Theoretical Computer Science 412 (2011) 2029–2047

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Realizability models and implicit complexity

Ugo Dal Lago [a,*], Martin Hofmann [b]

[a] Dipartimento di Scienze dell'Informazione, Università di Bologna, Italy
[b] Institut für Informatik, Ludwig-Maximilians-Universität, München, Germany

ARTICLE INFO

Keywords:
Implicit computational complexity
Realizability
Linear logic

ABSTRACT

New, simple, proofs of soundness (every representable function lies in a given complexity class) for Elementary Affine Logic, LFPL and Soft Affine Logic are presented. The proofs are obtained by instantiating a semantic framework previously introduced by the authors and based on an innovative modification of realizability. The proof is a notable simplification on the original already semantic proof of soundness for the above mentioned logical systems and programming languages. A new result made possible by the semantic framework is the addition of polymorphism and a modality to LFPL, thus allowing for an internal definition of inductive datatypes. The methodology presented proceeds by assigning both abstract resource bounds in the form of elements from a resource monoid and resource-bounded computations to proofs (respectively, programs).

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

Implicit computational complexity is an active research area lying in the intersection of logic and computer science whose goal is to characterize complexity classes as classes of functions or predicates definable in logical systems or lambda calculi. A question that has attracted particular interest in the last two decades is how to tame systems with higher order functions and recursion so as to capture small complexity classes, polynomial time in particular. At least three different principles have been used when characterizing complexity classes by languages with higher order functions, namely linear types [4,15], restricted modalities in the context of linear logic [12,1,20] and non-size-increasing computation [16]. Although related to each other, these systems have been studied with different, often unrelated methodologies and few results are known about their relative intensional expressive power. By intensional expressive power we mean the ability to represent natural algorithms as opposed to just extensionally capture classes of functions. We believe that this is one of the main reasons that there has been relatively little progress towards the main challenge in the field, namely finding systems capturing small complexity classes while being at the same time intensionally expressive.

In a recent paper [9], the authors introduced a new semantic framework based upon an innovative modification of realizability. The main idea underlying the proposal consists in considering bounded-time algorithms as realizers instead of taking plain Turing Machines as is usually the case in realizability constructions. Bounds are expressed abstractly as elements of a *resource monoid*. Given a resource monoid M, the notions of a *length space* on M and of a *morphism* between length spaces (on M) can both be defined. Noticeably, this is a symmetric monoidal closed category, independently of M.

But our goal here is not limited to defining a new realizability model for certain logical systems or programming languages. Given a (logical or type) system L, we define a model for L by choosing a resource monoid which is both:

- *flexible* enough to justify all the constructs or rules in L.
- *restricted* enough to induce proper bounds on the running time of the underlying realizers.

The model can then be *used* as a powerful tool for the analysis of the class of functions representable in L.

* Corresponding author. Tel.: +39 0512094991; fax: +39 0512094510.
E-mail addresses: dallago@cs.unibo.it (U. Dal Lago), mhofmann@informatik.uni-muenchen.de (M. Hofmann).

0304-3975/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2010.12.025

Quite remarkably, second order multiplicative affine logic (MAL) can be interpreted in the presented framework, *independently* on the underlying resource monoid. As a consequence, the flexibility requirement should only be checked for constructs which are not in MAL.

A logical system (or a programming language) is said to be *sound* with respect to a given complexity class iff the class of functions which can be represented in the logical system is included in the complexity class. In [9], we presented proofs of soundness theorems for the following systems: Light Affine Logic (LAL, [1]), Elementary Affine Logic (EAL, [6]), LFPL [16] and Soft Affine Logic (SAL, [2]). The one in [9] was the first entirely semantic proof of polytime soundness for light logics, providing a notable simplification on the original already semantic proof of polytime soundness for LFPL. On the other hand, the resource monoids for LAL, EAL and SAL were complicated compared to the one for LFPL. The latter was a *functional* monoid: elements of the carrier are pairs (n, f), where n is a natural number and f is a function from natural numbers to natural numbers bounded by a polynomial. The first three were not functional models and, more importantly, their definition was complex; as a consequence, proof of soundness for LAL was relatively long and could not be presented in the extended abstract [9].

In this paper, we introduce the semantic framework in full detail, together with concrete instances for EAL, SAL and LFPL. The three resource monoids are all functional. A companion paper by the authors [10] presents a new, simple, functional model for LAL.

Related work. Realizability has been used in connection with resource-bounded computation in several places. The most prominent is Cook and Urquhart's work [5], where terms of a language called PV^ω are used to realize formulas of bounded arithmetic. The contribution of that paper is related to ours in that realizability is used to show "polytime soundness" of a logic. There are important differences though. First, realizers in Cook and Urquhart [5] are typed and very closely related to the logic that is being realized. Second, the language of realizers PV^ω are terms of the simply-typed lambda calculus (endowed with first order recursion) and is therefore useless for systems like LFPL or LAL. In contrast, we use untyped realizers and interpret types as certain partial equivalence relations on those. This links our work to the untyped realizability model HEO (due to Kreisel [19]). This, in turn, has also been done by Crossley et al. [8]. There, however, one proves externally that untyped realizers (in this case of bounded arithmetic formulas) are polytime. In our work, and this happens for the first time, the untyped realizers are used to give meaning to the logic and obtain polytime soundness as a corollary. Thus, certain resource bounds are built into the untyped realizers by their very construction. Such a thing is not at all obvious, because untyped universes of realizers tend to be Turing complete, due to definability of fixed-point combinators. We get around this problem through our notion of a resource monoid and the addition of certain time bounds to Kleene applications of realizers. Indeed, we consider this the main innovation of our paper and hope that it proves useful elsewhere. Similar ideas were already present in some previous works by the second author [16,14,17]. The presented techniques, however, were designed with one particular system in mind and could not be easily adapted to other systems. Our presentation style is particularly similar to the one adopted in [17].

2. A computational model

In this paper, we adopt the lambda calculus [3] as the language of realizers. More precisely, realizers will be closed values of the pure, untyped, weak and call-by-value lambda calculus. This section summarizes those properties of the calculus which will be relevant in the rest of the paper. For more information, one can consult a recent paper by the first author and Simone Martini [11].

Λ denotes the set of *lambda terms*, defined inductively as follows:

$$M, N ::= x \mid \lambda x.M \mid MM$$

where x ranges over a denumerable set of *variables*. Given lambda terms M, N and a variable x, $M\{x/N\}$ is the lambda term obtained by substituting N for every free occurrence of x in M (see [3] for more details). The *size* $|M|$ of a term M is defined by induction on M: $|x| = 1$, $|\lambda x.M| = |M| + 1$ and $|MN| = |M| + |N|$. *Values* are abstractions and variables. Capital letters like V, U, W range over values. We consider weak call-by-value reduction on lambda terms, i.e. we take $\to$ as the closure of

$$(\lambda x.M)V \to M\{x/V\}$$

under all applicative contexts, i.e. reduction is governed by the following rules:

$$\frac{}{(\lambda x.M)V \to M\{x/V\}} \qquad \frac{M \to N}{ML \to NL} \qquad \frac{M \to N}{LM \to LN}.$$

A *realizer* is simply a closed value, i.e. an abstraction without free occurrences of variables. Realizers are ranged over by letters like e, f, g. $\mathscr{L}$ is the set of all realizers. The application $\{e\}(f)$ of two realizers is the normal form of ef relative to weak call-by-value reduction (if such a normal form exists). Observe that $\{e\}(f)$, if it exists, is always a realizer.

$\mathscr{B} = \{0, 1\}^*$ is the set of binary strings. Letters like s, t, u range over $\mathscr{B}$. The map $\Phi : \mathscr{B} \to \mathscr{L}$ is defined by induction as follows:

$$\Phi(\varepsilon) = \lambda x.\lambda y.\lambda z.z$$

$$\Phi(0s) = \lambda x.\lambda y.\lambda z.x\Phi(s)$$
$$\Phi(1s) = \lambda x.\lambda y.\lambda z.y\Phi(s).$$

In other words, $\Phi(s)$ is the lambda term corresponding to s in a numbering scheme attributed to Scott [23].

Pairs can be easily encoded in the lambda calculus: given two realizers e and f, $\langle e, f\rangle$ is simply the realizer $g \equiv \lambda x.xef$. Observe that $|\langle e, f\rangle| = |e| + |f| + cp$, where cp is a constant not depending on e or f.

But what is the cost of computing the normal form of a lambda term (provided it exists)? For this purpose, we define a (ternary) relation $\twoheadrightarrow \subseteq \Lambda \times \mathbb{N} \times \Lambda$. In the following, we will write $M \overset{n}{\twoheadrightarrow} N$ for $(M, n, N) \in \twoheadrightarrow$. The precise definition of $\twoheadrightarrow$ follows:

$$\frac{}{M \overset{0}{\twoheadrightarrow} M} \qquad \frac{M \to N \quad n = \max\{1, |N| - |M|\}}{M \overset{n}{\twoheadrightarrow} N} \qquad \frac{M \overset{n}{\twoheadrightarrow} N \quad N \overset{m}{\twoheadrightarrow} L}{M \overset{n+m}{\twoheadrightarrow} L}\ .$$

It turns out that for every M, N, L such that L is the normal form of MN, there is exactly one integer n such that $MN \overset{n}{\twoheadrightarrow} L$ (a proof of this result can be found in [11]). So, defining $Time(\{M\}(N))$ to be just n is unambiguous. Moreover, the cost model induced by $Time(\{\cdot\}(\cdot))$ is invariant (as shown in [11]), i.e. the lambda calculus and Turing machines can simulate each other with a polynomial overhead.

The properties of this computational model can be turned into an abstract definition. Any concrete computational model satisfying this definition is acceptable, provided the notion of cost induced by the computational model is polynomially invariant in the sense of [22] (otherwise one could prove non-realistic resource bounds in the semantics).

3. Length spaces

In this section, we introduce length spaces and study their properties. The length of any element of a length space will not necessarily be a number, but rather an element of certain commutative monoids called resource monoids. This generalization will give the necessary level of abstraction to capture many different systems without losing the possibility of deducing soundness from the definition of the model.

3.1. Resource monoids

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

(i) $(|M|, +)$ is a commutative monoid; in particular, there is an element 0 of $|M|$ which is an identity for $+$.
(ii) $\leq$ is a pre-order on $|M|$ which is compatible with $+$;
(iii) $\mathcal{D} : \{(\alpha, \beta) \in |M| \times |M| \mid \alpha \leq \beta\} \to \mathbb{N}$ is a function such that for every α, β, γ

$$\mathcal{D}(\alpha, \beta) + \mathcal{D}(\beta, \gamma) \leq \mathcal{D}(\alpha, \gamma)$$
$$\mathcal{D}(\alpha, \beta) \leq \mathcal{D}(\alpha + \gamma, \beta + \gamma)$$

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

Given a resource monoid $M = (|M|, +, \leq, \mathcal{D})$, the function $\mathcal{F} : |M| \to \mathbb{N}$ is defined by the equation $\mathcal{F}(\alpha) = \mathcal{D}(0, \alpha)$. We abbreviate $\sigma + \cdots + \sigma$ (n times) as $n.\sigma$. The value $\mathcal{D}(\alpha, \beta)$ should be thought of as the distance between α and β. This way, $\mathcal{F}(\alpha)$ is a measure of how big α is.

The intent of these axioms is as follows. We shall use elements of a resource monoid to bound (the size of) realizers and runtimes in the following way: an element φ bounds a realizer e if $\mathcal{F}(\varphi) \geq |e|$ and, more importantly, whenever α bounds an input f to e, there must be a third element $\beta \leq \varphi + \alpha$ bounding the result $y = \{e\}(f)$ and, most importantly, the runtime $Time(\{e\}(f))$ of that computation must be bounded by $\mathcal{D}(\beta, \varphi + \alpha)$. So, in a sense, we have the option of either producing a large output fast or to take a long time for a small output. The "inverse triangular" law above ensures that the composition of two algorithms bounded by φ_1 and φ_2, respectively, can be bounded by $\varphi_1 + \varphi_2$ or a simple modification thereof. In particular, the contribution of the unknown intermediate result in a composition cancels out using that law. Another useful intuition is that $\mathcal{D}(\alpha, \beta)$ behaves like the difference $\beta - \alpha$, indeed, $(\beta - \alpha) + (\gamma - \beta) \leq \gamma - \alpha$.

