Logical Methods in Computer Science
Vol. 10(4:8)2014, pp. 1–40
www.lmcs-online.org

Submitted Feb. 27, 2014
Published Dec. 9, 2014

CALL-BY-VALUE, CALL-BY-NAME AND THE VECTORIAL BEHAVIOUR OF THE ALGEBRAIC λ-CALCULUS

ALI ASSAF [a], ALEJANDRO DÍAZ-CARO [b], SIMON PERDRIX [c], CHRISTINE TASSON [d], AND BENOÎT VALIRON [e]

[a] École Polytechnique, Route de Saclay, 91120 Palaiseau, France

[a] INRIA, 23 avenue d'Italie, CS 81321, 75214 Paris Cedex 13, France
e-mail address: ali.assaf@inria.fr

[b] Universidad Nacional de Quilmes, Roque Sáenz Peña 352, 1876 Bernal, Buenos Aires, Argentina
e-mail address: alejandro@diaz-caro.info

[c] CNRS & LORIA, 615, rue du Jardin Botanique, BP-101, 54602 Villers-lès-Nancy, France
e-mail address: simon.perdrix@loria.fr

[d,e] PPS, Université Paris-Diderot – Paris 7, CNRS UMR 7126, 75205 Paris Cedex 13, France
e-mail address: christine.tasson@pps.univ-paris-diderot.fr, benoit.valiron@monoidal.net

[e] I2M, Université Aix-Marseille,, CNRS UMR 7373, Campus de Luminy, case 907, F13288 Marseille, France

Abstract. We examine the relationship between the *algebraic λ-calculus*, a fragment of the differential λ-calculus and the *linear-algebraic λ-calculus*, a candidate λ-calculus for quantum computation. Both calculi are algebraic: each one is equipped with an additive and a scalar-multiplicative structure, and their set of terms is closed under linear combinations. However, the two languages were built using different approaches: the former is a call-by-name language whereas the latter is call-by-value; the former considers algebraic equalities whereas the latter approaches them through rewrite rules.

In this paper, we analyse how these different approaches relate to one another. To this end, we propose four canonical languages based on each of the possible choices: call-by-name versus call-by-value, algebraic equality versus algebraic rewriting. We show that the various languages simulate one another. Due to subtle interaction between beta-reduction and algebraic rewriting, to make the languages consistent some additional hypotheses such as confluence or normalisation might be required. We carefully devise the required properties for each proof, making them general enough to be valid for any sub-language satisfying the corresponding properties.

2012 ACM CCS: [**Theory of computation**]: Models of computation—Computability—Lambda calculus.

Key words and phrases: algebraic λ-calculus; linear-algebraic λ-calculus; CPS simulation.

[b] Partially supported by the French-Argentinian Laboratory in Computer Science INFINIS.

[e] Partially supported by the ANR project ANR-2010-BLAN-021301 LOGOI..

LOGICAL METHODS IN COMPUTER SCIENCE
DOI:10.2168/LMCS-10(4:8)2014

1. Introduction

Two algebraic versions of the λ-calculus arise independently in distinct contexts: the algebraic λ-calculus (ALG) [Vau09] and the linear algebraic λ-calculus (LINEAL) [AD08]. Both languages are extensions of λ-calculus where linear combinations of terms are also terms. The former has been introduced in the context of linear logic as a fragment of the differential λ-calculus [ER03]: the algebraic structure allows to gather in a non deterministic manner different terms, *i.e.* each term in the linear combination represents one possible execution. The latter has been introduced as a candidate λ-calculus for quantum computation: in LINEAL, a linear combination of terms reflects the phenomenon of superposition, *i.e.* the capability for a quantum system to be in two or more states at the same time. Our purpose is to study the connections between the two systems.

In both languages, functions which are linear combinations of terms are interpreted pointwise: $(\alpha.f + \beta.g)\ x = \alpha.(f)\ x + \beta.(g)\ x$, where "." denotes the scalar multiplication. The two languages differ in the treatment of the arguments. In LINEAL, in order to deal with the algebraic structure, any function is considered as a linear map: $(f)\ (\alpha.x + \beta.y) \to^* \alpha.(f)\ x + \beta.(f)\ y$, reflecting the fact that any quantum evolution is a linear map. It reflects a call-by-value behaviour in the sense that the argument is evaluated until one has a base term. Conversely, ALG has a call-by-name evolution: $(\lambda x\, M)\ N \to M[x := N]$, without any restriction on N. As a consequence, the evolutions are different as illustrated by the following example. In LINEAL, $(\lambda x\, (x)\ x)\ (\alpha.y + \beta.z) \to^* \alpha.(y)\ y + \beta.(z)\ z$ while in ALG, $(\lambda x\, (x)\ x)\ (\alpha.y + \beta.z) \to (\alpha.y + \beta.z)\ (\alpha.y + \beta.z) = \alpha^2.(y)\ y + \alpha\beta.(y)\ z + \beta\alpha.(z)\ y + \beta^2.(z)\ z$.

Because they were designed for different purposes, another difference appears between the two languages: the way the algebraic part of the calculus is treated. In LINEAL, the algebraic structure is captured with a rewrite system, whereas in ALG terms are considered up to algebraic equivalence.

The two choices – call-by-value versus call-by-name and algebraic equality versus algebraic reduction – allow one to construct four possible calculi. We name them $\lambda^{\to}_{lin}$, $\lambda^{=}_{lin}$, $\lambda^{\to}_{alg}$, and $\lambda^{=}_{alg}$. See Figure 1 where they are presented according to their evaluation policy and the way they take care of the algebraic part of the language.

Inspired by LINEAL and ALG, the operational semantics of these four languages differ slightly from the original ones to better emphasise their characteristics: the reduction strategy and the handling of algebraic structure.

A first modification is that in all four languages, we avoid reduction under lambda abstractions. As a consequence, contrary to ALG, the λ-abstraction is not linear anymore: $\lambda x\, (\alpha.M + \beta.N) \neq \alpha.\lambda x\, M + \beta.\lambda x\, N$. This restriction is a common restriction: reducing under λ could be considered as "optimising the program".

	call-by-name	call-by-value
algebraic reduction	$\lambda^{\to}_{alg}$	$\lambda^{\to}_{lin}$
algebraic equality	$\lambda^{=}_{alg}$	$\lambda^{=}_{lin}$

Figure 1: The four algebraic λ-calculi.

Concerning $\lambda^{\rightarrow}_{lin}$ and $\lambda^{=}_{lin}$, restrictions originally imposed in LINEAL on the rewrite system to ensure confluence are replaced by restrictions making $\lambda^{\rightarrow}_{lin}$ and $\lambda^{=}_{lin}$ call-by-value also in the algebraic part. For example, in the rule $(M + N)\ L \to (M)\ L + (N)\ L$ the condition that $M + N$ be closed-normal is replaced by the restriction of L to values. Notice that even in the original language LINEAL, waiving the restrictions makes sense when confluence can be ensured by other means, see *e.g.* [ADC11, Val10b]. Since this change in the strategy is not trivial, we prove that LINEAL and $\lambda^{\rightarrow}_{lin}$ share the same behaviour, result formalized in Theorems 2.2 and 2.3.

Our contribution in this paper is first to introduce the four canonical languages related to the original languages LINEAL and ALG, and then to show the relation between LINEAL and ALG, by proving the simulation of all these languages. We show that call-by-value algebraic λ-calculi simulate call-by-name ones and *vice versa* by extending the continuation passing style (CPS) translation [Plo75] to the algebraic case. We also provide simulations between algebraic equality and algebraic reduction in both directions. The simulations we prove are summed up in Figure 2. The solid arrows stand for theorems that do not require confluence or normalisation in their hypothesis whereas the dotted arrows stand for theorems requiring confluence, and the dashed arrows for theorems requiring strong normalisation. The star in ALG and LINEAL means that the languages do not allow reduction under λ.

Related works. This paper connects two collections of works: one stemming from linear logic, the other one from quantum computation.

An important line of work in linear logic is concerned with the development of quantitative semantics: semantics where types are interpreted as (topological) vector spaces and lambda-terms as power series. Such semantics include for example finiteness spaces [Ehr05, Tas09], and Köthe spaces [Ehr03]. The models naturally question the introduction of new constructs into the logic, to account for the structure stemming from the vector spaces. In a seminal paper, Ehrhard and Regnier [ER03] develop a lambda-calculus with a structure of module and a differential operator capturing the inner structure of finiteness spaces. Because of the original understanding of lambda-terms as power-series, the resulting lambda-calculus ends up naturally call-by-name: $(\lambda x\,(f\,x)\,x)(y + z)$ is $(f\,(y + z))(y + z)$ and not $(f\,y)\,y + (f\,z)\,z$, and lambda-terms are considered modulo the equations of a module. Its properties were later extensively studied by Vaux [Vau07, Vau09].

These results also shed some light on another kind of lambda-calculus: the resource calculus [Bou93]. In the resource calculus the term $(\lambda x\,(f\,x)\,x)[y, z]$ reduces to the non-deterministic superposition $(f\,y)\,z + (f\,z)\,y$. As a non-deterministic calculus, the resource calculus is equipped with a sum (the non-deterministic choice), and many of the tools and techniques developed along with quantitative semantics are applicable. For example, resource lambda-terms can be equipped with a finiteness structure [Ehr10], and can be made in relation with the representation as power series of lambda-terms [BCEM12, BHP13].

