}}, u_0, u_1 \in \mathbf{SRN}^{\mathsf{n+1;s+1}}$. As for composition, we set $\mathsf{n} = \mathsf{s} = 1$ which is general enough to show the key point of the embedding. In the course of the iteration unfolding that $\lceil t\rceil^l$ carries out, the safe argument gets duplicated, so we must contract it. By induction:

$$\lceil u_\varepsilon\rceil^l \triangleright \mathscr{S}, \S^{k_\varepsilon}\mathscr{S} \vdash \S^{k_\varepsilon}\mathscr{S} \qquad \lceil u_i\rceil^{tb_t(l)} \triangleright \mathscr{S}, \mathscr{S}, \S^{k_i}\mathscr{S}, \S^{k_i}\mathscr{S} \vdash \S^{k_i}\mathscr{S} \qquad (i \in \{0,1\})$$

By letting $k = \max\{k_\varepsilon, k_0, k_1\}$, and using $\texttt{Lift}_{\cdot}[\cdot]$ in analogy to the translation of the composition, $\lceil t\rceil^l$ is in Figure 11. The Scott word that drives the recursion unfolding, becomes a word, and, then, it is necessary to reverse it by $\texttt{RevC}[w]$. Otherwise we would unfold the iteration according to a wrong bit order, as implied by Remark 9. Moreover, (i) Π projects the rightmost $\mathsf{n}+\mathsf{s}+1$-th element of type A it

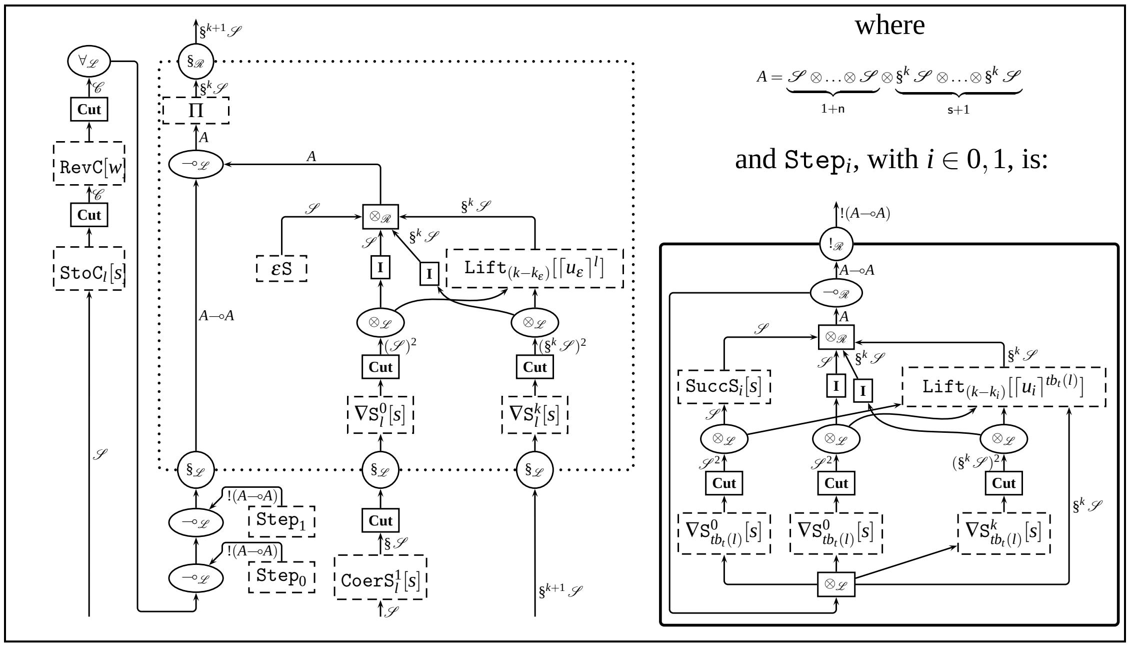

where
$A = \underbrace{\mathscr{S} \otimes \ldots \otimes \mathscr{S}}_{1+n} \otimes \underbrace{\S^k \mathscr{S} \otimes \ldots \otimes \S^k \mathscr{S}}_{s+1}$
and $\mathtt{Step}_i$, with $i \in 0,1$, is:
RevC[w]
StoC$_l$[s]
Cut
εS
Lift$_{(k-k_\varepsilon)}[\lceil u_\varepsilon \rceil^l]$
$\nabla S^0_l[s]$
$\nabla S^k_l[s]$
CoerS$^1_l[s]$
Step$_1$
Step$_0$
SuccS$_i[s]$
Lift$_{(k-k_i)}[\lceil u_i \rceil^{tb_t(l)}]$
$\nabla S^0_{tb_t(l)}[s]$
$\nabla S^k_{tb_t(l)}[s]$