Lemma 1. *If $M = (|M|, +, \leq, \mathcal{D})$ is a resource monoid, then $\mathcal{D}$ is non-increasing in its first argument and non-decreasing in its second argument.*

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

$$\mathcal{D}(\alpha, \gamma) \geq \mathcal{D}(\alpha, \beta) + \mathcal{D}(\beta, \gamma) \geq \mathcal{D}(\beta, \gamma);$$
$$\mathcal{D}(\gamma, \alpha) \leq \mathcal{D}(\gamma, \alpha) + \mathcal{D}(\alpha, \beta) \leq \mathcal{D}(\gamma, \beta). \quad \square$$

3.1.1. Examples

We now give some examples of resource monoids. Please observe that the carrier of any resource monoid must be an infinite set: otherwise the axioms on $\mathcal{D}$ would not be satisfied because the range of $\mathcal{D}$ is bounded whenever $|M|$ is finite.

As a consequence, the simplest resource monoid is maybe $\mathcal{N} = (\mathbb{N}, +, \leq, \mathcal{D})$ where $\mathbb{N}$ is the set of natural numbers, $+$ and $\leq$ have their usual meaning and $\mathcal{D}(n, m)$ is simply $m - n$. $\mathcal{N}$ can be easily proved to satisfy the axioms above: in particular, $\mathcal{D}(n, m) + \mathcal{D}(m, l) = \mathcal{D}(n, l)$ and $\mathcal{D}(n, m) = \mathcal{D}(n + l, m + l)$.

The resource monoid $\mathcal{N}$ can be slightly generalized as follows: for every natural number k, let $\mathcal{N}^k$ be $(\mathbb{N}^k, +, \leq, \mathcal{D}^k)$ where $+$ and $\leq$ are defined pointwise, while $\mathcal{D}^k((n_1, \ldots, n_k), (m_1, \ldots, m_k))$ is the sum $\sum_{i=1}^{k}(m_i - n_i)$.

More interesting resource monoids can be constructed whose carrier contains *functions* on natural numbers, together with natural numbers themselves. Examples are the resource monoids $\mathcal{E}$, $\mathcal{S}$ and $\mathcal{M}$ that will be introduced in the following sections. Elements of $|\mathcal{S}|$, as an example, are pairs (n, f) where $n \in \mathbb{N}$ and $f : \mathbb{N} \to \mathbb{N}$ is a non-decreasing function bounded by a polynomial.

3.2. The category of realizability interpretations

In our framework, types (formulas) will be interpreted as length spaces, while proofs will be interpreted as morphisms between length spaces.

A *length space* on a resource monoid $M = (|M|, +, \leq, \mathcal{D})$ is a pair $A = (|A|, \Vdash_A)$, where $|A|$ is a set and $\Vdash_A \subseteq |M| \times \mathscr{L} \times |A|$ is a (infix) relation satisfying the following conditions:

(i) If $\alpha, e \Vdash_A a$, then $\mathcal{F}(\alpha) \geq |e|$;
(ii) If $\alpha, e \Vdash_A a$ and $\alpha \leq \beta$, then $\beta, e \Vdash_A a$.

Informally, $\alpha, e \Vdash_A a$ means that a is an element of $|A|$ realized by e, itself *majorized* by α, i.e., $\mathcal{F}(\alpha) \geq |e|$.

A *morphism* from length space $A = (|A|, \Vdash_A)$ to length space $B = (|B|, \Vdash_B)$ (on the same resource monoid $M = (|M|, +, \leq, \mathcal{D})$) is a function $f : |A| \to |B|$ such that there exist $e \in \mathscr{L}$, $\varphi \in |M|$ with $\mathcal{F}(\varphi) \geq |e|$ and whenever $\alpha, d \Vdash_A a$, there must be β such that the following four conditions hold:

(i) $\beta, c \Vdash_B f(a)$, where $c = \{e\}(d)$;
(ii) $\beta \leq \varphi + \alpha$;
(iii) $\mathit{Time}(\{e\}(d)) \leq \mathcal{D}(\beta, \varphi + \alpha)$.

We call e a *realizer* of f and φ a *majorizer* of f. The set of all morphisms from A to B is denoted as $\mathit{Morph}(A, B)$. 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 $\varphi, e \Vdash_{A \multimap B} f$.

Remark 2. It is possible to alter the time bound in the definition of a morphism to $\mathit{Time}(\{e\}(d)) \leq \mathcal{D}(\beta, \varphi + \alpha)\mathcal{F}(\alpha + \varphi)$. This allows one to accommodate linear time operations by padding the majorizer for the morphism. All the subsequent proofs go through with this alternative definition, at the expense of simplicity and ease of presentation.

Independently on the underlying resource monoid, length spaces form a symmetric monoidal closed category, as we are going to show.

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 $\langle f, g\rangle, \alpha \Vdash_{A\otimes B} (a, b)$ iff $\mathcal{F}(\alpha) \geq |\langle f, g\rangle|$ and there are β, γ with

$$\beta, f \Vdash_A a;$$
$$\gamma, g \Vdash_B b;$$
$$\alpha \geq \beta + \gamma.$$

The structure $A \otimes B$ 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 $A \Rightarrow B$ is the set of functions from A to B and $e, \alpha \Vdash_{A\multimap B} f$ iff f is a morphism from A to B realized by e and majorized by α.

Morphisms can be composed and composition is itself a morphism:

Lemma 3 (*Composition*)**.** *Given length spaces A, B, C, there is a morphism*

$$\mathit{comp} : (B \multimap C) \otimes (A \multimap B) \to (A \multimap C)$$

such that $\mathit{comp}(f, g) = \lambda x.f(g(x))$.

Proof. Let $f : B \xrightarrow{x,\varphi} C$, $g : A \xrightarrow{y,\psi} B$ and let $e_{\mathit{comp}} \equiv \lambda w.w(\lambda x.\lambda y.\lambda z.x(yz))$. There are constants p, q, r such that $\{e_{\mathit{comp}}\}(\langle x, y\rangle) = z$ where $|z| \leq |x| + |y| + p$ and $\{z\}(w) = \{x\}(\{y\}(w))$; moreover, $\mathit{Time}(\{e_{\mathit{comp}}\}(\langle x, y\rangle)) \leq r$ and $\mathit{Time}(\{e_{\mathit{comp}}\}(w)) = \mathit{Time}(\{y\}(w)) + \mathit{Time}(\{x\}(\{y\}(w))) + q$. Let us now choose μ such that $\mathcal{F}(\mu) \geq p + q$. We will prove that $\mathit{comp}(f, g) : A \xrightarrow{z,\varphi+\psi+\mu} C$. Obviously, $\mathcal{F}(\varphi + \psi + \mu) \geq |z|$. If $\alpha, w \Vdash_A a$, then there must be β, t such that $\beta, t \Vdash_B f(a)$ 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:

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

and

$$\begin{aligned} \mathit{Time}(\{z\}(w)) &\leq \mathit{Time}(\{y\}(w)) + \mathit{Time}(\{x\}(t)) + q \\ &\leq \mathcal{D}(\beta, \alpha + \psi) + \mathcal{D}(\gamma, \beta + \varphi) + \mathcal{F}(\mu) \\ &\leq \mathcal{D}(\beta + \varphi, \alpha + \psi + \varphi) + \mathcal{D}(\gamma, \beta + \varphi) + \mathcal{D}(0, \mu) \\ &\leq \mathcal{D}(\gamma, \alpha + \psi + \varphi + \mu). \end{aligned}$$

This concludes the proof, since $\mathit{comp} : (B \multimap C) \otimes (A \multimap B) \xrightarrow{e_{comp}, \xi+\mu} A \multimap C$ where ξ is such that $\mathcal{F}(\xi) \geq r + |e_{comp}|$. □

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

Lemma 4 (*Basic Maps*). *Given length spaces A, B, C, there are morphisms:*

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

where

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

Proof. Let $e_{id} \equiv \lambda x.x.\ \{e_{id}\}(d)$ takes constant time, say at most p. Then, let $\varphi_{id} \in |M|$ be such that $\mathcal{F}(\varphi_{id}) \geq p + |e_{id}|$ (this can always be done). Now, let $\alpha, d \Vdash_A a$. We have that $\alpha, d \Vdash_A \mathit{id}(a), \alpha \leq \alpha + \varphi_{id}, \{e_{id}\}(d) = d$. Moreover

$$\begin{aligned} \mathit{Time}(\{e_{id}\}(d)) &\leq p \leq \mathcal{F}(\varphi_{id}) = \mathcal{D}(0, \varphi_{id}) \\ &\leq \mathcal{D}(\alpha, \alpha + \varphi_{id}). \end{aligned}$$

This proves *id* to be a morphism realized by e_{id} and majorized by φ_{id}.

Let $e_{swap} \equiv \lambda x.x(\lambda y.\lambda w.\lambda z.zwy).\ \{e_{swap}\}(\langle d, c\rangle)$ takes constant time, say at most p. Then, let $\varphi_{swap} \in |M|$ be such that $\mathcal{F}(\varphi_{id}) \geq p + |e_{swap}|$. Now, let $\alpha, e \Vdash_{A\otimes B} (a, b)$. This implies $e = \langle d, c\rangle$ and $\alpha, \langle c, d\rangle \Vdash_{B\otimes A} (b, a)$. We can then apply the same argument as for *id*. In particular:

$$\begin{aligned} \mathit{Time}(\{e_{swap}\}(e)) &\leq p \leq \mathcal{F}(\varphi_{swap}) = \mathcal{D}(0, \varphi_{swap}) \\ &\leq \mathcal{D}(\alpha, \alpha + \varphi_{swap}). \end{aligned}$$

This proves *swap* to be a morphism realized by e_{swap} and majorized by φ_{swap}. We can verify *assl* to be a morphism exactly in the same way.

Let $e_{eval} \equiv \lambda x.x(\lambda y.\lambda w.yw)$. We can easily verify that $\{e_{eval}\}(\langle d, c\rangle) = \{d\}(c)$ and that $\{e_{eval}\}(\langle d, c\rangle)$ takes constant overload time, say at most p. φ_{eval} is chosen as to satisfy $\mathcal{F}(\varphi_{eval}) \geq p$. 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

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

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

(i) $\delta, h \Vdash_B f(a)$ and $h = \{c\}(d)$;
(ii) $\delta \leq \beta + \gamma$;
(iii) $\mathit{Time}(\{c\}(d)) \leq \mathcal{D}(\delta, \beta + \gamma)$.

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

$$\begin{aligned} \mathit{Time}(\{e_{\mathit{eval}}\}(\langle d, c\rangle)) &\leq p + \mathit{Time}(\{c\}(d)) \leq \mathcal{F}(\varphi_{\mathit{eval}}) + \mathcal{D}(\delta, \beta + \gamma) \\ &\leq \mathcal{F}(\varphi_{\mathit{eval}}) + \mathcal{D}(\delta, \beta + \gamma) + \mathcal{D}(\beta + \gamma, \alpha) \\ &\leq \mathcal{D}(0, \varphi_{\mathit{eval}}) + \mathcal{D}(\delta, \alpha) \\ &\leq \mathcal{D}(\delta, \alpha + \varphi_{\mathit{eval}}). \end{aligned}$$

As a consequence, *eval* is a morphism, too.

