Quantitative models and Implicit Complexity

Ugo Dal Lago
Dipartimento di Scienze dell'Informazione
Università di Bologna
dallago@cs.unibo.it

Martin Hofmann
Institut für Informatik
Ludwig-Maximilians-Universität, München
mhofmann@informatik.uni-muenchen.de

Abstract

We give new proofs of soundness (all representable functions on base types lies in certain complexity classes) for Light Affine Logic, Elementary Affine Logic, LFPL *and Soft Affine Logic. The proofs are based on a common semantical framework which is merely instantiated in four different ways. The framework consists of an innovative modification of realizability which allows us to use resource-bounded computations as realisers as opposed to including all Turing computable functions as is usually the case in realizability constructions. For example, all realisers in the model for* LFPL *are polynomially bounded computations whence soundness holds by construction of the model. The work then lies in being able to interpret all the required constructs in the model. While being the first entirely semantical proof of polytime soundness for light logics, our proof also provides a notable simplification of the original already semantical proof of polytime soundness for* LFPL. *A new result made possible by the semantic framework is the addition of polymorphism and a modality to* LFPL.

1 Introduction

In recent years, a large number of characterizations of complexity classes based on logics and lambda calculi have appeared. At least three different principles have been exploited, namely linear types [3, 7], restricted modalities in the context of linear logic [5, 1, 9] and non-size-increasing computation [6]. Although related one to the other, these systems have been studied with different, often unrelated methodologies and few results are known about relative intentional expressive power. We believe that this area of implicit computational complexity needs unifying frameworks for the analysis of quantitative properties of computation. This would help to improve the understanding on existing systems. More importantly, unifying frameworks can be used *themselves* as a foundation for controlling the use of resources inside programming languages.

In this paper, we give new proofs of soundness (all representable functions on base types lies in certain complexity classes) for Light Affine Logic (LAL, [1]), Elementary Affine Logic (EAL, [4]), LFPL [6] and Soft Affine Logic (SAL, [2]). The proofs are based on a common semantical framework which is merely instantiated in four different ways. The framework consists of an innovative modification of realizability which allows us to use resource-bounded computations as realisers as opposed to including all Turing computable functions as is usually the case in realizability constructions. For example, all realisers in the model for LFPL are polynomially bounded computations whence soundness holds by construction of the model. The work then lies in being able to interpret all the required constructs in the model. While being the first entirely semantical proof of polytime soundness for light logics, our proof also provides a notable simplification of the original already semantical proof of polytime soundness for LFPL. A new result made possible by the semantic framework is the addition of polymorphism and a modality to LFPL.

The rest of the paper is organized as follows. In section 2 we describe an abstract computational model that will be used in the rest of the paper. In section 3 we introduce length spaces and show they can be used to interpret multiplicative linear logic with free weakening. Sections 4, 5 and 6 are devoted to present instances of the framework together with soundness results for elementary, soft and light affine logics. Section 7 presents a further specialization of length spaces and a new soundness theorem for LFPL based on it.

2 An Abstract Computational Model

In this paper, we rely on an abstract computational framework rather than a concrete one like Turing Machines. This, in particular, will simplify proofs.

Let $L = \{0,1\}^*$ be the set of finite binary sequences. We assume a pairing function $\langle \cdot, \cdot \rangle : L \times L \to L$ so that pairing and projections are computable in linear time or similar. Furthermore, we assume a length function $|\cdot| : L \to \mathbb{N}$ such that $|\langle x, y \rangle| = |x| + |y| + O(1)$ and $|x| \leq \mathit{length}(x)$. We assume a partial application $\{e\}(x) \in L$ for $e, x \in L$ and an abstract time measure $Time(\{e\}(x)) \in \mathbb{N}$ such that $Time(\{e\}(x))$ is defined whenever $\{e\}(x)$ is and, moreover, there exists a fixed polynomial p such that

- $\{e\}(x)$ can be evaluated on a Turing machine in time bounded by $p(Time(\{e\}(x)))$.
- L can be embedded into iself by a map $\Phi : L \to L$ such that both Φ and Φ^{-1} can be computed in polynomial time.
- For each Turing machine M there is $e \in L$ so that $\{e\}(\Phi(x))$ equals $\Phi(y)$, where y is the result of running M on input x. Furthermore, $Time(\{e\}(\Phi(x)))$ bounds the number of steps needed by M on input x.
- For any $e, d \in L$ there exists $d \circ e \in L$ such that $|d \circ e| = |d| + |e| + O(1)$ and $\{d \circ e\}(x) = \{d\}(y)$ where $y = \{e\}(x)$ and moreover $Time(\{d \circ e\}(x)) = Time(\{e\}(x)) + Time(\{d\}(y)) + O(|x| + |y|)$.
- There is e_{id} such that $\{e_{id}\}(x) = x$ for every x and $Time(\{e_{id}\}(x)) = O(|x|)$.
- There is e_{swap} such that $\{e_{swap}\}(\langle x, y \rangle) = \langle y, x \rangle$ and $Time(\{e_{swap}\}(z)) \leq O(|z|)$.
- There is e_{eval} such that $\{e_{eval}\}(\langle d, c \rangle) = \{d\}(c)$ and $Time(\{e_{eval}\}(\langle d, c \rangle)) = Time(\{d\}(c)) + O(|d| + |c|)$.
- For each $e \in L$ there is $e^* \in L$ with $|e^*| = |e| + O(1)$ such that $\{e^*\}(\langle m, n \rangle) = \langle \{e\}(m), n \rangle$ and $Time(\{e^*\}(\langle m, n \rangle)) = Time(\{e\}(m)) + O(|m| + |n| + |e| + |\{e\}(m)|)$.
- There is e_{assl} such that $\{e_{assl}\}(\langle x, \langle y, z \rangle \rangle) = \langle \langle x, y \rangle, z \rangle$ and $Time(\{e_{assl}\}(x)) = O(|x|)$.
- There is e_{contr} such that $\{e_{contr}\}(x) = \langle x, x \rangle$ and $Time(\{e_{contr}\}(x)) = O(|x|)$.
- There is e_{curry} such that, for each e, $d = \{e_{curry}\}(e)$ exists and satisfies $|d| = |e| + O(1)$ and $Time(\{e_{curry}\}(e)) = O(|e|)$; moreover, for every x, $c_x = \{d\}(x)$ exists and satisfies $|c_x| = |e| + |x| + O(1)$ and $Time(\{d\}(x)) = O(|x|)$; finally, for every y, $\{c_x\}(y) = \{e\}(\langle x, y \rangle)$ and $Time(\{c_x\}(y)) = Time(\{e\}(\langle x, y \rangle)) + O(|x| + |y|)$.

There are a number of ways to instantiate this framework, among them SECD machines, Turing machines with an adapted definition of length and time and call-by-value lambda calculi.

Now, we will sketch a possible instance of this abstract computational framework, namely a call-by-value lambda-calculus with constants. We assume an enumeration of Turing Machines by strings in L. $[e](x)$ is the result of running the Turing Machine corresponding to e on input x and $Time([e](x))$ denotes the number of steps needed by the Turing Machine corresponding to e when feeded by x.

Terms are defined by the following productions:

$$M ::= x \mid MM \mid \lambda x.M \mid \mathit{simul} \mid \mathit{appzero} \mid \mathit{appone} \mid \mathit{epsilon}$$

Every string $s \in L$ corresponds to a term $\Phi(s)$ in the following way:

$$\begin{aligned} \Phi(\varepsilon) &= \mathit{epsilon} \\ \Phi(0s) &= \mathit{appzero}\Phi(s) \\ \Phi(1s) &= \mathit{appone}\Phi(s) \end{aligned}$$

Values are defined by the following productions:

$$V ::= x \mid \lambda x.M \mid \mathit{simul} \mid \mathit{appzero} \mid \mathit{appone} \mid \mathit{epsilon} \mid \Phi(s)$$

where s ranges over L. We define the size $|M|$ of a term M as follows:

$$\begin{aligned} |x| = |\mathit{simul}| = |\mathit{appzero}| = |\mathit{appone}| = |\mathit{epsilon}| &= 1 \\ |MN| &= |M| + |N| + 1 \\ |\lambda x.M| &= |M| + 1 \end{aligned}$$

The number $FO(x,M)$ of free occurrences of x inside M can be defined in the usual way. We are now ready to define a ternary reduction relation $M \rightsquigarrow_n N$ where M, N are terms and n is a natural number. First of all, define $\rightarrow_n$ by the following two rules:

$$\begin{array}{rll} (\lambda x.M)V & \rightarrow_{|V|FO(x,M)} & M\{V/x\} \\ simulate\Phi(e)\Phi(x) & \rightarrow_{Time([e](x))} & \Phi([e](x)) \end{array}$$

$M \rightsquigarrow_n N$ is defined as follows:

$$\frac{M \rightarrow_n N}{M \rightsquigarrow_n N} \quad \frac{M \rightsquigarrow_n N}{ML \rightsquigarrow_n NL} \quad \frac{M \rightsquigarrow_n N}{LM \rightsquigarrow_n LN} \quad \frac{M \rightsquigarrow_n N \quad N \rightsquigarrow_m L}{M \rightsquigarrow_{n+m} L}$$

Let M, N be terms. Consider the following terms:

$$\begin{array}{rcl} \langle M, N\rangle & \equiv & \lambda x.xMN \\ N \circ M & \equiv & \lambda x.N(Mx) \\ M_{id} & \equiv & \lambda x.x \\ M_{swap} & \equiv & \lambda x.x(\lambda y.\lambda w.\lambda z.zwy) \\ M_{assl} & \equiv & \lambda x.x(\lambda y.\lambda w.w(\lambda z.\lambda q.\lambda r.r(\lambda s.syz)q)) \\ M_{eval} & \equiv & \lambda x.x(\lambda y.\lambda w.yw) \\ M_{contr} & \equiv & \lambda x.\lambda y.yxx \\ M_{curry} & \equiv & \lambda x.\lambda y.\lambda w.x(\lambda z.zyw) \\ M^* & \equiv & \lambda x.x(\lambda y.\lambda w.(\lambda x.\lambda z.zxw)(My)) \end{array}$$

If U and V are closed values and UV has a normal form W, then we will denote W by $\{U\}(V)$. Moreover, $Time(\{U\}(V))$ is the largest integer n such that $UV \rightsquigarrow_n W$. Now, let M, N, L, V, W, U be closed values such that $\{M\}(V) = W$ and $\{N\}(W) = U$. One can verify that

$$\begin{array}{rcl} Time(\{(N \circ M)V\}(U)) & = & |V| + Time(\{M\}(V)) + Time(\{N\}(W)) \\ Time(\{M_{id}\}(V)) & = & |V| \\ Time(\{M_{swap}\}(\langle M, N\rangle)) & = & 2|M| + 2|N| + 12 \\ Time(\{M_{assl}\}(\langle M, \langle N, L\rangle\rangle)) & = & 3|M| + 3|N| + 3|L| + 41 \\ Time(\{M_{eval}\}(\langle M, N\rangle)) & = & 2|M| + 2|N| + 9 + Time(\{M\}(N)) \\ Time(\{M_{contr}\}(M)) & = & 2|M| \\ Time(\{M^*\}(\langle V, N\rangle)) & = & 2|V| + 2|N| + |M| + |W| + 16 + Time(\{M\}(V)) \end{array}$$

Now, let M, V, W, U be closed values such that $\{M\}(\langle V, W\rangle) = U$. Then

$$\begin{array}{rll} M_{curry}M & \rightsquigarrow_{|M|} & \lambda y.\lambda w.M(\lambda w.wyw) \equiv N \\ NV & \rightsquigarrow_{|V|} & \lambda w.M(\lambda z.zVw) \equiv L \\ LW & \rightsquigarrow_{|W|} & M(\lambda z.zVW) \end{array}$$

As a consequence

$$\begin{array}{rcl} Time(\{M_{curry}\}(M)) & = & |M| \\ Time(\{N\}(V)) & = & |V| \\ Time(\{L\}(W)) & = & |W| + Time(\{M\}(\langle V, W\rangle)) \end{array}$$

This proves this lambda-calculus to satisfies all the axioms. In the following, cp will be a fixed constant such that $|\langle x, y\rangle| \leq |x| + |y| + cp$.

3 Length Spaces

In this section, we introduce the category of length spaces and study its properties. Lengths will not necessarily be numbers but rather elements of a commutative monoid.

A *resource monoid* is a quadruple $M = (|M|, +, \leq_M, \mathcal{D}_M)$ where

1. $(|M|, +)$ is a commutative monoid;
2. $\leq_M$ is a pre-order on $|M|$ which is compatible with $+$;
3. $\mathcal{D}_M : \{(\alpha, \beta) \mid \alpha \leq_M \beta\} \to \mathbb{N}$ is a function such that for every α, β, γ

$$\begin{array}{rcl} \mathcal{D}_M(\alpha,\beta) + \mathcal{D}_M(\beta,\gamma) & \leq & \mathcal{D}_M(\alpha,\gamma) \\ \mathcal{D}_M(\alpha,\beta) & \leq & \mathcal{D}_M(\alpha+\gamma,\beta+\gamma) \end{array}$$

and, moreover, for every $n \in \mathbb{N}$ there is α such that $\mathcal{D}_M(0,\alpha) \geq n$.

Given a resource monoid $M = (|M|, +, \leq_M, \mathcal{D}_M)$, the function $\mathcal{F}_M : |M| \to \mathbb{N}$ is defined by putting $\mathcal{F}_M(\alpha) = \mathcal{D}_M(0,\alpha)$.

Lemma 1 *If M is a resource monoid, then $\mathcal{D}_M$ is antitone on its first argument and monotone on its second argument.*

Proof. If $\alpha \leq_M \beta$, then

$$\begin{array}{rcl} \mathcal{D}_M(\alpha,\gamma) & \geq & \mathcal{D}_M(\alpha,\beta) + \mathcal{D}_M(\beta,\gamma) \geq \mathcal{D}_M(\beta,\gamma) \\ \mathcal{D}_M(\gamma,\alpha) & \leq & \mathcal{D}_M(\gamma,\alpha) + \mathcal{D}_M(\alpha,\beta) \geq \mathcal{D}_M(\gamma,\beta) \end{array}$$

□

A *length space* on a resource monoid $M = (|M|, +, \leq_M, \mathcal{D}_M)$ is a pair $A = (|A|, \Vdash_A)$, where $|A|$ is a set and

$$\Vdash \subseteq |M| \times L \times |A|$$

is a relation satisfying the following conditions:

- If $(\alpha, e, a) \in \Vdash_A$, then $\mathcal{F}_M(\alpha) \geq |e|$;
- For every $a \in |A|$, there are α, e such that $(\alpha, e, a) \in \Vdash_A$;
- If $(\alpha, e, a) \in \Vdash_A$ and $\alpha \leq_M \beta$, then $(\beta, e, a) \in \Vdash_A$;
- If $(\alpha, e, a) \in \Vdash_A$ and $(\alpha, e, b) \in \Vdash_A$, then $a = b$.