Figure 11: Safe recursion.

gets in input and which contains the result, and (ii) the two nets $\boxed{\otimes_{\mathscr{R}}}$, $\boxed{\otimes_{\mathscr{L}}}$ are obvious trees of Tensor R and L nodes. Finally, we can prove $|\lceil t \rceil^l| \in O\left(|\lceil u_\varepsilon \rceil^l| + |\lceil u_0 \rceil^{tb_t(l)}| + |\lceil u_1 \rceil^{tb_t(l)}| + tb_t(l)\right)$.

Definition 11 (Representing a term by a net) *Let t be in* $\mathbf{SRN}^{\mathsf{n};\mathsf{s}}$, $l \in \mathbb{N}$, *and* $\Pi \triangleright \overrightarrow{\mathscr{S}}^{\mathsf{n}}\ \overrightarrow{(\S^k \mathscr{S})}^{\mathsf{s}} \vdash \S^k \mathscr{S}$, *for some* $k \in \mathbb{N}$. *Then,* Π k-simulates t with l-bounded inputs *if, for every pair of vectors of natural numbers* $\overrightarrow{x}^{\mathsf{n}}, \overrightarrow{y}^{\mathsf{s}}$, *such that* $|\overrightarrow{x}^{\mathsf{n}}|, |\overrightarrow{y}^{\mathsf{s}}| \leq l$, *the net we get by plugging* $\lceil x_1 \rceil^l, \ldots, \lceil x_{\mathsf{n}} \rceil^l, \S^k \lceil y_1 \rceil^l, \ldots, \S^k \lceil y_{\mathsf{s}} \rceil^l$ *into the inputs of* Π, *in the natural way, normalizes to* $\S^k \lceil z \rceil^l$, *whenever z is the result of* $t(\overrightarrow{x}^{\mathsf{n}}; \overrightarrow{y}^{\mathsf{s}})$.

Proposition 12 (SRN embeds into LALL) *Let* $l \geq 0$, *and* $t \in \mathbf{SRN}^{\mathsf{n};\mathsf{s}}$. *Then,* $\lceil t \rceil^l$ *k-simulates t with l-bounded inputs. Moreover, (i)* $k \leq \partial_R(t) \cdot 2^{\partial_C(t)}$, *and (ii)* $|\lceil t \rceil^l|$ *is* $O\left(l^{\partial(p_t)^{(\partial_C(t)+\partial_R(t))}}\right)$, *namely a polynomial in l.*

The statement holds by induction on t, using the definition of $\lceil \cdot \rceil^{\cdot}$, the size of every net that $\lceil \cdot \rceil^{\cdot}$ generates, the definitions of $nb_t(\cdot)$, and $tb_t(\cdot)$, together with Facts 8, 6, and 10.

7 Conclusions and further works

We have shown that the compatibility between the *predicative analysis* over recursive functions that **SRN** encodes, and the proof theoretical *stratification*, that regulates the complexity of some Light Logic that characterize **PTIME**, can be improved, provided that (i) the stratification we find in Light Linear Logic and **ILAL** relaxes to boxes with "fuzzy" border, as in **ML**^{4} or **LALL**, and (ii) we move to a representation of words *alternative to the standard one*, able both to represent the basic functions of **SRN** in constant time, and to exploit the independence of the cut elimination complexity from the logical complexity of the formulæ in a net.

As a consequence, every term t of **SRN** maps to a family $\langle \lceil t \rceil^l \rangle_{l \in \mathbb{N}}$ of nets in **LALL**, where $\lceil t \rceil^l$ simulates t with inputs *at most as long as* l. The number of paragraph modalities in the type of the conclusion of $\lceil t \rceil^l$ depends on the structural complexity of t. The size of $\lceil t \rceil^l$ is a polynomial in l whose degree depends on the degree of the characteristic polynomial of t and on the structural complexity of t.

