Generalized Bounded Linear Logic and its Categorical Semantics

Yōji Fukihara[1](✉) and Shin-ya Katsumata[2]

[1] Kyoto University, Kyoto, Japan fukihara@kurims.kyoto-u.ac.jp
[2] National Institute of Informatics, Tokyo, Japan s-katsumata@nii.ac.jp

Abstract. We introduce a generalization of Girard et al.'s BLL called GBLL (and its affine variant GBAL). It is designed to capture the core mechanism of dependency in BLL, while it is also able to separate complexity aspects of BLL. The main feature of GBLL is to adopt a multi-object pseudo-semiring as a grading system of the !-modality. We analyze the complexity of cut-elimination in GBLL, and give a translation from BLL with constraints to GBAL with positivity axiom. We then introduce *indexed linear exponential comonads* (*ILEC* for short) as a categorical structure for interpreting the !-modality of GBLL. We give an elementary example of ILEC using folding product, and a technique to modify ILECs with symmetric monoidal comonads. We then consider a semantics of BLL using the folding product on the category of assemblies of a BCI-algebra, and relate the semantics with the realizability category studied by Hofmann, Scott and Dal Lago.

Keywords: Linear Logic · Categorical Semantics · Linear Exponential Comonad · Graded Comonad

1 Introduction

Girard's *linear logic* is a refinement of propositional logic by restricting weakening and contraction in proofs [15]. Linear logic also has an *of-course modality* !, which restores these structural rules to formulas of the form $!A$.

Later, Girard et al. extended the !-modality with quantitative information so that usage of !-modal formulas in proofs can be quantitatively controlled [16]. This extension, called *bounded linear logic* (BLL for short), is successfully applied to a logical characterization of P-time computations.

Their extension takes two steps. First, the !-modality is extended to the form $!_r A$, where the index r is an element of a semiring [16, Section 2.4]. The index r is called *grade* in modern terminology [11,13]. This extension and its variants have been employed in various logics and programming languages [7,30,14,26,28]. The categorical structure corresponding to $!_r A$ is identified as *graded linear exponential comonad* [7,13,22].

Second, the $!_r$-modality is further extended to the form $!_{x<p} A$, where p is a polynomial (called *resource polynomial*) giving the upper bound of x [16, Section 3]. The formula $!_{x<p} A$ also binds free occurrences of the resource variable

x in resource polynomials in A. Therefore, in BLL, both formulas and resource polynomials depend on the values stored in free resource variables. This dependency mechanism significantly increases the expressiveness of BLL, leading to a characterization of P-time complexity.

This characterization result was later revisited through a *realizability semantics* of BLL [16,19,10]. Inside this semantics, however, mechanisms for controlling complexity of program execution are hard-coded, and it is not very clear which semantics structure realizes the dependency mechanism of BLL. This leads us to seek a logical and categorical understanding of BLL's dependency mechanism hidden underneath the complexity-related features, such as resource polynomials and computability constraints.

As a result of the quest, we propose a generalization of BLL called GBLL, and study its categorical semantics. The central idea of the generalization is to replace the grading semiring of the $!_r$-modality with a particular *multi-object pseudo-semiring* realized as a 2-category. Let us see how this replacement works. In GBLL, each formula is formed by deriving a judgment of the form $\Delta \vdash A$, where Δ is a set (called *index set*) and A is a raw formula. We may think that such a well-formed formula $\Delta \vdash A$ denotes a Δ-indexed family $\{[\![A]\!]_i\}_{i\in\Delta}$ of denotations. The formation rule for !-modal formula in GBLL is the following:

$$\frac{\Delta' \vdash A \quad f \in \mathbf{Set}(\Delta, (\Delta')^*)}{\Delta \vdash\, !_f A} \qquad ((\_)^*\text{: Kleene closure})$$

where the function f abstractly represents dependency. This modality is enough to express the $!_{x<p}$-modality of BLL: we express the bindig $x < p$ under a resource variable context $\vec{y}$ as the *function* $f_p(\vec{y}) = (\vec{y}, 0) \cdots (\vec{y}, p(\vec{y}) - 1)$ that returns the list of environments extended with values less than $p(\vec{y})$. Then the denotation of the $!_f A$-modality is given by a variable-arity operator D. For each index $i \in \Delta$, the denotation is given by applying D to the denotations obtained by mapping A to list $f(i)$:

$$[\![!_f A]\!]_i = D([\![A]\!]_{j_1}, \cdots, [\![A]\!]_{j_n}) \text{ where } j_1 \cdots j_n = f(i).$$

A simple example of a variable-arity modal operator is the *folding product* $D(X_1, \cdots, X_n) = X_1 \otimes \cdots \otimes X_n$.

The pseudo-semiring structure on the class of functions of the form $\Delta \to (\Delta')^*$ is given as follows. For the multiplication $g \bullet f$, we adopt the *Kleisli composition* of the free monoid monad $(\_)^*$, while for the addition $f+g$, the pointwise concatenation $(f + g)(x) = f(x)g(x)$. However, these operations fail to satisfy one of the semiring axioms: $(f + g) \bullet h = f \bullet h + g \bullet h$. To fix this, we introduce (pointwise) list permutations as 2-cells between functions of type $\Delta \to (\Delta')^*$. These data form a 2-category **Idx**, which may be seen as a multi-object pseudo-semiring. Weakening, contraction, digging and dereliction in GBLL interact with these operations, much like the $!_r$-modality in [7].

We first study syntactic properties of GBLL. We introduce cut-elimination to GBLL and study its complexity property. It turns out that the proof technique used in BLL naturally extends to GBLL — as done in [16], we classify cuts into *reducible* and *irreducible* ones, introduce *proof weight*, and show that the reduction steps of reducible cuts will terminate in cubic time of proof weights. We also examine the expressive power of GBLL by giving a translation from an extension of BLL with *constraints* that are seen in Dal Lago et al.'s QBAL [10].

We next give a categorical semantics of GBLL. We introduce the concept of *indexed linear exponential comonad* (*ILEC*); it is an **Idx**-graded linear exponential comonad satisfying a commutativity condition with respect to an underlying indexed SMCCs. Then, we present a construction of ILEC from a symmetric monoidal closed category $\mathbb{C}$ with a symmetric monoidal comonad on it. We apply this construction to the case where $\mathbb{C}$ is the category of assemblies over a BCI algebra [2,20], and relate the semantics of GBLL with the constructed ILEC and the realizability category studied in [19,10].

Acknowledgment The first author was supported by JST ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603). The authors are grateful to anonymous reviewers for comments, and Masahito Hasegawa, Naohiko Hoshino, Clovis Eberhart and Jérémy Dubut for fruitful discussions.

Preliminaries For a set Δ, by Δ^* we mean the set of finite sequences of Δ. The empty sequence is denoted by $()$. Juxtaposition of Δ^*-elements denotes the concatenation of sequences. For $x \in \Delta^*$, by $|x|$ we mean the length of x. We identify a natural number n and the set $\{0, \cdots, n-1\}$; note that $0 = \emptyset$. We also identify a sequence $x \in \Delta^*$ and the function "$\lambda i \in |x|$. the i-th element of x".

2 Generalized Bounded Linear Logic

2.1 Indexing 2-Category

We first introduce a 2-category **Idx** (and its variant $\mathbf{Idx}_a$), which may be seen as a multi-object pseudo-semiring. It consists of the following data[3]: 0-cells are sets (called index sets), and the hom-category $\mathbf{Idx}(\Delta, \Delta')$, which is actually a groupoid, is defined by:

- An object (1-cell) is a function $f : \Delta \to (\Delta')^*$.
- A morphism (2-cell) from f to g in $\mathbf{Idx}(\Delta, \Delta')$ is a Δ-indexed family of bijections $\{\sigma_x : |g(x)| \to |f(x)|\}_{x \in \Delta}$ such that $f(x)(\sigma_x(i)) = g(x)(i)$.

The identity 1-cell and the composition of 1-cells in **Idx** are denoted by i_Δ and $(\bullet)$, respectively. The composition is defined by $(g \bullet f)(x) \overset{\text{def}}{=} g(y_1) \cdots g(y_n)$ where $y_1 \cdots y_n = f(x)$. The hom-category $\mathbf{Idx}(\Delta, \Delta')$ has a symmetric strict monoidal structure:

- the monoidal unit is the constant empty-sequence function $0(x) = ()$,

[3] This is a full sub-2-category of the Kleisli 2-category $\mathbf{CAT}_{\mathcal{S}}$, where $\mathcal{S}$ is the 2-monad of symmetric strict monoidal category [21].

– the tensor product of f, g, denoted by $f + g$, is defined by the index-wise concatenation $(f + g)(x) \overset{\mathrm{def}}{=} f(x)g(x)$.

We write $J : \mathbf{Set} \to \mathbf{Idx}$ for the inclusion, namely $J\Delta = \Delta$ and $(Jf)(x) = f(x)$ (the singleton sequence).

Proposition 2.1. *The composition $\bullet$ is symmetric strong monoidal in each argument. Especially, we have*

$$f \bullet 0 = 0 \quad 0 \bullet f = 0 \quad f \bullet (g + h) = f \bullet g + f \bullet h \quad (f + g) \bullet h \cong f \bullet h + g \bullet h.$$

We also define $\mathbf{Idx}_a$ by replacing "bijection" in the definition of 2-cell of $\mathbf{Idx}$ with "injection". The hom-category $\mathbf{Idx}_a(\Delta, \Delta')$ has the 1-cell 0 as the terminal object, hence is a symmetric *affine* monoidal category.

2.2 Formulas and Proofs

Definition of GBLL Formulas We first fix a set-indexed sets $\{\mathcal{A}(\Delta)\}_{\Delta \in \mathbf{Set}}$ of atomic propositions. Formulas are defined by the following BNF:

$$A ::= a \star r \mid A \otimes A \mid A \multimap A \mid !_f A$$

where $a \in \mathcal{A}(\Delta)$ for some set Δ, r is a function (called *reindexing function*) and f is a 1-cell in $\mathbf{Idx}$. Formula formation rules are introduced to derive the pair $\Delta \vdash A$ of an index set Δ and a formula A. They are defined as follows:

$$\frac{a \in \mathcal{A}(\Delta') \quad r \in \mathbf{Set}(\Delta, \Delta')}{\Delta \vdash a \star r} \qquad \frac{\Delta \vdash A \quad \Delta \vdash B}{\Delta \vdash A \otimes B} \qquad \frac{\Delta \vdash A \quad \Delta \vdash B}{\Delta \vdash A \multimap B}$$

$$\frac{\Delta' \vdash A \quad f \in \mathbf{Idx}(\Delta, \Delta')}{\Delta \vdash !_f A}$$

The formula $a \star r$ represents the atomic formula a precomposed with a reindexing function r. We write $\mathbf{Fml}(\Delta) = \{A \mid \Delta \vdash A\}$.

