Bounded Linear Logic, Revisited

Ugo Dal Lago[1] and Martin Hofmann[2]

[1] Dipartimento di Scienze dell'Informazione, Università di Bologna
[2] Institut für Informatik, LMU München

Abstract. We present QBAL, an extension of Girard, Scedrov and Scott's bounded linear logic. The main novelty of the system is the possibility of quantifying over resource variables. This generalization makes bounded linear logic considerably more flexible, while preserving soundness and completeness for polynomial time. In particular, we provide compositional embeddings of Leivant's RRW and Hofmann's LFPL into QBAL.

1 Introduction

After two decades from the pioneering works that started it [3,13,14], implicit computational complexity is now an active research area at the intersection of mathematical logic and computer science. Its aim is the study of machine-free characterizations of complexity classes. The correspondence between an ICC system and a complexity class holds *extensionally*, i.e., the class of functions (or problems) which are representable in the system equals the complexity class. Usually, the system is a fragment or subsystem of a larger programming language or logical system, the *base system*, in which other functions besides the ones in the complexity class can be represented. Sometimes, one of the two inclusions is shown by proving that any program (or proof) can be reduced in bounded time; in this case, we say that the system is intensionally sound. On the other hand, ICC systems are very far from being intensionally *complete*: there are many programs (or proofs) in the base system which are not in the ICC system, even if they can be evaluated with the prescribed complexity bounds. Observe that this does not contradict extensional completeness, since many different programs or proofs compute the same function.

Of course, a system that captures all and only the programs of the base system running within a prescribed complexity bound will in all but trivial cases (e.g., empty base system) fail to be recursively enumerable. Thus, in practice, one strives to improve intensional expressivity by capturing important classes of examples and patterns.

An obstacle towards applying ICC characterizations of complexity classes to programming language theory is their poor intensional expressive power: most ICC systems do not capture natural programs and therefore are not useful in practice. This problem has been already considered in the literature. Some papers try to address the poor intensional expressive power of ICC systems by defining new programming languages allowing to program in ways which are not allowed

P.-L. Curien (Ed.): TLCA 2009, LNCS 5608, pp. 80–94, 2009.

in existing ICC systems. This includes quasi-interpretations [15] and LFPL by the second author [10]. Other papers analyze the intensional expressive power of existing systems either by studying necessary conditions on captured programs or, more frequently, by studying relations between existing ICC systems. One nice example is Murawski and Ong's paper [16], in which the authors prove that there cannot be any embedding (satisfying certain properties) of Bellantoni and Cook function algebra BC [3] into light affine logic [1]. In this work, we somehow combine the two approaches, by showing that a new logical system, called QBAL is intensionally at least as expressive as two heterogeneous, existing systems, namely Leivant's RRW [14] and LFPL.

QBAL is a generalization of Girard, Scedrov and Scott's bounded linear logic (BLL, [8]), itself the first characterization of polynomial time computable functions as a fragment of Girard's linear logic [6]. Bounded linear logic has received relatively little attention in the past [11,17]. This is mainly due to its syntax, which is more involved than the one of other complexity-related fragments of linear logic appeared more recently [7,12,5].

In bounded linear logic, polynomials are part of the syntax and, as a consequence, computation time is controlled explicitly. However, it seems that BLL is not as intensionally expressive as to be able to embed any existing ICC system corresponding to polynomial time (except Lafont's SLL [12], which anyway was conceived as a very small fragment of BLL). QBAL is obtained by endowing BLL with bounded quantification on resource variables. In other words, formulas of QBAL includes the ones of BLL, plus formulas like $\exists x : \{x \leq y^2\}.A$ or $\forall x, y : \{x \leq z, y \leq z^3\}.B$. This new feature by itself increases the intensional expressive power: both RRW and LFPL can be compositionally embedded into QBAL. Moreover, QBAL remains sound with respect to polynomial time. For these reasons, QBAL is *not* just another system capturing polynomial time computable functions.

An extended version of this paper including all proofs is available [4].

2 Syntax

In this Section, we present the syntax of QBAL, together with some of its main properties. The lack of space prevents us from being exhaustive, but all the details can be found in [4]. In the following, we adhere to the notation adopted in the relevant literature on BLL [8,11].

Resource polynomials are finite sums of products of binomial coefficients, i.e. they can be written as $\sum_{j \leq m} \prod_{i \leq k_j} \binom{x_{ij}}{n_{ij}}$ where the x_{ij} are pairwise distinct and n_{ij} are natural numbers. Resource polynomials are closed under binary sum, binary product, bounded sum and composition [8]. Order relations between resource polynomials are captured by constraints and constraint sets:

Definition 1 (Constraints)

- *A* constraint *is an inequality in the form $p \leq q$, where p and q are resource polynomials. A* constraint set *is a finite set of constraints. Constraint sets are denoted with letters like $\mathscr{C}$ or $\mathscr{D}$.*

- *For each constraint set $\mathscr{C}$, we define an order $\sqsubseteq_{\mathscr{C}}$ on resource polynomials by imposing $p \sqsubseteq_{\mathscr{C}} q$ iff $\mathscr{C} \models p \leq q$, i.e., the (pointwise) inequality $p \leq q$ is a logical consequence of $\mathscr{C}$.*
- *$\mathscr{C} \models \mathscr{D}$ iff $\mathscr{C} \models p \leq q$ for every constraint $p \leq q$ in $\mathscr{D}$.*

Resource polynomials, constraints and constraint sets becomes, in turn, the essential ingredients in the definition of QBAL formulas:

Definition 2. *Formulas of QBAL are defined as follows:*

$$A ::= \alpha(p_1, \ldots, p_n) \mid A \otimes A \mid A \multimap A \mid \forall \alpha.A \mid !_{x<p}A \mid$$
$$\forall(x_1, \ldots, x_n) : \mathscr{C}.A \mid \exists(x_1, \ldots, x_n) : \mathscr{C}.A$$

where $x \notin FV(p)$, α ranges over a countable class of atoms (each with an arity n). We will restrict ourselves to bounded *first order quantification. In other words, whenever we write $\forall(x_1, \ldots, x_n) : \mathscr{C}.A$ or $\exists(x_1, \ldots, x_n) : \mathscr{C}.A$ we implicitly assume that for every i there is a resource polynomial p_i not containing the variables $x_1, \ldots, x_n$ such that $\mathscr{C} \models \{x_1 \leq p_1, \ldots, x_n \leq p_n\}$.*

Checking the boundedness condition on formulas is undecidable in general (but here, we are not concerned about the complexity of QBAL as a verification technique). Notice that resource polynomials and the variables in them can occur inside constraints, constraint sets and formulas. The following definition becomes natural:

