# Completeness of algebraic CPS simulations

Ali Assaf
LIG, Université Joseh Fourier
Grenoble, France
École Polytechnique
Palaiseau, France
Ali.Assaf@imag.fr

Simon Perdrix
CNRS, LIG, Université de Grenoble
Grenoble, France
Simon.Perdrix@imag.fr

The *algebraic lambda calculus* ($\lambda_{alg}$) [14] and the *linear algebraic lambda calculus* ($\lambda_{lin}$) [4] are two extensions of the classical lambda calculus with linear combinations of terms. They arise independently in distinct contexts: the former is a fragment of the differential lambda calculus, the latter is a candidate lambda calculus for quantum computation. They differ in the handling of application arguments and algebraic rules. The two languages can simulate each other [5] using an algebraic extension of the well-known call-by-value and call-by-name CPS translations. These simulations are sound, in that they preserve reductions. In this paper, we prove that the simulations are actually complete, strengthening the connection between the two languages.

## 1 Introduction

**Algebraic lambda calculi** The *algebraic lambda calculus* ($\lambda_{alg}$) [14] and the *linear algebraic lambda calculus* ($\lambda_{lin}$) [4] are two languages that extend the classical lambda calculus with linear combinations of terms such as $\alpha.M+\beta.N$. They have been introduced independently in two different contexts. The former is a fragment of the differential lambda calculus, and has been introduced in the context of linear logic with the purpose of quantifying nondeterminism: each term of a linear combination represents a possible evolution in a non determinisitic setting. The latter has been introduced as a candidate for a language of quantum computation, where a linear combination of terms corresponds to a superposition of states such as $\alpha.|0\rangle+\beta.|1\rangle$.

The two languages differ in their operational semantics. The first follows a *call-by-name* strategy while the second follows the equivalent of a *call-by-value* strategy. For example, $\lambda_{alg}$ reduces the term $(\lambda x.fxx)(\alpha.y+\beta.z)$ as follows.

$$(\lambda x.fxx)(\alpha.y+\beta.z) \quad \to \quad f(\alpha.y+\beta.z)(\alpha.y+\beta.z)$$

However, this does not agree with the nature of quantum computing. It leads to the cloning of the state $\alpha.y+\beta.z$, which contradicts the *no-cloning* theorem [15]. Only copying of base terms such as $y$ is allowed. Therefore, $\lambda_{lin}$ reduces the term as follows.

$$\begin{aligned}(\lambda x.fxx)(\alpha.y+\beta.z) &\to (\lambda x.fxx)(\alpha.y)+(\lambda x.fxx)(\beta.z)\\ &\to \alpha.(\lambda x.fxx)y+\beta.(\lambda x.fxx)z\\ &\to \alpha.fyy+\beta.fzz\end{aligned}$$

Despite these differences, the work in [5] showed that the two languages can simulate each other. This was accomplished by defining a translation from one language to the other. Given a term $M$ of $\lambda_{lin}$,

Preliminary Report. Final version to appear in:
DCM 2011

we can encode it into a term $N$ of $\lambda_{alg}$ such that reductions of $M$ in $\lambda_{lin}$ correspond to reductions of $N$ in $\lambda_{alg}$. The translation is an algebraic extension of the classical *continuation-passing style* (CPS) encoding used for simulating call-by-name and call-by-value [9, 10, 11].

**Contribution** The CPS transformations introduced in [5] have been proven to be sound, *i.e.* if a term $M$ reduces to a value $V$ in the source language, then the translation of $M$ reduces to the translation of $V$ in the target language. In this paper we prove that they are actually complete, *i.e.* that the converse is also true: if the translation of $M$ reduces to the translation of $V$ in the target language, then $M$ reduces to $V$ in the source language. We do so by adapting techniques used by Sabry and Wadler in [11] to define an inverse translation and showing that it also preserves reductions. The completeness of these CPS transformations strengthens the connection between works done in linear logic [6, 7, 8, 13] and works on quantum computation [1, 2, 3, 12].

**Plan of the paper** The rest of the paper is structured as follows. In section 2, the syntax and the reduction rules of both algebraic languages are presented. Section 3 is dedicated to the simulation of $\lambda_{lin}$ by $\lambda_{alg}$, and section 4 to the opposite simulation. In each of the two cases, the translation introduced in [5] is presented, the grammar of the encoded terms in the target language is given, the inverse translation is defined, and finally the completeness of the CPS translation is proven.

## 2 The algebraic lambda calculi

The languages $\lambda_{lin}$ and $\lambda_{alg}$ share the same syntax, defined by the following grammar, where $\alpha$ ranges over a defined ring, the *ring of scalars*.

$$\begin{array}{rcll} M,N,L & ::= & V \mid MN \mid \alpha.M \mid M+N & \text{(terms)} \\ U,V,W & ::= & B \mid 0 \mid \alpha.V \mid V+W & \text{(values)} \\ B & ::= & x \mid \lambda x.M & \text{(basis terms)} \end{array}$$

The rules of the two languages are presented in Figure 1. Notice how $\lambda_{alg}$ substitutes the argument directly in the body of a function, while $\lambda_{lin}$ delays the substitution until the argument is a base value. We use the same notation as in [5] to define the following rewrite systems obtained by combining the rules described in Figure 1 .

$$\begin{array}{rclcrcl} \to_{\beta_n} & ::= & \beta_n \cup \xi & \quad & \to_{\beta_v} & ::= & \beta_v \cup \xi \cup \xi_{\lambda_{lin}} \\ \to_a & ::= & A \cup L \cup \xi & & \to_l & ::= & A_l \cup A_r \cup L \cup \xi \cup \xi_{\lambda_{lin}} \end{array}$$

The rewrite systems for the two languages are then defined as follows.

| Language | Rewrite system |
|---|---|
| $\lambda_{lin}$ | $\to_{l\cup\beta} ::= (\to_l) \cup (\to_{\beta_v})$ |
| $\lambda_{alg}$ | $\to_{a\cup\beta} ::= (\to_a) \cup (\to_{\beta_n})$ |

## 3 Completeness of the call-by-value to call-by-name simulation

The translation in [5] is a direct extension of the CPS encoding used by Plotkin [10] to show that the call-by-name lambda calculus simulates call-by-value. The definition is the following.

Rules specific to $\lambda_{alg}$

Call-by-name ($\beta_n$)

$$(\lambda x.M)N \rightarrow M[x := N]$$

Linearity of application ($A$)

$$\begin{aligned} (M+N)L &\rightarrow ML+NL \\ (\alpha.M)N &\rightarrow \alpha.(MN) \\ (0)M &\rightarrow 0 \end{aligned}$$

Rules specific to $\lambda_{lin}$

Call-by-value ($\beta_v$)

$$(\lambda x.M)B \rightarrow M[x := B]$$

Right context rule ($\xi_{\lambda_{lin}}$)

$$\frac{M \rightarrow M'}{VM \rightarrow VM'}$$

Left linearity of application ($A_l$)

$$\begin{aligned} (M+N)V &\rightarrow MV+NV \\ (\alpha.M)V &\rightarrow \alpha.(MV) \\ (0)V &\rightarrow 0 \end{aligned}$$

Right linearity of application ($A_r$)

$$\begin{aligned} B(M+N) &\rightarrow BM+BN \\ B(\alpha.M) &\rightarrow \alpha.(BM) \\ B(0) &\rightarrow 0 \end{aligned}$$

Common rules

Ring rules ($L = Asso \cup Com \cup F \cup S$)

Associativity ($Asso$)

$$\begin{aligned} M+(N+L) &\rightarrow (M+N)+L \\ (M+N)+L &\rightarrow M+(N+L) \end{aligned}$$

Commutativity ($Com$)

$$M+N \rightarrow N+M$$

Factorization ($F$)

$$\begin{aligned} \alpha.M+\beta.M &\rightarrow (\alpha+\beta).M \\ \alpha.M+M &\rightarrow (\alpha+1).M \\ M+M &\rightarrow (1+1).M \\ \alpha.(\beta.M) &\rightarrow (\alpha\beta).M \end{aligned}$$

Simplification ($S$)

$$\begin{aligned} \alpha.(M+N) &\rightarrow \alpha.M+\alpha.N \\ 1.M &\rightarrow M \\ 0.M &\rightarrow 0 \\ \alpha.0 &\rightarrow 0 \\ 0+M &\rightarrow M \end{aligned}$$

Context rules ($\xi$)

$$\frac{M \rightarrow M'}{(M)\,N \rightarrow (M')\,N} \qquad \frac{M \rightarrow M'}{M+N \rightarrow M'+N}$$

$$\frac{M \rightarrow M'}{\alpha.M \rightarrow \alpha.M'} \qquad \frac{N \rightarrow N'}{M+N \rightarrow M+N'}$$

Figure 1: Rewrite rules for $\lambda_{lin}$ and $\lambda_{alg}$

$$\begin{array}{rcl}
[\![x]\!] & = & \lambda k.kx \\
[\![\lambda x.M]\!] & = & \lambda k.k(\lambda x.[\![M]\!]) \\
[\![MN]\!] & = & \lambda k.[\![M]\!](\lambda b_1.[\![N]\!](\lambda b_2.b_1 b_2 k)) \\
[\![0]\!] & = & 0 \\
[\![\alpha.M]\!] & = & \lambda k.(\alpha.[\![M]\!])k \\
[\![M+N]\!] & = & \lambda k.([\![M]\!]+[\![N]\!])k
\end{array}$$

Notice that the translation preserves the set of free variables. New variables names like $b$ or $k$ are chosen to be fresh so as to not collide with free variables in the term. In fact we reserve the name $k$ to abstract over continuations. In general we think of $k$ to mean the end of the computation, or the equivalent of a *return* statement.

