The Bounded Linear Calculus: A characterization of the class of polynomial-time computable functions based on bounded linear logic

by

François Pitt

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Abstract

The Bounded Linear Calculus:
A characterization of the class of
polynomial-time computable functions
based on bounded linear logic.

François Pitt
Master of Science
Department of Computer Science
University of Toronto
1994

Recently, Girard, Scedrov, and Scott gave a logical characterization of the class of polynomial time computable functions by modifying the system of intuitionistic linear logic. They added explicit bounds on the number of times a formula can be reused in a logical derivation, and obtained an intuitionistic system of "bounded linear logic".

Independently of this, a number of researchers have extended the Curry-Howard isomorphism (which establishes a correspondence between natural deduction and simply typed lambda calculus) to linear logic, obtaining different "linear calculi".

This work adapts a system of linear calculus to bounded linear logic, thus providing a functional framework in which exactly the polynomial-time computable functions can be represented. Also, the extension of a type reconstruction algorithm to our "bounded linear calculus" is discussed, and the computational procedure to be used on bounded linear terms is defined explicitly. Examples of bounded linear terms, and of computations on those terms, are provided.

Acknowledgments

I would like to thank Steve Cook, my advisor, for many helpful discussions and suggestions, as well as for providing me with many good references and putting me in touch with Philip Scott. I also thank him for bearing with me even when what I was doing was not very clear to him: without his trust and patience, this work would not have been possible. My thanks go also to Alasdair Urquhart for reading my thesis and for his several suggestions which helped improve the text substantially, and very special thanks go to Philip Scott, who helped me considerably with many technical details and without whom I would still be trying to puzzle out some subtleties for myself.

Thank you also to Ian Mackie, who provided me with a copy of his thesis, and to the staff at the library of the department of computer science who helped me with some of the more difficult to find references. Finally, Kathy Yen deserves my gratitude, for without her patience in answering my numerous questions, I would never have found my way out of the administrative labyrinth inherent in a work of this sort.

Contents

List of Figures

Chapter 1

Introduction

It is common knowledge that intuitionistic logic can be given a computational content via the Curry-Howard isomorphism, which equates reduction of lambda terms with normalization of natural deduction proofs. This relationship highlights the constructive nature of intuitionistic logic and certain logical issues related to the simply typed lambda calculus (for a good account of these matters, see Girard *et al.* 1988).

With the advent of linear logic, which promised to bring a new perspective to issues of parallelism, control of resource, and order of evaluation, it was only natural to try to extend the Curry-Howard isomorphism, a task which was performed by a number of people in slightly different ways, highlighting different aspects of the correspondence. Those systems were all shown to have interesting properties, such as being Church-Rosser, having a sound and complete type reconstruction algorithm, possessing a natural categorical model, etc., and some of them were even implemented in one way or another.

Linear logic imposes a stricter control on resources than intuitionistic logic by discarding, in the sequent calculus formulation of the logical system, the weakening and contraction rules. These rules are reintroduced through the storage (or "of course") connective, !, in a controlled fashion so that a formula of the form $!A$ (a "stored" formula) can be reused as often as wanted in a proof while all other formulas must be used exactly once (see Girard 1987 for more details).

Recently, a new refinement to linear logic was introduced by putting an explicit bound on the number of times a stored formula can be reused. The resulting system (bounded linear logic, or BLL for short) was shown to have one extremely interesting property: the class of functions representable in the system is exactly the class of polynomial time computable functions (Girard *et al.* 1992).

This remarkable result is the motivation behind the present work, which extends the linear term calculi developed for linear logic to a "bounded linear calculus" corresponding to bounded linear logic. With this new tool, it is expected that a closer investigation of some important properties of BLL will become possible.

The thesis is organized as follows. Chapter 2 presents an overview of the paper on BLL, along with the main results of that paper. In Chapter 3, a linear term system for intuitionistic linear logic is given, along with some of its theoretical properties. The system of the bounded linear calculus ("BLC" for short) is introduced in Chapter 4, and its properties are discussed in Chapter 5, along with other theoretical results.

Chapter 2

Bounded Linear Logic

In this chapter, the system of Bounded Linear Logic (BLL) will be described, as introduced by Girard, Scedrov, and Scott in their 1992 paper. The system is based on the multiplicative fragment of intuitionistic linear logic, which will be reviewed in the first section.

2.1 Intuitionistic Linear Logic

This section will describe the multiplicative fragment of intuitionistic linear logic without unity (ILL), which is based on ordinary intuitionistic logic.

Recall that Gentzen's sequent calculus for intuitionistic logic involves two structural rules which are problematic from the point of view of management of resources: weakening and contraction (in particular, contraction causes cut-elimination to be a hyper-exponential procedure). In ILL, these rules are replaced by logical rules for a new "storage" connective, !.

2.1.1 Formulas

Formulas are defined inductively, as follows:

1. Propositional variables $(\alpha, \beta, \ldots)$ are *atomic formulas*.

2. If A and B are formulas, then so are (in order of precedence of the connective)

(a) $!A$ (*storage* A),

(b) $A \otimes B$ (A *tensor* B: linear multiplicative conjunction),

(c) $A \multimap B$ (A *entails* B: linear implication; right associative), and

(d) $(\forall\alpha)A$ (*second-order quantification* of A).

For any ILL formula T, we denote by $A[\alpha := T]$ the result of substituting T for all free occurrences of the propositional variable α in A.

2.1.2 Sequents

Sequents are expressions of the form $\Gamma \vdash A$, where Γ is a finite multiset of formulas and A is a formula. A proof system for ILL based on sequents, the *sequent calculus*, is described by the following axiom scheme and rules (where $!\Gamma$ means that each formula in Γ is of the form $!B$) (Figure 2.1).

$$
\begin{array}{lllll}
\textit{Axiom} & A \vdash A & & & \\[1ex]
\textit{Cut} & \dfrac{\Gamma \vdash A \quad \Delta, A \vdash B}{\Gamma, \Delta \vdash B} & & & \\[2ex]
\textit{Logical} & \otimes L \quad \dfrac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C} & & \otimes R \quad \dfrac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B} & \\[2ex]
 & \multimap L \quad \dfrac{\Gamma \vdash A \quad \Delta, B \vdash C}{\Gamma, \Delta, A \multimap B \vdash C} & & \multimap R \quad \dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B} & \\[2ex]
\textit{Quantifier} & \forall L \quad \dfrac{\Gamma, A[\alpha := T] \vdash B}{\Gamma, (\forall\alpha)A \vdash B} & & \forall R \quad \dfrac{\Gamma \vdash A}{\Gamma \vdash (\forall\alpha)A} & (\alpha \text{ not free in } \Gamma) \\[2ex]
\textit{Storage} & \textit{Weakening} \quad \dfrac{\Gamma \vdash B}{\Gamma, !A \vdash B} & & \textit{Dereliction} \quad \dfrac{\Gamma, A \vdash B}{\Gamma, !A \vdash B} & \\[2ex]
 & \textit{Contraction} \quad \dfrac{\Gamma, !A, !A \vdash B}{\Gamma, !A \vdash B} & & \textit{Promotion} \quad \dfrac{!\Gamma \vdash A}{!\Gamma \vdash !A} &
\end{array}
$$

Figure 2.1: Sequent Calculus for ILL

A *proof* of a sequent $\Gamma \vdash A$ in the ILL sequent calculus is a finite rooted tree with nodes labeled by ILL sequents, such that the root of the tree is labeled with

$\Gamma \vdash A$, the leaves are labeled with instances of the axiom, and the label of each node is obtained from the label(s) of its immediate predecessor(s) by an instance of an ILL rule.

As was shown in Girard (1987), the expressive power of ILL is equivalent to that of system $\mathcal{F}$ (*i.e.*, second-order polymorphic lambda calculus, described in Girard *et al.* 1988). Also, by discarding the storage rules, we obtain a system which enjoys polynomial-time normalization (cut-elimination), although its expressive power is very weak: not even all feasible functions can be represented.

2.2 The Syntax of BLL

As we have seen, second-order intuitionistic logic without the weakening or contraction rules (equivalent to ILL without the storage rules) is a system with little expressive power but feasible normalization. In an attempt to regain some expressive power, linear logic introduces a new storage connective and rules for that connective which allow restricted forms of weakening and contraction. The new system is as strong as system $\mathcal{F}$ in terms of expressiveness, but does not enjoy feasible normalization. What is sought is an intermediate system, powerful enough to represent all feasible functions yet enjoying polynomial-time normalization; that system is BLL.

The exposition in Girard *et al.* (1992) will be followed closely in this section, with one minor modification: what they call "Storage rule" (S!) will be called "Promotion rule" (P!).

2.2.1 Resource Polynomials

A *resource polynomial* is any finite sum of monomials, where a *monomial* is any finite product of binomial coefficients, $\prod_{i=1}^{p} \binom{\mathbf{x}^i}{n_i}$, where the $\mathbf{x}^i$'s are distinct resource variables and the n_i's are nonnegative integer constants. Resource polynomials are closed under sum, product, and composition, and given resource polynomials P and Q, we write $P \sqsubseteq Q$ to denote that $Q - P$ is a resource polynomial and $P \equiv Q$ to denote that P and Q are identical up to renaming of their variables.

2.2.2 Formulas

Formulas are defined inductively, as follows:

1. Atomic formulas have the form $\alpha^n(\bar{P})$, where α^n is a second-order propositional variable of arity n (for $n \geq 1$) and $\bar{P}$ denotes a list of n resource polynomials.

2. If A and B are formulas, $\mathbf{x}$ is a free resource variable and P is a resource polynomial in which $\mathbf{x}$ does not occur, then the following are all formulas of BLL (in order of precedence of the connective):

   (a) $!_{\mathbf{x}<P}A$ (*bounded storage* of A),

   (b) $A \otimes B$,

   (c) $A \multimap B$, and

   (d) $(\forall \alpha)A$.

Positive and negative occurrences of resource polynomials in formulas are defined by induction, in the following way:

1. If $\bar{P} = P_1, \ldots, P_n$, then all P_i occur positively in $\alpha^n(\bar{P})$.

2. If P_1 occurs positively and P_2 occurs negatively in A, Q_1 occurs positively and Q_2 occurs negatively in B, and $\mathbf{x}$ is a free resource variable not occurring in P_1 or P_2, then

   (a) P_1 and Q_1 occur positively and P_2 and Q_2 occur negatively in $A \otimes B$,

   (b) P_1 and Q_2 occur negatively and P_2 and Q_1 occur positively in $A \multimap B$,

   (c) P_1 occurs positively and P_2 occurs negatively in $(\forall \alpha)A$,

   (d) P_1 occurs negatively and $\mathbf{x}$ is bound in $!_{\mathbf{x}<P_1}A$, and

   (e) P_2 occurs positively and $\mathbf{x}$ is bound in $!_{\mathbf{x}<P_2}A$.

Note that a bound resource variable does not occur either positively or negatively in a formula.

If B is a formula in which resource variables $\mathbf{x}^1, \ldots, \mathbf{x}^n$ occur only positively, then $\lambda \mathbf{x}^1, \ldots, \mathbf{x}^n.B$ is a second-order abstraction term, say T. Then, $A[\alpha := T]$ denotes the result of substituting T for all free occurrences of α in A, *i.e.*, of replacing the free atoms $\alpha^n(P_1, \ldots, P_n)$ by $B[\mathbf{x}^1 := P_1, \ldots, \mathbf{x}^n := P_n]$, where $A[\mathbf{x} := P]$ denotes the result of substituting resource polynomial P for all free occurrences of the free resource variable $\mathbf{x}$ in A.

Given formulas A and A', we write $A \sqsubseteq A'$ if A and A' differ only in their choice of resource polynomials and

1. for any resource polynomial P which occurs positively in A, the corresponding P' in A' is such that $P \sqsubseteq P'$,

2. for any resource polynomial P which occurs negatively in A, the corresponding P' in A' is such that $P' \sqsubseteq P$.

We extend the notation to multisets of formulas and write $\Gamma \sqsubseteq \Gamma'$ iff the relation holds componentwise.

We also define a surjective mapping "$\underline{\ }$" from formulas of BLL to formulas of ILL by simply discarding all resource information. For a multiset of formulas Γ, we denote by $\underline{\Gamma}$ the multiset of each transformed formula.

2.2.3 Sequents

The sequents of BLL have the form $\Gamma \vdash A$, where Γ is a finite (possibly empty) multiset of formulas and A is a formula. The sequent calculus for BLL is described in Figure 2.2, where P, Q, V, and W range over resource polynomials.

Proofs for the BLL sequent calculus are defined as for ILL. Note that the rules of BLL are written in such a way that, given any proof ρ of a sequent $\Gamma \vdash A$ and given any $\Gamma' \sqsubseteq \Gamma$ and $A' \sqsupseteq A$, a simple change of resource parameters yields a proof ρ' of $\Gamma' \vdash A'$ without altering the structure of the proof. Also, it is obvious that erasing all resource information from a BLL proof of $\Gamma \vdash A$ yields a proof in ILL of the corresponding sequent $\underline{\Gamma} \vdash \underline{A}$.

$$\mathsf{Id} \quad A \vdash A' \quad \text{where } A \sqsubseteq A'$$

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

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

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

$$\forall L \quad \frac{\Gamma, A[\alpha := T] \vdash B}{\Gamma, (\forall \alpha) A \vdash B} \qquad \forall R \quad \frac{\Gamma \vdash A}{\Gamma \vdash (\forall \alpha) A} \quad (\alpha \text{ not free in } \Gamma)$$

$$!W \quad \frac{\Gamma \vdash B}{\Gamma, !_{\mathbf{x}<W} A \vdash B} \qquad !D \quad \frac{\Gamma, A[\mathbf{x} := 0] \vdash B}{\Gamma, !_{\mathbf{x}<1+W} A \vdash B}$$

$$!C \quad \frac{\Gamma, !_{\mathbf{x}<P} A, !_{\mathbf{y}<Q} A[\mathbf{x} := P + \mathbf{y}] \vdash B}{\Gamma, !_{\mathbf{x}<P+Q+W} A \vdash B} \quad \text{where } P + \mathbf{y} \text{ is free for } \mathbf{x} \text{ in } A$$

$$P! \quad \frac{\ldots, !_{\mathbf{z}<Q_i(\mathbf{x})} A_i[\mathbf{y} := V_i(\mathbf{x}) + \mathbf{z}], \ldots \vdash B}{\ldots, !_{\mathbf{y}<V_i(P)+W_i} A_i, \ldots \vdash !_{\mathbf{x}<P} B}$$

where $V_i(\mathbf{x}) = \sum_{\mathbf{z}<\mathbf{x}} Q_i(\mathbf{z})$, $V_i(\mathbf{x}) + \mathbf{z}$ is free for $\mathbf{y}$ in A, and all formulas to the left of $\vdash$ have the indicated form, for $1 \leq i \leq n$

Figure 2.2: Sequent Calculus for BLL

2.2.4 Normalization

It is possible to define a normalization procedure for BLL sequents which gives rise to *irreducible forms*. These are not necessarily normal (cut-free) forms, but they allow Girard *et al.* to represent a number of common data structures in BLL in a way similar to what is done in system $\mathcal{F}$. Moreover, it is possible to define a general notion of representability of functions in BLL based on these irreducible forms (see Definition 5.2 in Girard *et al.* 1992 and Section 5.3 of the current work), and it is shown that the functions thus represented are all and only the polynomial-time computable functions (the reader is referred to Girard *et al.* 1992 for more details).

Chapter 3

Term Assignments for ILL

As Abramsky put it in the introduction to his 1993 paper,

> The paradigm followed by Girard in seeking to apply linear logic to computation is the Curry-Howard isomorphism, in which propositions (or logical formulae) are interpreted as types, proofs as programs, and the process of normalization or cut-elimination as computation. This paradigm has been a cornerstone of much recent work on the connections between intuitionistic logic, functional programming and category theory. It has by now firmly established itself as a major component of the logical foundations of programming. In the case of intuitionistic logic, it relates typed λ-calculus (*i.e.*, typed functional programs in a canonical syntax) to proofs in intuitionistic logic; and reduction of terms to normal form (*i.e.*, program execution) to normalization of proofs.

Since then, many researchers have provided a computational interpretation for intuitionistic linear logic by defining various extensions of the lambda calculus. This chapter contains a brief summary of parts of the work of Benton *et al.* (1992), who gave one such term assignment system for ILL, the *linear term calculus*. We will give a slightly modified version of the linear term calculus, using an alternate formulation of the dereliction rule which was mentioned but not used in Benton *et al.*'s paper. Terms of the linear term calculus will be referred to as *linear terms*.

3.1 Term Assignments in a Linear Context

We will use the standard notation $t : A$ to mean that term t has type A, where A is a formula of the logic under consideration, and t is a term of some (extended) lambda calculus. Note that in this work, "type" means only "formula of the logic under consideration" and has no semantic connotation (specifically, types do not correspond to families of functionals as they do in the classical Curry-Howard isomorphism).

Here, we will consider term assignment systems as "special" versions of sequent calculus or natural deduction (in sequent form) in which the objects being derived have the following form:

$$x_1 : A_1, \ldots, x_n : A_n \vdash t : A$$

where the x_i's are distinct and are all the free variables of term t. Intuitively, such a "term sequent" means that it is possible to give type A to term t under the assumptions that variable x_i has type A_i for all $1 \leq i \leq n$.

In a linear context, it is essential that a variable be used exactly once in a term except when the variable has type $!A$ for some linear formula A, thus reflecting the linearity of the underlying logic. This *linearity constraint* can always be met by requiring that each time a new variable is introduced by a rule (*e.g.*, in the *Axiom*, the left logical rules, and the four storage rules), the variable be new to the proof. We will implicitly assume that this is done in all that follows.

3.2 ILL Sequent Calculus

The first term assignment system is based on the sequent calculus formulation of ILL (Figure 3.1). The authors obtain the assignment from categorical considerations and use "naturality conditions" to derive a few equality relations between terms.

Here, I have omitted the author's rules for unity, added my own rules for the quantifiers, slightly modified the form of the promotion rule, and used the alternate formulation of the dereliction rule, but it is obvious that the system is basically the same and that all the results proven in Benton *et al.* (1992) still apply.

$$\textit{Axiom} \quad x : A \vdash x : A$$

$$\textit{Cut} \quad \frac{\Gamma \vdash e : A \quad \Delta, x : A \vdash f : B}{\Gamma, \Delta \vdash f[e/x] : B}$$

$$\otimes L \quad \frac{\Gamma, x : A, y : B \vdash f : C}{\Gamma, z : A \otimes B \vdash \texttt{let}\ z\ \texttt{be}\ x \otimes y\ \texttt{in}\ f : C} \qquad \otimes R \quad \frac{\Gamma \vdash e : A \quad \Delta \vdash f : B}{\Gamma, \Delta \vdash e \otimes f : A \otimes B}$$

$$\multimap L \quad \frac{\Gamma \vdash e : A \quad \Delta, x : B \vdash f : C}{\Gamma, \Delta, g : A \multimap B \vdash f[(ge)/x] : C} \qquad \multimap R \quad \frac{\Gamma, x : A \vdash e : B}{\Gamma \vdash \lambda x.e : A \multimap B}$$

$$\forall L \quad \frac{\Gamma, x : A[\alpha := T] \vdash e : B}{\Gamma, x : (\forall \alpha) A \vdash e : B} \qquad \forall R \quad \frac{\Gamma \vdash e : A}{\Gamma \vdash e : (\forall \alpha) A}\ (\alpha \text{ not free in } \Gamma)$$

$$\textit{Weakening} \quad \frac{\Gamma \vdash e : B}{\Gamma, z : !A \vdash \texttt{discard}\ z\ \texttt{in}\ e : B}$$

$$\textit{Dereliction} \quad \frac{\Gamma, x : A \vdash e : B}{\Gamma, z : !A \vdash \texttt{derelict}\ z\ \texttt{for}\ x\ \texttt{in}\ e : B}$$

$$\textit{Contraction} \quad \frac{\Gamma, x : !A, y : !A \vdash e : B}{\Gamma, z : !A \vdash \texttt{copy}\ z\ \texttt{as}\ x, y\ \texttt{in}\ e : B}$$

$$\textit{Promotion} \quad \frac{\bar{x} : !\Gamma \vdash e : A}{\bar{y} : !\Gamma \vdash \texttt{promote}\ \bar{y}\ \texttt{for}\ \bar{x}\ \texttt{in}\ !e : !A}$$

Figure 3.1: Term Assignment System for ILL Sequent Calculus

3.3 Linear Natural Deduction

The second term assignment system presented is based on a natural deduction formulation of ILL (Figure 3.2), and is obtained by extending the Curry-Howard isomorphism. The authors prove a few results, including the "substitution property": if $\Gamma \vdash a : A$ and $\Delta, x : A \vdash b : B$, then $\Gamma, \Delta \vdash b[a/x] : B$.