Another line of work focuses on lambda-calculi equipped with a module structure: works stemming from quantum computation. In quantum computation, the idea of working with linear combinations of terms is a natural one. Indeed, the state of a quantum register is modelled as a linear (complex) combination of classical states. If the states in superposition represent terms of a lambda-calculus, one naturally ends up with linear combinations of lambda-terms. However, it turns out not to be so simple: because quantum operations are supposed to preserve norm and orthogonality, van Tonder [vT04] shows that a general

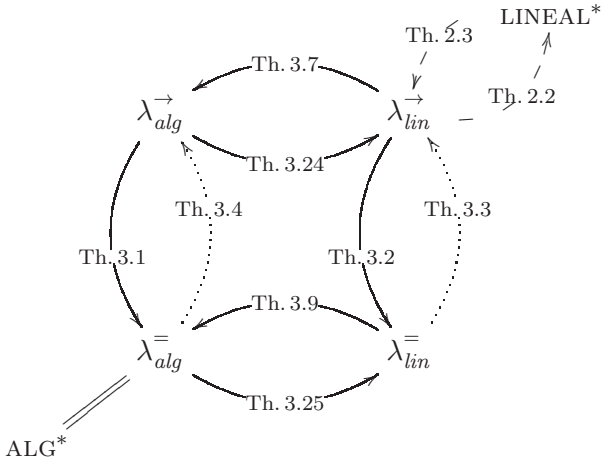
lambda-calculus encoded on quantum states with a reduction defined as a unitary map (i.e. an internal operation on quantum states), no non-trivial combinations of terms can occur, essentially falling back on a classical lambda-calculus manipulating quantum data.

In order to eventually figure out a solution to this problem, an obvious first-step is to study what would happen in a more general setting, and first study the computational aspect of vector spaces and lambda-calculi with vectorial structures. Arrighi and Dowek first proposed a computational definition of vector space [AD04] where the equations of vector spaces are oriented: a vector becomes a term evolving according to a confluent rewrite system. Later, they propose [AD08] one of the lambda-calculus with a vectorial structure on which we shall concentrate on this paper: LINEAL. The philosophy behind the presentation of the language is the following. First, beside associativity and commutativity, all the rewrite rules are oriented, whether they come from the regular structure of lambda-calculus or from the vector space structure. Then, the language is call-by-value in spirit: being an abstraction over quantum computation, a lambda-term is both an operator and a state. Therefore, unlike in the linear-logic approach, $(\lambda x\,(f\,x)\,x)(y+z)$ indeed corresponds to $(f\,y)\,y+(f\,z)\,z$.

Along with the description of the language, Arrighi and Dowek expose a main issue: while the language naïvely enjoy local confluence, the equational theory is inconsistent. They propose a solution by constraining the rewrite system, while other pieces of work explore other options. The first option proposed is by using a type system enforcing strong normalization, and thus recovering consistency. Several type systems capturing the algebraic aspect of the language have been developed to this end: a system-F type system with scalars [ADC11, ADC12], a system-F with a sum-type [BDCJ12, DCP12], a so-called "vectorial" type system, where linear combinations of types are valid types [ADCV12, ADCV13]. The second option proposed consists in minimally modifying the rewrite system and adding a type system; consistency is obtained through the development of a model [Val10b, Val13].

All these approaches are stepping stones working towards a unique goal: better understand the connection between lambda-calculus and vectorial structures, to eventually come back with an approach different from the one of van Tonder [vT04] for how to reconciliate lambda-calculi with vectorial structures and quantum computation. In [Val10a], one of the author hints at how one could do this, by using an intermediate language that can both be "run" by a quantum circuit and interpreted as a vectorial lambda-term.

A preliminary work-in-progress version of this paper has been presented in workshops in two parts in [DCPTV10] (informal) and [AP12]. The former briefly sketches a preliminary connection between call-by-value and call-by-name approaches in lambda-calculi with vectorial structures, using thunks to simulate call-by-name with call-by-value. In the current paper, we use instead CPS as it offers a nicer symmetric connection. The technique for the completeness of the simulation from call-by-value to call-by-name was developed in [AP12]. In these preliminary presentations, there are some small differences in the reduction rules with respect to the current work. The reason is that in the current work we tried to present the most general reduction strategy, adding as less conditions as possible. For example, in [DCPTV10] the left linearity rule is $(U+V)\,W \to U\,W+V\,W$ and requires all the subterms to be values in order to reduce, while in the present paper it is $(M+N)\,V \to M\,V+N\,V$ and only requires the argument to be a value.



$A \to B$ means "A is simulated by B"

Figure 2: Relations between the languages.

Plan of the paper. In Section 2.1, we present the original calculi LINEAL and ALG. In Section 2.2, we define the set of terms and the rewrite systems we consider in the paper. In Section 2.3, we establish the relation between the original setting and the setting used in this paper. In Section 2.4, we discuss the confluence of the algebraic rewrite systems. Section 3 is concerned with the actual simulations. In Section 3.1 we consider the correspondence between algebraic reduction and algebraic equality whereas in Sections 3.2 to 3.5 we consider the distinction call-by-name versus call-by-value. In Section 3.6, we show how the simulations can compose to obtain the correspondence between any two of the four languages. In Section 4 we conclude by providing some paths for future work. Omitted and sketched proofs are fully developed in the appendix.

2. The languages

In this section we present all the languages: the original setting, and our standardised versions of the algebraic calculi.

2.1. **The original setting.** The language LINEAL was first presented in [AD08] as summarised in Figure 3. The rewrite system is defined by structual induction on the left-hand-side. The factorisation and application rules ask for a particular subterm of the left-hand-side to *not* reduce (conditions (*) and (**)). Because of the inductive definition, this is indeed well-defined. These conditions (*) and (**) in particular ensure confluence: we refer the reader to the original paper [AD08] for more details.

The language ALG was first presented in [Vau09] as summarised in Figure 4. Notice the equality instead of rewrite in the treatment of the algebraic part of the language. Also note that it does not ask for any particular side-condition as it is the case in LINEAL.

Besides the treatment of the rewrite rules, we notice that LINEAL has a call-by-value behaviour while ALG has a call-by-name behaviour. However, we believe that it is not possible to draw in a straightforward manner a direct CPS translation between LINEAL and ALG. The intuitions behind this claim is that Vaux's calculus uses algebraic equalities

Terms: $M, N, L ::= B \mid (M)\ N \mid 0 \mid \alpha.M \mid M + N$
Base terms: $B ::= x \mid \lambda x\, M$

The rewrite system is defined inductively on the size of the left-hand-side.

Elementary rules:
$M + 0 \to M$,
$0.M \to 0$,
$1.M \to M$,
$\alpha.0 \to 0$,
$\alpha.(M + N) \to \alpha.M + \alpha.N$.

Beta reduction:
$(\lambda x\, M)\ B \to M[x := B]$.

Factorisation rules,
$\alpha.M + \beta.M \to (\alpha + \beta).M$ (*),
$\alpha.M + M \to (\alpha + 1).M$ (*),
$M + M \to (1 + 1).M$ (*),
$\alpha.(\beta.M) \to (\alpha\beta).M$.

Application rules:
$(M + N)\ L \to (M)\ L + (N)\ L$ (**),
$(L)\ (M+N) \to (L)\ M+(L)\ N$ (**),
$(\alpha.M)\ N \to \alpha.(M)\ N$ (*),
$(N)\ (\alpha.M) \to \alpha.(N)\ M$ (*),
$(0)\ M \to 0$,
$(M)0 \to 0$.

where $+$ is an associative-commutative (AC) symbol, $\alpha, \beta \in \mathcal{S}$, with $(\mathcal{S}, +, \times)$ a commutative ring and (see Section 2.1 for details)
(*) these rules apply only if M is closed and does *not* reduce.
(**) these rules apply only if $M + N$ is closed and does *not* reduce.

Figure 3: Syntax and reduction rules of LINEAL as they first appeared in [AD08]

Terms: $M, N, L ::= x \mid \lambda x\, M \mid (M)\ N \mid 0 \mid \alpha.M \mid M + N$

Axioms of commutative monoid:
$M + 0 = M$,
$M + N = N + M$,
$(M + N) + L = M + (N + L)$

Beta reduction:
$(\lambda x\, M)\ N \to M[x := N]$.

Axioms of module over the ring:
$\alpha.(M + N) = \alpha.M + \alpha.N$,
$\alpha.M + \beta.M = (\alpha + \beta).M$,
$\alpha.(\beta.M) = (\alpha\beta).M$,
$1.M = M$,
$0.M = M$,
$\alpha.0 = 0$.

Linearity in the λ-calculus:
$\lambda x\, 0 = 0$,
$\lambda x\, (\alpha.M) = \alpha.\lambda x\, M$,
$\lambda x\, (M + N) = \lambda x\, M + \lambda x\, N$,
$(0)\ M = 0$,
$(\alpha.M)\ N = \alpha.(M)\ N$,
$(M + N)\ L = (M)\ L + (N)\ L$.

Figure 4: Syntax and reduction rules of ALG as they first appeared in [Vau09].

in an unrestricted manner, whereas Lineal uses oriented algebraic rewrite rules in a very constraints manner (because of the side-conditions (*) and (**) on rules in Figure 3). A CPS translation from ALG to LINEAL would therefore have to force algebraic reductions sending for example $\alpha.x + \beta.x$ to $(\alpha + \beta).x$, which is feasable in ALG, but not in LINEAL. The translation would also have to deal with the case that ALG have unoriented algebraic

rewrites, whereas LINEAL has oriented ones. So the CPS translation would have to be able to map the translation of 0 to the translation of $0.M$, for all M.

In this paper, we fully desribe the connection between these two languages using a CPS translation, but the path we follow is therefore to first standardise the languages before establishing the simulations.

2.2. **Standardised algebraic λ-calculi.** The original languages LINEAL and ALG made particular assumptions both on the reduction strategy and the handling of algebraic structure under the reduction. In this paper, we consider separately the distinction call-by-name/call-by-value and the distinction algebraic equality/algebraic reduction. We develop therefore four languages: a call-by-value language $\lambda^{=}_{lin}$ with algebraic equality, a call-by-value language $\lambda^{\to}_{lin}$ with algebraic reduction, a call-by-name language $\lambda^{=}_{alg}$ with algebraic equality and a call-by-name language $\lambda^{\to}_{alg}$ with algebraic reduction. These four languages are summarised in Figure 1.

The languages share the same syntax, defined as follows:

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

where α ranges over a ring, the *ring of scalars*. We use the notation $M-N$ as a shorthand for $M+(-1).N$. Note that we could have asked for a semiring instead; in fact we shall see in Section 2.4 that the analysis we develop here can be adapted to semirings of scalars.

