# Elementary Affine Logic and the Call by Value Lambda Calculus

Paolo Coppola[1], Ugo Dal Lago[2], and Simona Ronchi Della Rocca[3]

[1] Università di Udine
[2] Università di Bologna
[3] Università di Torino

**Abstract.** Light and elementary linear logics have been introduced as logical systems enjoying quite remarkable normalization properties. Designing a type assignment system for pure lambda calculus from these logics, however, is problematic, as discussed in [1]. In this paper, we show that shifting from usual call-by-name to call-by-value lambda calculus allows to regain strong connections with the underlying logic. We will do this in the context of Elementary Affine Logic (EAL), designing a type system in natural deduction style assigning EAL formulae to lambda terms.

## 1 Introduction

Light and elementary linear logics [2–4] were all introduced as logical counterparts of complexity classes, namely polynomial and elementary time functions. After their introduction, they have been shown to be relevant for optimal reduction [5, 6], programming language design [4, 7] and set theory [8]. However, proof languages for these logics, designed through the Curry-Howard Correspondence, are syntactically quite complex and can hardly be proposed as programming languages. An interesting research challenge is the design of type systems assigning light logics formulae to pure lambda-terms, forcing the class of typable terms to enjoy the same remarkable normalization properties we can prove on logical systems. The difference between $\beta$-reduction and the normalization step in the logics, however, makes it difficult both getting the subject reduction property and inheriting the complexity properties from the logic, as discussed in [1]. Indeed, $\beta$-reduction is more permissive than the restrictive copying discipline governing calculi directly derived from light logics. Consider, for example, the following expression in $\Lambda_{LA}$ (see [7]):

$$let \;\; P \;\; be \;\; !x \;\; in \;\; N$$

This rewrites to $N\{x/P\}$ if $N$ is $!P$, but is not a redex if $N$ is, say, an application. It is not possible to map this mechanism into pure lambda calculus. The solution proposed by Baillot and Terui [1] in the context of light affine logic consists in defining a type-system which is strictly more restrictive than the one induced by the logic. In this way, they both achieve subject reduction and a strong notion

of polynomial time soundness. It is not clear, however, whether every light affine logic proof can be mapped to a typable term. Now, notice that mapping the above let expression to the application $(\lambda x.N)M$ is not meaningless if we shift from the usual call-by-name lambda-calculus to the call-by-value $\lambda$-calculus, where $(\lambda x.N)M$ is not necessarily a redex. In this paper, we make the best of this idea, introducing a type assignment system assigning to $\lambda$-calculus formulas of Elementary Affine Logic (EAL), that we call ETAS. ETAS enjoys the following remarkable properties:

- Every proof of EAL can be mapped into a type derivation in ETAS;
- (Call-by-value) subject reduction holds;
- Type-inference is decidable;
- Elementary bounds can be given on the length of any reduction sequence involving a typable term. A similar bound holds on the size of terms involved in the reduction.

The basic idea underlying ETAS consists in partitioning premises into three classes, depending on whether they are used once, or more than once, or they are in an intermediate status. We believe this approach can work for other light logics too, and some hints will be given.

The proposed system is the first one satisfying the given requirements for light logics. A notion of typability for $\lambda$-calculus has been defined in [5, 6] for EAL, and in [9] for Light Affine Logic. Type inference has been proved to be decidable. In both cases, however, the notion of typability is not preserved by $\beta$-reduction.

The paper is organized as follows: Section 2 recalls some preliminary notions about EAL and lambda-calculus, Section 3 introduces ETAS system, Section 4 and 5 explain ETAS main properties, namely complexity bounds and a type inference algorithm. Section 6 presents two possible extensions, allowing to reach completeness for elementary functions, and to apply our idea to other light logics. The Appendix contains some technical proofs.

## 2 Preliminaries

In this section we recall the proof calculus for Elementary Affine Logic, $\Lambda^{EA}$, and we discuss its relation with the $\lambda$-calculus.

**Definition 1.** *i) The set $\Lambda$ of terms of the $\lambda$-calculus is defined by the grammar $M ::= x \mid MM \mid \lambda x.M$, where $x \in Var$, a countable set of variables.*

*ii) The grammar generating the set $\Lambda^{EA}$ of terms of the $\lambda^{EA}$-calculus ($\Lambda^{EA}$) is obtained from the previous one by adding rules: $M ::= !(M) \left[{}^{M}/_{x}, \ldots, {}^{M}/_{x}\right] \mid [M]_{M=x,y}$ and by constraining all variables occurrence to occur (free) at most once.*

*iii)* EA-types *are formulas of Elementary Affine Logic (hereby* EAL*), and are generated by the grammar $A ::= \alpha \mid A \multimap A \mid !A$ where $\alpha$ belongs to a countable set of basic type constants. EA-types will be ranged over by $A, B, C$.*

*iv)* EA-contexts *are finite subsets of EA-type assignments to variables. Contexts are ranged over by $\Phi, \Psi$. If $\Phi = \{x_1 : A_1, \ldots, x_n : A_n\}$, then $dom(\Phi) =$*

$\{x_1, \ldots, x_n\}$. *Two contexts are* disjoint *if their domains have empty intersection.*

*iv) The type assignment system* $\vdash_{\text{NEAL}}$ *assigns EA-types to EA-terms, starting from a context. The system is given in Table 1. With a slight abuse of notation, we will denote by* NEAL *the set of typable terms in* $\Lambda^{EA}$.

Both $\Lambda^{EA}$ and $\Lambda$ are ranged over by $M, N, P, Q$. The context should help avoiding ambiguities. $\equiv$ denotes syntactic identity on terms, modulo names of bound variables and modulo permutation in the list ${}^{M}/x, \cdots, {}^{M}/x$ inside $!\,(M)\left[{}^{M}/x, \ldots, {}^{M}/x\right]$ and in contracted variables $x, y$ inside $[M]_{M=x,y}$.

$$\frac{}{\Phi, x : A \vdash_{\text{NEAL}} x : A}\ ax \qquad \frac{\Phi \vdash_{\text{NEAL}} M :!A \quad \Psi, x :!A, y :!A \vdash_{\text{NEAL}} N : B}{\Phi, \Psi \vdash_{\text{NEAL}} [N]_{M=x,y} : B}\ contr$$

$$\frac{\Phi, x : A \vdash_{\text{NEAL}} M : B}{\Phi \vdash_{\text{NEAL}} \lambda x.M : A \multimap B}\ (\multimap I) \qquad \frac{\Phi \vdash_{\text{NEAL}} M : A \multimap B \quad \Psi \vdash_{\text{NEAL}} N : A}{\Phi, \Psi \vdash_{\text{NEAL}} (M\ N) : B}\ (\multimap E)$$

$$\frac{\Psi_1 \vdash_{\text{NEAL}} M_1 :!A_1 \quad \cdots \quad \Psi_n \vdash_{\text{NEAL}} M_n :!A_n \quad x_1 : A_1, \ldots, x_n : A_n \vdash_{\text{NEAL}} N : B}{\Phi, \Psi_1, \ldots, \Psi_n \vdash_{\text{NEAL}} !\,(N)\left[{}^{M_1}/x_1, \ldots, {}^{M_n}/x_n\right] :!B}\ !$$

**Table 1.** Type assignment system for EA-terms. Contexts with different names are intended to be disjoint.

On $\Lambda$, both the call-by-name and the call-by-value $\beta$-reduction will be used, according to the following definition.

**Definition 2.** *i) The* call-by-name $\beta$-reduction *is the contextual closure of the following rule:* $(\lambda x.M)N \to_n M\{N/x\}$, *where* $M\{N/x\}$ *denotes the capture free substitution of* $N$ *to the free occurrences of* $x$ *in* $M$*;*

*ii)* Values $\mathcal{V}$ *are generated by the grammar* $V ::= x \mid \lambda x.M$ *where* $x$ *ranges over Var and* $M$ *ranges over* $\Lambda$. *Values are denoted by* $V, U, W$. *The* call-by-value $\beta$*-reduction is the contextual closure of the following rule:* $(\lambda x.M)V \to_v M\{V/x\}$ *where* $V$ *ranges over values.*

*iii) Let* $t \in \{n, v\}$: $\to_t^+$ *and* $\to_t^*$ *denote the transitive closure and the symmetric and transitive closure of* $\to_t$, *respectively.*