We next introduce the reindexing operation on formulas.

Definition 2.1. *For a reindexing function $r \in \mathbf{Set}(\Delta, \Delta')$, we define the* reindexing operator $(\_)|_r : \mathbf{Fml}(\Delta') \to \mathbf{Fml}(\Delta)$ *along r by*

$$a \star r|_{r'} \overset{\mathrm{def}}{=} a \star (r \circ r'), \qquad (A \otimes B)|_r \overset{\mathrm{def}}{=} A|_r \otimes B|_r,$$
$$(A \multimap B)|_r \overset{\mathrm{def}}{=} A|_r \multimap B|_r, \qquad (!_f A)|_r \overset{\mathrm{def}}{=} !_{f \bullet Jr} A.$$

We routinely extend reindexing operators to sequences of formulas well-formed under a common index set.

We quotient the set of well-formed formulas by the least congruent equivalence relation generated from the following binary relation:

$$\{(!_{Jr \bullet f} A, !_f (A|_r)) \mid r \in \mathbf{Set}(\Delta', \Delta''), f \in \mathbf{Idx}(\Delta, \Delta'), \Delta'' \vdash A\} \tag{2.1}$$

We see some formations of formulas in GBLL.

Example 2.1. Let us illustrate how a formula $!_{y<x^2}!_{z<x+y}A$ in BLL is represented in GBLL; here we assume that x, y, z are the only resource variables used in this formula. We first introduce a notation. Let E be a mathematical expression using variables $\mathbf{x}_1 \cdots \mathbf{x}_n$. Then by $[E]_n : \mathbb{N}^n \to (\mathbb{N}^{n+1})^*$ we mean the function

$$[E]_n(\vec{x}) = (\vec{x}, 0)(\vec{x}, 1) \cdots (\vec{x}, E[x_1/\mathbf{x}_1, \cdots, x_n/\mathbf{x}_n] - 1) \quad (\vec{x} \triangleq (x_1, \cdots, x_n) \in \mathbb{N}^n)$$

For instance, $[\mathbf{x}_1^2]_1(x) = (x, 0), \cdots, (x, x^2 - 1)$. Then from a well-formed formula $\mathbb{N}^3 \vdash A$, we obtain $\mathbb{N} \vdash !_{[\mathbf{x}_1^2]_1}!_{[\mathbf{x}_1+\mathbf{x}_2]_2}A$. Generalizing this, a BLL formula $!_{x<E}A$ containing resource variables $\mathbf{x}_1, \cdots, \mathbf{x}_n$ corresponds to the GBLL formula $!_{[E]_n}A$.

Example 2.2. We look at how we express the substitution of a resource polynomial $A[x := p(x_1, ..., x_n)]$. We define a function $\langle p \rangle_n : \mathbb{N}^n \to \mathbb{N}^{n+1}$ by

$$\langle p \rangle_n(x_1, ..., x_n) \overset{\text{def}}{=} (x_1, ..., x_n, p(x_1, ..., x_n)).$$

Then the reindexed formula $\mathbb{N}^n \vdash A|_{\langle p \rangle_n}$ corresponds to $A[x := p(x_1, \cdots, x_n)]$.

Example 2.3. We illustrate the equality between well-formed formulas. Consider a formula $\mathbb{N} \vdash A$ and a function $r \in \mathbf{Set}(\mathbb{N}^3, \mathbb{N})$. Then we equate formulas $\mathbb{N}^2 \vdash !_{[\mathbf{x}_1+\mathbf{x}_2]_2}(A|_r)$ and $\mathbb{N}^2 \vdash !_h A$, where $h \in \mathbf{Idx}(\mathbb{N}^2, \mathbb{N})$ is given by

$$h \overset{\text{def}}{=} Jr \bullet [\mathbf{x}_1 + \mathbf{x}_2]_2(x, y) = r(x, y, 0), \cdots, r(x, y, x + y - 1).$$

Definition of GBLL Proofs A *judgment* of GBLL is the form $\Delta \mid \Gamma \vdash A$, where Δ is an index set, Γ is a sequence of formulas well-formed under Δ, and A is a well-formed formula under Δ, respectively. The inference rules of GBLL are presented in Fig. 1. Similarly, we define GBAL to be the system obtained by replacing $\mathbf{Idx}$ in Fig. 1 with $\mathbf{Idx}_a$.

Example 2.4. We mimic a special case of the contraction rule in BLL

$$\frac{\Gamma, !_{x<x_i}A, !_{y<x_j}A\{^{x_i+y}/_x\} \vdash B}{\Gamma, !_{x<x_i+x_j}A \vdash B}$$

See also (!C)-rule of CBLL in Section 3.2. We use the *shift function* $s_{n,i} \in \mathbf{Set}(\mathbb{N}^{n+1}, \mathbb{N}^{n+1})$ defined by $s_{n,i}(x_1, \cdots, x_n, y) \overset{\text{def}}{=} (x_1, \cdots, x_n, x_i + y)$. Then we easily see $[\mathbf{x}_i]_n + Js_{n,i} \bullet [\mathbf{x}_j]_n = [\mathbf{x}_i + \mathbf{x}_j]_n$. By contraction rule of GBLL, we obtain the following derivation for well-formed formulas $\mathbb{N}^{n+1} \vdash A$ and $\mathbb{N}^n \vdash B$, mimicking the contraction of BLL:

$$\frac{!_{[\mathbf{x}_i]_n}A, !_{[\mathbf{x}_j]_n}(A|s_{n,i}) \vdash B}{!_{[\mathbf{x}_i+\mathbf{x}_j]_n}A = !_{[\mathbf{x}_i]_n + Js_{n,i}\bullet[\mathbf{x}_j]_n}A \vdash B}$$

Here, we use the formula equality $!_{Js_{n,i}\bullet[\mathbf{x}_j]_n}A = !_{[\mathbf{x}_j]_n}(A|s_{n,j})$.

$$\frac{\Delta \vdash A}{\Delta \mid A \vdash A}\text{(Ax) Axiom} \qquad \frac{\Delta \mid \Gamma, X, Y, \Gamma' \vdash A}{\Delta \mid \Gamma, Y, X, \Gamma' \vdash A}\text{(Exch) Exchange}$$

$$\frac{\Delta \mid \Gamma_1 \vdash A \quad \Delta \mid \Gamma_2, A \vdash B}{\Delta \mid \Gamma_1, \Gamma_2 \vdash B}\text{(Cut)}$$

$$\frac{\Delta \mid \Gamma, X, Y \vdash A}{\Delta \mid \Gamma, X \otimes Y \vdash A}(\otimes\text{L}) \qquad \frac{\Delta \mid \Gamma_1 \vdash X \quad \Delta \mid \Gamma_2 \vdash Y}{\Delta \mid \Gamma_1, \Gamma_2 \vdash X \otimes Y}(\otimes\text{R})$$

$$\frac{\Delta \mid \Gamma_1 \vdash X \quad \Delta \mid \Gamma_2, Y \vdash B}{\Delta \mid \Gamma_1, \Gamma_2, X \multimap Y \vdash B}(\multimap\text{L}) \qquad \frac{\Delta \mid \Gamma, X \vdash Y}{\Delta \mid \Gamma \vdash X \multimap Y}(\multimap\text{R})$$

$$\frac{\Delta \mid \Gamma \vdash B}{\Delta \mid \Gamma, !_0 A \vdash B}\text{(!W) Weakening} \qquad \frac{\Delta \mid \Gamma, A \vdash B}{\Delta \mid \Gamma, !_{\mathrm{id}} A \vdash B}\text{(!D) Dereliction}$$

$$\frac{\Delta \mid \Gamma, !_g A \vdash B \quad \sigma \in \mathbf{Idx}(\Delta, \Delta')(f, g)}{\Delta \mid \Gamma, !_f A \vdash B}\text{(!F) !-Functor}$$

$$\frac{\Delta \mid \Gamma, !_{f_1} A, !_{f_2} A \vdash B}{\Delta \mid \Gamma, !_{f_1 + f_2} A \vdash B}\text{(!C)Contraction}$$

$$\frac{\Delta' \mid !_{g_1} A_1, \cdots, !_{g_k} A_k \vdash B \quad f \in \mathbf{Idx}(\Delta, \Delta')}{\Delta \mid !_{g_1 \bullet f} A_1, \cdots, !_{g_k \bullet f} A_k \vdash !_f B}\text{(P!) Composition}$$

Fig. 1. GBLL Proof Rules

Example 2.5. The reindexing operator can be extended to proofs. Let r be a reindexing function in $\mathbf{Set}(\Delta, \Delta')$. Reindexing of the axiom rule $\Delta' \mid A \vdash A$, by r is the axiom rule $\Delta \mid A|_r \vdash A|_r$. Reindexing of other rules except (P!) can be easily defined—the judgment $\Delta' \mid \Gamma \vdash A$ in each rule is replaced with $\Delta \mid \Gamma|_r \vdash A|_r$ by reindexing. For (P!) rule, reindexing by r is given as follows:

$$\frac{\Delta'' \mid !_{g_1} A_1, \cdots, !_{g_k} A_k \vdash B \qquad f \bullet Jr \in \mathbf{Idx}(\Delta, \Delta'')}{\Delta \mid (!_{g_1 \bullet f} A_1)|_r, \cdots, (!_{g_k \bullet f} A_k)|_r \vdash (!_f B)|_r}$$

Remark 2.1. In this paper, indexing 2-category is either **Idx** or $\mathbf{Idx}_a$. Allowing more general indexing 2-categories in GBLL is a future work. In his PhD thesis, Breuvart designed a linear logic similar to GBLL upon an abstract indexing mechanism called *dependent semirings* [5, Definition 3.2.4.5]. It consists of categories $(\mathcal{S},\mathcal{U})$ such that 1) each hom-set in $\mathcal{S}$ carries a (not necessarily commutative) ordered monoid structure $(0, +)$ and the composition of $\mathcal{S}$ distributes over $0, +$, and 2) $\mathcal{U}$ acts on $\mathcal{S}$ from both sides. Roughly speaking, $\mathcal{S}$ and $\mathcal{U}$ corresponds to our $\mathbf{Idx}^{\mathrm{op}}$ and $\mathbf{Set}^{\mathrm{op}}$, respectively. We expect that a unification of dependent semirings and 2-categories $\mathbf{Idx}, \mathbf{Idx}_a$ would yield a suitable generalization of indexing categories for GBLL. This generalization will subsume the non-graded linear logic, and allow us to compare GBLLs over different idexing categories.