We summarise in Figure 5 all the rewrite rules. The rules are grouped with respect to their intuitive meaning. We use the usual notation regarding rewrite systems: Given a rewrite system R, we write R^* for its reflexive and transitive closure. That is, xR^*y is valid if $y=x$ or if there exists a rewrite sequence $x\,R\,x_1\,R\,\cdots\,R\,x_n\,R\,y$ linking x and y. We write $R^{\leftrightarrow}$ for the symmetric closure of R, that is, the relation that satisfies $x\,R^{\leftrightarrow}\,y$ if and only if $x\,R\,y$ or $y\,R\,x$.

Definition 2.1. We use the following notations for the rewrite systems obtained by combining the rules described in Figure 5:

$$\begin{array}{llllllll} \to_a & := & A\cup L\cup\xi & \to_\ell & := & A_l\cup A_r\cup L\cup\xi\cup\xi_{\lambda_{lin}} & \to_{\beta_v} & := & \beta_v\cup\xi\cup\xi_{\lambda_{lin}}\\ \overset{=}{\to}_a & := & (\to_a)^{\leftrightarrow} & \overset{=}{\to}_\ell & := & (\to_\ell)^{\leftrightarrow} & \to_{\beta_n} & := & \beta_n\cup\xi \end{array}$$

We hence define the following four languages and their associated rewrite systems:

Language	Corresponding Rewrite System
$\lambda^{\to}_{lin}$	$\to_{\ell\cup\beta_v} := (\to_\ell)\cup(\to_{\beta_v})$
$\lambda^{=}_{lin}$	$\overset{=}{\to}_{\ell\cup\beta_v} := (\overset{=}{\to}_\ell)\cup(\to_{\beta_v})$
$\lambda^{\to}_{alg}$	$\to_{a\cup\beta_n} := (\to_a)\cup(\to_{\beta_n})$
$\lambda^{=}_{alg}$	$\overset{=}{\to}_{a\cup\beta_n} := (\overset{=}{\to}_a)\cup(\to_{\beta_n})$

The "=" symbol over the arrow is to refer to $\lambda^{=}_{lin}$ and $\lambda^{=}_{alg}$, it does not mean that the beta rule is also reflexive.

For each language, the notion of *normal form* is defined as follows:

- a term M is in *normal form with respect to* $\to_{\ell\cup\beta_v}$ (respectively $\to_{a\cup\beta_n}$) if there is no term M such that $M\to_{\ell\cup\beta_v} N$ (respectively $M\to_{a\cup\beta_n} N$).

Specific rules for $\lambda^{\to}_{alg}$ and $\lambda^{=}_{alg}$

Call-by-name (β_n)

$$(\lambda x\, M)\ N \quad \to \quad M[x := N]$$

Linearity of the application (A)

$$\begin{array}{rcl} (M+N)\ L & \to & (M)\ L + (N)\ L \\ (\alpha.M)\ N & \to & \alpha.(M)\ N \\ (0)\ M & \to & 0 \end{array}$$

Specific rules for $\lambda^{\to}_{lin}$ and $\lambda^{=}_{lin}$

Call-by-value (β_v)

$$(\lambda x\, M)\ B \quad \to \quad M[x := B]$$

Context rule ($\xi_{\lambda_{lin}}$)

$$\frac{M \to M'}{(V)\ M \to (V)\ M'}$$

Linearity of the application

Left linearity (A_l)

$$\begin{array}{rcl} (M+N)\ V & \to & (M)\ V + (N)\ V \\ (\alpha.M)\ V & \to & \alpha.(M)\ V \\ (0)\ V & \to & 0 \end{array}$$

Right linearity (A_r)

$$\begin{array}{rcl} (B)\ (M+N) & \to & (B)\ M + (B)\ N \\ (B)\ (\alpha.M) & \to & \alpha.(B)\ M \\ (B)\ 0 & \to & 0 \end{array}$$

Common rules

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

Associativity ($Asso$)

$$\begin{array}{rcl} M + (N + L) & \to & (M + N) + L \\ (M + N) + L & \to & M + (N + L) \end{array}$$

Commutativity (Com)

$$M + N \quad \to \quad N + M$$

Factorisation (F)

$$\begin{array}{rcl} \alpha.M + \beta.M & \to & (\alpha+\beta).M \\ \alpha.M + M & \to & (\alpha+1).M \\ M + M & \to & (1+1).M \\ \alpha.(\beta.M) & \to & (\alpha\beta).M \end{array}$$

Simplification (S)

$$\begin{array}{rcl} \alpha.(M+N) & \to & \alpha.M + \alpha.N \\ 1.M & \to & M \\ 0.M & \to & 0 \\ \alpha.0 & \to & 0 \\ 0 + M & \to & M \end{array}$$

Context rules (ξ)

$$\frac{M \to M'}{(M)\ N \to (M')\ N} \qquad \frac{M \to M'}{M + N \to M' + N} \qquad \frac{N \to N'}{M + N \to M + N'} \qquad \frac{M \to M'}{\alpha.M \to \alpha.M'}$$

Figure 5: Rewrite rules with U, V and W values, B a basis term, M, M', N, N' and L any terms.

- a term M is in *normal form with respect to* $\Rightarrow_{\ell\cup\beta_v}$ (respectively $\Rightarrow_{a\cup\beta_n}$) if for all terms M such that $M \Rightarrow_{\ell\cup\beta_v} N$ (respectively $M \Rightarrow_{a\cup\beta_n} N$), $N \Rightarrow_{\ell} M$ (respectively $N \Rightarrow_{a} M$).

In other word, a term is in normal form is it does not reduce, potentially modulo the algebraic equalities inscribed within the rewrite system.

2.3. **Relation between the standardised languages and the original settings.** Let LINEAL* and ALG* be LINEAL and ALG respectively without reduction under λ. Not reducing under λ in ALG implies also removing the first three rules of "Linearity in the λ-calculus". Also, since we consider $\lambda^{\rightarrow}_{alg}$ with algebraic rewrite rules instead of the equalities used in ALG, we need two extra rules: $\alpha.M+M \to (\alpha+1).M$ and $M+M \to (1+1).M$. These rules were not needed with equalities, because $M = 1.M$. It is easy to check that ALG* and $\lambda^{=}_{alg}$ are actually the same language. However, $\lambda^{\rightarrow}_{lin}$ differs from LINEAL*. Besides eliminating conditions (*) and (**) (*cf.* Figure 3), we enforce a full call-by-value strategy in the algebraic part. The following results (Theorems 2.2 and 2.3) show that $\lambda^{\rightarrow}_{lin}$ and LINEAL* still have the same behaviour, in the sense that if M reduces to N either in LINEAL* or in $\lambda^{\rightarrow}_{lin}$, then there exists L such that both M and N reduce to L in the other language. We use the notation $\to_{\mathcal{L}}$ for reductions in LINEAL*.

Terms in normal form in LINEAL* are simply non-reducing terms, whereas normal forms in ALG* are non-reducing terms *modulo* the equalities on Figure 4.

Theorem 2.2. If M is closed and strongly normalising in LINEAL* and $M \to_{\ell\cup\beta_v} N$, then there exists L such that $M \to^*_{\mathcal{L}} L$ and $N \to^*_{\mathcal{L}} L$.

Proof. The proof is done by induction on the $\to_{\ell\cup\beta_v}$ rewrite relation. The details can be found in Appendix A.1. □

Theorem 2.3. If M is closed and strongly normalising in LINEAL and $M \to_{\mathcal{L}} N$, then there exists L such that $M \to^*_{\ell\cup\beta_v} L$ and $N \to^*_{\ell\cup\beta_v} L$.

Proof. We only need to verify the seven differing rules between the two languages. Notice that the normal form of a closed term is a value V. The full details can be found in Appendix A.2. □

2.4. **Discussion on consistency and confluence.** It is known that, without restrictions, both algebraic reductions and algebraic equalities cause problems of consistency, albeit differently. The original languages solve these problems using different techniques.

Let $Y_M = (\lambda x\,(M + (x)\ x))\ \lambda x\,(M + (x)\ x)$. In a system with algebraic reduction such as LINEAL, $\lambda^{\rightarrow}_{lin}$, or $\lambda^{\rightarrow}_{alg}$, the term $Y_M - Y_M$ reduces to 0, but also reduces to $M + Y_M - Y_M$ and hence to M, breaking confluence. In a system with algebraic equalities such as ALG, $\lambda^{=}_{lin}$, or $\lambda^{=}_{alg}$, it is even worse: for any terms M and N, because algebraic rewrites are not oriented, the following reductions hold

$$M \Rightarrow_a M + 0 \Rightarrow_a M + (Y_{N-M} - Y_{N-M}) \Rightarrow_{a\cup\beta_n} M + ((N - M + Y_{N-M}) - Y_{N-M}) \Rightarrow^*_a N$$

making any term M reduce to any other term N.

To solve this issue, several distinct techniques can be used to make an algebraic calculus confluent. In the original presentation of LINEAL (see Figure 3), restrictions on reduction rules are introduced. For example, $\alpha.M + \beta.M \to (\alpha + \beta).M$ only if M is closed and

normal (i.e. does not itself reduces under the same set of rules). Otherwise, in [ADC11, ADCV12, BDCJ12, Val10b, Val10a], type systems are set up instead, to forbid diverging terms (modulo associativity and commutativity of "+", or AC) such as Y_M. Finally, in ALG [Vau09] a restriction to positive scalars is proposed to solve the problem.

Since any of these techniques will solve the problem, in this paper we do not make a choice *a priori*. Instead, we show that the simulations between the calculi are correct, providing a methodology general enough to work in a large variety of restrictions on the languages. Therefore, we do not force a specific method to make the calculi consistent, leaving the choice to the reader.

The simulation theorems that we develop in this paper are correct in a general untyped setting (and in fact trivially true when we simulate a language with algebraic reduction with a language with algebraic equality as remarked above), but also true if one restricts the scalars to a semiring (as done in [Vau09]), or if we restrict the terms to any typed setting, provided that the languages $\lambda^{\to}_{lin}$ and $\lambda^{\to}_{alg}$ satisfy subject reduction and that the CPS translations given in Section 3 preserve typability. We propose a notion of *language fragments* to parameterise the simulation results. The definition of a fragment is general enough to capture many settings: various typed systems, but also the restrictions to a given set of terms such as the set of strongly normalising terms modulo AC or taking scalars from a semiring.