As before, I have omitted the author's rules for unity, added my own rules for the quantifiers, slightly modified the form of the promotion rule, and used the alternate formulation of the dereliction rule, but it is still obvious that the systems are basically the same and that all the authors' results apply.

$$\textit{Axiom} \quad x : A \vdash x : A$$

$$\otimes_{\mathcal{I}} \; \frac{\Gamma \vdash e : A \quad \Delta \vdash f : B}{\Gamma, \Delta \vdash e \otimes f : A \otimes B} \qquad \otimes_{\mathcal{E}} \; \frac{\Gamma \vdash e : A \otimes B \quad \Delta, x : A, y : B \vdash f : C}{\Gamma, \Delta \vdash \texttt{let}\ e\ \texttt{be}\ x \otimes y\ \texttt{in}\ f : C}$$

$$\multimap_{\mathcal{I}} \; \frac{\Gamma, x : A \vdash e : B}{\Gamma \vdash \lambda x.e : A \multimap B} \qquad \multimap_{\mathcal{E}} \; \frac{\Gamma \vdash e : A \multimap B \quad \Delta \vdash f : A}{\Gamma, \Delta \vdash ef : B}$$

$$\forall_{\mathcal{I}} \; \frac{\Gamma \vdash e : A}{\Gamma \vdash e : (\forall \alpha) A} \qquad \forall_{\mathcal{E}} \; \frac{\Gamma \vdash e : (\forall \alpha) A}{\Gamma \vdash e : A[\alpha := T]}$$

$(\alpha$ not free in $\Gamma)$

$$\textit{Weakening} \; \frac{\Gamma \vdash e : !A \quad \Delta \vdash f : B}{\Gamma, \Delta \vdash \texttt{discard}\ e\ \texttt{in}\ f : B}$$

$$\textit{Dereliction} \; \frac{\Gamma \vdash e : !A \quad \Delta, x : A \vdash f : B}{\Gamma \vdash \texttt{derelict}\ e\ \texttt{for}\ x\ \texttt{in}\ f : B}$$

$$\textit{Contraction} \; \frac{\Gamma \vdash e : !A \quad \Delta, x : !A, y : !A \vdash f : B}{\Gamma, \Delta \vdash \texttt{copy}\ e\ \texttt{as}\ x, y\ \texttt{in}\ f : B}$$

$$\textit{Promotion} \; \frac{\Delta_1 \vdash e_1 : !A_1 \quad \cdots \quad \Delta_n \vdash e_n : !A_n \quad x_1 : !A_1, \ldots, x_n : !A_n \vdash f : B}{\Delta_1, \ldots, \Delta_n \vdash \texttt{promote}\ e_1, \ldots, e_n\ \texttt{for}\ x_1, \ldots, x_n\ \texttt{in}\ !f : !B}$$

Figure 3.2: Term Assignment System for Linear Natural Deduction

3.4 Properties of the Linear Term Calculus

Benton *et al.* prove a number of properties of their system, but the only one of importance to us besides the substitution property is the relationship between the two term assignment systems. In fact, they are equivalent, *i.e.*, there exist transformations from natural deduction derivations to sequent calculus derivations and vice versa which preserve the term assignment (Theorem 4 in Benton *et al.* 1992, which still holds with the alternate dereliction and the quantifier rules.) It is also shown in that paper that if a term t can be given type A in the natural deduction system, then the derivation of $\Gamma \vdash t : A$ is unique, but this does not hold in our system because of the presence of the quantifier rules. Nevertheless, it is easy to show a weaker result: if a term t can be given a type A which contains no quantifiers, then there exists a unique derivation of $\Gamma \vdash t : A$ which does not use the quantifier rules.

Chapter 4

The Bounded Linear Calculus

Conceptually, the passage from ILL to BLL is a simple one: simply add explicit bounds on the number of times that a stored formula can be reused. In practice, as we have seen, it is necessary to introduce the concept of resource polynomials and to add explicit resource information to each formula in order to ensure that the new system is consistent and has the properties we seek.

Similarly, going from the linear term calculus to the bounded linear calculus is a simple conceptual step: simply add explicit bounds on the number of times certain variables can be reused. However, the technical apparatus required to achieve this simple conceptual step is far from trivial, as will be seen in the pages that follow. It will be necessary to add explicit resource information to each term and variable in a consistent way and ensure that the rules of term formation respect some essential constraints on the resources.

In this chapter, the syntax of bounded linear terms will be described, as well as their type assignment system based on BLL. The reduction procedure defined on those terms will also be described. Throughout the exposition, I will attempt to provide an intuitive motivation for the syntactic details, and many examples will be given at the end of the chapter, which the reader should refer to for a better understanding of certain subtleties.

4.1 Resource Variables

There are two types of variables in the BLC: *object variables* $(x_0, x_1, x_2, \ldots)$ and *resource variables* $(\mathbf{x}^0, \mathbf{x}^1, \mathbf{x}^2, \ldots)$. Free resource variables have the form $\mathbf{x}^i$ while bound resource variables have the form $\mathbf{x}^i_{P_i}$, where P_i is a resource polynomial in which $\mathbf{x}^i$ does not occur and is called the *bounding polynomial* of $\mathbf{x}^i$.

The order relation $\sqsubseteq$ on resource polynomials is extended to bound resource variables so that $\mathbf{x}^i_{P_i} \sqsubseteq \mathbf{x}^j_{P_j} \iff P_i \sqsupseteq P_j$. We also extend $\sqsubseteq$ to sets of resource variables, component-wise. The equivalence relation, $\equiv$, is redefined for bound resource variables so that $\mathbf{x}^i_{P_i} \equiv \mathbf{x}^j_{P_j}$ iff $P_i = P_j$ (syntactically), *i.e.*, only the name of the bound resource variable may change. We denote by $\mathcal{V}(P)$ the set of all resource variables occurring in resource polynomial P.

4.2 Bounded Linear Terms

Bounded linear terms have the same basic syntax as linear terms but each one is annotated with two sets of resource variables (bound and free, respectively) so that a bounded linear term has the form $t\langle\bar{\mathbf{y}};\bar{\mathbf{x}}\rangle$, where $\bar{\mathbf{y}}\cap\bar{\mathbf{x}} = \varnothing$. For the sake of readability, we will make two slight abuses of notation: we will write $\bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1$ to mean $\bar{\mathbf{x}}_0\cup\bar{\mathbf{x}}_1$, and we will identify a resource variable with the singleton set containing only that variable. For convenience, the sets of resource variables will be omitted when they are empty or when they are not needed in the context in which a variable is used. Also, it will be convenient to only indicate explicitly the resource variables which are relevant in a given context, so that $t\langle\mathbf{y}^i_{P_i}\rangle$ could stand for $t\langle\mathbf{y}^i_{P_i}, \bar{\mathbf{y}}_i; \bar{\mathbf{x}}_i\rangle$, for example. Finally, in some contexts, we will want to indicate that a certain rule applies irrespective of the presence or absence of certain resource variable(s). We will indicate this by enclosing the relevant resource variable(s) in brackets ().

In the inductive definition of bounded linear terms which follows, we let P, Q, and R range over resource polynomials, and we refer to bounded linear terms as simply "terms", for conciseness.

1. If x_i is any object variable, $\bar{\mathbf{y}}$ is any set of bound resource variables ($\bar{\mathbf{y}} = \{\mathbf{y}^j_{P_j}\}_{j\in J}$ for some set of indices J), and $\bar{\mathbf{x}}$ is any set of free resource variables such that $\bar{\mathbf{y}} \cap \bar{\mathbf{x}} = \varnothing$ and $(\bigcup_{j\in J} \mathcal{V}(P_j) \backslash \bar{\mathbf{y}}) \subseteq \bar{\mathbf{x}}$, then

$$x_i\langle \bar{\mathbf{y}}_i; \bar{\mathbf{x}}_i \rangle$$

   is a *term variable* (or "variable" for short).

   Intuitively, the resource information associated with each bounded linear term is used to decide which terms may or may not be combined together and in what way(s). The free resource variables of a term variable represent "unused" resources; resources on which no constraints have been put. If the list of bound resource variables is empty, then the term variable will be used exactly once in the term in which it appears—it is linear; otherwise, the term variable can be reused and each bound resource variable's bounding polynomial represents the number of times the term variable will be used.

   For example, if $\lambda x\langle \mathbf{x} \rangle.t$ is a bounded linear term, then we know the term variable $x\langle \mathbf{x} \rangle$ occurs exactly once in t while if $\lambda f\langle \mathbf{x}_3 \rangle.t$ is a term, then f will be used three times in t (the examples at the end of the chapter should help make this clearer).

2. If $t\langle \bar{\mathbf{y}}_0; \bar{\mathbf{x}}_0 \rangle$ and $u\langle \bar{\mathbf{y}}_1; \bar{\mathbf{x}}_1 \rangle$ are terms with no bound resource variables and no free object variables in common, then

$$(t \otimes u)\langle \bar{\mathbf{y}}_0, \bar{\mathbf{y}}_1; \bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1 \rangle$$

   is a term. Intuitively, " $\_ \otimes \_$ " is a pairing operator.

3. If $t\langle \bar{\mathbf{y}}_1, (\bar{\mathbf{y}}_0); \bar{\mathbf{x}}_1, (\bar{\mathbf{x}}_0) \rangle$ is a term with free object variable $x\langle \bar{\mathbf{y}}_0; \bar{\mathbf{x}}_0 \rangle$, then

$$(\lambda x.t)\langle \bar{\mathbf{y}}_0, \bar{\mathbf{y}}_1; \bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1 \rangle$$

   is a term in which x is bound. Intuitively, "$\lambda\ \_\ .\ \_$ " is a function-forming operator, just as in ordinary λ-calculus.

4. If $t\langle\bar{\mathbf{y}};\bar{\mathbf{x}}\rangle$ is a term with free object variables of the form $x\langle\bar{\mathbf{y}}_0;\bar{\mathbf{x}}_0\rangle$ and $y\langle\bar{\mathbf{y}}_1;\bar{\mathbf{x}}_1\rangle$, $u\langle\bar{\mathbf{y}}'_0,\bar{\mathbf{y}}'_1;\bar{\mathbf{x}}_0,\bar{\mathbf{x}}_1\rangle$ is a term with no free object variables in common with t, except maybe x and y, and $\bar{\mathbf{y}}'_0 \equiv \bar{\mathbf{y}}_0$, $\bar{\mathbf{y}}'_1 \equiv \bar{\mathbf{y}}_1$, then

$$(\texttt{let } u \texttt{ be } x \otimes y \texttt{ in } t)\langle\bar{\mathbf{y}};\bar{\mathbf{x}}\rangle$$

is a term in which x and y are bound. Intuitively, "`let _ be _` $\otimes$ `_ in _`" is a projection operator which uses both members of a pair at the same time.

5. If $t\langle\bar{\mathbf{y}}_0,\bar{\mathbf{y}}_1;\bar{\mathbf{x}}_0,\bar{\mathbf{x}}_1\rangle$ and $u\langle\bar{\mathbf{y}}'_0;\bar{\mathbf{x}}_0\rangle$ are terms with no free object variables in common, and $\bar{\mathbf{y}}'_0 \equiv \bar{\mathbf{y}}_0$, then

$$(t\ u)\langle\bar{\mathbf{y}}_1;\bar{\mathbf{x}}_1\rangle$$

is a term. Intuitively, juxtaposition is function application (associative to the left, *i.e.*, $t\ u\ v = (t\ u)\ v$).

6. If $t\langle\bar{\mathbf{y}};\bar{\mathbf{x}}\rangle$ and $u\langle\mathbf{y}_R,\bar{\mathbf{y}}';\bar{\mathbf{x}}',\mathcal{V}(R)\rangle$ are terms (where $\bar{\mathbf{x}}'$ may contain some or all of $\mathcal{V}(R)$) with no bound resource variables and no free object variables in common, then

$$(\texttt{discard } u \texttt{ in } t)\langle\bar{\mathbf{y}};\bar{\mathbf{x}}\rangle$$

is a term. Intuitively, discarding is a way of using the resources in a term when not all of them are needed.

7. If $t\langle\bar{\mathbf{y}};\bar{\mathbf{x}}\rangle$ is a term with free object variable $x\langle\bar{\mathbf{y}}_0;\bar{\mathbf{x}}_0\rangle$ (where $\bar{\mathbf{x}}_0$ may contain some or all of $\mathcal{V}(R)$), $u\langle\mathbf{y}_{1+R},\bar{\mathbf{y}}_1;\bar{\mathbf{x}}_0,\mathcal{V}(R)\rangle$ is a term with no free object variables in common with t, except maybe x, and $\bar{\mathbf{y}}_1 \equiv \bar{\mathbf{y}}_0$, then

$$(\texttt{derelict } u \texttt{ for } x \texttt{ in } t)\langle\bar{\mathbf{y}};\bar{\mathbf{x}}\rangle$$

is a term in which x is bound. Intuitively, dereliction is used to get rid of bound resources when a term is going to be used in a linear fashion.

8. If $t\langle\bar{\mathbf{y}};\bar{\mathbf{x}}\rangle$ is a term with free object variables of the form $x\langle\mathbf{x}^0_P,\bar{\mathbf{y}}_0;\bar{\mathbf{x}}',\mathcal{V}(P)\rangle$ and $y\langle\mathbf{x}^1_Q,\bar{\mathbf{y}}_1;\bar{\mathbf{x}}',\mathcal{V}(Q),(\mathcal{V}(P))\rangle$ (where $\bar{\mathbf{x}}'$ may contain some or all of $\mathcal{V}(P)$, $\mathcal{V}(Q)$, or

$\mathcal{V}(R))$, $u\langle \mathbf{x}^2_{P+Q+R}, \bar{\mathbf{y}}'; \bar{\mathbf{x}}', \mathcal{V}(P+Q+R)\rangle$ is a term with no free object variables in common with t, except maybe x and y, and $\bar{\mathbf{y}}' \equiv \bar{\mathbf{y}}_0 \equiv \bar{\mathbf{y}}_1$, then

$$(\texttt{copy}\ u\ \texttt{as}\ x, y\ \texttt{in}\ t)\langle \bar{\mathbf{y}}; \bar{\mathbf{x}}\rangle$$

is a term in which x and y are bound. Intuitively, copying allows a term to be used more than once.

9. If $t\langle \bar{\mathbf{y}}; \bar{\mathbf{x}}, (\mathbf{x})\rangle$ is a term (where $\bar{\mathbf{x}}$ may contain some or all of $\mathcal{V}(P)$) with free object variables $x_i\langle \mathbf{w}^i_{Q_i(\mathbf{x})}, \bar{\mathbf{y}}_i; \bar{\mathbf{x}}_i, (\mathbf{x})\rangle$ (where $\bar{\mathbf{x}}_i$ may contain some or all of $\mathcal{V}(R_i)$ or $\mathcal{V}(P)$) for $1 \le i \le n$ and $u_i\langle \mathbf{y}^i_{V_i(P)+R_i}, \bar{\mathbf{y}}_i; \bar{\mathbf{x}}_i, \mathcal{V}(R_i), (\mathcal{V}(P))\rangle$ is a term (where $V_i(\mathbf{x}) = \sum_{\mathbf{z}<\mathbf{x}} Q_i(\mathbf{z})$ and $\mathcal{V}(P)$ appears iff $V_i(\mathbf{x})$ is not constant) with no free object variables in common with t, except maybe x_i, for $1 \le i \le n$, then

$$(\texttt{promote}\ldots, u_i, \ldots\texttt{for}\ldots, x_i, \ldots\texttt{in}\ !_{\mathbf{x}<P}t)\langle \mathbf{x}_P, \bar{\mathbf{y}}; \bar{\mathbf{x}}, \mathcal{V}(P)\rangle$$

is a term in which x_i is bound for $1 \le i \le n$. If $n = 0$, then the term has the form $!_{\mathbf{x}<P}t$. Such terms will be referred to as *promoted* terms in what follows. Intuitively, a promoted term is one which can be reused, and a term can get promoted only if all its free object variables can be reused themselves.

Note that erasing all resource information from a bounded linear term t yields a linear term which will be denoted $\underline{t}$.

4.3 A Term Assignment for BLL

By extending the term assignment system for ILL given in Figure 3.1 to the sequent calculus for BLL (Figure 2.2), we obtain a term assignment system for BLL (Figure 4.1). Here, the linearity constraint which was defined in Chapter 3 is extended to include bound resource variables, *i.e.*, when a new term variable is introduced by a rule, the bound resource variables introduced with the term variable have to be new as well.

Note that this extension is *conservative* with respect to "_", so that if we are given a BLC term assignment

$$x_1 : A_1, \ldots, x_n : A_n \vdash t : A,$$

then

$$\underline{x_1} : \underline{A_1}, \ldots, \underline{x_n} : \underline{A_n} \vdash \underline{t} : \underline{A}$$

is a valid term assignment for ILL.

4.4 Reduction

The reduction procedure (cut-elimination) for BLL induces a similar reduction relation on bounded linear terms, denoted by " $\triangleright$ " (Figures 4.2 and 4.3; we let x, y, z, w range over term variables and t, u, a, b, c range over terms). The quantifier reductions have been omitted (they have no effect on terms) and none of these steps apply to a promoted term or to a subterm of a promoted term. We will use the symbol " $\rightsquigarrow$ " to denote the transitive closure of " $\triangleright$ ".

A term is called *irreducible* if none of the reduction steps can be applied to the term or to any of its subterms.