A term in $\Lambda^{EA}$ can be tranformed naturally in a term in $\Lambda$ by performing the substitutions which are explicit in it, and forgetting the modality !. Formally, the translation function $(\cdot)^* : \Lambda^{EA} \to \Lambda$ is defined by induction on the structure

of EA-terms as follows:

$$\begin{aligned}
(x)^* &= x \\
(\lambda x.M)^* &= \lambda x.(M)^* \\
(MN)^* &= (M)^*(N)^* \\
([M]_{N=x_1,x_2})^* &= (M)^*\{(N)^*/x_1, (N)^*/x_2\} \\
(!\,(N)\,[^{M_1}/x_1, \ldots, ^{M_n}/x_n])^* &= (N)^*\{(M_1)^*/x_1, \ldots, (M_n)^*/x_n\}
\end{aligned}$$

where $M\{^{M_1}/x_1, \cdots, ^{M_n}/x_n\}$ denotes the simultaneous substitution of all free occurrences of $x_i$ by $M_i$ $(1 \leq i \leq n)$.
The map $(\cdot)^*$ easily induces a type-assignment system for pure lambda-calculus: take NEAL and replace every occurrence of a term $M$ by $M^*$ in every rule. Normalization in EAL, however, is different from normalization in lambda-calculus — the obtained system does not even satisfy subject-reduction. Moreover, lambda calculus does not provide any mechanism for sharing: the argument is duplicated as soon as $\beta$-reduction fires. This, in turn, prevent from analyzing normalization in the lambda-calculus using the same techniques used in logical systems. This phenomenon has catastrophic consequences in the context of light affine logic, where polynomial time bounds cannot be transferred from the logic to pure lambda-calculus [1].
Consider now a different translation $(\cdot)^\# : \Lambda^{EA} \to \Lambda$:

$$\begin{aligned}
(x)^\# &= x \\
(\lambda x.M)^\# &= \lambda x.(M)^\# \\
(MN)^\# &= M^\#\ (N)^\# \\
([N]_{M=x,y})^\# &= \begin{cases} (N)^\#\{M/x, M/y\} & \textit{if } M \textit{ is a variable} \\ (\lambda z.(N)^\#\{z/x, z/y\})(M)^\# & \textit{otherwise} \end{cases} \\
(!\,(N)\,[^{M_1}/x_1, \ldots, ^{M_n}/x_n])^\# &= \begin{cases} (N)^\# \quad \textit{if } n = 0 \\ (!\,(N)\,[^{M_2}/x_2, \ldots, ^{M_n}/x_n])^\#\{M_1/x_1\} \\ \quad \textit{if } n \geq 1 \textit{ and } M_1 \textit{ is a variable} \\ (\lambda x_1.(!\,(N)\,[^{M_2}/x_2, \ldots, ^{M_n}/x_n])^\#)(M_1)^\# \\ \quad \textit{if } n \geq 1 \textit{ and } M_1 \textit{ is not a variable} \end{cases}
\end{aligned}$$

If lambda-calculus is endowed by ordinary $\beta$-reduction, then the two translation are almost equivalent. Indeed:

**Lemma 1.** *For every EA-term $M$, $(M)^\# \to_n^* (M)^*$.*

*Proof.* By induction on $M$.

However, it is not certainly true that $(M)^\# \to_v^* (M)^*$.
The map $(\cdot)^\#$, differently from $(\cdot)^*$ does not cause an exponential blowup on the lenght of terms. The *length* $L(M)$ of an $\lambda$-term $M$ is defined inductively as follows:

$$\begin{aligned}
L(x) &= 1 \\
L(\lambda x.M) &= 1 + L(M) \\
L(M\ N) &= 1 + L(M) + L(N)
\end{aligned}$$

The same definition can be extended to $EA$-terms by way of the following equations:

$$L(!\,(M)) = L(M) + 1$$
$$L(!\,(M)\,[^{M_1}/_{x_1}, \ldots, ^{M_n}/_{x_n}]) = L(!\,(M)\,[^{M_1}/_{x_1}, \ldots, ^{M_{n-1}}/_{x_{n-1}}]) + L(M_n) + 1$$
$$L([M]_{N=x,y}) = L(M) + L(N) + 1$$

**Proposition 1.** *For every $N \in \Lambda^{EA}$, $L(N^{\#}) \leq 2L(N)$.*

In the following section, we describe a type-system which types all the terms in (NEAL)$^{\#}$, satysfing call-by-value subject-reduction and enjoying many remarkable normalization properties.

## 3 The Elementary Type Assignment System

In this section we will define a type assignment system typing lambda-terms with EAL formulae. We want the system to be *almost* syntax directed, the difficulty being the handling of contraction and ! rules. This is solved by splitting the context into three parts, the *linear* context, the *intuitionistic* context, and the *parking* context. In particular the parking context is used to keep track of premises which must become modal in the future.

**Definition 3.** *i) An* EAL *formula $A$ is* modal *if it is $!B$ for some $B$, it is* linear *otherwise.*

*ii) A context is* linear *if it assigns linear EA-types to variables, while it is* intuitionistic *if it assigns modal EA-types to variables. If $\Phi$ is a context, $\Phi^L$ and $\Phi^I$ denote the linear and modal sub-contexts of $\Phi$, respectively.*

*iii) The Elementary Type Assignment System (*ETAS*) proves statements like $\Gamma \mid \Delta \mid \Theta \vdash M : A$ where $\Gamma$ and $\Theta$ are linear contexts and $\Delta$ is an intuitionistic context. The rules of the system are showed in Table 2. In what follows, $\Gamma$, $\Delta$ and $\Theta$ will range over linear, intuitionistic and parking contexts respectively.*

*iv) A term $M \in \Lambda$ is $EA$-*typable *if there are $\Gamma, \Delta, A$ such that $\Gamma \mid \Delta \mid \emptyset \vdash M : A$.*

Rules $A^L$ and $A^P$ are two variations on the classical axiom rule. Notice that a third axiom rule

$$\frac{}{\Gamma \mid x :!A, \Delta \mid \Theta \vdash x :!A}\ A^I$$

is derivable. Abstractions cannot be performed on variables in the parking context. The rule $E_{\multimap}$ is the standard rule for application. Rule ! is derived from the one traditionally found in sequent calculi and is weaker than the rule induced by NEAL via $(\cdot)^*$. Nevertheless, it is sufficient for our purposes and (almost) syntax-directed. The definition of an $EA$-typable term takes into account the auxiliary role of the parking context.

This system does not satisfy call-by-name subject-reduction. Consider, for example, the lambda term $M \equiv (\lambda x.yxx)(wz)$. A typing for it is the following:

$$y :!A \multimap !A \multimap A, w : A \multimap !A, z : A \mid \emptyset \mid \emptyset \vdash M : A$$

$$\frac{}{\Gamma, x : A \mid \Delta \mid \Theta \vdash x : A}\ A^L \qquad \frac{}{\Gamma \mid \Delta \mid x : A, \Theta \vdash x : A}\ A^P$$

$$\frac{\Gamma, x : A \mid \Delta \mid \Theta \vdash M : B}{\Gamma \mid \Delta \mid \Theta \vdash \lambda x.M : A \multimap B}\ I^L_{\multimap} \qquad \frac{\Gamma \mid \Delta, x : A \mid \Theta \vdash M : B}{\Gamma \mid \Delta \mid \Theta \vdash \lambda x.M : A \multimap B}\ I^I_{\multimap}$$

$$\frac{\Gamma_1 \mid \Delta \mid \Theta \vdash M : A \multimap B \quad \Gamma_2 \mid \Delta \mid \Theta \vdash N : A}{\Gamma_1, \Gamma_2 \mid \Delta \mid \Theta \vdash M\ N : B}\ E_{\multimap}$$

$$\frac{\Gamma_1 \mid \Delta_1 \mid \Theta_1 \vdash M : A}{\Gamma_2 \mid !\Gamma_1, !\Delta_1, !\Theta_1, \Delta_2 \mid \Theta_2 \vdash M : !A}\ !$$

**Table 2.** The Elementary Type Assignment System (ETAS). Contexts with different names are intended to be disjoint.

$M \to_n N$, where $N \equiv y(wz)(wz)$ and $y : !A \multimap !A \multimap A, w : A \multimap !A, z : A \mid \emptyset \mid \emptyset \not\vdash M : A$, because rule $E_{\multimap}$ requires the two linear contexts to be disjoint. Note that both $\emptyset \mid \emptyset \mid y : !A \multimap !A \multimap A, w : A \multimap !A, z : A \vdash M : A$ and $\emptyset \mid \emptyset \mid y : !A \multimap !A \multimap A, w : A \multimap !A, z : A \vdash N : A$, but this does not imply $N$ to be $EA$-typable. Moreover, $\lambda w.M \to_n \lambda w.M$, but while $M$ can be given type $(A \multimap !A) \multimap A$, $N$ cannot.
The subject reduction problem, however, disappears when switching from the call-by-name to the call-by-value reduction.

**Lemma 2 (Weakening Lemma).** *If $\Gamma_1 \mid \Delta_1 \mid \Theta_1 \vdash M : A$, then $\Gamma_1, \Gamma_2 \mid \Delta_1, \Delta_2 \mid \Theta_1, \Theta_2 \vdash M : A$.*

*Proof.* Easy induction. □

**Lemma 3 (Shifting Lemma).** *If $\Gamma, x : A \mid \Delta \mid \Theta \vdash M : B$, then $\Gamma \mid \Delta \mid x : A, \Theta \vdash M : B$.*

*Proof.* Easy induction. □

**Lemma 4 (Substitution Lemma).** *Suppose $\Gamma_1$ and $\Gamma_2$ to be disjoint. Then:*

- *If $\Gamma_1, x : A \mid \Delta \mid \Theta \vdash M : B$, $\Gamma_2 \mid \Delta \mid \Theta \vdash N : A$ and $N \in \mathcal{V}$, then $\Gamma_1, \Gamma_2 \mid \Delta \mid \Theta \vdash M\{N/x\} : B$.*
- *If $\Gamma_1 \mid \Delta \mid x : A, \Theta \vdash M : B$, $\Gamma_2 \mid \Delta \mid \Theta \vdash N : A$ and $N \in \mathcal{V}$, then $\Gamma_1 \mid \Delta \mid \Gamma_2, \Theta \vdash M\{N/x\} : B$.*
- *If $\Gamma_1 \mid \Delta, x : A \mid \Theta \vdash M : B$, $\Gamma_2 \mid \Delta \mid \Theta \vdash N : A$ and $N \in \mathcal{V}$, then $\Gamma_1, \Gamma_2 \mid \Delta \mid \Theta \vdash M\{N/x\} : B$.*

*Proof.* The first point can be easily proved by induction on the derivation for $\Gamma_1, x : A \mid \Delta \mid \Theta \vdash M : B$ using, in particular, the weakening lemma. Indeed, the hypothesis on $N$ is not needed.
Let us prove the second point (by the same induction). The case for $A^P$, can be proved by way of the previous lemmata. $I^L_{\multimap}$ and $I^I_{\multimap}$ are trivial. $E_{\multimap}$ comes

directly from the induction hypothesis. ! is very easy, because $x$ cannot appear free in $M$ and so $M\{N/x\}$ is just $M$.
The third point can be proved by induction, too, but it is a bit more difficult. First of all, observe that $A$ must be in the form $\underbrace{!...!}_{n} C$, with $n \geq 1$. Let us focus on rules $E_{\multimap}$ and ! (the other ones can be handled easily). Notice that the derivation for $\Gamma_2 \mid \Delta \mid \Theta \vdash N : A$ must end with $A^L$, $A^P$, $I^L_{\multimap}$ or $I^I_{\multimap}$ (depending on the shape of $N$), followed by exactly $n$ instances of the ! rule, being this rule the only one not syntax directed. This implies, in particular, that every variable appearing free in $N$ is in $\Delta$. So the proof follows easily by induction. Using Lemma 3 and Lemma 2, we can use the induction hypothesis and handle the case !. □

**Theorem 1 (Call-by-value Subject Reduction).** *$\Gamma \mid \Delta \mid \Theta \vdash M : A$ and $M \to_v N$ implies $\Gamma \mid \Delta \mid \Theta \vdash N : A$.*

*Proof.* By the Substitution Lemma. □

We are now going to prove that the set of typable $\lambda$-terms is exactly $(\text{NEAL})^{\#}$. To do this we need the following lemma.