2.3 Complexity of Cut-Elimination in GBLL

By a similar discussion to BLL [16], instances of Cut inference are divided in two classes: *reducible* cuts and *irreducible* cuts. We define the *weight* of proof $|\pi|$ for each proof $\pi \triangleright \Delta \mid \Gamma \vdash A$ and *reduction steps* of proofs, such that every reduction steps will terminate, for each index $\delta \in \Delta$, in polynomial steps of $|\pi|(\delta)$.

Definition 2.2. *[16, Appedix A] In GBLL (resp. GBAL) proofs, an instance of the Cut inference is* irreducible *if there are at least one Composition rule below it or if its left premise is obtained by a Composition rule with nonempty context and the other premise is obtained by a Weakening, !-Functor, Dereliction, Contraction or Composition inference. A* reducible *cut is Cut inferences that is not irreducible.*

The definition of (ir)reducibility and weight is diverted from Girard's paper. Therefore, our system inherits from BLL the conditions under which cuts can be reduced. See also Section 2.4 in [16].

Definition 2.3. *A GBLL or GBAL proof is* irreducible *if it contains only irreducible cut inferences.*

Following [16], we introduce the concept of *weight* of a proof. It is a function $|\pi| : \Delta \to \mathbb{N}$ assigning a weight number $|\pi|(\delta)$ to a proof π at an index $\delta \in \Delta$. The weight number never increases at any reduction step of Cut in π. In the original BLL, weights are expressed by resource polynomials, while here, they are generalized to arbitrary functions. We remark that weights of the proofs involving Composition rules, which introduce $!_f$ modality, use the length of the lists constructed by f.

Definition 2.4. *For a given proof $\pi \triangleright \Delta \mid \Gamma \vdash A$ of GBLL or GBAL, the* weight *of π is a function $|\pi| : \Delta \to \mathbb{N}$ inductively defined as follows. A) When $\Delta = \emptyset$, $|\pi|$ is the evident function. B) When $\Delta \neq \emptyset$, $|\pi|$ is defined by the following rules:*

1. *For an Axiom rule $\pi \triangleright \Delta \mid A \vdash A$, $|\pi|(\delta) \overset{\text{def}}{=} 1$.*
2. *If π is obtained from π' by a unary rule except Contraction and Composition, $|\pi|(\delta) \overset{\text{def}}{=} |\pi'|(\delta) + 1$.*
3. *If π is obtained from π_1 and π_2 by a binary rule except Cut, $|\pi|(\delta) \overset{\text{def}}{=} |\pi_1|(\delta)+ |\pi_2|(\delta) + 1$.*
4. *If π is obtained from π_1 and π_2 by a Cut rule, $|\pi|(\delta) \overset{\text{def}}{=} |\pi_1|(\delta) + |\pi_2|(\delta)$.*
5. *If π is obtained from π' by a Contraction rule, $|\pi|(\delta) \overset{\text{def}}{=} |\pi'|(\delta) + 2$*
6. *If π is obtained from π' by a Composition rule, such as*

$$\pi \triangleright \cfrac{\begin{array}{c}\vdots \pi' \\ \Delta' \mid !_{\alpha_1} A_1, \cdots, !_{\alpha_k} A_k \vdash B\end{array}}{\Delta \mid !_{\alpha_1 \bullet f} A_1, \cdots, !_{\alpha_k \bullet f} A_k \vdash !_f B}$$

then $|\pi|(\delta) \overset{\text{def}}{=} \sum_{\gamma\in f(\delta)} (|\pi'|(\gamma) + 2k + 1) + k + 1$. *Note that the summation* $\sum_{\gamma\in f(\delta)}$ *scans all elements in the list* $f(\delta)$*, hence the weight depends on the length of* $f(\delta)$*.*

Theorem 2.1. *For every proof* $\pi \rhd \Delta \mid \Gamma \vdash A$ *and every* $\delta \in \Delta$*, reduction steps of reducible cuts will terminate in at most* $(|\pi|(\delta))^3$ *steps.*

Proof (sketch). The proof is almost the same as Section 2.2 and Appendix A of [16], except for the definition of the weight. Suppose that π one-step reduces into π'. From the definition of the weight, either 1) for all index $\delta \in \Delta$, the weight decreases (that is, $|\pi|(\delta) > |\pi'|(\delta)$), or 2) for all index $\delta \in \Delta$, the weight keeps (that is, $|\pi|(\delta) = |\pi'|(\delta)$). The reduction of the former type is called symmetric or axiom reduction [16, Section 2.2.1 and 2.2.2], while the latter commutative reduction [16, Section 2.2.3].

In the case where the weight keeps, we introduce another measure called the *cut size* $\|\pi\| : \Delta \to \mathbb{N}$ of a proof π. Its definition is the same as the definition of weight except for Cut rule. For a proof π obtained by Cut rule from π_1 and π_2, the cut size $\|\pi\|(\delta)$ is defined to be $\|\pi_1\|(\delta) + \|\pi_2\|(\delta) + |\pi_1|(\delta) + |\pi_2|(\delta)$.

In each commutative reduction from π to π' the cut size decrease at all index (that is, for all $\delta \in \Delta$, $||\pi||(\delta) > ||\pi'||(\delta)$), and the cut size is at most the square of the weight (that is, for all $\delta \in \Delta, ||\pi||(\delta) \leq (|\pi|(\delta))^2$). Therefore, the total number of steps is at most the cube of the weight. $\square$

The number of reduction steps of a proof π and its weight depend on the length of lists computed by the **Idx**-morphisms occurring in π. However, to discuss the actual time complexity of cut-elimination, we further need to take into account the time complexity of the computation of **Idx**-morphisms. This would be achieved by looking at a subcategory of **Idx** computable within a certain time complexity. We leave this argument of analyzing the actual time complexity of cut-elimination as a future work.

3 Translation from Constrained BLL

We show that GBLL can express BLL via a translation. This translation is actually given to variants of these calculi, namely from *BLL with constraints* (called CBLL) to *GBAL with positivity axioms* (called GBAL$^+$).

CBLL is an extension of BLL with *constraints*, which are one of the features of Dal Lago and Hofmann's QBAL [10]. Constraints explicitly specify conditions imposed on resource variables, and it is natural to explicitly maintain these conditions throughout proofs. We also remark that in CBLL, weakening of !-formulas $!_{x<p+q}A \multimap !_{x<p}A$ is allowed, and atomic formulas are assumed to satisfy the positivity property (3.1).

GBAL$^+$ is designed for a sound translation from CBLL. Recall that GBAL is an extension of GBLL with weakening $!_{f+g}A \multimap !_f A$ on !-formulas. Then GBAL$^+$ is a further extension of GBAL with the following positivity axioms of atomic formulas: for every n-ary atomic formula $a \in \mathcal{A}$ in CBLL, we introduce an atomic formula $[a] \in \mathcal{A}(\mathbb{N}^n)$ to GBAL together with the axiom:

$$V_{\mathscr{C}}(F) \mid \emptyset \vdash [a] \star \langle p_1, \cdots, p_n \rangle \multimap [a] \star \langle q_1, \cdots, q_n \rangle \quad (\forall i. p_i \sqsubseteq_{\mathscr{C}} q_i).$$

Here the definition of each notation is given in Section 3.1 and 3.3. Positivity axiom induces proofs $V_{\mathscr{C}}(F) \mid A' \vdash A$ for every two formulas A, A' such that $A' \sqsubseteq_{\mathscr{C}} A$ (the relation $\sqsubseteq_{\mathscr{C}}$ for formulas is defined in Section 3.2).

3.1 Resource Polynomials and Constraints

We introduce basic concepts around CBLL, referring to its super-logic QBAL [10]. We put a reference in the beginning of each paragraph when the contents come from QBAL in [10].

[10, Definition 2.1] Given a countably infinite set $\mathcal{RV}$ of *resource variables*, a *resource monomial* over $\mathcal{RV}$ is a finite product of binomial coefficients $\prod_{i=1}^{m} \binom{x_i}{n_i}$, where the resource variables $x_1, \cdots, x_m$ are distinct and $n_1, \cdots, n_m \in \mathbb{N}$ are natural numbers. A *resource polynomial* over $\mathcal{RV}$ is a finite sum of resource monomials. We write 1 as $\binom{x}{0}$ and x as $\binom{x}{1}$ for short. Each positive natural number n denotes a resource polynomial $1 + 1 + \cdots + 1$. Resource polynomials are closed under sum, product, bounded sum and composition [10, Lemma 2.2].

[10, Definition 2.3] A *constraint* is an inequality $p \leq q$, where p and q are resource polynomials. We abbreviate $p+1 \leq q$ as $p < q$. A constraint $p \leq q$ *holds* (written $\vDash p \leq q$) if it is true in the standard model. A *constraint set* (denoted with $\mathscr{C}$, $\mathscr{D}$) is a finite set of constraints. A constraint $p \leq q$ *is a consequence* of a constraint set $\mathscr{C}$ (written $\mathscr{C} \vDash p \leq q$) if $p \leq q$ is a logical consequence of $\mathscr{C}$. For every constraint sets $\mathscr{C}$ and $\mathscr{D}$, we write $\mathscr{C} \vDash \mathscr{D}$ iff $\mathscr{C} \vDash p \leq q$ for every constraint $p \leq q$ in $\mathscr{D}$. For each constraint set $\mathscr{C}$, we define an order $\sqsubseteq_{\mathscr{C}}$ on resource polynomials by $p \sqsubseteq_{\mathscr{C}} q$ iff $\mathscr{C} \vDash p \leq q$.