Definition 3 (Positive and Negative Occurrences)

- *Any variable in p (respectively, in q) occurs negatively (respectively, positively) in the constraint $p \leq q$. Analogously for constraint sets.*
- *The definition of a positive (or negative) occurrence of a variable in a formula A is defined by induction on A:*
  - *All the variables in $FV(p_1) \cup \ldots \cup FV(p_n)$ occur positively in $\alpha(p_1, \ldots, p_n)$.*
  - *Polarities are propagated through compound formulas by noting that $\multimap$ is negative in the first slot, $!_{x<p}$ is negative in p and $\forall \overline{x}{:}\mathscr{C}$ is negative in $\mathscr{C}$. All other slots are positive. For example, the first occurrence of x in $\forall y{:}\{y \leq x\}.\alpha(x, y)$ is negative while the second one is positive. We omit the detailed definition.*
- *Let B be a formula where the free variables $x_1, \ldots, x_n$ occur only positively. Then $A\{B/\alpha(x_1, \ldots, x_n)\}$ denotes the formula obtained by replacing every free occurrence of $\alpha(p_1, \ldots, p_n)$ with $B\{p_1/x_1, \ldots, p_n/x_n\}$ inside A.*
- *The order $\sqsubseteq_{\mathscr{C}}$ can be extended to an order on formulas with the same skeleton in a natural way (see [4] for a formal definition).*

A QBAL judgement is an expression in the form $\Gamma \vdash_{\mathscr{C}} A$, where $\mathscr{C}$ is a constraint set, Γ is a multiset of formulas and A is a formula. Rules of inference for QBAL are in Figure 1. All rules except first order ones are the natural generalizations of BLL rules. First order rules are similar to the standard ones from predicate logic as a sequent calculus, with the additional complexity brought on by constraints.

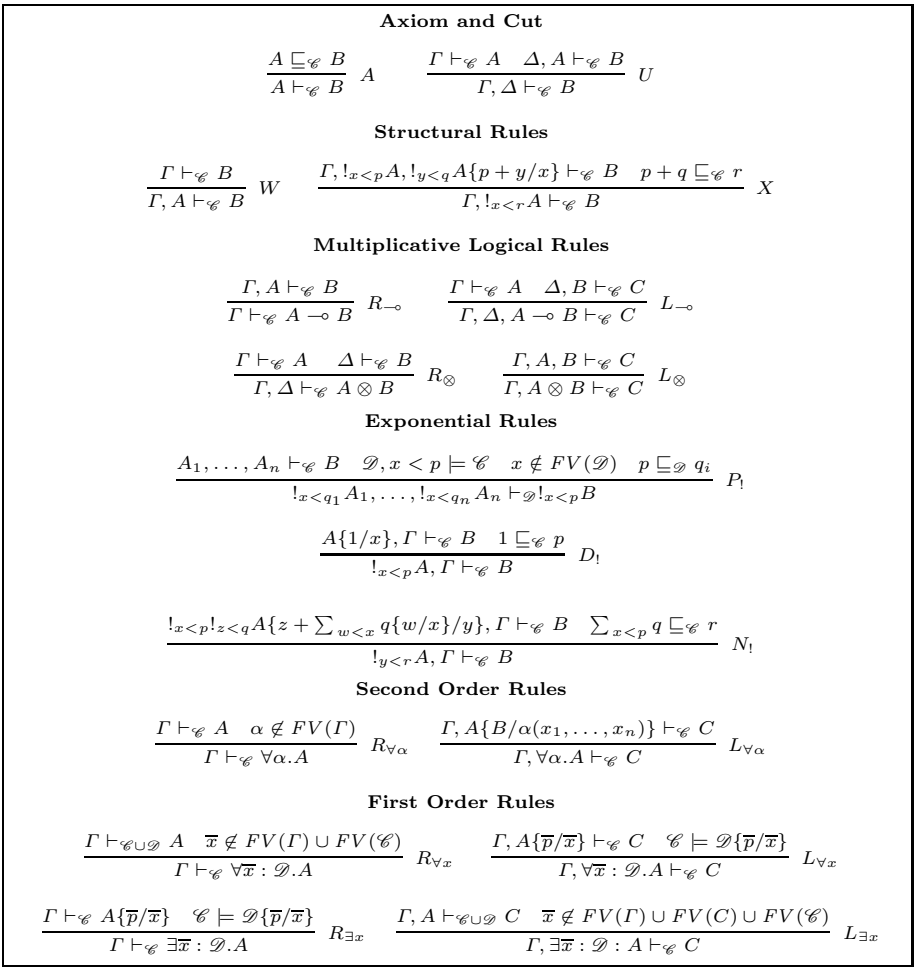

Fig. 1. A sequent calculus for QBAL

2.1 QBAL and Second Order Logic

Second order intuitionistic logic can be presented as a context-independent sequence calculus with explicit structural rules [18], G2i. There is a forgetful map $[\cdot]$ from the space of QBAL proofs to the space of G2i proofs. In particular $\multimap$ corresponds to $\rightarrow$ and $\otimes$ corresponds to $\wedge$. Essentially, $[\pi]$ have the same structure of π, except for exponential and first order rules, which have no formal correspondence in G2i. From our point of view, if $[\pi] = [\rho]$, then π and ρ correspond to the same *program*, i.e. QBAL can be seen as a proper decoration of second order logic proofs with additional information which are not necessary to perform the underlying computation.

2.2 Properties

QBAL inherits some nice properties from BLL. In particular, proofs can be manipulated in a uniform way by altering their conclusion without altering their structure, i.e., without changing the underlying second order logic proof. Suppose that $\pi : A_1, \ldots, A_n \vdash_{\mathscr{C}} B$ is a QBAL proof. Then, we can construct other proofs with the same skeleton as π. For example:

- Whenever $B \sqsubseteq_{\mathscr{C}} D$ and $C_i \sqsubseteq_{\mathscr{C}} A_i$ for every $1 \leq i \leq n$, there is $\rho : C_1, \ldots, C_n \vdash_{\mathscr{C}} D$ such that $[\rho] = [\pi]$.
- Whenever $p_1, \ldots, p_n$ are resource polynomials free for substitution for the free resource variables $x_1, \ldots, x_n$ in π, there is a proof $\pi\{\overline{p}/\overline{x}\} : A_1\{\overline{p}/\overline{x}\}, \ldots, A_n\{\overline{p}/\overline{x}\} \vdash_{\mathscr{C}\{\overline{p}/\overline{x}\}} B\{\overline{p}/\overline{x}\}$ such that $[\pi\{\overline{p}/\overline{x}\}] = [\pi]$.
- Whenever $\mathscr{D} \models \mathscr{C}$ there is a proof $\rho : A_1, \ldots, A_n \vdash_{\mathscr{D}} B$ such $[\rho] = [\pi]$.