The last requirement implies that each element of $|A|$ is uniquely determined by the (nonempty) set of it realisers and in particular limits the cardinality of any length space to the number of partial equivalence relations on L.

We will usually write $\alpha, e \Vdash_A a$ meaning $(\alpha, e, a) \in \Vdash_A$.

A *morphism* from length space $A = (|A|, \Vdash_A)$ to length space $B = (|B|, \Vdash_B)$ (on the same resource monoid $M = (|M|, +, \leq_M, \mathcal{D}_M)$) is a function $f : |A| \to |B|$ such that there exist $e \in \{0,1\}^*$, $\varphi \in |M|$ with $\mathcal{F}_M(\varphi) \geq |e|$ and whenever $\alpha, d \Vdash_A a$, there must be β, c such that

1. $\beta, c \Vdash_B f(a)$;
2. $\beta \leq_M \varphi + \alpha$;
3. $\{e\}(d) = c$;
4. $Time(\{e\}(d)) \leq \mathcal{F}_M(\varphi + \alpha)\mathcal{D}_M(\beta, \varphi + \alpha)$

We call e a realizer of f and φ a majorizer of f.

If f is a morphism from A to B realized by e and majorized by φ, then we will write $f : A \xrightarrow{e,\varphi} B$ or $\varphi, e \Vdash_{A \multimap B} f$.

Given two length spaces $A = (|A|, \Vdash_A)$ and $B = (|B|, \Vdash_B)$ on the same resource monoid M, we can build $A \otimes B = (|A| \times |B|, \Vdash_{A\otimes B})$ (on M) where $e, \alpha \Vdash_{A\otimes B} (a, b)$ iff there are f, g, β, γ with

$$\begin{gathered} f, \beta \Vdash_A a \\ g, \gamma \Vdash_B b \\ e = \langle f, g \rangle \\ \alpha \geq_M \beta + \gamma \\ \mathcal{F}_M(\alpha) \geq \mathcal{F}_M(\beta) + \mathcal{F}_M(\gamma) + cp \end{gathered}$$

$A \otimes B$ is a well-defined length space due to the axioms on M.

Given A and B as above, we can build $A \multimap B = (|A| \Rightarrow |B|, \Vdash_{A\multimap B})$ where $e, \alpha \Vdash_{A\multimap B} f$ iff f is a morphism from A to B realized by e and majorized by α.

Morphisms can be composed:

Lemma 2 (Composition) *If $f : A \to B$ and $g : B \to C$ are morphisms, then $g \circ f : A \to C$ is a morphism, too.*

Proof. Let $f : A \xrightarrow{e,\varphi} B$ and $g : B \xrightarrow{d,\psi} C$. We know that $Time(\{d \circ e\}(k))$ is bounded by $Time(\{e\}(k)) + Time(\{d\}(l))$ (where $l = \{e\}(k)$) plus some overhead proportional to $|k|$ and $|l|$, say $p|k| + q|l| + r$. Now, let us now choose μ such that $\mathcal{F}_M(\mu) \geq |d \circ e| + p + q + r$. We will prove that $g \circ f : A \xrightarrow{d\circ e, \varphi+\psi+\mu} C$. Obviously, $\mathcal{F}_M(\varphi + \psi + \mu) \geq |d \circ e|$. If $\alpha, n \Vdash_A a$, then there must be β, m such that $\beta, m \Vdash_B f(a)$ and the other conditions prescribed by the definition of a morphism hold. Moreover, there must be γ, s such that $\gamma, s \Vdash_C g(f(a))$ and, again, the other conditions are satisfied. Putting them together, we get:

$$\gamma \leq_M \beta + \psi \leq_M \alpha + \varphi + \psi \leq_M \alpha + \varphi + \psi + \mu$$

and

$$\begin{array}{rcl} Time(\{d \circ e\}(n)) & \leq & Time(\{e\}(n)) + Time(\{d\}(m)) + p|k| + q|l| + r \\ & \leq & \mathcal{F}_M(\alpha + \varphi)\mathcal{D}_M(\beta, \alpha + \varphi) + \mathcal{F}_M(\beta + \psi)\mathcal{D}_M(\gamma, \beta + \psi) \\ & & + p\mathcal{F}_M(\alpha) + q\mathcal{F}_M(\beta) + r\mathcal{F}_M(\mu) \\ & \leq & \mathcal{F}_M(\alpha + \varphi + \psi)\mathcal{D}_M(\beta + \psi, \alpha + \varphi + \psi) + \mathcal{F}_M(\alpha + \varphi + \psi)\mathcal{D}_M(\gamma, \beta + \psi) \\ & & + (p + q + r)(\mathcal{F}_M(\alpha + \varphi + \psi + \mu)) \\ & \leq & \mathcal{F}_M(\alpha + \varphi + \psi + \mu)\mathcal{D}_M(\gamma, \alpha + \varphi + \psi) + \mathcal{F}_M(\alpha + \varphi + \psi + \mu)\mathcal{F}_M(\mu) \\ & \leq & \mathcal{F}_M(\alpha + \varphi + \psi + \mu)\mathcal{D}_M(\gamma, \alpha + \varphi + \psi + \mu) \end{array}$$

This concludes the proof. □

Basic morphisms can be built independently on the underlying resource monoid. Noticeably, they correspond to axiom of multiplicative linear logic:

Lemma 3 (Basic Maps) *Given length spaces A, B, C, there are morphisms:*

$$\begin{array}{rcl} id & : & A \to A \\ swap & : & A \otimes B \to B \otimes A \\ assl & : & A \otimes (B \otimes C) \to (A \otimes B) \otimes C \\ eval & : & A \otimes (A \multimap B) \to B \\ curry & : & (A \otimes B) \multimap C) \to A \multimap (B \multimap C) \end{array}$$

where

$$\begin{array}{rcl} id(a) & = & a \\ swap(a, b) & = & (b, a) \\ assl(a, (b, c)) & = & ((a, b), c) \\ eval(a, f) & = & f(a) \\ curry(f) & = & \lambda a.\lambda b.f(a, b) \end{array}$$

Proof. We know that $\{e_{id}\}(d)$ take time linear in $|d|$, say at most $p|d|+q$. Then, let $\varphi_{id} \in M$ be such that $\mathcal{F}_M(\varphi_{id}) \geq p+q+|e_{id}|$ (this can always be done). Now, let $\alpha, d \Vdash_A a$. We have that $\alpha, d \Vdash_A id(a)$, $\alpha \leq_M \alpha + \varphi_{id}$, $\{e_{id}\}(d) = d$. Moreover

$$\begin{array}{rcl} Time(\{e_{id}\}(d)) & \leq & p|d|+q \leq (p+q)(|d|+p+q) \leq \mathcal{F}_M(\varphi_{id})(\mathcal{F}_M(\alpha)+\mathcal{F}_M(\varphi_{id})) \\ & \leq & (\mathcal{F}_M(\varphi_{id})+\mathcal{D}_M(\alpha,\alpha))(\mathcal{F}_M(\varphi_{id})+\mathcal{F}_M(\alpha)) \\ & \leq & \mathcal{D}_M(\alpha,\alpha+\varphi_{id})\mathcal{F}_M(\varphi_{id}+\alpha) \end{array}$$

This proves id to be a morphism.

We know that $\{e_{swap}\}(\langle d, c\rangle)$ takes time linear in $|d|+|c|$, say at most $p|d|+q|c|+r$. Then, let $\varphi_{swap} \in |M|$ be such that $\mathcal{F}_M(\varphi_{id}) \geq p+q+r+|e_{swap}|$. Now, let $\alpha, e \Vdash_{A\otimes B} (a,b)$. This means that $e = \langle d, c\rangle$ and $\alpha, \langle c, d\rangle \Vdash_{B\otimes A} (b,a)$. We can then apply the same argument as for id. In particular:

$$\begin{array}{rcl} Time(\{e_{swap}\}(e)) & \leq & p|d|+q|c|+r \leq (p+q+r)(|d|+|c|+p+q+r) \\ & \leq & (p+q+r)(|e|+p+q+r) \leq \mathcal{F}_M(\varphi_{swap})(\mathcal{F}_M(\alpha)+\mathcal{F}_M(\varphi_{swap})) \\ & \leq & (\mathcal{F}_M(\varphi_{swap})+\mathcal{D}_M(\alpha,\alpha))(\mathcal{F}_M(\varphi_{swap})+\mathcal{F}_M(\alpha)) \\ & \leq & \mathcal{D}_M(\alpha,\alpha+\varphi_{swap})\mathcal{F}_M(\varphi_{swap}+\alpha) \end{array}$$

This proves *swap* to be a morphism. We can verify *assl* to be a morphism exactly in the same way.

We know that $\{e_{eval}\}(\langle d, c\rangle) = \{d\}(c)$ and $\{e_{eval}\}(\langle d, c\rangle)$ takes overload time linear in $|d|+|c|$, say at most $p|d|+q|c|+r$. φ_{eval} is chosen as to satisfy $\mathcal{F}_M(\varphi_{eval}) \geq p+q+r+|e_{eval}|$. Let now $\alpha, e \Vdash_{A\otimes(A\multimap B)} (a,f)$. This means that $e = \langle d, c\rangle$ and there are β and γ such that

$$\begin{array}{c} \beta, d \Vdash_A a \\ \gamma, c \Vdash_{A\multimap B} f \\ \alpha \geq_M \beta+\gamma \\ \mathcal{F}_M(\alpha) \geq \mathcal{F}_M(\beta)+\mathcal{F}_M(\gamma)+cp \end{array}$$

From $\gamma, c \Vdash_{A\multimap B} f$ it follows that, by the definition of a morphism, there must be δ, h such that

1. $\delta, h \Vdash_B f(a)$
2. $\delta \leq_M \beta+\gamma$
3. $\{c\}(d) = h$
4. $Time(\{c\}(d)) \leq \mathcal{F}_M(\beta+\gamma)\mathcal{D}_M(\delta, \beta+\gamma)$

From $\delta \leq_M \beta+\gamma$ and $\beta+\gamma \leq_M \alpha$, it follows that $\delta \leq_M \alpha \leq_M \alpha+\mu$. Moreover:

$$\begin{array}{rcl} Time(\{e_{eval}\}(\langle d, c\rangle)) & \leq & p|d|+q|c|+r+Time(\{c\}(d)) \\ & \leq & (p+q+r)(|d|+|c|+p+q+r)+Time(\{c\}(d)) \\ & \leq & \mathcal{F}_M(\varphi_{eval})\mathcal{F}_M(\alpha+\varphi_{eval})+\mathcal{F}_M(\beta+\gamma)\mathcal{D}_M(\delta,\beta+\gamma) \\ & \leq & \mathcal{F}_M(\alpha+\varphi_{eval})\mathcal{D}_M(0,\varphi_{eval})+\mathcal{F}_M(\alpha+\varphi_{eval})\mathcal{D}_M(\delta,\alpha) \\ & \leq & \mathcal{F}_M(\alpha+\varphi_{eval})\mathcal{D}_M(\delta,\alpha+\varphi_{eval}) \end{array}$$

Now, let us prove that *curry* is a morphism. First of all, we know there must be constants $p_1, \ldots, p_9$ such that, for each e, x, y, there are d and c_x with

$$\begin{array}{rcl} Time(\{e_{curry}\}(e)) & \leq & p_1|e|+p_2 \\ d & = & \{e_{curry}\}(e) \\ |d| & \leq & |e|+p_3 \\ Time(\{d\}(x)) & \leq & p_4|x|+p_5 \\ c_x & = & \{d\}(x) \\ |c_x| & \leq & |e|+|x|+p_6 \\ Time(\{c_x\}(y)) & \leq & Time(\{e\}(\langle x, y\rangle))+p_7|x|+p_8|y|+p_9 \end{array}$$

Let $\mu, \theta, \xi, \sigma \in |M|$ be such that $\mathcal{F}_M(\mu) \geq cp$, $\mathcal{F}_M(\theta) \geq p_6 + p_7 + p_8 + p_9$, $\mathcal{F}_M(\xi) \geq p_3 + p_4 + p_5$ and $\mathcal{F}_M(\sigma) \geq p_1 + p_2$. Finally, put $\varphi_{curry} = \mu + \theta + \xi + \sigma$. Let now $\alpha, x \Vdash_A a$, $\beta, y \Vdash_B b$ and $\gamma, e \Vdash_{A \otimes B \multimap C} f$. By definition of a morphism, there must be δ, c such that $\alpha + \beta, c \Vdash_C f(a, b)$. Using the usual techiques, we can show that $\alpha + \gamma + \mu + \theta, c \Vdash_{B \multimap C} \lambda b.f(a, b)$, which in turn yields $\gamma + \mu + \theta + \xi, c \Vdash_{B \multimap C} \lambda a.\lambda b.f(a, b)$. Finally, this means that $curry$ is a morphism justified by e_{curry} and majorized by $\mu + \theta + \xi + \sigma = \varphi_{curry}$. This concludes the proof. $\Box$.

