Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 318 (2004) 121–137
Theoretical Computer Science
www.elsevier.com/locate/tcs

Realizability models for BLL-like languages

M. Hofmann[a,*,1], P.J. Scott[b,2]

[a] Institut für Informatik, Ludwig-Maximilians-Universität München, Oettingenstrasse 67, D-80538, München, Germany
[b] Department of Mathematics & Statistics, University of Ottawa, 585 King Edward, Ottawa Ont., Canada K1N6N5

Abstract

We give a realizability model of Girard–Scedrov–Scott's *Bounded Linear Logic* (BLL). This gives a new proof that all numerical functions representable in that system are polytime. Our analysis naturally justifies the design of the BLL syntax and suggests further extensions.

Keywords: Linear logic; Complexity lambda calculus; Finite model theory

1. Introduction

Bounded Linear Logic (BLL) [5] was an early attempt to provide an *intrinsic* notion of polynomial time computation within a logical system. That is, the aim was not merely to express polynomial time *computability* in terms of *provability* of certain restricted formulas, but rather to provide a typed logical system in which computation via cut-elimination or proof normalization is inherently polytime. Since the appearance of this paper, several different typed functional systems for analyzing ptime computability have appeared in the literature [4–8,12,13]. For deeper foundational purposes, we should mention Girard's light linear logic (LLL) [4] as a major improvement of the syntax of BLL, in that it eliminates the explicit polynomial I/O size-bounds, but at

* Corresponding author.
E-mail addresses: mhofmann@informatik.uni-muenchen.de (M. Hofmann), phil@mathstat.uottawa.ca (P.J. Scott).
[1] Research partially supported by EPSRC Grant No. GR/N28436.
[2] Research supported by an operating grant from the Natural Sciences and Engineering Research Council of Canada.

0304-3975/$ - see front matter
doi:10.1016/j.tcs.2003.10.019

the expense of introducing more subtle typing distinctions. Moreover, while capturing the same extensional class of polytime functions, it appears to be less flexible than BLL in terms of expressing concrete algorithms. Furthermore, BLL has its own merits: from the viewpoint of computer science, BLL is a natural polymorphically typed functional language in which bounded storage can represent bounded calls to memory.

The main theorem in [5] is that the number-theoretic functions representable in BLL are polytime. The proof of this result used sophisticated techniques from the proof theory of linear logic, notably a very detailed analysis of normalization of proof nets with boxes. The normalization strategy itself was of a special kind, inspired from Girard's Geometry of Interaction program. In this paper, we give a direct, semantic proof of this main result which does not involve any notion of reduction, term rewriting, or cut-elimination. Rather, we assign polytime algorithms to proofs in a compositional, syntax-directed manner. We use realizability to relate these algorithms to the intended set-theoretic meaning of the proofs themselves. All this is presented in the form of a concrete categorical model of BLL which interprets BLL-formulas as sets with some additional structure and proofs as functions witnessed by polytime algorithms operating on this additional structure. At the same time, our analysis gives a natural interpretation of the BLL syntax which justifies the fine points of its design and might suggest further extensions. For example, our analysis encompasses an affine variant of BLL.

Our proof is constructive, in the sense that it can be formalized in an extensional version of the calculus of inductive constructions [3]. This provides a new compilation method for turning BLL proofs into equivalent polytime algorithms. Of course, in practice, one would not use such a formalization, but rather derive the compilation algorithm by hand directly from our proof.

Our interpretation bears an intriguing relationship to approaches based on finite model theory, such as [6]. Namely, the polytime functions we obtain are of the form of Goerdt–Gurevich's *total global functions*: they take one extra argument which is interpreted as a bound on the size of all the actual inputs. However, unlike Goerdt's system which is a finite model interpretation of Gödel's system $\mathscr{T}$–hence the successor function is not injective—BLL supports the usual semantics, including Peano's axioms, thus can be seen as a meaning-preserving annotation to standard functional programming. We hope that this relationship can be used to transfer Goerdt's characterizations of LOGSPACE and PSPACE to a similar setting.

It would be interesting to see whether our techniques can be adapted to Girard's aformentioned system LLL in order to get a new proof of the fact that all of its representable functions are polytime computable. Our direct attempts at doing so have so far not succeeded which is due to the considerable differences in syntax, for example the self-dual modality section which has no obvious counterpart in BLL. An interesting possibility might be to modify Baillot's [2] recent denotational model for LLL according to the methods in this paper; that is putting polytime constraints into the morphism while maintaining closure of the model under the required rules. We remark in this context that the original model of Baillot as well as the phase semantics of LLL [11] do not guarantee the polytime computability of representable functions.

2. Bounded linear logic

We introduce the theory BLL of bounded linear logic, first proposed in [5].

2.1. Resource polynomials

Resource polynomials [5] are finite sums of products of binomial coefficients, i.e. $\sum_{j\leqslant q}\prod_{i\leqslant p}\binom{x_{ij}}{n_{ij}}$ where, for any fixed j, the variables x_{ij} are distinct and n_{ij} are non-negative integer constants.

Resource polynomials are closed under sum, product, and composition. Given such polynomials p,q write $p\leqslant q$ to denote that $q-p$ is a resource polynomial. If $p\leqslant p'$ and $q\leqslant q'$, then their composites satisfy $q\circ p\leqslant q'\circ p'$.

2.2. Syntax of BLL

Formulae are given by the following general syntax:

$$A,B ::= \alpha(\vec{p}) \mid A\otimes B \mid A\multimap B \mid \forall\alpha.A \mid !_{x<p}A.$$

Here *atomic formulae* have the form $\alpha(p_0,\dots,p_{n-1})$, where α is a second-order variable of given finite positive arity n and $\vec{p}=(p_0,\dots,p_{n-1})$ denotes a list of resource polynomials of length n. We assume that there are infinitely many second-order variables of each finite arity.

The formula $\forall\alpha.A$ denotes second-order universal quantification, while $!_{x<p}A$ is *bounded storage*, where p is a resource polynomial not containing x and x is bound in $!_{x<p}A$.

Positive and negative occurrences of resource terms in formulae are defined by induction as usual: in $!_{x<p}A$, p occurs negatively. The p_i and their subterms occur positively in an atomic formula $\alpha(\vec{p})$. The connectives $\otimes$ and $\forall\alpha$ are monotone; the connective $\multimap$ is antitone in its first and monotone in its second argument so that for example, p occurs positively in $\forall\alpha(!_{y<p}(\alpha(y)\multimap\alpha(y+1))\otimes\alpha(0))\multimap\alpha(p)$.

Let the free resource variables $x_0,\dots,x_{n-1}$ occur only positively in B. Then $\lambda x_0,\dots,x_{n-1}.B$ is a (second-order) abstraction term, say T. $A[\alpha := T]$ denotes the result of substituting T for α in A, i.e. of replacing the atoms $\alpha(p_0,\dots,p_{n-1})$ in A by $B[x_0{:=}p_0]\dots[x_{n-1}{:=}p_{n-1}]$.