This translation simulates the reductions of a term $M$ by the reductions of the term $[\![M]\!]k$, where $k$ is free. The soundness of the simulation uses an intermediate translation denoted by $M : K$ where $M$ is a $\lambda_{lin}$ term and $K$ is a continuation (a $\lambda_{alg}$ term). This *colon* translation was originally used by Plotkin [10] to describe intermediate reductions of translated terms, where initial *administrative redexes* had been eliminated.

$$\begin{array}{rcl}
\Psi(x) & = & x \\
\Psi(\lambda x.M) & = & \lambda x.[\![M]\!] \\
B : K & = & K\Psi(B) \\
0 : K & = & 0 \\
\alpha.M : K & = & \alpha.(M : K) \\
M+N : K & = & M : K + N : K
\end{array}
\qquad
\begin{array}{rcl}
BN : K & = & N : \lambda b.\Psi(B)bK \\
(MN)L : K & = & MN : \lambda b_1.[\![L]\!](\lambda b_2.b_1 b_2 K) \\
(0)N : K & = & 0 : K \\
(\alpha.M)N : K & = & \alpha.(MN) : K \\
(M+N)L : K & = & ML + NL : K
\end{array}$$

Soundness was proved by showing that this translation preserves reductions: for any term $M$, if $M$ reduces to $M'$, then $M : K$ reduces to $M' : K$ for all $K$. Combined with the fact that $[\![M]\!]k$ reduces initially to $M : k$, this gave the soundness of the simulation.

**Proposition 1** (Simulation [5])**.** *For any term $M$, if $M \to^*_{l\cup\beta} V$ then $[\![M]\!]k \to^*_{a\cup\beta} V : k$.*

We will show that the converse is also true:

**Theorem 2** (Completeness)**.** *If $[\![M]\!]k \to^*_{a\cup\beta} V : k$ then $M \to^*_{l\cup\beta} V$.*

To prove it, we define an inverse translation and show that it preserves reductions. First, we need to characterize the encoded terms. We define a subset of $\lambda_{alg}$ which contains the image of the translation and is closed by $\to_{a\cup\beta}$ reductions with the following grammar.

$$\begin{array}{rcll}
C & ::= & KB \mid B_1 B_2 K \mid TK & \text{(base computations)} \\
D & ::= & C \mid 0 \mid \alpha.D \mid D_1 + D_2 & \text{(computation combinations)} \\
S & ::= & \lambda k.C & \text{(base suspensions)} \\
T & ::= & S \mid 0 \mid \alpha.T \mid T_1 + T_2 & \text{(suspension combinations)} \\
K & ::= & k \mid \lambda b.BbK \mid \lambda b_1.T(\lambda b_2.b_1 b_2 K) & \text{(continuations)} \\
B & ::= & x \mid \lambda x.S & \text{(CPS-values)}
\end{array}$$

There are four main categories of terms: *computations*, *suspensions*, *continuations*, and *CPS-values*. We distinguish base computations $C$ from linear combinations of computations $D$, as well as base suspensions $S$ from linear combinations of suspensions $T$. The translation $[\![M]\!]$ gives a term of the class $T$, while $[\![M]\!]k$ and $M : K$ are of class $D$.

There are some restrictions on the names of the variables in this grammar. The variable name $k$ that appears in the class $K$ must be the same as the one used in suspensions of the form $\lambda k.C$. It cannot appear as a variable name in any other term. This is to agree with the requirement of freshness that we mentioned above. The same applies for the variables $b$, $b_1$ and $b_2$ of $K$ terms: they cannot appear (free) in any subterm. In particular, these restrictions ensure that the grammar is unambiguous. The three kinds of variables ($x$, $k$ and $b$) play different roles, which is why we distinguish them using different names.

Computations are the only terms that contain applications, so they are the only terms that can $\beta$-reduce, hence the name. In fact, notice that the arguments are always base values. This shows a simple alternative proof for the *indifference* property [5] of the CPS translation, namely that the reductions of a translated term are exactly the same in $\lambda_{lin}$ and $\lambda_{alg}$.

**Proposition 3** (Indifference [5])**.** *For any computations $D$ and $D'$, $D \to_{a\cup\beta} D'$ if and only if $D \to_{l\cup\beta} D'$. In particular, if $M \to^*_{l\cup\beta} V$ then $[\![M]\!]k \to^*_{l\cup\beta} V : k$.*

We define the inverse translation using the following four functions, corresponding to each of the four main categories in the grammar.

$$\begin{array}{rcl} \overline{KB} & = & \underline{K}[\psi(B)] \\ \overline{B_1B_2K} & = & \underline{K}[\psi(B_1)\psi(B_2)] \\ \overline{TK} & = & \underline{K}[\sigma(T)] \\ \overline{0} & = & 0 \\ \overline{\alpha.D} & = & \alpha.\overline{D} \\ \overline{D_1+D_2} & = & \overline{D_1}+\overline{D_2} \end{array} \qquad \begin{array}{rcl} \sigma(\lambda k.C) & = & \overline{C} \\ \sigma(0) & = & 0 \\ \sigma(\alpha.T) & = & \alpha.\sigma(T) \\ \sigma(T_1+T_2) & = & \sigma(T_1)+\sigma(T_2) \end{array}$$

$$\begin{array}{rcl} \psi(x) & = & x \\ \psi(\lambda x.S) & = & \lambda x.\sigma(S) \end{array} \qquad \begin{array}{rcl} \underline{k}[M] & = & M \\ \underline{\lambda b.BbK}[M] & = & \underline{K}[\psi(B)M] \\ \underline{\lambda b_1.T(\lambda b_2.b_1b_2K)}[M] & = & \underline{K}[M\sigma(T)] \end{array}$$

These functions are well-defined because the grammar is unambiguous. To prove the completeness of the simulation we need a couple of lemmas. The first states that the translation defined above is in fact an inverse.

**Lemma 4.** *For all $M$, $\overline{[\![M]\!]k} = M$.*

*Proof.* We have $\overline{[\![M]\!]k} = \underline{k}[\sigma([\![M]\!])] = \sigma([\![M]\!])$ so we have to show that $\sigma([\![M]\!]) = M$ for all $M$. The proof follows by induction on the structure of $M$. □

In general, $\overline{M : k} \neq M$. Although it would be true for a classical translation, it does not hold in the algebraic case. Specifically, we have $(\alpha.M)L : k = \alpha.(ML) : k$ and $(M+N)L : k = ML + NL : K$, so the translation is not injective. However it is still true for values.

**Lemma 5.** *For all $V$, $\overline{V : k} = V$.*

*Proof.* By induction on the structure of $V$. □

**Lemma 6** (Substitution)**.** *The following are true.*

1. $\psi(B_1)[x := \psi(B)] = \psi(B_1[x := B])$
2. $\sigma(T)[x := \psi(B)] = \sigma(T[x := B])$
3. $\overline{C}[x := \psi(B)] = \overline{C[x := B]}$
4. $\underline{K}[M][x := \psi(B)] = \underline{K[x := B]}[M[x := \psi(B)]]$

*Proof.* By induction on the structure of $B_1$, $T$, $C$, and $K$. □

The next lemma states that we can compose two continuations $K_1$ and $K_2$ by replacing $k$ by $K_1$ in $K_2$.