Length spaces can justify the usual rule for tensor as a map-former:

Lemma 4 (Tensor) *If $f : A \to B$ is a morphism and C is a length space, then $f \times id : A \otimes C \to B \otimes C$ is a morphism, too.*

Proof. Let $f : A \xrightarrow{e,\varphi} B$. We know that and $Time(\{e^*\}(\langle m, n\rangle))$ is at most $Time(\{e\}(m)) + p|\{e\}(m)| + q|n| + r|m| + s|e| + u$ where p, q, r, s, u are constants. Then, take $\psi \in M$ such that $\mathcal{F}_M(\psi) \geq p + q + r + s + u + |e^*|$, put $\sigma = \psi + \varphi + \mu$, where $\mathcal{F}_M(\mu) \geq cp$. Suppose $\langle m, n\rangle, \alpha \Vdash_{A \otimes C} (a, c)$. By definition, there are β, γ such that

$$\begin{gathered} m, \beta \Vdash_A a \\ n, \gamma \Vdash_C c \\ \alpha \geq_M \beta + \gamma \end{gathered}$$

By hypothesis, there are δ, t such that

$$\begin{gathered} t, \delta \Vdash_B f(a) \\ \delta \leq_M \varphi + \beta \\ \{e\}(m) = t \\ Time(\{e\}(m)) \leq \mathcal{F}_M(\varphi + \beta)\mathcal{D}_M(\delta, \varphi + \beta) \end{gathered}$$

Then, $\gamma + \delta + \mu, \langle t, n\rangle \Vdash_{A \otimes B} (f(a), c)$. Moreover,

$$\gamma + \delta + \mu \leq_M \gamma + \varphi + \beta + \mu \leq_M \alpha + \varphi + \mu \leq_M \alpha + \sigma$$

Finally:

$$\begin{array}{rcl} Time(\{e^*\}(\langle m, n\rangle)) & \leq & Time(\{e\}(m)) + p|\{e\}(m)| + q|n| + r|m| + s \\ & \leq & \mathcal{F}_M(\varphi + \beta)\mathcal{D}_M(\delta, \varphi + \beta) + \\ & & p\mathcal{F}_M(\delta) + q\mathcal{F}_M(\gamma) + r\mathcal{F}_M(\beta) + s\mathcal{F}_M(\phi) + u\mathcal{F}_M(\psi) \\ & \leq & \mathcal{F}_M(\sigma + \alpha)(\mathcal{D}_M(\delta, \varphi + \beta) + p + q + r + s + u) \\ & \leq & \mathcal{F}_M(\alpha + \sigma)(\mathcal{D}_M(\delta + \gamma + \mu, \varphi + \beta + \gamma + \mu) + p + q + r + s + u) \\ & \leq & \mathcal{F}_M(\alpha + \sigma)\mathcal{D}_M(\delta + \gamma + \mu, \alpha + \sigma) \end{array}$$

This concludes the proof. $\Box$

Thus:

Lemma 5 *Length spaces and their morphisms form a symmetric monoidal closed category with tensor and linear implication given as above.*

A length space I is defined by $|I| = \{0\}$ and $\alpha, e \Vdash_A 0$ when $\mathcal{F}_M(\alpha) \geq |e|$. For each length space A there are isomorphisms $A \otimes I \simeq A$ and a unique morphism $A \to I$. The latter serves to justify full weakening.

For every resource monoid M, there is a length space $L_M = (|L_M|, \Vdash_{L_M})$ where $|L_M| = L$ and $\alpha, t \Vdash_{L_M} t$ whenever $\mathcal{F}_M(\alpha) \geq t$. The function s_0 (respectively, s_1) from $\{0, 1\}^*$ to itself which appends 0 (respectively, 1) to the left of its argument can be computed in linear time on a Turing Machine and, as a consequence, is a morphism from L_M to itself.

Identity, Cut and Weakening.

$$\frac{}{A \vdash A}\ I \qquad \frac{\Gamma \vdash A \quad \Delta, A \vdash B}{\Gamma, \Delta \vdash B}\ U \qquad \frac{\Gamma \vdash A}{\Gamma, B \vdash A}\ W$$

Multiplicative Logical Rules.

$$\frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C}\ L_{\otimes} \qquad \frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}\ R_{\otimes}$$

$$\frac{\Gamma \vdash A \quad \Delta, B \vdash C}{\Gamma, \Delta, A \multimap B \vdash C}\ L_{\multimap} \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B}\ R_{\multimap}$$

Second Order Logical Rules.

$$\frac{\vdash \Gamma, A[C/\alpha] \vdash B}{\Gamma, \forall \alpha.A \vdash B}\ L^{\forall} \qquad \frac{\Gamma \vdash A \quad \alpha \notin FV(\Gamma)}{\Gamma \vdash \forall \alpha.A}\ R^{\forall}$$

Figure 1. Intuitionistic Multiplicative Affine Logic

3.1 Interpreting Multiplicative Affine Logic

We can now formally show that second order multiplicative affine logic (i.e. multiplicative linear logic plus full weakening) can be interpreted inside the category of length spaces on any monoid M. Doing this will simplify the analysis of richer systems presented in following sections. Formulae of (intuitionistic) multiplicative affine logic are generated by the following productions:

$$A ::= \alpha \mid A \multimap A \mid A \otimes A \mid \forall \alpha.A$$

where α ranges over a countable set of atoms. Rules are reported in figure 1. A *realizability environment* is a partial function assigning length spaces (on the same resource monoid) to atoms. Realizability semantics $[\![A]\!]^{\mathscr{R}}_{\eta}$ of a formula A on the realizability environment η is defined by induction on A:

$$\begin{aligned}
[\![\alpha]\!]^{\mathscr{R}}_{\eta} &= \eta(\alpha) \\
[\![A \otimes B]\!]^{\mathscr{R}}_{\eta} &= [\![A]\!]^{\mathscr{R}}_{\eta} \otimes [\![B]\!]^{\mathscr{R}}_{\eta} \\
[\![A \multimap B]\!]^{\mathscr{R}}_{\eta} &= [\![A]\!]^{\mathscr{R}}_{\eta} \multimap [\![B]\!]^{\mathscr{R}}_{\eta} \\
[\![\forall \alpha.A]\!]^{\mathscr{R}}_{\eta} &= (|[\![\forall \alpha.A]\!]^{\mathscr{R}}_{\eta}|, \Vdash_{[\![\forall \alpha.A]\!]^{\mathscr{R}}_{\eta}})
\end{aligned}$$

where

$$\begin{aligned}
|[\![\forall \alpha.A]\!]^{\mathscr{R}}_{\eta}| &= \prod_{C \in \mathscr{U}} |[\![A]\!]^{\mathscr{R}}_{\eta[\alpha \to C]}| \\
e, \alpha \Vdash_{[\![\forall \alpha.A]\!]^{\mathscr{R}}_{\eta}} a &\iff \forall C. e, \alpha \Vdash_{[\![A]\!]^{\mathscr{R}}_{\eta[\alpha \to C]}} a
\end{aligned}$$

Here $\mathscr{U}$ stands for the class of all length spaces. A little care is needed when defining the product since strictly speaking it does not exist for size reasons. The standard way out is to let the product range over those length spaces whose underlying set equals the set of equivalence classes of a partial equivalence relation on L. As already mentioned every length space is isomorphic to one such. When working with the product one has to insert these isomorphisms in appropriate places which, however, we elide to increase readability.

If $n \geq 0$ and $A_1, \ldots, A_n$ are formulas, the expression $[\![A_1 \otimes \ldots \otimes A_n]\!]^{\mathscr{R}}_{\eta}$ stands for $|I|$ if $n = 0$ and $[\![A_1 \otimes \ldots \otimes A_{n-1}]\!]^{\mathscr{R}}_{\eta} \otimes [\![A]\!]^{\mathscr{R}}_{\eta}$ if $n \geq 1$.

4 Elementary Length Spaces

In this section, we define a resource monoid $\mathcal{L}$ such that elementary affine logic can be interpreted in the category of resource monoids on $\mathcal{L}$. We then (re)prove that functions representable in EAL are elementary time computable.

A *list* is either $empty$ or $cons(n, l)$ where $n \in \mathbb{N}$ and l is itself a list. The sum $l + h$ of two lists l and h is defined as follows, by induction on l:

$$\begin{array}{rcl} empty + h = h + empty & = & h \\ cons(n, l) + cons(m, h) & = & cons(n + m, l + h) \end{array}$$

For every $e \in \mathbb{N}$, binary relations $\leq_e$ on lists can be defined as follows

- $empty \leq_e l$;
- $cons(n, l) \leq_e cons(m, h)$ iff there is $d \in \mathbb{N}$ such that
  1. $n \leq 2^e(m + e) - d$;
  2. $l \leq_d h$.

For every e and for every lists l and h with $l \leq_e h$, we define the natural number $\mathcal{D}_e(l, h)$ as follows:

$$\begin{array}{rcl} \mathcal{D}_e(empty, empty) & = & 0; \\ \mathcal{D}_e(empty, cons(n, l)) & = & 2^e(n + e) + \mathcal{D}_{2^e(n+e)}(empty, l); \\ \mathcal{D}_e(cons(n, l), cons(m, h)) & = & 2^e(m + e) - n + \mathcal{D}_{2^e(m+e)-n}(l, h); \end{array}$$

Given a list l, $!l$ stands for the list $cons(0, l)$. The depth $depth(l)$ of a list l is defined by induction on l: $depth(empty) = 0$ while $depth(cons(n, l)) = depth(l) + 1$. $|l|$ stands for the maximum integer appearing inside l, i.e. $|empty| = 0$ and $|cons(n, l)| = \max\{|l|, n\}$. For every natural number n, $[n]_{\mathcal{L}}$ stands for $cons(n, empty)$.

We can now verify that all the necessary conditions required by the definition of a resource monoid are satisfied. To do this, we need a number of preliminary results, which can all be proved by simple inductions and case-analysis:

Lemma 6 (Compatibility) *$empty \leq_e l$ for every l. Moreover, if l, h, j are lists and $l \leq_e h$, then $l + j \leq_e h + j$.*

Proof. The first claim is trivial. To prove the second, we proceed by an induction on j. If $j = empty$, then $l + j = l \leq_e h = h + j$. Now, suppose $j = cons(n, g)$. If $h = empty$, then $l = empty$ and, clearly $l + j = j \leq_e j = h + j$. If $l = empty$, we have to prove that $j \leq_e h + j$. Let $h = cons(m, f)$; then

$$\begin{array}{rcl} n & \leq & n + m \leq 2^e(n + m + e) - 0 \\ g & \leq_0 & g + f \end{array}$$

which means $j \leq_e h + j$. Finally, suppose $l = cons(m, f)$, $h = cons(p, r)$. Then we know that

$$\begin{array}{rcl} m & \leq & 2^e(p + e) - d \\ f & \leq_d & r \end{array}$$

But then, by inductive hypothesis,

$$\begin{array}{rcl} m + n & \leq & 2^e(p + e) + n - d \leq 2^e(p + n + e) - d \\ f + g & \leq_d & r + g \end{array}$$

which yields $l + j \leq_e h + j$. □

Lemma 7 (Transitivity) *If l, h, j are lists and $l \leq_e h$, $h \leq_d j$, then $l \leq_{d+e} j$.*

Proof. We can suppose all the involved lists to be different from $empty$, since all the other cases are trivial. $l = cons(n, g)$, $h = cons(m, f)$ and $j = cons(p, r)$. From the hypothesis, we have

$$\begin{array}{rcl} n & \leq & 2^e(m+e) - c \\ m & \leq & 2^d(p+d) - b \\ g & \leq_c & f \\ f & \leq_b & r \end{array}$$

But then, by inductive hypothesis, we get

$$\begin{array}{rcl} n & \leq & 2^e(m+e) - c \leq 2^e(2^d(p+d) - b + e) - c \leq 2^e 2^d(p+d+e) - b - c = 2^{e+d}(p+d+e) - (b+c) \\ g & \leq_{c+b} & r \end{array}$$

This means $l \leq_{d+e} j$. □

Lemma 8 *if l, h, j are lists and $l \leq_e h$, then $\mathcal{D}_e(l, h) \leq \mathcal{D}_e(l + j, h + j)$*

Proof. We proceed by an induction on j. If $j = empty$, then $l + j = l$ and $h + j = h$. Now, suppose $j = cons(n, g)$. If $h = empty$, then $l = empty$ and, clearly $l + j = j = h + j$. If $l = empty$, let $h = cons(m, f)$; then

$$\begin{array}{rcl} \mathcal{D}_e(l, h) & = & \mathcal{D}_e(empty, h) = 2^e(m+e) + \mathcal{D}_{2^e(m+e)}(empty, f) \\ & \leq & 2^e(m+e) + 2^e n - 2^e n + \mathcal{D}_{2^e(m+e)+2^e n-2^e n}(g, g+f) \\ & \leq & 2^e(m+n+e) - n + \mathcal{D}_{2^e(m+n+e)-n}(g, g+f) \\ & = & \mathcal{D}_e(j, h+j) = \mathcal{D}_e(l+j, h+j) \end{array}$$

Finally, suppose $l = cons(m, f)$, $h = cons(p, r)$. Then we know that

$$\begin{array}{rcl} \mathcal{D}_e(l, h) & = & 2^e(m+e) - p + \mathcal{D}_{2^e(m+e)-p}(f, r) \\ & \leq & 2^e(m+e) - p + \mathcal{D}_{2^e(m+e)-p}(f+g, r+g) \\ & \leq & 2^e(m+e) + 2^e n - (n+p) + \mathcal{D}_{2^e(m+e)+2^e n-n-p}(f+g, r+g) \\ & = & 2^e(m+n+e) - (n+p) + \mathcal{D}_{2^e(m+n+e)-(n+p)}(f+g, r+g) \\ & = & \mathcal{D}_e(l+j, h+j) \end{array}$$

Lemma 9 *If l, h, j are lists and $l \leq_e h$, $h \leq_d j$, then $\mathcal{D}_e(l, h) + \mathcal{D}_d(h, j) \leq \mathcal{D}_{e+d}(l, j)$.*

Proof. If either $h = empty$ or $j = empty$, then the thesis is trivial. So suppose $h = cons(n, g)$ and $j = cons(m, f)$. If $l = empty$, then

$$\begin{array}{rcl} \mathcal{D}_e(l, h) + \mathcal{D}_d(h, j) & = & 2^e(n+e) + \mathcal{D}_{2^e(n+e)}(empty, g) + 2^d(m+d) - n + \mathcal{D}_{2^e(m+d)-n}(g, f) \\ & \leq & 2^e(n+e) + 2^d(m+d) - n + \mathcal{D}_{2^e(n+e)+2^d(m+d)-n}(empty, f) \\ & \leq & (2^e-1)n + 2^e e + 2^d(m+d) + \mathcal{D}_{(2^e-1)n++2^e e+2^d(m+d)}(empty, f) \\ & \leq & (2^e-1)2^d(m+d) + 2^e e + 2^d(m+d) + \mathcal{D}_{(2^e-1)2^d(m+d)+2^e e+2^d(m+d)}(empty, f) \\ & = & 2^{d+e}(m+d+e) + \mathcal{D}_{2^{d+e}(m+d+e)}(empty, f) \\ & = & \mathcal{D}_{e+d}(l, j) \end{array}$$

If $l = cons(p, r)$, then

$$\begin{array}{rcl} \mathcal{D}_e(l, h) + \mathcal{D}_d(h, j) & = & 2^e(n+e) - p + \mathcal{D}_{2^e(n+e)-p}(r, g) + 2^d(m+d) - n + \mathcal{D}_{2^d(m+d)-n}(g, f) \\ & \leq & 2^e(n+e) - p + 2^d(m+d) - n + \mathcal{D}_{2^e(n+e)-p+2^d(m+d)-n}(r, f) \\ & \leq & (2^e-1)n + 2^e e + 2^d(m+d) - p + \mathcal{D}_{(2^e-1)n+2^e e+2^d(m+d)-p}(r, f) \\ & \leq & (2^e-1)2^d(m+d) + 2^e e + 2^d(m+d) - p + \mathcal{D}_{(2^e-1)2^d(m+d)+2^e e+2^d(m+d)-p}(r, f) \\ & = & 2^{d+e}(m+d+e) - p + \mathcal{D}_{2^{d+e}(m+d+e)-p}(r, f) \\ & = & \mathcal{D}_{e+d}(l, j) \end{array}$$

This concludes the proof. □

$|\mathcal{L}|$ will denote the set of all lists, while $\leq_{\mathcal{L}}, \mathcal{D}_{\mathcal{L}}$ will denote $\leq_0$ and $\mathcal{D}_0$, respectively.