In order for the reduction relation to properly deal with the resource information contained in the terms, it has been necessary to allow a slight abuse of notation: given a promoted term, the free resource variable $\mathbf{x}$ which is bound by the promotion can be annotated with a *base polynomial*: a resource polynomial in which $\mathbf{x}$ does not occur. This is denoted in the list of variables by appending the base polynomial to the variable, following the symbol "$\circledast$" (see the reduction rules **SP!C** and **SP!D**).

$$\mathsf{Id} \quad x\langle \bar{\mathbf{y}}; \bar{\mathbf{x}} \rangle : A \vdash x\langle \bar{\mathbf{y}}'; \bar{\mathbf{x}}' \rangle : A' \quad (A \sqsubseteq A' \ , \ \bar{\mathbf{y}} \sqsubseteq \bar{\mathbf{y}}')$$

where $\bar{\mathbf{y}}$ (respectively $\bar{\mathbf{y}}'$) is the set of all bound resource variables and $\bar{\mathbf{x}}$ ($\bar{\mathbf{x}}'$) is the set of all free resource variables of A (A')

$$\mathsf{cut} \quad \frac{\Gamma \vdash t : A \quad \Delta, x : A \vdash u : B}{\Gamma, \Delta \vdash u[t/x] : B}$$

$$\otimes L \quad \frac{\Gamma, x\langle \bar{\mathbf{y}}_0; \bar{\mathbf{x}}_0 \rangle : A, y\langle \bar{\mathbf{y}}_1; \bar{\mathbf{x}}_1 \rangle : B \vdash t : C}{\Gamma, z\langle \bar{\mathbf{y}}_0, \bar{\mathbf{y}}_1; \bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1 \rangle : A \otimes B \vdash (\texttt{let } z \texttt{ be } x \otimes y \texttt{ in } t) : C}$$

$$\otimes R \quad \frac{\Gamma \vdash t\langle \bar{\mathbf{y}}_0; \bar{\mathbf{x}}_0 \rangle : A \quad \Delta \vdash u\langle \bar{\mathbf{y}}_1; \bar{\mathbf{x}}_1 \rangle : B}{\Gamma, \Delta \vdash (t \otimes u)\langle \bar{\mathbf{y}}_0, \bar{\mathbf{y}}_1; \bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1 \rangle : A \otimes B}$$

$$\multimap L \quad \frac{\Gamma \vdash t\langle \bar{\mathbf{y}}_0; \bar{\mathbf{x}}_0 \rangle : A \quad \Delta, x\langle \bar{\mathbf{y}}_1; \bar{\mathbf{x}}_1 \rangle : B \vdash u : C}{\Gamma, \Delta, f\langle \bar{\mathbf{y}}_0, \bar{\mathbf{y}}_1; \bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1 \rangle : A \multimap B \vdash u[(ft)/x] : C}$$

$$\multimap R \quad \frac{\Gamma, x\langle \bar{\mathbf{y}}_0; \bar{\mathbf{x}}_0 \rangle : A \vdash t\langle \bar{\mathbf{y}}_1, (\bar{\mathbf{y}}_0); \bar{\mathbf{x}}_1, (\bar{\mathbf{x}}_0) \rangle : B}{\Gamma \vdash (\lambda x.t)\langle \bar{\mathbf{y}}_0, \bar{\mathbf{y}}_1; \bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1 \rangle : A \multimap B}$$

$$\forall L \quad \frac{\Gamma, x\langle \bar{\mathbf{y}}, \bar{\mathbf{y}}'; \bar{\mathbf{x}}, \bar{\mathbf{x}}' \rangle : A[\alpha := T] \vdash t : B}{\Gamma, x\langle \bar{\mathbf{y}}; \bar{\mathbf{x}} \rangle : (\forall \alpha) A \vdash t : B}$$

where $\bar{\mathbf{y}}'$ and $\bar{\mathbf{x}}'$ are all the new resource variables introduced in T

$$\forall R \quad \frac{\Gamma \vdash t : A}{\Gamma \vdash t : (\forall \alpha) A} \quad (\alpha \text{ not free in } \Gamma)$$

Figure 4.1: Term Assignment for BLL ("ordinary" rules)

$$!W \quad \frac{\Gamma \vdash t : B}{\Gamma, a\langle \mathbf{y}_R, \bar{\mathbf{y}}; \bar{\mathbf{x}}, \mathcal{V}(R)\rangle : !_{\mathbf{y}<R}A \vdash (\texttt{discard}\ a\ \texttt{in}\ t) : B}$$

where $\bar{\mathbf{x}}$ is the set of all free resource variables of A, except $\mathbf{y}$

$$!D \quad \frac{\Gamma, x\langle \bar{\mathbf{y}}; \bar{\mathbf{x}}\rangle : A[\mathbf{y} := 0] \vdash t : B}{\Gamma, y\langle \mathbf{y}_{1+R}, \bar{\mathbf{y}}; \bar{\mathbf{x}}, \mathcal{V}(R)\rangle : !_{\mathbf{y}<1+R}A \vdash (\texttt{derelict}\ y\ \texttt{for}\ x\ \texttt{in}\ t) : B}$$

where $\bar{\mathbf{x}}$ is the set of all free resource variables of A, except $\mathbf{y}$

$$!C \quad \frac{\begin{array}{l}\Gamma, x\langle \mathbf{x}^0_P, \bar{\mathbf{y}}_0; \bar{\mathbf{x}}, \mathcal{V}(P)\rangle : !_{\mathbf{x}^0<P}A[\mathbf{x}^2 := \mathbf{x}^0], \\ y\langle \mathbf{x}^1_Q, \bar{\mathbf{y}}_1; \bar{\mathbf{x}}, \mathcal{V}(Q), (\mathcal{V}(P))\rangle : !_{\mathbf{x}^1<Q}A[\mathbf{x}^2 := \mathbf{x}^1 + P] \vdash t : B\end{array}}{\begin{array}{l}\Gamma, z\langle \mathbf{x}^2_{P+Q+R}, \bar{\mathbf{y}}'; \bar{\mathbf{x}}, \mathcal{V}(P+Q+R)\rangle : !_{\mathbf{x}^2<P+Q+R}A \\ \vdash (\texttt{copy}\ z\ \texttt{as}\ x, y\ \texttt{in}\ t) : B\end{array}}$$

where $\bar{\mathbf{y}}_0 \equiv \bar{\mathbf{y}}_1 \equiv \bar{\mathbf{y}}'$, $\mathcal{V}(P)$ appears in y iff $\mathbf{x}^2$ occurs free in A, and $\bar{\mathbf{x}}$ is the set of all free resource variables of A, except $\mathbf{x}^2$

$$P! \quad \frac{\begin{array}{l}\ldots, x_i\langle \mathbf{w}^i_{Q_i(\mathbf{x})}, \bar{\mathbf{y}}_i; \bar{\mathbf{x}}_i, (\mathbf{x})\rangle : !_{\mathbf{w}^i<Q_i(\mathbf{x})}A_i[\mathbf{y}^i := V_i(\mathbf{x}) + \mathbf{w}^i], \ldots \\ \vdash t\langle \bar{\mathbf{y}}; \bar{\mathbf{x}}, (\mathbf{x})\rangle : B\end{array}}{\begin{array}{l}\ldots, y_i\langle \mathbf{y}^i_{V_i(P)+R_i}, \bar{\mathbf{y}}_i; \bar{\mathbf{x}}_i, \mathcal{V}(R_i), (\mathcal{V}(P))\rangle : !_{\mathbf{y}^i<V_i(P)+R_i}A_i, \ldots \\ \vdash (\texttt{promote} \ldots, y_i, \ldots \texttt{for} \ldots, x_i, \ldots \texttt{in}\ !_{\mathbf{x}<P}t)\langle \mathbf{x}_P, \bar{\mathbf{y}}; \bar{\mathbf{x}}, \mathcal{V}(P)\rangle : !_{\mathbf{x}<P}B\end{array}}$$

where $V_i(\mathbf{x}) = \sum_{\mathbf{z}<\mathbf{x}} Q_i(\mathbf{z})$, $V_i(\mathbf{x}) + \mathbf{w}^i$ is free for $\mathbf{y}^i$ in A_i, $\bar{\mathbf{x}}_i$ is the set of all free resource variables of A_i, except $\mathbf{y}^i$, $\bar{\mathbf{x}}$ is the set of all free resource variables of B, except $\mathbf{x}$, $\mathcal{V}(P)$ appears in y_i iff $V_i(\mathbf{x})$ is not constant, $\mathbf{x}$ appears in x_i iff $Q_i(\mathbf{x})$ is not constant or $V_i(\mathbf{x})$ is not constant and $\mathbf{y}^i$ occurs free in A_i, and $1 \leq i \leq n$

Figure 4.1: Term Assignment for BLL ("storage" rules)

IdR $y[t\langle \bar{\mathbf{y}}_0\rangle / y\langle \bar{\mathbf{y}}_0\rangle]\langle \bar{\mathbf{y}}_1\rangle \;\rhd\; t\langle \bar{\mathbf{y}}_1\rangle$
where $\bar{\mathbf{y}}_0 \sqsubseteq \bar{\mathbf{y}}_1$

IdL $t_{y\langle \bar{\mathbf{y}}_1\rangle}[x\langle \bar{\mathbf{y}}_1\rangle / y\langle \bar{\mathbf{y}}_1\rangle]\langle \bar{\mathbf{y}}_1, \bar{\mathbf{y}}\rangle \;\rhd\; t_{x\langle \bar{\mathbf{y}}_0\rangle}\langle \bar{\mathbf{y}}_0, \bar{\mathbf{y}}\rangle$
where $\bar{\mathbf{y}}_0 \sqsubseteq \bar{\mathbf{y}}_1$, x does not occur in t_y, and t_x denotes a term t in which variable x is free

S⊗ $$\begin{array}{ll} & (\texttt{let } z\langle \bar{\mathbf{y}}_0, \bar{\mathbf{y}}_1; \bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1\rangle \texttt{ be } x \otimes y \texttt{ in } c)[a\langle \bar{\mathbf{y}}'_0; \bar{\mathbf{x}}_0\rangle \otimes b\langle \bar{\mathbf{y}}'_1; \bar{\mathbf{x}}_1\rangle / z] \\ \rhd & c[a\langle \bar{\mathbf{y}}_0; \bar{\mathbf{x}}_0\rangle / x, b\langle \bar{\mathbf{y}}_1; \bar{\mathbf{x}}_1\rangle / y] \end{array}$$
where $\bar{\mathbf{y}}'_0 \sqsubseteq \bar{\mathbf{y}}_0$ and $\bar{\mathbf{y}}'_1 \sqsubseteq \bar{\mathbf{y}}_1$

S⊸ $c[(z\, a\langle \bar{\mathbf{y}}'; \bar{\mathbf{x}}\rangle)/y][(\lambda x\langle \bar{\mathbf{y}}; \bar{\mathbf{x}}\rangle.b)/z] \;\rhd\; c[b[a\langle \bar{\mathbf{y}}; \bar{\mathbf{x}}\rangle / x]/y]$
where $\bar{\mathbf{y}}' \sqsubseteq \bar{\mathbf{y}}$

SP!W $(\texttt{discard } z\langle \mathbf{y}_P\rangle \texttt{ in } b)[!_{\mathbf{x}<P'} a/z] \;\rhd\; b$
where $P' \sqsupseteq P$

SP!D $(\texttt{derelict } z\langle \mathbf{y}_P\rangle \texttt{ for } x \texttt{ in } b)[!_{\mathbf{x}<P'} a/z] \;\rhd\; b[a\langle \mathbf{x} := 0\rangle / x]$
where $P' \sqsupseteq P$ and $a\langle \mathbf{x} := 0\rangle$ is the result of substituting 0 for every free occurence of $\mathbf{x}$ in a (with the convention that $0 \circledast P = P$)

SP!C $$\begin{array}{ll} & (\texttt{copy } z\langle \mathbf{x}^2_{P+Q+R}\rangle \texttt{ as } x\langle \mathbf{x}^0_P\rangle, y\langle \mathbf{x}^1_Q\rangle \texttt{ in } b)[!_{\mathbf{x}^3<R} a/z] \\ \rhd & b[!_{\mathbf{x}^3<P} a/x, !_{\mathbf{x}^4<Q+R} a'\langle \mathbf{x}^4 \circledast P\rangle / y] \end{array}$$
where $R \sqsupseteq P+Q+R$, a' is a with all variables renamed, and P is added to whatever base polynomial $\mathbf{x}^3$ might have had already in a

SP!P $(\texttt{promote} \ldots, y_i\langle \mathbf{y}^i_{V_i(P)+R_i}\rangle, \ldots \texttt{for} \ldots, x_i\langle \mathbf{w}^i_{Q_i(\mathbf{x})}\rangle, \ldots \texttt{in } !_{\mathbf{x}<P} t)[!_{\mathbf{y}<R} a/y_i]$
$\rhd\; \texttt{promote} \ldots, y_{i-1}, y_{i+1}, \ldots \texttt{for} \ldots, x_{i-1}, x_{i+1}, \ldots \texttt{in } !_{\mathbf{x}<P} \left(t\left[!_{\mathbf{y}<Q_i(\mathbf{x})} a/x_i\right]\right)$
where $R \sqsupseteq V_i(P) + R_i$

Figure 4.2: Symmetric Reductions

CR⊗R $(b \otimes c)[a/x] \ \triangleright \ \begin{cases} (b[a/x] \otimes c) & \text{if } x \text{ occurs in } b \\ (b \otimes c[a/x]) & \text{if } x \text{ occurs in } c \end{cases}$
(x will occur in only one of b or c, by linearity)

CR⊸R $(\lambda y.c)[a/x] \ \triangleright \ (\lambda y.c[a/x])$

CL⊗L $b[(\texttt{let}\ w\ \texttt{be}\ y \otimes z\ \texttt{in}\ a)/x] \ \triangleright \ \texttt{let}\ w\ \texttt{be}\ y \otimes z\ \texttt{in}\ (b[a/x])$

CL⊸L $b[(a[(wc)/y])/x] \ \triangleright \ b[a/x][(wc)/y]$

CR⊗L $(\texttt{let}\ w\ \texttt{be}\ y \otimes z\ \texttt{in}\ b)[a/x] \ \triangleright \ \texttt{let}\ w\ \texttt{be}\ y \otimes z\ \texttt{in}\ (b[a/x])$

CR⊸L $b[(wc)/y][a/x] \ \triangleright \ \begin{cases} b[a/x][(wc)/y] & \text{if } x \text{ occurs in } b \\ b[(w(c[a/x]))/y] & \text{if } x \text{ occurs in } c \end{cases}$
(x will occur in only one of b or c, by linearity)

CL!W $b[(\texttt{discard}\ w\ \texttt{in}\ a)/x] \ \triangleright \ \texttt{discard}\ w\ \texttt{in}\ (b[a/x])$

CL!D $b[(\texttt{derelict}\ w\ \texttt{for}\ z\ \texttt{in}\ a)/x] \ \triangleright \ \texttt{derelict}\ w\ \texttt{for}\ z\ \texttt{in}\ (b[a/x])$

CL!C $b[(\texttt{copy}\ w\ \texttt{as}\ y, z\ \texttt{in}\ a)/x] \ \triangleright \ \texttt{copy}\ w\ \texttt{as}\ y, z\ \texttt{in}\ (b[a/x])$

CR!W $(\texttt{discard}\ w\ \texttt{in}\ b)[a/x] \ \triangleright \ \texttt{discard}\ w\ \texttt{in}\ (b[a/x])$

CR!D $(\texttt{derelict}\ w\ \texttt{for}\ z\ \texttt{in}\ b)[a/x] \ \triangleright \ \texttt{derelict}\ w\ \texttt{for}\ z\ \texttt{in}\ (b[a/x])$

CR!C $(\texttt{copy}\ w\ \texttt{as}\ y, z\ \texttt{in}\ b)[a/x] \ \triangleright \ \texttt{copy}\ w\ \texttt{as}\ y, z\ \texttt{in}\ (b[a/x])$

Figure 4.3: Commutative Reductions

4.5 Examples

This section contains examples of term variables, bounded linear terms, and reduction on terms which will help to clarify certain issues about the term assignment for BLL, bounded linear terms, and the reduction relation on terms.

4.5.1 Of Term Variables

In this subsection, a few simple examples of term variables associated with different BLL formulas will be given, to illustrate how the lists of resource variables are obtained from the formula.

formula of BLL	term variable
$\alpha^1(0)$	x
$\beta^2(3, \mathbf{z}(\mathbf{y})^4 + \mathbf{y}) \multimap \alpha^1(\mathbf{x})$	$x\langle ; \mathbf{x}, \mathbf{y}, \mathbf{z}\rangle$
$!_{\mathbf{y}^0<3}\alpha^1(\mathbf{y}^0) \otimes !_{\mathbf{y}^1<7}\beta^1(0)$	$x\langle \mathbf{y}^0_3, \mathbf{y}^1_7; \rangle$
$!_{\mathbf{y}<\mathbf{x}}\alpha^1(\mathbf{y} + \mathbf{y}\mathbf{x})$	$x\langle \mathbf{y}_\mathbf{x}; \mathbf{x}\rangle$
$(\forall\alpha)(\ !_{\mathbf{y}^0<\mathbf{x}}\alpha^1(\mathbf{y}^0) \multimap (\forall\beta)(\beta^2(\mathbf{x},\mathbf{x}) \otimes \alpha^1(\mathbf{z}))\ )$	$x\langle \mathbf{y}^0_\mathbf{x}; \mathbf{x}, \mathbf{z}\rangle$

4.5.2 Of Bounded Linear Terms

In this subsection, examples of bounded linear terms will be given, based on similar examples given in the paper on BLL. These examples should be sufficient to elucidate the way in which the different term formation operators function.

The tally natural number "1" is given by the following BLL proof, which has been annotated according to the term assignment thus giving us the term representation of *1*. (Tally natural numbers are simply natural numbers generated from zero and a successor function, *i.e.*, unary numbers.)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{x : \alpha^1(0) \vdash x : \alpha^1(0) \quad y : \alpha^1(1) \vdash y : \alpha^1(1)}{f : \alpha^1(0) \multimap \alpha^1(1), x : \alpha^1(0) \vdash (f\ x) : \alpha^1(1)}\ \multimap L}{\begin{array}{l} f_0\langle \mathbf{y}^0_1 \rangle : !_{\mathbf{y}^0<1}(\alpha^1(\mathbf{y}^0) \multimap \alpha^1(\mathbf{y}^0+1)), x : \alpha^1(0) \\ \vdash \mathtt{derelict}\ f_0\ \mathtt{for}\ f\ \mathtt{in}\ (f\ x) : \alpha^1(1) \end{array}}\ !D}{\begin{array}{l} f_0\langle \mathbf{y}^0_1 \rangle : !_{\mathbf{y}^0<1}(\alpha^1(\mathbf{y}^0) \multimap \alpha^1(\mathbf{y}^0+1)) \\ \vdash \lambda x.\mathtt{derelict}\ f_0\ \mathtt{for}\ f\ \mathtt{in}\ (f\ x) : \alpha^1(0) \multimap \alpha^1(1) \end{array}}\ \multimap R}{\begin{array}{l} \vdash \lambda f_0\langle \mathbf{y}^0_1 \rangle.\lambda x.\mathtt{derelict}\ f_0\ \mathtt{for}\ f\ \mathtt{in}\ (f\ x) : \\ !_{\mathbf{y}^0<1}(\alpha^1(\mathbf{y}^0) \multimap \alpha^1(\mathbf{y}^0+1)) \multimap \alpha^1(0) \multimap \alpha^1(1) \end{array}}\ \multimap R}{\begin{array}{l} \vdash \lambda f_0\langle \mathbf{y}^0_1 \rangle.\lambda x.\mathtt{derelict}\ f_0\ \mathtt{for}\ f\ \mathtt{in}\ (f\ x) : \\ (\forall \alpha)\ !_{\mathbf{y}^0<1}(\alpha^1(\mathbf{y}^0) \multimap \alpha^1(\mathbf{y}^0+1)) \multimap \alpha^1(0) \multimap \alpha^1(1) \end{array}}\ \forall R$$

On the other hand, the tally natural number "2" is given by the following term:

$$\begin{aligned} \mathit{2}\langle \mathbf{w}^0_2 \rangle \ = \ & \lambda g\langle \mathbf{w}^0_2 \rangle.\lambda z.\mathtt{copy}\ g\ \mathtt{as}\ g_1\langle \mathbf{w}^1_1 \rangle, g_2\langle \mathbf{w}^2_1 \rangle\ \mathtt{in}\ \mathtt{derelict}\ g_2\ \mathtt{for}\ g_3\ \mathtt{in} \\ & \mathtt{derelict}\ g_1\ \mathtt{for}\ g_4\ \mathtt{in}\ g_3\,(g_4\,z) \end{aligned}$$

The dyadic list "112" is given by the following term. (Dyadic lists are simply lists on two symbols, usually "1" and "2", generated from an empty list and two successor functions, "append 1" and "append 2". These lists have the property that every natural number has a unique dyadic representation, unlike the binary representation where leading zeros destroy the uniqueness.)

$$\begin{array}{l} \lambda f_0\langle \mathbf{x}^0_3 \rangle.\lambda f_1\langle \mathbf{x}^1_3 \rangle.\lambda x.\mathtt{copy}\ f_1\ \mathtt{as}\ g_1\langle \mathbf{y}^1_2 \rangle, g_2\langle \mathbf{y}^2_1 \rangle\ \mathtt{in}\ \mathtt{discard}\ g_1\ \mathtt{in} \\ \quad \mathtt{derelict}\ g_2\ \mathtt{for}\ g_3\ \mathtt{in}\ g_3\,(\mathtt{copy}\ f_0\ \mathtt{as}\ h_1\langle \mathbf{z}^1_1 \rangle, h_2\langle \mathbf{z}^2_1 \rangle\ \mathtt{in} \\ \quad\ \mathtt{derelict}\ h_2\ \mathtt{for}\ h_3\ \mathtt{in}\ \mathtt{derelict}\ h_1\ \mathtt{for}\ h_4\ \mathtt{in}\ h_3\,(h_4\,x)) \end{array}$$

While the next term represents the dyadic list "121":

$$\lambda f_0\langle \mathbf{x}_3^0\rangle.\lambda f_1\langle \mathbf{x}_3^1\rangle.\lambda x.\mathtt{copy}\ f_0\ \mathtt{as}\ g_1\langle \mathbf{y}_2^1\rangle, g_2\langle \mathbf{y}_1^2\rangle\ \mathtt{in\ derelict}\ g_2\ \mathtt{for}\ g_3\ \mathtt{in}$$
$$g_3\,(\mathtt{derelict}\ f_1\ \mathtt{for}\ f_2\ \mathtt{in\ derelict}\ g_1\ \mathtt{for}\ g_4\ \mathtt{in}\ f_2\,(g_4\,x)\,)$$

The tally successor function is represented by the following term:

$$t\langle \mathbf{x}\rangle = t\langle \mathbf{y}_\mathbf{x}^0, \mathbf{y}_{\mathbf{x}+1}^1; \mathbf{x}\rangle =$$
$$\lambda e\langle \mathbf{y}_\mathbf{x}^0; \mathbf{x}\rangle.\lambda f\langle \mathbf{y}_{\mathbf{x}+1}^1; \mathbf{x}\rangle.\lambda x.\mathtt{copy}\ f\ \mathtt{as}\ f_0\langle \mathbf{y}_\mathbf{x}^2; \mathbf{x}\rangle, f_1\langle \mathbf{y}_1^3; \mathbf{x}\rangle\ \mathtt{in}$$
$$\mathtt{derelict}\ f_1\ \mathtt{for}\ f_2\langle ; \mathbf{x}\rangle\ \mathtt{in}\ f_2\,(e\, f_0\, x)$$