**Lemma 7.** *For all terms $M$ and continuations $K_1$ and $K_2$, $\underline{K_1}[\underline{K_2}[M]] = \underline{K_2[k := K_1]}[M]$.*

*Proof.* By induction on the structure of $K_2$. □

**Lemma 8.** For all $K$ and $C$, $\underline{K}[\overline{C}] = \overline{C[k := K]}$.

*Proof.* By induction on the structure of $C$, using Lemma 7 where necessary. □

The following lemma is essential to the preservation of reductions. It shows that reductions of a term $M$ can always be carried in $\underline{K}[M]$.

**Lemma 9.** *For any continuation $K$ and term $M$, if $M \to_{l\cup\beta} M'$, then $\underline{K}[M] \to_{l\cup\beta} \underline{K}[M']$.*

*Proof.* By induction on the structure of $K$. □

**Lemma 10.** *The following are true.*

- $\underline{K}[M_1 + M_2] \to_l^* \underline{K}[M_1] + \underline{K}[M_2]$
- $\underline{K}[\alpha.M] \to_l^* \alpha.\underline{K}[M]$
- $\underline{K}[0] \to_l^* 0$

*Proof.* We prove each statement by induction on $K$, using Lemma 9 where necessary. □

**Lemma 11.** *If $T \to_a T'$ then $\sigma(T) \to_l \sigma(T')$.*

*Proof.* By induction on the reduction rule. Since $T$ terms do not contain applications, the only cases possible are $L \cup \xi$, which are common to both languages. □

We now show the core of the completeness theorem, namely that the inverse translation preserves reductions.

**Lemma 12.** *If $D \to_{a\cup\beta} D'$ then $\overline{D} \to_{l\cup\beta}^* \overline{D'}$.*

*Proof.* By induction on the reduction rule, using Lemmas 6, 8, 9, 10 and 11 where necessary. The cases are the following.

- Case $\beta_n$. There are several subcases.
  - Case $(\lambda b.B_1 bK)B_2 \to_{\beta_n} B_1 B_2 K$. Then

$$\begin{aligned}\overline{(\lambda b.B_1 bK)B_2} &= \underline{\lambda b.B_1 bK}[\psi(B_2)] \\ &= \underline{K}[\psi(B_1)\psi(B_2)] \\ &= \overline{B_1 B_2 K}\end{aligned}$$

  - Case $(\lambda b_1.S(\lambda b_2.b_1 b_2 K))B_1 \to_{\beta_n} S(\lambda b_2.B_1 b_2 K)$. Then

$$\begin{aligned}\overline{(\lambda b_1.S(\lambda b_2.b_1 b_2 K))B_1} &= \underline{\lambda b_1.S(\lambda b_2.b_1 b_2 K)}[\psi(B_1)] \\ &= \underline{K}[\psi(B_1)\sigma(S)]] \\ &= \underline{\lambda b_2.B_1 b_2 K}[\sigma(S)] \\ &= \overline{S(\lambda b_2.B_1 b_2 K)}\end{aligned}$$

– Case $(\lambda k.C)K \to_{\beta_n} C[k := K]$. Then

$$\begin{array}{rcl} \overline{(\lambda k.C)K} & = & \underline{K}[\overline{C}] \\ & = & \overline{C[k := K]} \\ & & \text{by Lemma 8} \end{array}$$

– Case $(\lambda x.S)BK \to_{\beta_n} S[x := B]K$. Then $\psi(B)$ is a base term, so $(\lambda x.\sigma(S))\psi(B) \to_{l\cup\beta} \sigma(S)[x := \psi(B)]$, therefore

$$\begin{array}{rcl} \overline{(\lambda x.S)BK} & = & \underline{K}[(\lambda x.\sigma(S))\psi(B)] \\ & \to_{l\cup\beta} & \underline{K}[\sigma(S)[x := \psi(B)]] \\ & & \text{by Lemma 9} \\ & = & \underline{K}[\sigma(S[x := B])] \\ & & \text{by Lemma 6} \\ & = & \overline{S[x := B]K} \end{array}$$

- Case $A$. Since $B$ and $K$ are base terms, the only term that can match the rules is $TK$. There are three subcases.
  – Case $(T_1+T_2)K \to_a T_1K+T_2K$. Then $\overline{(T_1+T_2)K} = \underline{K}[\sigma(T_1)+\sigma(T_2)] \to_l^* \underline{K}[\sigma(T_1)]+\underline{K}[\sigma(T_2)]$ by Lemma 10, and $\underline{K}[\sigma(T_1)]+\underline{K}[\sigma(T_2)] = \overline{T_1K+T_2K}$.
  – Case $(\alpha.T)K \to_a \alpha.(TK)$. Then $\overline{(\alpha.T)K} = \underline{K}[\alpha.\sigma(T)] \to_l^* \alpha.\underline{K}[M]$ by Lemma 10, and $\alpha.\underline{K}[\sigma(T)] = \overline{\alpha.(TK)}$.
  – Case $0K \to_a 0$. Then $\overline{0K} = \underline{K}[0] \to_l^* 0$ by Lemma 10, and $0 = \overline{0}$.
- Case $L$. Since the rules in $L$ are common to both languages and the inverse translation $\overline{D}$ distributes linearly over the computations, the proof for these cases is straightforward. We give the following example. Consider $D_1+(D_2+D_3) \to_a (D_1+D_2)+D_3$. Then

$$\begin{array}{rcl} \overline{D_1+(D_2+D_3)} & = & \overline{D_1}+(\overline{D_2}+\overline{D_3}) \\ & \to_l & (\overline{D_1}+\overline{D_2})+\overline{D_3} \\ & = & \overline{(D_1+D_2)+D_3} \end{array}$$

- Case $\xi$. There are 4 subcases.
  – Case $TK \to_a T'K$ and $T \to_a T'$. Then $\sigma(T) \to_l \sigma(T')$ by Lemma 11, therefore

$$\begin{array}{rcl} \overline{TK} & = & \underline{K}[\sigma(T)] \\ & \to_l & \underline{K}[\sigma(T')] \\ & & \text{by Lemma 9} \\ & = & \overline{T'K} \end{array}$$

  – The other three cases are similar to each other. We give the following example. Consider $D_1+D_2 \to_a D_1'+D_2$ and $D_1 \to_a D_1'$. Then by induction hypothesis $\overline{D_1} \to_l \overline{D_1'}$, therefore

$$\begin{array}{rcl} \overline{D_1+D_2} & = & \overline{D_1}+\overline{D_2} \\ & \to_l & \overline{D_1}'+\overline{D_2} \\ & = & \overline{D_1'+D_2} \end{array}$$

□

We can now prove the completeness theorem.

*Proof of Theorem 2.* By using Lemma 12 for each step of the reduction, we get $\overline{[\![M]\!]k} \to^*_{l\cup\beta} \overline{V : k}$. By Lemma 4 and Lemma 5, this implies $M \to^*_{l\cup\beta} V$. □

## 4 Completeness of the call-by-name to call-by-value simulation

This simulation is similar to the other one, and uses the same techniques. The adjustments we have to make are the same as in the classical case, and deal mainly with our treatment of variables and applications. The CPS translation, as defined in [5], is the following.

$$\begin{array}{rcl}
\{|x|\} & = & x \\
\{|\lambda x.M|\} & = & \lambda k.k(\lambda x.\{|M|\}) \\
\{|MN|\} & = & \lambda k.\{|M|\}(\lambda b.b\{|N|\}k) \\
\{|0|\} & = & 0 \\
\{|\alpha.M|\} & = & \lambda k.(\alpha.\{|M|\})k \\
\{|M+N|\} & = & \lambda k.(\{|M|\}+\{|N|\})k
\end{array}$$

The soundness of the simulation uses a similar colon translation.

$$\begin{array}{rcl@{\qquad}rcl}
\Phi(\lambda x.M) & = & \lambda x.\{|M|\} & (\lambda x.M)N : K & = & \Phi(\lambda x.M)\{|N|\}K \\
\lambda x.M : K & = & K\Phi(\lambda x.M) & xN : K & = & x : (\lambda b.b\{|N|\}K) \\
x : K & = & xK & (MN)L : K & = & MN : \lambda b.b\{|L|\}K \\
0 : K & = & 0 & (0)N : K & = & 0 : K \\
\alpha.M : K & = & \alpha.(M : K) & (\alpha.M)N : K & = & \alpha.(MN) : K \\
M+N : K & = & M : K + N : K & (M+N)L : K & = & ML + NL : K
\end{array}$$

**Proposition 13** (Simulation [5])**.** *For any term $M$, if $M \to^*_{a\cup\beta} V$ then $[\![M]\!]k \to^*_{l\cup\beta} V : k$.*

We will use the same procedure as in the previous section to show that the translation is also complete.