We define formally a fragment in the following way:

Definition 2.4. A *fragment* S of $\lambda^{\to}_{lin}$ (respectively $\lambda^{\to}_{alg}$) is a subset of terms closed under $\to_{\ell\cup\beta_v}$-reduction (respectively $\to_{a\cup\beta_n}$-reduction) together with the rewrite system inherited from $\lambda^{\to}_{lin}$ (respectively $\lambda^{\to}_{alg}$).

The definition of a fragment in the presence of algebraic equalities should be treated carefully. Indeed, note that the algebraic equalities are defined as $M \to^{=} N$ if and only if $M \to N$ or $N \to M$. As a consequence, for any subset S of terms closed under $\to^{=}$-reduction, if M is in S then for any N (in S or not), $M + N - N \in S$ since $M \to^{=} M + N - N$. We therefore need to define the algebraic equality with respect to the particular subset of terms under consideration.

Definition 2.5. A *fragment* S of $\lambda^{=}_{lin}$ (respectively $\lambda^{=}_{alg}$) is a fragment of $\lambda^{\to}_{lin}$ (respectively $\lambda^{\to}_{alg}$) together with an algebraic equality defined as $M \equiv\!\!\to^{S}_{\ell} N$ (respectively $M \equiv\!\!\to^{S}_{a} N$) if and only if $M, N \in S$ and $N \to_\ell M$ or $M \to_\ell N$ (respectively $N \to_a M$ or $M \to_a N$). The β-reduction is not modified.

Convention 2.6. When referring to a fragment of $\lambda^{=}_{lin}$ (respectively $\lambda^{=}_{alg}$), we use $\equiv\!\!\to_\ell$ (respectively $\equiv\!\!\to_a$) instead of $\equiv\!\!\to^{S}_{\ell}$ (respectively $\equiv\!\!\to^{S}_{a}$) for the restricted rewrite system when the fragment under consideration is clear.

3. Simulations

The core of the paper is concerned with the mutual simulations of the four languages. The first class of problems relates algebraic reduction with algebraic equality. While simulating a language with algebraic reduction using a language with algebraic equality is not difficult, going in the opposite direction is not possible in general. Indeed, while $0 =_\ell Y_M - Y_M \to_{\beta_v} Y_M + M - Y_M =_\ell M$ in $\lambda^{=}_{lin}$ (where $Y_M = (\lambda x\,(M + (x)\ x))\ \lambda x\,(M + (x)\ x)$), it is impossible

to reduce 0 reduce to M in $\lambda^{\rightarrow}_{lin}$, because 0 does not appear on the left-hand-side of any rule in Figure 5. In this section, we show that a fragment of a language with algebraic equality can be simulated by the corresponding fragment with algebraic reduction provided that the latter is *confluent* (Theorems 3.3 and 3.4).

The second class of problems is concerned with call-by-value and call-by-name. In this paper, the simulations of call-by-name by call-by-value and its reverse are treated using continuation passing style (CPS), extending the techniques described in [Fis72, Plo75] to the algebraic case (Theorems 3.7, 3.9, 3.24 and 3.25).

The results are summarised in Figure 2. Solid arrows correspond to results where no particular hypothesis on the language is made. Dotted arrows correspond to results where confluence is required. Dashed arrows correspond to results where strong normalisation is required.

3.1. **Algebraic reduction versus algebraic equality.** As the relation $\rightarrow_{\ell\cup\beta_v}$ is contained in $\Rightarrow_{\ell\cup\beta_v}$ and the relation $\rightarrow_{a\cup\beta_n}$ is contained in $\Rightarrow_{a\cup\beta_n}$, the first simulation theorems are trivial.

Theorem 3.1. For any term M if $M \rightarrow_{a\cup\beta_n} N$, then $M \Rightarrow_{a\cup\beta_n} N$. □

Theorem 3.2. For any term M if $M \rightarrow_{\ell\cup\beta_v} N$, then $M \Rightarrow_{\ell\cup\beta_v} N$. □

The simulations going in the other direction are only valid in the presence of confluence. In the following two theorems, the algebraic equality is defined with respect to the considered fragment (see Definition 2.5.)

Theorem 3.3. For any term M in a confluent fragment of $\lambda^{\rightarrow}_{lin}$, if $M \Rightarrow^*_{\ell\cup\beta_v} V$, then there exists V' such that $M \rightarrow^*_{\ell\cup\beta_v} V'$ and $V \Rightarrow^*_{\ell} V'$.

Proof. By induction on the length of the reduction. The proof mainly relies on the confluence of the fragment. The details can be found in Appendix A.3. □

Theorem 3.4. For any term M in a confluent fragment of $\lambda^{\rightarrow}_{alg}$, if $M \Rightarrow^*_{a\cup\beta_n} V$, then there exists V' such that $M \rightarrow^*_{a\cup\beta_n} V'$ and $V \Rightarrow^*_{a} V'$.

Proof. Similar to the previous theorem. □

3.2. **Call-by-name simulates call-by-value.** To prove the simulation of $\lambda^{\rightarrow}_{lin}$ in $\lambda^{\rightarrow}_{alg}$ and the simulation of $\lambda^{=}_{lin}$ in $\lambda^{=}_{alg}$, we introduce an algebraic extension of the continuation passing style translation used to prove that call-by-name simulates call-by-value in the regular λ-calculus [Plo75].

Let $[\![\cdot]\!]$ be the following encoding of terms from $\lambda^{\rightarrow}_{lin}/\lambda^{=}_{lin}$ to $\lambda^{\rightarrow}_{alg}/\lambda^{=}_{alg}$, where f, g and h are fresh variables.

$$\begin{array}{rclcrcl}
[\![x]\!] & = & \lambda f\ (f)\ x, & & [\![0]\!] & = & 0, \\
[\![\lambda x\, M]\!] & = & \lambda f\,(f)\ \lambda x\,[\![M]\!], & & [\![(M)\ N]\!] & = & \lambda f\,([\![M]\!])\ \lambda g\,([\![N]\!])\ \lambda h\,((g)\ h)\ f, \\
[\![\alpha.M]\!] & = & \lambda f\,(\alpha.[\![M]\!])\ f, & & [\![M+N]\!] & = & \lambda f\,([\![M]\!]+[\![N]\!])\ f.
\end{array}$$

Note that the CPS translation for the addition $(M + N)$ is defined in a non-standard manner. Indeed, if it were, say, a pairing construct, in call-by-name, the term $M + N$ would always be a value, whereas in call-by-value it would not necessarily be (for example if M or N reduces). One would therefore expect the CPS translation of $M + N$ to be

$\lambda f\,([\![M]\!])\lambda x\,([\![N]\!])\lambda y\,(f)(x+y)$ (similarly as it is done for the application) In the languages we are considering, however, if M or N reduces, then $M+N$ reduces, even in a call-by-name setting. Thus the definition we instead pick.

Let Ψ be the encoding of base terms defined by $\Psi(x) = x$, $\Psi(\lambda x\, M) = \lambda x\,[\![M]\!]$, This encoding is compatible with substitution:

Lemma 3.5. $[\![M[x := B]]\!] = [\![M]\!][x := \Psi(B)]$ with B a base term.

Proof. By induction on M. The details can be found in Appendix A.4. □

We also define a convenient infix operation (:) capturing the behaviour of translated terms. For example, if B is a base term, *i.e.* a variable or an abstraction, then its translation into $\lambda^{\to}_{alg}$ is $[\![B]\!] = \lambda f\ (f)\ \Psi(B)$. If we apply this translated term to a certain K, we obtain $(\lambda f\ (f)\ \Psi(B))\ K \to_{a\cup\beta_n} (K)\ \Psi(B)$. We define $B : K = (K)\ \Psi(B)$ and get that $([\![B]\!])\ K \to_{a\cup\beta_n} B : K$.

Definition 3.6. Let (:) be the infix binary operation defined by:

$$\begin{array}{rl} 0 : K = 0 & (0)\ N : K = 0 \\ B : K = (K)\ \Psi(B) & (B)\ N : K = N : \lambda f\,((\Psi(B))\ f)\ K \\ \alpha.M : K = \alpha.(M : K) & (\alpha.M)\ N : K = \alpha.(M)\ N : K \\ M + N : K = M : K + N : K & (M+N)\ L : K = ((M)\ L + (N)\ L) : K \\ & ((M)\,N)\,L : K = (M)\,N{:}\lambda\, g\,([\![L]\!])\,\lambda\, h\,((g)\,h)\,K \end{array}$$

Using this encoding, we can simulate $\lambda^{\to}_{lin}$ with $\lambda^{\to}_{alg}$, as formalised in the following theorem. The proof of this theorem is given later in this section.

Theorem 3.7 (Simulation)**.** For any term M and variable k, if $M \to^*_{\ell\cup\beta_v} V$ where V is a value, then $([\![M]\!])\ k \to^*_{a\cup\beta_n} V : k$.

Example 3.8. For any terms M and N, let $\langle M, N\rangle := \lambda y\,((y)\,M)\,N$. Let $\mathsf{copy} = \lambda x\ \langle x, x\rangle$, and let $B_1 = \lambda x\, N_1$ and $B_2 = \lambda x\, N_2$ be two base terms. Then $(\mathsf{copy})\ (B_1 + B_2) \to^*_{\ell\cup\beta_v} \langle B_1, B_1\rangle + \langle B_2, B_2\rangle$ and $(\mathsf{copy})\ (B_1 + B_2) \to^*_{a\cup\beta_n} \langle B_1 + B_2, B_1 + B_2\rangle$.