[10, Definition 2.3] We define the polarity of occurrences of free resource variables. For a constraint $p \leq q$, we say that an occurrence of a resource variable x in p is called *negative*, while the one in q is called *positive*.

3.2 Formulas and Inference Rules of CBLL

Let $\mathcal{A}$ be a set of atomic formulas and assume that each atomic formula $a \in \mathcal{A}$ is associated with an arity $\mathrm{ar}(a)$. Formulas of CBLL are defined by:

$$A, B ::= \quad a(p_1, \cdots, p_{\mathrm{ar}(a)}) \mid A \otimes B \mid A \multimap B \mid !_{x<p} A$$

where p in the formula $!_{x<p}A$ satisifes $x \notin \mathrm{FV}(p)$.

[10, Definition 2.6] Each occurrence of a free resource variable in a formula is classified into *positive* or *negative*. Below we inductively define a *positive occurrence* of a resource variable. An occurrence of x in:

- $a(p_1, \cdots, p_{\mathrm{ar}(a)})$ is always positive.
- $A \otimes B$ is positive iff it is in A and positive, or so in B.

$$\frac{A \sqsubseteq_{\mathscr{C}} B}{A \vdash_{\mathscr{C}} B}\ (\mathrm{Ax}) \qquad \frac{\Gamma \vdash_{\mathscr{C}} A \quad \mathscr{D} \models \mathscr{C}}{\Gamma \vdash_{\mathscr{D}} A}\ (\mathrm{Str}) \qquad \frac{\Gamma \vdash_{\mathscr{C}} B}{\Gamma, !_{x<0} A \vdash_{\mathscr{C}} B}\ (!\mathrm{W})$$

$$\frac{A\{0/x\}, \Gamma \vdash_{\mathscr{C}} B}{!_{x<1} A, \Gamma \vdash_{\mathscr{C}} B}\ (!\mathrm{D}) \qquad \frac{\Gamma, !_{x<p} A, !_{y<q} A\{p+y/x\} \vdash_{\mathscr{C}} B}{\Gamma, !_{x<p+q} A \vdash_{\mathscr{C}} B}\ (!\mathrm{C})$$

$$\frac{A_1, \cdots, A_n \vdash_{\mathscr{C} \cup \{x<p\}} B \quad x \notin \mathrm{FV}(\mathscr{C})}{!_{x<p} A_1, \cdots, !_{x<p} A_n \vdash_{\mathscr{C}} !_{x<p} B}\ (!\mathrm{P})$$

$$\frac{!_{y<p} !_{z<q\{y/w\}} A \left\{ \left(z + \sum_{w<y} q(w) \right) / x \right\}, \Gamma \vdash_{\mathscr{C}} B}{!_{x<\sum_{w<p} q(w)} A, \Gamma \vdash_{\mathscr{C}} B}\ (!\mathrm{N})$$

Fig. 2. Inference Rules for CBLL ($\otimes$ and $\multimap$ are omitted)

– $A \multimap B$ is positive iff it is in A and negative, or it is in B and positive.
– $!_{x'<p} A$ is positive iff it is in A and positive. We remark that an occurrence of a free resource variable in p is counted as negative in $!_{x'<p} A$.

[10, Definition 2.8] We extend the order $\sqsubseteq_{\mathscr{C}}$ on resource polynomials to the one on CBLL formulas.

$$\begin{aligned}
&a(p_1, \cdots, p_{\mathrm{ar}(a)}) \sqsubseteq_{\mathscr{C}} a(q_1, \cdots, q_{\mathrm{ar}(a)}) \text{ iff } \forall i. p_i \sqsubseteq_{\mathscr{C}} q_i \\
&A \otimes B \sqsubseteq_{\mathscr{C}} C \otimes D \text{ iff } (A \sqsubseteq_{\mathscr{C}} C) \wedge (B \sqsubseteq_{\mathscr{C}} D) \\
&A \multimap B \sqsubseteq_{\mathscr{C}} C \multimap D \text{ iff } (C \sqsubseteq_{\mathscr{C}} A) \wedge (B \sqsubseteq_{\mathscr{C}} D) \\
&!_{x<p} A \sqsubseteq_{\mathscr{C}} !_{x<q} B \text{ iff } (q \sqsubseteq_{\mathscr{C}} p) \wedge (x \notin \mathrm{FV}(\mathscr{C})) \wedge (A \sqsubseteq_{\mathscr{C} \cup \{x<q\}} B)
\end{aligned} \tag{3.1}$$

[10, Section 2.3] A CBLL *judgment* is an expression $\Gamma \vdash_{\mathscr{C}} A$, where $\mathscr{C}$ is a constraint set, Γ is a multiset of formulas and A is a formula. A judgment $\Gamma \vdash_{\mathscr{C}} A$ means that A is a consequence of Γ under the constraints $\mathscr{C}$.

Inference rules (Fig. 2) are almost the same as those of QBAL; we omit the rules for $\otimes, \multimap$ and Cut. Note that weakening is restricted to !-formulas. Every BLL proof of $\Gamma \vdash A$ can be translated to a CBLL proof of $\Gamma \vdash_{\emptyset} A$.

3.3 Translation into GBAL$^+$

As mentioned at the beginning of Section 3, we will give a translation from CBLL to GBAL$^+$. When translating a CBLL proof $\Gamma \vdash_{\mathscr{C}} A$, we also need to supply a set F of free resource variables satisfying $F \supseteq \mathrm{FV}(\Gamma) \cup \mathrm{FV}(A) \cup \mathrm{FV}(\mathscr{C})$. Then the translation of the proof of $\Gamma \vdash_{\mathscr{C}} A$ yields a proof of $V_{\mathscr{C}}(F) \mid [\Gamma]^{(F;\mathscr{C})} \vdash [A]^{(F;\mathscr{C})}$ in GBAL$^+$.

For Constraints We define an *environment* over a finite set F of resource variables to be a function from F to $\mathbb{N}$; by $V(F)$ we mean the set of environments over F. Given an environment $\rho \in V(F)$ and a resource variable $x \notin F$ and $n \in \mathbb{N}$, by $\rho\{x \mapsto n\}$ we mean the environment over $F \cup \{x\}$ that extends ρ

with a mapping $x \mapsto n$. Given a resource polynomial p such that $FV(p) \subseteq F$, by $[\![p]\!] : V(F) \to \mathbb{N}$ we mean the function that evaluates the resource polynomial p under a given environment. For resource polynomials $p_1, \cdots, p_n$ such that $\mathrm{FV}(p_i) \subseteq F$, we give a function $\langle p_1, \cdots, p_n \rangle : (V(F)) \to \mathbb{N}^n$ by $\langle p_1, \cdots, p_n \rangle \rho = ([\![p_1]\!]\rho, \cdots, [\![p_n]\!]\rho)$.

Let $\rho \vDash p \leq q$ denote $[\![p]\!]\rho \leq [\![q]\!]\rho$ for a constraint $p \leq q$ with a set F of free resource variables (such that $\mathrm{FV}(p) \cup \mathrm{FV}(q) \subseteq F$) and for an environment $\rho \in V(F)$. For a subset $S \subset V(F)$ and for a constraint set $\mathscr{C}$, $S \vDash \mathscr{C}$ is also defined similarly: for every $\rho \in S$ and for every $p \leq q \in \mathscr{C}$, $\rho \vDash p \leq q$. Given a constraint set $\mathscr{C}$ and a set F of resource variables such that $\mathrm{FV}(\mathscr{C}) \subseteq F$, let a set $V_{\mathscr{C}}(F)$ and a function $\iota_{F,\mathscr{C}} : V_{\mathscr{C}}(F) \to V(F)$ be given by:

$$V_{\mathscr{C}}(F) \stackrel{\mathrm{def}}{=} \{\rho \in V(F) \mid \rho \vDash \mathscr{C}\}, \qquad \iota_{F,\mathscr{C}}(\rho) \stackrel{\mathrm{def}}{=} \rho.$$

For a resource polynomial p, a free resource variable x such that $x \notin \mathrm{FV}(p)$, a constraint set $\mathscr{C}$ and a set F of resource variables such that $\mathrm{FV}(p) \cup \mathrm{FV}(\mathscr{C}) \subseteq F$, we introduce a map $[x < p]_{(F,\mathscr{C})} : V_{\mathscr{C}}(F) \to V_{\mathscr{C} \cup \{x<p\}}(F \cup \{x\})^*$ by

$$[x < p]_{(F,\mathscr{C})}\rho \stackrel{\mathrm{def}}{=} \rho\{x \mapsto 0\}, \rho\{x \mapsto 1\}, \cdots, \rho\{x \mapsto ([\![p]\!]\rho - 1)\}$$

For Formulas Given a CBLL formula A, a constraint set $\mathscr{C}$ and a set of resource variables F such that $F \supseteq \mathrm{FV}(A) \cup \mathrm{FV}(\mathscr{C})$, the translation $[A]^{(F;\mathscr{C})}$ of a well-formed formula $V_{\mathscr{C}}(F) \vdash A$ is defined inductively as follows:

$$[a(p_1, ..., p_n)]^{(F;\mathscr{C})} \stackrel{\mathrm{def}}{=} [a] \star (\langle p_1, ..., p_n \rangle \circ \iota_{F,\mathscr{C}})$$

$$[A \otimes B]^{(F;\mathscr{C})} \stackrel{\mathrm{def}}{=} [A]^{(F;\mathscr{C})} \otimes [B]^{(F;\mathscr{C})}$$

$$[A \multimap B]^{(F;\mathscr{C})} \stackrel{\mathrm{def}}{=} [A]^{(F;\mathscr{C})} \multimap [B]^{(F;\mathscr{C})}$$

$$[!_{x<p}A]^{(F;\mathscr{C})} \stackrel{\mathrm{def}}{=} !_{[x<p]_{(F,\mathscr{C})}}[A]^{(F\cup\{x\};\mathscr{C}\cup\{x<p\})}$$

For Proofs To give a translation of proofs, we define another notation. For a resource polynomial p, q, a set F of resource variables and a constraint set $\mathscr{C}$ such that $\mathrm{FV}(p) \cup \mathrm{FV}(\mathscr{C}) \subseteq F$, a set $[p, q)^{(F,\mathscr{C})}$ of environments is defined by

$$[p, q)^{(F,\mathscr{C})} = \{\rho \in V(F \cup \{t\}) \mid \rho \vDash \mathscr{C}, [\![p]\!](\rho) \leq \rho(t) < [\![p + q]\!]\rho\}$$

here t is a "fresh" resource variable such that $t \notin F$.