Now, let e_{curry} be the realizer $\lambda x.\lambda y.\lambda w.x(\lambda z.zyw)$. First of all, there must be constants p, q, r, s, t such that, for each e, x, y, there are d and c_x with

$$\begin{aligned} &\mathit{Time}(\{e_{\mathit{curry}}\}(e)) \leq p \\ &d = \{e_{\mathit{curry}}\}(e) \\ &|d| \leq |e| + q \\ &\mathit{Time}(\{d\}(x)) \leq r \\ &c_x = \{d\}(x) \\ &|c_x| \leq |e| + |x| + s \\ &\mathit{Time}(\{c_x\}(y)) \leq \mathit{Time}(\{e\}(\langle x, y\rangle)) + t \\ &\{e\}(\langle x, y\rangle) = \{c_x\}(y). \end{aligned}$$

Let $\xi, \mu, \sigma, \theta, \eta, \chi \in |M|$ be such that

$$\begin{aligned} &\mathcal{F}(\xi) \geq p \\ &\mathcal{F}(\mu) \geq q \\ &\mathcal{F}(\sigma) \geq r \\ &\mathcal{F}(\theta) \geq s \\ &\mathcal{F}(\eta) \geq t \\ &\mathcal{F}(\chi) \geq cp. \end{aligned}$$

Let now $\gamma, e \Vdash_{A \otimes B \multimap C} f$. We know that $|d| \leq |e| + q$ and $\mathit{Time}(\{e_{\mathit{curry}}\}(e)) \leq p$. In order to prove that *curry* is indeed a morphism realized by e_{curry} and majorized by $\mu + \xi + \sigma + \theta + \chi + \eta$, it suffices to prove that

$$\gamma + \mu + \sigma + \theta + \chi + \theta, d \Vdash_{A \multimap B \multimap C} \lambda a.\lambda b.f(a, b).$$

Let then $\alpha, x \Vdash_A a$. There is c_x such that $c_x = \{d\}(x)$, $|c_x| \leq |e| + |x| + s$ and $\mathit{Time}(\{d\}(x)) \leq r$. In order to prove that $\lambda a.\lambda b.f(a, b)$ is indeed a morphism realized by d and majorized by $\gamma + \mu + \sigma + \theta + \chi + \eta$, it suffices to prove that $\gamma + \alpha + \mu + \theta + \chi + \eta, c_x \Vdash_{B \multimap C} \lambda b.f(a, b)$. Let then $\beta, y \Vdash_B b$. There are δ, c such $\delta, c \Vdash_C f(a, b)$, where $\delta \leq \alpha + \beta + \chi + \gamma$. Moreover, we know that

$$\begin{aligned} \mathit{Time}(\{c_x\}(y)) &\leq \mathit{Time}(\{e\}(\langle x, y\rangle)) + t \leq \mathcal{D}(\delta, \alpha + \beta + \chi + \gamma) + t \\ &\leq \mathcal{D}(\delta, \alpha + \beta + \gamma + \chi) + \mathcal{D}(0, \eta + \mu + \theta) \\ &\leq \mathcal{D}(\delta, \alpha + \beta + \gamma + \chi + \eta + \mu + \theta). \end{aligned}$$

This concludes the proof. □

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

Lemma 5 (*Tensor*)**.** *Given length spaces A, B, C, there is a morphism*

$$\mathit{tens} : (A \multimap B) \to ((A \otimes C) \multimap (B \otimes C))$$

where $\mathit{tens}(f) = \lambda x.(f(\pi_1(x)), \pi_2(x))$.

Proof. Let $f : A \xrightarrow{x,\varphi} B$ and let e_{tens} be the realizer $\lambda x.\lambda y.y(\lambda z.\lambda q.(\lambda y.\lambda w.wyq)(xz))$. There are constants p, q, r such that $\{e_{\mathit{tens}}\}(x) = y$ where $|y| \leq |x| + p$ and $\{y\}(\langle z, w\rangle) = \langle\{x\}(z), w\rangle$; moreover, $\mathit{Time}(\{e_{\mathit{tens}}\}(x)) \leq q$ and $\mathit{Time}(\{y\}(\langle z, w\rangle)) \leq \mathit{Time}(\{x\}(z)) + r$. Then, take $\psi \in |M|$ such that $\mathcal{F}(\psi) \geq p + r$, put $\sigma = \psi + \varphi + \mu$, where $\mathcal{F}(\mu) \geq cp$. Suppose $\alpha, \langle z, w\rangle \Vdash_{A \otimes C} (a, c)$. By definition, there are β, γ such that

$$\begin{aligned} &\beta, z \Vdash_A a \\ &\gamma, w \Vdash_C c \\ &\alpha \geq \beta + \gamma. \end{aligned}$$

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.

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

Fig. 1. Intuitionistic multiplicative affine logic.

By hypothesis, there are δ, t such that

$$\begin{aligned}
&\delta, t \Vdash_B f(a)\\
&\delta \leq \varphi + \beta\\
&\{e\}(z) = t\\
&\mathit{Time}(\{e\}(z)) \leq \mathcal{D}(\delta, \varphi + \beta).
\end{aligned}$$

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

$$\gamma + \delta + \mu \leq \gamma + \varphi + \beta + \mu \leq \alpha + \varphi + \mu \leq \alpha + \sigma.$$

Finally:

$$\begin{aligned}
\mathit{Time}(\{y\}(\langle z, w\rangle)) &\leq \mathit{Time}(\{x\}(z)) + r\\
&\leq \mathcal{D}(\delta, \varphi + \beta) + \mathcal{F}(\psi)\\
&\leq \mathcal{D}(\delta, \varphi + \beta + \psi)\\
&\leq \mathcal{D}(\gamma + \delta + \mu, \gamma + \varphi + \beta + \mu + \psi)\\
&= \mathcal{D}(\gamma + \delta + \mu, \gamma + \beta + \sigma)\\
&\leq \mathcal{D}(\gamma + \delta + \mu, \alpha + \sigma).
\end{aligned}$$

This concludes the proof, since $\mathit{tens} : (A \multimap B) \xrightarrow{(f,g),\xi+\psi+\mu} (A \otimes C) \multimap (B \otimes C)$ where ξ is such that $\mathcal{F}(\xi) \geq q + |e_{\mathit{tens}}|$. □

Thus:

Lemma 6. *Length spaces and their morphisms form a symmetric monoidal closed category with tensor and linear implication given as above.*

For every resource monoid M, a length space I_M is defined by $|I_M| = \{0\}$ and $\alpha, \lambda x.x \Vdash_{I_M} 0$ when $\mathcal{F}(\alpha) \geq |\lambda x.x|$. 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 $\mathscr{B}_M = (\{0,1\}^*, \Vdash_{B_M})$ where $\alpha, \Phi(t) \Vdash_{\mathscr{B}_M} t$ whenever $\mathcal{F}(\alpha) \geq |\Phi(t)|$. 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 constant time and, as a consequence, is a morphism from $\mathscr{B}_M$ to itself. Moreover, the function $\varepsilon : \{0\} \to \mathscr{B}$ which returns the empty string is itself morphism from I_M to $\mathscr{B}_M$.

3.3. Interpreting multiplicative affine logic

We can now formally show that second order multiplicative affine logic (i.e. multiplicative linear logic plus full weakening, MAL) can be interpreted inside the category of length spaces on any monoid M. This will simplify the analysis of richer systems presented in the next three sections, since they all are extensions of MAL. Formulas 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*. *Sequents* have the form $\Gamma \vdash A$ where A is a formula and the *context* Γ is a multiset of formulas. As usual, contexts are denoted as sequences of formulas, i.e., expressions in the form $A_1, \ldots, A_n$. The context's elements are the formulas appearing in the sequence, while the number of occurrences of any formula gives the multiplicity of the formula in the context.

Rules for MAL are as in Fig. 1. A MAL proof π is simply a tree built from MAL rules. The size $|\pi|$ of any MAL proof π is the number of sequent occurrences in π.

In MAL, the second order existential quantification is defined, as in second order intuitionistic logic [13], i.e., one can define $\exists\alpha.A$ as $\forall\beta.(\forall\alpha.A \multimap \beta) \multimap \beta$. With this definition, the usual rules for existential quantification are both available. Moreover, the presence of full weakening allows to faithfully encode the so-called additive connectives inside MAL:

$$A\&B \equiv \exists\alpha.((\alpha \multimap A) \otimes (\alpha \multimap B) \otimes \alpha);$$
$$A \oplus B \equiv \forall\alpha.((A \multimap \alpha) \multimap (B \multimap \alpha) \multimap \alpha).$$

Actually, the tensor product $\otimes$ could be itself defined in terms of second order quantification and linear implication:

$$A \otimes B \equiv \forall\alpha.((A \multimap B \multimap \alpha) \multimap \alpha).$$

Set-theoretic and realizability interpretations of MAL's formulas and proofs will be now introduced. They are essential tools when proving soundness theorems for MAL's extensions. A *set-theoretic environment* is a partial function assigning sets to atoms. Given a formula A and a set-theoretic environment η, we can define a set $[\![A]\!]^{\mathscr{S}}_{\eta}$ by induction on the structure of A as follows:

$$[\![\alpha]\!]^{\mathscr{S}}_{\eta} = \eta(\alpha)$$
$$[\![A \otimes B]\!]^{\mathscr{S}}_{\eta} = [\![A]\!]^{\mathscr{S}}_{\eta} \times [\![B]\!]^{\mathscr{S}}_{\eta}$$
$$[\![A \multimap B]\!]^{\mathscr{S}}_{\eta} = [\![A]\!]^{\mathscr{S}}_{\eta} \Rightarrow [\![B]\!]^{\mathscr{S}}_{\eta}$$
$$[\![\forall\alpha.A]\!]^{\mathscr{S}}_{\eta} = \prod_{C\in\mathscr{U}} [\![A]\!]^{\mathscr{S}}_{\eta[\alpha\to C]}.$$

Here $\mathscr{U}$ stands for the class of all length spaces. If the underlying set theory is classical, $\mathscr{U}$ cannot exist. However, we follow [17] and assume to work in constructive set theory. A more detailed discussion on this problem can be found in Section 3.4. If $n \geq 0$ and $A_1, \ldots, A_n$ are formulas, the expression $[\![A_1 \otimes \cdots \otimes A_n]\!]^{\mathscr{S}}_{\eta}$ stands for I_M if $n = 0$ and $[\![A_1 \otimes \cdots \otimes A_{n-1}]\!]^{\mathscr{S}}_{\eta} \otimes [\![A_n]\!]^{\mathscr{S}}_{\eta}$ if $n \geq 1$. Given a MAL proof π of $A_1, \ldots, A_n \vdash B$ and a set-theoretic environment η, a function

$$[\![\pi]\!]^{\mathscr{S}}_{\eta} : [\![A_1 \otimes \cdots \otimes A_n]\!]^{\mathscr{S}}_{\eta} \to [\![B]\!]^{\mathscr{S}}_{\eta}$$

can be easily defined following the structure of π.

A *realizability environment* for a resource monoid M is a partial function assigning length spaces (on M) to atoms. Given a realizability environment η, $|\eta|$ is the set-theoretic environment returning the carrier of $\eta(\alpha)$ on argument α (provided $\eta(\alpha)$ is defined). Realizability semantics $[\![A]\!]^{\mathscr{R}}_{\eta}$ of a formula A on the realizability environment η is defined by induction on A:

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

where $\otimes$ and $\multimap$ are constructions on length spaces defined in Section 3.2 and $\alpha, e \Vdash_{[\![\forall\alpha.A]\!]^{\mathscr{R}}_{\eta}} a$ iff for every length space C, $\alpha, e \Vdash_{[\![A]\!]^{\mathscr{R}}_{\eta[\alpha\to C]}} a$. Observe that $|[\![A]\!]^{\mathscr{R}}_{\eta}| = [\![A]\!]^{\mathscr{S}}_{|\eta|}$.

Given a MAL proof π of $A_1, \ldots, A_n \vdash B$ and a realizability environment η, we can prove that $[\![\pi]\!]^{\mathscr{S}}_{|\eta|}$ is a morphism from $[\![A_1 \otimes \cdots \otimes A_n]\!]^{\mathscr{R}}_{\eta}$ to $[\![B]\!]^{\mathscr{R}}_{\eta}$. The proof goes by induction on the structure of π. Notice that the result holds independently on the underlying resource monoid, since the main ingredients for the proof (the lemmas from Section 3.2) hold for every resource monoid M. Formally:

Theorem 7. *For every resource monoid M there is a polynomial $p_M : \mathbb{N} \to \mathbb{N}$ such that the following holds. Let π be a* MAL *proof of a sequent $A_1, \ldots, A_n \vdash B$. Let η be a realizability environment assigning length spaces on M to all atoms appearing free in the sequent. Then the function $[\![\pi]\!]^{\mathscr{S}}_{|\eta|}$ is a morphism (of length spaces on M) from $[\![A_1 \otimes \cdots \otimes A_n]\!]^{\mathscr{R}}_{\eta}$ to $[\![B]\!]^{\mathscr{R}}_{\eta}$ majorized by some $\alpha_\pi \in |M|$, where $\mathcal{F}(\alpha_\pi) \leq p_M(|\pi|)$.*

Please observe that semantics is preserved by reduction: whenever π reduces to ρ then the set-theoretic semantics $[\![\pi]\!]^{\mathscr{S}}_{\eta}$ equals $[\![\rho]\!]^{\mathscr{S}}_{\eta}$. And, clearly, if $[\![\pi]\!]^{\mathscr{S}}_{\eta}$ is realized by e and majorized by α, then $[\![\rho]\!]^{\mathscr{S}}_{\eta}$ will be realized by the same e and majorized by the same α.