We consider the simulation $\lambda^{\to}_{lin}$ to $\lambda^{\to}_{alg}$.

$$[\![(\mathsf{copy})\ (B_1 + B_2)]\!] = \lambda f\,([\![\mathsf{copy}]\!])\,\lambda g\,([\![B_1 + B_2]\!])\lambda h\,((g)\,h)\,f\ ,$$

where $[\![\mathsf{copy}]\!] = \lambda f\,(f)\,\lambda x\,[\![\langle x, x\rangle]\!]$, with $[\![\langle M, N\rangle]\!] = \lambda f\,(f)\,\Psi(\langle M, N\rangle)$, and $[\![B_1 + B_2]\!] = \lambda f\,([\![B_1]\!] + [\![B_2]\!])\,f$, with $[\![B_1]\!] = \lambda g\,(g)\,\Psi(B_1)$.

$$\begin{aligned}
([\![(\mathsf{copy})\ (B_1 + B_2)]\!])\,k &\to_{a\cup\beta_n} ([\![\mathsf{copy}]\!])\,\lambda g\,([\![B_1 + B_2]\!])\lambda h\,((g)\,h)\,k \\
&= (\lambda f\,(f)\,\lambda x\,[\![\langle x, x\rangle]\!])\,\lambda g\,([\![B_1 + B_2]\!])\lambda h\,((g)\,h)\,k \\
&\to_{a\cup\beta_n} (\lambda g\,([\![B_1 + B_2]\!])\,\lambda h\,((g)\,h)\,k)\,\lambda x\,[\![\langle x, x\rangle]\!] \\
&\to_{a\cup\beta_n} ([\![B_1 + B_2]\!])\,\lambda h\,((\lambda x\,[\![\langle x, x\rangle]\!])\,h)\,k \\
&\to_{a\cup\beta_n} ([\![B]\!]_1 + [\![B]\!]_2)\ \lambda h\,((\lambda x\,[\![\langle x, x\rangle]\!])\,h)\,k \\
&\to_{a\cup\beta_n} ([\![B]\!]_1)\ \lambda h\,((\lambda x\,[\![\langle x, x\rangle]\!])\,h)\,k + ([\![B]\!]_2)\ \lambda h\,((\lambda x\,[\![\langle x, x\rangle]\!])\,h)\,k \\
&\to^*_{a\cup\beta_n} (\lambda h\,((\lambda x\,[\![\langle x, x\rangle]\!])\,h)\,k)\,\Psi(B_1) + (\lambda h\,((\lambda x\,[\![\langle x, x\rangle]\!])\,h)\,k)\,\Psi(B_2) \\
&\to^*_{a\cup\beta_n} ((\lambda x\,[\![\langle x, x\rangle]\!])\,\Psi(B_1))\,k + ((\lambda x\,[\![\langle x, x\rangle]\!])\,\Psi(B_2))\,k \\
&\to^*_{a\cup\beta_n} ([\![\langle x, x\rangle]\!][x := \Psi(B_1)])\,k + ([\![\langle x, x\rangle]\!][x := \Psi(B_2)])\,k
\end{aligned}$$

$$\begin{aligned}
&\to^*_{a\cup\beta_n} (\llbracket\langle B_1, B_1\rangle\rrbracket)\, k + (\llbracket\langle B_2, B_2\rangle\rrbracket)\, k \quad \text{(Lemma 3.5)}\\
&\to^*_{a\cup\beta_n} (k)\, \Psi(\langle B_1, B_1\rangle) + (k)\, \Psi(\langle B_2, B_2\rangle)\\
&= (\langle B_1, B_1\rangle + \langle B_2, B_2\rangle) : k
\end{aligned}$$

Similarly, one can relate fragments of $\lambda^{=}_{alg}$ to fragments of $\lambda^{=}_{lin}$ as follows.

Theorem 3.9 (Simulation)**.** For any two fragments S_ℓ of $\lambda^{=}_{lin}$ and S_a of $\lambda^{=}_{alg}$ and variable k, such that $\forall M \in S_\ell$, $(\llbracket M\rrbracket)\, k \in S_a$, and for any term M in S_ℓ, if $M \Rrightarrow^*_{\ell\cup\beta_v} V$ where V is a value, then $(\llbracket M\rrbracket)\, k \Rrightarrow^*_{a\cup\beta_n} V : k$.

The proof of Theorem 3.9 is detailed later in this section.

Remark 3.10. As we already noted several times in this paper, without restricting the languages, Theorem 3.9 would be trivial. If any term reduce to any other one, the desired reduction would of course be valid without restriction. This theorem shows that if the calculi are restricted to fragments, the result is still true. One example of such fragments is found by taking the restriction of scalars to non-negative elements, as in [Vau09].

Once a term is encoded it can be reduced either by $\to^*_{a\cup\beta_n}$ or by $\to^*_{\ell\cup\beta_v}$ (respectively $\Rrightarrow^*_{a\cup\beta_n}$ or $\Rrightarrow^*_{\ell\cup\beta_v}$) without distinction, and still obtain the same result. We state this fact as a corollary:

Corollary 3.11 (Indifference)**.**

(1) For any term M and variable k, if $M \to^*_{\ell\cup\beta_v} V$ where V is a value, then
$$(\llbracket M\rrbracket)\, k \to^*_{\ell\cup\beta_v} V : k$$
(2) For any fragment S of $\lambda^{=}_{lin}$ and variable k, such that $\forall M \in S$, $(\llbracket M\rrbracket)\, k \in S$, and for any term M in S, if $M \Rrightarrow^*_{\ell\cup\beta_v} V$ where V is a value, then
$$(\llbracket M\rrbracket)\, k \Rrightarrow^*_{\ell\cup\beta_v} V : k$$

Proof. It suffices to check that in the proofs of Theorems 3.7 and 3.9 all the reductions $\to^*_{a\cup\beta_n}$ are done by rules common to both languages. □

Example 3.12. Note that in Example 3.8 one could have as well rewritten with $\to_{\ell\cup\beta_v}$, which illustrates the indifference property (Corollary 3.11).

We now proceed to prove Theorems 3.7 and 3.9. We extend the proof in [Plo75] to the algebraic case.

Lemma 3.13. If K is a base term, then for any term M, $(\llbracket M\rrbracket)\, K \to^*_{a\cup\beta_n} M : K$.

Proof. By structural induction on M. The details can be found in Appendix A.5. □

The following lemma and corollary state that the (:) operation preserves reduction.

Lemma 3.14. If $M \to_\ell N$ then for all K base term, $M : K \to^*_a N : K$.

Proof. Case by case on the rules $\to_\ell$. The details can be found in Appendix A.6. □

Lemma 3.15. If $M \to_{\ell\cup\beta_v} N$ then for all K base term, $M : K \to^*_{a\cup\beta_n} N : K$.

Proof. Case by case on the rules $\to_{\ell\cup\beta_v}$. The details can be found in Appendix A.7. □

Corollary 3.16. If $M \Rightarrow_{\ell\cup\beta_v} N$ then for all K base terms, $M : K \Rightarrow^*_{a\cup\beta_n} N : K$.

Proof. By case distinction. If $M \to_{\ell\cup\beta_v} N$, then by Lemma 3.15, $M : K \to^*_{a\cup\beta_n} N : K$, which implies $M : K \Rightarrow^*_{a\cup\beta_n} N : K$. If $N \to_\ell M$, then by Lemma 3.14, $N : K \to^*_a M : K$, which also implies $M : K \Rightarrow^*_{a\cup\beta_n} N : K$. □

Now the proofs of Theorems 3.7 and 3.9 go as follows.

Proof of Theorem 3.7. Using Lemmas 3.13 and 3.15, we can deduce that the reduction $(\llbracket M \rrbracket)\ k \to^*_{a\cup\beta_n} M : k \to^*_{a\cup\beta_n} V : k$ happens. □

Proof of Theorem 3.9. From Lemma 3.13, $(\llbracket M \rrbracket)\ k \to^*_{a\cup\beta_n} M : k$, hence we also have $(\llbracket M \rrbracket)\ k \Rightarrow^*_{a\cup\beta_n} M : k$. From Corollary 3.16, this latter term $\Rightarrow^*_{a\cup\beta_n}$-reduces to $V : k$. Note that since $(\llbracket M \rrbracket)\ k \in S_a$, $M : k$ is also in S_ℓ due to the closeness under $\Rightarrow_a$ of S_a. The same applies to $M : k$, thus also to $V : k$. □

3.3. **Completeness of the call-by-value to call-by-name simulation.** We show that the converse of Theorem 3.7 is also true:

Theorem 3.17 (Completeness)**.** For any term M and variable k, if $\llbracket M \rrbracket k \to^*_{a\cup\beta_n} V : k$ then $M \to^*_{\ell\cup\beta_v} V$.

To prove it, we define an inverse translation and show that it preserves reductions. First, we need to characterise the structure of the encoded terms. We define a subset of $\lambda^{\to}_{alg}/\lambda^{=}_{alg}$ which contains the image of the translation and is closed under $\to_{a\cup\beta_n}$ reductions with the following grammar:

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

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 $\llbracket M \rrbracket$ gives a term of the class T, while $(\llbracket M \rrbracket)\ k$ and $M : K$ are of class D. One can easily check that each of the classes D, T, K and B is closed under $\to_{a\cup\beta_n}$ reductions.

For convenience, in this section we put some restrictions on the names of the variables in the grammar. The variable name k that appears in the class K is 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: they cannot appear (free) in any sub-term. In particular, these restrictions ensure that the grammar *for each category* 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 terms that simulate the steps of the reductions, hence the name. They are the only terms that contain top-level applications, so they are the only terms that can β-reduce. In fact, notice that the arguments in applications are always base values. This shows a simple alternative proof for Corollary 3.11.