**Theorem 14** (Completeness)**.** *If $\{|M|\}k \to^*_{l\cup\beta} V : k$ then $M \to^*_{a\cup\beta} V$.*

Here is the grammar of the target language. It is closed under $\to_{l\cup\beta}$ reductions.

$$\begin{array}{rcll}
C & ::= & KB \mid BSK \mid TK & \text{(base computations)} \\
D & ::= & C \mid 0 \mid \alpha.D \mid D_1 + D_2 & \text{(computation combinations)} \\
S & ::= & x \mid \lambda k.C & \text{(base suspensions)} \\
T & ::= & S \mid 0 \mid \alpha.T \mid T_1 + T_2 & \text{(suspension combinations)} \\
K & ::= & k \mid \lambda b.bSK & \text{(continuations)} \\
B & ::= & \lambda x.S & \text{(CPS-values)}
\end{array}$$

Notice how $x$ is now considered a suspension, not a CPS-value. This is because $x$ is replaced by a suspension after beta-reducing a term of the form $(\lambda x.S)SK$. This is the main difference between the call-by-name and call-by-value CPS simulations. Other than that, it satisfies the same properties. In particular, we have the same indifference property.

**Proposition 15** (Indifference [5])**.** *For any computations $D$ and $D'$, $D \to_{a\cup\beta} D'$ if and only if $C \to_{l\cup\beta} C'$. In particular, if $M \to^*_{a\cup\beta} V$ then $\{|M|\}k \to^*_{a\cup\beta} V : k$.*

We define the inverse translation using the following four functions.

$$\begin{array}{rclcrcl}
\overline{KB} & = & \underline{K}[\phi(B)] & & \sigma(x) & = & x \\
\overline{BSK} & = & \underline{K}[\phi(B)\sigma(S)] & & \sigma(\lambda k.C) & = & \overline{C} \\
\overline{TK} & = & \underline{K}[\sigma(T)] & & \sigma(0) & = & 0 \\
\overline{0} & = & 0 & & \sigma(\alpha.T) & = & \alpha.\sigma(T) \\
\overline{\alpha.D} & = & \alpha.\overline{D} & & \sigma(T_1+T_2) & = & \sigma(T_1)+\sigma(T_2) \\
\overline{D_1+D_2} & = & \overline{D_1}+\overline{D_2} & & & & \\
 & & & & \underline{k}[M] & = & M \\
\phi(\lambda x.S) & = & \lambda x.\sigma(S) & & \underline{\lambda b.bSK}[M] & = & \underline{K}[M\sigma(S)]
\end{array}$$

To prove the completeness of the simulation we need analogous lemmas. Their proofs are similar, but we need to account for the changes mentioned above.

**Lemma 16.** $\overline{\{|M|\}k} = M$

*Proof.* We have $\overline{\{|M|\}k} = \underline{k}[\sigma(\{|M|\})] = \sigma(\{|M|\})$ so we have to show that $\sigma(\{|M|\}) = M$ for all $M$. The proof follows by induction on the structure of $M$. □

**Lemma 17.** $\overline{V : k} = V$

*Proof.* By induction on the structure of $V$. □

**Lemma 18** (Substitution)**.** *The following are true.*

1. $\phi(B)[x := \sigma(S)] = \phi(B[x := S])$
2. $\sigma(T)[x := \sigma(S)] = \sigma(T[x := S])$
3. $\overline{C}[x := \sigma(S)] = \overline{C[x := S]}$
4. $\underline{K}[M][x := \sigma(S)] = \underline{K[x := S]}[M[x := \sigma(S)]]$

*Proof.* By induction on the structure of $B$, $T$, $C$ and $K$. □

**Lemma 19.** $\underline{K_1}[\underline{K_2}[M]] = \underline{K_2[k := K_1]}[M]$.

*Proof.* By induction on the structure of $K_2$. □

**Lemma 20.** $\underline{K}[\overline{C}] = \overline{C[k := K]}$.

*Proof.* By induction on the structure of $C$, using Lemma 19 where necessary. □

**Lemma 21.** *For any continuation $K$ and term $M$, if $M \to_{a\cup\beta} M'$ then $\underline{K}[M] \to_{a\cup\beta} \underline{K}[M']$.*

*Proof.* By induction on the structure of $K$. □

**Lemma 22.** *The following are true.*

- $\underline{K}[M_1 + M_2] \to^*_a \underline{K}[M_1] + \underline{K}[M_2]$
- $\underline{K}[\alpha.M] \to^*_a \alpha.\underline{K}[M]$
- $\underline{K}[0] \to^*_a 0$

*Proof.* We prove each statement by induction on $K$, using Lemma 21 where necessary. □

**Lemma 23.** *If $T \to_l T'$ then $\sigma(T) \to_a \sigma(T')$.*

*Proof.* By induction on the reduction rule. Since $T$ terms do not contain applications, the only cases possible are $L \cup \xi$, which are common to both languages. □

**Lemma 24.** *If $D \to_{l\cup\beta} D'$ then $\overline{D} \to^*_{a\cup\beta} \overline{D'}$.*

*Proof.* By induction on the reduction rule, using Lemmas 18, 20, 21, 22 and 23 where necessary. Notice that the rules $\xi_{\lambda_{lin}}$ and $A_r$ are not applicable since arguments in the target language are always base terms.

- Case $\beta_v$. There are several subcases.
  - Case $(\lambda b.bSK)B \to_{\beta_v} BSK$. Then
$$\begin{aligned}\overline{(\lambda b.bSK)B} &= \underline{\lambda b.bSK}[\phi(B)]\\ &= \underline{K}[\phi(B)\sigma(S)]\\ &= \overline{BSK}\end{aligned}$$
  - Case $(\lambda k.C)K \to_{\beta_v} C[k := K]$. Then
$$\begin{aligned}\overline{(\lambda k.C)K} &= \underline{K}[\overline{C}]\\ &= \overline{C[k := K]}\\ &\quad \text{by Lemma 20}\end{aligned}$$
  - Case $(\lambda x.S)S_2K \to_{\beta_v} S[x := S_2]K$. Then $(\lambda x.\sigma(S))\sigma(S_2) \to_a \sigma(S)[x := \sigma(S_2)]$, and
$$\begin{aligned}\overline{(\lambda x.S)S_2K} &= \underline{K}[(\lambda x.\sigma(S))\sigma(S_2)]\\ &\to_l \underline{K}[\sigma(S)[x := \sigma(S_2)]]\\ &\quad \text{by Lemma 21}\\ &= \underline{K}[\sigma(S[x := S_2])]\\ &\quad \text{by Lemma 18}\\ &= \overline{S[x := S_2]K}\end{aligned}$$
- Case $A_l$. Since $B$ and $K$ are base terms, the only term that can match the rules is $TK$. There are three subcases.
  - Case $(T_1 + T_2)K \to_l T_1K + T_2K$. Then $\overline{(T_1 + T_2)K} = \underline{K}[\sigma(T_1) + \sigma(T_2)] \to^*_a \underline{K}[\sigma(T_1)] + \underline{K}[\sigma(T_2)]$ by Lemma 22, and $\underline{K}[\sigma(T_1)] + \underline{K}[\sigma(T_2)] = \overline{T_1K + T_2K}$.
  - Case $(\alpha.T)K \to_l \alpha.(TK)$. Then $\overline{(\alpha.T)K} = \underline{K}[\alpha.\sigma(T)] \to^*_a \alpha.\underline{K}[M]$ by Lemma 22, and $\alpha.\underline{K}[\sigma(T)] = \overline{\alpha.(TK)}$.
  - Case $0K \to_l 0$. Then $\overline{0K} = \underline{K}[0] \to^*_a 0$ by Lemma 22, and $0 = \overline{0}$.
- Case $L$. Since the rules in $L$ are common to both languages and the inverse translation $\overline{D}$ distributes linearly over the computations, the proof for these cases is straightforward. We give the following example. Consider $D_1 + (D_2 + D_3) \to_l (D_1 + D_2) + D_3$. Then
$$\begin{aligned}\overline{D_1 + (D_2 + D_3)} &= \overline{D_1} + (\overline{D_2} + \overline{D_3})\\ &\to_a (\overline{D_1} + \overline{D_2}) + \overline{D_3}\\ &= \overline{(D_1 + D_2) + D_3}\end{aligned}$$

- Case $\xi$. There are 4 subcases.
  - Case $TK \to_l T'K$ and $T \to_l T'$. Then $\sigma(T) \to_a \sigma(T')$ by Lemma 23, therefore

$$\begin{array}{rcl} \overline{TK} & = & \underline{K}[\sigma(T)] \\ & \to_a & \underline{K}[\sigma(T')] \\ & & \text{by Lemma 21} \\ & = & \overline{T'K} \end{array}$$

  - The other three cases are similar to each other. We give the following example. Consider $D_1 + D_2 \to_l D_1' + D_2$ and $D_1 \to_l D_1'$. Then by induction hypothesis $\overline{D_1} \to_a \overline{D_1'}$, therefore

$$\begin{array}{rcl} \overline{D_1 + D_2} & = & \overline{D_1} + \overline{D_2} \\ & \to_a & \overline{D_1}' + \overline{D_2} \\ & = & \overline{D_1' + D_2} \end{array}$$

□

We can now prove the completeness theorem.