3.4. On the underlying set theory

In this brief section we discuss the use of a constructive metatheory along the lines of [17].

Recall that we assumed the existence of a universe $\mathscr{U}$ in our ambient set theory which is closed under $\mathscr{U}$-indexed products. As is well-known, no such universe exists in classical ZF set theory but its existence is consistent with constructive

set theories [18,21]. Assuming the existence of such a $\mathscr{U}$ is convenient, because it allows the use of informal set-theoretic arguments (provided they are constructive).

The logical systems we will study in this paper are all (essentially) subsystems of intuitionistic affine logic (IAL for short), i.e., intuitionistic multiplicative and exponential linear logic with free weakening. IAL proofs can be faithfully embedded into second order intuitionistic logic and the embedding is reduction-preserving. As a consequence, results about the possibility of handling second order quantification in a constructive setting [21] directly translate to equivalent results about the logical systems we treat here.

We here want to stress that our objective is *not* defining set-theoretic semantics for various subsystems of linear logic, even if set-theoretic semantics is necessary to pursue our main goal. Our focus is on quantitative properties of proofs and programs, which are not captured by standard set-theoretic semantics, but which are revealed by realizers and majorizers in our realizability framework. Noticeably, both realizers and majorizers are absolutely harmless from a set-theoretical point of view, even in a constructive setting: majorizers are lambda terms (i.e., terms in an inductively defined language), while resource monoids will be defined and reasoned about using natural numbers and their elementary properties.

For the reader who feels uneasy about our choice, we offer the following ways of making our arguments rigorous (all of which, however, complicate the presentation):

- Formalize the entire discussion in the Calculus of Inductive Constructions [7].
- Formalize the entire discussion in a realizability topos [21].
- Make the previous point explicit by stipulating that the carrier sets of realizability sets must be a subquotient (by a partial equivalence relation) of the set of untyped lambda terms (not necessarily call-by-value!). Morphisms between realizability sets are then required to be uniformly tracked by an untyped lambda term in the obvious sense. Doing so allows one to interpret polymorphic quantification as intersection of partial equivalence relations.

4. Elementary length spaces and EAL

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

Before going into the details, we need to introduce some useful notation: given two natural numbers $n, m \in \mathbb{N}$, $n+m \in \mathbb{N}$ is their sum, while $n|m \in \mathbb{N}$ is their maximum. Similarly for functions: given $f, g : \mathbb{N} \to \mathbb{N}$, $f+g$ (respectively, $f|g$) is the sum (respectively, the maximum) of f and g, defined pointwise:

$$(f+g)(n) = f(n) + g(n)$$
$$(f|g)(n) = f(n)|g(n).$$

The resource monoid $\mathcal{E} = (|\mathcal{E}|, +, \leq_{\mathcal{E}}, \mathcal{D}_{\mathcal{E}})$ is defined as follows:

- Elements of $|\mathcal{E}| \subseteq \mathbb{N} \times \mathbb{N} \times \mathbb{N}^{\mathbb{N}}$ are triples (n, m, f) such that $f : \mathbb{N} \to \mathbb{N}$ is a monotonically increasing elementary function.
- For every $(n, m, f), (l, k, g) \in |\mathcal{E}|$, $(n, m, f) + (l, k, g) = (n + l, m + k, f|g)$.
- For every $(n, m, f), (l, k, g) \in |\mathcal{E}|$, $(n, m, f) \leq_{\mathcal{E}} (l, k, g)$ iff $n \leq l$, $2^n m \leq 2^l k$ and $f \leq g$ pointwise.
- For every $(n, m, f), (l, k, g) \in |\mathcal{E}|$ such that $(n, m, f) \leq_{\mathcal{E}} (l, k, g)$,

$$\mathcal{D}_{\mathcal{E}}((n, m, f), (l, k, g)) = (l - n)g(l + 2^l k).$$

Observe that m and f do not appear in the expression defining $\mathcal{D}_{\mathcal{E}}((n, m, f), (l, k, g))$.

The triple $(0, 0, 0) \in |\mathcal{E}|$, denoted $0_{\mathcal{E}}$, is an identity for $+$.

We now need to prove that the structure $\mathcal{E}$ is indeed a resource monoid, i.e., that it satisfies all the required axioms from Section 3. First of all, observe that $(|\mathcal{E}|, +)$ is indeed a commutative monoid. The axioms concerning $\leq_{\mathcal{E}}$ are slightly more complicated to prove:

Lemma 8 (*Compatibility*). *$0_{\mathcal{E}} \leq_{\mathcal{E}} \alpha$ for every $\alpha \in |\mathcal{E}|$. Moreover, if α, β, γ are in $|\mathcal{E}|$ and $\alpha \leq_{\mathcal{E}} \beta$, then $\alpha + \gamma \leq_{\mathcal{E}} \beta + \gamma$.*

Proof. Let $(n, m, f) \in |\mathcal{E}|$. Clearly, $0 \leq n$, $2^0 0 = 0 \leq 2^n m$ and $0 \leq f$. As a consequence, $0_{\mathcal{E}} \leq_{\mathcal{E}} (n, m, f)$. Now, let $(n, m, f), (l, k, g), (p, q, h) \in |\mathcal{E}|$ and let $(n, m, f) \leq_{\mathcal{E}} (l, k, g)$. This implies $n \leq l$, $2^n m \leq 2^l k$ and $f \leq g$. Then:

$$\begin{aligned} & n + p \leq l + p; \\ & 2^{n+p}(m+q) = 2^n 2^p (m+q) = 2^n 2^p m + 2^n 2^p q \\ & \qquad \leq 2^l 2^p k + 2^n 2^p q \\ & \qquad \leq 2^l 2^p k + 2^l 2^p q \\ & \qquad = 2^l 2^p (k+q) = 2^{l+p}(k+q); \\ & f|h \leq g|h. \end{aligned}$$

This implies $(n, m, f) + (p, q, h) \leq_{\mathcal{E}} (l, k, g) + (p, q, h)$. □

Lemma 9 (*Transitivity*). *If α, β, γ are in $|\mathcal{E}|$, $\alpha \preceq_{\mathcal{E}} \beta$ and $\beta \preceq_{\mathcal{E}} \gamma$, then $\alpha \preceq_{\mathcal{E}} \gamma$.*

Proof. Let $(n, m, f), (l, k, g), (p, q, h) \in |\mathcal{E}|$ and let $(n, m, f) \preceq_{\mathcal{E}} (l, k, g)$, $(l, k, g) \preceq_{\mathcal{E}} (p, q, h)$. Trivially:

$$n \leq l \leq p$$
$$2^n m \leq 2^l k \leq 2^p q$$
$$f \leq g \leq h.$$

In other words $(n, m, f) \preceq_{\mathcal{E}} (p, q, h)$. □

Lemma 10. *If α, β, γ are in $|\mathcal{E}|$ and $\alpha \preceq_{\mathcal{E}} \beta$, then $\mathcal{D}_{\mathcal{E}}(\alpha, \beta) \leq \mathcal{D}_{\mathcal{E}}(\alpha + \gamma, \beta + \gamma)$.*

Proof. Let $(n, m, f), (l, k, g), (p, q, h) \in |\mathcal{E}|$ and let $(n, m, f) \preceq_{\mathcal{E}} (l, k, g)$. Trivially:

$$\begin{aligned}
\mathcal{D}_{\mathcal{E}}((n, m, f) + (p, q, h), (l, k, g) + (p, q, h)) &= \mathcal{D}_{\mathcal{E}}((n + p, m + q, f|h), (l + p, k + q, g|h)) \\
&= ((l + p) - (n + p))(g|h)(l + p + 2^{l+p}(k + q)) \\
&\geq (l - n)g(l + 2^l k) \\
&= \mathcal{D}_{\mathcal{E}}((n, m, f), (l, k, g)).
\end{aligned}$$

This concludes the proof. □

Finally, we need to prove the anti-triangular property for $\mathcal{D}_{\mathcal{E}}$.

Lemma 11. *If α, β, γ are in $|\mathcal{E}|$, $\alpha \preceq_{\mathcal{E}} \beta$ and $\beta \preceq_{\mathcal{E}} \gamma$, then $\mathcal{D}_{\mathcal{E}}(\alpha, \beta) + \mathcal{D}_{\mathcal{E}}(\beta, \gamma) \leq \mathcal{D}_{\mathcal{E}}(\alpha, \gamma)$.*

Proof. Let $(n, m, f), (l, k, g), (p, q, h) \in |\mathcal{E}|$ and let $(n, m, f) \preceq_{\mathcal{E}} (l, k, g)$, $(l, k, g) \preceq_{\mathcal{E}} (p, q, h)$. Trivially:

$$\begin{aligned}
\mathcal{D}_{\mathcal{E}}((n, m, f), (p, q, h)) &= (p - n)h(p + 2^p q) \\
&= ((p - l) + (l - n))h(p + 2^p q) \\
&= (p - l)h(p + 2^p q) + (l - n)h(p + 2^p q) \\
&\geq (p - l)h(p + 2^p q) + (l - n)g(p + 2^p q) \\
&\geq (p - l)h(p + 2^p q) + (l - n)g(l + 2^l k) \\
&= \mathcal{D}_{\mathcal{E}}((n, m, f), (l, k, g)) + \mathcal{D}_{\mathcal{E}}((l, k, g), (p, q, h)).
\end{aligned}$$

This concludes the proof. □

Lemma 12. *$\mathcal{E}$ is a resource monoid.*

Proof. $(|\mathcal{E}|, +)$ is certainly a commutative monoid. Compatibility of $\preceq_{\mathcal{E}}$ follows from Lemmas 8 and 9. The two required properties on $\mathcal{D}_{\mathcal{E}}$ come directly from Lemmas 10 and 11. If $n \in \mathbb{N}$, observe that $\mathcal{F}_{\mathcal{E}}(n, 0, x \mapsto x) = n$. This concludes the proof. □

An *elementary length space* is a length space on the resource monoid $\mathcal{E}$. Given $\alpha = (n, m, f) \in |\mathcal{E}|$, $!\alpha$ stands for $(1, n + m, f^+)$ where $f^+(x) = xf(x2^x)$ for every $x \in \mathbb{N}$. For every elementary length space $A = (|A|, \Vdash_A)$, we can build the length space $!A = (|A|, \Vdash_{!A})$, where $\alpha, e \Vdash_{!A} a$ iff there is $\beta \in |\mathcal{E}|$ such that $\beta, e \Vdash_A a$ and $!\beta \preceq_{\mathcal{E}} \alpha$. 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 corresponding precisely to the axioms and rules of EAL.

Lemma 13 (*Basic Maps*). *Given elementary length spaces A, B, there are morphisms:*

$$\mathit{contr} : !A \to !A \otimes !A$$
$$\mathit{distr} : !A \otimes !B \to !(A \otimes B)$$

where $\mathit{contr}(a) = (a, a)$ and $\mathit{distr}(a, b) = (a, b)$.