Given types A and A', write $A\leqslant A'$ if A and A' only differ in their choice of resource polynomials, and

(i) for any positive occurrence of resource polynomial p in A, the homologous p' in A' is such that $p\leqslant p'$,
(ii) for any negative occurrence of resource polynomial p in A, the homologous p' in A' is such that $p'\leqslant p$.

If Γ and Γ' are finite multisets of formulae, $\Gamma\leqslant\Gamma'$ iff it is true componentwise.

Proofs are given by Gentzen sequents, as follows.

2.3. BLL sequents

Sequents have the form $\Gamma \vdash B$, where Γ is a finite (possibly empty) multiset of formulae. In order to avoid mentioning the permutation rule, the formulae in Γ are considered indexed but not ordered. In what follows, p, q, w (possibly with subscripts) range over resource polynomials.

Axiom (*Waste of Resources*) $\quad A \vdash A'$, where $A \leqslant A'$ (Special case: $A \vdash A$).

Cut $$\frac{\Gamma \vdash A \quad \Delta, A \vdash B}{\Gamma, \Delta \vdash B}$$

$\otimes$L $$\frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C}$$ $\otimes$R $$\frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}$$

$\multimap$L $$\frac{\Gamma \vdash A \quad \Delta, B \vdash C}{\Gamma\ ,\ \Delta, A \multimap B \vdash C}$$ $\multimap$R $$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B}$$

$\forall$L $$\frac{\Gamma, A[\alpha := T] \vdash B}{\Gamma, \forall\alpha.A \vdash B}$$ $\forall$R $$\frac{\Gamma \vdash A}{\Gamma \vdash \forall\alpha.A}$$

(provided α is not free in Γ)

(!W) *Weakening* $$\frac{\Gamma \vdash B}{\Gamma, !_{x<w}A \vdash B}$$

(!D) *Dereliction* $$\frac{\Gamma, A[x := 0] \vdash B}{\Gamma, !_{x<1+w}A \vdash B}$$

(!C) *Contraction* $$\frac{\Gamma, !_{x<p}A, !_{y<q}A[x := p + y] \vdash B}{\Gamma, !_{x<p+q+w}A \vdash B}$$

where $p + y$ is free for x in A.

(S!) *Storage* $$\frac{!_{z<q_1(x)}A_1[y := v_1(x) + z], \ldots, !_{z<q_n(x)}A_n\,[y := v_n(x) + z] \vdash B}{!_{y<v_1(p)+w_1}A_1,\ \ldots, !_{y<v_n(p)+w_n}A_n \vdash !_{x<p}B,}$$

where $v_i(x) + z$ is free for y in A_i, where $v_i(x) = \sum_{z<x} q_i(z)$ and where all formulae to the left of the $\vdash$ have the indicated form.

Remark 1.

- The rules of BLL are written in such a way that given any proof π of a sequent $\Gamma \vdash A$ and given any $\Gamma' \leqslant \Gamma$ and $A \leqslant A'$ then a simple change of resource parameters will yield a proof π' of $\Gamma' \vdash A'$ *without altering the structure of the proof.*

  Note that the "waste" w in each of the rules associated with storage can without loss of generality be assumed 0 as the general case can be recovered by cutting with appropriate axioms. In this paper we are not interested in cut elimination, therefore, we will adopt this simplification.
- We also introduce a unit for $\otimes$, denoted I. The ordinary LL rules for I are as follows:

  $$\frac{\Gamma \vdash B}{\Gamma, I \vdash B} \qquad \vdash I.$$

- The data type of tally natural numbers of size at most x is

  $$\mathbf{N}_x = \forall\alpha\ !_{y<x}(\alpha(y) \multimap \alpha(y+1)) \multimap (\alpha(0) \multimap \alpha(x)).$$

  Moreover, there are proofs $\vdash 0 : \mathbf{N}_0$ and $S : \mathbf{N}_x \vdash \mathbf{N}_{x+1}$ representing "zero" and "successor", resp. (see [5]).
- The data type of dyadic lists of size at most x is

  $$\mathbf{N}^2_x = \forall\alpha\ !_{y<x}(\alpha(y) \multimap \alpha(y+1)) \multimap !_{y<x}(\alpha(y) \multimap \alpha(y+1)) \multimap (\alpha(0) \multimap \alpha(x)).$$

  There are proofs $\varepsilon : \mathbf{N}^2_0$ and $S_0, S_1 : \mathbf{N}^2_x \vdash \mathbf{N}^2_{x+1}$ representing zero and the two successor functions, resp. (see [5]).

The following two rules are not contained in the definition of BLL [5], and as far as we can see are not admissible in BLL.

$$(\mathit{Func}) \quad \frac{A \vdash B}{!_{x<p}A \vdash !_{x<p}B} \qquad (\mathit{Mon}) \quad \frac{}{!_{x<p}A \otimes !_{x<p}B \vdash !_{x<p}(A \otimes B)}$$

These rules express functoriality and monoidalness of $!_{x<p}$. Note that they can be subsumed under the following generalization of (*Func*):

$$(\mathit{Func}\text{-}\otimes) \quad \frac{A_1, \ldots, A_n \vdash B}{!_{x<p}A_1, \ldots, !_{x<p}A_n \vdash !_{x<p}B}$$

Our semantics validates these rules, so as a result their addition does not increase the computational strength of BLL. We note that in their presence the storage rule can be replaced by the following axiom:

$$!_{y<\sum_{x<p} q(x)}A \vdash !_{x<p}!_{z<q(x)}A\left[y := z + \sum_{\xi<x} q(\xi)\right]$$

3. Main result

We shall assume that our ambient set theory is constructive, so that we shall have a set (of sets) $\mathscr{U}$ containing the natural numbers $\mathbb{N}$, closed under product, function space, and $\mathscr{U}$-indexed products [10]. We discuss this point in more detail below in Section 5. This allows us to interpret types as sets in the following way: given a formula A and an environment ρ which assigns sets to type-variables, we obtain a set-theoretic interpretation $[\![A]\!]_\rho$ as follows:

$$[\![\alpha(\vec{p})]\!]_\rho = \rho(\alpha),$$

$$[\![A \otimes B]\!]_\rho = [\![A]\!]_\rho \times [\![B]\!]_\rho,$$

$$[\![A \multimap B]\!]_\rho = [\![A]\!]_\rho \Rightarrow [\![B]\!]_\rho,$$

$$[\![\forall\alpha A]\!]_\rho = \prod_{C \in \mathscr{U}} [\![A]\!]_{\rho[\alpha \mapsto C]},$$

$$[\![!_{x<p} A]\!]_\rho = [\![A]\!]_\rho.$$

Notice that this interpretation of types ignores the resource polynomials.