*Proof of Theorem 14.* By using Lemma 24 for each step of the reduction, we get $\overline{\{|M|\}k} \to^*_{a\cup\beta} \overline{V : k}$. By Lemma 16 and Lemma 17, this implies $M \to^*_{a\cup\beta} V$. □

## 5 Discussion and conclusion

We showed the completeness of two CPS translations simulating algebraic lambda calculi introduced in [5]. We did so by adapting techniques used in [11] to define an inverse translation and showing that it preserves reductions.

Our treatment differs from Sabry and Wadler's, which only considers the call-by-value to call-by-name simulation. They decompile continuations into abstractions. For example, they defined $\underline{\lambda b.Bbk}[M]$ as $\mathsf{let}\ b = M\ \mathsf{in}\ \phi(\mathrm{B})b$. This required the modification of the source language and led to the consideration of the computational lambda calculus as a source language. We avoid this by directly substituting and eliminating variables introduced by the forward translation, which allows us to obtain an exact inverse.

Originally, the work in [5] also considers another version of $\lambda_{lin}$ and $\lambda_{alg}$ with *algebraic equalities* instead of *algebraic reductions*. For example, we could go back and forth between $M + N - N$ and $M$, which is not permitted by the rules we presented above. Algebraic equalities can be formulated as the symmetric closure of the algebraic reductions $\to_a$ and $\to_l$. The resulting four systems $\lambda^{\rightarrow}_{lin}$, $\lambda^{\rightarrow}_{alg}$, $\lambda^{=}_{lin}$, and $\lambda^{=}_{alg}$ have all been shown to simulate each other. The results of this paper can be extended to these systems as well.

**Acknowledgements** Many thanks to Alejandro Díaz-Caro, Benoît Valiron, Pablo Arrighi, and Christophe Calvès for fruitful discussions and suggestions. This work is supported by the CNRS - INS2I PEPS project QuAND.

## References


[1] Thorsten Altenkirch & Jonathan J. Grattage (2005): *A functional quantum programming language*. In: *Proceedings of LICS-2005*, IEEE Computer Society, pp. 249–258.

[2] Pablo Arrighi & Alejandro Díaz-Caro (2011): *Scalar System F for Linear-Algebraic $\lambda$-Calculus: Towards a Quantum Physical Logic*. In Bob Coecke, Prakash Panangaden & Peter Selinger, editors: *Proceedings of QPL-2009, Electronic Notes in Theoretical Computer Science* 270/2, Elsevier, pp. 219–229.

[3] Pablo Arrighi & Gilles Dowek (2004): *A Computational Definition of the Notion of Vectorial Space*. In Narciso Martí-Oliet, editor: *Proceedings of WRLA-2004, Electronic Notes in Theoretical Computer Science* 117, Elsevier, pp. 249–261.

[4] Pablo Arrighi & Gilles Dowek (2008): *Linear-algebraic lambda-calculus: higher-order, encodings, and confluence*. In Andrei Voronkov, editor: *Proceedings of RTA-2008, Lecture Notes in Computer Science* 5117, Springer, pp. 17–31.

[5] Alejandro Díaz-Caro, Simon Perdrix, Christine Tasson & Benoît Valiron (2011): *Call by value, call by name and the vectorial behaviour of algebraic $\lambda$-calculus*. (Submitted) `http://membres-liglab.imag.fr/diazcaro/simulations.pdf`.

[6] Thomas Ehrhard (2005): *Finiteness spaces*. *Mathematical Structures in Computer Science* 15(4), pp. 615–646.

[7] Thomas Ehrhard (2010): *A Finiteness Structure on Resource Terms*. In: *Proceedings of LICS-2010*, IEEE Computer Society, pp. 402–410.

[8] Thomas Ehrhard & Laurent Regnier (2003): *The differential lambda-calculus*. *Theoretical Computer Science* 309(1), pp. 1–41.

[9] John Hatcliff & Olivier Danvy (1994): *A Generic Account of Continuation-Passing Styles*. In: *Proceedings of the Twenty-first Annual ACM Symposium on Principles of Programming Languages*, ACM Press, pp. 458–471.

[10] Gordon D. Plotkin (1975): *Call-by-name, call-by-value and the $\lambda$-calculus*. *Theoretical Computer Science* 1(2), pp. 125–159.

[11] Amr Sabry & Philip Wadler (1996): *A Reflection on Call-by-Value*. *ACM Transactions on Programming Languages and Systems* 19, pp. 13–24.

[12] Benoît Valiron (2010): *Semantics of a typed algebraic lambda-calculus*. In S. Barry Cooper, Prakash Panangaden & Elham Kashefi, editors: *Proceedings DCM-2010, Electronic Proceedings in Theoretical Computer Science* 26, Open Publishing Association, pp. 147–158.

[13] Lionel Vaux (2007): *On Linear Combinations of Lambda-Terms*. In Franz Baader, editor: *Proceedings of RTA-2007, Lecture Notes in Computer Science* 4533, Springer, pp. 374–388.

[14] Lionel Vaux (2009): *The algebraic lambda calculus*. *Mathematical Structures in Computer Science* 19(5), pp. 1029–1059.

[15] William K. Wootters & Wojciech .H. Zurek (1982): *A Single Quantum Cannot be Cloned*. *Nature* 299, pp. 802–803.

# A Proof details

## A.1 Lemma 4

*Proof.* By induction on $M$.

- Case $x$. Then $\sigma([\![x]\!]) = \sigma(\lambda k.kx) = \overline{kx} = \underline{k}[\psi(x)] = \psi(x) = x$.
- Case $\lambda x.M$. Then

$$\begin{array}{rcl}\sigma([\![\lambda x.M]\!]) & = & \sigma(\lambda k.k(\lambda x.[\![M]\!])) \\ & = & \overline{k(\lambda x.[\![M]\!])} \\ & = & \underline{k}[\psi(\lambda x.[\![M]\!])] \\ & = & \psi(\lambda x.[\![M]\!]) \\ & = & \lambda x.\sigma([\![M]\!]) \\ & = & \lambda x.M \\ & & \text{by induction hypothesis.}\end{array}$$

- Case $MN$. Then

$$\begin{array}{rcl}\sigma([\![MN]\!]) & = & \sigma(\lambda k.[\![M]\!](\lambda b_1.[\![N]\!](\lambda b_2.b_1b_2k))) \\ & = & \overline{[\![M]\!](\lambda b_1.[\![N]\!](\lambda b_2.b_1b_2k))} \\ & = & \underline{\lambda b_1.[\![N]\!](\lambda b_2.b_1b_2k)}[\sigma([\![M]\!])] \\ & = & \underline{k}[\sigma([\![M]\!])\sigma([\![N]\!])] \\ & = & \sigma([\![M]\!])\sigma([\![N]\!]) \\ & = & MN \\ & & \text{by induction hypothesis}\end{array}$$

- Case $0$. Then $\sigma([\![0]\!]) = \sigma(0) = 0$
- Case $\alpha.M$. Then

$$\begin{array}{rcl}\sigma([\![\alpha.M]\!]) & = & \sigma(\lambda k.(\alpha.[\![M]\!])k) \\ & = & \overline{(\alpha.[\![M]\!])k} \\ & = & \underline{k}[\sigma(\alpha.[\![M]\!])] \\ & = & \sigma(\alpha.[\![M]\!]) \\ & = & \alpha.\sigma([\![M]\!]) \\ & = & \alpha.M \\ & & \text{by induction hypothesis}\end{array}$$

- Case $M+N$. Then

$$\begin{array}{rcl}\sigma([\![M+N]\!]) & = & \sigma(\lambda k.([\![M]\!]+[\![N]\!])k) \\ & = & \overline{([\![M]\!]+[\![N]\!])k} \\ & = & \underline{k}[\sigma([\![M]\!]+[\![N]\!])] \\ & = & \sigma([\![M]\!]+[\![N]\!]) \\ & = & \sigma([\![M]\!])+\sigma([\![N]\!]) \\ & = & M+N \\ & & \text{by induction hypothesis}\end{array}$$

□

## A.2 Lemma 5

*Proof.* By induction on $V$.