Proof. Let e_{contr} be the realizer $\lambda x.\lambda y.yxx$. Computing $\{e_{\mathit{contr}}\}(d)$ takes time $|d| + p$, where p is a constant. Then, let $\alpha, \beta \in |\mathcal{E}|$ be such that $\mathcal{F}_{\mathcal{E}}(\alpha) \geq p + |e_{\mathit{contr}}|$, $\mathcal{F}_{\mathcal{E}}(\beta) \geq cp$. Define α_{contr} to be $\alpha + \beta + (2, 0, x \mapsto 0)$. Clearly, $\mathcal{F}_{\mathcal{E}}(\alpha_{\mathit{contr}}) \geq |e_{\mathit{contr}}|$. Now, let $(n, m, f), d \Vdash_{!A} a$. This implies that $(n, m, f) \geq_{\mathcal{E}} (1, l + k, g^+) = \gamma$ where $(l, k, g), d \Vdash_A a$. Then:

$$\beta + \gamma + \gamma \geq_{\mathcal{E}} \gamma + \gamma$$
$$\begin{aligned}
\mathcal{F}_{\mathcal{E}}(\beta + \gamma + \gamma) &\geq \mathcal{F}_{\mathcal{E}}(\beta) + \mathcal{F}_{\mathcal{E}}(\gamma) + \mathcal{F}_{\mathcal{E}}(\gamma) \\
&\geq cp + \mathcal{F}_{\mathcal{E}}(\gamma) + \mathcal{F}_{\mathcal{E}}(\gamma).
\end{aligned}$$

This yields $\beta+\gamma+\gamma, e \Vdash_{!A\otimes!A} (a,a)$. But now notice that:

$$\begin{aligned}\beta+\gamma+\gamma &= \beta+(2,2(l+k),g^+)\\ &\le_{\mathcal{E}} \beta+(3,l+k,g^+)\\ &= \beta+(2,0,x\mapsto 0)+(1,l+k,g^+)\\ &\le_{\mathcal{E}} \alpha+\beta+(2,0,x\mapsto 0)+(1,l+k,g^+)\\ &= \alpha_{contr}+(1,l+k,g^+)\\ &\le_{\mathcal{E}} \alpha_{contr}+(n,m,f).\end{aligned}$$

Notice that

$$\begin{aligned}\mathcal{D}_{\mathcal{E}}(\gamma+\gamma,\gamma+(2,0,x\mapsto 0)) &= \mathcal{D}_{\mathcal{E}}((2,2(l+k),g^+),(3,l+k,g^+))\\ &= (3-2)g^+(3+2^3(l+k))\\ &= (3+8l+8k)g((3+8l+8k)2^{3+8l+8k})\\ &\ge l\cdot g(l\cdot 2^{3+8l+8k}+k\cdot 2^{3+8l+8k})\\ &\ge l\cdot g(l\cdot 2^0+k\cdot 2^l)\\ &= l\cdot g(l+2^l k)\\ &= \mathcal{F}_{\mathcal{E}}(l,k,g).\end{aligned}$$

Finally:

$$\begin{aligned}Time(\{e_{contr}\}(d)) &\le |d|+p\le \mathcal{F}_{\mathcal{E}}(l,k,g)+p\\ &\le \mathcal{D}_{\mathcal{E}}(\gamma+\gamma,\gamma+(2,0,x\mapsto 0))+\mathcal{F}_{\mathcal{E}}(\alpha)\\ &\le \mathcal{D}_{\mathcal{E}}(\gamma+\gamma,\gamma+(2,0,x\mapsto 0)+\alpha)\\ &\le \mathcal{D}_{\mathcal{E}}(\gamma+\gamma+\beta,\gamma+(2,0,x\mapsto 0)+\alpha+\beta)\\ &= \mathcal{D}_{\mathcal{E}}(\gamma+\gamma+\beta,\gamma+\alpha_{contr}).\end{aligned}$$

This proves *contr* to be a morphism.

Let $e_{distr}=e_{id}\equiv\lambda x.x$. We know $\{e_{id}\}(d)$ takes constant time, say p. Then, let $\alpha,\beta\in|\mathcal{E}|$ be such that $\mathcal{F}_{\mathcal{E}}(\alpha)\ge p+|e_{distr}|$, and $\beta \ge_{\mathcal{E}} (1,cp,x\mapsto x)$. α_{distr} is then defined as $\alpha+\beta$. Now, let $(n,m,f),\langle d,c\rangle \Vdash_{!A\otimes!B} (a,b)$. This means that $(n,m,f)\ge_{\mathcal{E}}(1,l+k,g^+)+(1,p+q,h^+)$, where $(l,k,g),d\Vdash_A a$ and $(p,q,h),c\Vdash_B b$. This in turn means that $(l+p+cp,k+q,g|h|(x\mapsto 1)),\langle d,c\rangle\Vdash_{A\otimes B}(a,b)$ and

$$(1,l+p+cp+k+q,(g|h|(x\mapsto 1))^+),\langle d,c\rangle\Vdash_{!(A\otimes B)}(a,b).$$

Moreover

$$\begin{aligned}(1,l+p+cp+k+q,(g|h|(x\mapsto 1))^+) &= (1,l+p+cp+k+q,g^+|h^+|(x\mapsto x))\\ &\le_{\mathcal{E}} (3,l+p+cp+k+q,g^+|h^+|(x\mapsto x))\\ &= (1,l+k,g^+)+(1,p+q,h^+)+(1,cp,x\mapsto x)\\ &\le_{\mathcal{E}} (1,l+k,g^+)+(1,p+q,h^+)+\beta\\ &\le_{\mathcal{E}} (1,l+k,g^+)+(1,p+q,h^+)+\alpha_{distr}\\ &\le_{\mathcal{E}} (n,m,f)+\alpha_{distr}.\end{aligned}$$

Finally:

$$\begin{aligned}Time(\{e_{distr}\}(\langle d,c\rangle)) &\le p\le \mathcal{F}_{\mathcal{E}}(\alpha)\\ &\le \mathcal{D}_{\mathcal{E}}((1,l+p+cp+k+q,(g|h|(x\mapsto x))^+),(n,m,f)+\beta)+\mathcal{F}_{\mathcal{E}}(\alpha)\\ &\le \mathcal{D}_{\mathcal{E}}((1,l+p+cp+k+q,(g|h|(x\mapsto x))^+),(n,m,f)+\beta+\alpha)\\ &\le \mathcal{D}_{\mathcal{E}}((1,l+p+cp+k+q,(g|h|(x\mapsto x))^+),(n,m,f)+\alpha_{distr}).\end{aligned}$$

This proves *distr* to be a morphism. □

Lemma 14 (*Functoriality*). *If* $f : A \xrightarrow{e,\alpha} B$, *then* $f : !A \xrightarrow{e,!\alpha} !B$.

Proof. Let α be (n,m,f) and suppose $d,(l,k,g)\Vdash_{!A} a$. Then $(l,k,g)\ge_{\mathcal{E}}(1,i+j,h^+)$, where $d,(i,j,h)\Vdash_A a$. Observe that there must be γ,c such that $c,\gamma\Vdash_B f(a)$, $\gamma\le_{\mathcal{E}}(n,m,f)+(i,j,h)$ and $Time(\{e\}(d))\le\mathcal{D}_{\mathcal{E}}(\gamma,(n,m,f)+(i,j,h))$. As a consequence, $c,(n,m,f)+(i,j,h)\Vdash_B f(a)$, that is to say $c,(n+i,m+j,f|h)\Vdash_B f(a)$. By definition, $c,(1,n+i+m+$

Exponential and Contraction Rules.

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

Fig. 2. Intuitionistic elementary affine logic.

$j, (f|h)^+) \Vdash_{!B} f(a)$, that is to say $c, (1, n+i+m+j, f^+|h^+) \Vdash_{!B} f(a)$. But now notice that:

$$\begin{aligned}(1, n+i+m+j, f^+|h^+) &\leq_\mathcal{E} (2, n+i+m+j, f^+|h^+)\\ &\leq_\mathcal{E} (1, n+m, f^+) + (1, i+j, h^+)\\ &\leq_\mathcal{E} (1, n+m, f^+) + (l, k, g).\end{aligned}$$

Moreover,

$$\begin{aligned}Time(\{e\}(d)) &\leq \mathcal{D}_\mathcal{E}(\gamma, (n, m, f) + (i, j, h))\\ &\leq (n+i)(f|h)(n+i+2^{n+i}(m+j))\\ &\leq (n+i+m+j+2)(f|h)(2^n + 2^i + 2^{n+i+m+j+2}(m+j))\\ &\leq (n+i+m+j+2)(f|h)(2^{n+i+m+j+2} + 2^{n+i+m+j+2} + 2^{n+i+m+j+2}(n+i+m+j))\\ &= (n+i+m+j+2)(f|h)(2^{n+i+m+j+2}(n+i+m+j+2))\\ &= (f|h)^+(n+i+m+j+2)\\ &\leq (f|h)^+(2^2(n+i+m+j)+2)\\ &= 1\cdot(f^+|h^+)(2^2(n+i+m+j)+2)\\ &= \mathcal{D}_\mathcal{E}((1, n+m+i+j, f^+|h^+), (2, n+m+i+j, f^+|h^+))\\ &= \mathcal{D}_\mathcal{E}((1, n+m+i+j, f^+|h^+), (1, n+i, f^+) + (1, m+j, h^+))\\ &\leq \mathcal{D}_\mathcal{E}((1, n+m+i+j, f^+|h^+), (1, n+m, f^+) + (i, j, h)).\end{aligned}$$

This means that $f :!A \xrightarrow{e,(1,n+m,f^+)} !B$. □

4.1. Interpreting elementary affine logic

EAL can be obtained by endowing MAL with a restricted modality. The grammar of formulas is enriched with a new production $A ::= !A$ while modal rules are as in Fig. 2.

For every EAL proof π, we define the *box depth* $\partial(\pi) \in \mathbb{N}$ of π as the maximum number of instances of the P rule on any path starting at the root of π and ending at one of its leaves.

Set-theoretic interpretations are not affected by !: $[\![!A]\!]^\mathcal{S}_\eta = [\![A]\!]^\mathcal{S}_\eta$. Realizability semantics is extended by $[\![!A]\!]^\mathcal{R}_\eta = ![\![A]\!]^\mathcal{R}_\eta$.

Elementary length spaces form a model of EAL:

Theorem 15. *For every natural number $m \in \mathbb{N}$ there is an elementary function $f_m : \mathbb{N} \to \mathbb{N}$ such that, for every* EAL *proof $\pi : A_1, \ldots, A_n \vdash B$ and for every realizability environment η assigning elementary length spaces to all atoms appearing free in the sequent, we have that*

$$[\![\pi]\!]^\mathcal{S}_{|\eta|} : [\![A_1 \otimes \cdots \otimes A_n]\!]^\mathcal{R}_\eta \xrightarrow{\varphi, e} [\![B]\!]^\mathcal{R}_\eta$$

and $\mathcal{F}_\mathcal{E}(\varphi) \leq f_{\partial(\pi)}(|\pi|)$.

Proof. The Theorem follows from Lemmas 14 and 13,together with Theorem 7. The prescribed bound on $\mathcal{F}_\mathcal{E}(\varphi)$ can be proved by observing that the only semantic construction which induces a significant increase in the "size" of the underlying majorizer is the one from Lemma 14. More formally, the proof goes by induction on π, where the only interesting inductive cases are the ones corresponding to rules P and C, since all the others follow from the results in Section 3.2. The fact that P can be justified follows from Lemmas 13 and 14; observe, in particular, that $\partial(\pi)$ increases by one whenever P is applied and, on the other hand, the underlying majorizer becomes α^+. C can be justified since *contr* is a morphism (Lemma 14) and morphisms compose. □

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 $List_{\mathsf{EAL}}$ by the usual, impredicative, encoding (see, for example, [13]). In other words, any binary list w can be put in correspondence to a proof π_w, following the so-called Church encoding. Let $BtoEAL : \mathscr{B} \to [\![List_{\mathsf{EAL}}]\!]^\mathcal{S}$ be the function mapping each binary list $s \in \mathscr{B}$ to (the denotation of) its encoding.

Exponential and Contraction Rules.

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

Fig. 3. Intuitionistic soft affine logic.

There is a function *EALtoB* from $[\![List_{\mathsf{EAL}}]\!]^{\mathscr{S}}$ to $\mathscr{B}$ which maps (the denotation of) each cut-free proof representing $w \in \mathscr{B}$ to w: $EALtoB(a)$ simply returns the application of a to ε, s_0, s_1. Actually, *EALtoB* is a morphism from $[\![List_{\mathsf{EAL}}]\!]^{\mathscr{R}}$ to $!\mathscr{B}_{\mathcal{E}}$.

Now, let π be a proof with conclusion $\vdash !^j List_{\mathsf{EAL}} \multimap !^k List_{\mathsf{EAL}}$. Any such proofs corresponds to a function f_π from $\mathscr{B}$ to itself, namely to $EALtoB \circ [\![\pi]\!]^{\mathscr{S}} \circ BtoEAL$. But f_π can be computed in elementary time. Indeed, from the denotation $[\![\pi]\!]^{\mathscr{S}}$ we can build a morphism g from $[\![!^j List_{\mathsf{EAL}}]\!]^{\mathscr{R}}$ to $!^{k+1}\mathscr{B}_{\mathcal{E}}$ by composition with *EALtoB*. This morphism then induces an algorithm computing f_π: given $w \in \mathscr{B}$, first compute a realizer for the cut-free proof of $!^j List_{\mathsf{EAL}}$ corresponding to w, then apply the result to a realizer for g. But any cut-free proof ρ of $!^j List_{\mathsf{EAL}}$ has box depth $\partial(\rho) = j + 1$, which does not depend on the particular binary list w. By Theorem 15, we get:

Corollary 16 (*Soundness*)**.** *Let π be an* EAL *proof with conclusion $\vdash !^j List_{\mathsf{EAL}} \multimap !^k List_{\mathsf{EAL}}$. Then f_π is computable in elementary time.*

5. Soft length spaces

The grammar of formulas for SAL is the same as the one of Elementary Affine Logic. Rules are the ones of MAL, plus two new rules governing the connective !, as in Fig. 3. The box depth $\partial(\pi)$ of every SAL proof π can be defined exactly as for EAL.