A "reuse" operator on tally natural numbers is given by

$$R\langle \mathbf{x}\rangle = R\langle \bar{\mathbf{y}}; \mathbf{x}\rangle = \lambda f\langle \mathbf{y}_\mathbf{x}^1, \bar{\mathbf{y}}_0, \bar{\mathbf{y}}_1, \mathbf{y}_1^2, \mathbf{y}_\mathbf{x}^3; \mathbf{x}\rangle.f\ !_{\mathbf{y}^1<\mathbf{x}} s\langle \bar{\mathbf{y}}_0; \mathbf{y}^1\rangle\ T_0\langle \bar{\mathbf{y}}_1; \rangle$$

where

$$\begin{aligned} T_0\langle \mathbf{z}_1^0, \mathbf{z}_0^1; \rangle &= !_{\mathbf{z}^0<1}(\lambda f\langle \mathbf{z}_0^1\rangle.\lambda x.\mathtt{discard}\ f\ \mathtt{in}\ x) \\ s\langle \mathbf{x}\rangle = s\langle \mathbf{z}_1^2, \mathbf{z}_1^5, \mathbf{z}_\mathbf{x}^3, \mathbf{z}_{\mathbf{x}+1}^6; \mathbf{x}\rangle &= \lambda e\langle \mathbf{z}_1^2, \mathbf{z}_\mathbf{x}^3; \mathbf{x}\rangle.\mathtt{promote}\ e\ \mathtt{for}\ e_0\langle \mathbf{z}_1^4, \mathbf{z}_\mathbf{x}^3; \mathbf{x}\rangle\ \mathtt{in} \\ & \quad !_{\mathbf{z}^5<1}(\mathtt{derelict}\ e_0\ \mathtt{for}\ e_1\langle \mathbf{z}_\mathbf{x}^3; \mathbf{x}\rangle\ \mathtt{in}\ t\langle \mathbf{z}_\mathbf{x}^3, \mathbf{z}_{\mathbf{x}+1}^6; \mathbf{x}\rangle\, e_1) \end{aligned}$$

The last four terms correspond to key proofs in BLL, as will be seen in the next chapter; they define a reuse operator on dyadic lists.

$$E_0\langle \mathbf{z}_1^0, \mathbf{y}_0^0, \mathbf{y}_0^1; \rangle =$$
$$!_{\mathbf{z}^0<1}(\lambda f_0\langle \mathbf{y}_0^0\rangle.\lambda f_1\langle \mathbf{y}_0^1\rangle.\lambda x.\mathtt{discard}\ f_0\ \mathtt{in\ discard}\ f_1\ \mathtt{in}\ x) : !_{\mathbf{z}^0<1} N_0^2$$

$$\sigma_1\langle \mathbf{x}\rangle = \sigma_1\langle \mathbf{z}_1^1, \mathbf{y}_\mathbf{x}^2, \mathbf{y}_\mathbf{x}^3, \mathbf{z}_1^2, \mathbf{y}_{\mathbf{x}+1}^0, \mathbf{y}_{\mathbf{x}+1}^1; \mathbf{x}\rangle =$$
$$\lambda e_1\langle \mathbf{z}_1^1, \mathbf{y}_\mathbf{x}^2, \mathbf{y}_\mathbf{x}^3; \mathbf{x}\rangle.\mathtt{promote}\ e_1\ \mathtt{for}\ e_0\langle \mathbf{z}_1^0, \mathbf{y}_\mathbf{x}^2, \mathbf{y}_\mathbf{x}^3; \mathbf{x}\rangle\ \mathtt{in}$$
$$!_{\mathbf{z}^2<1}(\mathtt{derelict}\ e_0\ \mathtt{for}\ e\langle \mathbf{y}_\mathbf{x}^2, \mathbf{y}_\mathbf{x}^3; \mathbf{x}\rangle\ \mathtt{in}$$
$$\lambda f_0\langle \mathbf{y}_{\mathbf{x}+1}^0; \mathbf{x}\rangle.\lambda f_1\langle \mathbf{y}_{\mathbf{x}+1}^1; \mathbf{x}\rangle.\lambda x.\mathtt{copy}\ f_0\ \mathtt{as}\ f_2\langle \mathbf{y}_\mathbf{x}^4; \mathbf{x}\rangle, f_3\langle \mathbf{x}_1^5; \mathbf{x}\rangle\ \mathtt{in}$$
$$\mathtt{derelict}\ f_3\ \mathtt{for}\ f_4\langle ; \mathbf{x}\rangle\ \mathtt{in}\ (f_4\ (e\ f_2\ f_1\ x))) : !_{\mathbf{z}^1<1} N_\mathbf{x}^2 \multimap !_{\mathbf{z}^2<1} N_{\mathbf{x}+1}^2$$

$$\sigma_2\langle\mathbf{x}\rangle = \sigma_2\langle \mathbf{z}_1^1, \mathbf{y}_\mathbf{x}^2, \mathbf{y}_\mathbf{x}^3, \mathbf{z}_1^2, \mathbf{y}_{\mathbf{x}+1}^0, \mathbf{y}_{\mathbf{x}+1}^1; \mathbf{x}\rangle =$$
$$\lambda e_1\langle \mathbf{z}_1^1, \mathbf{y}_\mathbf{x}^2, \mathbf{y}_\mathbf{x}^3; \mathbf{x}\rangle.\texttt{promote } e_1 \texttt{ for } e_0\langle \mathbf{z}_1^0, \mathbf{y}_\mathbf{x}^2, \mathbf{y}_\mathbf{x}^3; \mathbf{x}\rangle \texttt{ in}$$
$$!_{\mathbf{z}^2<1}(\texttt{derelict } e_0 \texttt{ for } e\langle \mathbf{y}_\mathbf{x}^2, \mathbf{y}_\mathbf{x}^3; \mathbf{x}\rangle \texttt{ in}$$
$$\lambda f_0\langle \mathbf{y}_{\mathbf{x}+1}^0; \mathbf{x}\rangle.\lambda f_1\langle \mathbf{y}_{\mathbf{x}+1}^1; \mathbf{x}\rangle.\lambda x.\texttt{copy } f_1 \texttt{ as } f_2\langle \mathbf{y}_\mathbf{x}^4; \mathbf{x}\rangle, f_3\langle \mathbf{x}_1^5; \mathbf{x}\rangle \texttt{ in}$$
$$\texttt{derelict } f_3 \texttt{ for } f_4\langle ; \mathbf{x}\rangle \texttt{ in } (f_4\ (e\ f_0\ f_2\ x))) : !_{\mathbf{z}^1<1} N_\mathbf{x}^2 \multimap !_{\mathbf{z}^2<1} N_{\mathbf{x}+1}^2$$

$$r\langle\mathbf{x}\rangle = r\langle\bar{\mathbf{y}};\mathbf{x}\rangle = (\lambda f\langle \mathbf{w}_\mathbf{x}^1, \bar{\mathbf{y}}_1, \mathbf{w}_\mathbf{x}^2, \bar{\mathbf{y}}_1, \bar{\mathbf{y}}_0, \mathbf{w}_1, \mathbf{y}_\mathbf{x}^6, \mathbf{y}_\mathbf{x}^7; \mathbf{x}\rangle.$$
$$f\ !_{\mathbf{w}^1<\mathbf{x}}\sigma_1\langle\bar{\mathbf{y}}_1; \mathbf{w}^1\rangle\ !_{\mathbf{w}^2<\mathbf{x}}\sigma_2\langle\bar{\mathbf{y}}_2; \mathbf{w}^2\rangle\ E_0\langle\bar{\mathbf{y}}_0;\rangle) : N_\mathbf{x}^2 \multimap !_{\mathbf{w}<1} N_\mathbf{x}^2$$

where $N_\mathbf{x}^2 =$

$$(\forall\alpha)\ !_{\mathbf{y}^0<\mathbf{x}}(\alpha^1(\mathbf{y}^0) \multimap \alpha^1(\mathbf{y}^0+1)) \multimap !_{\mathbf{y}^1<\mathbf{x}}(\alpha^1(\mathbf{y}^1) \multimap \alpha^1(\mathbf{y}^1+1)) \multimap \alpha^1(0) \multimap \alpha^1(\mathbf{x})$$

4.5.3 Of Reductions on Terms

In this subsection, two examples of reductions on bounded linear terms will be given. The reader should have Figures 4.2 and 4.3 handy, as I will label each reduction step by its name as it appears in those figures.

The first example consists of the tally successor function applied to the tally natural number 2, *i.e.*, we will reduce the term $(t\langle 2\rangle\ \mathit{2}\langle \mathbf{w}_2^0\rangle)$.

$$(\ \lambda e\langle \mathbf{y}_2^0\rangle.\lambda f\langle \mathbf{y}_3^1\rangle.\lambda x.\texttt{copy } f \texttt{ as } f_0\langle \mathbf{y}_2^2\rangle, f_1\langle \mathbf{y}_1^3\rangle \texttt{ in derelict } f_1 \texttt{ for } f_2$$
$$\texttt{in } f_2\ (e\ f_0\ x)\ )\ \mathit{2}$$

apply reduction "S⊸"

$$\rhd\quad (\ \lambda f\langle \mathbf{y}_3^1\rangle.\lambda x.\texttt{copy } f \texttt{ as } f_0\langle \mathbf{y}_2^2\rangle, f_1\langle \mathbf{y}_1^3\rangle \texttt{ in derelict } f_1 \texttt{ for } f_2 \texttt{ in}$$
$$f_2\ (e\langle \mathbf{y}_2^0\rangle\ f_0\ x)\ )[\mathit{2}/e]$$

apply reduction "CR⊸R"

$$\rhd\quad \lambda f\langle \mathbf{y}_3^1\rangle.(\ \lambda x.\texttt{copy } f \texttt{ as } f_0\langle \mathbf{y}_2^2\rangle, f_1\langle \mathbf{y}_1^3\rangle \texttt{ in derelict } f_1 \texttt{ for } f_2 \texttt{ in}$$
$$f_2\ (e\langle \mathbf{y}_2^0\rangle\ f_0\ x)\ )[\mathit{2}/e]$$

⋮ apply the appropriate commutative reductions

$$\rhd\quad \lambda f\langle \mathbf{y}_3^1\rangle.\lambda x.\texttt{copy } f \texttt{ as } f_0\langle \mathbf{y}_2^2\rangle, f_1\langle \mathbf{y}_1^3\rangle \texttt{ in derelict } f_1 \texttt{ for } f_2 \texttt{ in}$$
$$(\ (f_2\ (e\langle \mathbf{y}_2^0\rangle\ f_0\ x))[\mathit{2}/e]\ )$$

apply reduction "CR—∘L"

$\triangleright$ $\lambda f\langle \mathbf{y}_3^1\rangle.\lambda x.\mathtt{copy}\ f\ \mathtt{as}\ f_0\langle \mathbf{y}_2^2\rangle, f_1\langle \mathbf{y}_1^3\rangle\ \mathtt{in}\ \ \mathtt{derelict}\ f_1\ \mathtt{for}\ f_2\ \mathtt{in}$
$f_2\ (\ (e\langle \mathbf{y}_2^0\rangle\ f_0\ x)[2/e]\ )$

apply reduction "S—∘"

$\triangleright$ $\lambda f\langle \mathbf{y}_3^1\rangle.\lambda x.\mathtt{copy}\ f\ \mathtt{as}\ f_0\langle \mathbf{y}_2^2\rangle, f_1\langle \mathbf{y}_1^3\rangle\ \mathtt{in}\ \ \mathtt{derelict}\ f_1\ \mathtt{for}\ f_2\ \mathtt{in}\ f_2$
$(\ \lambda z.\mathtt{copy}\ g\ \mathtt{as}\ g_1\langle \mathbf{w}_1^1\rangle, g_2\langle \mathbf{w}_1^2\rangle\ \mathtt{in\ derelict}\ g_2\ \mathtt{for}\ g_3\ \mathtt{in}$
$\mathtt{derelict}\ g_1\ \mathtt{for}\ g_4\ \mathtt{in}\ g_3\,(g_4\ z)\ )[f_0\langle \mathbf{y}_2^2\rangle/g\langle \mathbf{w}_2^0\rangle]\ x$

apply reductions "CR—∘R" and "IdL"

$\triangleright$ $\lambda f\langle \mathbf{y}_3^1\rangle.\lambda x.\mathtt{copy}\ f\ \mathtt{as}\ f_0\langle \mathbf{y}_2^2\rangle, f_1\langle \mathbf{y}_1^3\rangle\ \mathtt{in}\ \ \mathtt{derelict}\ f_1\ \mathtt{for}\ f_2\ \mathtt{in}\ f_2$
$(\ \lambda z.\mathtt{copy}\ f_0\ \mathtt{as}\ g_1\langle \mathbf{w}_1^1\rangle, g_2\langle \mathbf{w}_1^2\rangle\ \mathtt{in\ derelict}\ g_2\ \mathtt{for}\ g_3\ \mathtt{in}$
$\mathtt{derelict}\ g_1\ \mathtt{for}\ g_4\ \mathtt{in}\ g_3\,(g_4\ z)\ )\ x$

apply reduction "S—∘"

$\triangleright$ $\lambda f\langle \mathbf{y}_3^1\rangle.\lambda x.\mathtt{copy}\ f\ \mathtt{as}\ f_0\langle \mathbf{y}_2^2\rangle, f_1\langle \mathbf{y}_1^3\rangle\ \mathtt{in}\ \ \mathtt{derelict}\ f_1\ \mathtt{for}\ f_2\ \mathtt{in}\ f_2$
$(\ \mathtt{copy}\ f_0\ \mathtt{as}\ g_1\langle \mathbf{w}_1^1\rangle, g_2\langle \mathbf{w}_1^2\rangle\ \mathtt{in\ derelict}\ g_2\ \mathtt{for}\ g_3\ \mathtt{in}$
$\mathtt{derelict}\ g_1\ \mathtt{for}\ g_4\ \mathtt{in}\ g_3\,(g_4\ z)\ )[x/z]$

apply reduction "IdL"

$\triangleright$ $\lambda f\langle \mathbf{y}_3^1\rangle.\lambda x.\mathtt{copy}\ f\ \mathtt{as}\ f_0\langle \mathbf{y}_2^2\rangle, f_1\langle \mathbf{y}_1^3\rangle\ \mathtt{in}\ \ \mathtt{derelict}\ f_1\ \mathtt{for}\ f_2\ \mathtt{in}\ f_2$
$(\ \mathtt{copy}\ f_0\ \mathtt{as}\ g_1\langle \mathbf{w}_1^1\rangle, g_2\langle \mathbf{w}_1^2\rangle\ \mathtt{in\ derelict}\ g_2\ \mathtt{for}\ g_3\ \mathtt{in}$
$\mathtt{derelict}\ g_1\ \mathtt{for}\ g_4\ \mathtt{in}\ g_3\,(g_4\ x)\ )$

which is exactly the usual representation for the tally natural number 3.

In the next example, the reduction of $(R\langle 2\rangle\ 2\langle \mathbf{w}_2^0\rangle)$ will be shown, but not all steps in the reduction will be given explicitly, for conciseness.

$(\lambda f\langle \mathbf{y}_2^1\rangle.f\ !_{\mathbf{y}^1<2}s\langle \mathbf{y}^1\rangle\ T_0)\ 2\langle \mathbf{w}_2^0\rangle$

apply "S—∘", "CR—∘R", and "S—∘"

$\rightsquigarrow$ $(\ \mathtt{copy}\ g\langle \mathbf{w}_2^0\rangle\ \mathtt{as}\ g_1\langle \mathbf{w}_1^1\rangle, g_2\langle \mathbf{w}_1^2\rangle\ \mathtt{in\ derelict}\ g_2\ \mathtt{for}\ g_3\ \mathtt{in}$
$\mathtt{derelict}\ g_1\ \mathtt{for}\ g_4\ \mathtt{in}\ g_3\,(g_4\ x)\ )[!_{\mathbf{y}^1<2}s\langle \mathbf{y}^1\rangle/g][T_0/x]$

apply "SP!C"

$\triangleright$ $(\ \mathtt{derelict}\ g_2\langle \mathbf{w}_1^2\rangle\ \mathtt{for}\ g_3\ \mathtt{in\ derelict}\ g_1\langle \mathbf{w}_1^1\rangle\ \mathtt{for}\ g_4\ \mathtt{in}\ g_3\,(g_4\ x)\ )$
$[!_{\mathbf{y}^1<1}s\langle \mathbf{y}^1\rangle/g_1, !_{\mathbf{y}^2<1}s'\langle \mathbf{y}^2 \circledast 1\rangle/g_2][T_0/x]$

apply "SP!D", "CR!D", "SP!D"

$\rightsquigarrow \quad (\ g_3\ (g_4\ x)\ )[s'\langle 1\rangle/g_3][s\langle 0\rangle/g_4][T_0/x]$

apply "CR⊸L" and "S⊸"

$\rightsquigarrow \quad (g_3\ (\ \mathtt{promote}\ e\langle \mathbf{z}_1^2, \mathbf{z}_0^3\rangle\ \mathtt{for}\ e_0\langle \mathbf{z}_1^4, \mathbf{z}_0^3\rangle\ \mathtt{in}\ !_{\mathbf{z}^5<1}(\mathtt{derelict}\ e_0\ \mathtt{for}\ e_1\langle \mathbf{z}_0^3\rangle$
$\mathtt{in}\ t\langle 0\rangle\ e_1)\ )[T_0/e])[s'\langle 1\rangle/g_3]$

apply "SP!P"

$\triangleright \quad (\ g_3\ (\ !_{\mathbf{z}^5<1}(\mathtt{derelict}\ e_0\ \mathtt{for}\ e_1\langle \mathbf{z}_0^3\rangle\ \mathtt{in}\ (t\langle 0\rangle\ e_1)\ )[T_0/e_0])\ )[s'\langle 1\rangle/g_3]$

apply "S⊸", "S⊸", and "SP!P"

$\rightsquigarrow \quad !_{\mathbf{z}^{11}<1}(\ \mathtt{derelict}\ !_{\mathbf{z}^5<1}(\ \mathtt{derelict}\ T_0\ \mathtt{for}\ e_1\langle \mathbf{z}_0^3\rangle\ \mathtt{in}\ (t\langle 0\rangle\ e_1)\ )$
$\mathtt{for}\ e_4\langle \mathbf{z}_1^{12}\rangle\ \mathtt{in}\ (t\langle 1\rangle\ e_4)\ )$

which is an irreducible term.

Chapter 5

Properties of the BLC

In this chapter, some interesting theoretical properties of the bounded linear calculus will be described. All of these properties are "inherited" from either BLL or the linear term calculus.

5.1 Type Reconstruction Algorithm

As was noted in Benton *et al.* (1992), there exists a type reconstruction algorithm which is sound and complete for the term assignment to ILL (without the quantifier rules), based on the algorithm given in Mackie (1991). In this section, I will show how to extend this algorithm for bounded linear terms.

5.1.1 Bounded Linear Natural Deduction

First, it will be necessary to define a system of "bounded linear natural deduction" (BLND), along with a term assignment system (Figure 5.1). Here, the notation of Mackie (1991) is used: $\Gamma|\Theta \vdash t : A$ stands for $\Gamma\backslash\Theta \vdash t : A$, where Γ and Θ are multisets and $\Theta \subseteq \Gamma$. Intuitively, Θ represents the "unused" portion of the type assignments in Γ, *i.e.*, when giving a type to t using the assumptions in Γ, the leftover assumptions are returned in Θ. Also, I have included the quantifier rules even though the algorithm will not deal with formulas containing quantifiers, for completeness.