Given a proof $\pi \triangleright \Gamma \vdash_{\mathscr{C}} A$, a translation $[\pi]^{(F;\mathscr{C})} \triangleright V_{\mathscr{C}}(F) \mid [\Gamma]^{(F;\mathscr{C})} \vdash [A]^{(F;\mathscr{C})}$ is defined inductively on the structure of the proof:

– For Axiom rule, we can prove $V_{\mathscr{C}}(F) \mid [A]^{(F;\mathscr{C})} \vdash [B]^{(F;\mathscr{C})}$ for formulas A, B such that $A \sqsubseteq_{\mathscr{C}} B$.
– For rules (Cut), ($\otimes$L), ($\otimes$R), ($\multimap$L), ($\multimap$R) and (!W), the translation is simple replacement of each formula A with $[A]^{(F;\mathscr{C})}$.

- For (Str) rule, we have a map $r \in \mathbf{Set}(V_{\mathscr{D}}(F), V_{\mathscr{C}}(F))$. Then the translation is given as reindexed proof $[\pi']^{(F;\mathscr{C})}|_r$ of the translation $[\pi']^{(F;\mathscr{C})}$ of the premise.
- For (!D) rule, the premise is translated to $V_{\mathscr{C}}(F) \mid A', [\Gamma]^{(F;\mathscr{C})} \vdash [B]^{(F;\mathscr{C})}$, where $A' = [A]^{(F\cup\{x\};\mathscr{C}\cup\{x<1\})}|_r$ and r is a map such that $Jr = [x < 1]_{(F,\mathscr{C})}$.
- For (!C) rule, we define a morphism $s_{p,q}^{(F;\mathscr{C})}$ in $\mathbf{Idx}_a$ and functions $r_{p,q}^{(F;\mathscr{C})}$, $i_1^{(p,q;F;\mathscr{C})}$, $i_2^{(p,q;F;\mathscr{C})}$ (s, r, i_1 and i_2 for short) by

$$\begin{aligned}
s_{p,q}^{(F;\mathscr{C})}: \quad & V_{\mathscr{C}}(F) \to [p,q)^{(F,\mathscr{C})} \\
& \rho \mapsto \rho\{t \mapsto [\![p]\!]\rho\}, \cdots, \rho\{t \mapsto ([\![p+q]\!]\rho - 1)\} \\
r_{p,q}^{(F;\mathscr{C})}: \quad & [p,q)^{(F,\mathscr{C})} \xrightarrow{\sim} V_{\mathscr{C}\cup\{y<q\}}(F\cup\{y\}) \\
& \rho\{t \mapsto ([\![p]\!]\rho + k)\} \mapsto \rho\{y \mapsto k\} \\
i_1^{(p,q;F;\mathscr{C})}: \quad & V_{\mathscr{C}\cup\{x<p\}}(F\cup\{x\}) \to V_{\mathscr{C}\cup\{x<p+q\}}(F\cup\{x\}) \\
& \rho\{x \mapsto k\} \mapsto \rho\{x \mapsto k\} \\
i_2^{(p,q;F;\mathscr{C})}: \quad & [p,q)^{(F,\mathscr{C})} \to V_{\mathscr{C}\cup\{x<p+q\}}(F\cup\{x\}) \\
& \rho\{t \mapsto ([\![p]\!]\rho + k)\} \mapsto \rho\{x \mapsto [\![p]\!]\rho + k\}
\end{aligned}$$

They satisfy $!_{[x<p]}[A]^{(F\cup\{x\};\mathscr{C}\cup\{x<p\})} =!_{[x<p]}([A]^{(F\cup\{x\};\mathscr{C}\cup\{x<p+q\})}|_{i_1})$ and $!_{[y<q]}[A\{{}^{p+y}/_x\}]^{(F\cup\{y\};\mathscr{C}\cup\{y<q\})} =!_{Jr\bullet s}([A]^{(F\cup\{x\};\mathscr{C}\cup\{x<p+q\})}|_{i_2\circ r^{-1}})$.
Then the conclusion of (!C) is obtained:

$$V_{\mathscr{C}}(F) \mid [\Gamma]^{(F;\mathscr{C})}, !_{(Ji_i\bullet[x<p])+(Ji_2\bullet s)}[A]^{(F\cup\{x\};\mathscr{C}\cup\{x<p+q\})} \vdash [B]^{(F;\mathscr{C})}.$$

- For (!P) rule, let $F' = F \cup \{x\}$ and $\mathscr{C}' = \mathscr{C} \cup \{x < p\}$. We can prove the translated conclusion from the translated premise by the following proof:

$$\dfrac{\dfrac{\dfrac{V_{\mathscr{C}'}(F') \mid [A_1]^{(F';\mathscr{C}')}, \cdots, [A_n]^{(F';\mathscr{C}')} \vdash [B]^{(F';\mathscr{C}')}}{n \text{ times (!D)'s } \vdots}}{V_{\mathscr{C}'}(F') \mid !_{\mathrm{id}}[A_1]^{(F';\mathscr{C}')}, \cdots, !_{\mathrm{id}}[A_n]^{(F';\mathscr{C}')} \vdash [B]^{(F';\mathscr{C}')}}}{V_{\mathscr{C}}(F) \mid !_{[x<p]}[A_1]^{(F';\mathscr{C}')} \cdots !_{[x<p]}[A_n]^{(F';\mathscr{C}')} \vdash !_{[x<p]}[B]^{(F';\mathscr{C}')}}$$

- For (!N) rule, we define index sets $\Delta_0, \Delta_1, \Delta_2$ and constraints $\mathscr{C}_0, \mathscr{C}_1, \mathscr{C}_2$ by

$$\begin{aligned}
\mathscr{C}_0 &= \mathscr{C} \cup \{y < p\} & \Delta_0 &= V_{\mathscr{C}_0}(F \cup \{y\}) \\
\mathscr{C}_1 &= \mathscr{C} \cup \{y < p, z < q\{{}^y/_w\}\} & \Delta_1 &= V_{\mathscr{C}_1}(F \cup \{y, z\}) \\
\mathscr{C}_2 &= \mathscr{C} \cup \{x < \sum_{w<p} q(w)\} & \Delta_2 &= V_{\mathscr{C}_2}(F \cup \{x\})
\end{aligned}$$

There is an isomorphism $r \in \mathbf{Set}(\Delta_1, \Delta_2)$, and it holds an equation $[z < q\{{}^y/_w\}]_{(F\cup\{y\},\mathscr{C}_0)} \bullet [y < p]_{(F,\mathscr{C})} = Jr^{-1} \bullet [x < \sum_{w<p} q(w)]_{(F,\mathscr{C})}$. Therefore, (!N) rule can be translated to the following provable judgment:

$$V_{\mathscr{C}}(F) | !_{[x<\sum_{w<p} q]}[A]^{(F\cup\{x\};\mathscr{C}_2)} \vdash !_{[y<p]}!_{[z<q\{{}^y/_w\}]}[A\{{}^{z+\sum_{w<y} q}/_x\}]^{(F\cup\{y,z\};\mathscr{C}_1)}$$

Since every BLL proof $\Gamma \vdash A$ can be translated to a CBLL proof $\Gamma \vdash_{\emptyset} A$, it can further be translated to a GBAL$^+$ proof $V_{\emptyset}(F) \mid [\Gamma]^{(F;\emptyset)} \vdash [A]^{(F;\emptyset)}$.

4 Categorical Semantics for GBLL

We give a categorical semantics of GBLL. First, notice that each index set Δ determines a multiplicative linear logic under Δ. We model this situation by a *set-indexed symmetric monoidal closed categories*, given by a functor $C : \mathbf{Set}^{op} \to \mathbf{SMCC}_{\text{strict}}$. That is, for each $\Delta \in \mathbf{Set}$, a symmetric monoidal closed category $C\Delta$ is given, and any function $f : \Delta \to \Delta'$ induces a strict symmetric monoidal closed functor $Cf : C\Delta' \to C\Delta$, performing renaming of indexes.

Upon this indexed symmetric monoidal closed categories, we introduce a categorical structure that models the $!_f$ modality. We call it *indexed linear exponential comonad*. This is a generalization of the *semiring-graded linear exponential comonad* studied in [13,22]. Our generalization replaces the semiring with $\mathbf{Idx}$, which may be regarded as a many-object pseudo-semiring (Proposition 2.1).

We write $[\mathbb{C}, \mathbb{D}]_l$ for the category of symmetric lax monoidal functors from $\mathbb{C}$ to $\mathbb{D}$ and monoidal natural transformations between them. We equip it with the pointwise symmetric monoidal structure $(\dot{I}, \dot{\otimes})$ given by $\dot{I}X = I$ and $(F \dot{\otimes} G)X = FX \otimes GX$ for $X \in \mathbb{C}$.

Definition 4.1. *An* indexed linear exponential comonad *(ILEC for short) over a set-indexed SMCC C consists of:*

- *A collection of symmetric colax monoidal functors*

$$(D, w^{\Delta,\Delta'}, c^{\Delta,\Delta'}) : \mathbf{Idx}(\Delta, \Delta') \to [C\Delta', C\Delta]_l \quad (\Delta, \Delta' \in \mathbf{Set}).$$

  The symmetric lax monoidal structure of Df is denoted by $m_f : I \to DfI$ and $m_{f,A,B} : DfA \otimes DfB \to Df(A \otimes B)$.
- *Monoidal natural transformations $\epsilon^\Delta : D(\mathrm{i}_\Delta) \to \mathrm{Id}_{D\Delta}$ and $\delta_{g,f} : D(g \bullet f) \to Df \circ Dg$ satisfying axioms in Figure 3.*
- *$Cr' \circ Df \circ Cr = D(Jr \bullet f \bullet Jr')$ holds for any morphism f in $\mathbf{Idx}$ and r, r' in $\mathbf{Set}$ of appropriate type.*

The last axiom has two purposes: the equality $Cr'(DfA) = D(f \bullet Jr')A$ is to allow reindexing functions to act from outside, and the other equality $Df(CrA) = D(Cr \bullet f)A$ is to make D invariant under internal reindexing of formulas. These equalities are tied up with the formula equivalence in (2.1) and the definition of reindexing at $!_f A$ in Definition 2.1, respectively. We postpone a concrete example of ILEC to Section 4.2.