Elementary length spaces are not adequate for SAL, because only elementary bounds can be inferred from them. Moreover, rule *M* cannot be justified by elementary length spaces. In other words, another resource monoid is needed.

The resource monoid $\mathcal{S}$ is the quadruple $(|\mathcal{S}|, +, \leq_{\mathcal{S}}, \mathcal{D}_{\mathcal{S}})$ such that:

- Elements of $|\mathcal{S}| \subseteq \mathbb{N} \times \mathbb{N}^{\mathbb{N}}$ are pairs (n, f) such that $f : \mathbb{N} \to \mathbb{N}$ is a non-decreasing function bounded by a polynomial.
- For every $(n, f), (m, g) \in |\mathcal{S}|$, $(n, f) + (m, g) = (n|m, f + g)$.
- For every $(n, f), (m, g) \in |\mathcal{S}|$, $(n, f) \leq_{\mathcal{S}} (m, g)$ iff $n \leq m, f(x) \leq g(x)$ for every $x \geq m$ and $(g - f)(x) \leq (g - f)(y)$ whenever $y \geq x \geq m$.
- For every $(n, f), (m, g) \in |\mathcal{S}|$ such that $(n, f) \leq_{\mathcal{S}} (m, g)$,

$$\mathcal{D}_{\mathcal{S}}((n, f), (m, g)) = g(m) - f(m).$$

Observe that the expression defining $\mathcal{D}_{\mathcal{S}}((n, f), (m, g))$ does not depend on n.

The definition of $\leq_{\mathcal{S}}$ can be rephrased as follows: $(n, f) \leq_{\mathcal{S}} (m, g)$ iff the function $g - f$ is non-decreasing and non-negative for $x \geq m$. The pair $(0, 0) \in |\mathcal{S}|$, denoted $0_{\mathcal{S}}$, is an identity for $+$.

Lemma 17. *$\mathcal{S}$ is a resource monoid.*

Proof. $(|\mathcal{S}|, +)$ is certainly a monoid. Compatibility of $\leq_{\mathcal{S}}$ can be easily proved: observe, in particular that $n|l \leq m|l$ whenever $n \leq m$ and that $(g + h) - (f + h) = g - f$. The two required properties on $\mathcal{D}_{\mathcal{S}}$ follow directly from its definition. If $n \in \mathbb{N}$, observe that $\mathcal{F}_{\mathcal{S}}(n, x \mapsto x) = n$. This concludes the proof. □

A *soft length space* is a length space on the resource monoid $\mathcal{S}$. Given $\alpha = (n, f) \in |\mathcal{S}|$, $!\alpha$ stands for (n, f^+) where $f^+(x) = (x + 1)f(x)$. Given a soft length space $A = (|A|, \Vdash_A)$, we can build the length space $!A = (|A|, \Vdash_{!A})$, where $\alpha, e \Vdash_{!A} a$ iff there is $\beta \in |\mathcal{S}|$ such that $\beta, e \Vdash_A a$ and $!\beta \leq_{\mathcal{S}} \alpha$.

Lemma 18 (*Basic Maps*)**.** *Given soft length spaces A, B and a natural number $n \geq 1$, there are morphisms:*

$$mplex_n : !A \to \overbrace{A \otimes \cdots \otimes A}^{n \text{ times}}$$
$$distr : !A \otimes !B \to !(A \otimes B)$$

where $mplex_n(a) = (\overbrace{a, \ldots, a}^{n \text{ times}})$ and $distr(a, b) = (a, b)$.

Proof. We define realizers e^n_{mplex} for every $n \geq 1$ by induction on n:

$$e^1_{mplex} \equiv e_{id};$$
$$e^{n+1}_{mplex} \equiv e_{comp}\langle e_{tens} e^n_{mplex}, e_{contr}\rangle;$$

where $e_{id} \equiv \lambda x.x$ and $e_{contr} \equiv \lambda x.\lambda y.yxx$. Clearly, e^n_{mplex} is a realizer for $mplex_n$. Moreover, $Time(\{e^n_{mplex}\}(x)) \leq n|x| + q_n$, where q_n does not depend on x. Now, let α_n, β_n be such that $\mathcal{F}_{\mathcal{S}}(\alpha_n) \geq cp \cdot n$, $\mathcal{F}_{\mathcal{S}}(\beta_n) \geq q_n$ and α^n_{mplex} be $(n, x \mapsto 0) + \alpha_n + \beta_n$ for every $n \geq 1$. Now, let $(m, g), j \Vdash_{!A} a$. This implies $(m, g) \geq_{\mathcal{S}} (l, x \mapsto (x+1)h(x))$, where $(l, h), j \Vdash_A a$. Notice that

$$n.(l,h) + \alpha_n, \overbrace{\langle j, \ldots, j\rangle}^{n \text{ times}} \Vdash_{\underbrace{A \otimes \ldots \otimes A}_{n \text{ times}}} \overbrace{(a, \ldots, a)}^{n \text{ times}}.$$

We finally get

$$\begin{array}{rcl}
n.(l,h) + \alpha_n & = & (l, x \mapsto n \cdot h(x)) + \alpha_n \\
& \leq_{\mathcal{S}} & (l|2n, x \mapsto (x+1)h(x)) + \alpha_n \\
& = & (2n, x \mapsto 0) + (l, x \mapsto (x+1)h(x)) + \alpha_n \\
& \leq_{\mathcal{S}} & (2n, x \mapsto 0) + (m, g) + \alpha_n \\
& \leq_{\mathcal{S}} & (m, g) + \alpha^n_{mplex}
\end{array}$$

$$\begin{array}{rcl}
Time(\{e^n_{mplex}\}(j)) & \leq & n|j| + q_n \\
& \leq & n\mathcal{F}_{\mathcal{S}}(l,h) + q_n \\
& = & n \cdot h(l) + q_n \\
& \leq & ((l|2n) + 1 - n) \cdot h(l) + q_n \\
& = & (((l|2n) + 1)h(l|2n) - nh(l|2n)) + q_n \\
& \leq & \mathcal{D}_{\mathcal{S}}(n.(l,h), (l|2n, x \mapsto (x+1) \cdot h(x))) + \mathcal{F}_{\mathcal{S}}(\beta_n) \\
& \leq & \mathcal{D}_{\mathcal{S}}(n.(l,h) + \alpha_n, (l|2n, x \mapsto (x+1) \cdot h(x)) + \alpha_n + \beta_n) \\
& \leq & \mathcal{D}_{\mathcal{S}}(n.(l,h) + \alpha_n, (m,g) + \alpha^n_{mplex}).
\end{array}$$

This proves each e^n_{mplex} to be a morphism.

Let $e_{distr} = e_{id}$. We know that $\{e_{id}\}(d)$ takes constant time, say p. Then, let $\alpha, \beta \in |\mathcal{S}|$ be such that $\mathcal{F}_{\mathcal{S}}(\alpha) \geq p + |e_{distr}|$, $\beta = (0, x \mapsto cp(x+1))$. α_{distr} is then defined as $\alpha + \beta$. Now, suppose that $(n, f), \langle d, c\rangle \Vdash_{!A \otimes !B} (a, b)$. From the definition of $!A \otimes !B$ in terms of the length spaces $!A$ and $!B$, there exist $(m, g), (l, h) \in |\mathcal{S}|$ such that $(n, f) \geq_{\mathcal{S}} (m, x \mapsto (x+1)g(x)) + (l, x \mapsto (x+1)h(x))$, $(m, g), d \Vdash_A a$ and $(l, h), c \Vdash_B b$. This in turn implies

$$(m, g) + (l, h) + (0, x \mapsto cp), \langle d, c\rangle \Vdash_{A \otimes B} (a, b),$$

that is to say

$$(m|l, g + h + (x \mapsto cp)), \langle d, c\rangle \Vdash_{A \otimes B} (a, b),$$

and $(m|l, x \mapsto (x+1)(g(x) + h(x) + cp), \langle d, c\rangle \Vdash_{!(A \otimes B)} (a, b)$. Moreover:

$$\begin{array}{rcl}
(m|l, x \mapsto (x+1)(g(x) + h(x) + cp)) & = & (m, x \mapsto (x+1)g(x)) + (l, x \mapsto (x+1)h(x)) + (0, x \mapsto cp(x+1)) \\
& \leq_{\mathcal{S}} & (n, f) + \beta \leq_{\mathcal{S}} (n, f) + \alpha_{distr}.
\end{array}$$

Finally:

$$\begin{array}{rcl}
Time(\{e_{distr}\}(\langle d, c\rangle)) & \leq & p \leq \mathcal{F}_{\mathcal{S}}(\alpha) \\
& \leq & \mathcal{D}_{\mathcal{S}}((m|l, x \mapsto (x+1)(g(x) + h(x) + cp)), (n, f) + \beta) + \mathcal{F}_{\mathcal{S}}(\alpha) \\
& \leq & \mathcal{D}_{\mathcal{S}}((m|l, x \mapsto (x+1)(g(x) + h(x) + cp)), (n, f) + \beta + \alpha) \\
& = & \mathcal{D}_{\mathcal{S}}((m|l, x \mapsto (x+1)(g(x) + h(x) + cp)), (n, f) + \alpha_{distr}).
\end{array}$$

This proves *distr* to be a morphism. □

Lemma 19 (*Functoriality*). *If* $f : A \xrightarrow{e,\alpha} B$, *then* $f : !A \xrightarrow{e,!\alpha} !B$.

Proof. Let α be (i, p) and suppose $(n, f), d \Vdash_{!A} a$. Then there is $(m, g) \in |\mathcal{S}|$ such that $(n, f) \geq_{\mathcal{S}} (m, x \mapsto (x+1)g(x))$, and $(m, g), d \Vdash_A a$. Observe that there must be $(l, h), c$ such that $(l, h), c \Vdash_B f(a)$, $(l, h) \leq_{\mathcal{S}} (m, g) + (i, p)$ and $Time(\{e\}(d)) \leq \mathcal{D}_{\mathcal{S}}((l, h), (m|i, g + p))$. But then $(l, x \mapsto (x+1)h(x)), c \Vdash_{!B} f(a)$ and, moreover:

$$(l, x \mapsto (x+1)h(x)) \leq_{\mathcal{S}} (m|i, x \mapsto (x+1)(g(x) + p(x)))).$$

Finally:

$$\begin{array}{rcl}
Time(\{e\}(d)) & \leq & \mathcal{D}_{\mathcal{S}}((l,h), (m|i, g+p)) = (g+p)(m|i) - h(m|i) \\
& \leq & (m|i+1)((g+p)(m|i) - h(m|i)) \\
& = & (m|i+1)(g+p)(m|i) - (m|i+1)h(m|i) \\
& = & \mathcal{D}_{\mathcal{S}}((l, x \mapsto (x+1)h(x)), (m|i, x \mapsto (x+1)(g(x) + p(x)))).
\end{array}$$

This implies $f : !A \xrightarrow{e,(i,x \mapsto (x+1)p(x))} !B$. □

Soft length spaces form a model of SAL:

Theorem 20. *For every natural number $m \in \mathbb{N}$ there is a polynomial $p_m : \mathbb{N} \to \mathbb{N}$ such that, for every* SAL *proof $\pi : A_1, \ldots, A_n \vdash B$ and for every realizability environment η assigning soft length spaces to all atoms appearing free in the sequent, we have that*

$$[\![\pi]\!]^{\mathscr{S}}_{|\eta|} : [\![A_1 \otimes \cdots \otimes A_n]\!]^{\mathscr{R}}_{\eta} \xrightarrow{\varphi,e} [\![B]\!]^{\mathscr{R}}_{\eta}$$

where $\mathcal{F}_{\mathscr{S}}(\varphi) \leq p_{\partial(\pi)}(|\pi|)$.