Id $\quad (\Gamma, x\langle \bar{\mathbf{y}}; \bar{\mathbf{x}}\rangle : A)|\Gamma \vdash x\langle \bar{\mathbf{y}}'; \bar{\mathbf{x}}'\rangle : A' \quad (A \sqsubseteq A' \ , \ \bar{\mathbf{y}} \sqsubseteq \bar{\mathbf{y}}')$

where $\bar{\mathbf{y}}$ (respectively $\bar{\mathbf{y}}'$) is the set of all bound resource variables and $\bar{\mathbf{x}}$ ($\bar{\mathbf{x}}'$) is the set of all free resource variables of A (A')

$$\otimes_{\mathcal{I}} \quad \frac{\Gamma|\Delta \vdash t\langle \bar{\mathbf{y}}_0; \bar{\mathbf{x}}_0\rangle : A \quad \Delta|\Theta \vdash u\langle \bar{\mathbf{y}}_1; \bar{\mathbf{x}}_1\rangle : B}{\Gamma|\Theta \vdash (t \otimes u)\langle \bar{\mathbf{y}}_0, \bar{\mathbf{y}}_1; \bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1\rangle : A \otimes B}$$

$$\otimes_{\mathcal{E}} \quad \frac{\begin{array}{c}\Gamma|\Delta \vdash u\langle \bar{\mathbf{y}}_0', \bar{\mathbf{y}}_1'; \bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1\rangle : A \otimes B \\ (\Delta, x\langle \bar{\mathbf{y}}_0; \bar{\mathbf{x}}_0\rangle : A, y\langle \bar{\mathbf{y}}_1; \bar{\mathbf{x}}_1\rangle : B)|\Theta \vdash t : C\end{array}}{\Gamma|\Theta \vdash (\mathtt{let}\ u\ \mathtt{be}\ x \otimes y\ \mathtt{in}\ t) : C}$$

where $\bar{\mathbf{y}}_0' \equiv \bar{\mathbf{y}}_0$ and $\bar{\mathbf{y}}_1' \equiv \bar{\mathbf{y}}_1$

$$\multimap_{\mathcal{I}} \quad \frac{(\Gamma, x\langle \bar{\mathbf{y}}_0; \bar{\mathbf{x}}_0\rangle : A)|\Theta \vdash t\langle \bar{\mathbf{y}}_1, (\bar{\mathbf{y}}_0); \bar{\mathbf{x}}_1, (\bar{\mathbf{x}}_0)\rangle : B}{\Gamma|\Theta \vdash (\lambda x.t)\langle \bar{\mathbf{y}}_0, \bar{\mathbf{y}}_1; \bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1\rangle : A \multimap B}$$

$$\multimap_{\mathcal{E}} \quad \frac{\Gamma|\Delta \vdash t\langle \bar{\mathbf{y}}_0, \bar{\mathbf{y}}_1; \bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1\rangle : A \multimap B \quad \Delta|\Theta \vdash u\langle \bar{\mathbf{y}}_0'; \bar{\mathbf{x}}_0\rangle : A}{\Gamma|\Theta \vdash (t\ u)\langle \bar{\mathbf{y}}_1; \bar{\mathbf{x}}_1\rangle : B}$$

where $\bar{\mathbf{y}}_0' \equiv \bar{\mathbf{y}}_0$

$$\forall_{\mathcal{I}} \quad \frac{\Gamma|\Theta \vdash t : A}{\Gamma|\Theta \vdash t : (\forall\alpha)A} \quad (\alpha \text{ not free in } \Gamma)$$

$$\forall_{\mathcal{E}} \quad \frac{\Gamma|\Theta \vdash t\langle \bar{\mathbf{y}}; \bar{\mathbf{x}}\rangle : (\forall\alpha)A}{\Gamma|\Theta \vdash t\langle \bar{\mathbf{y}}, \bar{\mathbf{y}}'; \bar{\mathbf{x}}, \bar{\mathbf{x}}'\rangle : A[\alpha := T]}$$

where $\bar{\mathbf{y}}'$ and $\bar{\mathbf{x}}'$ are all the new resource variables introduced in T

Figure 5.1: Term Assignment for BLND ("ordinary" rules)

$$!W \quad \frac{\Gamma|\Delta \vdash a\langle \mathbf{y}_R, \bar{\mathbf{y}}; \bar{\mathbf{x}}, \mathcal{V}(R)\rangle : !_{\mathbf{y}<R}A \quad \Delta|\Theta \vdash t : B}{\Gamma|\Theta \vdash (\mathtt{discard}\ a\ \mathtt{in}\ t) : B}$$

where $\bar{\mathbf{x}}$ is the set of all free resource variables of A, except $\mathbf{y}$

$$!D \quad \frac{\begin{array}{l}\Gamma|\Delta \vdash u\langle \mathbf{y}_{1+R}, \bar{\mathbf{y}}'; \bar{\mathbf{x}}, \mathcal{V}(R)\rangle : !_{\mathbf{y}<1+R}A \\ (\Delta, x\langle \bar{\mathbf{y}}; \bar{\mathbf{x}}\rangle : A[\mathbf{y} := 0])|\Theta \vdash t : B\end{array}}{\Gamma|\Theta \vdash (\mathtt{derelict}\ u\ \mathtt{for}\ x\ \mathtt{in}\ t) : B}$$

where $\bar{\mathbf{y}}' \equiv \bar{\mathbf{y}}$ and $\bar{\mathbf{x}}$ is the set of all free resource variables of A, except $\mathbf{y}$

$$!C \quad \frac{\begin{array}{l}\Gamma|\Delta \vdash u\langle \mathbf{x}^2_{P+Q+R}, \bar{\mathbf{y}}'; \bar{\mathbf{x}}, \mathcal{V}(P+Q+R)\rangle : !_{\mathbf{x}^2<P+Q+R}A \\ (\Delta, x\langle \mathbf{x}^0_P, \bar{\mathbf{y}}_0; \bar{\mathbf{x}}, \mathcal{V}(P)\rangle : !_{\mathbf{x}^0<P}A[\mathbf{x}^2 := \mathbf{x}^0], \\ y\langle \mathbf{x}^1_Q, \bar{\mathbf{y}}_1; \bar{\mathbf{x}}, \mathcal{V}(Q), (\mathcal{V}(P))\rangle : !_{\mathbf{x}^1<Q}A[\mathbf{x}^2 := \mathbf{x}^1 + P])|\Theta \vdash t : B\end{array}}{\Gamma|\Theta \vdash (\mathtt{copy}\ u\ \mathtt{as}\ x, y\ \mathtt{in}\ t) : B}$$

where $\bar{\mathbf{y}}' \equiv \bar{\mathbf{y}}_0 \equiv \bar{\mathbf{y}}_1$, $\mathcal{V}(P)$ appears in u iff $\mathbf{x}^2$ occurs free in A, and $\bar{\mathbf{x}}$ is the set of all free resource variables of A, except $\mathbf{x}^2$

$$P! \quad \frac{\begin{array}{l}\Delta|\Delta_1 \vdash u_1\langle \mathbf{y}^1_{V_1(P)+R_1}, \bar{\mathbf{y}}_1; \bar{\mathbf{x}}_1, \mathcal{V}(R_1), (\mathcal{V}(P))\rangle : !_{\mathbf{y}^1<V_1(P)+R_1}A_1 \\ \quad\vdots \\ \Delta_{n-1}|\Delta_n \vdash u_n\langle \mathbf{y}^n_{V_n(P)+R_n}, \bar{\mathbf{y}}_n; \bar{\mathbf{x}}_n, \mathcal{V}(R_n), (\mathcal{V}(P))\rangle : !_{\mathbf{y}^n<V_n(P)+R_n}A_n \\ (\Delta_n, \Theta)|\Delta_n \vdash t\langle \bar{\mathbf{y}}; \bar{\mathbf{x}}, (\mathbf{x})\rangle : B\end{array}}{\begin{array}{l}\Delta|\Delta_n \\ \vdash (\mathtt{promote} \ldots, u_i, \ldots \mathtt{for} \ldots, x_i, \ldots \mathtt{in}\ !_{\mathbf{x}<P}t)\langle \mathbf{x}_P, \bar{\mathbf{y}}; \bar{\mathbf{x}}, \mathcal{V}(P)\rangle : !_{\mathbf{x}<P}B\end{array}}$$

where $\Theta = \ldots, x_i\langle \mathbf{w}^i_{Q_i(\mathbf{x})}, \bar{\mathbf{y}}_i; \bar{\mathbf{x}}_i, (\mathbf{x})\rangle : !_{\mathbf{w}^i<Q_i(\mathbf{x})}A_i[\mathbf{y}^i := V_i(\mathbf{x}) + \mathbf{w}^i], \ldots$, $V_i(\mathbf{x}) = \sum_{\mathbf{z}<\mathbf{x}} Q_i(\mathbf{z})$, $\mathcal{V}_i(\mathbf{x}) + \mathbf{w}^i$ is free for $\mathbf{y}^i$ in A_i, $\bar{\mathbf{x}}_i$ is the set of all free resource variables of A_i, except $\mathbf{y}^i$, $\mathbf{x}$ is the set of all free resource variables of B, except $\mathbf{x}$, $\mathcal{V}(P)$ appears in u_i iff $V_i(\mathbf{x})$ is not constant, $\mathbf{x}$ appears in x_i iff $Q_i(\mathbf{x})$ is not constant or $V_i(\mathbf{x})$ is not constant and $\mathbf{y}^i$ occurs free in A_i, and $1 \leq i \leq n$

Figure 5.1: Term Assignment for BLND ("storage" rules)

This system is a straightforward generalization of the corresponding linear natural deduction for ILL, and so enjoys the substitution property. Also, the transformation procedures given in Section 5 of Benton *et al.* (1992) along with Theorems 3 and 4 of that paper still hold for the present system (*i.e.*, the term assignment for BLL is equivalent to the term assignment for BLND).

5.1.2 Substitutions and Notation

Before presenting the algorithms to be used in this section, I will define a few useful notions and introduce some new notation. The definition of formulas of BLL will be extended slightly by allowing the use of propositional variables of undefined arity (which we will denote by a propositional letter with no arity superscript and no list of resource parameters). All formulas in what follows are formulas of BLL.

- "*list*" is a function which, given as input a set of resource variables, returns the corresponding sorted list.
- Given a resource polynomial P, $\mathcal{V}(P)$ is the set of all resource variables occurring in P.
- Given a resource polynomial P and two sets of resource variables $\bar{\mathbf{z}}$ and $\bar{\mathbf{x}}$ of the same cardinality, $P[\bar{\mathbf{x}} := \bar{\mathbf{z}}]$ is the resource polynomial obtained by replacing, in P, each variable of $\bar{\mathbf{x}}$ with the corresponding variable in $\bar{\mathbf{z}}$, where the correspondence is established through the *list* function.
- The *resource arity* of a formula A (denoted $\#A$) is the number of distinct free resource variables which occur in the formula.
- Given a formula A, "$\mathcal{R}(A)$" denotes the sorted list of all free resource variables of A.
- "$\mathcal{I}$" is the identity substitution.
- Given a formula A and any two symbols $\triangle_1$ and $\triangle_2$, $A[\triangle_1 \mapsto \triangle_2]$ is the formula obtained by replacing all occurrences of $\triangle_1$ by $\triangle_2$ in A. We extend the notation

to strings of symbols, lists of symbols, and lists of strings of symbols in the obvious ways.

- (Exception to the preceding definition.) Given a formula A, a propositional variable of undefined arity α and a propositional variable $\beta^n(\bar{P})$, $A[\alpha \mapsto \beta^n(\bar{P})]$ is the formula obtained by replacing all occurrences of $(\forall\alpha)$ with $(\forall\beta)$ and all other occurrences of α by $\beta^n(\bar{P})$ in A.

- Given a formula A, a list of resource variables $\bar{\mathbf{y}}$ of length n, and a list of strings of symbols Δ of length n, $A[\bar{\mathbf{y}} := \Delta]$ is the formula obtained by replacing, in A, each free occurrence of a variable of $\bar{\mathbf{y}}$ by the corresponding string in Δ. If Δ consists of a list of resource polynomials, bound resource variables in A are renamed when needed to avoid conflict. If $\bar{\mathbf{y}}$ is a *set* of resource variables, it is ordered using the *list* function before applying the substitution.

- A *resource substitution* is of the form $[\bar{\mathbf{z}} := \bar{\mathbf{x}}]$ where $\bar{\mathbf{x}}$ and $\bar{\mathbf{z}}$ are sets of resource variables of the same cardinality; its effect on a formula A is to replace, in A, each free occurrence of a variable of $\bar{\mathbf{z}}$ with the corresponding one in $\bar{\mathbf{x}}$, where the correspondence is established through the *list* function.

- A *formula substitution* is of the form $[A[\mathcal{R}(A) := \Delta]/\alpha^n(\Delta)]$ (or of the form $[A[\mathcal{R}(A) := \Delta]/\alpha^{n+1}(\Delta_1, \mathbf{y}, \Delta_2)]$), where $\bar{\mathbf{y}}$ is any resource variable not occurring free in A, $\Delta_1 = \vartriangle_1, \ldots, \vartriangle_s$, $\Delta_2 = \vartriangle_{s+1}, \ldots, \vartriangle_n$, and $\Delta = \Delta_1, \Delta_2$ is a list of tokens used as "place-holders" for the variables in $\mathcal{R}(A)$; its effect on a formula C is to replace, in C, all free occurrences of $\alpha^n(\bar{P})$ (respectively, $\alpha^{n+1}(\bar{P}_1, \mathbf{z}, \bar{P}_2)$) with $A[\mathcal{R}(A) := \bar{P}]$ $(A[\mathcal{R}(A) := (\bar{P}_1, \bar{P}_2)])$.

- A *substitution* is either a resource substitution, a formula substitution, or a composition of two other substitutions. Composition of substitutions is denoted by juxtaposition and is read from right to left, *i.e.*, given two substitutions σ and μ, their composition is denoted $\sigma\mu$ and $(\sigma\mu)A = \sigma(\mu A)$ for any formula A.

5.1.3 Unification

For the type reconstruction algorithm, we will have need of a unification algorithm for bounded linear types (formulas). Before presenting the algorithm itself, which is based on the well known algorithm discussed in Field & Harrison (1988), we will define the notion of "disagreement pair" for two formulas of BLL. Informally, the disagreement pair of two formulas is the first place in the formulas where they differ. Formally, the disagreement pair of two formulas A and B is computed by the following algorithm which outputs the "type" of disagreement and a pair of subformulas of A and B which disagree, given A and B as inputs.

Algorithm D: The following cases are checked one by one in the indicated order. The value returned by the algorithm is given by the first case which matches the input.

0. $D(A, A) = (\text{none}, \epsilon, \epsilon)$, where ϵ represents the "empty" formula.

1. $\begin{cases} \text{(a)} \quad D(\alpha, A) = (\text{simple}, \alpha, A) \\ \text{(b)} \quad D(A, \alpha) = (\text{simple}, \alpha, A) \end{cases}$

2. $\begin{cases} \text{(a)} \quad D(\alpha^n(\bar{P}), A) = (\text{resource}, \alpha^n(\bar{P}), A) \\ \text{(b)} \quad D(A, \alpha^n(\bar{P})) = (\text{resource}, \alpha^n(\bar{P}), A) \end{cases}$

3. $D((\forall\alpha)A, (\forall\alpha)A') = D(A, A')$

4. $D((\forall\alpha)A, (\forall\beta)A') = (\text{quantifier}, (\forall\alpha)A, (\forall\beta)A')$

5. $D(!_{\mathbf{x}<P}A, !_{\mathbf{x}<P}A') = D(A, A')$

6. $D(!_{\mathbf{x}<P}A, !_{\mathbf{y}<Q}A') = (\text{storage}, !_{\mathbf{x}<P}A, !_{\mathbf{y}<Q}A')$

7. $D(A \otimes B, A \otimes B') = D(B, B')$

8. $D(A \otimes B, A' \otimes B') = D(A, A')$

9. $D(A \multimap B, A \multimap B') = D(B, B')$

10. $D(A \multimap B, A' \multimap B') = D(A, A')$

11. $D(A, B) = (\text{FAIL}, A, B)$ (If no other case matched the inputs, it means the main connectives of A and B are different and they cannot be unified.)

Obviously, the algorithm's running time is linear in the lengths of A and B since each recursive call is on a proper subformula of one of the inputs, with no repetitions (*i.e.*, during one call to $D(A, B)$, a given subformula of A or B will never get examined twice).

The unification algorithm will be given as input a substitution S, two formulas A and B to unify, and a resource variable $\mathbf{y}$ (which may be the "empty" variable ε). The substitution S is used as an accumulating parameter; in any normal call to the algorithm, S will initially be the identity substitution. The input parameter $\mathbf{y}$ identifies a resource variable which may be discarded in some formula substitutions; it will be used by some cases in the reconstruction algorithm. The unification algorithm will return a substitution S' which is a most general unifier for A and B (up to renaming of bound resource variables and bound propositional variables) and is an "instance" of S (*i.e.*, there exists a substitution T such that $S' = TS$), if such a substitution exists; otherwise, it will fail.

Algorithm *unify*$(S, A, B, \mathbf{y})$:

- if $SA = SB$ (up to renaming of the bound resource and propositional variables of A and B), then return(S) else let $(t, u_1, u_2) = D(SA, SB)$, do:

  1. case $t = \text{FAIL} : \text{FAIL}$
  2. case $t = \text{simple} :$ let $u_1 = \alpha$, $u_2 = C$; return *unify*$([C/\alpha]S, A, B, \mathbf{y})$
  3. case $t = \text{quantifier} :$ let $u_1 = (\forall\alpha)C$, $u_2 = (\forall\beta)C'$
     - return *unify*$(S, A, B, \mathbf{y})$ (u_1 and u_2 are identical except for the name of one bound propositional variable)
  4. case $t = \text{storage} :$ let $u_1 = !_{\mathbf{x}<P}C$, $u_2 = !_{\mathbf{z}<Q}C'$
     - if $P \not\equiv Q$ then FAIL
     - else return *unify*$([\mathcal{V}(Q) := \mathcal{V}(P)]S, A, B, \mathbf{y})$

5. case $t =$ resource : let $u_1 = \alpha^n(\bar{P})$, $u_2 = C$
   - if all the propositional variables in C have undefined arity, then do:
     (a) let $\Delta = \triangle_1, \ldots, \triangle_n$
     (b) let $\alpha_1, \ldots, \alpha_k$ be all the propositional variables of C (both free and bound)
     (c) let $C' = C[\alpha_1 \mapsto \alpha_1^n(\Delta), \ldots, \alpha_k \mapsto \alpha_k^n(\Delta)]$
         and $\sigma = [\alpha_1 \mapsto \alpha_1^n(\bar{P}), \ldots, \alpha_k \mapsto \alpha_k^n(\bar{P})]$
     (d) return *unify* $(\sigma[C'/\alpha^n(\Delta)]S, A, B, \mathbf{y})$
   - else, do:
     (a) if $\mathbf{y} \neq \varepsilon$ and $\bar{P} = (\bar{P}_1, \mathbf{y}, \bar{P}_2)$, then
       * let $\Delta_1 = \triangle_1^1, \ldots, \triangle_{|\bar{P}_1|}^1$, $\Delta_2 = \triangle_1^2, \ldots, \triangle_{|\bar{P}_2|}^2$
       * if $\mathcal{R}(C) = (\mathcal{R}_1, \mathbf{y}, \mathcal{R}_2)$, then
         i. if $\#C \neq n$ then FAIL
         ii. else let $\Delta = \Delta_1, \triangle_0, \Delta_2$;
             return *unify* $([C[\mathcal{V}(C) := \Delta]/\alpha^n(\Delta)]\, S, A, B, \mathbf{y})$
       * else,
         i. if $\#C \neq n - 1$ then FAIL
         ii. else let $\Delta = \Delta_1, \Delta_2$;
             return *unify* $([C[\mathcal{V}(C) := \Delta]/\alpha^n(\Delta_1, \mathbf{y}, \Delta_2)]\, S, A, B, \mathbf{y})$
     (b) else,
       * if $\#C \neq n$ then FAIL
       * else let $\Delta = \triangle_1, \ldots, \triangle_n$;
         return *unify* $([C[\mathcal{V}(C) := \Delta]/\alpha^n(\Delta)]\, S, A, B, \mathbf{y})$

It is easy to see that the algorithm runs in polynomial time in the lengths of its inputs, because at every call, a different subformula of A and B gets unified, and because of the complexity of $D(A, B)$.

5.1.4 The Reconstruction Algorithm

The type reconstruction algorithm will be given as input a multiset $\mathcal{A}$ of type assignments for the free variables of the second argument, a bounded linear type e. It will return a BLL type A which is the most general type of e, or "FAIL". The type returned by the algorithm is the most general in the following sense: if the algorithm returns type A for term e, then for every type B such that e can be given type B, there exists a substitution σ such that $B = \sigma A$.