- Case 0. Then $\overline{0:k} = \overline{0} = 0$.
- Case $B$. Then $\overline{B:k} = \overline{k\Psi(B)} = \sigma(\lambda k.k\Psi(B)) = \sigma([\![B]\!]) = B$ by Lemma 4.
- Case $\alpha.V$. Then $\overline{\alpha.V:k} = \overline{\alpha.(V:k)} = \alpha.\overline{V:k} = \alpha V$ by induction hypothesis.
- Case $U+V$. Then $\overline{U+V:k} = \overline{U:k+V:k} = \overline{U:k}+\overline{V:k} = U+V$ by induction hypothesis.

□

## A.3 Lemma 6

*Proof.* By induction on $B_1$, $T$, $C$, and $K$.

1. Cases for $B_1$.
   - Case $x$. Then $\psi(x)[x := \psi(B)] = x[x := \psi(B)] = \psi(B) = \psi(x[x := B])$.
   - Case $y \neq x$. Then $\psi(y)[x := \psi(B)] = y[x := \psi(B)] = y = \psi(y) = \psi(y[x := B])$.
   - Case $\lambda y.S$. Then

$$\begin{array}{rcl}\psi(\lambda x.S)[x := \psi(B)] & = & (\lambda y.\sigma(S))[x := \psi(B)] \\ & = & \lambda y.\sigma(S[x := B]) \\ & & \text{by induction hypothesis} \\ & = & \psi((\lambda y.S)[x := B])\end{array}$$

2. Cases for $T$.
   - Case $\lambda k.C$. Then

$$\begin{array}{rcl}\sigma(\lambda k.C)[x := \psi(B)] & = & \overline{C}[x := \psi(B)] \\ & = & \overline{C[x := B]} \\ & & \text{by induction hypothesis} \\ & = & \sigma((\lambda k.C)[x := B])\end{array}$$

- Case 0. Then $\sigma(0)[x := \psi(B)] = 0 = \sigma(0[x := B])$.
- Case $\alpha.T$. Then

$$\begin{array}{rcl} \sigma(\alpha.T)[x := \psi(B)] & = & (\alpha.\sigma(T))[x := \psi(B)] \\ & = & \alpha.\sigma(T[x := B]) \\ & & \text{by induction hypothesis} \\ & = & \sigma((\alpha.T)[x := B]) \end{array}$$

- Case $T_1 + T_2$. Then

$$\begin{array}{rcl} \sigma(T_1 + T_2)[x := \psi(B)] & = & (\sigma(T_1) + \sigma(T_2))[x := \psi(B)] \\ & = & \sigma(T_1[x := B]) + \sigma(T_2[x := B]) \\ & & \text{by induction hypothesis} \\ & = & \sigma((T_1 + T_2)[x := B]) \end{array}$$

3. Cases for $C$.

- Case $KB_1$. Then

$$\begin{array}{rcl} \overline{KB_1}[x := \psi(B)] & = & \underline{K}[\psi(B_1)][x := \psi(B)] \\ & = & \underline{K[x := B]}[\psi(B_1)[x := \psi(B)]] \\ & & \text{by induction hypothesis} \\ & = & \underline{K[x := B]}[\psi(B_1[x := B])] \\ & & \text{by induction hypothesis} \\ & = & \overline{(KB_1)[x := B]} \end{array}$$

- Case $B_1B_2K$. Then

$$\begin{array}{rcl} \overline{B_1B_2K}[x := \psi(B)] & = & \underline{K}[\psi(B_1)\psi(B_2)][x := \psi(B)] \\ & = & \underline{K[x := B]}[(\psi(B_1)\psi(B_2))[x := \psi(B)]] \\ & & \text{by induction hypothesis} \\ & = & \underline{K[x := B]}[\psi(B_1[x := B])\psi(B_2[x := B])] \\ & & \text{by induction hypothesis} \\ & = & \overline{(B_1B_2K)[x := B]} \end{array}$$

- Case $TK$. Then

$$\begin{array}{rcl} \overline{TK}[x := \psi(B)] & = & \underline{K}[\sigma(T)][x := \psi(B)] \\ & = & \underline{K[x := B]}[\sigma(T)[x := \psi(B)]] \\ & & \text{by induction hypothesis} \\ & = & \underline{K[x := B]}[\sigma(T[x := B])] \\ & & \text{by induction hypothesis} \\ & = & \overline{(TK)[x := B]} \end{array}$$

4. Cases for $K$.

   - Case $k$. Then $\underline{k}[M][x := \psi(B)] = M[x := \psi(B)] = \underline{k[x := B]}[M[x := \psi(B)]]$.
   - Case $\lambda b.B_1 bK$. Then

$$\begin{aligned}
\underline{\lambda b.B_1 bK}[M][x := \psi(B)] &= \underline{K}[\psi(B_1)M][x := \psi(B)] \\
&= \underline{K[x := B]}[(\psi(B_1)M)[x := \psi(B)]] \\
&\quad \text{by induction hypothesis} \\
&= \underline{K[x := B]}[\psi(B_1[x := B])M[x := \psi(B)]] \\
&\quad \text{by induction hypothesis} \\
&= \underline{(\lambda b.B_1 bK)[x := B]}[M[x := \psi(B)]]
\end{aligned}$$

   - Case $\lambda b_1.T(\lambda b_2.b_1 b_2 K)$. Then

$$\begin{aligned}
\underline{\lambda b_1.T(\lambda b_2.b_1 b_2 K)}[M][x := \psi(B)] &= \underline{K}[M\sigma(T)][x := \psi(B)] \\
&= \underline{K[x := B]}[(M\sigma(T))[x := \psi(B)]] \\
&\quad \text{by induction hypothesis} \\
&= \underline{K[x := B]}[M[x := \psi(B)]\sigma(T[x := B])] \\
&\quad \text{by induction hypothesis} \\
&= \underline{(\lambda b_1.T(\lambda b_2.b_1 b_2 K))[x := B]}[M[x := \psi(B)]]
\end{aligned}$$

□

## A.4 Lemma 7

*Proof.* By induction on $K_2$.

- Case $k$. Then $\underline{K_1}[\underline{k}[M]] = \underline{K_1}[M] = \underline{k[k := K_1]}[M]$.
- Case $\lambda b.BbK$. Then

$$\begin{aligned}
\underline{K_1}[\underline{\lambda b.BbK}[M]] &= \underline{K_1}[\underline{K}[\psi(B)M]] \\
&= \underline{K[k := K_1]}[\psi(B)M] \\
&\quad \text{by induction hypothesis} \\
&= \underline{(\lambda b.BbK)[k := K_1]}[M]
\end{aligned}$$

- Case $\lambda b_1.T(\lambda b_2.b_1 b_2 K)$. Then

$$\begin{aligned}
\underline{K_1}[\underline{\lambda b_1.T(\lambda b_2.b_1 b_2 K)}[M]] &= \underline{K_1}[\underline{K}[M\sigma(T)]] \\
&= \underline{K[k := K_1]}[M\sigma(T)] \\
&\quad \text{by induction hypothesis} \\
&= \underline{(\lambda b_1.T(\lambda b_2.b_1 b_2 K))[k := K_1]}[M]
\end{aligned}$$

□

## A.5 Lemma 8

*Proof.* By induction on $C$.