To every proof π of a sequent $A_1, \ldots, A_n \vdash B$ and environment ρ, we can assign a set-theoretic function

$$[\![\pi]\!]_\rho : [\![A_1 \otimes \cdots \otimes A_n]\!]_\rho \to [\![B]\!]_\rho$$

by induction on derivations, in the obvious way. Observe that

$$[\![\mathbf{N}_p]\!] = \prod_{C \in \mathscr{U}} (C \Rightarrow C) \Rightarrow (C \Rightarrow C)$$

There is a pair of functions $\phi : \mathbb{N} \to [\![\mathbf{N}_p]\!]$ and $\psi : [\![\mathbf{N}_p]\!] \to \mathbb{N}$ satisfying $\psi \circ \phi = id_{\mathbb{N}}$, defined as follows:

$$\phi(n)_C(f, z) = f^n(z),$$
$$\psi(x) = x_{\mathbb{N}}(S)(0),$$

where in the definition of ψ, the symbols S and 0 are the usual successor and zero on $\mathbb{N}$. Note that $\psi[\![0]\!] = 0$ and $\psi \circ [\![S]\!] \circ \phi = S$.

Our main goal is to give a new proof of the following theorem, which is equivalent to Theorem 5.4 in [5].

Theorem 2. *Let π be a proof of $\vdash \mathbf{N}_x \multimap \mathbf{N}_{p(x)}$, where p does not contain any other free resource variables except x. Let $f : \mathbb{N} \to \mathbb{N}$ be the function*

$$f(n) = \psi([\![\pi]\!](\phi(n))).$$

Then $f(n)$ is computable in polynomial time in n. Moreover, $f(n) \leqslant p(n)$ and an algorithm for f can be effectively obtained from the proof π.

An analogous result holds for the type of dyadic lists, as in Theorem 5.3 of [5].

4. A realizability model for BLL

We now introduce a refined model $\mathscr{B}$ for BLL based on realizability and ideas from [7]. This will allow us to obtain the above theorem as a direct corollary of soundness of the interpretation. See the proof after Theorem 23.

4.1. Preliminaries

For $x \in \mathbb{N}$, we write $|x| = \lceil \log_2(x+1) \rceil$ for the binary length of x. We fix a linear time computable pairing function $\langle \cdot, \cdot \rangle : \mathbb{N}^2 \to \mathbb{N}$ satisfying $|\langle x, y \rangle| = |x| + |y| + \mathrm{O}(\log(|x|))$. We should also remark that the inverses of the pairing function are assumed to be linear time computable.

Let X be a finite set of variables. We write $\mathscr{V}(X)$ for $\mathbb{N}^X$—the elements of $\mathscr{V}(X)$ are called *valuations* (over X). If $\eta \in \mathscr{V}(X)$ and $c \in \mathbb{N}$ then $\eta[x \mapsto c]$ denotes the valuation which maps x to c and acts like η otherwise. We assume some reasonable encoding of valuations as integers allowing them to be passed as arguments to algorithms.

We write $\mathscr{P}(X)$ for the set of resource polynomials over X. If $p \in \mathscr{P}(X)$ and $\eta \in \mathscr{V}(X)$ we write $p(\eta)$ for the number obtained by evaluating p with $x \mapsto \eta(x)$ for each $x \in X$.

Let X, Y be finite sets of variables. A *substitution* from X to Y is a function $\sigma : Y \to \mathscr{P}(X)$. We may write a substitution from X to $Y = \{y_0, \ldots, y_{n-1}\}$ in the form $\sigma = [X; y_0 := p_0, y_1 := p_1, \ldots, y_{n-1} := p_{n-1}]$. This is defined if $p_i \in \mathscr{P}(X)$ and in this case we have $\sigma(y_i) = p_i$. If the domain X is clear from the context, we may simply write $\sigma = [y_0 := p_0, y_1 := p_1, \ldots, y_{n-1} := p_{n-1}]$.

A substitution σ from X to Y induces functions $\sigma(-) : \mathscr{V}(X) \to \mathscr{V}(Y)$ and $-[\sigma] : \mathscr{P}(Y) \to \mathscr{P}(X)$ in the obvious way, i.e., $(\sigma(\eta))(y) \stackrel{\text{def}}{=} \sigma(y)(\eta)$ and $p[\sigma] \stackrel{\text{def}}{=} p[y_0 := \sigma(y_0)] \ldots [y_{n-1} := \sigma(y_{n-1})]$.

We assume the known untyped lambda calculus as defined e.g. in [1]. An untyped lambda term is *affine linear* if each variable (free or bound) appears at most once (up to α-congruence). E.g. $\lambda x \lambda y. yx$ and $\lambda x \lambda y. y$ and $\lambda x. xy$ are affine linear; the term $\lambda x. xx$ is not. Notice that such a term t is strongly normalisable in less than $|t|$ steps where $|t|$ is the size of the term. The runtime of the computation leading to the normal form is therefore $\mathrm{O}(|t|^2)$. We will henceforth use the expression *affine lambda term* for an untyped affine linear lambda term which is in normal form. If s, t are affine lambda terms then their application st is defined as the normal form of the lambda term st. Notice that the application st can be computed in time $\mathrm{O}((|s| + |t|)^2)$.

If s, t are affine lambda terms we write $s \otimes t$ for the affine lambda term $\lambda f. fst$. If t is an affine lambda term possibly containing the free variables x, y then we write $\lambda x \otimes y. t$ for $\lambda u. u(\lambda x \lambda y. t)$. Notice that $(\lambda x \otimes y. t)(u \otimes v) = t[x := u][y := v]$.

More generally, if $(t_i)_{i<n}$ is a family of affine lambda terms, we write $\bigotimes_{i<n} t_i$ for $\lambda f. f t_0 t_1 \ldots t_{n-1}$ and $\lambda \bigotimes_{i<n} x_i. t$ for $\lambda u. u(\lambda x_0 \lambda x_1 \cdots \lambda x_{n-1}. t)$. Again, $(\lambda \bigotimes_{i<n} x_i. t)(\bigotimes_{i<n} t_i) = t[x_0 := t_0] \ldots [x_{n-1} := t_{n-1}]$.

Tally natural numbers may be encoded as affine lambda terms by $\ulcorner 0 \urcorner = (\lambda xy. x) \otimes (\lambda xy. x)$ and $\ulcorner n+1 \urcorner = (\lambda xy. y) \otimes \ulcorner n \urcorner$. Dyadic lists may be encoded as $\ulcorner \varepsilon \urcorner = (\lambda xyz. x) \otimes (\lambda xy. x)$ and $\ulcorner 0w \urcorner = (\lambda xyz. y) \otimes \ulcorner w \urcorner$ and $\ulcorner 1w \urcorner = (\lambda xyz. z) \otimes \ulcorner w \urcorner$.