Unfortunately, space constraints prevent us from explicitly giving the details and proofs of the results above (see [4]).

2.3 Cut-Elimination

A nice application of the observations we have just given is cut-elimination. Indeed, the new rules $R_{\forall x}$, $L_{\forall x}$, $R_{\exists x}$ and $L_{\exists x}$ do not cause any problem in the cut-elimination process. See [4] for more information. In this paper, we will not study cut-elimination. And polynomial time soundness will be itself proved semantically.

2.4 Programming in QBAL

The Curry-Howard correspondence allows to use BLL and QBAL as programming languages. In particular, following the usual impredicative encoding of data into second order intuitionistic logic, natural numbers can be represented as cut-free proofs of the formula

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

However, only natural numbers less or equal to p can be represented.

This can be generalized to any word algebra. Given a word algebra $\mathbb{W}$, we will denote by $\varepsilon_{\mathbb{W}}$ the only 0-ary constructor of $\mathbb{W}$ and by $c^1_{\mathbb{W}}, \ldots, c^w_{\mathbb{W}}$ the 1-ary constructors of the same algebra. Notice that these objects can be thought of both as term formers and as (0-ary or unary) functions on terms. Terms of a free algebra $\mathbb{W}$ of length at most p can be represented as cut-free proofs of the formula

$$\mathbf{W}_p = \underbrace{\forall\alpha.!_{y<p}(\alpha(y) \multimap \alpha(y+1)) \multimap \ldots \multimap !_{y<p}(\alpha(y) \multimap \alpha(y+1))}_{w \text{ times}} \multimap \alpha(0) \multimap \alpha(p).$$

Functions on natural numbers can be represented by proofs with conclusion $\mathbf{N}_x \vdash \mathbf{N}_p$, where p is a resource polynomial depending on x, only. More generally, functions on the word algebra $\mathbb{W}$ can be represented by proofs with conclusion $\mathbf{W}_x \vdash \mathbf{W}_p$. For example, all constructors $c^1_{\mathbb{W}}, \ldots, c^w_{\mathbb{W}}$ corresponds to proofs with

conclusion $\mathbf{W}_x \vdash \mathbf{W}_{x+1}$. More generally, the polynomial p gives a bound on the size of the result, as a function of the size of the input. QBAL supports iteration on any word algebra (including natural numbers). As an example, for every p and for every A where x only appears positively, there is a proof π_p^A of $\mathbf{N}_p, A\{y/x\} \multimap A\{y+1/x\}, A\{0/x\} \vdash A\{p/x\}$.

2.5 Unbounded First Order Quantification Is Unsound

One may wonder why quantification on numerical variables is restricted to be bounded (see Definition 2). The reason is very simple: in presence of unbounded quantification, QBAL would immediately become unsound. To see that, define $\mathbf{N}_\infty$ to be the formula $\exists(x) : \emptyset.\mathbf{N}_x$. The composition of the successor with itself yields a proof with conclusion $\mathbf{N}_x \vdash \mathbf{N}_{x+2}$ which, by rules $R_{\exists x}$ and $L_{\exists x}$, becomes a proof with conclusion $\mathbf{N}_\infty \vdash \mathbf{N}_\infty$. Iterating it, we obtain a proof of $\mathbf{N}_x \vdash \mathbf{N}_\infty$ which computes the function $n \mapsto 2n$. But by rule $L_{\exists x}$, it can be turned into a proof of $\mathbf{N}_\infty \vdash \mathbf{N}_\infty$, and iterating it again we obtain a proof computing the exponential function. The boundedness assumption will be indeed critical in Section 4, where we establish that any functions which is representable in QBAL is polynomial time computable.

3 Set-Theoretic Semantics

In this Section, we give a set-theoretic semantics for QBAL. We assume that our ambient set-theory is constructive. This way we have a set of sets $\mathcal{U}$ which contains the natural numbers, closed under binary products, function spaces and $\mathcal{U}$-indexed products. An alternative to assuming a constructive ambient set theory consists of replacing plain sets with PERs (partial equivalence relations) or domains or similar structures. See [11] for a more detailed discussion of this issue.

A formula A can be interpreted as a set $\llbracket A \rrbracket_\rho$, where ρ is an environment mapping atoms to sets. For example, $\llbracket \alpha(\overline{p}) \rrbracket_\rho = \rho(\alpha)$, while $\llbracket \forall \overline{x} : \mathscr{C}.A \rrbracket_\rho$ is simply $\llbracket A \rrbracket_\rho$. Please observe that the interpretation of any formula A is completely independent from the resource polynomials appearing in A.

To any QBAL proof π of $A_1, \ldots, A_n \vdash_{\mathscr{C}} B$ we can associate a set-theoretic function $\llbracket \pi \rrbracket_\rho : \llbracket A_1 \otimes \ldots \otimes A_n \rrbracket_\rho \to \llbracket B \rrbracket_\rho$ by induction on π, in the obvious way. $\llbracket \pi \rrbracket_\rho$ is *equal* to the set-theoretic semantics of $[\pi]$ as a proof of second order intuitionistic logic. Set-theoretic semantics of proofs is preserved by cut-elimination: if π reduces to σ by cut-elimination, then $\llbracket \pi \rrbracket_\rho = \llbracket \sigma \rrbracket_\rho$.

Observe that $\llbracket A \rrbracket_\rho$ only depends on the values of ρ on atoms appearing free in A. So, in particular,

$$\llbracket \mathbf{N}_q \rrbracket_\rho = \prod_{C \in \mathcal{U}} (C \Rightarrow C) \Rightarrow (C \Rightarrow C)$$

is independent on ρ and on q, since $\mathbf{N}_q$ is a closed formula. Similarly for $\llbracket \mathbf{W}_q \rrbracket_\rho$. Actually, there are functions $\varphi_{\mathbb{N}} : \mathbb{N} \to \llbracket \mathbf{N}_p \rrbracket$ and $\psi_{\mathbb{N}} : \llbracket \mathbf{N}_p \rrbracket \to \mathbb{N}$ such that $\psi_{\mathbb{N}} \circ \varphi_{\mathbb{N}}$ is the identity on natural numbers. They are defined as follows:

$$(\varphi_{\mathbb{N}}(n))_C(f,z) = f^n(z)$$
$$\psi_{\mathbb{N}}(x) = x_{\mathbb{N}}(x \mapsto x+1)(0)$$

So, given a proof $\pi : \mathbf{N}_x \vdash \mathbf{N}_p$, the numeric function represented by π is simply $\psi_{\mathbb{N}} \circ [\![\pi]\!] \circ \varphi_{\mathbb{N}}$. Similar arguments hold for functions with conclusion $\mathbf{W}_x \vdash \mathbf{W}_p$.