**Lemma 5 (Contraction Lemma).**
- *If $\Gamma \mid \Delta \mid x : A, y : A, \Theta \vdash M : B$, then $\Gamma \mid \Delta \mid z : A, \Theta \vdash M\{z/x, z/y\} : B$*
- *If $\Gamma \mid x : A, y : A, \Delta \mid \Theta \vdash M : B$, then $\Gamma \mid z : A, \Delta \mid \Theta \vdash M\{z/x, z/y\} : B$*

*Proof.* By induction. □

**Proposition 2.** *i) If $\Phi \vdash_{\text{NEAL}} M : A$ then $\Phi^L \mid \Phi^I \mid \emptyset \vdash (M)^{\#} : A$.*
*i) If $\Gamma \mid \Delta \mid \emptyset \vdash M : A$, there is $N \in \Lambda^{EA}$ such that $(N)^{\#} = M$ and $\Gamma, \Delta \vdash_{\text{NEAL}} N : A$.*

*Proof.* i) By induction on the structure of the derivation for $\Phi \vdash_{\text{NEAL}} M : A$. Let us focus on nontrivial cases.
If the last rule used is $E_{\multimap}$, the two premises are $\Phi \vdash_{\text{NEAL}} N : B \multimap C$ and $\Phi_2 \vdash_{\text{NEAL}} P : B$. By induction hypothesis, $\Phi_1^L \mid \Phi_1^I \mid \emptyset \vdash (N)^{\#} : B \multimap C$, and $\Phi_2^L \mid \Phi_2^I \mid \emptyset \vdash (P)^{\#} : B$ and, by weakening lemma, $\Phi_1^L \mid \Phi_1^I, \Phi_2^I \mid \emptyset \vdash (N)^{\#} : B \multimap C$, $\Phi_2^L \mid \Phi_1^I, \Phi_2^I \mid \emptyset \vdash (P)^{\#} : B$ Rule $E_{\multimap}$ leads to the thesis.
If the last rule used is *contr*, the two premises are $\Phi_1 \vdash_{\text{NEAL}} N : !A$ and $\Phi_2, x : !A, y : !A \vdash_{\text{NEAL}} P : B$. By induction hypothesis, $\Phi_1^L \mid \Phi_1^I \mid \emptyset \vdash (N)^{\#} : !A$, $\Phi_2^L \mid \Phi_2^I, x : !A, y : !A \mid \emptyset \vdash (P)^{\#} : B$. By contraction lemma, $\Phi_2^L \mid \Phi_2^I, z : !A \mid \emptyset \vdash (P)^{\#}\{z/x, z/y\} : B$ and so $\Phi_2^L \mid \Phi_2^I \mid \emptyset \vdash \lambda z.(P)^{\#}\{z/x, z/y\} : !A \multimap B$ By rule $E_{\multimap}$ and weakening lemma, we finally get $\Phi_1^L, \Phi_2^L \mid \Phi_1^L, \Phi_2^I \mid \emptyset \vdash (\lambda z.(P)^{\#}\{z/x, z/y\})(N)^{\#} : B$.
ii) The following, stronger, statement can be proved by induction on $\pi$: if $\pi : \Gamma \mid \Delta \mid x_1 : A_1, \ldots, x_n : A_n \vdash M : A$, then there is $N \in \Lambda^{EA}$ such that $(N)^{\#} = M\{x_1/y_1^1, \ldots, x_1/y_1^{m_1}, \ldots, x_n/y_n^1, \ldots, x_n/y_n^{m_n}\}$ and $\Gamma, \Delta, y_1^1 : A_1, \ldots, y_1^{m_1} : A_1, \ldots, y_n^1 : A_n, \ldots, y_n^{m_n} : A_n \vdash_{\text{NEAL}} N : A$. □

We have just established a deep *static* correspondence between NEAL and the class of typable lambda terms. But what about *dynamics*? Unfortunately, the two systems are not bisimilar. Nevertheless, every call-by-value $\beta$-step in the lambda calculus corresponds to at least one normalization step in $\Lambda^{EA}$. A normalization step in $\Lambda^{EA}$ is denoted by $\rightarrow$ (reduction rules can be found in the Appendix); $\rightarrow^+$ denotes the transitive closure of $\rightarrow$:

**Proposition 3.** *For every $M \in \Lambda^{EA}$, if $(M)^{\#} \rightarrow_v N$, then there is $L \in \Lambda^{EA}$ such that $(L)^{\#} = N$ and $M \rightarrow^+ L$.*

## 4 Bounds on Normalization Time

In order to prove elementary bounds on reduction sequences, we need to define a refined measure on lambda terms. We can look at a type derivation $\pi : \Gamma \mid \Delta \mid \Theta \vdash M : A$ as a labelled tree, where every node is labelled by a rule instance. We can give the following definition:

**Definition 4.** *Let $\pi : \Gamma \mid \Delta \mid \Theta \vdash M : A$.*

*i) A subderivation $\rho$ of $\pi$ has* level *$i$ if there are $i$ applications of the rule ! in the path from the root of $\rho$ to the root of $\pi$.*

*ii) An occurrence of a subterm $N$ of $M$ has level $i$ in $\pi$ if $i$ is the maximum level of a subderivation of $\pi$ having $N$ as subject.*

*iii) The level $\partial(\pi)$ of a whole derivation $\pi$ is the maximum level of subderivations of $\pi$.*

Notice that the so defined level corresponds to the notion of box-nesting depth in a proof-nets [3]. The size $|M|$ of a typable lambda term $M$ does not take into account levels as we have just defined them. The following definitions reconcile them, allowing $|M|$ to be "splitted" on different levels.

**Definition 5.** *Let $\pi : \Gamma \mid \Delta \mid \Theta \vdash M : A$.*

*i) $S(\pi, i)$ is defined by induction on $\pi$ as follows:*

- *If $\pi$ consists of an axiom, then $S(\pi, 0) = 1$ and $S(\pi, i) = 0$ for every $i \geq 1$;*
- *If the last rule in $\pi$ is $I^I_{\multimap}$ or $I^L_{\multimap}$, then $S(\pi, 0) = S(\rho, 0) + 1$ and $S(\pi, i) = S(\rho, i)$ for every $i \geq 1$, where $\rho$ is the immediate subderivation of $\pi$;*
- *If the last rule in $\pi$ is $E_{\multimap}$ then $S(\pi, 0) = S(\rho, 0) + S(\sigma, 0) + 1$ and $S(\pi, i) = S(\rho, i) + S(\sigma, i)$ for every $i \geq 1$, where $\rho$ and $\sigma$ are the immediate subderivations of $\pi$;*
- *If the last rule in $\pi$ is !, then $S(\pi, 0) = 0$ and $S(\pi, i) = S(\rho, i-1)$ for every $i \geq 1$, where $\rho$ is the immediate subderivation of $\pi$.*

*ii) Let $n$ be the level of $\pi$. The* size *of $\pi$ is $S(\pi) = \sum_{i \leq n} S(\pi, i)$. Observe that $S(\pi)$ is just $|M|$.*

Substitution Lemma can be restated in the following way:

**Lemma 6 (Substitution Lemma, revisited).**

*i) If* $\pi : \Gamma_1, x : A \mid \Delta \mid \Theta \vdash M : B$, $\rho : \Gamma_2 \mid \Delta \mid \Theta \vdash N : A$ *and* $N \in \mathcal{V}$, *then there is* $\sigma : \Gamma_1, \Gamma_2 \mid \Delta \mid \Theta \vdash M\{N/x\} : B$ *such that* $S(\sigma, i) \leq S(\rho, i) + S(\pi, i)$ *for every* $i$.
*ii) If* $\pi : \Gamma_1 \mid \Delta \mid x : A, \Theta \vdash M : B$, $\rho : \Gamma_2 \mid \Delta \mid \Theta \vdash N : A$ *and* $N \in \mathcal{V}$, *then there is* $\sigma : \Gamma_1 \mid \Delta \mid \Gamma_2, \Theta \vdash M\{N/x\} : B$ *such that* $S(\sigma, i) \leq S(\pi, i)S(\rho, 0) + S(\pi, i)$ *for every* $i$.
*iii) If* $\pi : \Gamma_1 \mid \Delta, x : A \mid \Theta \vdash M : B$, $\rho : \Gamma_2 \mid \Delta \mid \Theta \vdash N : A$ *and* $N \in \mathcal{V}$, *then there is* $\sigma : \Gamma_1, \Gamma_2 \mid \Delta \mid \Theta \vdash M\{N/x\} : B$ *such that* $S(\sigma, 0) \leq S(\pi, 0)$ *and* $S(\sigma, i) \leq (\sum_{j \leq i} S(\pi, j))S(\rho, i) + S(\pi, i)$ *for every* $i \geq 1$.