We notice that $\ulcorner n \urcorner$, resp. $\ulcorner w \urcorner$ can be computed in linear time from n, resp. w, and vice versa. Of course, only a few functions f on natural numbers or dyadic lists can be represented by affine terms t in the sense that $t\ulcorner w \urcorner = \ulcorner f(w) \urcorner$. To begin, by the time bound on application any such function will be computable in quadratic time. Moreover, we believe that any such function can only inspect a constant-size prefix of its argument. For example, a function on dyadic lists which permutes the first five bits is representable in this way.

We write Λ_a for the set of closed affine lambda terms. Our subsequent development will be modular in the sense that Λ_a can be replaced by any other polynomial-time computable BCK-algebra in the sense of [8]. For example, we can take Turing machines with the application defined by $ex = \{e\}(x)$ if this result can be obtained in time at most $\wp(d(\ell(e) + \ell(x)))$ and 0 otherwise. Here ℓ is a length function defined inductively by $\ell(\langle x, y \rangle) = \ell(x) + \ell(y) + 1$, $\ell(x) = |x|$ otherwise. The *defect* d is $\ell(e) + \ell(x) - \ell(\{e\}(x))$ and we additionally require $\ell(\{e\}x) \leqslant \ell(e) + \ell(x)$. Finally, $\wp$ is a monotone, sublinear function satisfying $\wp(\ell(x)) \leqslant |x|$. For example, $\wp(x) = Cx^{1+\varepsilon}$ for arbitrary $\varepsilon > 0$ and appropriate C, see [7].

4.2. Realizability sets

Definition 3. Let X be a finite set of resource variables. A *realizability set over* X is a pair $A = (|A|, \Vdash_A)$, where $|A|$ is a set and $\Vdash_A \subseteq \mathscr{V}(X) \times \Lambda_a \times |A|$ is a ternary relation between valuations over X, affine lambda terms, and the set $|A|$. We write $\eta, t \Vdash_A a$ for $(\eta, t, a) \in \Vdash_A$.

The intuition behind $\eta, t \Vdash_A a$ is that a is an abstract semantic value, η measures the abstract size of a, and the affine lambda term t encodes the abstract value a.

Example 4. The following are some useful examples of realizability sets, cf. Section 5 of [4]:

(i) The realizability set $\mathbf{N}_x$ over $\{x\}$ of *tally natural numbers* ("of size at most x") is defined by $|\mathbf{N}_x| = \mathbb{N}$ and

$$\eta, t \Vdash_{\mathbf{N}_x} n \text{ if } t = \ulcorner n \urcorner \quad \text{and} \quad \eta(x) \geqslant n.$$

(ii) The realizability set $\mathbf{N}_x^2$ over $\{x\}$ of *dyadic lists* ("of length at most x") is defined by $|\mathbf{N}_x^2| = \{0, 1\}^*$ and

$$\eta, t \Vdash_{\mathbf{N}_x^2} w \quad \text{if } t = \ulcorner w \urcorner \quad \text{and} \quad \eta(x) \geqslant lh(w)$$

(iii) The realizability set I over $\emptyset$ is defined by $|I| = \{*\}$ and $\emptyset, \lambda x.x \Vdash_I *$

(iv) Given a substitution σ from X to Y, and a realizability set A over Y, then a new realizability set $A[\sigma]$ over X is defined by: $|A[\sigma]| = |A|$ and

$$\eta, t \Vdash_{A[\sigma]} a \quad \text{iff} \quad \sigma(\eta), t \Vdash_{A} a$$

The realizability sets $\mathbf{N}_x$ and $\mathbf{N}_x^2$ will turn out to be retracts of the denotations of the eponymous BLL formulas in our model. More precisely, the mediating functions ϕ and ψ described in the beginning of Section 3 are shown to be morphisms between realizability sets in the proof of Theorem 2.

In order to model the notion of *positive occurrence* of a resource variable in BLL formulas we introduce a corresponding concept for realizability sets.

Definition 5. Let A be a realizability set over X. We say that $x \in X$ is *positive* (resp. negative) in A, if for all $\eta, \eta' \in \mathscr{V}(X), t \in \Lambda_a, a \in |A|$ where η and η' agree on $X \backslash \{x\}$ and $\eta(x) \leqslant \eta'(x)$ (resp. $\eta(x) \geqslant \eta'(x)$) then $\eta, t \Vdash_A a$ implies $\eta', t \Vdash_A a$.

We notice that x is positive in $\mathbf{N}_x$ and $\mathbf{N}_x^2$.

Definition 6. Let A, B be realizability sets over some set X. A *morphism* from A to B is a function $f : |A| \to |B|$ satisfying the following condition:

There exists a function $e : \mathscr{V}(X) \to \Lambda_a$ such that $e(\eta)$ is computable in time $q(\eta)$ for some resource polynomial q and for each $\eta \in \mathscr{V}(X)$, $t \in \Lambda_a$, $a \in |A|$, we have

$$\eta, t \Vdash_A a \quad \text{implies} \quad \eta, e(\eta)t \Vdash_B f(a) \tag{1}$$

In this case we say that e *witnesses* f and write $A \xrightarrow[e]{f} B$ where in the notation the algorithm e is presumed to exist.

Example 7. The following are some useful examples of numerical-valued morphisms:

- A morphism $f : \mathbf{N}_x \to \mathbf{N}_x$ is a function $f : \mathbb{N} \to \mathbb{N}$ that is computable in time polynomial in the input n (not in $|n|$). Moreover, it satisfies $f(n) \leqslant n$ (by letting $\eta(x) = n$).
- Similarly, a morphism $f : \mathbf{N}_x^2 \to \mathbf{N}_x^2$ is a function $f : \{0,1\}^* \to \{0,1\}^*$ that is polytime computable (in the usual sense) and moreover satisfies $lh(f(w)) \leqslant lh(w)$.
- Let $p(x)$ be a unary resource polynomial in x and let σ be the substitution $[\{x\}; x := p]$.

A morphism $f : \mathbf{N}_x \to \mathbf{N}_x[\sigma]$ is a function $f : \mathbb{N} \to \mathbb{N}$ that is polytime computable in the input (as above) for which $f(n) \leqslant p(n)$. Composed with the above-mentioned retractions, these morphisms will be denotations of closed proofs of $\vdash \mathbf{N}_x \multimap \mathbf{N}_{p(x)}$.

The following lemma illustrates how realizability sets model the syntactical iteration lemma for BLL (cf. Lemma 6.2 of [5]).

Lemma 8 (Iteration lemma). *Let T be a realizability set over $\{x\}$ such that x is positive in T. Let $z : I \to T[x := 0]$ be a morphism (over $\emptyset$) and let $s : T \to T[x := x+1]$*

be a morphism (over $\{x\}$). The function $f : \mathbb{N} \to |T|$ defined by $f(n)=s^n(z(*))$ is a morphism from $\mathbf{N}_x$ to T.