Lemma 10 *$\mathcal{L} = (|\mathcal{L}|, +, \leq_{\mathcal{L}}, \mathcal{D}_{\mathcal{L}})$ is a resource monoid.*

Proof. $(\mathcal{L}, +)$ is certainly a monoid. Compatibility of $\leq_{\mathcal{L}}$ follows from lemmas 6 and 7. The two required property on $\mathcal{D}_{\mathcal{L}}$ come directly from lemmas 8 and 9. If $n \in \mathbb{N}$, observe that $\mathcal{F}_{\mathcal{L}}(cons(n, empty)) = n$. This concludes the proof. □

An *elementary length space* is a length space on the resource monoid $(\mathcal{L}, +, \leq_{\mathcal{L}}, \mathcal{D}_{\mathcal{L}})$. Given an elementary length space $A = (|A|, \Vdash_A)$, we can build the length space $!A = (|A|, \Vdash_{!A})$, where $e, l \Vdash_{!A} a$ iff $e, h \Vdash_{!A} a$ and $l \geq_{\mathcal{L}} !h$. The construction ! on elementary length spaces serves to capture the exponential modality of elementary affine logic. Indeed, the following two results prove the existence of morphisms and morphisms-forming rules precisely corresponding to axioms and rules from EAL.

Lemma 11 (Basic Maps) *Given elementary length spaces A, B, there are morphisms:*

$$\begin{array}{rcl} contr & : & !A \multimap !A \otimes !A \\ distr & : & !A \otimes !B \multimap !(A \otimes B) \end{array}$$

where $contr(a) = (a, a)$ and $distr(a, b) = (a, b)$

Proof. We know $\{e_{contr}\}(d)$ takes time linear in $|d|$, say at most $p|d| + q$. Then, let $l, h \in \mathcal{L}$ be such that $\mathcal{F}_{\mathcal{L}}(l) \geq p + q + |e_{contr}|$, $\mathcal{F}_{\mathcal{L}}(h) \geq cp$. Define l_{contr} to be $l + h + [1]_{\mathcal{L}}$. Clearly, $\mathcal{F}_{\mathcal{L}}(l_{contr}) \geq |e_{contr}|$ Now, let $j, d \Vdash_{!A} a$. This means that $j \geq_{\mathcal{L}} !k$ where $k, d \Vdash_A a$. Then:

$$\begin{array}{rcl} h{+}!k{+}!k & \geq_{\mathcal{L}} & !k{+}!k \\ \mathcal{F}_{\mathcal{L}}(h{+}!k{+}!k) & \geq & \mathcal{F}_{\mathcal{L}}(h) + \mathcal{F}_{\mathcal{L}}(!k) + \mathcal{F}_{\mathcal{L}}(!k) \\ & \geq & cp + \mathcal{F}_{\mathcal{L}}(!k) + \mathcal{F}_{\mathcal{L}}(!k) \end{array}$$

This means that $h{+}!k{+}!k, e \Vdash_{!A \otimes !A} (a, a)$. Moreover, $h{+}!k{+}!k \leq_{\mathcal{L}} h{+}!k + [1]_{\mathcal{L}} \leq_{\mathcal{L}} h{+}!j + l_{contr}$. Finally,

$$\begin{array}{rcl} Time(\{e_{contr}\}(d)) & \leq & p|d| + q \leq (p+q)(|d| + p + q) \\ & \leq & \mathcal{F}_{\mathcal{L}}(l_{contr})(\mathcal{F}_{\mathcal{L}}(j) + \mathcal{F}_{\mathcal{L}}(l_{contr})) \\ & \leq & (\mathcal{F}_{\mathcal{L}}(l_{contr}) + \mathcal{D}_{\mathcal{L}}(j, j))(\mathcal{F}_{\mathcal{L}}(l_{contr}) + \mathcal{F}_{\mathcal{L}}(j)) \\ & \leq & \mathcal{D}_{\mathcal{L}}(j, j + l_{contr})\mathcal{F}_{\mathcal{L}}(l_{contr} + j) \end{array}$$

This proves $contr$ to be a morphism.

Let $e_{distr} = e_{id}$. We know $\{e_{id}\}(d)$ takes time linear in $|d|$, say at most $p|d| + q$. Then, let $l, h \in \mathcal{L}$ be such that $\mathcal{F}_{\mathcal{L}}(l) \geq p + q + |e_{distr}|$, $\mathcal{F}_{\mathcal{L}}(h) \geq cp$. l_{distr} is then defined as $l{+}!h$. Now, let $j, \langle d, c\rangle \Vdash_{!A \otimes !B} (a, b)$. This means that $j \geq !k{+}!i$, where $k, d \Vdash_A a$ and $i, c \Vdash_B b$. This in turn means that $k + i + h, \langle d, c\rangle \Vdash_{A \otimes B} (a, b)$ and $!(k + i + h), \langle d, c\rangle \Vdash_{!A \otimes !B} (a, b)$. Moreover

$$!(k + i + h) = !k{+}!i{+}!h \leq_{\mathcal{L}} j + l_{distr}$$

Finally:

$$\begin{array}{rcl} Time(\{e_{distr}\}(\langle d, c\rangle)) & \leq & p|\langle d, c\rangle| + q \leq (p+q)(|\langle d, c\rangle| + p + q) \leq \mathcal{F}_{\mathcal{L}}(l_{distr})(\mathcal{F}_{\mathcal{L}}(j) + \mathcal{F}_{\mathcal{L}}(l_{distr}) \\ & \leq & (\mathcal{F}_{\mathcal{L}}(l_{distr}) + \mathcal{D}_{\mathcal{L}}(j, j))(\mathcal{F}_{\mathcal{L}}(l_{distr}) + \mathcal{F}_{\mathcal{L}}(l)) \\ & \leq & \mathcal{D}_{\mathcal{L}}(j, j + l_{distr})\mathcal{F}_{\mathcal{L}}(l_{distr} + j) \end{array}$$

This proves $distr$ to be a morphism. □

Lemma 12 (Functoriality) *If $f : A \xrightarrow{e,\varphi} B$, then there are d, ψ such that $f : !A \xrightarrow{e,\psi} !B$*

Exponential Rules and Contraction.

$$\frac{\Gamma \vdash A}{!\Gamma \vdash !A}\ P \qquad \frac{\Gamma, !A, !A \vdash B}{\Gamma, !A \vdash B}\ C$$

Figure 2. Intuitionistic Elementary Affine Logic

Proof. Now, let θ be $!\varphi$ and suppose $d, l \Vdash_{!A} a$. Then $l \geq_{!} h$, where $d, h \Vdash_A a$. Observe that there must be j, c such that $c, j \Vdash_B f(a)$, $j \leq_{\mathcal{L}} h + \varphi$ and $Time(\{e\}(d)) \leq \mathcal{F}_{\mathcal{L}}(h+\varphi)\mathcal{D}_{\mathcal{L}}(j, h+\varphi)$. But then $c, !j \Vdash_{!B} f(a)$ and, moreover

$$\begin{array}{rcl} !j & \leq_{\mathcal{L}} & !(h+\varphi) = !h + !\varphi \leq_{\mathcal{L}} !h + \theta \\ Time(\{e\}(d)) & \leq & \mathcal{F}_{\mathcal{L}}(h+\varphi)\mathcal{D}_{\mathcal{L}}(j, h+\varphi) \\ & \leq & \mathcal{F}_{\mathcal{L}}(!(h+\varphi))\mathcal{D}_{\mathcal{L}}(!j, !(h+\varphi)) \\ & \leq & \mathcal{F}_{\mathcal{L}}(!h + !\varphi)\mathcal{D}_{\mathcal{L}}(!j, !h + !\varphi)) \\ & \leq & \mathcal{F}_{\mathcal{L}}(l+\theta)\mathcal{D}_{\mathcal{L}}(!j, l+\theta) \end{array}$$

This means that $f :!A \xrightarrow{e,\theta} !B$. □

Elementary bounds can be given on $\mathcal{F}_{\mathcal{L}}(l)$ depending on $|l|$ and $depth(l)$:

Proposition 1 *For every $n \in \mathbb{N}$ there is an elementary function $p_n : \mathbb{N} \to \mathbb{N}$ such that $\mathcal{F}_{\mathcal{L}}(l) \leq p_{depth(l)}(|l|)$.*

Proof. We prove a stronger statement by induction on n: for every $n \in \mathbb{N}$ there is an elementary function $q_n : \mathbb{N}^2 \to \mathbb{N}$ such that for every l, e, $\mathcal{D}_e(empty, l) \leq q_{depth(l)}(|l|, e)$. First of all, we know that $\mathcal{D}_e(empty, empty) = 0$, so q_0 is just the function which always returns 0. q_{n+1} is defined from q_n as follows: $q_{n+1}(x, y) = 2^y(x+y) + q_n(x, 2^y(x+y))$. Indeed:

$$\begin{array}{rcl} \mathcal{D}_e(empty, cons(n, l)) & = & 2^e(n+e) + \mathcal{D}_{2^e(n+e)}(empty, l) \\ & \leq & 2^e(|cons(n,l)| + e) + q_{depth(l)}(|cons(n,l)|, 2^e|cons(n,l)| + e) \\ & = & q_{depth(cons(n,l))}(|cons(n,l)|, e) \end{array}$$

At this point we just put $p_n(x) = q_n(x, 0)$. □

4.1 Interpreting Elementary Affine Logic

EAL can be obtained by endowing multiplicative affine logic with a restricted modality. The grammar of formulae is enriched with a new production $A ::= !A$ while modal rules are reported in figure 2. Realizability semantics is extended by $[\![!A]\!]^{\mathscr{R}}_{\eta} = !\,[\![A]\!]^{\mathscr{R}}_{\eta}$.

Theorem 1 *Elementary length spaces form a model of* EAL.

Now, consider the formula

$$List_{\mathsf{EAL}} \equiv \forall\alpha.!(\alpha \multimap \alpha) \multimap !(\alpha \multimap \alpha) \multimap !(\alpha \multimap \alpha)$$

Binary lists can be represented as cut-free proofs with conclusion $List_{\mathsf{EAL}}$. Suppose you have a proof $\pi :!^j List_{\mathsf{EAL}} \multimap !^k List_{\mathsf{EAL}}$. From the denotation $[\![\pi]\!]^{\mathscr{R}}$ we can build a morphism from $[\![List_{\mathsf{EAL}}]\!]^{\mathscr{R}}$ to L_M by internal application to ε, s_0, s_1. This map then induces a function $f : L \to L$ as follows: given a list $w \in L$ first compute a realizer for the closed proof corresponding to it, then apply p to the result.

Corollary 1 (Soundness) *Let π be an* EAL *proof with conclusion $\vdash !^j List_{\mathsf{EAL}} \multimap !^k List_{\mathsf{EAL}}$ and let $f : L \to L$ be the function induced by $[\![\pi]\!]^{\mathscr{R}}$. Then f is computable in elementary time.*

The function f in the previous result equals the function denoted by the proof π in the sense of [8]. This intuitively obvious fact can be proved straightforwardly but somewhat tediously using a logical relation or similar, see also [8].

Exponential Rules and Contraction.

$$\frac{\Gamma \vdash A}{!\Gamma \vdash !A}\ P \qquad \frac{\Gamma, A, \ldots, A \vdash B}{\Gamma, !A \vdash B}\ C$$

Figure 3. Intuitionistic Soft Affine Logic

5 Soft Length Spaces

The grammar of formulae for SAL is the same as the one of Elementary Affine Logic. Rules are reported in figure 3. We here use a resource monoid whose underlying carrier set is $|\mathcal{I}| = |\mathcal{L}| \times \mathbb{N}$. The sum $(l, n) + (h, m)$ of two elements in $|\mathcal{I}|$ is defined as $(l + h, \max\{n, m\})$. For every $e \in \mathbb{N}$, binary relations $\leq_e$ on $|\mathcal{I}|$ can be defined as follows

- $(empty, n) \leq_0 (empty, m)$ iff $n \leq m$;
- $(empty, n) \leq_e (cons(m, l), p)$ iff there is $d \in \mathbb{N}$ such that
  1. $e \leq m + pd$
  2. $(empty, n) \leq_d (l, p)$
- $(cons(n, l), m) \leq_e (cons(p, h), q)$ iff there is $d \in \mathbb{N}$ such that
  1. $e + n \leq p + qd$;
  2. $(l, m) \leq_d (h, q)$.

If $\alpha = (l, n) \in |\mathcal{I}|$, then $!\alpha$ will be the couple $(cons(0, l), n) \in |\mathcal{I}|$. If there is e such that $\alpha \leq_e \beta$, then we will simply write $\alpha \leq_{\mathcal{I}} \beta$. For every α and β with $\alpha \leq_{\mathcal{I}} \beta$, we define the natural number $\mathcal{D}_{\mathcal{I}}(\alpha, \beta)$ as follows:

$$\begin{aligned}
\mathcal{D}_{\mathcal{I}}((empty, n), (empty, m)) &= 0 \\
\mathcal{D}_{\mathcal{I}}((empty, n), (cons(m, l), p)) &= m + p\mathcal{D}_{\mathcal{I}}((empty, n), (l, p)) \\
\mathcal{D}_{\mathcal{I}}((cons(n, l), m), (cons(p, h), q)) &= n - p + q\mathcal{D}_{\mathcal{I}}((l, m), (h, q))
\end{aligned}$$

Analogously, we can define $\mathcal{D}_{\mathcal{I}}(\alpha, \beta)$ simply as the maximum integer e such that $\alpha \leq_e \beta$. $|\alpha|$ is the maximum integer appearing inside α, i.e. $|(l, n)| = \max\{|l|, m\}$. The depth $depth(\alpha)$ of $\alpha = (l, n)$ is $depth(l)$.