As an example, the following program, which returns y if $w \neq 0$ contains a digit '0' that is not the lowermost digit, and z otherwise, is in **SRN** but not in $\mathbf{BC}^-$:

$$\begin{cases} g(0;y,z) & = z \\ g(s_i w;y,z) & = h(w;y,z,g(w;y,z)) \end{cases} , \qquad (*)$$

where $h(w;y,z,t) = \mathtt{cond}(\,;w,y,t)$. The embedding we propose gives the family of nets that implement it in **LALL**, while, it is worth remarking, it is unknown how to represent $g(w;y,z)$ inside **ILAL**.

Admittedly, the representation of a term of **SRN** by a family of nets, rather than as a unique net, is not standard. For example, one might be tempted to observe that every function with finite domain is the initial fragment of some polynomial time function, so **LALL** represents *every function with finite domain*. Beware, however, that *not every algorithm* is in **LALL**. For example, in analogy with [7], we show an algorithm `exp` that cannot exist as a net of **LALL** because it calculates a non-polytime function. `exp` will be defined using the traditional *non-predicative* recursive schemes, so that it is not a program of **SRN**. `exp` is defined as follows. We know that the two programs $\mathtt{concat}(x;y)$, which concatenates two strings of bits, and $\mathtt{double}(x;\,) = \mathtt{concat}(x;\pi^{1,0}_1(x;\,))$ belong to **SRN**. Then, we can define the recursive function `exp`:

$$\mathtt{exp}(0;\,) = \mathtt{s}_1(\,;\pi^{1,0}_1(0;\,)) \qquad \mathtt{exp}(\mathtt{s}_i(\,;x);\,) = \mathtt{double}(\mathtt{exp}(x;\,);\,) \qquad (i \in \{0,1\})$$

The program `exp` *is not* in **SRN** because of the position of the argument that drives the unfolding. *This reflects into* **LALL**, where $\lceil \mathtt{concat} \rceil^l : \mathscr{S}, \S\mathscr{S} \vdash \S\mathscr{S}$ and $\lceil \mathtt{double} \rceil^l : \mathscr{S} \vdash \S\mathscr{S}$ exist, but $\lceil \mathtt{double} \rceil^l$ cannot be iterated because of the form of its type. So, $\lceil \mathtt{exp} \rceil^l$ cannot be defined as a net of **LALL** using the constructions of this paper.

We conclude by an example about how the approach "**SRN** as family-of-proofs" we present here can be rewarding. We consider the following program:

$$\begin{cases} \mathtt{gt}(0,y) & = \mathtt{False} \\ \mathtt{gt}(\mathtt{s}_i x,y) & = \mathtt{if}\ (y=0)\ \mathtt{then\ True\ else}\ \mathtt{gt}(x,\mathtt{pred}(y)). \end{cases}$$

The program `gt` is such that $\mathtt{gt}(x,y) = \mathtt{True}$ iff $|x| > |y|$. It has a recursive definition more liberal than the primitive recursion scheme, as the recursive call of `gt` applies a function on the parameter y that does not drive the unfolding. Namely, `gt` incorporates a *double iteration*. Certainly, `gt` cannot exist in **SRN** in the form here above. Instead, **LALL** admits to represent `gt` as follows:

$$\begin{aligned} \mathtt{gt}^{\mathscr{C} \multimap \mathscr{S} \multimap \mathbb{B}} &= \lambda x^{\mathscr{C}}.\lambda y^{\mathscr{S}}.\pi_2\,(x\{\mathscr{S} \otimes \mathbb{B}\}(\mathtt{step})(y \otimes \mathtt{F})) \\ \mathtt{step}^{\mathscr{S} \otimes \mathbb{B} \multimap \mathscr{S} \otimes \mathbb{B}} &= \lambda s^{\mathscr{S}} \otimes b^{\mathbb{B}}.b\{\mathscr{S} \otimes \mathbb{B}\}(0 \otimes \mathtt{T})\left(s(0 \otimes \mathtt{T})(\lambda x^{\mathbb{B}}.\lambda y^{\mathscr{S}}.y \otimes \mathtt{F})\right) \end{aligned}$$