The following can be thought of as a strenghtening of subject reduction and is a corollary of Lemma 6.

**Proposition 4.** *If* $\pi : \Gamma \mid \Delta \mid \Theta \vdash M : A$, *and* $M \to_v N$ *by reducing a redex at level* $i$ *in* $\pi$, *then there is* $\rho : \Gamma \mid \Delta \mid \Theta \vdash N : A$ *such that*

$$\begin{aligned} \forall j < i. S(\rho, j) &= S(\pi, j) \\ S(\rho, i) &< S(\pi, i) \\ \forall j > i. S(\rho, j) &\leq S(\pi, j)(\sum_{k \leq j} S(\pi, k)) \end{aligned}$$

*In this case, we will write* $\pi \to_v^i \rho$.

*Proof.* Type derivation $\rho$ is identical to $\pi$ up to level $i$, so the equality $S(\rho, j) = S(\pi, j)$ holds for all levels $j < i$. At levels $j \geq i$, the only differences between $\rho$ and $\pi$ are due to the replacement of a type derivation $\phi$ for $(\lambda x.L)P$ with another type derivation $\psi$ for $L\{P/x\}$. $\psi$ is obtained by Lemma 6. The needed inequalitites follows from the ones in Lemma 6. □

Consider now a term $M$ and a derivation $\pi : \Gamma \mid \Delta \mid \Theta \vdash M : A$. By Proposition 4, every reduction sequence $M \to_v N \to_v L \to_v \ldots$ can be put in correspondence with a sequence $\pi \to_v^i \rho \to_v^j \sigma \to_v^k \ldots$ (where $\rho$ types $N$, $\sigma$ types $L$, etc.). The following result give bounds on the lengths of such sequences and on the possible growth during normalization.

**Proposition 5.** *For every* $d \in \mathbb{N}$, *there are elementary functions* $f_d, g_d : \mathbb{N} \to \mathbb{N}$ *such that, for every sequence*

$$\pi_0 \to_v^{i_0} \pi_1 \to_v^{i_1} \pi_2 \to_v^{i_2} \ldots$$

*it holds that*
- *For every* $i \in \mathbb{N}$, $\sum_{e \leq d} S(\pi_i, e) \leq f_d(S(\pi_0))$;
- *There are at most* $g_d(S(\pi_0))$ *reduction steps with indices* $e \leq d$.

**Theorem 2.** *For every* $d \in \mathbb{N}$ *there are elementary functions* $p_d, q_d : \mathbb{N} \to \mathbb{N}$ *such that whenever* $\pi : \Gamma \mid \Delta \mid \Theta \vdash M : A$, *the length of reduction sequences starting from* $M$ *is at most* $p_{\partial(\pi)}(|M|)$ *and the length of any reduct of* $M$ *is at most* $q_{\partial(\pi)}(|M|)$.

*Proof.* This is immediate from proposition 5. □

## 5 Type Inference

We prove, in a constructive way, that the type inference problem for the system $\vdash$ is decidable. Namely a type inference algorithm is designed, giving, for every $\lambda$-term $M$, a finite set of *principal typings*, from which all and only its typings can be obtained. If this set is empty, then $M$ is not typable. The design of the algorithm is based on the following Generation Lemma.

**Lemma 7 (Generation Lemma).** *Let $\Gamma \mid \Delta \mid \Theta \vdash M : A$.*

*i) Let $M \equiv x$. If $A$ is linear, then either $\{x : A\} \in \Gamma$ or $\{x : A\} \in \Theta$. Otherwise, $\{x : A\} \in \Delta$.*

*ii) Let $M \equiv \lambda x.N$. Then $A$ is of the shape $\underbrace{!...!}_{n}(B \multimap C)$ $(n \geq 0)$.*

*ii.1) Let $n = 0$. If $B$ is linear then $\Gamma, x : B \mid \Delta \mid \Theta \vdash N : C$, else $\Gamma \mid \Delta, x : B \mid \Theta \vdash N : C$.*

*ii.2) Let $n > 0$. Then $\emptyset \mid \Delta \mid \emptyset \vdash M : A$ and $\Gamma' \mid \Delta' \mid \Theta' \vdash N : C$, where $\Delta = \underbrace{!...!}_{n}((\Gamma' \cup \Delta' \cup \Theta') - \{x : B\})$.*

*iii) Let $M \equiv PQ$. Then $A$ is of the shape $\underbrace{!...!}_{n} B$; $(n \geq 0)$ and $\Gamma_1 \mid \Delta' \mid \Theta' \vdash P : C \multimap \underbrace{!...!}_{m} B$ and $\Gamma_2 \mid \Delta' \mid \Theta' \vdash Q : C$.*

*iii.1) If $n - m = 0$, then $\Gamma = \Gamma_1 \cup \Gamma_2$, $\Delta = \Delta'$ and $\Theta = \Theta'$.*

*iii.2) If $n - m > 0$, then $\emptyset \mid \Delta \mid \emptyset \vdash M : A$, where $\Delta = \underbrace{!...!}_{n-m}(\Gamma_1 \cup \Gamma_2 \cup \Delta' \cup \Theta')$*

The principal typings are described through the notion of a *type scheme*, which is a variation on the one used in [6] in the context of $\Lambda^{EA}$ and NEAL. Roughly speaking, a type scheme describes a class of types, i.e. it can be transformed into a type through a suitable notion of a substitution.

**Definition 6.** *i)* Functional type schemes *and* type schemes *are respectively defined by the following grammars:*

$$\mu ::= \alpha \mid \sigma \multimap \sigma$$
$$\sigma ::= \beta \mid \mu \mid !^p\sigma$$

*where the* exponential *$p$ is defined by the following grammar:*

$$p ::= n \mid p + p$$

*$\alpha$ and $\beta$ belong respectively to $NVar$ and $MVar$, two disjoint countable sets of scheme variables, and $n$ belongs to a countable set of literals. A generic scheme variable is ranged over by $\phi$, functional type schemes are ranged over by $\mu, \nu$, type schemes are ranged over by $\sigma, \tau, \rho$, and exponentials are ranged over by $p, q, r$. Let $\mathcal{T}$ denote the set of type schemes. A type scheme of the shape $!^p\sigma$ is called* modal.

*ii)* A scheme substitution $S$ *is a function from type schemes to types, replacing scheme variables in* $NVar$ *by linear types, scheme variables in* $MVar$ *by types and literals by natural numbers greater than 0. The application of* $S$ *to a type scheme is defined inductively as follows:*

$$S(\phi) = A \text{ if } [\phi \mapsto A] \in S;$$

$$S(\sigma \multimap \tau) = S(\sigma) \multimap S(\tau);$$

$$S(!^{n_1+\ldots+n_i}\sigma) = \underbrace{!\ldots!}_{q} S(\sigma),$$

*where* $q = S(n_1) + \ldots + S(n_i)$.

*iii)* A scheme basis *is a finite subset of type scheme associations to variables. With an abuse of notation, we will let* $\Xi, \Gamma, \Delta, \Theta$ *range over scheme basis, with the constraint that* $\Gamma$ *and* $\Theta$ *denote scheme basis associating non-modal schemes to variables, while* $\Delta$ *denotes a scheme basis associating modal type schemes to variables.*

$\equiv$ is extended to denote the syntactical identity between both types and type schemes.

In order to define the principal typing, we will use a unification algorithm for type schemes, which is a variant of that defined in [6]. Let $=_e$ be the relation between type schemes defined as follows: $\phi =_e \phi$; $\sigma =_e \sigma'$ and $\tau =_e \tau'$ imply $\sigma \multimap \tau =_e \sigma' \multimap \tau'$; $\sigma =_e \tau$ implies $!^p\sigma =_e !^q\tau$. Roughly speaking, two type schemes are in $=_e$ if and only if they are identical modulo the exponentials.

The unification algorithm, which we will present in SOS style in Table 3, is a function $U$ from $\mathcal{T} \times \mathcal{T}$ to pairs of the shape $\langle C, s\rangle$, where $C$ (the *modality* set) is a set of natural linear constraints, in the form $p = q$, where $p$ and $q$ are exponentials, and $s$ is a substitution, replacing scheme variables by type schemes. A set $C$ of linear constraints is *solvable* if there is a scheme substitution $S$ such that, for every constraint $n_1 + \ldots + n_i = m_1 + \ldots + m_j$ in $C$, $S(n_1) + \ldots + S(n_i) = S(m_1) + \ldots + S(m_j)$. Clearly the solvability of a set of linear constraints is a decidable problem.

The following two technical lemmas prove that, if $U(\sigma, \tau) = \langle C, s\rangle$, then this result supplies a more general unifiers for type schemes (modulo $=_e$ ) and moreover this can be extended to types.

**Lemma 8.** *i) (correctness)* $U(\sigma, \tau) = \langle C, s\rangle$ *implies* $s(\sigma) =_e s(\tau)$.

*ii) (completeness)* $s(\sigma) =_e s(\tau)$ *implies* $U(\sigma, \tau) = \langle C, s'\rangle$ *and* $s = s' \circ s''$, *for some* $s''$.

The extension to types must take into consideration the set of linear constraints generated by $U$, which imposes some relations between the number of modalities in different subtypes of the same type.