Unfortunately, the algorithm cannot deal with quantifiers since they are not reflected in any way in the terms. This somewhat limits its usefulness in that for a given term t, the algorithm cannot recover any valid type A containing quantifiers even though it can recover a type A' which can be unified with $\tilde{A}$ (where $\tilde{A}$ is A with all quantifiers removed). Fortunately, this does not affect any of the results presented later in this chapter, where we assume that we are always given typed terms $t : A$ (*i.e.*, the types of the terms are always explicitly given).

Throughout this section, it is understood that when a failure occurs at one step in the algorithm, such as during a call to the unification algorithm, the reconstruction fails (this will not be dealt with explicitly, for conciseness).

Algorithm *reconstruct*$(\mathcal{A}, e)$:

1. let $(\sigma, A, \mathcal{A}') = \mathcal{L}(\mathcal{A}, e)$
2. if $\mathcal{A}' \neq \varnothing$ then FAIL else return A

Algorithm $\mathcal{L}$ performs all the actual work. Given a multiset of type assignments $\mathcal{A}$ and a bounded linear term e as input, it outputs a substitution σ, a formula A, and a multiset $\mathcal{A}'$ which is a subset of $\mathcal{A}$ such that $\sigma(\mathcal{A}|\mathcal{A}') \vdash e : A$ is a valid derivation in BLND (where $\sigma(\mathcal{A}|\mathcal{A}') = (\sigma\mathcal{A}|\sigma\mathcal{A}')$).

Note that algorithm *reconstruct* fails if the set of type assignments returned by $\mathcal{L}$ is non-empty, because that indicates some non-linearity in the term (*i.e.*, not all type assignments in $\mathcal{A}$ were used to derive the type of e).

In what follows, we will use the notation "$\uplus$" to denote multiset union. Also, no propositional variable can have arity zero so when this happens, it is understood to mean that the variable has undefined arity.

Algorithm $\mathcal{L}(\mathcal{A}, e)$:

1. case $e = x_i\langle\bar{\mathbf{y}}_i;\bar{\mathbf{x}}_i\rangle$:
   (a) if $(x_i\langle\bar{\mathbf{y}}_i';\bar{\mathbf{x}}_i'\rangle : A') \in \mathcal{A}$ and $\bar{\mathbf{y}}_i' \sqsubseteq \bar{\mathbf{y}}_i$ then
   return $(\mathcal{I}, A, \mathcal{A}\backslash\{x_i\langle\bar{\mathbf{y}}_i';\bar{\mathbf{x}}_i'\rangle : A'\})$, where $A \sqsupseteq A'$ and the resource information in A (respectively A') corresponds to $\bar{\mathbf{y}}_i, \bar{\mathbf{x}}_i$ $(\bar{\mathbf{y}}_i', \bar{\mathbf{x}}_i')$ (extra variables in $\bar{\mathbf{x}}_i$ are simply added to each propositional variable of A).
   (b) else FAIL
2. case $e = t \otimes u$:
   (a) let $(\sigma, A, \mathcal{A}_1) = \mathcal{L}(\mathcal{A}, t)$
   (b) let $(\eta, B, \mathcal{A}_2) = \mathcal{L}(\sigma\mathcal{A}_1, u)$
   (c) return $(\eta\sigma, \eta(A \otimes B), \mathcal{A}_2)$
3. case $e =$ `let` $u\langle\bar{\mathbf{y}};\bar{\mathbf{x}}\rangle$ `be` $x\langle\bar{\mathbf{y}}_0;\bar{\mathbf{x}}_0\rangle \otimes y\langle\bar{\mathbf{y}}_1;\bar{\mathbf{x}}_1\rangle$ `in` t:
   (a) let $\bar{\mathbf{w}}_0 = \mathit{list}(\bar{\mathbf{x}}_0)$, $\bar{\mathbf{w}}_1 = \mathit{list}(\bar{\mathbf{x}}_1)$, $n_0 = |\bar{\mathbf{w}}_0|$, $n_1 = |\bar{\mathbf{w}}_1|$,
   and $\alpha^{n_0}(\bar{\mathbf{w}}_0), \beta^{n_1}(\bar{\mathbf{w}}_1)$ be new propositional variables
   (b) let $(\sigma, A, \mathcal{A}_1) = \mathcal{L}(\mathcal{A}, u)$
   (c) let $\mu = \mathit{unify}(\mathcal{I}, \alpha^{n_0}(\bar{\mathbf{w}}_0) \otimes \beta^{n_1}(\bar{\mathbf{w}}_1), A, \varepsilon)$
   (d) let $(\eta, B, \mathcal{A}_2) = \mathcal{L}(\mu\sigma\,(\mathcal{A}_1 \uplus \{x : \alpha^{n_0}(\bar{\mathbf{w}}_0), y : \beta^{n_1}(\bar{\mathbf{w}}_1)\})\,, t)$
   (e) return $(\eta\mu\sigma, B, \mathcal{A}_2)$
4. case $e = \lambda x\langle\bar{\mathbf{y}}_0;\bar{\mathbf{x}}_0\rangle.t\langle\bar{\mathbf{y}}_1;\bar{\mathbf{x}}_1\rangle$:
   (a) let $\bar{\mathbf{w}}_0 = \mathit{list}(\bar{\mathbf{x}}_0)$, $n_0 = |\bar{\mathbf{w}}_0|$, and $\alpha^{n_0}(\bar{\mathbf{w}}_0)$ be a new propositional variable
   (b) let $(\sigma, A, \mathcal{A}_1) = \mathcal{L}((\mathcal{A} \uplus \{x : \alpha^{n_0}(\bar{\mathbf{w}}_0)\})\,, t)$

(c) return $(\sigma, \sigma(\alpha^{n_0}(\bar{\mathbf{w}}_0) \multimap A), \mathcal{A}_1)$

5. case $e = (t\ u)$:

   (a) let α be a new propositional variable

   (b) let $(\sigma, A, \mathcal{A}_1) = \mathcal{L}(\mathcal{A}, t)$

   (c) let $(\eta, B, \mathcal{A}_2) = \mathcal{L}(\sigma\mathcal{A}_1, u)$

   (d) let $\mu = \mathit{unify}(\mathcal{I}, \eta A, B \multimap \alpha, \varepsilon)$

   (e) return $(\mu\eta\sigma, \mu\alpha, \mathcal{A}_2)$

6. case $e = \mathtt{discard}\ u\langle \mathbf{y}_R, \bar{\mathbf{y}}; \bar{\mathbf{x}}, \mathcal{V}(R)\rangle\ \mathtt{in}\ t$:

   (a) let $\bar{\mathbf{w}} = \mathit{list}(\mathbf{y}, \bar{\mathbf{x}})$, $n = |\bar{\mathbf{w}}|$, and $\alpha^n(\bar{\mathbf{w}})$ be a new propositional variable

   (b) let $(\sigma, A, \mathcal{A}_1) = \mathcal{L}(\mathcal{A}, u)$

   (c) let $\mu = \mathit{unify}(\mathcal{I}, !_{\mathbf{y}<R}\alpha^n(\bar{\mathbf{w}}), A, \mathbf{y})$

   (d) let $(\eta, B, \mathcal{A}_2) = \mathcal{L}(\mu\sigma\mathcal{A}_1, t)$

   (e) return $(\eta\mu\sigma, B, \mathcal{A}_2)$

7. case $e = \mathtt{derelict}\ u\langle \mathbf{y}_{1+R}, \bar{\mathbf{y}}'; \bar{\mathbf{x}}, \mathcal{V}(R)\rangle\ \mathtt{for}\ x\langle \bar{\mathbf{y}}; \bar{\mathbf{x}}\rangle\ \mathtt{in}\ t$:

   (a) let $\bar{\mathbf{w}} = \mathit{list}(\mathbf{y}, \bar{\mathbf{x}})$, $n = |\bar{\mathbf{w}}|$, and $\alpha^n(\bar{\mathbf{w}})$ be a new propositional variable

   (b) let $(\sigma, A, \mathcal{A}_1) = \mathcal{L}(\mathcal{A}, u)$

   (c) let $\mu = \mathit{unify}(\mathcal{I}, !_{\mathbf{y}<1+R}\alpha^n(\bar{\mathbf{w}}), A, \mathbf{y})$

   (d) let $(\eta, B, \mathcal{A}_2) = \mathcal{L}\left(\mu\sigma\left(\mathcal{A}_1 \uplus \{x : \alpha^n(\bar{\mathbf{w}})[\mathbf{y} := 0]\}\right), t\right)$

   (e) return $(\eta\mu\sigma, B, \mathcal{A}_2)$

8. case $e = \mathtt{copy}\ \ u\langle \mathbf{x}^2_{P+Q+R}, \bar{\mathbf{y}}'; \bar{\mathbf{x}}, \mathcal{V}(P+Q+R)\rangle\ \mathtt{as}\ x\langle \mathbf{x}^0_P, \bar{\mathbf{y}}_0; \bar{\mathbf{x}}, \mathcal{V}(P)\rangle$, $y\langle \mathbf{x}^1_Q, \bar{\mathbf{y}}_1; \bar{\mathbf{x}}, \mathcal{V}(Q), (\mathcal{V}(P))\rangle\ \mathtt{in}\ t$:

   (a) let $\bar{\mathbf{w}} = \mathit{list}(\mathbf{x}^2, \bar{\mathbf{x}})$, $n = |\bar{\mathbf{w}}|$, and $\alpha^n(\bar{\mathbf{w}})$ be a new propositional variable

   (b) let $(\sigma, A, \mathcal{A}_1) = \mathcal{L}(\mathcal{A}, u)$

(c) let $\mu = \mathit{unify}(\mathcal{I}, !_{\mathbf{x}^2<P+Q+R}\alpha^n(\bar{\mathbf{w}}), A, \mathbf{x}^2)$

(d) let $(\eta, B, \mathcal{A}_2) =$
$\mathcal{L}\left(\mu\sigma\left(\mathcal{A}_1 \uplus \{x : !_{\mathbf{x}^0<P}\alpha^n(\bar{\mathbf{w}})[\mathbf{x}^2 := \mathbf{x}^0], y : !_{\mathbf{x}^1<Q}\alpha^n(\bar{\mathbf{w}})[\mathbf{x}^2 := \mathbf{x}^1 + P]\}\right), t\right)$

(e) return $(\eta\mu\sigma, B, \mathcal{A}_2)$

9. case $e = \texttt{promote}\ldots, u_i\langle \mathbf{y}^i_{V_i(P)+R_i}, \bar{\mathbf{y}}_i; \bar{\mathbf{x}}_i, \mathcal{V}(R_i), (\mathcal{V}(P))\rangle, \ldots$
$\texttt{for}\ldots, x_i\langle \mathbf{w}^i_{Q_i(\mathbf{x})}, \bar{\mathbf{y}}_i; \bar{\mathbf{x}}_i, (\mathbf{x})\rangle, \ldots \texttt{in}\ !_{\mathbf{x}<P}t$:

   (a) let $\bar{\mathbf{w}}_1 = \mathit{list}(\mathbf{y}^1, \bar{\mathbf{x}}_1)$, $m_1 = |\bar{\mathbf{w}}_1|$, and $\alpha_1^{m_1}(\bar{\mathbf{w}}_1)$ be a new propositional variable

   (b) let $(\sigma_1, A_1, \mathcal{A}_1) = \mathcal{L}(\mathcal{A}, u_1)$

   (c) let $\mu_1 = \mathit{unify}(\mathcal{I}, !_{\mathbf{y}^1<V_1(P)+R_1}\alpha_1^{m_1}(\bar{\mathbf{w}}_1), A_1, \mathbf{y}^1)$

   $\vdots \qquad \vdots$

   (d) let $\bar{\mathbf{w}}_n = \mathit{list}(\mathbf{y}^n, \bar{\mathbf{x}}_n)$, $m_n = |\bar{\mathbf{w}}_n|$, and $\alpha_n^{m_n}(\bar{\mathbf{w}}_n)$ be a new propositional variable

   (e) let $(\sigma_n, A_n, \mathcal{A}_n) = \mathcal{L}\left(\mu_{n-1}\sigma_{n-1}\cdots\mu_1\sigma_1\mathcal{A}_{n-1}, u_n\right)$

   (f) let $\mu_n = \mathit{unify}(\mathcal{I}, !_{\mathbf{y}^n<V_n(P)+R_n}\alpha_n^{m_n}(\bar{\mathbf{w}}_n), A_n, \mathbf{y}^n)$

   (g) let $\sigma' = \mu_n\sigma_n\cdots\mu_1\sigma_1$

   (h) let $(\sigma, B, \mathcal{A}') =$
   $\mathcal{L}\left(\sigma'\left(\mathcal{A}_n \uplus \biguplus_{i=1}^n \left\{x_i : !_{\mathbf{w}^i<Q_i(\mathbf{x})}\alpha_i^{m_i}(\bar{\mathbf{w}}_i)[\mathbf{y}^i := \mathbf{w}^i + V_i(\mathbf{x})]\right\}\right), t\right)$

   (i) if $\sigma'\mathcal{A}_n = \sigma'\mathcal{A}'$ then return $(\sigma\sigma', !_{\mathbf{x}<P}B, \mathcal{A}')$ else FAIL

Note that the algorithm runs in polynomial time in the length of the input term e, since each recursive call is on a proper subterm of e, with no repetitions, and because of the complexity of *unify*.

5.1.5 Completeness and Soundness of the Algorithm

The proofs of soundness and completeness given in Mackie (1991) (Theorems 3.4.2 and 3.5.1, as well as Lemma 3.4.1) can easily be extended to the present version of algorithm $\mathcal{L}$, with only minor changes to reflect the resource information of the terms and a new case to consider because our promotion rule differs from his (the proofs are all by induction on the structure of the term e given as input). Formally, we have the following lemma and two theorems.

Lemma 5.1 *If there is a valid derivation of $\mathcal{A}|\mathcal{A}' \vdash e : A$ which does not use the quantifier rules, then for any substitution σ, there is also a valid derivation of $\sigma(\mathcal{A}|\mathcal{A}') \vdash e : \sigma(A)$.*

Proof: (for the promotion case only)
If $e = \texttt{promote}\ldots, u_i\langle \mathbf{y}^i_{V_i(P)+R_i}\rangle, \ldots \texttt{for} \ldots, x_i, \ldots \texttt{in}\ !_{\mathbf{x}<P}t$ and we have a derivation $\mathcal{A}|\mathcal{A}_n \vdash e : !_{\mathbf{x}<P}B$, then the last rule used in that derivation was the promotion rule so we know there exist derivations

1. $\mathcal{A}|\mathcal{A}_1 \vdash u_1 : !_{\mathbf{y}^1<V_1(P)+R_1}A_1$

$\vdots \qquad \vdots$

n. $\mathcal{A}_{n-1}|\mathcal{A}_n \vdash u_n : !_{\mathbf{y}^n<V_n(P)+R_n}A_n$

$n+1$. $(\mathcal{A}_n, \mathcal{A}')|\mathcal{A}_n \vdash e : B$, where $\mathcal{A}' = \{\ldots, x_i : !_{\mathbf{w}^i<Q_i(\mathbf{x})}A_i[\ldots], \ldots\}$.

By the induction hypothesis, and since $\sigma(!_{\mathbf{z}<R}A) = !_{\mathbf{z}<R}\sigma(A)$ for any substitution σ, we have the following derivations, for any substitution σ:

1. $\sigma(\mathcal{A}|\mathcal{A}_1) \vdash u_1 : !_{\mathbf{y}^1<V_1(P)+R_1}\sigma(A_1)$

$\vdots \qquad \vdots$

n. $\sigma(\mathcal{A}_{n-1}|\mathcal{A}_n) \vdash u_n : !_{\mathbf{y}^n<V_n(P)+R_n}\sigma(A_n)$

$n+1$. $\sigma(\ (\mathcal{A}_n, \mathcal{A}')|\mathcal{A}_n\ ) \vdash e : \sigma(B)$, where $\mathcal{A}' = \{\ldots, x_i : !_{\mathbf{w}^i<Q_i(\mathbf{x})}\sigma(A_i[\ldots]), \ldots\}$.

which allows us to use the promotion rule and get the derivation that we want: $\sigma(\mathcal{A}|\mathcal{A}_n) \vdash e : \sigma(!_{\mathbf{x}<P}B)$. $\Box$

Theorem 5.2 (Soundness) *If $\mathcal{L}(\mathcal{A}, e)$ succeeds with $(\sigma, A, \mathcal{A}')$, then there is a valid derivation of $\sigma(\mathcal{A}|\mathcal{A}') \vdash e : A$ which does not use the quantifier rules.*

Proof: (for the promotion case only)

If e = $\texttt{promote}\ldots, u_i\langle \mathbf{y}^i_{V_i(P)+R_i}\rangle, \ldots \texttt{for} \ldots, x_i, \ldots \texttt{in}\ !_{\mathbf{x}<P} t$ and algorithm $\mathcal{L}(\mathcal{A}, e)$ succeeds with $(\sigma\sigma', !_{\mathbf{x}<P} A, \mathcal{A}')$, then we know that

1. $\mathcal{L}(\mathcal{A}, u_1)$ succeeded with $(\sigma_1, A_1, \mathcal{A}_1)$

and *unify*$(\mathcal{I}, !_{\mathbf{y}^1<V_1(P)+R_1}\alpha_1^{m_1}(\bar{\mathbf{w}}_1), A_1, \mathbf{y}^1)$ succeeded with μ_1,

$\vdots \qquad \vdots$

n. $\mathcal{L}(\mu_{n-1}\sigma_{n-1}\cdots\mu_1\sigma_1\mathcal{A}_{n-1}, u_n)$ succeeded with $(\sigma_n, A_n, \mathcal{A}_n)$

and *unify*$(\mathcal{I}, !_{\mathbf{y}^n<V_n(P)+R_n}\alpha_n^{m_n}(\bar{\mathbf{w}}_n), A_n, \mathbf{y}^n)$ succeeded with μ_n,

$n+1$. $\mathcal{L}\left(\mu_n\sigma_n\cdots\mu_1\sigma_1\left(\mathcal{A}_n \uplus \biguplus_{i=1}^n \left\{x_i : !_{\mathbf{w}^i<Q_i(\mathbf{x})}\alpha_i^{m_i}(\bar{\mathbf{w}}_i)[\mathbf{y}^i := \mathbf{w}^i + V_i(\mathbf{x})]\right\}\right), e\right)$
succeeded with $(\sigma, A, \mathcal{A}')$

and $\sigma'\mathcal{A}_n = \sigma'\mathcal{A}'$, where $\sigma' = \mu_n\sigma_n\cdots\mu_1\sigma_1$.

By the induction hypothesis, we can conclude that there exist derivations

1. $\sigma_1(\mathcal{A}|\mathcal{A}_1) \vdash u_1 : A_1$

$\vdots \qquad \vdots$

n. $\sigma_n\mu_{n-1}\sigma_{n-1}\cdots\mu_1\sigma_1(\mathcal{A}_{n-1}|\mathcal{A}_n) \vdash u_n : A_n$

$n+1$. $\sigma\sigma'\left(\left(\mathcal{A}_n \uplus \biguplus_{i=1}^n\{x_i : !_{\mathbf{w}^i<Q_i(\mathbf{x})}\alpha_i^{m_i}(\bar{\mathbf{w}}_i)[\mathbf{y}^i := \mathbf{w}^i + V_i(\mathbf{x})]\}\right)|\mathcal{A}'\right) \vdash e : A.$

By Lemma 5.1, and because $\sigma'\mathcal{A}_n = \sigma'\mathcal{A}'$, we can then say that there exist derivations

1. $\sigma\sigma'(\mathcal{A}|\mathcal{A}_1) \vdash u_1 : \sigma\mu_n\sigma_n\cdots\mu_1 A_1$

$\vdots \qquad \vdots$

n. $\sigma\sigma'(\mathcal{A}_{n-1}|\mathcal{A}') \vdash u_n : \sigma\mu_n A_n$

$n+1$. $\sigma\sigma'\left(\left(\mathcal{A}' \uplus \biguplus_{i=1}^n\{x_i : !_{\mathbf{w}^i<Q_i(\mathbf{x})}\alpha_i^{m_i}(\bar{\mathbf{w}}_i)[\mathbf{y}^i := \mathbf{w}^i + V_i(\mathbf{x})]\}\right)|\mathcal{A}'\right) \vdash e : A.$

But then, we can apply the promotion rule and get the derivation we need, *i.e.*, $\sigma\sigma'(\mathcal{A}|\mathcal{A}') \vdash e : !_{\mathbf{x}<P} A$. $\square$

Theorem 5.3 (Completeness) *If, for some substitution η, there is a valid derivation of $\eta(\mathcal{A}|\mathcal{A}') \vdash e : A$ which does not use the quantifier rules, then $\mathcal{L}(\mathcal{A}, e)$ succeeds with $(\sigma, A', \mathcal{A}')$ and there exists a substitution ν such that $\nu\sigma(\mathcal{A}|\mathcal{A}') = \eta(\mathcal{A}|\mathcal{A}')$ and $\nu A' = A$.*

Proof: (for the promotion case only)
If $e = \texttt{promote}\ldots, u_i\langle \mathbf{y}^i_{V_i(P)+R_i}\rangle, \ldots \texttt{for} \ldots, x_i, \ldots \texttt{in}\ !_{\mathbf{x}<P} t$ and there exists a derivation of $\eta(\mathcal{A}|\mathcal{A}_n) \vdash e : !_{\mathbf{x}<P} B$ for some substitution η, then the last rule used in that derivation was the promotion rule so we know there exist derivations

1. $\eta(\mathcal{A}|\mathcal{A}_1) \vdash u_1 : !_{\mathbf{y}^1<V_1(P)+R_1} A_1$

$\vdots \quad \vdots$

n. $\eta(\mathcal{A}_{n-1}|\mathcal{A}_n) \vdash u_n : !_{\mathbf{y}^n<V_n(P)+R_n} A_n$

$n+1$. $\eta(\ (\mathcal{A}_n, \mathcal{A}')|\mathcal{A}_n\ ) \vdash e : B$, where $\mathcal{A}' = \{\ldots, x_i : !_{\mathbf{w}^i<Q_i(\mathbf{x})} A_i[\ldots], \ldots\}$.