where $\mathbb{B}, \mathtt{T}, \mathtt{F}$ are at page 69. The existence of `gt` in **LALL** implies the existence of a family $\langle \mathtt{ord}_l \rangle_{l \in \mathbb{N}}$ of nets. Every $\mathtt{ord}_l$ takes two Scott words with *at most l bits*, and gives them back sorted according to their length. The definition of every $\mathtt{ord}_l$ is a net of **LALL** that we compactly write as a λ-term:

$$\begin{aligned} \mathtt{ord}_l^{\mathscr{S} \otimes \mathscr{S} \multimap \mathscr{S} \otimes \mathscr{S}} &= \lambda x^{\mathscr{S}} \otimes y^{\mathscr{S}}.(\lambda x_1^{\mathscr{S}} \otimes x_2^{\mathscr{S}}.\lambda y_1^{\mathscr{S}} \otimes y_2^{\mathscr{S}}.\mathtt{BtoB}\,(\mathtt{gt}\,(\mathtt{StoC}_l[x_1])\,y_1)\,(y_2 \otimes x_2))(\nabla \mathtt{S}_l\, x)(\nabla \mathtt{S}_l\, y) \\ \mathtt{BtoB}^{\mathbb{B} \multimap \mathbf{B}} &= \lambda x^{\mathbb{B}}.x\{\mathbf{B}\}(\Lambda \gamma.\lambda w^{\gamma} \otimes z^{\gamma}.w \otimes z)(\Lambda \gamma.\lambda w^{\gamma} \otimes z^{\gamma}.z \otimes w) \end{aligned}$$

$$
\begin{aligned}
L(X) &= \forall\alpha.!(X \multimap \alpha \multimap \alpha) \multimap \S(\alpha \multimap \alpha) \qquad \text{is the type of lists,}\\
\emptyset^{L(X)} &= \lambda c^{!(X\multimap\alpha\multimap\alpha)}.\lambda z^{\alpha}.z\\
\mathtt{sort}_l^{L(\mathscr{S})\multimap\S L(\mathscr{S})} &= \lambda t^{L(\mathscr{S})}.t\{L(\mathscr{S})\}(\mathtt{insert}_l)\ \emptyset^{L(\mathscr{S})}\\
\mathtt{insert}_l^{\mathscr{S}\multimap L(\mathscr{S})\multimap L(\mathscr{S})} &= \lambda n^{\mathscr{S}}.\lambda t^{L(\mathscr{S})}.\lambda c^{!(\mathscr{S}\multimap\alpha\multimap\alpha)}.\lambda z^{\alpha}.\\
&\quad (\lambda x^{\mathscr{S}} \otimes y^{\alpha}.cxy)((t\,\mathtt{putTop}\,[\mathtt{c}]_l)(\mathtt{CoerS}_l^1[n]\otimes z))\\
\mathtt{putTop}[c]_l^{\mathscr{S}\multimap\mathscr{S}\otimes\alpha\multimap\mathscr{S}\otimes\alpha} &= \lambda a^{\mathscr{S}}.\lambda b^{\mathscr{S}} \otimes t^{\alpha}.(\lambda i^{\mathscr{S}} \otimes \lambda j^{\mathscr{S}}.i\otimes c\,jt)(\mathtt{ord}_l\, a\otimes b)
\end{aligned}
$$

Figure 12: Insertion sort for Scott Words no longer than l.

where $\mathbf{B} = \forall\gamma.(\gamma\otimes\gamma) \multimap (\gamma\otimes\gamma)$ is a linear version of the booleans, $\mathtt{StoC}_l^{\mathscr{S}\multimap\mathscr{C}}$ is at page 71, and the *safe duplication* $\nabla\mathtt{S}_l^{\mathscr{S}\multimap\mathscr{S}\otimes\mathscr{S}}$ is at page 71.