Lemma 13 (Compatibility) *$(empty, 0) \leq_0 \alpha$ for every α. Moreover, if $\alpha, \beta, \gamma \in |\mathcal{I}|$ and $\alpha \leq_e \beta$, then $\alpha + \gamma \leq_e \beta + \gamma$.*

Proof. The first claim is trivial. To prove the second, we proceed by an induction on the structure of the first component of γ. We just consider the case where the first components of α, β, γ are all different from $empty$. So, suppose $\alpha = (cons(n, l), m)$, $\beta = (cons(p, h), q)$, $\gamma = (cons(r, j), s)$. By hypothesis, we get

$$\begin{aligned}
n &\leq p + dq - e \\
(l, m) &\leq_d (h, q)
\end{aligned}$$

Then, $n+r \leq p+r+dq-e \leq p+r+d\max\{q, s\}-e$ and, by induction hypothesis, $(l+j, \max\{m, s\}) \leq_d (h+j, \max\{q, s\})$. This means that $\alpha + \gamma \leq_e \beta + \gamma$. □

Lemma 14 (Transitivity) *If $\alpha, \beta, \gamma \in |\mathcal{I}|$ are lists and $\alpha \leq_e \beta$, $\beta \leq_d \gamma$, then $\alpha \leq_{d+e} \gamma$.*

Proof. We go by induction on the structure of the first component of γ and we suppose the first components of α, β, γ to be different from $empty$. So, let $\alpha = (cons(n, l), m)$, $\beta = (cons(p, h), q)$ and $\gamma = (cons(r, j), s)$. From the hypothesis, we have

$$\begin{aligned}
n &\leq p + cq - e \\
p &\leq r + bs - d \\
(l, m) &\leq_c (h, q) \\
(h, q) &\leq_b (j, s)
\end{aligned}$$

But then, by inductive hypothesis, we get

$$\begin{array}{rcl} n & \leq & r + bs - d + cq - e \leq r + (c+b)s + (e+d) \\ (l,m) & \leq_{c+b} & (j,s) \end{array}$$

which yields $\alpha \leq_{d+e} \gamma$. □

Lemma 15 *if $\alpha, \beta, \gamma \in \mathcal{I}$ and $\alpha \leq_e \beta$, then $\mathcal{D}_{\mathcal{I}}(\alpha, \beta) \leq \mathcal{D}_{\mathcal{I}}(\alpha + \gamma, \beta + \gamma)$*

Proof. This is trivial in view of 13 and the fact that $\mathcal{D}_{\mathcal{I}}(\alpha, \beta)$ is just $\max\{e \in \mathbb{N} \mid \alpha \leq_e \beta\}$. □

Lemma 16 *If $\alpha, \beta, \gamma \in \mathcal{I}$ and $\alpha \leq_e \beta$, $\beta \leq_d \gamma$, then $\mathcal{D}_e(\alpha, \beta) + \mathcal{D}_d(\beta, \gamma) \leq \mathcal{D}_{e+d}(\alpha, \gamma)$.*

Proof. This is trivial in view of 14 and the fact that $\mathcal{D}_{\mathcal{I}}(\alpha, \beta)$ is just $\max\{e \in \mathbb{N} \mid \alpha \leq_e \beta\}$. □

Lemma 17 *$(\mathcal{I}, +, \leq_{\mathcal{I}}, \mathcal{D}_{\mathcal{I}})$ is a resource monoid.*

Proof. $(|\mathcal{I}|, +)$ is certainly a commutative monoid. Compatibility of $\leq_{\mathcal{I}}$ follows from lemmas 13 and 14. The two required property on $\mathcal{D}_{\mathcal{I}}$ come directly from lemmas 15 and 16. If $n \in \mathbb{N}$, observe that $\mathcal{F}_{\mathcal{I}}((cons(n, empty), 0)) = n$. This concludes the proof. □

A *soft length space* is a length space on the resource monoid $(\mathcal{I}, +, \leq_{\mathcal{I}}, \mathcal{D}_{\mathcal{I}})$.

Given a soft length space $A = (|A|, \Vdash_A)$, we can build the length space $!A = (|A|, \Vdash_{!A})$, where $e, \alpha \Vdash_{!A} a$ iff $e, \beta \Vdash_{!A} a$ and $\alpha \geq_{\mathcal{I}} !\beta$.

Lemma 18 (Basic Maps) *Given soft length spaces A, B and a natural number n, there are morphisms:*

$$\begin{array}{rcl} contr & : & !A \to \overbrace{A \otimes \ldots \otimes A}^{n \text{ times}} \\ distr & : & !A \otimes !B \to !(A \otimes B) \end{array}$$

where $contr(a) = (a, \ldots, a)$ and $distr(a, b) = (a, b)$

Lemma 19 (Functoriality) *If $f : A \xrightarrow{e,\varphi} B$, then there are d, ψ such that $f : !A \xrightarrow{d,\psi} !B$*

Proposition 2 *For every $n \in \mathbb{N}$ there is a polynomial $p_n : \mathbb{N} \to \mathbb{N}$ such that $\mathcal{F}_{\mathcal{I}}(\alpha) \leq p_{depth(\alpha)}(|\alpha|)$.*

Proof. We go by induction on n. First of all, we know that $\mathcal{D}_{\mathcal{I}}((empty, 0), (empty, m)) = 0$, so p_0 is just the function which always returns 0. p_{n+1} is defined from p_n as follows: $p_{n+1}(x) = x + xp_n(x)$. Indeed:

$$\begin{array}{rcl} \mathcal{D}_{\mathcal{I}}((empty, 0), (cons(n, l), m)) & = & n + m\mathcal{D}_{\mathcal{I}}((empty, 0), (l, m)) \\ & \leq & |(cons(n, l), m)| + |(cons(n, l), m)| p_{depth((l,m))}(|(cons(n, l), m)|) \\ & = & p_{depth((cons(n,l),m))}((cons(n, l), m)). \end{array}$$

This concludes the proof. □

Theorem 2 *Soft length spaces form a model of* SAL.

Binary lists can be represented in SAL as cut-free proofs with conclusion

$$List_{\mathsf{SAL}} \equiv \forall\alpha.!(\alpha \multimap \alpha) \multimap (\alpha \multimap \alpha) \multimap (\alpha \multimap \alpha)$$

Corollary 2 (Soundness) *Let π be an* SAL *proof with conclusion $\vdash !^j List_{\mathsf{LAL}} \multimap !^k List_{\mathsf{LAL}}$ and let $f : L \to L$ be the function induced by $[\![\pi]\!]^{\mathscr{R}}$. Then f is computable in polynomial time.*

6 Light Length Spaces

A *tree* is either $empty$ or a triple $node(n,t,T)$ where $n \in \mathbb{N}$, t is itself a tree and T is a finite nonempty set of trees. We write $[n]_{\mathcal{T}}$ for the tree defined by $[n]_{\mathcal{T}} = node(n, empty, \{empty\})$. The sum $t+s$ of two trees t and s is defined as follows, by induction on n:

$$\begin{array}{rcl} empty + t & = & t + empty = t; \\ node(n,t,T) + node(n,u,U) & = & node(n+m, t+u, T \cup U); \end{array}$$

Here, more sophisticated techniques are needed. For every $n, e \in \mathbb{N}$, binary relations $\leq_e^n$ on trees can be defined as follows

- $t \leq_e^0 u$
- $empty \leq_e^{n+1} t$;
- $node(m,t,T) \leq_e^{n+1} empty$ iff there is $d \in \mathbb{N}$ such that
  1. $m \leq e - d$;
  2. $t \leq_{d^2}^n empty$;
  3. For every $s \in T$, $s \leq_d^n empty$.
- $node(m,t,T) \leq_e^{n+1} node(l,u,U)$ iff there is $d \in \mathbb{N}$ such that
  1. $m \leq l + e - d$;
  2. There is a function $f : \{1, \ldots, d\} \to U$ such that $t \leq_{d^2}^n u + \sum_1^d f(i)$;
  3. For every $s \in T$ there is $z \in U$ with $s \leq_d^n z$.

For every e, n and for every trees t and u with $t \leq_e^n u$, we define the natural number $\mathcal{D}_e^n(t,u)$ as follows:

$$\begin{array}{rcl} \mathcal{D}_e^0(t,u) & = & 0 \\ \mathcal{D}_e^{n+1}(empty, empty) & = & e + \mathcal{D}_e^n(empty, empty) \\ \mathcal{D}_e^{n+1}(empty, node(m,t,T)) & = & m + e + \max_f\{\mathcal{D}_{(m+e)^2}^n(empty, t + \sum_{i=1}^{m+e} f(i))\} \\ \mathcal{D}_e^{n+1}(node(m,t,T), empty) & = & e - m + \mathcal{D}_{(e-m)^2}^n(t, empty) \\ \mathcal{D}_e^{n+1}(node(m,t,T), node(l,u,U)) & = & l + e - m + \max_f\{\mathcal{D}_{(l+e-m)^2}^n(t, u + \sum_{i=1}^{l+e-m} f(i))\} \end{array}$$

If t is a tree, then $|t|$ is the greatest integer appearing in t, i.e. $|empty| = 0$ and $|node(n,t,T)| = \max\{n, |t|, \max_{u \in T} |u|\}$.

The depth $depth(t)$ of a tree t is defined as follows: $depth(empty) = 0$ and

$$depth(node(n,t,T)) = 1 + \max\{depth(t), \max_{u \in T} depth(u)\}.$$

Given a tree $t \in \mathcal{T}$, we define $!t$ as the tree $node(0, empty, \{empty, t\})$ and $\S t$ as the tree $node(0, t, \{empty\})$. In this context, a notion of isomorphism between trees is needed: we say that trees t and u are *ismorphic* and we write $t \cong u$ iff for every $e, n \in \mathbb{N}$ and for every tree v the following hold:

$$\begin{array}{rcl} v \leq_e^n t & \Leftrightarrow & v \leq_e^n u \\ t \leq_e^n v & \Leftrightarrow & u \leq_e^n v \\ \mathcal{D}_e^n(v,t) & = & \mathcal{D}_e^n(v,u) \\ \mathcal{D}_e^n(t,v) & = & \mathcal{D}_e^n(t,u) \end{array}$$

Lemma 20 *$empty \cong [0]_{\mathcal{T}}$. Moreover, for every tree t, $t + empty \cong t + [0]_{\mathcal{T}}$.*

Proof. We go by induction on n, considering the case where $n \geq 1$, since the base case is trivial. First of all, observe that both $empty \leq_e^{n+1} t$ and $[0]_{\mathcal{T}} \leq_e^{n+1} t$ for every t. Moreover, $empty \leq_e^{n+1} empty$ and $[0]_{\mathcal{T}} \leq_e^{n+1} empty$. Suppose now that $node(m,t,T) \leq_e^{n+1} empty$. This means that there is d such that

1. $m \leq e - d$,
2. $t \leq_{d^2}^n empty$;
3. Forall $s \in T$, $s \leq_d^n empty$.

If we put $f(i) = empty$ for every i, we get $t \leq_{d^2}^n empty + \sum_{i=1}^{d} f(i)$, which yields $node(m,t,T) \leq_e^{n+1} [0]_{\mathcal{T}}$. In the same way, we can prove that if $node(m,t,T) \leq_e^{n+1} [0]_{\mathcal{T}}$, then $node(m,t,T) \leq_e^{n+1} empty$.

We have:

$$\begin{array}{rcl}
\mathcal{D}_e^{n+1}(empty, empty) & = & e + \mathcal{D}_{e^2}^n(empty, empty) \\
\mathcal{D}_e^{n+1}(empty, [0]_{\mathcal{T}}) & = & e + \mathcal{D}_{e^2}^n(empty, empty) \\
\mathcal{D}_e^{n+1}([0]_{\mathcal{T}}, empty) & = & e + \mathcal{D}_{e^2}^n(empty, empty) \\
\mathcal{D}_e^{n+1}(empty, node(m,t,T)) & = & m + e + \max_f\{\mathcal{D}_{(m+e)^2}^n(empty, t + \sum_{i=1}^{m+e} f(i))\} \\
 & = & \mathcal{D}_e^{n+1}([0]_{\mathcal{T}}, node(m,t,T)) \\
\mathcal{D}_e^{n+1}(node(m,t,T), empty) & = & e - m + \mathcal{D}_{(e-m)^2}^n(t, empty) \\
 & = & \mathcal{D}_e^{n+1}(node(m,t,T), [0]_{\mathcal{T}})
\end{array}$$

Moreover, observe that

$$\begin{array}{rcl}
empty + empty = empty & \cong & [0]_{\mathcal{T}} = [0]_{\mathcal{T}} + empty \\
node(m,t,T) + empty & = & node(m,t,T) + [0]_{\mathcal{T}}
\end{array}$$

This concludes the proof. □

Proposition 3 (Compatibility) *For every $n, e \in \mathbb{N}$, $empty \leq_e^n t$ for every t and, moreover, if $t \leq_e^n u$ then $t + v \leq_e^n u + v$ for every t, u, v.*

Proof. $empty \leq_e^n t$ is trivial. The second statement can be proved by induction on n. The base case is trivial. In the inductive case, we can suppose all the involved trees to be different from $empty$. Suppose that $node(m,t,T) \leq_e^{n+1} node(l,u,U)$. We should prove $node(m+k, t+v, T \cup V) \leq_e^{n+1} node(l+k, u+v, U \cup V)$. However,

$$\begin{array}{rcl}
m + k & \leq & (l + e) - d + k = (l + k + e) - d \\
t + v & \leq_{d^2}^n & u + \sum_{i=1}^{d} f(i) + v = u + v + \sum_{i=1}^{d} f(i)
\end{array}$$

Moreover, for every $z \in T \cup V$ there certainly exists $w \in U \cup V$ such that $z \leq_d^n w$. □

Proposition 4 (Transitivity) *If $t \leq_e^n u \leq_d^n v$, then $t \leq_{d+e}^n v$.*

Proof. We go by induction on n. We can directly go to the inductive case, since if $n = 0$, then the thesis is trivial. We can assume all the involved trees to be different from $empty$. Let us suppose $node(m,t,T) \leq_e^{n+1} node(l,u,U)$ and $node(l,u,U) \leq_d^{n+1} node(k,v,V)$ First of all, we have $m \leq l + e - c$ and $l \leq k + d - b$, which yields $m \leq k + d - b + e - c = k + (d + e) - (b + c)$. Moreover, by hypothesis, there are functions $f : \{1, \ldots, d\} \to U$ and $g : \{1, \ldots, e\} \to V$ such that

$$\begin{array}{rcl}
t & \leq_{c^2}^n & u + \sum_{i=1}^{c} f(i) \\
u & \leq_{b^2}^n & v + \sum_{i=1}^{b} g(i)
\end{array}$$