By the induction hypothesis, we can conclude that

1. $\mathcal{L}(\mathcal{A}, u_1)$ will succeed with $(\sigma_1, A'_1, \mathcal{A}_1)$ and there will exist a substitution ν_1 such that $\nu_1\sigma_1(\mathcal{A}|\mathcal{A}_1) = \eta(\mathcal{A}|\mathcal{A}_1)$ and $\nu_1 A'_1 = !_{\mathbf{y}^1<V_1(P)+R_1} A_1$

and *unify*$(\mathcal{I}, !_{\mathbf{y}^1<V_1(P)+R_1} \alpha_1^{m_1}(\bar{\mathbf{w}}_1), A'_1, \mathbf{y}^1)$ will succeed with μ_1 since there exists a unifying substitution for the two formulas, namely
$[\ A_1[(\mathcal{V}(A_1) + \mathbf{y}_1) := \Delta]/\alpha_1^{m_1}(\Delta)\ ]\ \nu_1$ (where $\Delta = \triangle_1, \ldots, \triangle_{m_1}$ and $\mathcal{V}(A_1) + \mathbf{y}_1$ is the sorted list containing all the variables of $\mathcal{V}(A_1)$ to which $\mathbf{y}_1$ is added, if it does not already occur),

$\vdots \quad \vdots$

n. $\mathcal{L}(\mu_{n-1}\sigma_{n-1}\cdots\mu_1\sigma_1\mathcal{A}_{n-1}, u_n)$ will succeed with $(\sigma_n, A'_n, \mathcal{A}_n)$ and there will exist a substitution ν_n such that $\nu_n\sigma_n\mu_{n-1}\sigma_{n-1}\cdots\mu_1\sigma_1(\mathcal{A}_{n-1}|\mathcal{A}_n) = \eta(\mathcal{A}_{n-1}|\mathcal{A}_n)$ and $\nu_n A'_n = !_{\mathbf{y}^n<V_n(P)+R_n} A_n$

and $\mathit{unify}(\mathcal{I}, !_{\mathbf{y}^n < V_n(P)+R_n} \alpha_n^{m_n}(\bar{\mathbf{w}}_n), A'_n, \mathbf{y}^n)$ will succeed with μ_n since there exists a unifying substitution for the two formulas, namely
$[\ A_n[(\mathcal{V}(A_n) + \mathbf{y}_n) := \Delta]/\alpha_n^{m_n}(\Delta)\ ]\ \nu_n$ (where $\Delta = \Delta_1, \ldots, \Delta_{m_n}$),

$n+1$. $\mathcal{L}\left(\mu_n\sigma_n \cdots \mu_1\sigma_1 \left(\mathcal{A}_n \uplus \biguplus_{i=1}^n \left\{x_i : !_{\mathbf{w}^i < Q_i(\mathbf{x})} \alpha_i^{m_i}(\bar{\mathbf{w}}_i)[\mathbf{y}^i := \mathbf{w}^i + V_i(\mathbf{x})]\right\}\right), t\right)$ will succeed with $(\sigma, B', \mathcal{A}_n)$ and, letting $\sigma' = \mu_n\sigma_n \cdots \mu_1\sigma_1$, there will exist a substitution τ such that $\tau B' = B$ and $\eta(\ (\mathcal{A}_n, \mathcal{A}')|\mathcal{A}_n\ ) =$
$\tau\sigma\sigma'\left(\left(\mathcal{A}_n \uplus \biguplus_{i=1}^n \left\{x_i : !_{\mathbf{w}^i < Q_i(\mathbf{x})} \alpha_i^{m_i}(\bar{\mathbf{w}}_i)[\mathbf{y}^i := \mathbf{w}^i + V_i(\mathbf{x})]\right\}\right) |\mathcal{A}_n\right)$.

and $\sigma'\mathcal{A}_n = \sigma'\mathcal{A}_n$.

But then, if we let $\nu = \tau$, the algorithm will succeed with $(\sigma\sigma', !_{\mathbf{x}<P} B', \mathcal{A}_n)$ and we will have $\nu\sigma\sigma'(\mathcal{A}|\mathcal{A}_n) = \eta(\mathcal{A}|\mathcal{A}_n)$ and $\nu(!_{\mathbf{x}<P} B') = !_{\mathbf{x}<P} B$, as required. □

5.1.6 Examples

To help the reader understand the functioning of the type reconstruction algorithm and the unification algorithm, here are two examples of its execution. Each call to the routine $\mathcal{L}$ will be numbered as well as each return, so that the execution is easier to follow.

Execution of *reconstruct*(∅, *1*) (where *1* was defined in Chapter 4.)

1. $\mathcal{L}(\ \varnothing, \lambda f\langle \mathbf{y}_1^1\rangle.\lambda x.\texttt{derelict}\ f\langle \mathbf{y}_1^1\rangle\ \texttt{for}\ f_1\ \texttt{in}\ (f_1\ x)\ )$

   1... Case 4:

   (a) α

   (b) $\mathcal{L}(\ \{f\langle \mathbf{y}_1^1\rangle : \alpha\}, \lambda x.\texttt{derelict}\ f\langle \mathbf{y}_1^1\rangle\ \texttt{for}\ f_1\ \texttt{in}\ (f_1\ x)\ )$

   2... Case 4:

   (a) β

   (b) $\mathcal{L}(\ \{f\langle \mathbf{y}_1^1\rangle : \alpha, x : \beta\}, \texttt{derelict}\ f\langle \mathbf{y}_1^1\rangle\ \texttt{for}\ f_1\ \texttt{in}\ (f_1\ x)\ )$

   3... Case 7:

   (a) $\gamma_0^1(\mathbf{y}^1)$

(b) $\mathcal{L}(\{f\langle \mathbf{y}_1^1\rangle : \alpha, x : \beta\}, f\langle \mathbf{y}_1^1\rangle)$

4. Case 1: return $(\mathcal{I}, \alpha, \{x : \beta\})$

(c) $\mathit{unify}(\mathcal{I}, !_{\mathbf{y}^1<1}\gamma_0^1(\mathbf{y}^1), \alpha, \mathbf{y}^1)$

- return $[!_{\mathbf{y}^1<1}\gamma_0^1(\mathbf{y}^1)/\alpha]$

(d) $\mathcal{L}(\ \{x : \beta, f_1 : \gamma_0^1(0)\}, (f_1\ x)\ )$

5... Case 5:

(a) γ_1

(b) $\mathcal{L}(\{x : \beta, f_1 : \gamma_0^1(0)\}, f_1)$

6. Case 1: return $(\mathcal{I}, \gamma_0^1(0), \{x : \beta\})$

(c) $\mathcal{L}(\{x : \beta\}, x)$

7. Case 1: return $(\mathcal{I}, \beta, \varnothing)$

(d) $\mathit{unify}(\mathcal{I}, \gamma_0^1(0), \beta \multimap \gamma_1, \varepsilon)$

- return $\sigma_0 = ([\beta \mapsto \beta^1(0), \gamma_1 \mapsto \gamma_1^1(0)][(\beta^1(\triangle_1) \multimap \gamma_1^1(\triangle_1))/\gamma_0^1(\triangle_1)])$

...**5** return $(\sigma_0, \gamma_1^1(0), \varnothing)$

...**3** return $(\sigma_0[!_{\mathbf{y}^1<1}\gamma_0^1(\mathbf{y}^1)/\alpha], \gamma_1^1(0), \varnothing)$

...**2** return $(\sigma_0[!_{\mathbf{y}^1<1}\gamma_0^1(\mathbf{y}^1)/\alpha], \beta^1(0) \multimap \gamma_1^1(0), \varnothing)$

...**1** return $(\sigma_0[!_{\mathbf{y}^1<1}\gamma_0^1(\mathbf{y}^1)/\alpha], !_{\mathbf{y}^1<1}(\beta^1(\mathbf{y}^1) \multimap \gamma_1^1(\mathbf{y}^1)) \multimap \beta^1(0) \multimap \gamma_1^1(0), \varnothing)$

2. return $!_{\mathbf{y}^1<1}(\beta^1(\mathbf{y}^1) \multimap \gamma_1^1(\mathbf{y}^1)) \multimap \beta^1(0) \multimap \gamma_1^1(0)$, which is indeed the most general type of *1*. Note that we can recover the usual type of *1* (except for the quantifier) with the substitution $[\alpha^1(\triangle)/\beta^1(\triangle), \alpha^1(\triangle +1)/\gamma_1^1(\triangle)]$.

Execution of *reconstruct*($\varnothing$, $t\langle\mathbf{x}\rangle$) (where $t\langle\mathbf{x}\rangle$ was defined in Chapter 4.)

1. $\mathcal{L}(\varnothing, \lambda e\langle \mathbf{y}_{\mathbf{x}}^0;\mathbf{x}\rangle.\lambda f\langle \mathbf{y}_{\mathbf{x}+1}^1;\mathbf{x}\rangle.\lambda x.$`copy` f `as` $f_0\langle \mathbf{y}_{\mathbf{x}}^2;\mathbf{x}\rangle, f_1\langle \mathbf{y}_1^3;\mathbf{x}\rangle$ `in` `derelict` f_1 `for` $f_2\langle;\mathbf{x}\rangle$ `in` $f_2\ (e\ f_0\ x))$

1... Case 4:

(a) $\alpha^1(\mathbf{x})$

(b) $\mathcal{L}(\{e\langle \mathbf{y}^0_{\mathbf{x}};\mathbf{x}\rangle : \alpha^1(\mathbf{x})\}, \lambda f\langle \mathbf{y}^1_{\mathbf{x}+1};\mathbf{x}\rangle.\lambda x.\mathtt{copy}\ f\ \mathtt{as}\ f_0\langle \mathbf{y}^2_{\mathbf{x}};\mathbf{x}\rangle,$
$f_1\langle \mathbf{y}^3_1;\mathbf{x}\rangle\ \mathtt{in}\ \mathtt{derelict}\ f_1\ \mathtt{for}\ f_2\langle;\mathbf{x}\rangle\ \mathtt{in}\ f_2\ (e\ f_0\ x))$

2... Case 4:

(a) $\beta^1(\mathbf{x})$

(b) $\mathcal{L}(\{e\langle \mathbf{y}^0_{\mathbf{x}};\mathbf{x}\rangle : \alpha^1(\mathbf{x}), f\langle \mathbf{y}^1_{\mathbf{x}+1};\mathbf{x}\rangle : \beta^1(\mathbf{x})\}, \lambda x.\mathtt{copy}\ f\ \mathtt{as}\ f_0\langle \mathbf{y}^2_{\mathbf{x}};\mathbf{x}\rangle,$
$f_1\langle \mathbf{y}^3_1;\mathbf{x}\rangle\ \mathtt{in}\ \mathtt{derelict}\ f_1\ \mathtt{for}\ f_2\langle;\mathbf{x}\rangle\ \mathtt{in}\ f_2\ (e\ f_0\ x))$

3... Case 4:

(a) γ

(b) $\mathcal{L}(\{e\langle \mathbf{y}^0_{\mathbf{x}};\mathbf{x}\rangle : \alpha^1(\mathbf{x}), f\langle \mathbf{y}^1_{\mathbf{x}+1};\mathbf{x}\rangle : \beta^1(\mathbf{x}), x:\gamma\}, \mathtt{copy}\ f\langle \mathbf{y}^1_{\mathbf{x}+1};\mathbf{x}\rangle\ \mathtt{as}$
$f_0\langle \mathbf{y}^2_{\mathbf{x}};\mathbf{x}\rangle, f_1\langle \mathbf{y}^3_1;\mathbf{x}\rangle\ \mathtt{in}\ \mathtt{derelict}\ f_1\ \mathtt{for}\ f_2\langle;\mathbf{x}\rangle\ \mathtt{in}\ f_2\ (e\ f_0\ x))$

4... Case 8:

(a) $\delta^2_0(\mathbf{x},\mathbf{y}^1)$

(b) $\mathcal{L}(\{e\langle \mathbf{y}^0_{\mathbf{x}};\mathbf{x}\rangle : \alpha^1(\mathbf{x}), f\langle \mathbf{y}^1_{\mathbf{x}+1};\mathbf{x}\rangle : \beta^1(\mathbf{x}), x:\gamma\}, f\langle \mathbf{y}^1_{\mathbf{x}+1};\mathbf{x}\rangle)$

5. Case 1: return $(\mathcal{I}, \beta^1(\mathbf{x}), \{e\langle \mathbf{y}^0_{\mathbf{x}};\mathbf{x}\rangle : \alpha^1(\mathbf{x}), x:\gamma\})$

(c) $\mathit{unify}(\mathcal{I}, !_{\mathbf{y}^1<\mathbf{x}+1}\delta^2_0(\mathbf{x},\mathbf{y}^1), \beta^1(\mathbf{x}), \mathbf{y}^1)$

- $\mathcal{I}(!_{\mathbf{y}^1<\mathbf{x}+1}\delta^2_0(\mathbf{x},\mathbf{y}^1)) \neq \mathcal{I}(\beta^1(\mathbf{x}))$, so
- $D(!_{\mathbf{y}^1<\mathbf{x}+1}\delta^2_0(\mathbf{x},\mathbf{y}^1), \beta^1(\mathbf{x}))$
  - [case 2(b)] (resource, $\beta^1(\mathbf{x}), !_{\mathbf{y}^1<\mathbf{x}+1}\delta^2_0(\mathbf{x},\mathbf{y}^1)$)
- the propositional variables in $!_{\mathbf{y}^1<\mathbf{x}+1}\delta^2_0(\mathbf{x},\mathbf{y}^1)$ do not all have undefined arity, $\mathbf{y}^1$ does not occur in $\beta^1(\mathbf{x})$, and $\#!_{\mathbf{y}^1<\mathbf{x}+1}\delta^2_0(\mathbf{x},\mathbf{y}^1) = 1$, so
- $\mathit{unify}(\sigma_1 = [!_{\mathbf{y}^1<\triangle_1+1}\delta^2_0(\triangle_1,\mathbf{y}^1)/\beta^1(\triangle_1)], !_{\mathbf{y}^1<\mathbf{x}+1}\delta^2_0(\mathbf{x},\mathbf{y}^1), \beta^1(\mathbf{x}), \mathbf{y}^1)$
  - $\sigma_1(!_{\mathbf{y}^1<\mathbf{x}+1}\delta^2_0(\mathbf{x},\mathbf{y}^1)) = !_{\mathbf{y}^1<\mathbf{x}+1}\delta^2_0(\mathbf{x},\mathbf{y}^1) = \sigma_1(\beta^1(\mathbf{x}))$,
    so return(σ_1)

(d) $\mathcal{L}(\{e\langle \mathbf{y}^0_{\mathbf{x}};\mathbf{x}\rangle : \alpha^1(\mathbf{x}), x:\gamma, f_0\langle \mathbf{y}^2_{\mathbf{x}};\mathbf{x}\rangle : !_{\mathbf{y}^2<\mathbf{x}}\delta^2_0(\mathbf{x},\mathbf{y}^2), f_1\langle \mathbf{y}^3_1;\mathbf{x}\rangle :$
$!_{\mathbf{y}^3<1}\delta^2_0(\mathbf{x},\mathbf{y}^3+\mathbf{x})\}, \mathtt{derelict}\ f_1\langle \mathbf{y}^3_1;\mathbf{x}\rangle\ \mathtt{for}\ f_2\langle;\mathbf{x}\rangle\ \mathtt{in}\ f_2\ (e\ f_0\ x))$

6.... Case 7:

(a) $\delta_1^2(\mathbf{x}, \mathbf{y}^3)$

(b) $\mathcal{L}(\{e\langle \mathbf{y}_\mathbf{x}^0; \mathbf{x}\rangle : \alpha^1(\mathbf{x}), x : \gamma, f_0\langle \mathbf{y}_\mathbf{x}^2; \mathbf{x}\rangle : !_{\mathbf{y}^2 < \mathbf{x}} \delta_0^2(\mathbf{x}, \mathbf{y}^2),$
$f_1\langle \mathbf{y}_1^3; \mathbf{x}\rangle : !_{\mathbf{y}^3 < 1} \delta_0^2(\mathbf{x}, \mathbf{y}^3 + \mathbf{x})\}, f_1\langle \mathbf{y}_1^3; \mathbf{x}\rangle)$

7. Case 1: return $(\mathcal{I}, !_{\mathbf{y}^3 < 1} \delta_0^2(\mathbf{x}, \mathbf{y}^3 + \mathbf{x}),$
$\{e\langle \mathbf{y}_\mathbf{x}^0; \mathbf{x}\rangle : \alpha^1(\mathbf{x}), x : \gamma, f_0\langle \mathbf{y}_\mathbf{x}^2; \mathbf{x}\rangle : !_{\mathbf{y}^2 < \mathbf{x}} \delta_0^2(\mathbf{x}, \mathbf{y}^2)\})$

(c) $\mathit{unify}(\mathcal{I}, !_{\mathbf{y}^3 < 1} \delta_1^2(\mathbf{x}, \mathbf{y}^3), !_{\mathbf{y}^3 < 1} \delta_0^2(\mathbf{x}, \mathbf{y}^3 + \mathbf{x}), \mathbf{y}^3)$

- $\mathcal{I}(!_{\mathbf{y}^3 < 1} \delta_1^2(\mathbf{x}, \mathbf{y}^3)) \neq \mathcal{I}(!_{\mathbf{y}^3 < 1} \delta_0^2(\mathbf{x}, \mathbf{y}^3 + \mathbf{x}))$, so
- $D(!_{\mathbf{y}^3 < 1} \delta_1^2(\mathbf{x}, \mathbf{y}^3), !_{\mathbf{y}^3 < 1} \delta_0^2(\mathbf{x}, \mathbf{y}^3 + \mathbf{x}))$
  - [case 5] $D(\delta_1^2(\mathbf{x}, \mathbf{y}^3), \delta_0^2(\mathbf{x}, \mathbf{y}^3 + \mathbf{x}))$
  - [case 2(a)] (resource, $\delta_1^2(\mathbf{x}, \mathbf{y}^3), \delta_0^2(\mathbf{x}, \mathbf{y}^3 + \mathbf{x})$)
- the propositional variables in $\delta_0^2(\mathbf{x}, \mathbf{y}^3 + \mathbf{x})$ do not all have undefined arity, $\mathbf{y}^3$ occurs in both $\delta_1^2(\mathbf{x}, \mathbf{y}^3)$ and $\delta_0^2(\mathbf{x}, \mathbf{y}^3 + \mathbf{x})$, and $\#\delta_0^2(\mathbf{x}, \mathbf{y}^3 + \mathbf{x}) = 2$, so
- $\mathit{unify}(\sigma_2 = [\delta_0^2(\triangle_2, \triangle_0 + \triangle_2)/\delta_1^2(\triangle_2, \triangle_0)], !_{\mathbf{y}^3 < 1} \delta_1^2(\mathbf{x}, \mathbf{y}^3),$
  $!_{\mathbf{y}^3 < 1} \delta_0^2(\mathbf{x}, \mathbf{y}^3 + \mathbf{x}), \mathbf{y}^3)$
  - $\sigma_2(!_{\mathbf{y}^3 < 1} \delta_1^2(\mathbf{x}, \mathbf{y}^3)) = !_{\mathbf{y}^3 < 1} \delta_0^2(\mathbf{x}, \mathbf{y}^3 + \mathbf{x}) = \sigma_2(!_{\mathbf{y}^3 < 1} \delta_0^2(\mathbf{x}, \mathbf{y}^3 + \mathbf{x}))$, so return$(\sigma_2)$