Proof. The proof goes by induction on the structure of π. Rules coming from MAL can be trivially justified, because $\mathscr{S}$ is anyway a resource monoid. The two rules P and M can be justified themselves, due to Lemmas 18 and 19. Again, observe that in Lemma 19 the (second component of the) underlying realizer increases in complexity, and that is the only place where this phenomenon happens. □

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

$$\mathit{List}_{\mathsf{SAL}} \equiv \forall\alpha.!(\alpha \multimap \alpha) \multimap !(\alpha \multimap \alpha) \multimap (\alpha \multimap \alpha).$$

Moreover, functions $\mathit{BtoSAL} : \mathscr{B} \to [\![\mathit{List}_{\mathsf{SAL}}]\!]^{\mathscr{S}}$ and $\mathit{SALtoB} : [\![\mathit{List}_{\mathsf{SAL}}]\!]^{\mathscr{S}} \to \mathscr{B}$ can be defined exactly as in EAL. And *EALtoB* is in fact a morphism from $[\![\mathit{List}_{\mathsf{SAL}}]\!]^{\mathscr{R}}$ to $!\mathscr{B}_{\mathscr{S}}$. So, in particular, for every SAL proof π there is a function $f_\pi : \mathscr{B} \to \mathscr{B}$. By the same argument we used for EAL, we get:

Corollary 21 (*Soundness*)**.** *Let π be an* SAL *proof with conclusion $\vdash !^j \mathit{List}_{\mathsf{SAL}} \multimap !^k \mathit{List}_{\mathsf{SAL}}$. Then f_π is computable in polynomial time.*

6. Interpreting LFPL

In [16] the second author introduced a functional language, LFPL, with the property that all functions on natural numbers definable in LFPL are polynomial time computable. The key difference between LFPL and other systems like LAL is that programs defined by iteration or recursion are not marked as such using modalities or similar (e.g., tiering) and can therefore be used as step functions of subsequent recursive definitions.

6.1. Overview of LFPL

LFPL is a linear functional programming language. Its types include inductive datatypes for natural numbers, lists and trees, the resource type $\Diamond$ and binary connectives $\otimes$, $\multimap$, $\oplus$, &.

Each inductive datatype constructor takes an extra argument of type $\Diamond$, intended as a token for using the constructor, and preventing its repeated invocation. 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 $T(A)$ of A-labelled binary trees has constructors $\mathit{leaf} : \Diamond \multimap T(A)$ and $\mathit{node} : \Diamond \multimap A \multimap T(A) \multimap T(A) \multimap T(A)$. The iteration principle allows one to define a term of type $T(A) \multimap B$ from closed terms of type $\Diamond \multimap B$ and $\Diamond \multimap A \multimap B \multimap B \multimap B$. This definition of LFPL slightly differs from the one presented in [16]: a $\Diamond$-resource is needed in *every* constructor (even in those with nullary arity) but is symmetrically available in *every* step function of an iteration. We believe this to be a more uniform and elegant presentation of LFPL.

In this section, we show that length spaces on a resource monoid $\mathcal{M}$ (to be defined shortly) can justify a !-modality and some constructions on it. Moreover, a form of dependent quantification can be justified. Finally, we show that LFPL can be embedded into (second order) MAL endowed with the two constructions above. This, in particular, yields a proof of soundness for LFPL. This is essentially the same as the original proof [16], but more structured and, hopefully, easier to understand. The new approach also yields some new results, namely the justification of a construction which was not available in LFPL as presented in [16], namely a new type of binary trees based on a linear pairing function which allows alternative but not simultaneous access to subtrees, in the spirit of linear logic's additive connectives.

This is a revised and extended version of Section 5 of [9].

6.2. A resource monoid for LFPL

The resource monoid $\mathcal{M} = (|\mathcal{M}|, +, \leq_{\mathcal{M}}, \mathcal{D}_{\mathcal{M}})$ is defined as follows:

- Elements of $|\mathcal{M}| \subseteq \mathbb{N} \times \mathbb{N}^{\mathbb{N}}$ are pairs (n, f) such that $f : \mathbb{N} \to \mathbb{N}$ is a monotonically increasing function bounded by a polynomial.
- For every $(n, f), (m, g) \in |\mathcal{M}|$, $(n, f) + (m, g) = (n + m, f + g)$.
- For every $(n, f), (m, g) \in |\mathcal{M}|$, $(n, f) \leq_{\mathcal{M}} (m, g)$ iff $n \leq m, f(x) \leq g(x)$ for every $x \geq m$ and $(g - f)(x) \leq (g - f)(y)$ whenever $y \geq x \geq m$.
- For every $(n, f), (m, g) \in |\mathcal{M}|$ such that $(n, f) \leq_{\mathcal{M}} (m, g)$,

$$\mathcal{D}_{\mathcal{M}}((n, f), (m, g)) = g(m) - f(m).$$

Observe that n does not appear in the expression defining $\mathcal{D}_{\mathcal{M}}((n, f), (m, g))$.

The pair $(0,0) \in |\mathcal{M}|$, denoted $0_{\mathcal{M}}$, is an identity for $+$. We have a submonoid $\mathcal{M}_0$ of $\mathcal{M}$ whose carrier is $\{(l,p) \in |\mathcal{M}| \mid l = 0\}$. We can pad elements of $\mathcal{M}$ by adding a constant to the second component. The following is now obvious.

Lemma 22. *Both $\mathcal{M}$ and $\mathcal{M}_0$ are resource monoids.*

Proof. Both $(|\mathcal{M}|, +)$ and $(|\mathcal{M}_0|, +)$ are certainly monoids, with $0_{\mathcal{M}} = 0_{\mathcal{M}_0} = (0,0)$. Observe that $n + l \leq m + l$ whenever $n \leq m$ and that $(g + h) - (f + h) = g - f$; compatibility of $\leq_{\mathcal{M}}$ and $\leq_{\mathcal{M}_0}$ easily follows. The two required properties on $\mathcal{D}_{\mathcal{M}}$ and $\mathcal{D}_{\mathcal{M}_0}$ follow directly from their definition. □

A simple inspection of the proofs in Section 3.3 shows that the realizers 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 majorizers may nevertheless be drawn from all of $\mathcal{M}$. We are thus led to the following definition: an LFPL *length space* is a length space on the resource monoid $\mathcal{M}$. A *non-size-increasing* morphism from LFPL length space A to B is a morphism between A and B which admits a majorizer from $\mathcal{M}_0$.

Proposition 23. LFPL *length spaces with non-size-increasing morphisms form a symmetric monoidal closed category.*

As a consequence, LFPL length spaces and non-size-increasing morphisms can justify all MAL rules. In the next two sections we will show that additional constructs can be justified.

6.3. A modality

Before describing the !-construction on LFPL length spaces, we need to introduce indexed LFPL length spaces and families of LFPL length spaces. Let A be an LFPL length space and n be a positive integer. The LFPL length space A^n is defined by $|A^n| = |A|$ and $\alpha, e \Vdash_{A^n} a$ iff $\alpha \geq (nm, 3np)$ for some (m,p) such that $(m,p), e \Vdash_A a$.

So, A^n corresponds to the subset of $\overbrace{A \otimes \cdots \otimes A}^{n \text{ times}}$ consisting of those n-uples with identical components. The factor $3n$ instead of just n is needed in order to justify the linear time needed to compute the copying involved in the obvious morphism from A^{m+n+1} to $A^m \otimes A^n \otimes A$.

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

Proposition 24. *For each A, the function $mplex_3 : |A| \to |A| \times |A| \times |A|$ (seen as a trivial family of maps over $\mathbb{N}^+ \times \mathbb{N}^+$) is a uniform (in m, n) map from A^{m+n+1} to $A^m \otimes A^n \otimes A$. Moreover, the identity is a morphism from A^1 to A.*

Consistently with what we have done so far, we will use the same symbols for formulas (or types) and length spaces interpreting them. In particular, we will now define a length space $\Diamond$ interpreting the diamond type. Formally, the LFPL length space $\Diamond$ is defined by $|\Diamond| = \{0\}$ and $\alpha, \lambda x.x \Vdash_{\Diamond} 0$ if both $\alpha \geq (1,0)$ and $\mathcal{F}_{\mathcal{M}}(\alpha) \geq |\lambda x.x|$. Please notice the similarities between the length space $\Diamond$ and the length space $I_{\mathcal{M}}$: the only difference is the additional constraint $\alpha \geq (1,0)$.

For each LFPL length space A we define LFPL length space $!A$ by $|!A| = |A|$ and $\alpha, e \Vdash_{!A} a$ if there exists $\beta = (0,p) \in \mathcal{M}_0$ with $\beta, e \Vdash_A a$ and $\alpha \geq (0, (3x+1)p)$. Observe that $(3 \cdot 0 + 1)p(0) = p(0) \geq |e|$; as a consequence, this is a well-defined length space.

Proposition 25. *For every* LFPL *length spaces A and B:*

(i) *If $f : A \to B$ then $f : !A \to !B$.*
(ii) *$!(A \otimes B) \simeq !A \otimes !B$*
(iii) *The identity on $|A| \times \{0\}$ (seen as a trivial family of functions over $\mathbb{N}^+$) is a uniform map $!A \otimes \Diamond^n \to A^n \otimes \Diamond^n$.*

Proof. About (i): assume that $\varphi, e \Vdash f$ where $\varphi = (0,q) \in \mathcal{M}_0$. We claim that $f : !A \xrightarrow{e,(0,(3x+1)q)} !B$. Suppose that $\alpha, t \Vdash_{!A} a$ where $\alpha \geq (0, (3x+1)p)$ and $(0,p), t \Vdash_A a$. Since f is a morphism, we obtain v, β such that $\beta, v \Vdash_B f(a)$ and $\beta \leq \varphi + (0,p)$. 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 $(0, (3x+1)r), v \Vdash_{!B} f(a)$. On the other hand $(3x+1)r \leq (3x+1)(p+q)$. The resource bounds are obvious. (ii) can be trivially proved. Finally, consider (iii): the required morphism $!A \otimes \Diamond^n \to A^n \otimes \Diamond^n$. Clearly, it may be realized by the identity; we claim that 0 can serve as a majorizer. Indeed, a majorizer of (a,d) in $|A| \times \{0\}$ is of the form $(n, (3x+1)p)$ where $(0,p)$ majorizes a in A. Now, $(n, 3np)$ is a majorizer of (a,d) in $A^n \otimes \Diamond^n$. But $(3x+1)p - 3np$ is non-decreasing and non-negative above n. □

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.

Remark 26. Please notice the strong similarity between the resource monoids $\mathcal{M}$ and $\mathcal{S}$. Indeed, the only difference lies in the way addition acts on the first component of pairs: in one case, we sum the two natural numbers, while in the other case, we take the maximum. This points to a close relationship between LFPL and SAL and also shows a certain trade-off 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 internalized 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.

6.4. Dependent typing

Let $\{T_i\}_{i\in I}$ be a family of LFPL length spaces such that $|T_i| = |T_j|$ for every $i, j \in I$. The LFPL space $\exists i.T_i$ is defined by $|\exists i.T_i| = |T_j|$ (where j is any element of I) and $\alpha, e \Vdash_{\exists i.T_i} t$ if $\alpha, e \Vdash_{T_j} t$ for some $j \in I$.

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 formalized would amount to an extension of LFPL with indexed type dependency in the style of Dependent ML [24].

6.5. Some constructions on LFPL length spaces

In this section, we show how to justify the constructs of LFPL as originally presented [16] in the category of LFPL length spaces.

First, we recall (see Section 3.3) that additive conjunction can be defined as

$$A\&B = \exists\alpha.(\alpha \multimap A) \otimes (\alpha \multimap B) \otimes \alpha.$$

The first projection map $A\&B \to A$ 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 $\lambda c^C.(f, g, c)$.