4 QBAL and Polynomial Time

In this Section we show that all functions on natural numbers definable in QBAL are polynomial time computable. To this end, we follow the semantic approach in [11] which we now summarise.

4.1 Realizability Sets

Let X be a finite set of variables. We write $\mathcal{V}(X)$ for $\mathbb{N}^X$—the elements of $\mathcal{V}(X)$ are called *valuations* (over X). If $\eta \in \mathcal{V}(X)$ 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 natural numbers allowing them to be passed as arguments to algorithms.

If $\mathscr{C}$ is a constraint set involving at most the variables in X (*over* X) then we write $\eta \models \mathscr{C}$ to mean that the valuation $\eta \in \mathcal{V}(X)$ satisfies all the constraints in $\mathscr{C}$. We write $\mathcal{P}(X)$ for the set of resource polynomials over X. If $p \in \mathcal{P}(X)$ and $\eta \in \mathcal{V}(X)$ we write $p(\eta)$ for the number obtained by evaluating p with $x \mapsto \eta(x)$ for each $x \in X$.

We assume known the untyped lambda calculus as defined e.g. in [2]. An untyped lambda term is *affine linear* if each variable (free or bound) appears at most once (up to α-congruence). For example, $\lambda x.\lambda y.yx$ and $\lambda x.\lambda y.y$ and $\lambda x.xy$ are affine linear while the term $\lambda x.xx$ is not. Notice that every affine linear 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 $O(|t|^2)$. 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 $O((|s| + |t|)^2)$.

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 $(\lambda x \otimes y.t)(u \otimes v) = t\{u/x, v/y\}$.

More generally, if $(t_i)_{i<n}$ is a family of affine lambda terms, we write $\bigotimes_{i<n} t_i$ for $\lambda f.ft_0t_1 \dots t_{n-1}$ and $\lambda \bigotimes_{i<n} x_i.t$ for $\lambda u.u(\lambda x_0 \lambda x_1 \dots \lambda x_{n-1}.t)$. Again,

$$(\lambda \bigotimes_{i<n} x_i.t)(\bigotimes_{i<n} t_i) = t\{t_0/x_0, \dots, t_{n-1}/x_{n-1}\}$$

We write Λ_a for the set of closed affine lambda terms.

There is a canonical way of representing terms of any word algebra $\mathbb{W}$ as affine lambda terms, which is attributed to Dana Scott [19]. For example, the natural number 2 corresponds to the term $\ulcorner 2 \urcorner = \lambda x.\lambda y.x(\lambda x.\lambda y.x(\lambda x.\lambda y.y))$.

Definition 4. *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 $\Vdash_A \subseteq \mathcal{V}(X) \times \Lambda_a \times |A|$ 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 1. (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

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

(ii) The realizability set $\mathbf{W}_x$ over $\{x\}$ of *free terms of* $\mathbb{W}$ ("of length at most x") is defined by: $|\mathbf{W}_x| = \mathbb{W}$ and

$$\eta, t \Vdash_{\mathbf{W}_x} w \text{ if } t = \ulcorner w \urcorner \text{ and } \eta(x) \geq |w|$$

These realizability sets $\mathbf{N}_x$ and $\mathbf{W}_x$ turn out to be retracts of the denotations of the BLL formulas from Section 2.4.

Definition 5. *Let A be a realizability set over X. We say that $x \in X$ is* positive *(*negative*, respectively) in A, if for all $\eta, \mu \in \mathcal{V}(X), t \in \Lambda_a, a \in |A|$ where η and μ agree on $X \setminus \{x\}$ and $\eta(x) \leq \mu(x)$ ($\eta(x) \geq \mu(x)$, respectively), $\eta, t \Vdash_A a$ implies $\mu, t \Vdash_A a$.*

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

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

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

In this case we say that e witnesses *f and write $A \to^f_e B$ where in the notation the algorithm e is presumed to exist.*

The following definition summarises the interpretation of formulas according to [11]:

Definition 7. *Let A, B be realizability sets over X. Then the following are realizability sets over X:*

- *$A \otimes B$ as given by $|A \otimes B| = |A| \times |B|$ and $\eta, t \Vdash_{A\otimes B} (a, b)$ iff $t = u \otimes v$, where $\eta, u \Vdash_A a$ and $\eta, v \Vdash_B b$.*
- *$A \multimap B$ is given by $|A \multimap B| = |A| \Rightarrow |B|$ and $\eta, t \Vdash_{A\multimap B} f$ iff whenever $\eta, u \Vdash_A a$ it holds that $\eta, t\, u \Vdash_B f(a)$.*

If C is a realizability set over $X \cup \{x\}$ and $p \in \mathcal{P}(X)$ then a realizability set $!_{x<p}C$ over X is defined by $|!_{x<p}C| = |C|$ and $\eta, t \Vdash_{!_{x<p}C} a$ if

- $t = \bigotimes_{i<p(\eta)} t_i$ *for some family* $(t_i)_{i<p(\eta)}$*;*
- $\eta[x \mapsto i], t_i \Vdash_C a$ *for each* $i < p(\eta)$*.*

We refer to [11] for the in principle straightforward but notationally cumbersome account of universal quantification over propositions.

Using these semantic constructions one defines for each formula A with free resource variables contained in X and assignment ρ of realizability sets to atoms, a realizability set $[\![A]\!]^{\mathcal{B}}_{\rho}$ over X in such a way that $|[\![A]\!]^{\mathcal{B}}_{\rho}| = [\![A]\!]_{|\rho|}$ (where $|\rho|$ is the assignment of sets to atoms obtained form ρ in the obvious way) , that is to say, the underlying set of the realizability set interpreting a formula A coincides with the set-theoretic meaning of A.

The main result of [11] then asserts that if π is a proof (in BLL) of a sequent $\Gamma \vdash B$ then the function $[\![\pi]\!]_{|\rho|}$ is a morphism from $[\![\Gamma]\!]^{\mathcal{B}}_{\rho}$ to $[\![B]\!]^{\mathcal{B}}_{\rho}$ (where we interpret a context Γ as a $\otimes$-product over its components as usual). From this, polynomial time soundness is a direct corollary since polynomial time computability is built into the notion of a morphism.

It thus only remains to extend the realizability model to cover the constructs of QBAL which we do in the next Section.