Therefore, by inductive hypothesis and by proposition 3:

$$\begin{aligned} t &\leq^n_{c^2+b^2} v + \sum_{i=1}^{c} f(i) + \sum_{i=1}^{b} g(i) \\ &\leq^n_{bc} v + \sum_{i=1}^{c} f(i) + \sum_{i=1}^{b} h(i) \end{aligned}$$

where $h : \{1, \ldots, d\} \to V$. We can then find a function $k : \{1, \ldots, d+e\} \to V$ such that

$$t \leq^n_{(c+b)^2} v + \sum_{i=1}^{c+b} k(i).$$

Finally, if $z \in T$ then we find $w \in U$ such that $z \leq^n_c w$. We then find $x \in V$ such that $w \leq^n_b x$ and so $z \leq^n_{c+b} x$. □

Proposition 5 *For every n, e and for every t, u, v, $\mathcal{D}^n_e(t, u) \leq \mathcal{D}^n_e(t + v, u + v)$*

Proof. We can proceed by induction on n and, again, the case $n = 0$ is trivial. In the inductive case, as usual, we can suppose all the involved trees to be different from $empty$. We have

$$\begin{aligned} & \mathcal{D}^{n+1}_e(node(m, t, T), node(l, u, U)) \\ = \;& l + e - m + \max_f \{\mathcal{D}^n_{(l+e-m)^2}(t, u + \sum_{i=1}^{l+e-m} f(i))\} \\ = \;& l + e - m + \mathcal{D}^n_{(l+e-m)^2}(t, u + \sum_{i=1}^{l+e-m} f(i)) \end{aligned}$$

where f and realizes the max. By induction hypothesis,

$$\begin{aligned} & \mathcal{D}^{n+1}_e(node(m, t, T), node(l, u, U)) \\ \leq \;& (l + k) + e - (m + k) + \mathcal{D}^n_{((l+k)+e-(m+k))^2}(t + v, u + v + \sum_{i=1}^{(l+k)+e-(m+k)} f(i)) \\ \leq \;& \mathcal{D}^{n+1}_e(node(m, t, T) + node(k, v, V), node(l, u, U) + node(k, v, V)) \end{aligned}$$

This concludes the proof. □

Proposition 6 $\mathcal{D}^n_e(t, u) + \mathcal{D}^n_d(u, v) \leq \mathcal{D}^n_{e+d}(t, v)$

Proof. We can proceed by induction on n and, again, the case $n = 0$ is trivial. In the inductive case, as usual, we can suppose

all the involved trees to be different from $empty$. Now

$$\begin{aligned}
& \mathcal{D}_e^{n+1}(node(m,t,T),node(l,u,U)) + \mathcal{D}_d^{n+1}(node(l,u,U),node(k,v,V)) \\
= \;& l+e-m+\max_f\{\mathcal{D}^n_{(l+e-m)^2}(t,u+\sum_{i=1}^{l+e-m} f(i))\} \\
& +k+d-l+\max_g\{\mathcal{D}^n_{(n+d-l)^2}(u,v+\sum_{i=1}^{k+d-l} g(i))\} \\
= \;& k+(e+d)-m+\mathcal{D}^n_{(l+e-m)^2}(t,u+\sum_{i=1}^{l+e-m} f(i)) \\
& +\mathcal{D}^n_{(k+d-l)^2}(u,v+\sum_{i=1}^{k+d-l} g(i)) \\
= \;& k+(e+d)-m+\mathcal{D}^n_{(l+e-m)^2}(t,u+\sum_{i=1}^{l+e-m} f(i)) \\
& +\mathcal{D}^n_{(k+d-l)^2}(u+\sum_{i=1}^{l+e-m} f(i),v+\sum_{i=1}^{k+d-l} g(i)+\sum_{i=1}^{l+e-m} f(i)) \\
\leq \;& k+(e+d)-m+\mathcal{D}^n_{(l+e-m)^2+(k+d-l)^2}(t,v+\sum_{i=1}^{k+d-l} g(i)+\sum_{i=1}^{l+e-m} f(i))
\end{aligned}$$

A function $h:\{1,\ldots,l+e-m\}\to V$ such that $\sum_{i=1}^{l+e-m} f(i) \leq^n_{(l+e-m)(k+d+l)} \sum_{i=1}^{l+e-m} h(i)$ can be easily defined, once we remember that $node(l,u,U) \leq^n_d node(k,v,V)$. This yields

$$\begin{aligned}
& \mathcal{D}_e^{n+1}(node(m,t,T),node(l,u,U)) + \mathcal{D}_d^{n+1}(node(l,u,U),node(k,v,V)) \\
\leq \;& k+(e+d)-m+\mathcal{D}^n_{(l+e-m)^2+(k+d-l)^2}(t,v+\sum_{i=1}^{k+d-l} g(i)+\sum_{i=1}^{l+e-m} f(i)) \\
& +\mathcal{D}^n_{(l+e-m)(k+d-l)}(v+\sum_{i=1}^{k+d-l} g(i)+\sum_{i=1}^{l+e-m} f(i),v+\sum_{i=1}^{k+d-l} g(i)+\sum_{i=1}^{l+e-m} h(i)) \\
\leq \;& k+(e+d)-m+\mathcal{D}^n_{(k+(e+d)-m)^2}(t,v+\sum_{i=1}^{k+d-l} g(i)+\sum_{i=1}^{l+e-m} h(i)) \\
\leq \;& k+(e+d)-m+\mathcal{D}^n_{(k+(e+d)-m)^2}(t,v+\sum_{i=1}^{l+(d+e)-m} p(i))
\end{aligned}$$

where $p:\{1,\ldots,l+(d+e)-m\}\to V$, $p(i)=f(i)$ if $i\leq l+e-m$ and $p(i)=g(i-(l+e-m))$ otherwise. But, then

$$\begin{aligned}
& \mathcal{D}_e^{n+1}(node(m,t,T),node(l,u,U)) + \mathcal{D}_d^{n+1}(node(l,u,U),node(k,v,V)) \\
\leq \;& \mathcal{D}^n_{e+d}(node(m,t,T),node(k,v,V))
\end{aligned}$$

This concludes the proof. □

Lemma 21 *For every t,u,e, if $t \leq_e^{\max\{depth(t),depth(u)\}} u$, then for every $n > \max\{depth(t),depth(u)\}$, $t \leq_e^n u$ and $\mathcal{D}^n_e(t,u) = \mathcal{D}_e^{\max\{depth(t),depth(u)\}}(t,u)$.*

Proof. A straightforward induction on $\max\{depth(t),depth(u)\}$. □

$\mathcal{T}$ is the set of all trees. The binary relation $\leq_{\mathcal{T}}$ on $\mathcal{T}$ is defined by putting $t\leq_{\mathcal{T}} u$ whenever $depth(t)\leq depth(u)$ and $t\leq_0^{depth(u)} u$. $\mathcal{D}_{\mathcal{T}}$ is defined by letting $\mathcal{D}_{\mathcal{T}}(t,u)=\mathcal{D}_0^{depth(u)}(t,u)$.

Lemma 22 *$(\mathcal{T}, +, \leq_{\mathcal{T}}, \mathcal{D}_{\mathcal{T}})$ is a resource monoid.*

Proof. $(\mathcal{T}, +)$ is certainly a commutative monoid. For every t, $t \leq_{\mathcal{T}} t$, as can be proved by induction on t: $empty \leq_0^0 empty$ by definition and, moreover, $t = node(m, u, U) \leq_0^{depth(t)} t$ because, by inductive hypothesis, $u \leq_0^{depth(u)} u$ which yields, by lemma 21, $u \leq_0^{depth(t)-1} u$. In the same way, we can prove that, for every $v \in U$, $v \leq_0^{depth(t)-1} v$. Now, suppose $t \leq_{\mathcal{T}} u$ and $u \leq_{\mathcal{T}} v$. This means that $t \leq_0^{depth(u)} u$, $u \leq_0^{depth(v)} v$, $depth(t) \leq depth(u)$ and $depth(u) \leq depth(v)$. We can then conclude that $depth(t) \leq depth(v)$, that $t \leq_0^{depth(v)} u$ (by lemma 21) and $t \leq_0^{depth(v)} v$ (by proposition 6). This in turn yields $t \leq_{\mathcal{T}} v$. Let us now prove compatibility: Suppose $t \leq_{\mathcal{T}} u$ and let v be a tree. Then $depth(t) \leq depth(u)$ and $t \leq_0^{depth(u)} u$. If $depth(v) \leq depth(u)$, then $depth(u + v) = depth(u)$ and we can proceed by getting $t + v \leq_0^{depth(u+v)} u + v$ (by proposition 3), which means $t + v \leq_{\mathcal{T}} u + v$. If, on the other hand, $depth(v) > depth(u)$, then we can first apply lemma 21 obtaining $t \leq_0^{depth(u+v)} u$ and then $t + v \leq_0^{depth(u+v)} u + v$ (by proposition 3). By way of lemma 21 and propositions 6 and 5 we get

$$\begin{array}{rcl}
\mathcal{D}_{\mathcal{T}}(t,u) + \mathcal{D}_{\mathcal{T}}(u,v) & = & \mathcal{D}_0^{depth(u)}(t,u) + \mathcal{D}_0^{depth(v)}(u,v) \\
& = & \mathcal{D}_0^{depth(v)}(t,u) + \mathcal{D}_0^{depth(v)}(u,v) \\
& \leq & \mathcal{D}_0^{depth(v)}(t,v) = \mathcal{D}_{\mathcal{T}}(t,v) \\
\mathcal{D}_{\mathcal{T}}(t,u) & = & \mathcal{D}_0^{depth(u)}(t,u) \leq \mathcal{D}_0^{depth(u+v)}(t,u) \\
& \leq & \mathcal{D}_0^{depth(u+v)}(t+v,u+v) = \mathcal{D}_{\mathcal{T}}(t+v,u+v)
\end{array}$$

This concludes the proof. □

A *light length space* is a length space on the resource monoid $(\mathcal{T}, +, \leq_{\mathcal{T}}, \mathcal{D}_{\mathcal{T}})$. Given a light length space $A = (|A|, \Vdash_A)$, we can define:

- The light length space $!A = (|A|, \Vdash_{!A})$ where $e, t \Vdash_{!A} a$ iff $e, u \Vdash_A a$ and $t \geq_{\mathcal{T}} !u$.
- The light length space $\S A = (|A|, \Vdash_{\S A})$ where $e, t \Vdash_{\S A} a$ iff $e, u \Vdash_A a$ and $t \geq_{\mathcal{T}} \S u$.

The following results states the existence of certain morphisms and will be useful when interpreting light affine logic.

Lemma 23 (Basic Maps) *Given light length spaces A, B, there are morphisms: $contr : !A \to !A \otimes !A$, $distr : \S A \otimes \S B \to \S(A \otimes B)$ and $derelict : !A \to \S A$ where $contr(a) = (a, a)$ and $distr(a, b) = (a, b)$ and $derelict(a) = a$.*

Proof. We know that $\{e_{contr}\}(d)$ takes time linear in $|d|$, say at most $p|d| + q$. Then, let $t, u \in \mathcal{T}$ be such that $\mathcal{F}_{\mathcal{T}}(t) \geq p + q + |e_{contr}|$, $\mathcal{F}_{\mathcal{T}}(u) \geq cp$. Define t_{contr} to be $t + v + [1]_{\mathcal{T}}$. Clearly, $\mathcal{F}_{\mathcal{T}}(t_{contr}) \geq |e_{contr}|$ Now, let $v, d \Vdash_{!A} a$. This means that $v \geq_{\mathcal{T}} !w$ where $w, d \Vdash_A a$. Then:

$$\begin{array}{rcl}
u{+}!w{+}!w & \geq_{\mathcal{T}} & !w{+}!w \\
\mathcal{F}_{\mathcal{T}}(u{+}!w{+}!w) & \geq & \mathcal{F}_{\mathcal{T}}(u) + \mathcal{F}_{\mathcal{T}}(!w) + \mathcal{F}_{\mathcal{T}}(!w) \\
& \geq & cp + \mathcal{F}_{\mathcal{T}}(!w) + \mathcal{F}_{\mathcal{T}}(!w)
\end{array}$$

This means that $u{+}!w{+}!w, e \Vdash_{!A \otimes !A} (a, a)$. Moreover, $u{+}!w{+}!w = u{+}!w + [1]_{\mathcal{T}} \leq_{\mathcal{T}} u{+}!w + t_{contr}$. Finally,

$$\begin{array}{rcl}
Time(\{e_{contr}\}(d)) & \leq & p|d| + q \leq (p+q)(|d| + p + q) \\
& \leq & \mathcal{F}_{\mathcal{T}}(t_{contr})(\mathcal{F}_{\mathcal{T}}(v) + \mathcal{F}_{\mathcal{T}}(t_{contr})) \\
& \leq & (\mathcal{F}_{\mathcal{T}}(t_{contr}) + \mathcal{D}_{\mathcal{T}}(v,v))(\mathcal{F}_{\mathcal{T}}(t_{contr}) + \mathcal{F}_{\mathcal{T}}(v)) \\
& \leq & \mathcal{D}_{\mathcal{T}}(v, v + t_{contr})\mathcal{F}_{\mathcal{T}}(t_{contr} + v)
\end{array}$$

This proves *contr* to be a morphism.

Let $e_{distr} = e_{id}$. We know that $\{e_{id}\}(d)$ takes time linear in $|d|$, say at most $p|d| + q$. Then, let $t, u \in \mathcal{T}$ be such that $\mathcal{F}_{\mathcal{T}}(t) \geq p + q + |e_{distr}|$, $\mathcal{F}_{\mathcal{T}}(u) \geq cp$. t_{distr} is then defined as $t + \S u$. Now, let $v, \langle d, c\rangle \Vdash_{\S A \otimes \S B} (a, b)$. This

means that $v \geq \S w + \S x$, where $w, d \Vdash_A a$ and $x, c \Vdash_B b$. This in turn means that $w + x + u, \langle d, c\rangle \Vdash_{A\otimes B} (a, b)$ and $\S(w + x + u), \langle d, c\rangle \Vdash_{A\otimes B} (a, b)$. Moreover

$$\S(w + x + u) = \S w + \S x + \S u \leq v + t_{distr}$$

Finally:

$$\begin{array}{rcl} Time(\{e_{distr}\}(\langle d, c\rangle)) & \leq & p|\langle d, c\rangle| + q \leq (p + q)(|\langle d, c\rangle| + p + q) \leq \mathcal{F}_{\mathcal{T}}(t_{distr})(\mathcal{F}_{\mathcal{T}}(v) + \mathcal{F}_M(t_{distr}) \\ & \leq & (\mathcal{F}_{\mathcal{T}}(t_{distr}) + \mathcal{D}_M(v, v))(\mathcal{F}_{\mathcal{T}}(t_{distr}) + \mathcal{F}_{\mathcal{T}}(v)) \\ & \leq & \mathcal{D}_{\mathcal{T}}(v, v + t_{distr})\mathcal{F}_{\mathcal{T}}(t_{distr} + v) \end{array}$$

This $distr$ to be a morphism.