**Lemma 9.** *i) (correctness) Let* $U(\sigma, \tau) = \langle C, s\rangle$. *Then, for every scheme substitution* $S$, *such that* $S$ *is a solution of* $C$, $S(s(\sigma)) \equiv S(s(\tau))$.

$$\frac{}{U(\phi,\phi)=\langle\emptyset,[]\rangle}\ (U1)\quad \frac{\alpha \text{ does not occur in } \mu}{U(\alpha,\mu)=\langle\emptyset,[\alpha\mapsto\mu]\rangle}\ (U2)$$

$$\frac{\alpha \text{ does not occur in } \mu}{U(\mu,\alpha)=\langle\emptyset,[\alpha\mapsto\mu]\rangle}\ (U3)\quad \frac{\beta \text{ does not occur in } \sigma}{U(\beta,\sigma)=\langle\emptyset,[\beta\mapsto\sigma]\rangle}\ (U4)$$

$$\frac{\beta \text{ does not occur in } \sigma}{U(\sigma,\beta)=\langle\emptyset,[\beta\mapsto\sigma]\rangle}\ (U5)$$

$$\frac{U(\mu,\nu)=\langle C,s\rangle}{U(!^{p_1}...!^{p_n}\mu,!^{q_1}...!^{q_m}\nu)=\langle C\cup\{p_1+...+p_n=q_1+...+q_m\},s\rangle}\ (U6)$$

$$\frac{U(\sigma_1,\tau_1)=\langle C_1,s_1\rangle\quad U(s_1(\sigma_2),s_1(\tau_2))=\langle C_2,s_2\rangle}{U(\sigma_1\multimap\sigma_2,\tau_1\multimap\tau_2)=\langle C_1\cup C_2,s_1\circ s_2\rangle}\ (U7)$$

In all other cases, $U$ is undefined: for example both $U(\alpha,\alpha\multimap\beta)$ and $U(!^p\alpha,\sigma\multimap\tau)$ are undefined. $s_1\circ s_2$ is the substitution such that $s_1\circ s_2(\sigma)=s_2(s_1(\sigma))$.

**Table 3.** The unification algorithm U

*ii) (completeness) $S(\sigma)\equiv S(\tau)$ implies $U(\sigma,\tau)=\langle C,s\rangle$, and $S(\sigma)\equiv S'(s(\sigma))$, $S(\tau)\equiv S'(s(\tau))$ , for some $S'$ satisfying $C$.*

The set of principal type schemes of a term is a set of 5-tuples $<\Gamma;\Delta;\Theta;\sigma;C>$, where $\Gamma;\Delta;\Theta$ are scheme basis, $\sigma$ is a type scheme and $C$ is a set of constraints. It is defined in Table 4.

**Theorem 3 (Type Inference).**

*i) (correctness) $<\Gamma;\Delta;\Theta;\sigma;C>\in PT(M)$ implies that, for all scheme substitution $S$, for all substitution $s$, $S(s(\Gamma))\mid S(s(\Delta))\mid S(s(\Theta))\vdash M:S(s(\sigma))$.*
*ii) (completeness) $\Gamma\mid\Delta\mid\Theta\vdash M:A$ implies there is $<\Gamma';\Delta';\Theta';\sigma;C>\in PT(M)$ such that $A=S(\sigma)$ and $S(\Xi')\subseteq\Xi$ ($\Xi\in\{\Gamma,\Delta,\Theta\}$ ).*

*Proof.* i) By induction on $M$.
ii) By induction on the derivation proving $\Gamma\mid\Delta\mid\Theta\vdash M:A$, using the Generation Lemma.

□

## 6 Extensions

### 6.1 Achieving Completeness

The type-system we introduced in this paper is not complete for the class of elementary time functions, at least if we restrict to uniform encodings. One

| |
|---|
| $PT(x) = \{< x : \alpha; \emptyset; \emptyset; \alpha; \epsilon >, < \emptyset; \emptyset; x : \alpha; \alpha; \epsilon >, < \emptyset; x :!^n \alpha; \emptyset; !^n \alpha; \epsilon >\}$ |
| $PT(\lambda x.M) = \{< \Gamma; \Delta; \Theta; \sigma \multimap \tau; C >, < \emptyset; !^m(\Gamma \cup \Delta \cup \Theta); \emptyset; !^m \sigma \multimap \tau; C >\|$<br>$< \Gamma, x : \sigma; \Delta; \Theta; \tau; C > \in PT(M)$ `or`<br>$(< \Gamma; \Delta, x : \sigma; \Theta; \tau; C > \in PT(M))$ `or`<br>$(< \Gamma; \Delta; \Theta; \tau; C > \in PT(M)$ `and`<br>$\sigma \equiv \alpha$ `and` $x \notin dom(\Gamma) \cup dom(\Delta) \cup dom(\Theta))$ $\}$ |
| $PT(MN) = \{< \Gamma; \Delta; \Theta; \sigma; C >, < \emptyset; !^m(\Gamma \cup \Delta \cup \Theta); \emptyset; !^m \sigma; C >\|$<br>$< \Gamma_1; \Delta_1; \Theta_1; \sigma_1; C_1 > \in PT(M)$ `and`<br>$< \Gamma_2; \Delta_2; \Theta_2; \sigma_2; C_2 > \in PT(N)$ `(disjoint) and`<br>$dom(\Gamma_1) \cap dom(\Gamma_2) = \emptyset$ `and`<br>$i \neq j$ `implies` $dom(\Xi_i) \cap dom(\Xi_j) = \emptyset$ `and`<br>$U(\sigma_1, \sigma_2 \multimap \beta) =< s, C' >$ `(`$\beta$ `fresh) and`<br>$\sigma = s \circ s_1 \circ ... \circ s_k(\beta)$<br>`where`<br>$U(s(\tau_1^1), s(\tau_2^1)) =< s_1, C'_{1>}$ `and`<br>$U(s_i(\tau_1^i), s_i(\tau_2^i)) =< s_{i+1}, C'_{i+1} >$ `and`<br>$x_i \in dom(\Xi_1) \cup dom(\Xi_2)$, $x_i : \tau_j^i \in \Xi_j$ `and`<br>$\Xi = \{x : s \circ s_1 \circ ... \circ s_k(\tau) \mid x : \tau \in \Xi_j\}$ `and`<br>$C = C' \bigcup_{1 \leq i \leq k} C'_i$<br>$(1 \leq i \leq k,\ 1 \leq j \leq 2,\ \Xi \in \{\Gamma, \Delta, \Theta\}\ )\}$ |

**Table 4.** The Type Inference Algorithm. (two 5-tuples are disjoint if and only if they are built from different scheme variables)

possible solution could consist in extending the type system with second order quantification. This, however, would make type inference much harder (if not undecidable). Another approach, consists in extending the language with basic types and constants. In this section, we will sketch one possible extension going exactly in this direction.
Suppose we fix a finite set of free algebras $\mathcal{A} = \{\mathbb{A}_1, \ldots, \mathbb{A}_n\}$. The constructors of $\mathbb{A}_i$ will be denoted as $c^1_{\mathbb{A}_1}, \ldots, c^{k(\mathbb{A}_i)}_{\mathbb{A}_1}$. The arity of constructor $c^j_{\mathbb{A}_i}$ will be denoted as $\mathcal{R}^j_{\mathbb{A}_i}$. The algebra $\mathbb{U}$ of unary integers has two constructors $c^1_{\mathbb{U}}, c^2_{\mathbb{U}}$, where $\mathcal{R}^1_{\mathbb{U}} = 1$ and $\mathcal{R}^1_{\mathbb{U}} = 0$.
The language of types will be extended by the production $A ::= \mathbb{A}$, while the space of terms will be extended by:

$$M ::= iter_{\mathbb{A}} \mid cond_{\mathbb{A}} \mid c^i_{\mathbb{A}}$$

where $\mathbb{A}$ ranges over $\mathscr{A}$ and $c_{\mathbb{A}}$ ranges over constructors for free algebra $\mathbb{A}$. The set of values is extended as follows:

$$V ::= iter_{\mathbb{A}} \mid cond_{\mathbb{A}} \mid c^i_{\mathbb{A}} \mid t$$

where $t$ ranges over free algebra terms. The new constants receive the following types in any context:

$$iter_{\mathbb{A}} : \mathbb{A} \multimap !(\underbrace{A \multimap \ldots \multimap A}_{\mathcal{R}^1_{\mathbb{A}})\ times} \multimap A) \multimap \ldots \multimap !(\underbrace{A \multimap \ldots \multimap A}_{\mathcal{R}^{k(\mathbb{A})}_{\mathbb{A}}\ times} \multimap A) \multimap !A$$

$$cond_{\mathbb{A}} : \mathbb{A} \multimap (\underbrace{\mathbb{A} \multimap \ldots \multimap \mathbb{A}}_{\mathcal{R}^1_{\mathbb{A}})\ times} \multimap A) \multimap \ldots \multimap (\underbrace{\mathbb{A} \multimap \ldots \multimap \mathbb{A}}_{\mathcal{R}^{k(\mathbb{A})}_{\mathbb{A}})\ times} \multimap A) \multimap A$$

$$c^i_{\mathbb{A}} : \underbrace{\mathbb{A} \multimap \ldots \multimap \mathbb{A}}_{\mathcal{R}^i_{\mathbb{A}})\ times} \multimap \mathbb{A}$$

New (call by value) reduction rules are the following ones:

$$iter_{\mathbb{A}} t V_1 \ldots V_{k(\mathbb{A})} \to_v t\{V_1 \ldots V_{k(\mathbb{A})}\}$$

$$cond_{\mathbb{A}} c^i_{\mathbb{A}}(t_1 \ldots t_{\mathcal{R}^i_{\mathbb{A}}}) V_1 \ldots V_{k(\mathbb{A})} \to_v V_i t_1 \ldots t_{\mathcal{R}^i_{\mathbb{A}}}$$

It is easy to check that proposition 6 continue to be true in the presence of the new constants. Moreover, we can prove the following theorem