4.2 Extending the Realizability Model to QBAL

The notion of a realizability set above is adequate to model formulas of QBAL. The notion of a morphism, however, should be slightly generalized in order to capture constraints:

Definition 8. *Let* A, B *be realizability sets over some set* X *and* $\mathscr{C}$ *a constraint set over* X*. A function* $f : |A| \to |B|$*: is a* $\mathscr{C}$*-morphism from* A *to* B *iff there exists a function* $e : \mathcal{V}(X) \to \Lambda_a$ *such that* $e(\eta)$ *is computable in time* $q(\eta)$ *for some resource polynomial* q *and for each* $\eta \in \mathcal{V}(X)$ *with* $\eta \models \mathscr{C}$*,* $t \in \Lambda_a$*,* $a \in |A|$*, we have that* $\eta, t \Vdash_A a$ *implies* $\eta, e(\eta)t \Vdash_B f(a)$*.*

In order to define realizability sets $\forall \overline{y}{:}\mathscr{C}.A$ and $\exists \overline{y}{:}\mathscr{C}.A$, we fix some encoding of environments η as affine lambda terms using the $\ulcorner \cdot \urcorner$ encoding of natural numbers. We do not notationally distinguish environments from their encodings.

Definition 9. *Let* X, Y *be disjoint sets of variables. Let* A *be a realizability set over* $X \cup Y$ *and* $\mathscr{C}$ *a constraint set over* $X \cup Y$ *where we put* $Y = \{y_1, \ldots, y_n\}$ *and* $\overline{y} = (y_1, \ldots, y_n)$*. Furthermore, for each* $i = 1, \ldots, n$ *let* $p_i \in \mathcal{P}(X)$ *be such that* $\mathscr{C} \models \{\overline{y} \leq \overline{p}\}$*.*

- $|\forall \overline{y}{:}\mathscr{C}.A| = |\exists \overline{y}{:}\mathscr{C}.A| = |A|$*,*
- $\eta, t \Vdash_{\forall \overline{y}:\mathscr{C}.A} a \iff \forall \mu \in \mathcal{V}(Y).\eta \cup \mu \models \mathscr{C} \Rightarrow \eta \cup \mu, t\mu \Vdash_A a.$
- $\eta, \mu \otimes t \Vdash_{\exists \overline{y}:\mathscr{C}.A} a \iff \mu \in \mathcal{V}(Y) \wedge \eta \cup \mu \models \mathscr{C} \wedge \eta \cup \mu, t \Vdash_A a$

Recall that $\forall \overline{y}{:}\mathscr{C}.A$ *and* $\exists \overline{y}{:}\mathscr{C}.A$ *are well-formed only if for each* i *there is a resource polynomial* p_i *such that* $\mathscr{C} \models y_i < p_i$*. Therefore, the set* $\{\mu \mid \eta \cup \mu \models \mathscr{C}\}$ *is finite and in fact computable in polynomial time from* η*.*

We are now able to prove the main result of this Section:

Theorem 1. *Let π be a proof of a sequent $\Gamma \vdash_{\mathscr{C}} B$ and ρ a mapping of atoms to realizability sets. Then $[\![\pi]\!]_{|\rho|}$ is a $\mathscr{C}$-morphism from $[\![\Gamma]\!]^{\mathcal{B}}_{\rho}$ to $[\![B]\!]^{\mathcal{B}}_{\rho}$.*

Proof. The proof is by induction on derivations. We only show the cases that differ significantly from the development in [11].
Case $P_!$. For simplicity, suppose that $n = 1$, $q_1 = p$ and $A_1 = A$. The induction hypothesis shows that $[\![\pi]\!]_{|\rho|}$ is a $\mathscr{C}$-morphism from $[\![A]\!]^{\mathcal{B}}_{\rho}$ to $[\![B]\!]^{\mathcal{B}}_{\rho}$ witnessed by e. As in the proof of the main result in [11], we define

$$d(\eta) = \lambda \bigotimes_{i<p(\eta)} x_i. \bigotimes_{i<p(\eta)} e(\eta[x \mapsto i])x_i.$$

Now, if $\eta \models \mathscr{D}$, then $\eta[x \mapsto i] \models \mathscr{C}$ whenever $i < p(\eta)$ by the side condition from rule $P_!$. We obtain that $[\![\pi]\!]$ is a $\mathscr{D}$-morphism from $\left[\!\left[!_{x<p}A\right]\!\right]^{\mathcal{B}}_{\rho}$ to $\left[\!\left[!_{x<p}B\right]\!\right]^{\mathcal{B}}_{\rho}$ witnessed by d.
The remaining cases are the four rules for first order quantifiers. In each case, we assume by the induction hypothesis that $[\![\pi]\!]$ is a morphism realizing the premise of the rule and let e be its witness. We have to show that $[\![\pi]\!]$ is a morphism realizing the conclusion of the rule. Note that the set-theoretic meaning of a proof does not change upon application of any of the quantifier rules. We only consider universal quantification (existential quantification is dual).
Case $R_{\forall x}$. Suppose that $\eta \models \mathscr{C}$ and $\eta, t_\gamma \Vdash_{[\![\Gamma]\!]^{\mathcal{B}}_{\rho}} \gamma$. Now suppose $\eta \cup \mu \models \mathscr{D}$. By the induction hypothesis $\eta \cup \mu, e(\eta \cup \mu)t_\gamma \Vdash_{[\![A]\!]^{\mathcal{B}}_{\rho}} [\![\pi]\!](\gamma)$. We thus define d by $d(\eta) = u$ where $u \in \Lambda_a$ is such that $ut\mu = e(\eta \cup \mu)t$ whenever $\eta \cup \mu \models \mathscr{C}$. Recall that for a given η there are only $q(\eta)$ such μ (for a fixed resource polynomial q), so that t can be construed as a big case distinction over all those μ. It is then clear that d is polynomial time computable and realizes the conclusion of the rule.
Case $L_{\forall x}$. Assume $\eta \models \mathscr{C}$ and $\eta, t_\gamma \Vdash_{[\![\Gamma]\!]^{\mathcal{B}}_{\rho}} \gamma$ and $\eta, t_a \Vdash_{[\![\forall \overline{x}:\mathscr{D}.A]\!]^{\mathcal{B}}_{\rho}} a$. Define $\mu_\eta \in \mathcal{V}(X)$ by $\mu_\eta(x_i) = p_i(\eta)$ so that $\eta \cup \mu_\eta \models \mathscr{D}$ by the side condition to the rule. Now, $\eta \cup \mu_\eta, t_a\mu_\eta \Vdash_A a$ by Definition 9. Hence, $\eta, t_a\mu_\eta \Vdash_{A\{\overline{p}/\overline{x}\}} a$. By the induction hypothesis, $e(\eta)(t_\gamma \otimes t_a\mu_\eta) \Vdash_C [\![\pi]\!](\gamma, a)$, so $d(\eta) = \lambda x_\gamma \otimes x_a.e(\eta)(x_\gamma \otimes x_a(\mu_\eta))$ does the job. □