Proof. The witnesses of z, s give rise to an element $e_z \in \Lambda_a$ and to a function $e_s : \mathbb{N} \to \Lambda_a$ such that

$$[x \mapsto 0], e_z || \underset{T}{\vdash\!\!-\!\!-} z(*)$$

and for each $n \in \mathbb{N}$,

$$[x \mapsto n], t || \underset{T}{\vdash\!\!-\!\!-} v \Rightarrow [x \mapsto n+1], e_s(n)t || \underset{T}{\vdash\!\!-\!\!-} s(v)$$

This is because by definition of substitution we have

$$\begin{array}{lcl} \emptyset, t || \underset{T[x:=0]}{\vdash\!\!-\!\!-} v & \iff & [x \mapsto 0], t || \underset{T}{\vdash\!\!-\!\!-} v \\ [x \mapsto n], t || \underset{T[x:=x+1]}{\vdash\!\!-\!\!-} v & \iff & [x \mapsto n+1], t || \underset{T}{\vdash\!\!-\!\!-} v \end{array}$$

We now have

$$[x \mapsto N], e_s(N-1)(e_s(N-2) \ldots (e_s(N-n)e_z) \ldots) || \underset{T}{\vdash\!\!-\!\!-} f(n)$$

whenever $n \leqslant N$. This follows by induction on n and the fact that x is positive in T.

We define $e : \mathbb{N} \to \Lambda_a$ recursively by

$$\begin{array}{l} e(0) = e_z, \\ e(N+1) = \lambda b \otimes r.b\ e_z\ (e_s(N)\ (e(N)\ r)). \end{array}$$

It follows that

$$[x \mapsto N], e(N)\ulcorner n \urcorner || \underset{T}{\vdash\!\!-\!\!-} f(n)$$

provided that $n \leqslant N$.

By induction on N one shows that the size of $e(N)$ is $O(N \cdot p(N))$ where $p(N)$ bounds the size of $e_s(N)$. This in turn shows that the primitive recursive definition of e yields a polytime algorithm.

Therefore, $\eta \mapsto e(\eta(x))$ witnesses the function $f(n)=s^n(z(*))$. $\square$

The following analogous version for dyadic lists (cf. Lemma 6.2 of [5]) is proved similarly.

Lemma 9 (Iteration lemma for dyadic lists). *Let T be a realizability set over $\{x\}$ such that x is positive in T. Let $z : I \to T[x := 0]$ be a morphism (over $\emptyset$) and let $s_0, s_1 : T \to T[x := x+1]$ be morphisms (over $\{x\}$). The function $f : \{0,1\}^* \to |T|$ defined by $f(\varepsilon)=z$ and $f(iw)=s_i(f(w))$ is a morphism from $\mathbf{N}_x^2$ to T.*

We remark that one can also prove more general versions of the preceding iteration lemmas allowing for extra resource variables in T as parameters.

Proposition 10. *Let X be a finite set (of resource variables). Realizability sets over X and morphisms between them form a category $\mathscr{B}(X)$ such that the mapping $A=(|A|, \Vdash_A) \mapsto |A|$ extends to a functor from $\mathscr{B}(X)$ to the category Set of sets. This means that composition in $\mathscr{B}(X)$ is given by ordinary set-theoretic composition of functions.*

Proof. The identity function $id: |A| \longrightarrow |A|$ is witnessed by the algorithm $e(\eta)=\lambda x.x$ which is clearly polytime computable.

If $A_0 \xrightarrow[e_0]{f} A_1$ and $A_1 \xrightarrow[e_1]{g} A_2$ then the composition $g \circ f: |A_0| \longrightarrow |A_2|$ can be witnessed by

$$e(\eta) = \lambda z.e_1(\eta)(e_0(\eta)z) = B\ e_1(\eta)\ e_0(\eta),$$

where $B=\lambda x y z.x(yz)$. Now e is polytime using the fact that application in Λ_a is polytime. □

Recall the definition in Example 4 of the realizability set $A[\sigma]$ over X when A is a realizability set over Y and σ is a substitution from X to Y.

Proposition 11. *Let σ be a substitution from X to Y. The assignment $A \mapsto A[\sigma]$ extends to a functor $-[\sigma]: \mathscr{B}(Y) \to \mathscr{B}(X)$ with $f[\sigma] \stackrel{\text{def}}{=} f$.*

Proof. We have to show that if $A \xrightarrow[e]{f} B$ then we can find e' so that $A[\sigma] \xrightarrow[e']{f} B[\sigma]$. Unfolding the definitions reveals that $e'(\eta)=e(\sigma(\eta))$ does the job. □

This allows us to consider morphisms between realizability sets over different sets of resource variables. Namely, if $X \subseteq Z$ we have a "weakening substitution" $weak_{X,Z}$ from Z to X given by $weak_{X,Z}(x)=x$. Thus, if A is a realizability set over X and B is a realizability set over Y we can consider morphisms from $A[weak_{X,X\cup Y}]$ to $B[weak_{Y,X\cup Y}]$. Such a morphism is a function $f: |A| \to |B|$ such that there exists an algorithm $e: \mathscr{V}(X \cup Y) \to \Lambda_a$ such that $e(\eta)$ is computable in time $q(\eta)$ for some resource polynomial q and

$$\eta|_X, t \Vdash_A a \quad \text{implies} \quad \eta|_Y, e(\eta)t \Vdash_B b$$

where $\eta|_X$ denotes the restriction of η to X. We shall sloppily refer to such morphisms as being morphisms from A to B. In this sense, the only morphism from $\mathbf{N}_x$ to $\mathbf{N}_y$ where $x \neq y$ is the constant zero function.

The following is immediate.

Lemma 12. *Suppose that $A \in \mathscr{B}(Y)$ is a realizability set and $\sigma, \sigma': X \to Y$ are substitutions. Suppose furthermore that $\sigma(y) \leqslant \sigma'(y)$ if y occurs positively in A, that $\sigma(y) \geqslant \sigma'(y)$ if y occurs negatively in A, and that $\sigma(y)=\sigma'(y)$ otherwise. Then the identity function is a morphism from $A[\sigma]$ to $A[\sigma']$.*

4.3. The category of realizability sets

We will now show that the categories $\mathscr{B}(X)$ have the appropriate categorical structure to model the BLL connectives.

Definition 13. We define the following monoidal structure on $\mathscr{B}(X)$:

- $I=(|I|, \Vdash_{I})$, where $|I|=\{*\}$ and $\eta, t \Vdash_{I} *$ for $t=\lambda x.x$ and η arbitrary.
- If A_1, A_2 are realizability sets over X we define $A_1 \otimes A_2$ by $|A_1 \otimes A_2|=|A_1| \times |A_2|$ and $\eta, t \Vdash_{A_1\otimes A_2} (a_1, a_2)$ iff $t=t_1 \otimes t_2$, where $\eta, t_i \Vdash_{A_i} a_i$ for $i=1,2$.