**Theorem 4.** *There is a finite set of free algebra $\mathscr{A}$ including the algebra $\mathbb{U}$ of unary integers such that for every elementary function $f : \mathbb{N} \to \mathbb{N}$, there is a term $M_f : \mathbb{U} \to !^k\mathbb{U}$ such that $M_f\lceil u \rceil \to^*_v \lceil f(u) \rceil$ (where $\lceil n \rceil$ is the term in $\mathbb{U}$ corresponding to the natural number $n$).*

### 6.2 Adapting the System to other Logics

We believe the approach described in this paper to be applicable to other logics besides elementary affine logic. Two examples are Light Affine Logic [3] and Soft

Affine Logic [10]. Light affine logic needs two modalities. So, there will be two rules:

$$\frac{\Gamma_1 \mid \Delta_1 \mid \Theta_1 \vdash M : A \quad |\Gamma_1| + |\Delta_1| + |\Theta_1| \leq 1}{\Gamma_2 \mid !\Gamma_1, !\Delta_1, !\Theta_1, \Delta_2 \mid \Theta_2 \vdash M :!A} \; !$$

$$\frac{\Gamma_1, \Gamma_2 \mid \Delta_1, \Delta_2 \mid \Theta_1 \vdash M : A}{\S\Gamma_1, \S\Delta_1, \S\Theta_1, \Gamma_3 \mid !\Gamma_2, !\Delta_2, !\Theta_2, \Delta_3 \mid \Theta_2 \vdash M : \S A} \; \S$$

Soft affine logic is even simpler that elementary affine logic: there is just one modality and the context is splitted in just two sub-context. The ! rule becomes:

$$\frac{\Gamma \mid \Delta \vdash M : A}{!\Gamma \mid !\Delta \vdash M :!A} \; !$$

Multiplexing (the equivalent of contraction) can only be made contextually to arrow-introduction:

$$\frac{\Gamma \mid x : A, \Delta \vdash M : B}{\Gamma \mid \Delta \vdash \lambda x.M :!A \multimap B} \; A^I$$

## References


1. Baillot, P., Terui, K.: Light types for polynomial time computation in lambda-calculus. In: Proceedings of the 19th IEEE Syposium on Logic in Computer Science. (2004) 266–275
2. Girard, J.Y.: Light linear logic. Information and Computation **143(2)** (1998) 175–204
3. Asperti, A.: Light affine logic. In: Proceedings of the 13th IEEE Syposium on Logic in Computer Science. (1998) 300–308
4. Asperti, A., Roversi, L.: Intuitionistic light affine logic. ACM Transactions on Computational Logic **3(1)** (2002) 137–175
5. Coppola, P., Martini, S.: Typing lambda terms in elementary logic with linear constraints. In: Proceedings of the 6th International Conference on Typed Lambda-Calculus and Applications. (2001) 76–90
6. Coppola, P., Ronchi della Rocca, S.: Principal typing in elementary affine logic. In: Proceedings of the 7th International Conference on Typed Lambda-Calculus and Applications. (2003) 90–104
7. Terui, K.: Light affine lambda calculus and polytime strong normalization. In: 16th Annual IEEE Symposium on Logic in Computer Science. (2001) 209–220
8. Terui, K.: Light logic and polynomial time computation. PhD thesis, Keio University (2002)
9. Baillot, P.: Checking polynomial time complexity with types. In: Proceedings of the 2nd IFIP International Conference on Theoretical Computer Science. (2002)
10. Baillot, P., Mogbil, V.: Soft lambda-calculus: a language for polynomial time computation. In: Proceedings of the 7th International Conference on Foundations of Software Science and Computational Structures. (2004)

## Appendix

$$(\lambda x.M\ N) \qquad \to_{\beta} \qquad M\{N/x\}$$

$$[N]_{!(M)\left[{}^{M_1}/_{x_1},\ldots,{}^{M_n}/_{x_n}\right]=x,y} \qquad \to_{\mathsf{dup}}$$
$$[\ldots[N\{{}^{!(M)\left[{}^{x'_1}/_{x_1},\ldots,{}^{x'_n}/_{x_n}\right]}/x\}\{{}^{!(M')\left[{}^{y'_1}/_{y_1},\ldots,{}^{y'_n}/_{y_n}\right]}/y\}]_{M_1=x'_1,y'_1}\cdots]_{M_n=x'_n,y'_n}$$

$$!(M)[{}^{M_1}/_{x_1},\cdots,{}^{!(N)\left[{}^{P_1}/_{y_1},\ldots,{}^{P_m}/_{y_m}\right]}/_{x_i},\cdots,{}^{M_n}/_{x_n}] \qquad \to_{!-!}$$
$$!(M\{N/x_i\})[{}^{M_1}/_{x_1},\cdots,{}^{P_1}/_{y_1},\cdots,{}^{P_m}/_{y_m},\cdots{}^{M_n}/_{x_n}]$$

$$([M]_{M_1=x_1,x_2}\ N) \qquad \to_{@-\mathsf{c}} \qquad [(M\{x'_1/x_1,x'_2/x_2\}\ N)]_{M_1=x'_1,x'_2}$$

$$(M\ [N]_{N_1=x_1,x_2}) \qquad \to_{@-\mathsf{c}} \qquad [(M\ N\{x'_1/x_1,x'_2/x_2\})]_{N_1=x'_1,x'_2}$$

$$!(M)[{}^{M_1}/_{x_1},\cdots,{}^{[M_i]_{N=y,z}}/_{x_i},\cdots,{}^{M_n}/_{x_n}] \qquad \to_{!-c}$$
$$[!(M)[{}^{M_1}/_{x_1},\cdots,{}^{M_i\{y'/y,z'/z\}}/_{x_i},\cdots,{}^{M_n}/_{x_n}]]_{N=y',z'}$$

$$[M]_{[N]_{P=y_1,y_2}=x_1,x_2} \qquad \to_{\mathsf{c}-\mathsf{c}} \qquad [[M]_{N\{y'_1/y_1,y'_2/y_2\}=x_1,x_2}]_{P=y'_1,y'_2}$$

$$\lambda x.[M]_{N=y,z} \qquad \to_{\lambda-\mathsf{c}} \qquad [\lambda x.M]_{N=y,z} \text{ where } x \notin \mathtt{FV}(N)$$

where $M'$ in the $\to_{\mathsf{dup}}$-rule is obtained from $M$ replacing all its free variables with fresh ones ($x_i$ is replaced with $y_i$); $x'_1$ and $x'_2$ in the $\to_{@-\mathsf{c}}$-rule, $y'$ and $z'$ in the $\to_{!-c}$-rule and $y'_1, y'_2$ in the $\to_{\mathsf{c}-\mathsf{c}}$-rule are fresh variables.

**Table 5.** Normalization rules in $\Lambda^{EA}$.

**Proposition 1** *For every $N \in \Lambda^{EA}$, $L(N^{\#}) \leq 2L(N)$.*

*Proof.* By induction on $N$. The cases for variables, abstractions and applications are trivial. Let us now consider the other two inductive cases. Suppose $N = [P]_{Q=x,y}$. If $Q$ is a variable, then $L(N^{\#}) = L(P^{\#}) \leq 2L(P) \leq 2L(N)$. If $Q$ is not a variable, then $L(N^{\#}) = L(P^{\#}) + L(Q^{\#}) + 2 \leq 2L(P) + 2L(Q) + 2 = 2(L(P) + L(Q) + 1) = 2L(N)$. If, on the other hand, $N = !\,(M)\left[{}^{M_1}/_{x_1},\ldots,{}^{M_n}/_{x_n}\right]$, then we can proceed by induction on $n$. If $n = 0$, then the inequality is trivially verified. If, on the other hand, $n > 0$, then we must distinguish two different cases: if $M_n$ is a variable, then the inequality is trivially satified; if $M_n$ is not a variable, then $N^{\#}$ is $(\lambda x_n.(!\,(M)\left[{}^{M_1}/_{x_1},\ldots,{}^{M_{n-1}}/_{x_{n-1}}\right])^{\#}))M_n^{\#}$ and, by the induction hypothesis on $n$ and $M_n$, we get

$$\begin{aligned} L(N^{\#}) &= 2 + L((!\,(M)\left[{}^{M_1}/_{x_1},\ldots,{}^{M_{n-1}}/_{x_{n-1}}\right])^{\#}) + L(M_n^{\#}) \\ &\leq 2 + 2L(!\,(M)\left[{}^{M_1}/_{x_1},\ldots,{}^{M_{n-1}}/_{x_{n-1}}\right]) + 2L(M_n^{\#}) \\ &= 2L(N) \end{aligned}$$

□

An *expansion* is a term in $\Lambda^{EA}$ that can be written either as $!\,(M)\,[^{x_1}/y_1,\ldots,^{x_n}/y_n]$ or as $[N]_{z=x,y}$, where $N$ is itself an expansion.