5 On Compositional Embeddings

In this Section, we try to justify our emphasis on compositional embeddings. An embedding of a logical system or programming language L into QBAL is a function $\langle\cdot\rangle$ from the space of proofs (or programs) of L into the space of proofs for QBAL. Clearly, for an embedding to be relevant from a computational point of view, any proof π of L should be mapped to an equivalent proof $\langle\pi\rangle$, e.g., $[\![\langle\pi\rangle]\!] = [\![\pi]\!]$. The existence of an embedding of L into QBAL implicitly proves that QBAL is *extensionally* at least as powerful as L. Such an embedding $\langle\cdot\rangle$ is not necessarily computable nor natural. But whenever L is a sound and complete ICC characterization of polynomial time, it must exists, since the classes of

definable functions in L and in QBAL are exactly the same. Indeed, QBAL is both extensionally sound (see Section 4) and extensionally complete (since BLL can be compositionally embedded into it, see below).

Typically, one would like to go beyond extensionality and prove that QBAL is *intensionally* as powerful as L. And if this is the goal, $\langle\cdot\rangle$ should be easily computable. Ideally, we would like $\langle\cdot\rangle$ to act homeomorphically on the space of proofs of L. In other words, whenever a proof π of L is obtained applying a proof-forming rule R to $\rho_1, \ldots, \rho_n$, then $\langle\pi\rangle$ should be obtainable from $\langle\rho_1\rangle, \ldots, \langle\rho_n\rangle$ in a uniform way, i.e., dependently on R but independently on $\langle\rho_1\rangle, \ldots, \langle\rho_n\rangle$. An embedding satisfying the above constraint is said to be *strongly compositional*. The embeddings we will present in the following two sections are only *weakly* compositional: $[\langle\pi\rangle]$ can be uniformly built from $[\langle\rho_1\rangle], \ldots, [\langle\rho_n\rangle]$ whenever π is obtained applying R to $\rho_1, \ldots, \rho_n$. We believe that the existence of a weakly compositional embedding of L into QBAL is sufficient to guarantee that QBAL is intensionally as powerful as L because, as we pointed out in Section 2.1, $[\pi]$ can be thought as the program hidden in the proof π.

Notice that BLL can be embedded into QBAL: for every BLL proof $\pi : \Gamma \vdash A$, there is a QBAL proof $\langle\pi\rangle : \Gamma \vdash_\emptyset A$ such that $[\pi] = [\langle\pi\rangle]$. Moreover, the embedding is strongly compositional.

6 Embedding LFPL

LFPL is a calculus for non-size-increasing computation introduced by the second author [10]. It allows to capture natural algorithms computing functions such that the size of the result is smaller or equal to the size of the arguments. This way, polynomial soundness is guaranteed despite the possibility of arbitrarily nested recursive definitions.

We here show that a core subset of LFPL can be compositionally embedded into QBAL. LFPL types are generated by the following grammar:

$$A ::= \diamond \mid \mathbf{N} \mid A \otimes A \mid A \multimap A.$$

Rules for LFPL in natural-deduction style are in Figure 2. We omit terms, since the computational content of type derivations is implicit in their skeleton. The set-theoretic semantics $[\![A]\!]$ of an LFPL formula A can be defined very easily: $[\![\diamond]\!] = \prod_{C\in\mathcal{U}} C \Rightarrow C$, $[\![\mathbf{N}]\!] = \prod_{C\in\mathcal{U}}(C \Rightarrow C) \Rightarrow (C \Rightarrow C)$, while the operators $\otimes$ and $\multimap$ are interpreted as usual. Notice that the interpretation of an LFPL formula does not depend on any environment ρ. This way, any LFPL proof $\pi : A_1, \ldots, A_n \vdash B$ can be given a semantics $[\![\pi]\!] : [\![A_1 \otimes \ldots \otimes A_n]\!] \to [\![B]\!]$, itself independent on any ρ.

LFPL types can be translated to QBAL formulas in the following way:

$$\begin{aligned}
\langle\diamond\rangle_p^q &= \exists\varepsilon : \{1 \le p\}.\forall\alpha.\alpha \multimap \alpha \\
\langle\mathbf{N}\rangle_p^q &= \mathbf{N}_p \\
\langle A \otimes B\rangle_p^q &= \exists(x,y) : \{x + y \le p\}.\langle A\rangle_x^q \otimes \langle B\rangle_y^q \\
\langle A \multimap B\rangle_p^q &= \forall(x) : \{x + p \le q\}.\langle A\rangle_x^q \multimap \langle B\rangle_{p+x}^q
\end{aligned}$$

Axiom, Base Types and Weakening

$$\frac{}{A \vdash A}\ A \qquad \frac{}{\mathbf{N}, \diamond \vdash \mathbf{N}}\ S \qquad \frac{\diamond \vdash A \multimap A \quad \vdash A}{\vdash \mathbf{N} \multimap A}\ T \qquad \frac{\Gamma \vdash A}{\Gamma, B \vdash A}\ W$$

Multiplicative Rules

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B}\ I_{\multimap} \qquad \frac{\Gamma \vdash A \multimap B \quad \Delta \vdash A}{\Gamma, \Delta \vdash B}\ E_{\multimap}$$

$$\frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}\ I_{\otimes} \qquad \frac{\Gamma \vdash A \otimes B \quad \Delta, A, B \vdash C}{\Gamma, \Delta \vdash C}\ E_{\otimes}$$

Fig. 2. LFPL

Please observe that the interpretation of any LFPL is parametrized on two resource polynomials p and q. If a variable x occurs in p, but not in q, then x occurs only positively in $\langle A \rangle_p^q$: this is an easy induction on the structure of A.