4.1 Semantics of GBLL

We interpret a well-formed formula $\Delta \vdash A$ as an object $[\![\Delta \vdash A]\!] \in C\Delta$. This is done by induction on the structure of the formula. We assume that each atomic formula $a \in \mathcal{A}(\Delta)$ comes with its interpretation as an object $[a] \in C\Delta$.

$$[\![\Delta \vdash a \star r]\!] \overset{\text{def}}{=} Cr[a] \qquad [\![\Delta \vdash !_f A]\!] \overset{\text{def}}{=} Df[\![\Delta' \vdash A]\!]$$

$$[\![\Delta \vdash A \otimes B]\!] \overset{\text{def}}{=} [\![\Delta \vdash A]\!] \otimes [\![\Delta \vdash B]\!] \quad [\![\Delta \vdash A \multimap B]\!] \overset{\text{def}}{=} [\![\Delta \vdash A]\!] \multimap [\![\Delta \vdash B]\!]$$

$$\begin{tikzcd}
D(f \bullet h + g \bullet h)A \arrow[r, equal] \arrow[d] & D((f+g) \bullet h)A \arrow[d] \\
D(f \bullet h)A \otimes D(g \bullet h)A \arrow[d] & Dh(D(f+g)A) \arrow[d] \\
Dh(DfA) \otimes Dh(DgA) \arrow[r] & Dh(DfA \otimes DgA)
\end{tikzcd}
\qquad
\begin{tikzcd}
D0A \arrow[r, equal] \arrow[dd] & D(0 \bullet h)A \arrow[d] \\
 & Dh(D0A) \arrow[d] \\
I \arrow[r] & DhI
\end{tikzcd}$$

$$\begin{tikzcd}
D(h \bullet f + h \bullet g)A \arrow[r, "D\simeq A"] \arrow[d] & D(h \bullet (f+g))A \arrow[d] \\
D(h \bullet f)A \otimes D(h \bullet f)A \arrow[d] & D(f+g)(DhA) \arrow[d] \\
Df(DhA) \otimes Dg(DhA) \arrow[r, equal] & (Df \otimes Dg)(DhA)
\end{tikzcd}
\qquad
\begin{tikzcd}
D0A \arrow[r, equal] \arrow[dd] & D(h \bullet 0)A \arrow[d] \\
 & D0(DhA) \arrow[d] \\
I \arrow[r, equal] & I
\end{tikzcd}$$

$$\begin{tikzcd}
DfA \arrow[r] \arrow[d] \arrow[dr, equal] & D(\mathrm{i}_\varDelta)(DfA) \arrow[d] \\
Df(D(\mathrm{i}_\varDelta)A) \arrow[r] & DfA
\end{tikzcd}
\qquad
\begin{tikzcd}
D(h \bullet g \bullet f)A \arrow[r] \arrow[d] & D(g \bullet f)(DhA) \arrow[d] \\
Df(D(h \bullet g)A) \arrow[r] & Df(Dg(DhA))
\end{tikzcd}$$

Fig. 3. Axioms of Indexed Linear Exponential Comonad

Proposition 4.1. *For any $r \in \mathbf{Set}(\varDelta, \varDelta')$ and well-formed formula $\varDelta' \vdash A$, we have $[\![\varDelta \vdash A|_r]\!] = Cr[\![\varDelta' \vdash A]\!]$.*

Proposition 4.2. $[\![\varDelta \vdash !_{Jr \bullet f} A]\!] = [\![\varDelta' \vdash !_f (A|_r)]\!]$.

Each proof $\pi \rhd \varDelta \mid \varGamma \vdash A$ of GBLL is interpreted as a morphism $[\![\varDelta \mid \varGamma \vdash A]\!] : [\![\varDelta \vdash \varGamma]\!] \to [\![\varDelta \vdash A]\!]$ in $C\varDelta$. Here, for a sequence $\varGamma = C_1, \cdots, C_m$ of formulas, $[\![\varDelta \vdash \varGamma]\!]$ denotes $[\![\varDelta \vdash C_1]\!] \otimes \cdots \otimes [\![\varDelta \vdash C_m]\!]$. We write out the interpretation only for the cases of modalities, because the other rules, Axiom, Exchange, Cut, $\otimes$(L, R) and $\multimap$(L, R) are interpreted similarly to the semantics of multiplicative intuitionistic linear logic. Fig. 4 shows the interpretation of rules related to $!_f$.

Theorem 4.1. *For a proof $\pi \rhd \varDelta \mid \varGamma \vdash A$, if π has a reducible cut and reduces into π' by a reduction step, then $[\![\pi]\!] = [\![\pi']\!]$ in $C\varDelta$.*

4.2 Construction of an Indexed Linear Exponential Comonad

We present a construction of an indexed SMCCs $C : \mathbf{Set}^{op} \to \mathbf{SMCC}_{\mathrm{strict}}$ and an ILEC $D : \mathbf{Idx}(\varDelta, \varDelta') \to [C\varDelta', C\varDelta]_l$ over C from a SMCC $\langle \mathbb{C}, \otimes, I, \multimap \rangle$, and a symmetric lax monoidal comonad $\langle V, m^V, m^V_{X,Y}, \epsilon, \delta \rangle$ on $\mathbb{C}$.

Construction of Indexed SMCCs First, for each index set $\varDelta$, we define the category $\varDelta \pitchfork \mathbb{C}$ to be the product of $\varDelta$-many copies of $\mathbb{C}$. We represent objects and morphisms of this category by maps $\mathsf{X} : \varDelta \to \mathrm{Obj}(\mathbb{C})$ and maps

$$\left\llbracket \frac{\pi' \triangleright \Delta \mid \Gamma \vdash B}{\Delta \mid \Gamma, !_{0_{\Delta,\Delta'}} A \vdash B} \right\rrbracket = \llbracket \Gamma, !_{0_{\Delta,\Delta'}} A \rrbracket \xrightarrow{\mathrm{id} \otimes w^{\Delta,\Delta'}_{\llbracket A \rrbracket}} \llbracket \Gamma \rrbracket \otimes I \xrightarrow{\llbracket \pi' \rrbracket \circ \cong} \llbracket B \rrbracket$$

$$\left\llbracket \frac{\pi' \triangleright \Delta \mid \Gamma, A \vdash B}{\Delta \mid \Gamma, !_{\mathrm{id}} A \vdash B} \right\rrbracket = \llbracket \Gamma, !_{\mathrm{i}_\Delta} A \rrbracket \xrightarrow{\mathrm{id} \otimes \epsilon^{\Delta}_{\llbracket A \rrbracket}} \llbracket \Gamma \rrbracket \otimes \llbracket A \rrbracket \xrightarrow{\llbracket \pi' \rrbracket} \llbracket B \rrbracket$$

$$\left\llbracket \frac{\pi' \triangleright \Delta \mid \Gamma, !_g A \vdash B \quad \sigma : f \Rightarrow g}{\Delta \mid \Gamma, !_f A \vdash B} \right\rrbracket = \llbracket \Gamma, !_f A \rrbracket \xrightarrow{\mathrm{id} \otimes (D\sigma)_{\llbracket A \rrbracket}} \llbracket \Gamma, !_g A \rrbracket \xrightarrow{\llbracket \pi' \rrbracket} \llbracket B \rrbracket$$

$$\left\llbracket \frac{\pi' \triangleright \Delta \mid \Gamma, !_f A, !_g A \vdash B}{\Delta \mid \Gamma, !_{f+g} A \vdash B} \right\rrbracket = \llbracket \Gamma, !_{f+g} A \rrbracket \xrightarrow{\mathrm{id} \otimes c_{f,g}} \llbracket \Gamma \rrbracket \otimes (\llbracket !_f A \rrbracket \otimes \llbracket !_g A \rrbracket) \xrightarrow{\llbracket \pi' \rrbracket \circ \cong} \llbracket B \rrbracket$$

$$\left\llbracket \frac{\pi' \triangleright \Delta, g \mid !_{g_1} A_1, \cdots, !_{g_k} A_k \vdash B}{\Delta \mid !_{g_1 \bullet f} A_1, \cdots, !_{g_k \bullet f} A_k \vdash !_f B} \right\rrbracket = \begin{aligned} & \textstyle\bigotimes_i \llbracket !_{g_i \bullet f} A_i \rrbracket \xrightarrow{\bigotimes_i \delta_{g_i, f, \llbracket A_i \rrbracket}} \textstyle\bigotimes_i Df(\llbracket !_{g_i} A_i \rrbracket) \\ & \xrightarrow{m_{f, \cdots \llbracket !_{g_i} A_i \rrbracket \cdots}} Df\left(\textstyle\bigotimes_i \llbracket !_{g_i} A_i \rrbracket\right) \\ & \xrightarrow{Df(\llbracket \pi' \rrbracket)} \llbracket !_f B \rrbracket \end{aligned}$$

Here, 1) $\llbracket A \rrbracket$ denotes $\llbracket \Delta \vdash A \rrbracket$ for each well-formed formula $\Delta \vdash A$. 2) π' denotes the proof of the premise of each rule.

Fig. 4. Interpretations of Modal Rules.

$\mathsf{f} : \Delta \to \mathrm{Mor}(\mathbb{C})$, respectively. Since SMCCs are closed under products, $\Delta \pitchfork \mathbb{C}$ is a SMCC by the component-wise tensor product and internal hom:

$$\mathsf{I}(d) \overset{\mathrm{def}}{=} I, \quad \mathsf{X} \dot{\otimes} \mathsf{Y}(d) \overset{\mathrm{def}}{=} \mathsf{X}(d) \otimes \mathsf{Y}(d), \quad \mathsf{X} \dot{\multimap} \mathsf{Y}(d) \overset{\mathrm{def}}{=} \mathsf{X}(d) \multimap \mathsf{Y}(d)$$

We then define the indexed SMCCs C by $C\Delta \overset{\mathrm{def}}{=} \Delta \pitchfork \mathbb{C}$.