**Lemma 10.** *If $M$ is an expansion, then*
- $[L]_{M=x,y} \to^* P$, *where* $P^{\#} \equiv L\{M^{\#}/x, M^{\#}/y\}$;
- *If* $M \equiv P_i$, *then* $!\,(L)\,[^{x_1}/P_1,\ldots,^{x_n}/P_n] \to^* Q$ *where*

$$Q^{\#} \equiv (!\,(L)\,[^{x_1}/P_1,\ldots,^{x_{i-1}}/P_{i-1},{}^{x_{i+1}}/P_{i+1},\ldots,^{x_n}/P_n])^{\#}\{M^{\#}/x_i\}.$$

**Proposition 3** *For every $M \in \Lambda^{EA}$, if $(M)^{\#} \to_v^* N$, then there is $L \in \Lambda^{EA}$ such that $(L)^{\#} = N$ and $M \to^* L$.*

*Proof.* We can proceed by induction on the structure of $M$. If $M$ is a variable, then $M^{\#}$ is a variable, too, and so the premise is false. If $M$ is an abstraction, then the thesis follows from the inductive hypothesis. If $M$ is an application $P\,Q$, then we can assume $P$ to be an abstraction $\lambda x.R$ and $N$ to be $R^{\#}\{Q^{\#}/x\}$ (in all the other cases the thesis easily follows). It is easy to see that $R^{\#}\{Q^{\#}/x\} \equiv (R\{Q/x\})^{\#}$ and so we can take $R\{Q/x\}$ as our $L$. If $M$ is $[P]_{Q=x,y}$ and $Q$ is not a variable (otherwise the thesis easily follows), then $M^{\#} = (\lambda z.P^{\#}\{z/x, z/y\})Q^{\#}$ and we can restrict to the case where $N$ is $P^{\#}\{Q^{\#}/x, Q^{\#}/y\}$. First of all, we can observe that $Q^{\#}$ must be an abstraction. This means that $Q$ is an abstraction itself enclosed in one or more $!\,(\cdot)\,[^{x_1}/y_1,\ldots,^{x_n}/y_n]$ contexts and zero or more $[\cdot]_{z=x,y}$. This means $Q$ is an expansion and so, by lemma 10, we know there must be a term $R$ such that $R\# \equiv P^{\#}\{Q^{\#}/x, Q^{\#}/y\}$, and $M \to^* R$, that is the thesis. $!\,(P)\,[^{x_1}/Q_1,\ldots,^{x_n}/Q_n]$ can be managed in a similar way. □

**Lemma 11 (Weakening Lemma, revisited).** *If $\pi : \Gamma_1 \mid \Delta_1 \mid \Theta_1 \vdash M : A$, then there is $\rho : \Gamma_1, \Gamma_2 \mid \Delta_1, \Delta_2 \mid \Theta_1, \Theta_2 \vdash M : A$. such that $S(\pi, i) = S(\rho, i)$ for every $i$.*

**Lemma 12 (Shifting Lemma,revisited).** *If $\pi : \Gamma, x : A \mid \Delta \mid \Theta \vdash M : B$, then there is $\rho : \Gamma \mid \Delta \mid x : A, \Theta \vdash M : B$ such that $S(\pi, i) = S(\rho, i)$ for every $i$.*

**Lemma 6** *i) If $\pi : \Gamma_1, x : A \mid \Delta \mid \Theta \vdash M : B$, $\rho : \Gamma_2 \mid \Delta \mid \Theta \vdash N : A$ and $N \in \mathcal{V}$, then there is $\sigma : \Gamma_1, \Gamma_2 \mid \Delta \mid \Theta \vdash M\{N/x\} : B$ such that $S(\sigma, i) \leq S(\rho, i) + S(\pi, i)$ for every $i$.*

*ii) If $\pi : \Gamma_1 \mid \Delta \mid x : A, \Theta \vdash M : B$, $\rho : \Gamma_2 \mid \Delta \mid \Theta \vdash N : A$ and $N \in \mathcal{V}$, then there is $\sigma : \Gamma_1 \mid \Delta \mid \Gamma_2, \Theta \vdash M\{N/x\} : B$ such that $S(\sigma, i)S(\rho, 0) + S(\pi, i)$ for every $i$.*

*iii) If $\pi : \Gamma_1 \mid \Delta, x : A \mid \Theta \vdash M : B$, $\rho : \Gamma_2 \mid \Delta \mid \Theta \vdash N : A$ and $N \in \mathcal{V}$, then there is $\sigma : \Gamma_1, \Gamma_2 \mid \Delta \mid \Theta \vdash M\{N/x\} : B$ such that $S(\sigma, 0) \leq S(\pi, 0)$ and $S(\pi, i) \leq (\sum_{j \leq i} S(\pi, j))S(\rho, i) + S(\pi, i)$ for every $i \geq 1$.*

*Proof.* For each of the three claims, we can go by induction on the structure of $\pi$. Here, we do not concentrate on proving the existence of $\sigma$ (it follows from lemma 4) but on proving that $\sigma$ satisfy the given bounds. In particular,

we implicitly used lemmas 11 and 12 without explicitly citing them. Let us first analyze the claim i). We will prove that $S(\sigma, i) \leq S(\rho, i) \min\{1, S(\pi, 0)\} + S(\pi, i)\}$ for every $i$ by induction on $\pi$. If $\pi$ is just an axiom, then $\sigma$ is obtained by $\pi$ or $\rho$ by the weakening lemma and the bound holds. If the last rule in $\pi$ is $I^L_{-\circ}$ ($I^I_{-\circ}$), then $\rho$ is obtained by using the inductive hypothesis on the immediate premise $\phi$ of $\pi$ obtaining $\psi$ and applying $I^L_{-\circ}$ ($I^I_{-\circ}$) to $\psi$. In both cases

$$\begin{aligned} S(\sigma, 0) = S(\psi, 0) + 1 &\leq S(\rho, 0) \min\{1, S(\phi, 0)\} + S(\phi, 0) + 1 \\ &\leq S(\rho, 0) \min\{1, S(\pi, 0\} + S(\pi, 0) \\ \forall i \geq 1. S(\sigma, i) = S(\psi, i) &\leq S(\rho, i) \min\{1, S(\phi, i)\} + S(\phi, i) \\ &\leq S(\rho, 0) \min\{1, S(\pi, i)\} + S(\pi, i) \end{aligned}$$

If the last rule in $\pi$ is $E_{-\circ}$, then $\sigma$ is obtained by using the inductive hypothesis on one of the immediate premises $\phi$ of $\pi$ obtaining $\psi$, applying $E_{-\circ}$ to $\psi$ and the other premise $\xi$ of $\pi$. We have:

$$\begin{aligned} S(\sigma, 0) &= S(\psi, 0) + S(\xi, 0) + 1 \\ &\leq S(\rho, 0) \min\{1, S(\phi, 0)\} + S(\phi, 0) + S(\xi, 0) + 1 \\ &\leq S(\rho, 0) \min\{1, S(\pi, 0)\} + S(\pi, 0) \\ \forall i \geq 1. S(\sigma, i) &= S(\psi, i) + S(\xi, i) + 1 \\ &\leq S(\rho, i) \min\{1, S(\phi, i)\} + S(\phi, i) + S(\xi, i) \\ &\leq S(\rho, i) \min\{1, S(\pi, i)\} + S(\pi, i) \end{aligned}$$

If the last rule in $\pi$ is !, then $\sigma$ is just obtained from $\pi$ by weakening lemma, cause $x$ cannot appear free in $M$. The inequality easily follows.

Let us now prove point ii). If $\pi$ is just an axiom, we can proceed as previously. If the last rule in $\pi$ is $I^L_{-\circ}$ ($I^I_{-\circ}$), then $\rho$ is obtained as in point i) and, in both cases:

$$\begin{aligned} S(\sigma, 0) = S(\psi, 0) + 1 &\leq S(\rho, 0) S(\phi, 0) + S(\phi, 0) + 1 \\ &\leq S(\rho, 0) S(\pi, 0) + S(\pi, 0) \\ \forall i \geq 1. S(\sigma, i) = S(\psi, i) &\leq S(\rho, i) S(\phi, i) + S(\phi, i) \\ &\leq S(\rho, 0) S(\pi, i) + S(\pi, i) \end{aligned}$$

If the last rule in $\pi$ is $E_{-\circ}$, then $\sigma$ is obtained by using the inductive hypothesis on both the immediate premises $\phi$ and $\psi$ of $\pi$ obtaining $\xi$ and $\chi$ and applying

$E_{-\circ}$ to them. We obtain:

$$\begin{aligned}
S(\sigma,0) &= S(\xi,0)+S(\chi,0)+1\\
&\leq (S(\rho,0)S(\phi,0)+S(\phi,0))+(S(\rho,0)S(\psi,0)+S(\psi,0))+1\\
&\leq S(\rho,0)(S(\phi,0)+S(\psi,0)+1)+(S(\phi,0)+S(\psi,0)+1)\\
&= S(\rho,0)S(\pi,0)+S(\pi,0)\\
\forall i\geq 1.S(\sigma,i) &= S(\xi,i)+S(\chi,i)\\
&\leq (S(\rho,i)S(\phi,i)+S(\phi,i))+(S(\rho,i)S(\psi,i)+S(\psi,i))\\
&= S(\rho,i)(S(\phi,i)+S(\psi,i))+(S(\phi,i)+S(\psi,i))\\
&= S(\rho,0)S(\pi,0)+S(\pi,0)
\end{aligned}$$

If the last rule in $\pi$ is !, the $\sigma$ is again obtained by $\pi$ and the inequality follows. Let us now prove claim iii). Notice that the last rule in $\rho$ must be ! rule, cause $A$ is modal and $N$ is a value. If the last rule in $\pi$ is $I^L_{-\circ}$ ($I^I_{-\circ}$), then $\sigma$ is obtained in the usual way and:

$$\begin{aligned}
S(\sigma,0) &= S(\psi,0)+1\leq S(\phi,0)+1 = S(\pi,0)\\
\forall i\geq 1.S(\sigma,i) &= S(\psi,i)\leq S(\rho,i)(\sum_{j\leq i}S(\phi,j))+S(\phi,i)\\
&= S(\phi,i)(\sum_{j\leq i}S(\pi,j))+S(\pi,i)\ \ \forall i\geq 1
\end{aligned}$$