We define the inverse translation using the following four functions, corresponding to each of the four main categories in the grammar. These functions are well-defined because the grammar for each category is unambiguous.

$$\begin{array}{rcl}
\overline{(K)\ B} & = & \underline{K}[\psi(B)] \\
\overline{((B_1)\ B_2)\ K} & = & \underline{K}[(\psi(B_1))\ \psi(B_2)] \\
\overline{(T)\ K} & = & \underline{K}[\sigma(T)] \\
\overline{0} & = & 0 \\
\overline{\alpha.D} & = & \alpha.\overline{D} \\
\overline{D_1 + D_2} & = & \overline{D_1} + \overline{D_2}
\end{array}$$

$$\begin{array}{rcl}
\psi(x) & = & x \\
\psi(\lambda x\, S) & = & \lambda x\, \sigma(S)
\end{array}$$

$$\begin{array}{rcl}
\sigma(\lambda k\, C) & = & \overline{C} \\
\sigma(0) & = & 0 \\
\sigma(\alpha.T) & = & \alpha.\sigma(T) \\
\sigma(T_1 + T_2) & = & \sigma(T_1) + \sigma(T_2)
\end{array}$$

$$\begin{array}{rcl}
\underline{k}[M] & = & M \\
\lambda b\,((B)\ b)\ \underline{K}[M] & = & \underline{K}[(\psi(B))\ M] \\
\underline{\lambda b_1\,(T)\ \lambda b_2\,((b_1)\ b_2)\ \overline{K}}[M] & = & \underline{K}[(M)\ \sigma(T)]
\end{array}$$

To prove the completeness of the simulation we need several technical lemmas. The first two lemmas state that the translation defined above is in fact an inverse.

Lemma 3.18. For any term M, $\overline{(\llbracket M \rrbracket)\ k} = M$.

Proof. We have $\overline{(\llbracket M \rrbracket)\ k} = \underline{k}[\sigma(\llbracket M \rrbracket)] = \sigma(\llbracket M \rrbracket)$ so we have to show that $\sigma(\llbracket M \rrbracket) = M$ for all M. The proof follows by induction on the structure of M. The details can be found in Appendix A.8. □

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 $(\alpha.M)L : k = \alpha.((M)\ L) : k$ and $(M + N)L : k = (M)\ L + (N)\ L : K$, so the translation is not injective. However it is still true for values.

Lemma 3.19. For any value V, $\overline{V : k} = V$.

Proof. By induction on the structure of V. The details can be found in Appendix A.9. □

The last lemma (Lemma 3.20) states that the inverse translation preserves reductions.

Lemma 3.20. For any computation D, if $D \to_{a \cup \beta_n} D'$ then $\overline{D} \to^*_{\ell \cup \beta_v} \overline{D'}$. Also, if $D \to_a D'$ then $\overline{D} \to^*_\ell \overline{D'}$.

Proof. The proof of this lemma follows by induction on the reduction rules. It uses the following several intermediary results.

- The following equalities hold:
  (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) $\underline{K}[M][x := \psi(B)] = \underline{K[x := B]}[M[x := \psi(B)]]$
- For all terms M and continuations K_1 and K_2, $\underline{K_1}[\underline{K_2}[M]] = \underline{K_2[k := K_1]}[M]$.
- For all K and C, $\underline{K}[\overline{C}] = \overline{C[k := K]}$ (using the previous item)

- For any continuation K and term M, if $M \to_{\ell\cup\beta_v} M'$, then $\underline{K}[M] \to_{\ell\cup\beta_v} \underline{K}[M']$.
- The following relations hold (by induction on K, using the previous item)
  - $\underline{K}[M_1 + M_2] \to^*_\ell \underline{K}[M_1] + \underline{K}[M_2]$
  - $\underline{K}[\alpha.M] \to^*_\ell \alpha.\underline{K}[M]$
  - $\underline{K}[0] \to^*_\ell 0$
- For any suspension T, if $T \to_a T'$ then $\sigma(T) \to_\ell \sigma(T')$.

The details can be found in Appendix A.10. □

With these we can prove the completeness theorem.

Proof of Theorem 3.17. Using Lemma 3.20 for each step of the reduction, we get that $\overline{(\llbracket M \rrbracket)\ k} \to^*_{\ell\cup\beta_v} \overline{V : k}$. By Lemma 3.18 and Lemma 3.19, this implies $M \to^*_{\ell\cup\beta_v} V$. □

3.4. **Call-by-value simulates call-by-name.** To state that $\lambda^{\to}_{lin}$ simulates $\lambda^{\to}_{alg}$, we again use an algebraic extension of the continuation passing style encoding of [Plo75]. Let $\{\!|\cdot|\!\}$ be the following encoding of terms from $\lambda^{\to}_{alg}$ to $\lambda^{\to}_{lin}$ where f, g and h are fresh variables.

$$\begin{array}{rclcrcl}
\{\!|x|\!\} & = & x, & & \{\!|0|\!\} & = & \lambda f\,(0)\ f, \\
\{\!|\lambda x\, M|\!\} & = & \lambda f\,(f)\ \lambda x\,\{\!|M|\!\}, & & \{\!|(M)\ N|\!\} & = & \lambda f\,(\{\!|M|\!\})\ \lambda g\,((g)\ \{\!|N|\!\})\ f, \\
\{\!|\alpha.M|\!\} & = & \lambda f\,(\alpha.\{\!|M|\!\})\ f, & & \{\!|M + N|\!\} & = & \lambda f\,(\{\!|M|\!\} + \{\!|N|\!\})\ f.
\end{array}$$

This encoding satisfies two useful properties.

Lemma 3.21. For all terms M, the term $\{\!|M|\!\}$ is a base term. □

Lemma 3.22. $\{\!|M[x := N]|\!\} = \{\!|M|\!\}[x := \{\!|N|\!\}]$.

Proof. By structural induction on M. The details can be found in Appendix A.11. □

Let Φ be the encoding of abstractions defined by $\Phi(\lambda x\, M) = \lambda x\,\{\!|M|\!\}$. We keep the same notation ":" for the administrative infix operation capturing the behaviour of translated terms.

Definition 3.23. Let $(:)$ be the infix binary operation defined by:

$$\begin{array}{rcl}
0 : K & = & 0 \\
x : K & = & (x)\ K \\
\lambda\, x\, M : K & = & (K)\ \Phi(\lambda\, x\, M) \\
\alpha.M : K & = & \alpha.(M : K) \\
M + N : K & = & M : K + N : K
\end{array}
\qquad
\begin{array}{rcl}
(0)\ N : K & = & 0 \\
(x)\ N : K & = & x : \lambda\, f\,((f)\,\{\!|N|\!\})\,K \\
(\lambda\, x\, M)\ N : K & = & ((\Phi(\lambda\, x\, M))\ \{\!|N|\!\})\ K \\
(\alpha.M)\ N : K & = & \alpha.(M)\ N : K \\
(M + N)\ L : K & = & ((M)\ L + (N)\ L) : K \\
((M)\,N)\,L : K & = & (M)\,N : \lambda\, f\,((f)\,\{\!|L|\!\})\,K
\end{array}$$

Simulation theorems, similar to Theorems 3.7 and 3.9, can be stated as follows.

Theorem 3.24 (Simulation)**.** For any term M and variable k, if $M \to^*_{a\cup\beta_n} V$ where V is a value, then $(\{\!|M|\!\})\ k \to^*_{\ell\cup\beta_v} V : k$.

Theorem 3.25 (Simulation)**.** For any two fragments S_a of $\lambda^{=}_{alg}$ and S_ℓ of $\lambda^{=}_{lin}$ and variable k, such that $\forall M \in S_a$, $(\{\!|M|\!\})\ k \in S_\ell$, and for any term M in S_a, if $M \equiv\!\!\to^*_{a\cup\beta_n} V$ where V is a value, then $(\{\!|M|\!\})\ k \equiv\!\!\to^*_{\ell\cup\beta_v} V : k$.

A result similar to Corollary 3.11 can also be formulated. It is proven in a similar manner.

Corollary 3.26 (Indifference)**.**

(1) For any term M and variable k, if $M \to^*_{a\cup\beta_n} V$ where V is a value, then
$$(\{\!|M|\!\})\ k \to^*_{a\cup\beta_n} V : k$$
(2) For any fragment S of $\lambda^{=}_{alg}$ and variable k, such that $\forall M \in S$, $(\{\!|M|\!\})\ k \in S$, and for any term M in S, if $M \Rightarrow^*_{a\cup\beta_n} V$ where V is a value, then
$$(\{\!|M|\!\})\ k \Rightarrow^*_{a\cup\beta_n} V : k$$

Before moving to the description of the proofs of Theorems 3.24 and 3.25, let us consider an example.

Example 3.27. We illustrate Theorem 3.24 using the term $(\mathsf{copy})\ (B_1+B_2)$ of Example 3.8 which reduces to $\langle B_1, B_1\rangle + \langle B_2, B_2\rangle$ in $\lambda^{\to}_{lin}$ and to $\langle B_1+B_2, B_1+B_2\rangle$ in $\lambda^{\to}_{alg}$. The translation $\{\!|(\mathsf{copy})\ (B_1+B_2)|\!\}$ is the term $\lambda f\,(\{\!|\mathsf{copy}|\!\})\,\lambda g\,((g)\,\{\!|B_1+B_2|\!\})\,f$, where $\{\!|\mathsf{copy}|\!\}$ is $\lambda f\,(f)\,\lambda x\,\{\!|\langle x,x\rangle|\!\}$. $\{\!|\langle M,N\rangle|\!\}$ is $\lambda f\,(f)\,\Phi(\langle M,N\rangle)$, $\{\!|B_1+B_2|\!\}$ is $\lambda f\,(\{\!|B_1|\!\}+\{\!|B_2|\!\})\,f$ and $\{\!|B_1|\!\}$ is $\lambda g\,(g)\,\Phi(B_1)$

$$\begin{array}{rll}
(\{\!|(\mathsf{copy})\ (B_1+B_2)|\!\})\,k & \to_{\ell\cup\beta_v} & (\{\!|\mathsf{copy}|\!\})\,\lambda g\,((g)\,\{\!|B_1+B_2|\!\})\,k\\
& = & (\lambda f\,(f)\,\lambda x\,\{\!|\langle x,x\rangle|\!\})\,\lambda g\,((g)\,\{\!|B_1+B_2|\!\})\,k\\
& \to_{\ell\cup\beta_v} & (\lambda g\,((g)\,\{\!|B_1+B_2|\!\})\,k)\,\lambda x\,\{\!|\langle x,x\rangle|\!\}\\
& \to_{\ell\cup\beta_v} & ((\lambda x\,\{\!|\langle x,x\rangle|\!\})\,\{\!|B_1+B_2|\!\})\,k\\
\text{(Lemma 3.21)} & \to_{\ell\cup\beta_v} & (\{\!|\langle x,x\rangle|\!\}[x := \{\!|B_1+B_2|\!\}])\,k\\
\text{(Lemma 3.22)} & = & (\{\!|\langle B_1+B_2, B_1+B_2\rangle|\!\})\,k\\
& \to_{\ell\cup\beta_v} & (k)\,\Phi(\langle B_1+B_2, B_1+B_2\rangle)\\
& = & \langle B_1+B_2, B_1+B_2\rangle : k
\end{array}$$

In Section 3.2, the proofs of the simulations theorems were performed using two intermediate results, as follows (the term K is taken as a base term).