Let $e_{derelict} = e_{id}$. We know that $\{e_{derelict}\}(d)$ takes time linear in $|d|$, say at most $p|d| + q$. Then, let $t_{distr} \in \mathcal{T}$ be such that $\mathcal{F}_{\mathcal{T}}(t_{distr}) \geq p + q + |e_{derlict}|$. Now, let $v, d \Vdash_{!A} a$. This means that $v \geq !w$, where $w, d \Vdash_A a$. This in turn means that $\S w, d \Vdash_{\S A} a$. Moreover

$$\S w \leq !w \leq !w + t_{derelict}.$$

Finally:

$$\begin{array}{rcl} Time(\{e_{distr}\}(d)) & \leq & p|d| + q \leq (p + q)(|d| + p + q) \leq \mathcal{F}_{\mathcal{T}}(t_{derelict})(\mathcal{F}_{\mathcal{T}}(v) + \mathcal{F}_{\mathcal{T}}(t_{derelict}) \\ & \leq & (\mathcal{F}_{\mathcal{T}}(t_{derelict}) + \mathcal{D}_{\mathcal{T}}(v, v))(\mathcal{F}_{\mathcal{T}}(t_{derelict}) + \mathcal{F}_{\mathcal{T}}(v)) \\ & \leq & \mathcal{D}_{\mathcal{T}}(v, v + t_{derelict})\mathcal{F}_{\mathcal{T}}(t_{derelict} + v) \end{array}$$

This proves $derelict$ to be a morphism. □

Lemma 24 *For every $t \in \mathcal{T}$, there is u such that, for every v, $!(v + t) \leq_{\mathcal{T}} !v + u$.*

Proof. First of all we will prove the following statement by induction on t: for every t, there is an integer $\overline{t}$ such that for every u, $u + t \leq_{\overline{t}}^{\max\{depth(u), depth(t)\}} u$. If $t = empty$, we can choose $\overline{t}$ to be just 0, since $u \leq_0^n u$ for every u. If $t = node(m, v, V)$, then we put $\overline{t} = m + \overline{v} + \sum_{w\in V} \overline{w}$. Let u be an arbitrary tree and let us assume, without losing generality, that $u = node(l, w, W)$. Let $d = \overline{v} + \sum_{w\in V} \overline{w}$. We get

$$\begin{array}{rcl} l + m & \leq & l + m + (\overline{v} + \sum_{w\in V} \overline{w}) - (\overline{v} + \sum_{w\in V} \overline{w}) \\ & = & l + \overline{t} - d \\ v + w & \leq_{\overline{v}}^{\max\{depth(v), depth(w)\}} & w \\ & \leq_0^{\max\{depth(v), depth(w)\}} & w + \sum_{i=1}^{d} empty \\ \forall x \in V.x & \leq_{\overline{x}}^{depth(x)} & empty \\ \forall x \in W.x & \leq_0^{depth(x)} & x \end{array}$$

Using known results, we can rewrite these inequalities as follows

$$\begin{array}{rcl} l + m & \leq & l + \overline{t} - d \\ v + w & \leq_{\overline{t}}^{\max\{depth(t), depth(u)\}-1} & w + \sum_{i=1}^{d} empty \\ \forall x \in V.x & \leq_{\overline{t}}^{\max\{depth(t), depth(u)\}-1} & empty \\ \forall x \in W.x & \leq_{\overline{t}}^{\max\{depth(t), depth(u)\}-1} & x \end{array}$$

This yields $u + t \leq_{\overline{t}}^{\max\{depth(u), depth(t)\}} t$.

Let us now go back to the lemma we are proving. We will now prove that for every t, every term $u = node(\overline{t}, u, U)$ such that $depth(u) \geq depth(t) + 1$ satisfies the thesis. Indeed, if we put $d = \overline{t}$ and $n = depth(!v + u) - 1$, we get:

$$\begin{array}{rcl} 0 & \leq & \overline{t} - d \\ empty & \leq^n_{d^2} & u \\ v + t & \leq^n_d & v \end{array}$$

This, in turn implies $!(v + t) \leq^{n+1}_0 !v + u$, which yields $!(v + t) \leq_{\mathcal{T}} !v + u$. □

Lemma 25 (Functoriality) *If $f : A \xrightarrow{e,\varphi} B$, then there are ψ, θ such that $f : !A \xrightarrow{e,\psi} !B$ and $f : \S A \xrightarrow{e,\theta} \S B$.*

Proof. Let ξ be the tree obtained from φ by lemma 24 and put $\psi = \xi + \varphi$. Suppose that $d, t \Vdash_{!A} a$. Then $t \geq !u$, where $d, u \Vdash_A a$. Observe that there must be v, c such that $c, v \Vdash_B f(a)$, $v \leq_{\mathcal{T}} u+\varphi$ and $Time(\{e\}(d)) \leq \mathcal{F}_{\mathcal{T}}(u+\varphi)\mathcal{D}_{\mathcal{T}}(v, u+\varphi)$. But then $c, !v \Vdash_{!B} f(a)$ and moreover

$$\begin{array}{rcl} !v & \leq_{\mathcal{T}} & !(u + \varphi) \leq_{\mathcal{T}} !u + \xi \leq_{\mathcal{T}} t + \psi \\ Time(\{e\}(d)) & \leq & \mathcal{F}_{\mathcal{T}}(u + \varphi)\mathcal{D}_{\mathcal{T}}(v, u + \varphi) \\ & \leq & \mathcal{F}_{\mathcal{T}}(!(u + \varphi))\mathcal{D}_{\mathcal{T}}(!v, !(u + \varphi)) \\ & \leq & \mathcal{F}_{\mathcal{T}}(!u + \xi)\mathcal{D}_{\mathcal{T}}(!v, !u + \xi) \\ & \leq & \mathcal{F}_{\mathcal{T}}(t + \psi)\mathcal{D}_{\mathcal{T}}(!v, t + \psi) \end{array}$$

This means that $f : !A \xrightarrow{e,\psi} !B$. Now, let θ be $\S\varphi$ and suppose $d, t \Vdash_{\S A} a$. Then $t \geq \S u$, where $d, u \Vdash_A a$. Observe that there must be v, c such that $e, v \Vdash_B f(a)$, $v \leq_{\mathcal{T}} u + \varphi$ and $Time(\{e\}(d)) \leq \mathcal{F}_{\mathcal{T}}(u + \varphi)\mathcal{D}_{\mathcal{T}}(v, u + \varphi)$. But then $c, \S v \Vdash_{\S B} f(a)$ and, moreover

$$\begin{array}{rcl} \S v & \leq_{\mathcal{T}} & \S(u + \varphi) = \S u + \S\varphi \leq_{\mathcal{T}} t + \theta \\ Time(\{e\}(d)) & \leq & \mathcal{F}_{\mathcal{T}}(u + \varphi)\mathcal{D}_{\mathcal{T}}(v, u + \varphi) \\ & \leq & \mathcal{F}_{\mathcal{T}}(\S(u + \varphi))\mathcal{D}_{\mathcal{T}}(\S v, \S(u + \varphi)) \\ & \leq & \mathcal{F}_{\mathcal{T}}(\S u + \S\varphi)\mathcal{D}_{\mathcal{T}}(\S v, \S u + \S\varphi)) \\ & \leq & \mathcal{F}_{\mathcal{T}}(t + \theta)\mathcal{D}_{\mathcal{T}}(\S v, t + \theta) \end{array}$$

This means that $f : \S A \xrightarrow{e,\theta} \S B$. □

Now, we can prove a polynomial bound on $\mathcal{F}_T(t)$:

Proposition 7 *For every $n \in \mathbb{N}$ there is a polynomial $p_n : \mathbb{N} \to \mathbb{N}$ such that $\mathcal{F}_{\mathcal{T}}(t) \leq p_{depth(t)}(|t|)$.*

Proof. We prove a stronger statement by induction on n: for every $n \in \mathbb{N}$ there is a polynomial $q_n : \mathbb{N}^2 \to \mathbb{N}$ such that for every t, e, $\mathcal{D}^n_e(empty, t) \leq q_n(|t|, e)$. First of all, we know that $\mathcal{D}^0_e(empty, t) = 0$, so q_0 is just the function which always returns 0. q_{n+1} is defined from q_n as follows: $q_{n+1}(x, y) = x + y + q_n(x(x + y + 1), (x + y)^2)$. Indeed:

$$\begin{array}{rcl} \mathcal{D}^{n+1}_e(empty, empty) & = & e + \mathcal{D}^n_e(empty, empty) \\ & \leq & e + q_n(0, e) \leq e + |empty| \\ & & +q_n(|empty|(|empty| + e + 1), (|empty| + e)^2) \\ \mathcal{D}^{n+1}_e(empty, node(m, t, T)) & = & m + e + \max_f\{\mathcal{D}^n_{(m+e)^2}(empty, t + \sum_{i=1}^{m+e} f(i))\} \\ & \leq & m + e + q_n((m + e + 1)(|node(m, t, T)|), (m + e)^2) \\ & \leq & |node(m, t, T)| + e \\ & & +q_n((|node(m, t, T)| + e + 1)(|node(m, t, T)|), (|node(m, t, T)| + e)^2) \end{array}$$

At this point, however, it suffices to put $p_n(x) = q_n(x, 0)$. □

Exponential Rules and Contraction.

$$\frac{\Gamma, \Delta \vdash A}{\S\Gamma, !\Delta \vdash \S A}\ P_{\S} \qquad \frac{A \vdash B}{!A \vdash !B}\ P_{!}^{1} \qquad \frac{\vdash A}{\vdash !A}\ P_{!}^{2} \qquad \frac{\Gamma, !A, !A \vdash B}{\Gamma, !A \vdash B}\ C$$

Figure 4. Intuitionistic Light Affine Logic

6.1 Interpreting Light Affine Logic

The grammar of formulae is the one from Elementary Affine Logic, enriched with a new production $A ::= \S A$. Rules are reported in figure 4. As for the ! modality, $[\![\S A]\!]_{\eta}^{\mathscr{R}} = \S[\![A]\!]_{\eta}^{\mathscr{R}}$.

Theorem 3 *Light length spaces form a model of* LAL.

Binary lists can be represented in LAL as cut-free proofs with conclusion

$$List_{\mathsf{LAL}} \equiv \forall\alpha.!(\alpha \multimap \alpha) \multimap !(\alpha \multimap \alpha) \multimap \S(\alpha \multimap \alpha)$$

Corollary 3 (Soundness) *Let π be an* LAL *proof with conclusion $\vdash \{!, \S\}^{j} List_{\mathsf{LAL}} \multimap \{!, \S\}^{k} List_{\mathsf{LAL}}$ and let $f : L \to L$ be the function induced by $[\![\pi]\!]^{\mathscr{R}}$. Then f is computable in polynomial time.*

7 LFPL

In [6] one of us had introduced another language, LFPL, with the property that all definable functions on natural numbers are polynomial time computable. The key difference between LFPL and other systems is that a function defined by iteration or recursion is not marked as such using modalities or similar and can therefore be used as a step function of subsequent recursive definitions.

In this section we will describe a resource monoid $\mathcal{M}$ for that language as well which will provide a proof of polytime soundness for that system which is essentially the same as the proof from loc. cit., but more structured and, hopefully, easier to understand.

The new approach also yields some new results, namely the justification of second-order quantification, a !-modality, and a new type of binary trees based on cartesian product which allows alternative but not simultaneous access to subtrees.

7.1 Overview of LFPL

LFPL is intuitionistic, affine linear logic, i.e., a linear functional language with $\otimes, \multimap, +, \times$. Unlike in the original presentation we also add polymorphic quantification here. In addition, LFPL has basic types for inductive datatypes, for example unary and binary natural numbers, lists, and trees. There is one more basic type, namely $\Diamond$, the resource type.

The recursive constructors for the inductive datatypes each take an additional argument of type $\Diamond$ which prevents one to invoke more constructor functions than one. Dually to the constructors one has iteration principles which make the $\Diamond$-resource available in the branches of a recursive definition. For example, the type $T(X)$ of X-labelled binary trees has constructors $\mathbf{leaf} : T(X)$ and $\mathbf{node} : \Diamond \multimap X \multimap T(X) \multimap T(X) \multimap T(X)$. The iteration principle allows one to define a function $T(X) \multimap A$ from closed terms A and $\Diamond \multimap X \multimap A \multimap A \multimap A$.

In this paper we "internalise" the assumption of closedness using a !-modality.

Using this iteration principle one can encode recursive definitions by ML-style pattern matching provided recursive calls are made on structurally smaller arguments only.

Here is a fragment of an LFPL program for "treesort" written in functional notation: the additional arguments of type $\Diamond$ are supplied using @. Note that the insert function takes an extra argument of type $\Diamond$.

```
let insert x t d = match t with
   Leaf -> Node(x,Leaf,Leaf)@d
 | Node(y,l,r)@d' ->
  if x<=y then Node(y,insert x l d,r)@d'
```

```
          else Node(y,l,insert x r d)@d'

let extract t = match t with
   Leaf -> nil
 | Node(x,l,r)@d ->
  append (extract l) (cons(x,extract r)@d)
```

7.2 A resource monoid for LFPL

The underlying set of $\mathcal{M}$ is the set of pairs where $l \in \mathbb{N}$ is a natural number and p is a monotone polynomial in a single variable x. The addition is defined by $(l_1, p_1) + (l_2, p_2) = (l_1 + l_2, p_1 + p_2)$, accordingly, the neutral element is $0 = (0, 0)$. We have a submonoid $\mathcal{M}_0 = \{(l, p) \in \mathcal{M} \mid l = 0\}$.

To define the ordering we set $(l_1, p_1) \leq (l_2, p_2)$ iff $l_1 \leq l_2$ and $(p_2 - p_1)(x)$ is monotone and nonnegative for all $x \geq l_2$. For example, we have $(1, 42x) \leq (42, x^2)$, but $(1, 42x) \not\leq (41, x^2)$. The distance function is defined by

$$\mathcal{D}_{\mathcal{M}}((l_1, p_1), (l_2, p_2)) = (p_2 - p_1)(l_2)$$

We can pad elements of $\mathcal{M}$ by adding a constant to the polynomial. The following is now obvious.

Lemma 26 *Both $\mathcal{M}$ and $\mathcal{M}_0$ are resource monoids.*

A simple inspection of the proofs in Section 3.1 shows that the realisers for all maps can be chosen from $\mathcal{M}_0$. This is actually the case for an arbitrary submonoid of a resource monoid. We note that realisers of elements may nevertheless be drawn from all of $\mathcal{M}$. We are thus led to the following definition.