Proposition 14. *Let $f: A \rightarrow B$ be a $\mathscr{B}(X)$ morphism, $C \in \mathscr{B}(X)$. Then*
(i) *The function $f \otimes C: |A \otimes C| \rightarrow |B \otimes C|$ defined by $(f \otimes C)(a,c)=(f(a),c)$ is a morphism from $A \otimes C$ to $B \otimes C$.*
(ii) *The canonical set-theoretic maps $|A \otimes (B \otimes C)| \rightarrow |(A \otimes B) \otimes C|$, $|A \otimes B| \rightarrow |B \otimes A|$, and $|A \otimes I| \rightarrow |A|$ induce isomorphisms between the associated objects.*
(iii) *For appropriately typed substitution $I[\sigma]=I$ and $(A_1 \otimes A_2)[\sigma]=A_1[\sigma] \otimes A_2[\sigma]$.*
This says in particular that $\mathscr{B}(X)$ is a symmetric monoidal category, and the forgetful functor $\mathscr{B}(X) \rightarrow Set$ is a monoidal functor. Clause (iii) *states that substitution is a monoidal functor. This says that the collection of the categories $\mathscr{B}(X)$ forms a fibred* (*indexed*) *symmetric monoidal category* (*cf.* [16], *Section* 2.5(3)).

Proof. Ad (i). If e witnesses f then we define $e'(\eta)=\lambda x \otimes y.e(\eta)x \otimes y$. Obviously, e' witnesses $f \otimes C$ and, since $e'(\eta)=P\ e(\eta)$ for some $P \in \Lambda_a$ the function e' is polytime. The other cases are analogous. □

Proposition 15. *For any two objects $A, B \in \mathscr{B}(X)$, there is a linear function space object $A \multimap B \in \mathscr{B}(X)$, where*
(i) $|A \multimap B|=|A| \Rightarrow |B|$,
(ii) $\eta, t \Vdash_{A\multimap B} f$ *iff whenever* $\eta, t' \Vdash_{A} a$ *then* $\eta, t\,t' \Vdash_{B} f(a)$.
This structure makes $\mathscr{B}(X)$ a symmetric monoidal closed category, *i.e. there is a natural bijection $\mathscr{B}(X)(C \otimes A, B) \cong \mathscr{B}(X)(C, A \multimap B)$. Moreover*, $(A \multimap B)[\sigma]=A[\sigma] \multimap B[\sigma]$ *so that $-[\sigma]$ is a monoidal closed functor.*

Proof. The evaluation map $|A \otimes (A \multimap B)| \rightarrow |B|$ given by $(a,f) \mapsto f(a)$ is witnessed by $e(\eta)=\lambda x \otimes y.yx$. If $C \otimes A \xrightarrow[e]{f} B$ then $\lambda(f): C \longrightarrow A \multimap B$ given by $\lambda(f)(c)(a)=f(c,a)$ is witnessed by $e'(\eta)=\lambda xy.e(\eta)(x \otimes y)$. Just as in Proposition 14(i) we have $e'(\eta)=P\ e(\eta)$ for some $P \in \Lambda_a$, which establishes that e' is polytime. □

We notice that the forgetful functor $\mathscr{B}(X) \rightarrow Set$ is also monoidal closed, i.e. sends $\otimes$ to $\times$ and $\multimap$ to $\Rightarrow$.

Definition 16. Given a polynomial $p \in \mathscr{P}(X)$ and a realizability set A over $X \cup \{x\}$ where $x \notin X$ then we define a realizability set $!_{x<p}A$ over X (i.e. x is "bound" by $!_{x<p}$) by

- $|!_{x<p}A| = |A|$,
- $\eta, t \Vdash_{!_{x<p}A} a$ if
  - $t = \bigotimes_{i<p(\eta)} t_i$ for some family $(t_i)_{i<p(\eta)}$,
  - $\eta[x \mapsto i], t_i \Vdash_A a$ for each $i < p(\eta)$.

Whenever we write $!_{x<p}A$ in the sequel we implicitly assume that x does not occur in p.

Proposition 17. *If $f : A \to B \in \mathscr{B}(X \cup \{x\})$ is a morphism then $!_{x<p}(f) \stackrel{\text{def}}{=} f$ is a morphism $!_{x<p}A \to !_{x<p}B$. This says that $!_{x<p}$ extends to a functor from $\mathscr{B}(X \cup \{x\})$ to $\mathscr{B}(X)$ which is mapped to the identity by the forgetful functor to Set.*

Proof. If $A \xrightarrow[e]{f} B$ then we can witness $f : !_{x<p}A \to !_{x<p}B$ by

$$e'(\eta) = \lambda \bigotimes_{i<p(\eta)} x_i. \bigotimes_{i<p(\eta)} e(\eta)x_i.$$

We notice that

$$e'(\eta) = P\, \underbrace{e(\eta) \ldots e(\eta)}_{p(\eta)\ \text{times}}$$

for some $P \in \Lambda_a$ so that $e'(\eta)$ is computable in time $O((p(\eta)q(\eta))^2)$ if $e(\eta)$ is computable in $O(q(\eta))$. Note that this is a rather generous estimate. $\square$

We now show that we have the appropriate categorical structure to interpret the rules of BLL.

Proposition 18. *The following are morphisms*:

$$\begin{aligned}
&\varepsilon_A : !_{x<1}A \to A[x := 0] \textit{ where } \varepsilon_A(a) = a,\\
&e_A : !_{x<0}A \to I \textit{ where } e_A(a) = *,\\
&d_A : !_{x<p+q}A \to !_{x<p}A \otimes !_{y<q}A[x := p + y],\\
&\quad \textit{where } d_A(a) = (a, a),\\
&\delta_A : !_{y<\sum_{x<p} q(x)}A \to !_{x<p}!_{z<q(x)}A\left[y := z + \sum_{\xi<x} q(\xi)\right],\\
&\quad \textit{where } \delta_A(a) = a \textit{ and } p \in \mathscr{P}(X), q \in \mathscr{P}(X \cup \{x\})\\
&\quad \textit{and } x, y \textit{ are fresh},\\
&\tau_{A,B} : !_{x<p}A \otimes !_{x<p}B \to !_{x<p}(A \otimes B),\\
&\quad \textit{where } \tau_{A,B}(a, b) = (a, b),\\
&w_A : !_{x<p}A \to !_{x<q}A \textit{ where } w_A(a) = a \textit{ and } q \leqslant p.
\end{aligned}$$