(1) Prove that $(\llbracket M\rrbracket)\,K \to^*_{a\cup\beta_n} M : K$;
(2) prove that if $M \to_{\ell\cup\beta_v} N$ then $M : K \to^*_{a\cup\beta_n} N : K$.

For the simulation theorems of the present section, we use a similar procedure. The three lemmas needed for the proof of the simulation theorems now read as follow.

Lemma 3.28. If K is a base term, then for every term M we have $(\{\!|M|\!\})\ K \to^*_{\ell\cup\beta_v} M : K$.

Proof. By structural induction on M. The details can be found in Appendix A.12. □

Lemma 3.29. If $M \to_{a\cup\beta_n} N$ then for every K base term we have $M : K \to^*_{\ell\cup\beta_v} N : K$.

Proof. Case by case on the rules of λ_{alg}. The details can be found in Appendix A.13. □

We are now ready to prove the simulation theorems. As advertised, these proofs reflect the exact same structures of the proofs of Theorems 3.7 and 3.9.

Proof of Theorem 3.24. From Lemmas 3.28 and 3.29, we have the following reduction. $(\{\!|M|\!\})\ k \to^*_{\ell\cup\beta_v} M : k \to^*_{\ell\cup\beta_v} V : k$. □

Proof of Theorem 3.25. From Lemma 3.28, $(\{\!|M|\!\})\ k \to^*_{\ell\cup\beta_v} M : k$, hence we also have $(\{\!|M|\!\})\ k \Rightarrow^*_{\ell\cup\beta_v} M : k$. A result equivalent to Corollary 3.16 can be shown as easily: if $M \Rightarrow_a N$ then for all base terms K, $M : K \Rightarrow^*_\ell N : K$. This entails that $M : k \Rightarrow^*_{\ell\cup\beta_v} V : k$. Note that since $(\{\!|M|\!\})\ k \in S_\ell$, $M : k$ is also in S_ℓ due to the closeness under $\Rightarrow_\ell$ of S_ℓ. The same argument applies to $M : k$, thus also to $V : k$. □

3.5. **Completeness of the call-by-name to call-by-value simulation.** We use the same procedure as in Section 3.3 to show that the translation is also complete.

Theorem 3.30 (Completeness)**.** For any term M and variable k, if $(\{\!|M|\!\})\ k \to^*_{\ell\cup\beta_v} V : k$ then $M \to^*_{a\cup\beta_n} V$.

The adjustments we have to make are the same as in the classical case: we treat variables and applications differently. Bellow is the grammar of the target language. It is closed under $\to_{\ell\cup\beta_v}$ reductions.

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

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 $((\lambda x\, S)\ S)\ K$. This is the main difference between the call-by-name and call-by-value CPS simulations. Apart from that, it satisfies the same properties.

We define the inverse translation using the following four functions.

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

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 3.31. For any term M, $\overline{(\{\!|M|\!\})\ k} = M$.

Proof. By induction on M. The details can be found in Appendix A.14. □

Lemma 3.32. For any value V, $\overline{V : k} = V$.

Proof. By induction on V. The details can be found in Appendix A.15. □

Lemma 3.33. For any computation D, if $D \to_{\ell\cup\beta_v} D'$ then $\overline{D} \to^*_{a\cup\beta_n} \overline{D'}$. Also, if $D \to_\ell D'$ then $\overline{D} \to^*_a \overline{D'}$.

Proof. The proof of this lemma is very similar to the one of Lemma 3.20. It follows by induction, involving several intermediary results similar to the ones in the proof of Lemma 3.20, where: ϕ is replaced with σ and σ with ψ in the four first equalities; then: $\to^*_\ell$ is replaced with $\to^*_a$ and $\to_\ell$ is replaced with $\to_a$. In other words:

- The following equalities hold.
  (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) $\underline{K}[M][x := \sigma(S)] = \underline{K[x := S]}[M[x := \sigma(S)]]$
- For all terms M and continuations K_1 and K_2, $\underline{K_1}[\underline{K_2}[M]] = \underline{K_2[k := K_1]}[M]$.
- For all K and C, $\underline{K}[\overline{C}] = \overline{C[k := K]}$.
- For any continuation K and term M, if $M \to_{a\cup\beta_n} M'$ then $\underline{K}[M] \to_{a\cup\beta_n} \underline{K}[M']$.
- For any continuation K, scalar α and terms M, M_1 and M_2, the following relations hold.
  – $\underline{K}[M_1 + M_2] \to_a^* \underline{K}[M_1] + \underline{K}[M_2]$
  – $\underline{K}[\alpha.M] \to_a^* \alpha.\underline{K}[M]$
  – $\underline{K}[0] \to_a^* 0$
- For any suspension T, if $T \to_\ell T'$ then $\sigma(T) \to_a \sigma(T')$.

The details can be found in Appendix A.16. □

Finally, we can prove the completeness theorem using the previous lemmas.

Proof of Theorem 3.30. Using Lemma 3.33 for each step of the reduction, we get that $\overline{(\{\!|M|\!\rangle\ k} \to_{a\cup\beta_n}^* \overline{V : k}$. By Lemma 3.31 and Lemma 3.32, this implies $M \to_{a\cup\beta_n}^* V$. □

3.6. **The remaining simulations.** In Figure 2, some arrows are missing, for example from $\lambda_{lin}^{\to}$ to $\lambda_{alg}^{=}$. We now show that the already existing arrows "compose" well. The first two simulations are $\lambda_{alg}^{\to} \to \lambda_{lin}^{=}$ and $\lambda_{lin}^{\to} \to \lambda_{alg}^{=}$ and do not require confluence.

Theorem 3.34. For any term M and variable k, if $M \to_{\ell\cup\beta_v}^* V$ (respectively $M \to_{a\cup\beta_n}^* V$) where V is a value, then $(\llbracket M \rrbracket)\ k \Rrightarrow_{a\cup\beta_n}^* V : k$ (respectively $(\{\!|M|\!\rangle\ k \Rrightarrow_{\ell\cup\beta_v}^* V : k$).

Proof. Given that $M \to_{\ell\cup\beta_v}^* V$, by Theorem 3.7, $(\llbracket M \rrbracket)\ k \to_{a\cup\beta_n}^* V : k$, which by Theorem 3.1 implies $(\llbracket M \rrbracket)\ k \Rrightarrow_{a\cup\beta_n}^* V : k$. Analogously, given that $M \to_{a\cup\beta_n}^* V$, by Theorem 3.24, we have $(\{\!|M|\!\rangle\ k \to_{\ell\cup\beta_v}^* V : k$, which by Theorem 3.2 implies that $(\{\!|M|\!\rangle\ k \Rrightarrow_{\ell\cup\beta_v}^* V : k$. □

The other two simulations are $\lambda_{alg}^{=} \to \lambda_{lin}^{\to}$ and $\lambda_{lin}^{=} \to \lambda_{alg}^{\to}$ and they do require confluence.

Theorem 3.35. For any term M in a confluent fragment of $\lambda_{lin}^{=}$ (respectively $\lambda_{alg}^{=}$) and variable k, if $M \Rrightarrow_{\ell\cup\beta_v}^* V$ (respectively $M \Rrightarrow_{a\cup\beta_n}^* V$) then we have $(\llbracket M \rrbracket)\ k \to_{a\cup\beta_n}^* V' : k$ with $V \Rrightarrow_\ell^* V'$ (respectively $(\{\!|M|\!\rangle\ k \to_{\ell\cup\beta_v}^* V' : k$ with $V \Rrightarrow_a^* V'$).

Proof. Given that $M \Rrightarrow_{\ell\cup\beta_v}^* V$ and that M is in a confluent fragment, Theorem 3.3 states that $M \to_{\ell\cup\beta_v}^* V'$ with $V \Rrightarrow_\ell^* V'$. In addition, Theorem 3.7 states that $(\llbracket M \rrbracket)\ k \to_{a\cup\beta_n}^* V' : k$. The other result is similar using Theorems 3.4 and 3.24. □

Finally, we show that we can also compose with the inverse translations to give the completeness of the remaining simulations. This time however, the requirements of confluence are reversed.

Theorem 3.36. For any term M and variable k, if $(\llbracket M \rrbracket)\ k \to^*_{a\cup\beta_n} V : k$ (respectively $(\{\!|M|\!\})\ k \to^*_{\ell\cup\beta_v} V : k$) then for any fragment S of $\lambda^{=}_{lin}$ (respectively $\lambda^{=}_{alg}$) such that $M \in S$, we have $M \Rrightarrow^*_{\ell\cup\beta_v} V'$ (respectively $M \Rrightarrow^*_{a\cup\beta_n} V$) in S.

Proof. If $(\llbracket M \rrbracket)\ k \to^*_{a\cup\beta_n} V : k$ then by Theorem 3.17 $M \to^*_{\ell\cup\beta_v} V$ which implies $M \Rrightarrow^*_{\ell\cup\beta_v} V$. Similarly if $(\{\!|M|\!\})k \to^*_{\ell\cup\beta_v} V : k$ then by Theorem 3.30 $M \to^*_{a\cup\beta_n} V$ which implies $M \Rrightarrow^*_{a\cup\beta_n} V$. □

Theorem 3.37. Let S be a confluent fragment of $\lambda^{=}_{alg}$ (respectively $\lambda^{=}_{lin}$). For any term M and variable k such that $(\llbracket M \rrbracket)\ k$ (respectively $(\{\!|M|\!\})\ k$) is in S, if $\llbracket M \rrbracket k \Rrightarrow^*_{a\cup\beta_n} V : k$ (respectively $(\{\!|M|\!\})\ k \Rrightarrow^*_{\ell\cup\beta_v} V : k$) then $M \to^*_{\ell\cup\beta_v} V'$ with $V \Rrightarrow^*_{\ell} V'$ (respectively $M \to^*_{a\cup\beta_n} V'$ with $V \Rrightarrow^*_{a} V'$).