The correspondence scales to proofs:

Theorem 2. *LFPL can be embedded into QBAL. In other words, for every LFPL proof $\pi : A_1, \ldots, A_n \vdash B$, there exists a QBAL proof*

$$\langle \pi \rangle : \langle A_1 \rangle_{x_1}^y, \ldots, \langle A_n \rangle_{x_n}^y \vdash_{\{\sum_i x_i \leq y, 1 \leq y\}} \langle B \rangle_{\sum_i x_i}^y$$

such that $[\![\pi]\!] = [\![\langle \pi \rangle]\!]$. *Moreover, the correspondence $\langle \cdot \rangle$ is weakly compositional.*

Proof. As expected, the proof goes by induction on π. We just give two interesting cases:

- If the last rule used in π is $I_{\multimap}$, then $\pi : \Gamma \vdash A \multimap B$ and the immediate sub-proof is ρ, with conclusion $\Gamma, A \vdash B$. By induction hypothesis, there is $\langle \rho \rangle$ with the appropriate conclusion and $\langle \pi \rangle$ becomes:

$$\dfrac{\dfrac{\sigma : \langle \Gamma \rangle_{\overline{x}}^y, \langle A \rangle_z^y \vdash_{\{\sum_i x_i \leq y, \sum_i x_i + z \leq y, 1 \leq y\}} \langle B \rangle_{\sum_i x_i + z}^y}{\langle \Gamma \rangle_{\overline{x}}^y \vdash_{\{\sum_i x_i \leq y, \sum_i x_i + z \leq y, 1 \leq y\}} \langle A \rangle_z^y \multimap \langle B \rangle_{\sum_i x_i + z}^y}\ R_{\multimap}}{\langle \Gamma \rangle_{\overline{x}}^y \vdash_{\{\sum_i x_i \leq y, 1 \leq y\}} \langle A \multimap B \rangle_{\sum_i x_i}^y}\ R_{\forall x}$$

  where σ can be easily obtained from $\langle \rho \rangle$ by strengthening the underlying constraint set (see Section 2.2).
- If the last rule in π is T, then $\pi : \mathbf{N} \vdash A$ and the immediate sub-proofs are ρ with conclusion $\diamond \vdash A \multimap A$ and σ with conclusion $\vdash A$. By the induction hypothesis, there are $\langle \rho \rangle$ and $\langle \sigma \rangle$ with the appropriate conclusions and $\langle \pi \rangle$ becomes:

$$\dfrac{\dfrac{\dfrac{\theta : \langle \diamond \rangle_1^y \vdash_{\{z \leq y, 1 \leq y\}} \langle A \multimap A \rangle_1^y}{\vdash_{\{z \leq y, 1 \leq y\}} \langle A \multimap A \rangle_1^y}}{\vdash_{\{z \leq y, 1 \leq y\}} \langle A \rangle_z^y \multimap \langle A \rangle_{z+1}^y} \qquad \xi :\vdash_{\{z \leq y, 1 \leq y\}} \langle A \rangle_0^y}{\langle \mathbf{N} \rangle_z^y \vdash_{\{z \leq y, 1 \leq y\}} \langle A \rangle_z^y}$$

where θ and ξ are obtained by instantiating resource variables in $\langle \rho \rangle$ and $\langle \sigma \rangle$, respectively (see Section 2.2).

This concludes the proof. □

One may ask whether such an embedding might work for BLL proper. We believe this to be unlikely for several reasons. In particular, it seems that BLL lacks a mechanism for turning the information about the size of the manipulated objects from being global to being local. In QBAL, this rôle is played by first order quantifiers. As an example, considered the split function for lists of natural numbers that splits a list into two lists, one containing the even entries and one containing the odd entries. The type of that function in LFPL is $\mathbf{L}(\mathbf{N}) \multimap \mathbf{L}(\mathbf{N}) \otimes \mathbf{L}(\mathbf{N})$ where $\mathbf{L}(\cdot)$ denotes the type of lists that we have elided from our formal treatment for the sake of simplicity. In QBAL this function gets the type

$$\mathbf{L}_x(\mathbf{N}_y) \multimap \exists (u, v) : \{u + v \leq x\}.\mathbf{L}_u(\mathbf{N}_y) \otimes \mathbf{L}_v(\mathbf{N}_y)$$

The only conceivable BLL formula for this function is $\mathbf{L}_x(\mathbf{N}_y) \multimap \mathbf{L}_x(\mathbf{N}_y) \otimes \mathbf{L}_x(\mathbf{N}_y)$. In LFPL and in QBAL we can compose the split function with "append" yielding a function of type $\mathbf{L}_x(\mathbf{N}_y) \multimap \mathbf{L}_x(\mathbf{N}_y)$ that can be iterated. In BLL this composition receives the type $\mathbf{L}_x(\mathbf{N}_y) \multimap \mathbf{L}_{2x}(\mathbf{N}_y)$ which of course is not allowed in an iteration. But a hypothetical compositional embedding of LFPL into BLL would have to be able to mimic this construction.

7 Embedding RRW

Ramified recurrence on words (RRW) is a function algebra extensionally corresponding to polynomial time functions introduced by Leivant in the early nineties [14]. Bellantoni and Cook's algebra BC can be easily embedded into RRW.

Given a word algebra $\mathbb{W}$, id is the identity function on $\mathbb{W}$ and π_i^m are the m-ary projection functions from m-tuples of terms into terms. Functions on $\mathbb{W}$ can be defined by composition, primitive recursion, and conditional selection. The expression $\mathsf{rec}(f_{c_{\mathbb{W}}^1}, \ldots, f_{c_{\mathbb{W}}^w}, f_{\varepsilon_{\mathbb{W}}})$ denotes the function obtained from $f_{c_{\mathbb{W}}^1}, \ldots, f_{c_{\mathbb{W}}^w}, f_{\varepsilon_{\mathbb{W}}}$ by primitive recursion. Similarly for $\mathsf{comp}(g, f_1, \ldots, f_n)$ and $\mathsf{cond}(f_{c_{\mathbb{W}}^1}, \ldots, f_{c_{\mathbb{W}}^w}, f_{\varepsilon_{\mathbb{W}}})$. Not every function obtained this way is in RRW: indeed, they correspond to primitive recursive functions on $\mathbb{W}$. In Figure 3, a formal system for judgements in the form $\vdash f : \mathbb{W}^{i_1} \times \ldots \times \mathbb{W}^{i_n} \to \mathbb{W}^i$ (where $i_1, \ldots, i_n, i$ are natural numbers) is defined. If such a judgement can be derived from the rules in Figure 3, then f is said to be an RRW function (the definition

$$\frac{}{\vdash \mathsf{id} : \mathbb{W}^i \to \mathbb{W}^i} \quad \frac{}{\vdash \varepsilon_{\mathbb{W}} : \mathbb{W}^i} \quad \frac{}{\vdash c_{\mathbb{W}} : \mathbb{W}^i \to \mathbb{W}^i} \quad \frac{j_i = j}{\vdash \pi_i^m : \mathbb{W}^{j_1} \times \ldots \times \mathbb{W}^{j_m} \to \mathbb{W}^j}$$

$$\frac{\vdash g : \mathbb{W}^{i_1} \times \ldots \times \mathbb{W}^{i_n} \to \mathbb{W}^i \quad \vdash f_k : \mathbb{W}^{j_1} \times \ldots \times \mathbb{W}^{j_m} \to \mathbb{W}^{i_k}}{\vdash \mathsf{comp}(g, f_1, \ldots, f_n) : \mathbb{W}^{j_1} \times \ldots \mathbb{W}^{j_m} \to \mathbb{W}^i}$$