In order to interpret *unary natural numbers*, we define Nat_{LFPL} as $\exists n.N_n$ where $\{N_n\}_{n>0}$ is a family of LFPL length spaces and, for every positive n:

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

Natural numbers smaller of equal to n can be encoded as functions in $|N_{n+1}|$ using the usual impredicative, Church-style scheme. We can internally define a successor map $\Diamond \otimes N_n \to N_{n+1}$ as follows: starting from $d : \Diamond, e : \Diamond^n$ and $f : \forall\alpha.((\alpha \multimap \alpha)\&\alpha)^n \multimap \alpha$ we obtain a member of $\Diamond^{n+1}$ (from d and e) and we define $g : \forall\alpha.((\alpha \multimap \alpha)\&\alpha)^{n+1} \multimap \alpha$ as $\lambda(u^{(\alpha\multimap\alpha)\&\alpha}, w^{((\alpha\multimap\alpha)\&\alpha)^n}).(\pi_1 u)(f\ w)$, where π_1 is the first projection. From this, we obtain a map $\Diamond \otimes Nat_{\mathsf{LFPL}} \to Nat_{\mathsf{LFPL}}$ by existential introduction and elimination. Of course, we also have a constant zero $\Diamond \to N_1$ yielding a map $\Diamond \to Nat_{\mathsf{LFPL}}$ by existential introduction. Finally, we can define a (uniform) iteration map

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

as follows: given $t : !(\Diamond \multimap ((A \multimap A)\&A))$ and $(e, f) \in N_n$ we unpack t using Proposition 25 to yield $u \in (\Diamond \multimap ((A \multimap A)\&A))^n$ as well as $e \in \Diamond^n$. Feeding these "diamonds" one by one to the components of u we obtain $w \in ((A \multimap A)\&A)^n$. But then $f\ w$ yields the required element of A. Existential elimination now yields a single map

$$iternat_{\mathsf{LFPL}} : !(\Diamond \multimap ((A \multimap A)\&A)) \multimap Nat_{\mathsf{LFPL}} \multimap A.$$

Analogous arguments allow us to interpret *binary lists* as an LFPL length space

$$List_{\mathsf{LFPL}} = \exists n.\Diamond^n \otimes \forall\alpha.((\alpha \multimap \alpha)\&(\alpha \multimap \alpha)\&\alpha)^n \multimap \alpha.$$

Similarly, we can interpret *binary A-labelled trees* using a type family $\{T_n(A)\}_{n>0}$ where

$$T_n(A) = \Diamond^n \otimes \forall\alpha.((A \multimap \alpha \multimap \alpha \multimap \alpha)\&\alpha)^n \multimap \alpha$$

and defining proper trees as $Tree_{\mathsf{LFPL}}(A) = \exists n.T_n(A)$. We get maps $leaf : \Diamond \to Tree_{\mathsf{LFPL}}(A)$ and $node : \Diamond \otimes A \otimes Tree_{\mathsf{LFPL}}(A) \otimes Tree_{\mathsf{LFPL}}(A) \to Tree_{\mathsf{LFPL}}(A)$ and an analogous iteration construct $itertree_{\mathsf{LFPL}}(A)$.

Finally, and this goes beyond what was already known, we can define *lazy trees* using additive conjunction. Following the pattern of the binary trees above, we define another family $\{T^\times_n(A)\}_{n>0}$ where

$$T^\times_n(A) = \Diamond^n \otimes \forall\alpha.((A \multimap (\alpha\&\alpha) \multimap \alpha)\&\alpha)^n \multimap \alpha$$

and $Tree^\times_{\mathsf{LFPL}}(A) = \exists n.T^\times_n(A)$. We get maps $leaf^\times : \Diamond \to Tree^\times_{\mathsf{LFPL}}(A)$ and $node^\times : \Diamond \otimes A \otimes (Tree^\times_{\mathsf{LFPL}}(A)\&Tree^\times_{\mathsf{LFPL}}(A)) \to Tree^\times_{\mathsf{LFPL}}(A)$ as well as an analogous iteration construct

$$itertree^\times_{\mathsf{LFPL}}(A) : !(\Diamond \multimap ((A \multimap (B\&B) \multimap B)\&B)) \multimap Tree^\times_{\mathsf{LFPL}}(A) \multimap B.$$

We describe in detail the construction of the map $node^\times$ above, which is not entirely straightforward. First, we note that for any LFPL length spaces A, B and for every $m, n > 0$ the obvious map $(\Diamond^m \otimes A)\&(\Diamond^n \otimes B) \to \Diamond^{\max(m,n)} \otimes (A\&B)$ is a morphism. This is because a majorizer of an element of $(\Diamond^m \otimes A)\&(\Diamond^n \otimes B)$ 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) $d : \Diamond, x : A, s : T^\times_{d_1}(A)\&T^\times_{d_2}(A)$. Using the just described morphism we decompose s into $e : \Diamond^{\max(d_1,d_2)}$ and $r : W_{d_1}(A)\&W_{d_2}(A)$ where $W_n(A) = ((A \multimap (\alpha\&\alpha) \multimap \alpha)\&\alpha)^n \multimap \alpha$. We have stripped off the universal quantifier. Now d and e together yield an element of $\Diamond^{1+\max(d_1,d_2)}$. It remains to construct a member of $W_{1+\max(d_1,d_2)}(A)$. To this end, we assume $u : (A \multimap (\alpha\&\alpha) \multimap \alpha)\&\alpha$ and $f : ((A \multimap (\alpha\&\alpha) \multimap \alpha)\&\alpha)^{\max(d_1,d_2)}$ and define the required element of α as $(\pi_1(u))\ x\ \langle \pi_1(r)f, \pi_2(r)f\rangle$. Here π_1 and

π_2 denote the projections from the cartesian product. The sharing of the variables f and r is legal in the two components of a cartesian (i.e., additive) pairing, but would of course not be acceptable in a multiplicative pairing. We have elided the obvious coercions from $(\cdot)^{\max(d_1,d_2)}$ to $(\cdot)^{d_i}$.

We remark that these cartesian trees are governed by their height rather than their size. In other words, any binary tree with height n corresponds to an element of $|T_n^{\times}(A)|$, even if its size is exponential in n (e.g., if the tree is a complete binary tree of height n). Note that if $A = I$ we can form the function $\lambda d^{\Diamond}.\lambda t^{Tree^{\times}_{\mathsf{LFPL}}(A)}.node^{\times}\, d\, 0\, \langle t, t\rangle : \Diamond \multimap Tree^{\times}_{\mathsf{LFPL}}(A) \multimap Tree^{\times}_{\mathsf{LFPL}}(A)$. Iterating this map yields a function $Nat_{\mathsf{LFPL}} \to Tree^{\times}_{\mathsf{LFPL}}(I)$ computing complete binary trees of a given height. Of course, at the level of the realizers, 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.

6.6. *Interpreting* LFPL

LFPL as described in [16] is a formal system for constructing derivations for judgements in the form $A_1, \ldots, A_n \vdash \mathsf{M} : B$. For our purposes, we can consider derivations of such judgements as *proofs*, forgetting about the underlying term M. LFPL types include base types for natural numbers, lists and binary trees. Following the semantic constructions sketched in the previous subsection, it is easy to define both the set-theoretic semantics $[\![\pi]\!]^{\mathscr{S}}_{\eta}$ and the realizability semantics $[\![\pi]\!]^{\mathscr{R}}_{\eta}$ of any LFPL derivation π. In this sense, LFPL length spaces form a model of LFPL:

Theorem 27. *For every* LFPL *derivation* $\pi : A_1, \ldots, A_n \vdash B$ *and for every realizability environment* η *assigning* LFPL *length spaces to all atoms appearing free in* $A_1, \ldots, A_n, B$, *we have that*

$$[\![\pi]\!]^{\mathscr{S}}_{|\eta|} : [\![A_1 \otimes \cdots \otimes A_n]\!]^{\mathscr{R}}_{\eta} \xrightarrow{\varphi,e} [\![B]\!]^{\mathscr{R}}_{\eta}.$$

Proof. As usual, the proof proceeds by induction on π. In particular, every LFPL construct can be shown to be justifiable by LFPL length spaces following the informal arguments in the last section. □

As for EAL or SAL showing that the obvious function from $\mathscr{B}$ to $List_{\mathsf{LFPL}}$ is a (non-size-increasing) morphism is easy. Moreover, any LFPL proof π of $\vdash List_{\mathsf{LFPL}} \multimap List_{\mathsf{LFPL}}$ corresponds to a function $f_\pi : \mathscr{B} \to \mathscr{B}$. As a consequence:

Corollary 28 (*Soundness*)**.** *Let* π *be an* LFPL *proof with conclusion* $\vdash List_{\mathsf{LFPL}} \multimap List_{\mathsf{LFPL}}$ *and let* $f_\pi : \mathscr{B} \to \mathscr{B}$ *be the function induced by* π. *Then* f_π *is computable in polynomial time.*

7. Conclusions

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 justified explicitly. While resulting in a considerable simplification of previous soundness proofs for EAL, SAL, LFPL and LAL (see [10]), our method has also lead to new results, in particular the justification of polymorphism and a modality for LFPL.

Acknowledgements

The authors would like to thank the anonymous referees for many useful comments.

References

[1] Andrea Asperti, Luca Roversi, Intuitionistic light affine logic, ACM Transactions on Computational Logic 3 (1) (2002) 137–175.
[2] Patrick Baillot, 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, in: LNCS, vol. 2987, Springer, 2004.
[3] Henk Barendregt, Lambda Calculus: Its Syntax and Semantics, Academic Press, 1984.
[4] Stephen Bellantoni, Karl Heinz Niggl, Helmut Schwichtenberg, Higher type recursion, ramification and polynomial time, Annals of Pure and Applied Logic 104 (2000) 17–30.
[5] Stephen Cook, Alasdair Urquhart, Functional interpretations of feasible constructive arithmetic, Annals of Pure and Applied Logic 63 (2) (1993) 103–200.
[6] Paolo Coppola, Simone Martini, Typing lambda terms in elementary logic with linear constraints, in: Proceedings of the 6th International Conference on Typed Lambda-Calculus and Applications, in: LNCS, vol. 3461, Springer, 2001, pp. 76–90.
[7] Thierry Coquand, Christine Paulin, Inductively defined types, in: Conference on Computer Logic, in: LNCS, vol. 417, Springer, 1988, pp. 50–66.
[8] John Crossley, Gerald Mathai, Robert Seely, A logical calculus for polynomial-time realizability, Journal of Methods of Logic in Computer Science 3 (1994) 279–298.
[9] Ugo Dal Lago, Martin Hofmann, Quantitative models and implicit complexity, in: Foundations of Software Technology and Theoretical Computer Science, Proceedings, in: LNCS, vol. 3821, Springer, 2005, pp. 189–200.
[10] Ugo Dal Lago, Martin Hofmann, A semantic proof of polytime soundness of light affine logic, Theory of Computing Systems 46 (4) (2010) 673–689.
[11] Ugo Dal Lago, Simone Martini, The weak lambda calculus as a reasonable machine, Theoretical Computer Science 398 (1–3) (2008) 32–50.
[12] Jean-Yves Girard, Light linear logic, Information and Computation 143 (2) (1998) 175–204.
[13] Jean-Yves Girard, Yves Lafont, Paul Taylor, Proof and Types, Cambridge University Press, 1987.
[14] Martin Hofmann, Type systems for polynomial-time computation, in: Habilitationsschrift, Darmstadt University of Technology, 1999.
[15] Martin Hofmann, Safe recursion with higher types and BCK-algebra, Annals of Pure and Applied Logic 104 (2000) 113–166.

[16] Martin Hofmann, Linear types and non-size-increasing polynomial time computation, Information and Computation 183 (1) (2003) 57–85.
[17] Martin Hofmann, Philip Scott, Realizability models for BLL-like languages, Theoretical Computer Science 318 (1–2) (2004) 121–137.
[18] Martin Hyland, A small complete category, Annals of Pure and Applied Logic 40 (1988) 135–165.
[19] Georg Kreisel, Interpretation of analysis by means of constructive functions of finite types, in: Arend Heyting (Ed.), Constructivity in Mathematics, North-Holland, 1959, pp. 101–128.
[20] Yves Lafont, Soft linear logic and polynomial time, Theoretical Computer Science 318 (2004) 163–180.
[21] Andrew M. Pitts, Polymorphism is set theoretic, constructively, in: Category Theory and Computer Science, in: LNCS, vol. 283, Springer, 1987, pp. 12–39.
[22] Peter van Emde Boas, Machine models and simulation, in: Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity, MIT Press, 1990, pp. 1–66.
[23] Christopher Wadsworth, Some unusual λ-calculus numeral systems, in: J.P. Seldin, J.R. Hindley (Eds.), To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism, Academic Press, 1980.
[24] Hongwei Xi, Frank Pfenning, Dependent types in practical programming, in: Proceedings of the 26th ACM SIGPLAN Symposium on Principles of Programming Languages, 1999, pp. 214–227.