Folding Product We next introduce the *folding product* functor T; we later compose it with the symmetric lax monoidal comonad V so that we can derive various ILECs over C. Note that T itself *is* also an ILEC; set $V = \mathrm{Id}$. The type of T is $\Delta^* \times (\Delta \pitchfork \mathbb{C}) \longrightarrow \mathbb{C}$, and is defined by

$$\mathsf{T}(i_1 i_2 \cdots i_n, \mathsf{A}) \overset{\mathrm{def}}{=} \mathsf{A}(i_1) \otimes \mathsf{A}(i_2) \otimes \cdots \otimes \mathsf{A}(i_n), \quad \mathsf{T}((), \mathsf{A}) \overset{\mathrm{def}}{=} I$$

On morphisms, T maps a list permutation in the first argument to the symmetry morphism in $\mathbb{C}$. T is symmetric strong monoidal in each argument. Moreover, each strong monoidal structure interacts well with each other, concluding that it becomes a multi-symmetric strong monoidal functor in the sense of [21].

Proposition 4.3. *For $f \in \mathbf{Idx}(\Delta, \Delta')$ and $l = i_1 \cdots i_k \in \Delta^*$, let $f(l)$ denote $f(i_1) \cdots f(i_k)$. Then it holds $\mathsf{T}(f(l), \mathsf{A}) \simeq \mathsf{T}(l, \mathsf{T}(f(\_), \mathsf{A}))$ and this isomorphism is natural for A.*

Remark 4.1. Usually the !-modal formula $!A$ in linear logic is interpreted by the object consisting of many copies of the same data (referred as *uniformity* of $!A$ [8]). We leave the development of uniform folding product as a future work.

Construction of ILEC We now compose the folding product functor with the symmetric lax monoidal comonad V, to derive another ILEC. Let Δ, Δ' be index sets. We define a symmetric strong (hence colax) monoidal functor $D : \mathbf{Idx}(\Delta, \Delta') \longrightarrow [C\Delta', C\Delta]_l$ by

$$Df\mathsf{A}(i) \overset{\text{def}}{=} \mathsf{T}(f(i), V \circ \mathsf{A}) \quad Df\mathsf{p}(i) \overset{\text{def}}{=} \mathsf{T}(f(i), V\mathsf{p}) \quad D\alpha\mathsf{A} \overset{\text{def}}{=} \mathsf{T}(\alpha, V \circ \mathsf{A}). \tag{4.1}$$

Here, $\mathsf{A} \in \Delta' \pitchfork \mathbb{C}$, and p and α are morphisms in $\Delta' \pitchfork \mathbb{C}$ and $\mathbf{Idx}(\Delta, \Delta')$, respectively. We also define a helper morphism $\gamma^l_{\mathsf{A}} : \mathsf{T}(l, V \circ \mathsf{A}) \to V\mathsf{T}(l, \mathsf{A})$ for $(l_1 \cdots l_k) \in \Delta^*$ and $\mathsf{A} \in \Delta \pitchfork \mathbb{C}$. It is the multiple composite of $m_{A,B}$:

$$V\mathsf{A}(l_1) \otimes \cdots \otimes V\mathsf{A}(l_k) \to V\left(\mathsf{A}(l_1) \otimes \cdots \otimes \mathsf{A}(l_k)\right).$$

It is routine to verify that this morphism is monoidal natural on l and A.

Two monoidal natural transformations $\epsilon : D\mathrm{i}_\Delta \to \mathrm{Id}_{\Delta \pitchfork \mathbb{C}}$ and $\delta_{g,f} : D(g \bullet f) \to Df \circ Dg$ are defined by:

$$\epsilon_{\mathsf{A},i} : \mathsf{T}(i, V \circ \mathsf{A}) = V\mathsf{A}(i) \tag{4.2}$$

$$\begin{aligned}\delta_{g,f;\mathsf{A};i} :& \mathsf{T}((g \bullet f)(i), V \circ \mathsf{A}) \xrightarrow{\sim} \mathsf{T}(f, \mathsf{T}(g(\_), V \circ \mathsf{A})) \\ & \xrightarrow{\mathsf{T}(f, \mathsf{T}(g(\_), \delta_{\mathsf{A}}))} \mathsf{T}(f, \mathsf{T}(g(\_), V \circ V \circ \mathsf{A})) \xrightarrow{\mathsf{T}(f, \gamma^{g(\_)}_{\mathsf{A}})} Df(Dg\mathsf{A})(i).\end{aligned} \tag{4.3}$$

Theorem 4.2. *The symmetric colax monoidal functor D (4.1) and monoidal natural transformations ϵ, δ (4.2,4.3) determine an ILEC over C.*

4.3 GBLL Semantics by Realizability Category

Hofmann et al., and also Dal Lago et al. employ a *realizability semantics* to show that the complexity of BLL proof reductions belongs to P-time [19,10]. In this section we compare their semantics and the simple semantics of GBLL constructed in the previous section.

We instantiate $\mathbb{C}$ in the previous section with the realizability category over a BCI algebra $(A, \cdot)$, which is a combinatory algebra based on B, C, I-combinators; see e.g. [2,20]. We then form the realizability category $\mathbf{Ass}(A)$ by the following data: an object is a function f into P^+A, where P^+ is the nonempty powerset construction, and a morphism from f to g is a function $h : \operatorname{dom} f \to \operatorname{dom} g$ with the following property: there exists an element $e \in A$ such that for any $x \in \operatorname{dom} f$ and $a \in f(x)$, we have $e \cdot a \in g(h(x))$. The category $\mathbf{Ass}(A)$ is symmetric monoidal closed; see e.g. [20, Proposition 4]. The tensor product of f and g is given by $(f \otimes g)(x, y) = \{u \boxtimes v \mid u \in f(x), v \in g(y)\}$, where $u \boxtimes v$ is the BCI-algebra element corresponding to $\lambda x.xuv$ [20, Section 2].

Next, let Δ be a set and consider the power category $\Delta \pitchfork \mathbf{Ass}(A)$. Under the axiom of choice, $\Delta \pitchfork \mathbf{Ass}(A)$ is equivalently described as follows: an object is a family of functions $\{f_i\}_{i\in\Delta}$ into P^+A, and a morphism from $\{f_i\}_{i\in\Delta}$ to $\{g_i\}_{i\in\Delta}$ is a family of functions $\{h_i : \mathrm{dom}\, f_i \to \mathrm{dom}\, g_i\}_{i\in\Delta}$ with the following property: there exists a function $e : \Delta \to A$ such that for any $i \in \Delta$, $x \in \mathrm{dom}\, f_i$ and $a \in f_i(x)$, we have $e(i) \cdot a \in g_i(h_i(x))$.

This power category is quite close to the realizability category introduced in [19, Section 4] and [10, Section 4]. A membership statement $a \in f_i(x)$ for an object $\{f_i\}_{i\in\Delta} \in \Delta \pitchfork \mathbf{Ass}(A)$ corresponds to a realizability statement $i, a \Vdash x$ in the realizability category (see [19]). The major difference between these categories is twofold: 1) In the realizability category, a computability constraint is imposed on $e : \Delta \to A$ to achieve the characterization of P-time complexity. 2) Objects in the realizability category are limited to $\Delta \pitchfork \mathbf{Ass}(A)$-objects such that all f_i share the common domain. This is to synchronize with the set-theoretic semantics *ignoring* resource polynomials [19, Section 3] [10, Section 3].

We compute the bounded !-modality using the folding product ILEC T with respect to the indexed SMCC $(\_) \pitchfork \mathbf{Ass}(A)$. Let F be a finite set of variables, $x \notin F$ be a resource variable, p be a resource polynomial and $\mathscr{C}$ be a constraint set under F. For any object X in $V_{\mathscr{C}\cup\{v\leq p\}}(F\cup\{v\}) \pitchfork \mathbf{Ass}(A)$, the folding product $\mathsf{T}([v \leq p]_{(F,\mathscr{C})}, \mathsf{X})$ is an object in $V_{\mathscr{C}}(F) \pitchfork \mathbf{Ass}(A)$ satisfying

$$\begin{aligned}&\mathsf{T}([v < p]_{(F,\mathscr{C})}, \mathsf{X})(i)\\ &\quad = \lambda(x_0, \cdots, x_{[\![p]\!]i-1}) \ . \ \{a_0 \otimes \cdots \otimes a_{[\![p]\!]i-1} \mid a_j \in \mathsf{X}(i\{v \mapsto j\})(x_j)\}\end{aligned} \tag{4.4}$$

This is different from the modality over the realizability category introduced in [19, Definition 16] and [10, Definition 4.6]:

$$(!_{v<p}\mathsf{X})(i) = \lambda x \ . \ \{a_0 \otimes \cdots \otimes a_{[\![p]\!]i-1} \mid a_j \in \mathsf{X}(i\{v \mapsto j\})(x)\};$$

it only takes a single argument. This is again because their realizability semantics is designed to synchronize with the set-theoretic semantics ignoring resource polynomials — especially it interprets $[\![!_{x\leq p}A]\!] = [\![A]\!]$. On the other hand, the bounded quantification computed in (4.4) does *not* ignore resource polynomials and indexing, as the domain of (4.4) is the index-dependent product $\prod_j \mathrm{dom}(\mathsf{X}(i\{v \mapsto j\}))$. From this, we conjecture that the semantics of BLL using the ILEC T over $(\_) \pitchfork \mathbf{Ass}(A)$ realizes an *index-dependent* set-theoretic semantics of BLL — we leave this semantics as a future work.

5 Conclusion and Related Work

We introduced GBLL, a generalization of Girard et al.'s BLL. We analyzed the complexity of cut-elimination in GBLL, and gave a translation from CBLL, an extension of BLL with constraints to GBAL^+. We then introduced ILEC as a categorical structure for interpreting the !-modality of GBLL. The ILEC is a **Idx**-graded linear exponential comonad interacting well with a specified indexed SMCCs. We gave an elementary construction of ILEC using the folding product, and a technique to derive its variants by inserting symmetric monoidal comonads. We gave the semantics of BLL using the folding product on the category of assemblies of a BCI-algebra, and related with the realizability category studied in [19,10].

Girard's BLL has a great influence on the subsequent development of indexed modalities and implicit complexity theory [16]. Hofmann and Scott introduced the realizability technique to BLL and semantically proved that BLL characterizes P-time complexity [19]. Their work was further enriched and studied by Dal Lago and Hofmann [10]. Gaboardi combined the !-modality involving variable binding with PCF and showed that the combined system is relatively complete [24].