$$\frac{\vdash f_{c_{\mathbb{W}}} : \mathbb{W}^{i_1} \times \ldots \times \mathbb{W}^{i_n} \times \mathbb{W}^i \times \mathbb{W}^j \to \mathbb{W}^i \quad \vdash f_\varepsilon : \mathbb{W}^{i_1} \times \ldots \times \mathbb{W}^{i_n} \to \mathbb{W}^i \quad i < j}{\vdash \mathsf{rec}(f_{c^1_{\mathbb{W}}}, \ldots, f_{c^w_{\mathbb{W}}}, f_{\varepsilon_{\mathbb{W}}}) : \mathbb{W}^{i_1} \times \ldots \times \mathbb{W}^{i_n} \times \mathbb{W}^j \to \mathbb{W}^i}$$

$$\frac{\vdash f_{c_{\mathbb{W}}} : \mathbb{W}^{i_1} \times \ldots \times \mathbb{W}^{i_n} \times \mathbb{W}^j \to \mathbb{W}^i \quad \vdash f_\varepsilon : \mathbb{W}^{i_1} \times \ldots \times \mathbb{W}^{i_n} \to \mathbb{W}^i}{\vdash \mathsf{cond}(f_{c^1_{\mathbb{W}}}, \ldots, f_{c^w_{\mathbb{W}}}, f_{\varepsilon_{\mathbb{W}}}) : \mathbb{W}^{i_1} \times \ldots \times \mathbb{W}^{i_n} \times \mathbb{W}^j \to \mathbb{W}^i}$$

Fig. 3. RRW as a formal system

of RRW given here is slightly different but essentially equivalent to the original one [14]). Leivant [14] proved that RRW functions are exactly the polytime computable functions on $\mathbb{W}$. But RRW can be compositionally embedded into QBAL, at least in a weak sense:

Theorem 3. *RRW can be embedded into QBAL. Suppose, in other words, that* $\pi :\vdash f : \mathbb{W}^{i_1} \times \ldots \times \mathbb{W}^{i_n} \to \mathbb{W}^i$ *and that* $i < i_{j_1}, \ldots, i_{j_m}$*, while* $i = i_{k_1}, \ldots, i_{k_h}$*. Then, there exists an QBAL proof*

$$\langle \pi \rangle : \mathbf{W}_{x_1}, \ldots, \mathbf{W}_{x_n} \vdash_{\{x_{k_1} \leq x, \ldots, x_{k_h} \leq x\}} \mathbf{W}_{q(x_{j_1}, \ldots, x_{j_m}) + x}$$

where $(\prod_{i=1}^n \psi_{\mathbb{W}}) \circ [\![\pi]\!] \circ \varphi_{\mathbb{W}} = f$*. Moreover* $\langle \cdot \rangle$ *is weakly compositional.*

Quite interestingly, the proof of Theorem 3 (see [4]) is very similar in structure to the proof of polynomial time soundness for BC given in [3], which is based on the following observation: the size of the output of a BC function is bounded by a polynomial on the sizes of normal arguments plus the *maximum* of sizes of safe arguments. This cannot be formalized in BLL, because the resource polynomials do not include any function computing the maximum of its arguments. On the other hand, this can be captured in QBAL by way of constraints.

8 Conclusions

We presented QBAL, a new ICC system embedding both known systems of impredicative recursion in the sense of [9]. QBAL allows to overcome the main weakness of BLL, namely that all resource variables are global. In the authors' view, this constitutes the first step towards unifying ICC systems into a single framework. The next step consists in defining an embedding of light linear logic into QBAL and the authors are currently investigating on that.

References

1. Asperti, A., Roversi, L.: Intuitionistic light affine logic. ACM Transactions on Computational Logic 3(1), 137–175 (2002)
2. Barendregt, H.: The Lambda Calculus: Its Syntax and Semantics. In: Studies in Logic and the Foundations of Mathematics. North-Holland, Amsterdam (1984)
3. Bellantoni, S., Cook, S.: A new recursion-theoretic characterization of the polytime functions. Computational Complexity 2, 97–110 (1992)
4. Dal Lago, U., Hofmann, M.: Bounded linear logic, revisited. Extended Version (2009), `http://arxiv.org/abs/0904.2675`
5. Danos, V., Joinet, J.-B.: Linear logic and elementary time. Information and Computation 183(1), 123–137 (2003)
6. Girard, J.-Y.: Linear logic. Theoretical Computer Science 50, 1–102 (1987)
7. Girard, J.-Y.: Light linear logic. Information and Computation 143(2), 175–204 (1998)
8. Girard, J.-Y., Scedrov, A., Scott, P.: Bounded linear logic: A modular approach to polynomial-time computability. Theoretical Computer Science 97(1), 1–66 (1992)
9. Hofmann, M.: Programming languages capturing complexity classes. SIGACT News Logic Column 9, 12 (2000)
10. Hofmann, M.: Linear types and non-size-increasing polynomial time computation. Information and Computation 183(1), 57–85 (2003)
11. Hofmann, M., Scott, P.: Realizability models for BLL-like languages. Theoretical Computer Science 318(1-2), 121–137 (2004)
12. Lafont, Y.: Soft linear logic and polynomial time. Theoretical Computer Science 318(1-2), 163–180 (2004)
13. Leivant, D.: A foundational delineation of computational feasiblity. In: Sixth IEEE Symposium on Logic in Computer Science, Proceedings, pp. 2–11 (1991)
14. Leivant, D.: Stratified functional programs and computational complexity. In: 20th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Proceedings, pp. 325–333 (1993)
15. Marion, J.-Y., Moyen, J.-Y.: Efficient first order functional program interpreter with time bound certifications. In: Parigot, M., Voronkov, A. (eds.) LPAR 2000. LNCS, vol. 1955, pp. 25–42. Springer, Heidelberg (2000)
16. Murawski, A., Ong, L.: On an interpretation of safe recursion in light affine logic. Theoretical Computer Science 318(1-2), 197–223 (2004)
17. Schöpp, U.: Stratified bounded affine logic for logarithmic space. In: 22nd IEEE Symposium on Logic in Computer Science, Proceedings, pp. 411–420 (2007)
18. Troelstra, A., Schwichtenberg, H.: Basic proof theory. Cambridge University Press, Cambridge (1996)
19. Wadsworth, C.: Some unusual λ-calculus numeral systems. In: Seldin, J.P., Hindley, J.R. (eds.) To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism. Academic Press, London (1980)