Given $\mathtt{ord}_l$, we can write a family of insertion sorts that sort lists of Scott Words as much long as l. Figure 12 describes one element of the family. We warn the reader about the syntax we use. It does not perfectly adhere to the one in Figure 5, but nets would consume too much space. The effort to move from the terms in Figure 12 to the nets of **LALL** they represent should be a reasonably simple exercise. We observe that $\mathtt{putTop}_l$ is a linear algorithm that manipulates only the *head* of a given list. Instead, $\mathtt{insert}_l$ takes a number and a sorted list, and puts the number at the correct position of the list, so to preserve the sorting. While performing an iteration, $\mathtt{insert}_l$ does *not* add any paragraph § in front of the type of the output. The reason is that it exploits the general scheme that allows to write a perfectly linear *predecessor* on Church numerals in the λ-calculus [9]. Finally, $\mathtt{sort}_l$, iterates $\mathtt{insert}_l$ in the usual way, thus adding a § in front of its output type.

Future lines of research. We must say that the representation of **SRN** as a family of nets of **LALL** that we present in this work has been an alternative approach to the standard one, which would explore the relations between **SRN** and stratification by mapping a single term of **SRN** into a single net. That standard approach has been developed in [11, 10, 13]. Those works make some progress as compared to [8]. This means that they identify a Light Logic that strictly contains **ILAL**, and which allows to represent a strict extension of **BC**$^-$. However, the whole **SRN** still escapes any full representation inside a Light Logic. So, it has been natural to look for an alternative approach; and this brought us to this work.

Naturally enough, future work is about "integrating" both *by level technology* and *multimodality*. Multimodality is in the framework **MS** developed in the previously cited works [11, 10, 13]. The conjecture is that the two technologies together may lead to a more refined proof theoretical representation of the principles underpinning the definition of **SRN**, and of the predicative analysis it encodes, possibly increasing the set of algorithms that we can represent inside Light Logics.

References

[1] Martìn Abadi, Luca Cardelli & Gordon D. Plotkin (1993): *Types for Scott numerals*. Available at `http://lucacardelli.name/Papers/Notes/scott2.pdf`.

[2] Andrea Asperti & Luca Roversi (2002): *Intuitionistic Light Affine Logic*. *ACM Trans. Comput. Log.* 3(1), pp. 137–175. Available at `http://doi.acm.org/10.1145/504077.504081`.

[3] Patrick Baillot & Damiano Mazza (2010): *Linear Logic by Levels and Bounded Time Complexity*. *Theor. Comp. Sci.* 411(2), pp. 470–503.

[4] Stephen Bellantoni & Stephen A. Cook (1992): *A New Recursion-Theoretic Characterization of the Polytime Functions*. *Computational Complexity* 2, pp. 97–110.

[5] J.-Y. Girard (1998): *Light Linear Logic*. *Inf. Comput.* 143(2), pp. 175–204.

[6] D. Leivant (1994): *A Foundational Delineation of Poly-time*. *I.&C.* 110(2), pp. 391–420.

[7] Daniel Leivant (1993): *Stratified Functional Programs and Computational Complexity*. In: *POPL*, pp. 325–333.

[8] Andrzej S. Murawski & C.-H. Luke Ong (2004): *On an interpretation of safe recursion in light affine logic*. *Theor. Comput. Sci.* 318(1-2), pp. 197–223.

[9] L. Roversi (1999): *A P-Time Completeness Proof for Light Logics*. In: *Ninth Annual Conference of the EACSL (CSL'99)*, *LNCS* 1683, Springer-Verlag, Madrid (Spain), pp. 469 – 483.

[10] L. Roversi & L. Vercelli (2009): *A structural and local criterion for polynomial time computations*. Available at `http://www.di.unito.it/~rover/RESEARCH/PUBLICATIONS/2009-FOPARA/RoversiVercelli2009FOPARA.pdf`. Accepted for the publication in the post-proceedings of the Workshop FOPARA09 (FM2009 affiliated).

[11] L. Roversi & L. Vercelli (2009): *Some Complexity and Expressiveness Results on Multimodal and Stratified Proof Nets*. In: *Proceedings of TYPES'08*, pp. 306 – 322. Available at `http://dx.doi.org/10.1007/978-3-642-02444-3_19`.

[12] Kazushige Terui (2004): *Proof Nets and Boolean Circuits*. In: *Proceedings of LICS'04*, pp. 182–191.

[13] Luca Vercelli: *On the Complexity of Stratified Logics*. Ph.D. thesis, Scuola di Dottorato in Scienze e Alta Tecnologia — Università di Torino — Italy. Available at `http://arxiv.org/abs/1002.3453v1`.