(d) $\mathcal{L}(\{e\langle \mathbf{y}_\mathbf{x}^0; \mathbf{x}\rangle : \alpha^1(\mathbf{x}), x : \gamma, f_0\langle \mathbf{y}_\mathbf{x}^2; \mathbf{x}\rangle : !_{\mathbf{y}^2 < \mathbf{x}} \delta_0^2(\mathbf{x}, \mathbf{y}^2), f_2\langle ; \mathbf{x}\rangle : \delta_0^2(\mathbf{x}, \mathbf{x})\},$
$f_2\ (e\ f_0\ x))$

8.... Case 5:

(a) δ_2

(b) $\mathcal{L}(\{e\langle \mathbf{y}_\mathbf{x}^0; \mathbf{x}\rangle : \alpha^1(\mathbf{x}), x : \gamma, f_0\langle \mathbf{y}_\mathbf{x}^2; \mathbf{x}\rangle : !_{\mathbf{y}^2 < \mathbf{x}} \delta_0^2(\mathbf{x}, \mathbf{y}^2), f_2\langle ; \mathbf{x}\rangle : \delta_0^2(\mathbf{x}, \mathbf{x})\},$
$f_2\langle ; \mathbf{x}\rangle)$

9. Case 1: return $(\mathcal{I}, \delta_0^2(\mathbf{x}, \mathbf{x}),$
$\{e\langle \mathbf{y}_\mathbf{x}^0; \mathbf{x}\rangle : \alpha^1(\mathbf{x}), x : \gamma, f_0\langle \mathbf{y}_\mathbf{x}^2; \mathbf{x}\rangle : !_{\mathbf{y}^2 < \mathbf{x}} \delta_0^2(\mathbf{x}, \mathbf{y}^2)\})$

(c) $\mathcal{L}(\{e\langle \mathbf{y}_\mathbf{x}^0; \mathbf{x}\rangle : \alpha^1(\mathbf{x}), x : \gamma, f_0\langle \mathbf{y}_\mathbf{x}^2; \mathbf{x}\rangle : !_{\mathbf{y}^2 < \mathbf{x}} \delta_0^2(\mathbf{x}, \mathbf{y}^2)\}, (e\ f_0)\ x)$

10.... Case 5:

(a) δ_3

(b) $\mathcal{L}(\{e\langle \mathbf{y}^0_{\mathbf{x}}; \mathbf{x}\rangle : \alpha^1(\mathbf{x}), x : \gamma, f_0\langle \mathbf{y}^2_{\mathbf{x}}; \mathbf{x}\rangle : !_{\mathbf{y}^2<\mathbf{x}}\delta^2_0(\mathbf{x}, \mathbf{y}^2)\}, (e\ f_0))$

11.... Case 5:

(a) δ_4

(b) $\mathcal{L}(\{e\langle \mathbf{y}^0_{\mathbf{x}}; \mathbf{x}\rangle : \alpha^1(\mathbf{x}), x : \gamma, f_0\langle \mathbf{y}^2_{\mathbf{x}}; \mathbf{x}\rangle : !_{\mathbf{y}^2<\mathbf{x}}\delta^2_0(\mathbf{x}, \mathbf{y}^2)\}, e\langle \mathbf{y}^0_{\mathbf{x}}; \mathbf{x}\rangle)$

12. Case 1: return $(\mathcal{I}, \alpha^1(\mathbf{x}), \{x : \gamma, f_0\langle \mathbf{y}^2_{\mathbf{x}}; \mathbf{x}\rangle : !_{\mathbf{y}^2<\mathbf{x}}\delta^2_0(\mathbf{x}, \mathbf{y}^2)\})$

(b) $\mathcal{L}(\{x : \gamma, f_0\langle \mathbf{y}^2_{\mathbf{x}}; \mathbf{x}\rangle : !_{\mathbf{y}^2<\mathbf{x}}\delta^2_0(\mathbf{x}, \mathbf{y}^2)\}, f_0\langle \mathbf{y}^2_{\mathbf{x}}; \mathbf{x}\rangle)$

13. Case 1: return $(\mathcal{I}, !_{\mathbf{y}^2<\mathbf{x}}\delta^2_0(\mathbf{x}, \mathbf{y}^2), \{x : \gamma\})$

(d) *unify*$(\mathcal{I}, \alpha^1(\mathbf{x}), !_{\mathbf{y}^2<\mathbf{x}}\delta^2_0(\mathbf{x}, \mathbf{y}^2) \multimap \delta_4, \varepsilon)$

- $\mathcal{I}(\alpha^1(\mathbf{x})) \neq \mathcal{I}(!_{\mathbf{y}^2<\mathbf{x}}\delta^2_0(\mathbf{x}, \mathbf{y}^2) \multimap \delta_4)$, so
- $D(\alpha^1(\mathbf{x}), !_{\mathbf{y}^2<\mathbf{x}}\delta^2_0(\mathbf{x}, \mathbf{y}^2) \multimap \delta_4)$
  - [case 2(a)] (resource, $\alpha^1(\mathbf{x}), !_{\mathbf{y}^2<\mathbf{x}}\delta^2_0(\mathbf{x}, \mathbf{y}^2) \multimap \delta_4$)
- the propositional variables in $!_{\mathbf{y}^2<\mathbf{x}}\delta^2_0(\mathbf{x}, \mathbf{y}^2) \multimap \delta_4$ do not all have undefined arity and $\#!_{\mathbf{y}^2<\mathbf{x}}\delta^2_0(\mathbf{x}, \mathbf{y}^2) \multimap \delta_4 = 1$, so
- *unify*$(\sigma_3 = [!_{\mathbf{y}^2<\Delta_3}\delta^2_0(\Delta_3, \mathbf{y}^2) \multimap \delta_4/\alpha^1(\Delta_3)], \alpha^1(\mathbf{x}),$
  $!_{\mathbf{y}^2<\mathbf{x}}\delta^2_0(\mathbf{x}, \mathbf{y}^2) \multimap \delta_4, \varepsilon)$
  - $\sigma_3(\alpha^1(\mathbf{x})) = !_{\mathbf{y}^2<\mathbf{x}}\delta^2_0(\mathbf{x}, \mathbf{y}^2) \multimap \delta_4 = \sigma_3(!_{\mathbf{y}^2<\mathbf{x}}\delta^2_0(\mathbf{x}, \mathbf{y}^2) \multimap \delta_4)$, so return$(\sigma_3)$

...**11** return $(\sigma_3, \delta_4, \{x : \gamma\})$

(c) $\mathcal{L}(\{x : \gamma\}, x)$

14. Case 1: return $(\mathcal{I}, \gamma, \varnothing)$

(d) *unify*$(\mathcal{I}, \delta_4, \gamma \multimap \delta_3, \varepsilon)$

- $\mathcal{I}(\delta_4) \neq \mathcal{I}(\gamma \multimap \delta_3)$, so
- $D(\delta_4, \gamma \multimap \delta_3)$
  - [case 1(a)] (simple, $\delta_4, \gamma \multimap \delta_3$)

- *unify*$(\sigma_4 = [\gamma \multimap \delta_3/\delta_4], \delta_4, \gamma \multimap \delta_3, \varepsilon)$
  - $\sigma_4(\delta_4) = \gamma \multimap \delta_3 = \sigma_4(\gamma \multimap \delta_3)$, so return$(\sigma_4)$

...**10** return $(\sigma_4\sigma_3, \delta_3, \varnothing)$

(d) *unify*$(\mathcal{I}, \delta_0^2(\mathbf{x},\mathbf{x}), \delta_3 \multimap \delta_2, \varepsilon)$

- $\mathcal{I}(\delta_0^2(\mathbf{x},\mathbf{x})) \neq \mathcal{I}(\delta_3 \multimap \delta_2)$, so
- $D(\delta_0^2(\mathbf{x},\mathbf{x}), \delta_3 \multimap \delta_2)$
  - [case 2(a)] (resource, $\delta_0^2(\mathbf{x},\mathbf{x}), \delta_3 \multimap \delta_2$)
- all the propositional variables in $\delta_3 \multimap \delta_2$ have undefined arity, so
- *unify*$(\sigma_5 = [\delta_2 \mapsto \delta_2^2(\mathbf{x},\mathbf{x}), \delta_3 \mapsto \delta_3^2(\mathbf{x},\mathbf{x})]$
  $[(\delta_3^2(\triangle_4, \triangle_5) \multimap \delta_2^2(\triangle_4, \triangle_5))/\delta_0^2(\triangle_4, \triangle_5)], \delta_0^2(\mathbf{x},\mathbf{x}), \delta_3 \multimap \delta_2, \varepsilon)$
  - $\sigma_5(\delta_0^2(\mathbf{x},\mathbf{x})) = \delta_3^2(\mathbf{x},\mathbf{x}) \multimap \delta_2^2(\mathbf{x},\mathbf{x}) = \sigma_5(\delta_3 \multimap \delta_2)$, so return$(\sigma_5)$

...**8** return $(\sigma_5\sigma_4\sigma_3, \delta_2^2(\mathbf{x},\mathbf{x}), \varnothing)$

...**6** return $(\sigma_5\sigma_4\sigma_3\sigma_2, \delta_2^2(\mathbf{x},\mathbf{x}), \varnothing)$

...**4** return $(\sigma = (\sigma_5\sigma_4\sigma_3\sigma_2\sigma_1), \delta_2^2(\mathbf{x},\mathbf{x}), \varnothing)$
($\sigma = [\delta_2 \mapsto \delta_2^2(\mathbf{x},\mathbf{x}), \delta_3 \mapsto \delta_3^2(\mathbf{x},\mathbf{x})][(\delta_3^2(\triangle_4, \triangle_5) \multimap \delta_2^2(\triangle_4, \triangle_5))/\delta_0^2(\triangle_4, \triangle_5)]$
$[\gamma \multimap \delta_3/\delta_4][!_{\mathbf{y}^2 < \triangle_3}\delta_0^2(\triangle_3, \mathbf{y}^2) \multimap \delta_4/\alpha^1(\triangle_3)][\delta_0^2(\triangle_2, \triangle_0 + \triangle_2)/\delta_1^2(\triangle_2, \triangle_0)]$
$[!_{\mathbf{y}^1 < \triangle_1 + 1}\delta_0^2(\triangle_1, \mathbf{y}^1)/\beta^1(\triangle_1)]$)

...**3** return $(\sigma, \gamma \multimap \delta_2^2(\mathbf{x},\mathbf{x}), \varnothing)$ (because $\sigma(\gamma \multimap \delta_2^2(\mathbf{x},\mathbf{x})) = \gamma \multimap \delta_2^2(\mathbf{x},\mathbf{x})$)

...**2** return $(\sigma, !_{\mathbf{y}^1 < \mathbf{x}+1}(\delta_3^2(\mathbf{x},\mathbf{y}^1) \multimap \delta_2^2(\mathbf{x},\mathbf{y}^1)) \multimap \gamma \multimap \delta_2^2(\mathbf{x},\mathbf{x}), \varnothing)$
(because $\sigma\beta^1(\mathbf{x}) = !_{\mathbf{y}^1 < \mathbf{x}+1}(\delta_3^2(\mathbf{x},\mathbf{y}^1) \multimap \delta_2^2(\mathbf{x},\mathbf{y}^1))$)

...**1** return $(\sigma, (!_{\mathbf{y}^2 < \mathbf{x}}(\delta_3^2(\mathbf{x},\mathbf{y}^2) \multimap \delta_2^2(\mathbf{x},\mathbf{y}^2)) \multimap \gamma \multimap \delta_3^2(\mathbf{x},\mathbf{x})) \multimap$
$!_{\mathbf{y}^1 < \mathbf{x}+1}(\delta_3^2(\mathbf{x},\mathbf{y}^1) \multimap \delta_2^2(\mathbf{x},\mathbf{y}^1)) \multimap \gamma \multimap \delta_2^2(\mathbf{x},\mathbf{x}), \varnothing)$
(because $\sigma\alpha^1(\mathbf{x}) = !_{\mathbf{y}^2 < \mathbf{x}}(\delta_3^2(\mathbf{x},\mathbf{y}^2) \multimap \delta_2^2(\mathbf{x},\mathbf{y}^2)) \multimap \gamma \multimap \delta_3^2(\mathbf{x},\mathbf{x})$)

2. return $(!_{\mathbf{y}^2 < \mathbf{x}}(\delta_3^2(\mathbf{x},\mathbf{y}^2) \multimap \delta_2^2(\mathbf{x},\mathbf{y}^2)) \multimap \gamma \multimap \delta_3^2(\mathbf{x},\mathbf{x})) \multimap$ $!_{\mathbf{y}^1 < \mathbf{x}+1}(\delta_3^2(\mathbf{x},\mathbf{y}^1) \multimap \delta_2^2(\mathbf{x},\mathbf{y}^1)) \multimap \gamma \multimap \delta_2^2(\mathbf{x},\mathbf{x})$. Note that the usual type of $t\langle\mathbf{x}\rangle$ can be recovered (except for the quantifiers) with the following substitution: $[\alpha^1(\triangle)/\delta_3^2(\triangle', \triangle), \alpha^1(\triangle + 1)/\delta_2^2(\triangle', \triangle), \alpha^1(0)/\gamma]$.

5.2 Reduction Properties

Because every typed term $t : A$ of the BLC corresponds to a proof ρ of A in BLL and each reduction step on t corresponds to a reduction step on ρ, the following important properties of reduction in BLL are immediately true for the BLC:

1. Reduction is strongly normalizable and feasible, *i.e.*, any sequence of reductions on a reducible typed term must terminate in at most a polynomial number of steps.

2. Reduction is Church-Rosser, *i.e.*, if $t \rightsquigarrow u$ and $t \rightsquigarrow v$, then there exists a term t' such that $u \rightsquigarrow t'$ and $v \rightsquigarrow t'$.

3. Reduction preserves types, *i.e.*, if $t : A$ and $t \rightsquigarrow t'$, then $t' : A$.

5.3 BLC and Polynomial Time

A function ϕ from dyadic lists to dyadic lists is *represented* in the BLC by a typed term $f\langle \mathbf{x} \rangle : N^2_{\mathbf{x}} \multimap N^2_{P(\mathbf{x})}$ if, for every dyadic list b of length at most n and the corresponding normal typed term $\bar{b} : N^2_n$, the irreducible form of $r\langle P(n) \rangle \; (f\langle n \rangle \; \bar{b}) : !_{\mathbf{z}<1} N^2_{P(n)}$ is the same as the irreducible form of $r\langle m \rangle \; \overline{\phi(b)} : !_{\mathbf{z}<1} N^2_m$, where $\overline{\phi(b)} : N^2_m$ is the normal typed term corresponding to $\phi(b)$ and $r\langle \mathbf{x} \rangle$ was defined in Chapter 4. Note that this definition is automatically satisfied if $f\langle n \rangle \; \bar{b} \rightsquigarrow \overline{\phi(b)}$.

A function ϕ is *representable* if there exist a polynomial $P(\mathbf{x})$ and a typed term $f\langle \mathbf{x} \rangle$ as above which represent ϕ. It is straightforward to extend the definition in the case of functions on several arguments and for tally natural numbers.

The above definitions are identical in nature to the corresponding ones for BLL, which gives us the following theorems.

Theorem 5.4 *Any function representable in the BLC is computable in polynomial time.*

Proof: If a certain typed term represents a function in the BLC, then the BLL proof corresponding to that term represents the same function in BLL, because of the correspondence between the definitions in the two systems. The result then follows immediately from Theorem 5.3 in Girard *et al.* (1992), which states that any function representable in BLL is computable in polynomial time. □

Theorem 5.5 *Any function computable in polynomial time is representable in the BLC.*

Proof: From Theorem 6.1 in Girard *et al.* (1992), any polynomial-time function can be represented by a BLL proof, to which corresponds a typed term of the BLC because of the term assignment for BLL. The result follows immediately since this BLC term will represent the same function as the corresponding BLL proof, because of the correspondence between the definitions in the two systems. □

Thus, the class of functions computable in polynomial time is the same as the class of functions representable in the BLC, and this result gives a new functional representation of $\mathcal{P}$. Unfortunately, as was noted in the section on the type reconstruction algorithm, this result is not as strong as it seems since the terms which represent functions have to be explicitly typed, in effect requiring us to know the complete BLL proof with which the term is associated. In that sense, what we have accomplished here is more to give a functional framework in which computation on BLL proofs can be easily studied than to provide a new representation for those same proofs.

Chapter 6

Conclusion

This work has introduced a computational interpretation for bounded linear logic, in the form of a modified linear calculus. The syntax of the bounded linear calculus was given, as well as the reduction relation on bounded linear terms. A type reconstruction algorithm was described and proven to be sound and complete with respect to the quantifier-free system, and a number of other important theoretical properties of the BLC were exhibited. In a sense, just as bounded linear logic refines linear logic, so the system of the bounded linear calculus introduced here refines previous systems of linear calculus studied by many researchers.

The system of bounded linear logic is interesting mainly because it provides a logical characterization of the class of polynomial time computable functions, but it is very awkward to work with; manipulating the long and cumbersome proof expressions which represent functions is not easily done. This work started as a simple attempt to gain a better insight into the representation of functions in BLL, but soon, the importance of finding a practical "encoding" mechanism for the BLL proofs became obvious. By similarity with the term assignment systems developed for linear logic, the best way to achieve this was to develop a term calculus for BLL. Also, the type reconstruction algorithm was studied because the advantages of having the bounded linear term calculus are lost if the types of the terms have to be explicitly specified at all times. With the results we have shown, one immediate advantage of the BLC is that it provides a functional framework in which computational issues about BLL can be studied more directly. Unfortunately, the type reconstruction algorithm is not complete with respect to the quantifiers so that the terms have to be explicitly typed, there are still some issues to work out concerning the correctness of the unification algorithm with respect to the class of substitutions used, and the explicit resource information needed in the BLC is still awkward.

As it is, the BLC is a useful tool, but there are still a number of interesting and important issues to be dealt with.

1. Is there a way to extend the calculus to provide a true isomorphism between the BLC and BLL (*i.e.*, including the quantifiers)?

2. Is there a more direct way to represent functions in the BLC?

3. For that matter, is there a direct proof that the bounded linear calculus represents the polynomial time computable functions (*i.e.*, without going through BLL)?

4. In the same vein, are there more direct proofs of the other properties of the BLC which are based on BLL (Church-Rosser, polynomial-time reduction, etc.)?

5. Is there a way to modify the BLC or BLL so as to capture other complexity classes (perhaps by changing the way in which resource information is expressed or manipulated)?

6. The reduction procedure along with the term assignment provide a "computational semantics" for the BLC, but what about trying to give a categorical semantics similar to what is done in Benton *et al.* (1992)? Is it even possible? What about other kinds of interpretations?

7. There have been implementations of linear calculi. Would it be interesting to try and implement the BLC? What would such an implementation be like?

8. What about extending the system to non-intuitionistic versions of BLL: how would we deal with the other connectives?

Bibliography

S. Abramski, Computational interpretations of linear logic. *Theoretical Computer Science* **111** (1993), 3–57.

N. Benton, J. Bierman, V. de Paiva, and M. Hyland, Term assignment for intuitionistic linear logic. Technical Report 262, Computer Laboratory, University of Cambridge, 1992.

A. J. Field and P. G. Harrison, *Functional Programming*. International Computer Science. Addison-Wesley, 1988.

J.-Y. Girard, Linear logic. *Theoretical Computer Science* **50** (1987), 1–102.

J.-Y. Girard, Y. Lafont, and P. Taylor, *Proofs and Types*. Cambridge Tracts in Theoretical Computer Science 7. Cambridge University Press, 1988.

J.-Y. Girard, A. Scedrov, and P. J. Scott, Bounded linear logic: A modular approach to polynomial time computability. *Theoretical Computer Science* **97** (1992), 1–66.

I. Mackie, Lilac - a functional programming language based on linear logic. Master's Thesis, Imperial College, London, 1991.

A. Scedrov, A brief guide to linear logic. *EATCS Bulletin* **41** (1990), 154–165.

P. J. Scott. Personal Correspondence, 1993, 1994.