Proof. The map ε_A can be witnessed by $e(\eta)=\lambda\bigotimes_{i<1}x_i.x_0$. The map e_A may be witnessed by $e(\eta)=\lambda y.\lambda x.x$. The map d_A may be witnessed by

$$e(\eta)=\lambda\bigotimes_{i<(p+q)(\eta)}x_i.\left(\bigotimes_{i<p(\eta)}x_i\right)\otimes\left(\bigotimes_{j<q(\eta)}x_{j+p(\eta)}\right).$$

To see this, assume $\eta,t\Vdash_{!_{x<p+q}A} a$, i.e., $t=\bigotimes_{i<(p+q)(\eta)}t_i$ and $\eta[x\mapsto i],\ t_i\Vdash_A a$. Then for each $i<p(\eta)$ we have $\eta[x\mapsto i],\ t_i\Vdash_A a$, so $\eta,\bigotimes_{i<p(\eta)}t_i\Vdash_{!_{x<p}A} a$. Moreover, for each $j<q(\eta)$ we have $\eta[x\mapsto j+p(\eta)],t_{j+p(\eta)}\Vdash_A a$, hence $\eta[y\mapsto j],t_{j+p(\eta)}\Vdash_{A[x:=p+y]} a$. Therefore, $\eta,\bigotimes_{j<q(\eta)}t_{j+p(\eta)}\Vdash_{!_{y<q}A[x:=p+y]} a$, so the result follows.

Next, δ_A may be witnessed by

$$e(\eta)=\lambda\bigotimes_{j<\sum_{i<p(\eta)}q(\eta[x\mapsto i])}x_j.\bigotimes_{i<p(\eta)}\bigotimes_{j<q(\eta[x\mapsto i])}x_{j+\sum_{k<i}q(\eta[x\mapsto k])}.$$

The verification is similar to the one of d_A.

Finally, $\tau_{A,B}$ may be witnessed by

$$e(\eta)=\lambda u\otimes v.u\left(\lambda x_1x_2\ldots x_{p(\eta)-1}.v\left(\lambda y_1y_2\ldots y_{p(\eta)-1}.\bigotimes_{i<p(\eta)}x_i\otimes\bigotimes_{i<p(\eta)}y_i\right)\right).$$

We omit the "waste" morphism w_A, which is obvious. □

4.4. *Interpreting the syntax in $\mathscr{B}$*

We shall now give the details of the interpretation of the syntax of BLL in terms of realizability sets and morphisms between them.

Definition 19. Let X be a set of resource variables. A *second-order environment* over X is a partial function ρ which assigns to a second-order variable α of arity n a pair (l,C) such that

- $l=(y_0,\ldots,y_{n-1})$ is a list of n pairwise different resource variables not occurring in X,
- C is a realizability set over $X\cup\{y_0,\ldots,y_{n-1}\}$ in which the y_i are positive.

For second-order environment ρ we write $|\rho|$ for the mapping $\alpha\mapsto|C|$ when $\rho(\alpha)=(l,C)$.

If $\sigma:X\to Y$ is a substitution and ρ is a second-order environment over Y we define a second-order environment $\rho[\sigma]$ over X by $\rho[\sigma](\alpha)=(l,C[\sigma])$ when $\rho(\alpha)=(l,C)$. We assume here that the variables in l are not contained in Y. Otherwise, the substitution cannot be defined.

Recall the set-theoretic semantics $[\![-]\!]$ defined in Section 3. From now on, we will write $[\![-]\!]^{\mathrm{Set}}$ for this set-theoretic semantics to distinguish it from a realizability semantics $[\![-]\!]^{\mathscr{B}}$ which we now define.

Let A be a BLL formula with free resource variables in X and ρ be an environment over X defined on all free second-order variables occurring in A. We assume, without loss of generality, that all bound variables in A and in ρ are distinct from each other.

By induction on A we define a realizability set $[\![A]\!]^{\mathscr{B}}_{\rho}$ such that

$$|[\![A]\!]^{\mathscr{B}}_{\rho}| = [\![A]\!]^{\mathrm{Set}}_{|\rho|}.$$

The defining clauses are as follows:

$$[\![\alpha(\vec{p})]\!]^{\mathscr{B}}_{\rho} = C[\sigma],$$

where $\rho(\alpha) = ((y_0, \dots, y_{n-1}), C)$ and $\vec{p} = (p_0, \dots, p_{n-1})$ and $\sigma(y_i) = p_i$,

$$[\![A \otimes B]\!]^{\mathscr{B}}_{\rho} = [\![A]\!]^{\mathscr{B}}_{\rho} \otimes [\![B]\!]^{\mathscr{B}}_{\rho},$$

$$[\![A \multimap B]\!]^{\mathscr{B}}_{\rho} = [\![A]\!]^{\mathscr{B}}_{\rho} \multimap [\![B]\!]^{\mathscr{B}}_{\rho},$$

$$[\![!_{x<p} A]\!]^{\mathscr{B}}_{\rho} = !_{x<p} [\![A]\!]^{\mathscr{B}}_{\rho[weak_{X \cup \{x\}, X}]},$$

$$[\![\forall \alpha A]\!]^{\mathscr{B}}_{\rho} = \left(\prod_{C \in \mathscr{U}} [\![A]\!]^{\mathrm{Set}}_{|\rho|[\alpha \mapsto C]}, \Vdash_{[\![\forall \alpha A]\!]^{\mathscr{B}}_{\rho}} \right),$$

where

$$\eta, t \Vdash_{[\![\forall \alpha A]\!]^{\mathscr{B}}_{\rho}} f \iff \eta, t \Vdash_{[\![A]\!]^{\mathscr{B}}_{\rho[\alpha \mapsto (l, C)]}} f_{|C|} \text{ for all } (l, C) \text{ as in Definition 19.}$$

The following lemmas are immediate by structural induction.

Lemma 20. *Let σ be a substitution from X to $Y = \{y_0, \dots, y_{n-1}\}$ and let A be a BLL formula with free resource variables from Y and let ρ be an environment over Y such that $\rho[\sigma]$ is defined. Then*

$$[\![A[y_0 := \sigma(y_0), \dots, y_{n-1} := \sigma(y_{n-1})]]\!]^{\mathscr{B}}_{\rho[\sigma]}. = [\![A]\!]^{\mathscr{B}}_{\rho}[\sigma].$$

Lemma 21. *Let A be a BLL formula possibly containing the second-order variable α of arity n. Let B be a BLL formula containing free resource variables $\{y_0, \dots, y_{n-1}\}$ which do not occur in A. Then*

$$[\![A[\alpha := \lambda y_0 \dots \lambda y_{n-1} B]]\!]^{\mathscr{B}}_{\rho} = [\![A]\!]_{\rho[\alpha \mapsto ((y_0, \dots, y_{n-1}), [\![B]\!]^{\mathscr{B}}_{\rho})]}.$$

Lemma 22. *Let A be a BLL formula in which resource variable x occurs positively (resp. negatively) and ρ be an appropriate second-order environment. Then x is positive (resp. negatively) in $[\![A]\!]^{\mathscr{B}}_{\rho}$.*