- Case $K_2B$. Then

$$\begin{array}{rcl} \underline{K}[\overline{K_2B}] & = & \underline{K}[\underline{K_2}[\psi(B)]] \\ & = & \underline{K_2[k:=K]}[\psi(B)] \\ & & \text{by Lemma 7} \\ & = & \overline{(K_2B)[k:=K]} \end{array}$$

- Case $B_1B_2K_2$. Then

$$\begin{array}{rcl} \underline{K}[\overline{B_1B_2K_2}] & = & \underline{K}[\underline{K_2}[\psi(B_1)\psi(B_2)]] \\ & = & \underline{K_2[k:=K]}[\psi(B_1)\psi(B_2)] \\ & & \text{by Lemma 7} \\ & = & \overline{(B_1B_2K_2)[k:=K]} \end{array}$$

- Case $TK_2$. Then

$$\begin{array}{rcl} \underline{K}[\overline{TK_2}] & = & \underline{K}[\underline{K_2}[\sigma(T)]] \\ & = & \underline{K_2[k:=K]}[\sigma(T)] \\ & & \text{by Lemma 7} \\ & = & \overline{(TK_2)[k:=K]} \end{array}$$

□

## A.6 Lemma 9

*Proof.* By induction on $K$.

- Case $k$. Then $\underline{k}[M] = M \to_{l\cup\beta} M' = \underline{k}[M']$.
- Case $\lambda b.BbK$. Then $\psi(B)M \to_{l\cup\beta} \psi(B)M'$ since $\psi(B)$ is a base term, and

$$\begin{array}{rcl} \underline{(\lambda b.BbK)}[M] & = & \underline{K}[\psi(B)M] \\ & \to_{l\cup\beta} & \underline{K}[\psi(B)M'] \\ & & \text{by induction hypothesis} \\ & = & \underline{\lambda b.BbK}[M'] \end{array}$$

- Case $\lambda b_1.T(\lambda b_2.b_1b_2K)$. Then $M\sigma(T) \to_{l\cup\beta} M'\sigma(T)$, and

$$\begin{array}{rcl} \underline{\lambda b_1.T(\lambda b_2.b_1b_2K)}[M] & = & \underline{K}[M\sigma(T)] \\ & \to_{l\cup\beta} & K[M'\sigma(T)] \\ & & \text{by induction hypothesis} \\ & = & \underline{\lambda b_1.T(\lambda b_2.b_1b_2K)}[M'] \end{array}$$

□

### A.7 Lemma 10

*Proof.* By induction on $K$. We prove only the first statement, as the others are similar.

- Case $k$. Then $\underline{k}[M_1+M_2] = M_1+M_2 = \underline{k}[M_1]+\underline{k}[M_2]$.
- Case $\lambda b.BbK$. Then

$$\begin{array}{rcl}
\underline{\lambda b.BbK}[M_1+M_2] & = & \underline{K}[\psi(B)(M_1+M_2)] \\
& \to_l & \underline{K}[\psi(B)M_1+\psi(B)M_2] \\
& & \text{by Lemma 9} \\
& \to_l^* & \underline{K}[\psi(B)M_1]+\underline{K}[\psi(B)M_2] \\
& & \text{by induction hypothesis} \\
& = & \underline{\lambda b.BbK}[M_1]+\underline{\lambda b.BbK}[M_2]
\end{array}$$

- Case $\lambda b_1.S(\lambda b_2.b_1b_2K)$. Then

$$\begin{array}{rcl}
\underline{\lambda b_1.S(\lambda b_2.b_1b_2K)}[M_1+M_2] & = & \underline{K}[(M_1+M_2)\sigma(S)] \\
& \to_l & \underline{K}[M_1\sigma(S)+\psi(B)M_2\sigma(S)] \\
& & \text{by Lemma 9} \\
& \to_l^* & \underline{K}[M_1\sigma(S)]+\underline{K}[M_2\sigma(S)] \\
& & \text{by induction hypothesis} \\
& = & \underline{\lambda b_1.S(\lambda b_2.b_1b_2K)}[M_1]+\underline{\lambda b_1.S(\lambda b_2.b_1b_2K)}[M_2]
\end{array}$$

□

### A.8 Lemma 11

*Proof.* By induction on the reduction rule.

- Case $L$. Using linearity of $\sigma$. We give the following example.

$$\begin{array}{rcl}
\sigma(T_1+(T_2+T_3)) & = & \sigma(T_1)+(\sigma(T_2)+\sigma(T_3)) \\
& \to_l & (\sigma(T_1)+\sigma(T_2))+\sigma(T_3) \\
& = & \sigma((T_1+T_2)+T_3)
\end{array}$$

- Case $\xi$. Using linearity and the induction hypothesis. We give the following example. Consider the case $T_1+T_2 \to_a T_1'+T_2$ with $T_1 \to_a T_1'$. Then

$$\begin{array}{rcl}
\sigma(T_1+T_2) & = & \sigma(T_1)+\sigma(T_2) \\
& \to_l & \sigma(T_1')+\sigma(T_2) \\
& & \text{by induction hypothesis} \\
& = & \sigma(T_1'+T_2)
\end{array}$$

□

## A.9 Lemma 16

*Proof.* By induction on $M$.

- Case $x$. Then $\sigma(\{|x|\}) = \sigma(x) = x$.
- Case $\lambda x.M$. Then

$$\begin{aligned}
\sigma(\{|\lambda x.M|\}) &= \sigma(\lambda k.k(\lambda x.\{|M|\})) \\
&= \overline{k(\lambda x.\{|M|\})} \\
&= \underline{k}[\phi(\lambda x.\{|M|\})] \\
&= \phi(\lambda x.\{|M|\}) \\
&= \lambda x.\sigma(\{|M|\}) \\
&= \lambda x.M \\
&\quad \text{by induction hypothesis.}
\end{aligned}$$

- Case $MN$. Then

$$\begin{aligned}
\sigma(\{|MN|\}) &= \sigma(\lambda k.\{|M|\}(\lambda b.b\{|N|\}k)) \\
&= \overline{\{|M|\}(\lambda b.b\{|N|\}k)} \\
&= \underline{(\lambda b.b\{|N|\}k)}[\sigma(\{|M|\})] \\
&= \underline{k}[\sigma(\{|M|\})\sigma(\{|N|\})] \\
&= \sigma(\{|M|\})\sigma(\{|N|\}) \\
&= MN \\
&\quad \text{by induction hypothesis}
\end{aligned}$$

- Case 0. Then $\sigma(\{|0|\}) = \sigma(0) = 0$
- Case $\alpha.M$. Then

$$\begin{aligned}
\sigma(\{|\alpha.M|\}) &= \sigma(\lambda k.(\alpha.\{|M|\})k) \\
&= \overline{(\alpha.\{|M|\})k} \\
&= \underline{k}[\sigma(\alpha.\{|M|\})] \\
&= \sigma(\alpha.\{|M|\}) \\
&= \alpha.\sigma(\{|M|\}) \\
&= \alpha.M \\
&\quad \text{by induction hypothesis}
\end{aligned}$$

- Case $M+N$. Then

$$\begin{aligned}
\sigma(\{|M+N|\}) &= \sigma(\lambda k.(\{|M|\}+\{|N|\})k) \\
&= \overline{(\{|M|\}+\{|N|\})k} \\
&= \underline{k}[\sigma(\{|M|\}+\{|N|\})] \\
&= \sigma(\{|M|\}+\{|N|\}) \\
&= \sigma(\{|M|\})+\sigma(\{|N|\}) \\
&= M+N \\
&\quad \text{by induction hypothesis}
\end{aligned}$$

□

### A.10 Lemma 17

*Proof.* By induction on $V$.

- Case 0. Then $\overline{0 : k} = \overline{0} = 0$.
- Case $x$. Then $\overline{x : k} = \overline{xk} = \underline{k}[x] = x$.
- Case $\lambda x.M$. Then $\overline{\lambda x.M : k} = \overline{k\Phi(\lambda x.M)} = \sigma(\lambda k.k\Phi(\lambda x.M)) = \sigma(\{|\lambda x.M|\}) = \lambda x.M$ by Lemma 16.
- Case $\alpha.V$. Then $\overline{\alpha.V : k} = \overline{\alpha.(V : k)} = \alpha.\overline{V : k} = \alpha V$ by induction hypothesis.
- Case $U + V$. Then $\overline{U + V : k} = \overline{U : k + V : k} = \overline{U : k} + \overline{V : k} = U + V$ by induction hypothesis.

□

### A.11 Lemma 18

*Proof.* By induction on $B$, $T$, $C$, and $K$.