Bucciarelli and Ehrhard's *indexed linear logic with exponential* [9] is one of the closest systems to GBLL. However, the type of the !-modality is different: their system derives $\Delta \vdash !_f A$ from $\Delta' \vdash A$ and an *almost injective function* $f : \Delta' \to \Delta$; it is a function where each $f^{-1}(i)$ is finite. To relate their system and GBLL, let us use the finite powerset construction P_{fin} and convert f into its inverse $f^{-1} : \Delta \to P_{\mathrm{fin}}(\Delta')$. This exhibits the similarity with GBLL: GBLL relaxes P_{fin} to $(\_)^*$, and takes the inverse as the parameter for the !-modality. The novelty of this work to [9] is that a categorical axiomatization for the $!_f$ modality is identified as an extension of the graded linear exponential comonads [7,22]. Another novelty is to show that GBLL is enough to encode BLL.

As described in Section 1, the simple form of !-modality $!_r A$ is also widely used in various type systems and programming languages. Examples include: INTML [30], coeffect calculus [28,7] and its combination with effect systems [13], Granule language [26], bounded linear type system [14,26], type systems for the analysis of higher-order model-checking [18,17], a generic BLL-like logic $\mathrm{B}_{\mathcal{S}}\mathrm{LL}$ over semirings [6], Fuzz type system for function sensitivity and differential privacy [29,12,3], and many more. A combination of $!_r A$ with dependent type theory called QTT is also introduced in [25] and [4]. Among these systems, each of [12,26,1] supports 1) full universal and existential, 2) full universal and 3) partial universal quantification over grades, respectively.

The categorical structure corresponding to the simple form of !-modality appears in [7,13,22] and is identified as *semiring-graded linear exponential comonad*. Breuvart constructed various examples of semiring-graded linear exponential comonads on relational models of linear logic [6] using his *slicing* technique. In this work we replaced semirings to **Idx**, which may be seen as a multi-object pseudo-semiring. In the study of graded monad, Orchard et al. generalize the grading structure from ordered monoids to 2-categories [27]. The main difference from this work is that their generalized graded monad is defined over a single category, while an ILEC is defined over an *indexed* SMCCs.

References

1. Abel, A., Bernardy, J.P.: A unified view of modalities in type systems. Proc. ACM Program. Lang. **4**(ICFP) (Aug 2020). https://doi.org/10.1145/3408972

2. Abramsky, S., Lenisa, M.: Linear realizability and full completeness for typed lambda-calculi. Ann. Pure Appl. Log. **134**(2-3), 122–168 (2005). https://doi.org/10.1016/j.apal.2004.08.003
3. de Amorim, A.A., Gaboardi, M., Hsu, J., Katsumata, S., Cherigui, I.: A semantic account of metric preservation. In: Castagna, G., Gordon, A.D. (eds.) Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017. pp. 545–556. ACM (2017). https://doi.org/10.1145/3009837, http://dl.acm.org/citation.cfm?id=3009890
4. Atkey, R.: Syntax and semantics of quantitative type theory. In: Dawar, A., Grädel, E. (eds.) Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018. pp. 56–65. ACM (2018). https://doi.org/10.1145/3209108.3209189
5. Breuvart, F.: Dissecting Denotational Semantics: From the Well-established $\mathcal{H}^*$ to the More Recent Quantitative Coeffects. Ph.D. thesis, Université Paris Diderot (2015), https://lipn.univ-paris13.fr/~breuvart/These_breuvart.pdf
6. Breuvart, F., Pagani, M.: Modelling coeffects in the relational semantics of linear logic. In: Kreutzer [23], pp. 567–581, http://www.dagstuhl.de/dagpub/978-3-939897-90-3
7. Brunel, A., Gaboardi, M., Mazza, D., Zdancewic, S.: A core quantitative coeffect calculus. In: Shao, Z. (ed.) Programming Languages and Systems. pp. 351–370. Springer Berlin Heidelberg, Berlin, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54833-8_19
8. Bucciarelli, A., Ehrhard, T.: On phase semantics and denotational semantics in multiplicative-additive linear logic. Ann. Pure Appl. Log. **102**(3), 247–282 (2000). https://doi.org/10.1016/S0168-0072(99)00040-8
9. Bucciarelli, A., Ehrhard, T.: On phase semantics and denotational semantics: the exponentials. Annals of Pure and Applied Logic **109**(3), 205 – 241 (2001). https://doi.org/10.1016/S0168-0072(00)00056-7, http://www.sciencedirect.com/science/article/pii/S0168007200000567
10. Dal Lago, U., Hofmann, M.: Bounded linear logic, revisited. In: Curien, P.L. (ed.) Typed Lambda Calculi and Applications. pp. 80–94. Springer Berlin Heidelberg, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02273-9_8
11. Fujii, S., Katsumata, S., Melliès, P.: Towards a formal theory of graded monads. In: Jacobs, B., Löding, C. (eds.) Foundations of Software Science and Computation Structures - 19th International Conference, FOSSACS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings. Lecture Notes in Computer Science, vol. 9634, pp. 513–530. Springer (2016). https://doi.org/10.1007/978-3-662-49630-5_30
12. Gaboardi, M., Haeberlen, A., Hsu, J., Narayan, A., Pierce, B.C.: Linear dependent types for differential privacy. In: Giacobazzi, R., Cousot, R. (eds.) The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '13, Rome, Italy - January 23 - 25, 2013. pp. 357–370. ACM (2013). https://doi.org/10.1145/2429069.2429113, http://dl.acm.org/citation.cfm?id=2429069
13. Gaboardi, M., Katsumata, S., Orchard, D., Breuvart, F., Uustalu, T.: Combining effects and coeffects via grading. SIGPLAN Not. **51**(9), 476–489 (Sep 2016). https://doi.org/10.1145/3022670.2951939
14. Ghica, D.R., Smith, A.I.: Bounded linear types in a resource semiring. In: Shao, Z. (ed.) Programming Languages and Systems. pp. 331–350. Springer Berlin Heidelberg, Berlin, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54833-8_18

15. Girard, J.: Linear logic. Theor. Comput. Sci. **50**, 1–102 (1987). https://doi.org/10.1016/0304-3975(87)90045-4
16. Girard, J., Scedrov, A., Scott, P.J.: Bounded linear logic: A modular approach to polynomial-time computability. Theor. Comput. Sci. **97**(1), 1–66 (1992). https://doi.org/10.1016/0304-3975(92)90386-T
17. Grellois, C., Melliès, P.: An infinitary model of linear logic. In: Pitts, A.M. (ed.) Foundations of Software Science and Computation Structures - 18th International Conference, FoSSaCS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings. Lecture Notes in Computer Science, vol. 9034, pp. 41–55. Springer (2015). https://doi.org/10.1007/978-3-662-46678-0_3
18. Grellois, C., Melliès, P.: Relational semantics of linear logic and higher-order model checking. In: Kreutzer [23], pp. 260–276, http://www.dagstuhl.de/dagpub/978-3-939897-90-3
19. Hofmann, M., Scott, P.J.: Realizability models for bll-like languages. Theor. Comput. Sci. **318**(1-2), 121–137 (2004). https://doi.org/10.1016/j.tcs.2003.10.019
20. Hoshino, N.: Linear realizability. In: Duparc, J., Henzinger, T.A. (eds.) Computer Science Logic, 21st International Workshop, CSL 2007, 16th Annual Conference of the EACSL, Lausanne, Switzerland, September 11-15, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4646, pp. 420–434. Springer (2007). https://doi.org/10.1007/978-3-540-74915-8_32
21. Hyland, M., Power, J.: Pseudo-commutative monads and pseudo-closed 2-categories. Journal of Pure and Applied Algebra **175**(1), 141 – 185 (2002). https://doi.org/10.1016/S0022-4049(02)00133-0, http://www.sciencedirect.com/science/article/pii/S0022404902001330, special Volume celebrating the 70th birthday of Professor Max Kelly
22. Katsumata, S.: A double category theoretic analysis of graded linear exponential comonads. In: Baier, C., Dal Lago, U. (eds.) Foundations of Software Science and Computation Structures. pp. 110–127. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-89366-2_6
23. Kreutzer, S. (ed.): 24th EACSL Annual Conference on Computer Science Logic, CSL 2015, September 7-10, 2015, Berlin, Germany, LIPIcs, vol. 41. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2015), http://www.dagstuhl.de/dagpub/978-3-939897-90-3
24. Lago, U.D., Gaboardi, M.: Linear dependent types and relative completeness. In: Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science, LICS 2011, June 21-24, 2011, Toronto, Ontario, Canada. pp. 133–142. IEEE Computer Society (2011). https://doi.org/10.1109/LICS.2011.22, https://ieeexplore.ieee.org/xpl/conhome/5968099/proceeding
25. McBride, C.: I got plenty o' nuttin'. In: Lindley, S., McBride, C., Trinder, P.W., Sannella, D. (eds.) A List of Successes That Can Change the World - Essays Dedicated to Philip Wadler on the Occasion of His 60th Birthday. Lecture Notes in Computer Science, vol. 9600, pp. 207–233. Springer (2016). https://doi.org/10.1007/978-3-319-30936-1_12
26. Orchard, D., Liepelt, V.B., Eades III, H.: Quantitative program reasoning with graded modal types. Proc. ACM Program. Lang. **3**(ICFP) (Jul 2019). https://doi.org/10.1145/3341714
27. Orchard, D., Wadler, P., Eades, H.: Unifying graded and parameterised monads. Electronic Proceedings in Theoretical Computer Science **317**, 18–38 (May 2020). https://doi.org/10.4204/eptcs.317.2

28. Petricek, T., Orchard, D., Mycroft, A.: Coeffects: Unified static analysis of context-dependence. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) Automata, Languages, and Programming. pp. 385–397. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39212-2_35
29. Reed, J., Pierce, B.C.: Distance makes the types grow stronger: a calculus for differential privacy. In: Hudak, P., Weirich, S. (eds.) Proceeding of the 15th ACM SIGPLAN international conference on Functional programming, ICFP 2010, Baltimore, Maryland, USA, September 27-29, 2010. pp. 157–168. ACM (2010). https://doi.org/10.1145/1863543.1863568
30. Schöpp, U.: Computation-by-interaction with effects. In: Yang, H. (ed.) Programming Languages and Systems. pp. 305–321. Springer Berlin Heidelberg, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25318-8_23