Theorem 23 (Soundness). *Let π be a proof of a sequent $A_1, \dots, A_n \vdash B$ involving resource variables from X. Let ρ be a second-order environment binding all the second-order variables occurring in the sequent. Then the set-theoretic function*

$$[\![\pi]\!]^{\mathrm{Set}}_{|\rho|} : [\![A_1 \otimes \dots \otimes A_n]\!]^{\mathrm{Set}}_{|\rho|} \to [\![B]\!]^{\mathrm{Set}}_{|\rho|}$$

is a morphism of realizability sets from $[\![A_1 \otimes \cdots \otimes A_n]\!]^{\mathscr{B}}_{\rho}$ *to* $[\![B]\!]^{\mathscr{B}}_{\rho}$. *Recall that* $[\![A_1 \otimes \cdots \otimes A_n]\!]^{\mathsf{Set}}_{|\rho|} = |[\![A_1 \otimes \cdots \otimes A_n]\!]^{\mathscr{B}}_{\rho}|$ *and* $[\![B]\!]^{\mathsf{Set}}_{|\rho|} = |[\![B]\!]^{\mathscr{B}}_{\rho}|$.

Proof. By induction on BLL derivations using Lemma 12 for the axiom; using Propositions 14, 15, 17, 18 for the term formers associated with $\otimes, \multimap, !_{x<p}$, and using the above three lemmas for universal application and abstraction. We also make use of obvious translations of syntactic constructs into categorical combinators, e.g. application into evaluation and composition or storage into a combination of δ, τ, and functoriality of $!_{x<p}$. □

Proof of Theorem 2. Applying the iteration principle (Lemma 8) to the denotations of 0 and S shows that the function $\phi : \mathbb{N} \to [\![\mathbf{N}_x]\!]^{\mathsf{Set}}$ is a morphism in $\mathscr{B}(\{x\})$ from $\mathbf{N}_x$ to $[\![\mathbf{N}_x]\!]^{\mathscr{B}}$. Similarly, the function $\psi : [\![\mathbf{N}_{p(x)}]\!]^{\mathsf{Set}} \to \mathbb{N}$ is a morphism in the other direction by instantiating the second-order variable α with $\mathbf{N}_x$ and afterwards applying it to the zero and successor of the realizability set $\mathbf{N}_x$. Thus the function $f = \psi \circ [\![\pi]\!]^{\mathsf{Set}} \circ \phi$ in the theorem is a morphism. The result follows by the analysis in Example 7. We proceed analogously in the case of dyadic lists, using the corresponding iteration principle. □

5. Additional remarks

We notice that for any realizability set A over X the unique function $|A| \to \{*\}$ is a morphism $A \to I$ of realizability sets witnessed by $e(\eta) = \lambda y.\lambda x.x$. In particular, this gives projections for $\otimes$. This shows that we can model an affine variant of BLL which has the following additional rule

$$\mathit{Weak} : \frac{\Gamma \vdash B}{\Gamma, A \vdash B}$$

In particular, we see that this rule does not add computational strength.

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 it is consistent with constructive set theories [10,15]. We found it convenient to assume the existence of such a $\mathscr{U}$ because it allows the use of informal set-theoretic arguments (provided they are constructive). For the reader who feels uneasy about such sleight-of-hand, we offer the following ways of making this rigorous (all of which, however, complicate the argument):

- Formalize the entire discussion in the Calculus of Inductive Constructions [3].
- Formalize the entire discussion in a realizability topos [7,15]. More explicitly, we can stipulate that the carrier sets of realizability sets must be a subquotient (by a per, i.e. by a partial equivalence relation) of the set of untyped lambda terms (not necessarily linear!). And furthermore, a morphism between realizability sets must be uniformly tracked by an untyped lambda term, in the obvious sense. This would allow one to interpret polymorphic quantification as intersection of pers, in the familiar manner.

References

[1] H.P. Barendregt, The Lambda Calculus, North-Holland, Amsterdam, 1984.

[2] P. Baillot, Stratified coherent spaces: a denotational semantics for light linear logic, Tech. Report EDI-INF-RR-0025 submitted, available from Informatics Division Report Series, University of Edinburgh, August 2000, http://www.informatics.ed.ac.uk/publications/report/0025.html

[3] T. Coquand, C. Paulin-Mohring, Inductively Defined Types, in: Lecture Notes in Computer Science, Vol. 389, Springer, Berlin, 1989.

[4] J.-Y. Girard, Light linear logic, Inform. and Comput. 143 (1998).

[5] J.-Y. Girard, A. Scedrov, P.J. Scott, Bounded linear logic, Theoret. Comput. Sci. 97 (1992) 1–66.

[6] A. Goerdt, Characterizing complexity classes by higher-type primitive recursive definitions, Theoret. Comput. Sci. 100 (1992) 45–66.

[7] M. Hofmann, Linear types and non-size-increasing polynomial time computation, Inform. Comput. 183 (1999) 57–85.

[8] M. Hofmann, Type systems for polynomial-time computation, Darmstadt University of Technology, Habilitationsschrift, 1999, appears as Tech. Report ECS-LFCS-99-406, Department of Computer Science, University of Edinburgh, A revised and abridged version has appeared as “Safe recursion with higher types and BCK algebra” in Ann. Pure Appl. Logic 104 (2000) 113–166.

[9] M. Hofmann, Programming Languages capturing complexity classes, SIGACT News (Logic Column 9), ftp.research.bell-labs.com/dist/riecke/hofmann.ps.gz

[10] M. Hyland, A small complete category, Ann. Pure Appl. Logic 40 (1988) 135–165.

[11] M. Kanovich, M. Okada, A. Scedrov, Phase semantics for light linear logic, Theoret. Comput. Sci. 294 (2003) 525–549 (extended abstract in ENTCS, Vol. 6, 1997).

[12] D. Leivant, Stratified functional programs and computational complexity, Proc. 20th IEEE Symp. on Principles of Programming Languages (POPL’93), Charleston, South Africa, 1993.

[13] D. Leivant, J.-Y. Marion, Lambda calculus characterisations of polytime, Fund. Inform. 19 (1993) 167–184.

[14] S. Mac Lane, Categories for the Working Mathematician, 2nd Edition, Springer, Berlin, 1998.

[15] A. Pitts, Polymorphism is set-theoretic, constructively, in: D.H. Pitt (Ed.), Category Theory and Computer Science, Lecture Notes in Computer Science, Vol. 283, Springer, Berlin, 1987, pp. 12–39.

[16] R. Seely, Linear logic, $*$-autonomous categories, and cofree coalgebras, in: J. Gray, A. Scedrov, (Eds.), Categories in Computer Science and Logic, Contemporary Mathematics, Vol. 92, American Mathematical Society, Providence, RI, 1989.