1. Cases for $B$.
   - Case $\lambda y.S$. Then

$$\begin{aligned} \phi(\lambda x.S_1)[x := \sigma(S)] &= (\lambda y.\sigma(S_1))[x := \sigma(S)] \\ &= \lambda y.\sigma(S_1[x := S]) \\ &\quad \text{by induction hypothesis} \\ &= \phi((\lambda y.S_1)[x := S]) \end{aligned}$$

2. Cases for $T$.
   - Case $x$. Then $\sigma(x)[x := \sigma(S)] = x[x := \sigma(S)] = \sigma(S) = \sigma(x[x := S])$.
   - Case $y \neq x$. Then $\sigma(y)[x := \sigma(S)] = y[x := \sigma(S)] = y = \sigma(y) = \sigma(y[x := S])$.
   - Case $\lambda k.C$. Then

$$\begin{aligned} \sigma(\lambda k.C)[x := \sigma(S)] &= \overline{C}[x := \sigma(S)] \\ &= \overline{C[x := S]} \\ &\quad \text{by induction hypothesis} \\ &= \sigma((\lambda k.C)[x := S]) \end{aligned}$$

   - Case 0. Then $\sigma(0)[x := \sigma(S)] = 0 = \sigma(0[x := S])$.
   - Case $\alpha.T$. Then

$$\begin{aligned} \sigma(\alpha.T)[x := \sigma(S)] &= (\alpha.\sigma(T))[x := \sigma(S)] \\ &= \alpha.\sigma(T[x := S]) \\ &\quad \text{by induction hypothesis} \\ &= \sigma((\alpha.T)[x := S]) \end{aligned}$$

- Case $T_1 + T_2$. Then

$$\begin{aligned}\sigma(T_1+T_2)[x:=\sigma(S)] &= (\sigma(T_1)+\sigma(T_2))[x:=\sigma(S)]\\ &= \sigma(T_1[x:=S])+\sigma(T_2[x:=S])\\ &\quad \text{by induction hypothesis}\\ &= \sigma((T_1+T_2)[x:=S])\end{aligned}$$

3. Cases for $C$.
   - Case $KB$. Then

$$\begin{aligned}\overline{KB}[x:=\sigma(S)] &= \underline{K}[\phi(B)][x:=\sigma(S)]\\ &= \underline{K[x:=S]}[\phi(B)[x:=\sigma(S)]]\\ &\quad \text{by induction hypothesis}\\ &= \underline{K[x:=S]}[\phi(B[x:=S])]\\ &\quad \text{by induction hypothesis}\\ &= \overline{(KB)[x:=S]}\end{aligned}$$

   - Case $BS_2K$. Then

$$\begin{aligned}\overline{BS_2K}[x:=\sigma(S)] &= \underline{K}[\psi(B)\sigma(S_2)][x:=\sigma(S)]\\ &= \underline{K[x:=S]}[(\psi(B)\sigma(S_2))[x:=\sigma(S)]]\\ &\quad \text{by induction hypothesis}\\ &= \underline{K[x:=S]}[\psi(B[x:=S])\sigma(S_2[x:=S])]\\ &\quad \text{by induction hypothesis}\\ &= \overline{(BS_2K)[x:=S]}\end{aligned}$$

   - Case $TK$. Then

$$\begin{aligned}\overline{TK}[x:=\sigma(S)] &= \underline{K}[\sigma(T)][x:=\sigma(S)]\\ &= \underline{K[x:=S]}[\sigma(T)[x:=\sigma(S)]]\\ &\quad \text{by induction hypothesis}\\ &= \underline{K[x:=S]}[\sigma(T[x:=S])]\\ &\quad \text{by induction hypothesis}\\ &= \overline{(TK)[x:=S]}\end{aligned}$$

4. Cases for $K$.
   - Case $k$. Then $\underline{k}[M][x:=\sigma(S)] = M[x:=\sigma(S)] = \underline{k[x:=S]}[M[x:=\sigma(S)]]$.
   - Case $\lambda b.bS_2K$. Then

$$\begin{aligned}\underline{\lambda b.bS_2K}[M][x:=\sigma(S)] &= \underline{K}[M\sigma(S_2)][x:=\sigma(S)]\\ &= \underline{K[x:=S]}[(M\sigma(S_2))[x:=\sigma(S)]]\\ &\quad \text{by induction hypothesis}\\ &= \underline{K[x:=S]}[M[x:=\sigma(S)]\sigma(S_2[x:=S])]\\ &\quad \text{by induction hypothesis}\\ &= \underline{(\lambda b.bS_2K)[x:=S]}[M[x:=\sigma(S)]]\end{aligned}$$

□

### A.12 Lemma 19

*Proof.* By induction on $K_2$.

- Case $k$. Then $\underline{K_1}[\underline{k}[M]] = \underline{K_1}[M] = \underline{k[k := K_1]}[M]$.
- Case $\lambda b.bSbK$. Then

$$\begin{array}{rcl} \underline{K_1}[\underline{\lambda b.bSK}[M]] & = & \underline{K_1}[\underline{K}[M\sigma(S)]] \\ & = & \underline{K[k := K_1]}[M\sigma(S)] \\ & & \text{by induction hypothesis} \\ & = & \underline{(\lambda b.bSK)[k := K_1]}[M] \end{array}$$

□

### A.13 Lemma 20

*Proof.* By induction on $C$.

- Case $K_2B$. Then

$$\begin{array}{rcl} \underline{K}[\overline{K_2B}] & = & \underline{K}[\underline{K_2}[\phi(B)]] \\ & = & \underline{K_2[k := K]}[\phi(B)] \\ & & \text{by Lemma 19} \\ & = & \overline{(K_2B)[k := K]} \end{array}$$

- Case $BSK_2$. Then

$$\begin{array}{rcl} \underline{K}[\overline{BSK_2}] & = & \underline{K}[\underline{K_2}[\phi(B)\sigma(S)]] \\ & = & \underline{K_2[k := K]}[\phi(B)\sigma(S)] \\ & & \text{by Lemma 19} \\ & = & \overline{(BSK_2)[k := K]} \end{array}$$

- Case $TK_2$. Then

$$\begin{array}{rcl} \underline{K}[\overline{TK_2}] & = & \underline{K}[\underline{K_2}[\sigma(T)]] \\ & = & \underline{K_2[k := K]}[\sigma(T)] \\ & & \text{by Lemma 19} \\ & = & \overline{(TK_2)[k := K]} \end{array}$$

□

### A.14 Lemma 21

*Proof.* By induction on $K$.

- Case $k$. Then $\underline{k}[M] = M \to_{a\cup\beta} M' = \underline{k}[M']$.
- Case $\lambda b.bSK$. Then $M\sigma(S) \to_{a\cup\beta} M'\sigma(S)$, and

$$\begin{array}{rcl} \underline{(\lambda b.bSK)}[M] & = & \underline{K}[M\sigma(S)] \\ & \to_{a\cup\beta} & \underline{K}[M'\sigma(S)] \\ & & \text{by induction hypothesis} \\ & = & \underline{\lambda b.bSK}[M'] \end{array}$$

□

### A.15 Lemma 22

*Proof.* By induction on $K$. We prove only the first statement, as the others are similar.

- Case $k$. Then $\underline{k}[M_1+M_2] = M_1+M_2 = \underline{k}[M_1]+\underline{k}[M_2]$.
- Case $\lambda b.bSK$. Then

$$\begin{array}{rcl} \underline{\lambda b.bSK}[M_1+M_2] & = & \underline{K}[(M_1+M_2)\sigma(S)] \\ & \to_a & \underline{K}[M_1\sigma(S)+\psi(B)M_2\sigma(S)] \\ & & \text{by Lemma 9} \\ & \to_a^* & \underline{K}[M_1\sigma(S)]+\underline{K}[M_2\sigma(S)] \\ & & \text{by induction hypothesis} \\ & = & \underline{\lambda b.bSK}[M_1]+\underline{\lambda b.bSK}[M_2] \end{array}$$

□

### A.16 Lemma 23

*Proof.* By induction on the reduction rule.

- Case $L$. Using linearity of $\sigma$. We give the following example.

$$\begin{array}{rcl} \sigma(T_1+(T_2+T_3)) & = & \sigma(T_1)+(\sigma(T_2)+\sigma(T_3)) \\ & \to_a & (\sigma(T_1)+\sigma(T_2))+\sigma(T_3) \\ & = & \sigma((T_1+T_2)+T_3) \end{array}$$

- Case $\xi$. Using linearity and the induction hypothesis. We give the following example. Consider the case $T_1+T_2 \to_l T_1'+T_2$ with $T_1 \to_l T_1'$. Then

$$\begin{array}{rcl} \sigma(T_1+T_2) & = & \sigma(T_1)+\sigma(T_2) \\ & \to_a & \sigma(T_1')+\sigma(T_2) \\ & & \text{by induction hypothesis} \\ & = & \sigma(T_1'+T_2) \end{array}$$

□