Definition 1 *An* LFPL*-space is a length space over the resource monoid $\mathcal{M}$. A morphism from* LFPL *length space A to B is a morphism between length spaces which admits a majorizer from $\mathcal{M}_0$.*

Proposition 8 LFPL *length spaces with their maps form a symmetric monoidal closed category.*

7.3 A !-modality for LFPL

We abbreviate $\sigma + \cdots + \sigma$ (n times) as $n.\sigma$.

Lemma 27 *There is an element $\delta \in \mathcal{M}$ with the following "anti-archimedean" property. For each $\sigma \in \mathcal{M}_0$ there exists $\sigma^* \in \mathcal{M}_0$ such that for all $n \in \mathbb{N}$*

$$n.(\sigma + \delta) \leq \sigma^* + n.\delta$$

Proof. Choose $\delta = (l, q)$ where $l \geq 1$ and q arbitrary, e.g., $q = 1$. Given $\sigma = (0, p) \in \mathcal{M}_0$ define $\sigma^* = (0, xp)$. Now, $n.(\sigma + \delta) \geq (nl, np + nq)$ and $\sigma^* + n.\delta = (nl, xp + nq)$. But, $xp - np$ is monotone and nonnegative when $x \geq n$ so $(nl, xp + nq) \geq (nl, np + nq)$ as required. □

Definition 2 *Let A be an* LFPL *space and $n \in \mathbb{N}$. The* LFPL *space A^n is defined by $|A^n| = |A|$ and $e, \alpha \Vdash_{A^n} a$ iff $\alpha \geq n.\beta$ for some β such that $e, \beta \Vdash_A a$.*

So, A^n corresponds to the subset of $A \otimes \cdots \otimes A$ consisting of those tuples with all n components equal to each other.

Let I be an index set and A_i, B_i be I-indexed families of LFPL spaces. A uniform map from $(A_i)_i$ to $(B_i)_i$ consists of a family of maps $f_i : A_i \to B_i$ such that there exist e, α with the property that $e, \alpha \Vdash f_i$ for all i. Recall that, in particular, the denotations of proofs with free type variables are uniform maps.

It is clear that we have uniform isomorphisms $A^{m+n} \to A^m \otimes A^n$ and similar ones.

Proposition 9 *There is an* LFPL *space $\Diamond$ and for each* LFPL *space A there is an* LFPL *space $!A$ with the following properties:*

- $|!A| = |A|$.
- *If $f : A \to B$ then $f : !A \to !B$.*

- $!(A \otimes B) \simeq !A \otimes !B$
- *The obvious maps* $!A \otimes \Diamond^n \to A^n \otimes \Diamond^n$ *are a uniform morphism.*

The last property means intuitively that with n *"diamonds" we can extract* n *copies from an element of type* $!A$ *and get the* n *"diamonds" back for later use.*

Proof. Let δ be as in the proof of Lemma 27. Pick any $d \in L$ so that $|d| \leq \mathcal{F}_\delta$. Define $|\Diamond| = \{\Diamond\}$ and put $d, \alpha \Vdash_\Diamond \Diamond$ if $\alpha \geq \delta$.

If A is an LFPL-space form the length space $!A$ by $|!A| = |A|$ and $t, \alpha \Vdash_{!A} a$ if there exists $\alpha' = (0, p) \in \mathcal{M}_0$ with $t, \alpha' \Vdash_A a$ and $\alpha \geq (0, (x+1)p)$.

We have $(0+1)p(0) = p(0) \geq |t|$. Compatibility with $\otimes$ is obvious.

For functoriality assume that $e, \phi \Vdash f$ where $\phi = (0, q) \in \mathcal{M}_0$. We claim that $e, (0, (x+1)q) \Vdash f$ *qua* morphism from $!A$ to $!B$. Suppose that $t, \alpha \Vdash_{!A} a$ where $\alpha \geq (0, (x+1)p)$ and $t, (0, p) \Vdash_A$. Since f is a morphism, we obtain v, β such that $v, \beta \Vdash_B f(a)$ and $\beta \leq \phi + (0, p)$. This implies that $\beta \in \mathcal{M}_0$ as well, say, $\beta = (0, r)$ where $r \leq p + q$. We also know that $r(0) \geq |v|$ by the definition of length spaces. Now $v, (0, (x+1)r \Vdash_{!B} f(b)$. On the other hand $(x+1)r \leq (x+1)(p+q)$. The resource bounds are obvious.

Finally, consider the required morphism $!A \otimes \Diamond^n \to A^n \otimes \Diamond^n$. Clearly, it may be realised by the identity; we claim that 0 can serve as a majoriser. Indeed, a majoriser of $(a, d) \in |!A \otimes \Diamond^{\otimes n}|$ is of the form $(n, (x+1)p)$ where $(0, p)$ majorises a in A. Now, (n, np) is a majoriser of (a, d) in $A^n \otimes \Diamond^n$. But $(x+1) - np$ is monotone and nonnegative above n. □

Remark We remark at this point that we obtain an alternative resource monoid $\mathcal{M}_S$ for SAL whose underlying set and ordering are as in $\mathcal{M}$, but whose addition is given by addition as $(l_1, p_1) + (l_2, p_2) = (\max(l_1, l_2), p_1 + p_2)$. Length spaces over $\mathcal{M}_S$ with maps majorised by $\mathcal{M}_S$ (not $\mathcal{M}_0$) then also form a sound model of SAL. This points to a close relationship between LFPL and SAL and also shows a certain tradeoff between the two systems. The slightly more complex model is needed for LFPL since in LFPL the C-rule of SAL is so to say internalised in the form of the uniform map $!A \otimes \Diamond^n \to A^n \otimes \Diamond^n$. Notice that SAL's map $!A \to A^n$ cannot be uniform. This uniformity of LFPL allows for an internal implementation of datatypes and recursion as we now show.

7.4 Dependent typing

Definition 3 *Let* T_i *be a family of* LFPL *spaces such that* $|T_i| = T$ *independent of* i*. The* LFPL *space* $\exists i.T_i$ *is defined by* $|\exists i.T_i| = |T|$ *and* $e, \alpha \Vdash_{\exists i.T_i} t$ *if* $e, \alpha \Vdash_{T_i} t$ *for some* i*.*

Note that if we have a uniform family of maps $T_i \to U$ where U does not depend on i then we obtain a map $\exists i.T_i \to U$ (existential elimination).

Conversely, if we have a uniform family of maps $U_i \to V_{f(i)}$ then we get a uniform family of maps $U_i \to \exists j.V_j$ (existential introduction). We will use an informal "internal language" to denote uniform maps which when formalised would amount to an extension of LFPL with indexed type dependency in the style of Dependent ML [10].

7.5 Inductive datatypes

In order to interpret unary natural numbers, we define $N = \exists n.N_n$ where

$$N_n = \Diamond^n \otimes \forall A.(A \multimap A)^n \multimap A \multimap A$$

We can internally define a successor map $\Diamond \otimes N_n \to N_{n+1}$ as follows: starting from $d : \Diamond, \vec{d} : \Diamond^n$ and $f : \forall(A \multimap A)^n \multimap A \multimap A$ we obtain a member of $\Diamond^{n+1}$ (from d and $\vec{d}$) and we define $f' : \forall(A \multimap A)^{n+1} \multimap A \multimap A$ as $\lambda(u^{A \multimap A}, \vec{u}^{(A \multimap A)^n}).\lambda z^A.u(f\ \vec{u}\ z)$. From this, we obtain a map $\Diamond \otimes N \to N$ by existential introduction and elimination.

Of course, we also have a constant zero $I \to N_0$ yielding a map $I \to N$ by existential introduction.

Finally, we can define an iteration map

$$!(\Diamond \otimes A \multimap A) \multimap N_n \multimap A \multimap A$$

as follows: Given $t :!(\Diamond \otimes A \multimap A)$ and $(\vec{d}, f) \in N_n$ we unpack t using Proposition 9 to yield $t' \in ((\Diamond \otimes A) \multimap A)^n$ as well as $\vec{d} \in \Diamond^n$. Feeding these "diamonds" one by one to the components of t' we obtain $t'' \in (A \multimap A)^{\otimes n}$. But then $f\ t''$ yields the required element of $A \multimap A$.

Existential elimination now yields a single map

$$!(\Diamond \otimes A \multimap A) \multimap N \multimap A \multimap A$$

Similarly, we can interpret binary X-labelled trees using a type family

$$T_n = \Diamond^n \otimes \forall (X \multimap A \multimap A \multimap A)^n \multimap A^{n+1} \multimap A$$

and defining trees proper as $\exists n.T_n$. We get maps $\mathbf{leaf} : T_0$ and $\mathbf{node} : \Diamond \otimes X \otimes T_{n_1} \otimes T_{n_2} \to T_{n_1+n_2+1}$ and an analogous iteration construct.

Finally, and this goes beyond what was already known, we can define "lazy trees" using cartesian product (also known as additive conjunction).

First, we recall from ordinary affine linear logic that an additive conjunction can be defined as

$$A \times B = \forall C.(C \multimap A) \otimes (C \multimap B) \otimes C$$

The first projection map $A \times B \to A$ is given internally by $\lambda(f^{C \multimap A}, g^{C \multimap B}, c^C).f\ c$. Analogously, we have a second projection. Given maps $f : C \to A$ and $g : C \to B$ we obtain a map $\langle f, g \rangle : C \to A \times B$ internally as $\lambda c^C.(f, g, c)$.

Now, following the pattern of the binary trees $T_{m,n}$ above, we define another family

$$T^{\times}_d = \Diamond^d \otimes \forall A.(X \multimap (A \times A) \multimap A)^d \multimap A \multimap A$$

and $T^{\times} = \exists d.T^{\times}_d$. We get maps $\mathbf{leaf} : \Diamond \to T^{\times}_0$ and $\mathbf{node} : \Diamond \otimes X \otimes (T_{d_1} \times T_{d_2}) \to T_{1+\max(d_1,d_2)}$ as well as an analogous iteration construct.

We describe in detail the construction of the "node" map which is not entirely straightforward. First, we note that for any length spaces A, B and m, n the obvious map $(\Diamond^m \otimes A) \times (\Diamond^n \otimes B) \to \Diamond^{\max(m,n)} \otimes (A \times B)$ is a morphism. This is because a majoriser of an element of $(\Diamond^m \otimes A) \times (\Diamond^n \otimes B)$ must be of the form (k, p) where $k \geq \max(m, n)$ in view of the existence of the projection maps.

Now suppose we are given (internally) $d : \Diamond, x : X, lr : T^{\times}_{d_1} \times T^{\times}_{d_2}$. Using the just described morphism we decompose lr into $\vec{d} : \Diamond^{\max(d_1,d_2)}$ and $lr' : W_{d_1} \times W_{d_2}$ where $W_i = (X \multimap (A \times A) \multimap A)^i \multimap A \multimap A$. We have stripped off the universal quantifier.

Now d and $\vec{d}$ together yield an element of $\Diamond^{1+\max(d_1,d_2)}$. It remains to construct a member of $W_{1+\max(d_1,d_2)}$. To this end, we assume $u : X \multimap (A \times A) \multimap A$ and $f : (X \multimap (A \times A) \multimap A)^{\max(d_1,d_2)}$ and define the required element of A as $u\ x\ \langle lr'.1\ f\ a, lr'.2\ f\ a \rangle$. Here .1 and .2 denote the projections from the cartesian product. The sharing of the variables f, a, lr' is legal in the two components of a cartesian pairing, but would of course not be acceptable in a $\otimes$ pairing. We have elided the obvious coercions from $(\_)^{\max(d_1,d_2)}$ to $(\_)^{d_i}$.

We remark that these cartesian trees are governed by their depth rather than their number of nodes. We also note that if $X = I$ we can form the function $\lambda d^{\Diamond}.\lambda t^{T^{\times}}.\mathbf{node}\ d\ ()\ \langle t, r \rangle : \Diamond \multimap T^{\times} \multimap T^{\times}$. Iterating this map yields a function $N \multimap T^{\times}$ computing full binary trees of a given depth. Of course, on the level of the realisers, such a tree is not laid out in full as this would require exponential space, but computed lazily as subtrees are being accessed. Exploring the implications of this for programming is left to future work.

8 Conclusion and Related Work

We have given a unified semantic framework with which to establish soundness of various systems for capturing complexity classes by logic and programming. Most notably, our framework has all of second-order multiplicative linear logic built in, so that only the connectives and modalities going beyond this need to be verified explicitly.

While resulting in a considerable simplification of previous soundness proofs, in particular for LFPL and LAL, our method has also lead to new results, in particular polymorphism and "!" for LFPL.

The method proceeds by assiging both abstract resource bounds in the form of elements from a resource monoid and resource-bounded computations to proofs, resp. programs. In this way, our method can be seen as a combination of traditional Kleene-style realisability (which only assigns computations) and polynomial and quasi interpretation known from

term rewriting (which only assigns resource bounds). An altogether new aspect is the introduction of more general notions of resource bounds than just numbers or polynomials as formalised in the concept of resource monoid. We thus believe that our methods can also be used to generalise polynomial interpretations to (linear) higher-order.

References

[1] Andrea Asperti and Luca Roversi. Intuitionistic light affine logic. *ACM Transactions on Computational Logic*, 3(1):137–175, 2002.

[2] Patrick Baillot and Virgile Mogbil. 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.

[3] Stephen Bellantoni, Karl Heinz Niggl, and Helmut Schwichtenberg. Higher type recursion, ramification and polynomial time. *Annals of Pure and Applied Logic*, 104:17–30, 2000.

[4] Paolo Coppola and Simone Martini. Typing lambda terms in elementary logic with linear constraints. In *Proceedings of the 6th International Conference on Typed Lambda-Calculus and Applications*, pages 76–90, 2001.

[5] Jean-Yves Girard. Light linear logic. *Information and Computation*, 143(2):175–204, 1998.

[6] Martin Hofmann. Linear types and non-size-increasing polynomial time computation. In *Proceedings of the 14th IEEE Syposium on Logic in Computer Science*, pages 464–473, 1999.

[7] Martin Hofmann. Safe recursion with higher types and BCK-algebra. *Annals of Pure and Applied Logic*, 104:113–166, 2000.

[8] Martin Hofmann and Philip Scott. Realizability models for BLL-like languages. *Theoretical Computer Science*, 318(1-2):121–137, 2004.

[9] Yves Lafont. Soft linear logic and polynomial time. *Theoretical Computer Science*, 318:163–180, 2004.

[10] Hongwei Xi and Frank Pfenning. Dependent types in practical programming. In *Proceedings of the 26th ACM SIGPLAN Symposium on Principles of Programming Languages*, pages 214–227, 1999.