If the last rule in $\pi$ is $E_{-\circ}$, then $\sigma$ is obtained as in in point ii). Now we have:

$$\begin{aligned}
S(\sigma,0) &= S(\xi,0)+S(\chi,0)+1\leq S(\phi,0)+S(\psi,0)+1 = S(\pi,0)\\
\forall i\geq 1.S(\sigma,i) &= S(\xi,i)+S(\chi,i)\\
&\leq S(\rho,i)(\sum_{j\leq i}S(\phi,j))+S(\phi,i)+S(\rho,i)(\sum_{j\leq i}S(\psi,j))+S(\psi,i)\\
&= S(\rho,i)(\sum_{j\leq i}(S(\phi,j)+S(\psi,j)))+S(\phi,i)+S(\psi,i)\\
&= S(\rho,i)(\sum_{j\leq i}S(\pi,j))+S(\pi,i)
\end{aligned}$$

If the last rule in $\pi$ is !, then we can suppose the last rule in $\rho$ to be a ! and let $\psi$ be the immediate premise of $\rho$. We first apply the induction hypothesis (or one of the other two claims) to the immediate premise $\phi$ of $\pi$ and to $\psi$ obtaining $\xi$; then, we apply rule ! to $\xi$ and we get $\sigma$. Clearly, $S(\sigma,0)=0$ by definition. For every $i\geq 0$, we have that

$$S(\xi,i)\leq (\sum_{j\leq i}S(\phi,j))S(\psi,i)+S(\phi,i)$$

independently on the way we get $\xi$. As a consequence, for every $i \geq 1$,

$$\begin{aligned} S(\sigma, i) = S(\xi, i-1) &\leq (\sum_{j \leq i-1} S(\phi, j))S(\psi, i-1) + S(\phi, i-1) \\ &\leq (\sum_{j \leq i} S(\pi, j))S(\rho, i) + S(\pi, i) \end{aligned}$$

This concludes the proof. □

**Proposition 5** *For every $d \in \mathbb{N}$, there are elementary functions $f_d, g_d : \mathbb{N} \to \mathbb{N}$ such that, for every sequence*

$$\pi_0 \to_v^{i_0} \pi_1 \to_v^{i_1} \pi_2 \to_v^{i_2} \dots$$

*it holds that*

- *For every $i \in \mathbb{N}$, $\sum_{e \leq d} S(\pi_i, e) \leq f_d(S(\pi_0))$.*
- *There are at most $g_d(S(\pi_0))$ reduction steps with indices $e \leq d$.*

*Proof.* We go by induction on $d$ and define $f_d$ and $g_d$ such that the given inequalities hold and, additionally, $f_d(n) \geq n$ for each $n \in \mathbb{N}$. $f_0$ and $g_0$ are both the identity on natural numbers, because $S(\pi_0, 0)$ can only decrease during reduction and it can do that at most $S(\pi_0, 0)$ times. Consider now $d \geq 1$. Each time $S(\pi_i, d)$ grows, its value goes from $m$ to at most $m(m + f_{d-1}(S(\pi_0)))$. We claim that after having increased $n$ times, $S(\pi_i, d)$ is at most $(f_{d-1}(S(\pi_0)) + n)^{2^{n+1}}$. Indeed, initially

$$S(\pi_i, d) \leq S(\pi_0, d) \leq S(\pi_0) \leq (f_{d-1}(S(\pi_0)))^2$$

And, after $n \geq 1$ increases,

$$\begin{aligned} S(\pi_i, d) &\leq (f_{d-1}(S(\pi_0)) + n - 1)^{2^n}((f_{d-1}(S(\pi_0)) + n - 1)^{2^n} + f_{d-1}(S(\pi_0))) \\ &\leq (f_{d-1}(S(\pi_0)) + n)^{2^n}((f_{d-1}(S(\pi_0)) + n - 1)^{2^n} + (f_{d-1}(S(\pi_0)) + n - 1)^{2^{n-1}}) \\ &\leq (f_{d-1}(S(\pi_0)) + n)^{2^n}((f_{d-1}(S(\pi_0)) + n - 1 + 1)^{2^{n-1}})^2 \\ &= (f_{d-1}(S(\pi_0)) + n)^{2^{n+1}} \end{aligned}$$

From the above discussion, it follows that the functions

$$\begin{aligned} f_d(n) &= f_{d-1}(n) + (f_{d-1}(S(\pi_0)) + g_{d-1}(n))^{2^{g_{d-1}(n)+1}} \\ g_d(n) &= g_{d-1}(n) + \sum_{i=0}^{g_{d-1}(n)} (f_{d-1}(S(\pi_0)) + i)^{2^{i+1}} \end{aligned}$$

are elementary and satisfy the conditions above. This concludes the proof. □

**Theorem 4** *There is a finite set of free algebras $\mathscr{A}$ including the algebra $\mathbb{U}$ of unary integers such that for every elementary function $f : \mathbb{N} \to \mathbb{N}$, there is a term $M_f : \mathbb{U} \to !^k\mathbb{U}$ such that $M_f\lceil u \rceil \to_v^* \lceil f(u) \rceil$ (where $\lceil n \rceil$ is the term in $\mathbb{U}$ corresponding to the natural number $n$).*

*Proof.* We will show that if $f : \mathbb{N} \to \mathbb{N}$ is computable by a Turing Machine $\mathcal{M}$ running in elementary time, then there is a term $M_f$ representing that same function. First of all, $\mathscr{A}$ will contain a free algebra $\mathbb{C}$ with four constructors $c^1_{\mathbb{C}}, c^2_{\mathbb{C}}, c^3_{\mathbb{C}}, c^4_{\mathbb{C}}$ having arities $\mathcal{R}^1_{\mathbb{C}} = 4, \mathcal{R}^2_{\mathbb{C}} = 1, \mathcal{R}^3_{\mathbb{C}} = 1, \mathcal{R}^4_{\mathbb{C}} = 0$. Constructors $, c^2_{\mathbb{C}}, c^3_{\mathbb{C}}, c^4_{\mathbb{C}}$ can be used to build binary strings and a configuration will correspond to a term $c^1_{\mathbb{C}} t_1 t_2 t_3 t_4$ where $t_1$ represent the current state, $t_2$ represents the current symbol, $t_3$ represents the left-hand side of the tape and $t_4$ represents the right-hand side of the tape. A closed term $\mathit{trans} : \mathbb{C} \to \mathbb{C}$ encoding the transition function can be built using, in particular, the new constant $\mathit{cond}_{\mathbb{C}}$. Iteration, on the other hand, helps when writing $\mathit{init} : \mathbb{U} \to !\mathbb{C}$ (whose purpose is to translate a unary integer $t$ into the initial configuration of $\mathcal{M}$ for $t$) and $\mathit{final} : \mathbb{C} \to !\mathbb{U}$ (which extracts a unary integer from the final configuration of $M$). In this way, the so-called qualitative part of the encoding can be done. The quantitative part, on the other hand, can be encoded as follows. We will show there are terms $\mathit{tower}^n : \mathbb{U} \to !^{2n}\mathbb{U}$ such that $\mathit{tower}^n \lceil m \rceil \to_v^* \lceil 2_n(m) \rceil$ where $2_0(m) = m$ and $2_{n+1}(m) = 2^{2_n(m)}$ for every $n \geq 0$. We will prove the existence of such terms by induction on $n$. $\mathit{tower}^0 : \mathbb{U} \to \mathbb{U}$ is simply the identity $\lambda x.x$. Consider now the term

$$\mathit{exp} \equiv \lambda x.\mathit{iter}_{\mathbb{U}} x (\lambda y \lambda z.y(yz))(\lambda y.c^1_{\mathbb{U}} y) : \mathbb{U} \to !!(\mathbb{U} \to \mathbb{U})$$

We now prove that for every $m \in \mathbb{N}$, $\mathit{exp} \lceil m \rceil \to_v^* V_m$ where $V_m$ is a value such that $V_m \lceil p \rceil \to_v^* \lceil 2^m + p \rceil$ for every $p \in \mathbb{N}$. We go by induction on $m$. If $m = 0$, then

$$\mathit{exp} \lceil m \rceil \to_v^* (\lambda x.c^1_{\mathbb{U}} x)$$

and $(\lambda x.c^1_{\mathbb{U}} x) \lceil p \rceil \to_v^* \lceil 1 + p \rceil \equiv \lceil 2^m + p \rceil$. If $m > 0$, then

$$\mathit{exp} \lceil m \rceil \to_v^* (\lambda x.\lambda y.x(xy)) V_{m-1} \to_v \lambda y.V_{m-1}(V_{m-1} y)$$

and

$$\lambda y.V_{m-1}(V_{m-1} y) \lceil p \rceil \to_v^* V_{m-1} \lceil 2^{m-1} + p \rceil \to_v^* \lceil 2^{m-1} + 2^{m-1} + p \rceil \equiv \lceil 2^m + p \rceil$$

$\mathit{tower}^n$ is

$$\lambda x.(\lambda y.\mathit{tower}^{n-1} y)((\lambda z.z \lceil 0 \rceil)(\mathit{esp}\ x))$$

Finally, we need terms $\mathit{coerc}^n : \mathbb{U} \to !^n \mathbb{U}$ such that $\mathit{coerc}^n \lceil m \rceil \to_v^* \lceil m \rceil$. $\mathit{coerc}^0$ is simply the identity, while $\mathit{coerc}^n$ is $\lambda x.\mathit{iter}_{\mathbb{U}} x c^1_{\mathbb{U}} c^2_{\mathbb{U}}$ for every $n \geq 1$. We can suppose there is $d$ such that $\mathcal{M}$ performs at most $2_d(n)$ steps processing any input of length $n$. The term $M_f$ encoding $f$ will then be:

$$\lambda x.(\lambda y.\mathit{final}\ y)((\lambda z.\lambda v.\mathit{iter}_{\mathbb{U}}\ z\ \mathit{trans}\ (\mathit{init}\ v))(\mathit{coerc}^{2d}\ x)(\mathit{tower}^d\ x))$$

□