Proof. If $(\llbracket M \rrbracket)\ k \Rrightarrow^*_{a\cup\beta_n} V : k$ then by Theorem 3.3 $(\llbracket M \rrbracket)\ k \to^*_{a\cup\beta_n} W$ with $V : k \Rrightarrow^*_{a} W$. By Lemma 3.20, we get $\overline{(\llbracket M \rrbracket)\ k} \to^*_{\ell\cup\beta_v} \overline{W}$ with $\overline{V : k} \Rrightarrow^*_{\ell} \overline{W}$, which by Lemmas 3.18 and 3.19 imply $M \to^*_{\ell\cup\beta_v} V' := \overline{W}$ with $V \Rrightarrow^*_{\ell} V'$.
Similarly, if $(\{\!|M|\!\})\ k \Rrightarrow^*_{\ell\cup\beta_v} V : k$ then by Theorem 3.4 $(\{\!|M|\!\})\ k \to^*_{\ell\cup\beta_v} W$ with $V : k \Rrightarrow^*_{\ell} W$. By Lemma 3.33, we get $\overline{(\{\!|M|\!\})\ k} \to^*_{a\cup\beta_n} \overline{W}$ with $\overline{V : k} \Rrightarrow^*_{a} \overline{W}$, which by Lemmas 3.31 and 3.32 imply $M \to^*_{a\cup\beta_n} V' := \overline{W}$ with $V \Rrightarrow^*_{a} V'$. □

4. Discussion and perspectives

4.1. **Simulating cloning.** As we discussed in the introduction, LINEAL is a language whose original purpose was to emulate quantum superpositions of states with linear combinations of terms. However, we saw in Example 3.27 that we can emulate the "cloning" operation $(\mathsf{copy})\ (B_1 + B_2) \to^*_{a\cup\beta_n} \langle B_1 + B_2, B_1 + B_2\rangle$ in LINEAL using a CPS encoding. How does this relate to the no-cloning theorem [WZ82] stating that a quantum state cannot be duplicated?

Since LINEAL is a higher-order language, a term both represents a quantum operator (i.e. a linear map) and a state of the system (i.e. a vector in the space of states). The choice of a call-by-value reduction strategy enforces this philosophy: an application $(M)\,(N + L)$ is really $(M)\,N + (M)\,L$, and there is no reduction under lambda's, making lambda-abstractions correspond to "pieces of code" to be executed only when applied. So the term $\lambda y\,(N + L)$ really stands for a piece of code that would input a base vector y and produce (possibly after some process) a superposition $N + L$. But in itself, the lambda-abstraction is a base vector – it is not a linear combination. If we were to use it, say as argument of $M = \lambda f\,\lambda x\,(f)\,(f)\,x$, it would actuallly get duplicated.

On the contrary, the term $\lambda y\,N + \lambda y\,L$ is the linear combination of two operators: one that inputs y and produces N, the other one that inputs y and produces L. Fed to the same term $M = \lambda f\,\lambda x\,(f)\,(f)\,x$, the distributivity of addition over application would take precedence: the term M behaves as a linear operator and $(M)\,(\lambda y\,N + \lambda y\,L)$ is really $(M)\,\lambda y\,N + (M)\,\lambda y\,L$.

We can see the same pattern in the term $(\mathsf{copy})\ (B_1 + B_2)$: the argument to the (linear) operator copy is a linear combination of B_1 and B_2, therefore in LINEAL the term $(\mathsf{copy})\ (B_1 + B_2)$ really corresponds to $\langle B_1, B_1\rangle + \langle B_2, B_2\rangle$ and not to $\langle B_1 + B_2, B_1 + B_2\rangle$. Along the CPS transformation from $\lambda^{\to}_{alg}$ to $\lambda^{\to}_{lin}$, recall from Section 3.4 that the argument $B_1 + B_2$ is transformed into $\lambda f\,(\{\!|B_1|\!\} + \{\!|B_2|\!\})\,f$. We are therefore not anymore in presence of a linear combination, but of a program that eventually produces a superposition. But this program is not a superposition: it is a base state that will be fed unchanged to an operator. In particular, if this operator is duplicating its argument, the code $\lambda f\,(\{\!|B_1|\!\} + \{\!|B_2|\!\})\,f$ will be duplicated. But instead of duplicating a superposition of terms, we are really duplicating the description of a program eventually producing a superposition.

4.2. **Conclusion.** In this paper we have shown the relation between two algebraic λ-calculi, ALG and LINEAL, via four canonical languages. These canonical algebraic λ-calculi account for all the different choices we can make between call-by-value versus call-by-name and algebraic reduction versus algebraic equality. We showed how each language can simulate the other, by taking care of marking where confluence was used.

This study opens the door to other questions. The calculus ALG admits finiteness spaces as a model [Ehr05, Ehr10]. What is the structure of the model of the linear algebraic λ-calculus induced by the continuation-passing style translation in finiteness spaces? The algebraic lambda-calculus can be equipped with a differential operator. What is the corresponding operator in call-by-value through the translation? The linear-algebraic lambda-calculus can encode quantum programs [ADCV13]. Can this translation help elucidate the relation between quantum computing and finiteness spaces?

Acknowledgements. We would like to thank Pablo Arrighi and Lionel Vaux for fruitful discussions and suggestions.

References

[AD04] Pablo Arrighi and Gilles Dowek. A computational definition of the notion of vectorial space. In *WRLA'04*, volume 117 of *ENTCS*, pages 249–261, 2004.

[AD08] Pablo Arrighi and Gilles Dowek. Linear-algebraic lambda-calculus: higher-order, encodings, and confluence. In *RTA'08*, volume 5117 of *LNCS*, pages 17–31, 2008.

[ADC11] Pablo Arrighi and Alejandro Díaz-Caro. Scalar System F for linear-algebraic λ-calculus: Towards a quantum physical logic. In *QPL'09*, volume 270/2 of *ENTCS*, pages 206–215, 2011.

[ADC12] Pablo Arrighi and Alejandro Díaz-Caro. A System F accounting for scalars. *Logical Methods in Computer Science*, 8(1:11), 2012.

[ADCV12] Pablo Arrighi, Alejandro Díaz-Caro, and Benoît Valiron. A type system for the vectorial aspects of the linear-algebraic lambda-calculus. In *DCM'11*, volume 88 of *EPTCS*, pages 1–15, 2012.

[ADCV13] Pablo Arrighi, Alejandro Díaz-Caro, and Benoît Valiron. The vectorial lambda-calculus. Available at `arXiv:1308.1138`, 2013.

[AP12] Ali Assaf and Simon Perdrix. Completeness of algebraic cps simulations. In *DCM'11*, volume 88 of *EPTCS*, pages 16–27, 2012.

[BCEM12] Antonio Bucciarelli, Alberto Carraro, Thomas Ehrhard, and Giulio Manzonetto. Full abstraction for the resource lambda calculus with tests, through taylor expansion. *LMCS*, 8(4), 2012.

[BDCJ12] Pablo Buiras, Alejandro Díaz-Caro, and Mauro Jaskelioff. Confluence via strong normalisation in an algebraic λ-calculus with rewriting. In *LSFA'11*, volume 81 of *EPTCS*, pages 16–29, 2012.

[BHP13] Pierre Boudes, Fanny He, and Michele Pagani. A characterization of the taylor expansion of lambda-terms. In *CSL'13*, pages 101–115, 2013.

[Bou93] G. Boudol. The lambda-calculus with multiplicities. Research Report RR-2025, INRIA, 1993.

[DCP12] Alejandro Díaz-Caro and Barbara Petit. Linearity in the non-deterministic call-by-value setting. In *WoLLIC'12*, volume 7456 of *LNCS*, pages 216–231, 2012.

[DCPTV10] Alejandro Díaz-Caro, Simon Perdrix, Christine Tasson, and Benoît Valiron. Equivalence of algebraic λ-calculi. In *Informal proceedings of HOR-2010*, pages 6–11, Edinburgh, UK, 2010.

[Ehr03] Thomas Ehrhard. On Köthe sequence spaces and linear logic. *Mathematical Structures in Computer Science*, 12(5):579–623, 2003.

[Ehr05] Thomas Ehrhard. Finiteness spaces. *Mathematical Structures in Computer Science*, 15(4):615–646, 2005.

[Ehr10] Thomas Ehrhard. A finiteness structure on resource terms. In *LICS'10*, pages 402–410. IEEE Computer Society, 2010.

[ER03] Thomas Ehrhard and Laurent Regnier. The differential lambda-calculus. *Theoretical Computer Science*, 309(1):1–41, 2003.

[Fis72] Michael J. Fischer. Lambda calculus schemata. *SIGPLAN Notice*, 7(1):104–109, 1972.

[Plo75] Gordon D. Plotkin. Call-by-name, call-by-value and the λ-calculus. *Theoretical Computer Science*, 1(2):125–159, 1975.

[Tas09] Christine Tasson. Algebraic totality, towards completeness. In *TLCA'09*, volume 5608 of *LNCS*, pages 325–340, 2009.

[Val10a] Benoît Valiron. Orthogonality and algebraic lambda-calculus. In *QPL'10*, pages 169–175, 2010. Available at `http://www.cs.ox.ac.uk/people/bob.coecke/QPL_proceedings.html`.

[Val10b] Benoît Valiron. Semantics of a typed algebraic lambda-calculus. In *DCM'10*, volume 26 of *EPTCS*, pages 147–158, 2010.

[Val13] Benoît Valiron. A typed, algebraic, computational lambda-calculus. *MSCS*, 23(2):504–554, 2013.

[Vau07] Lionel Vaux. On linear combinations of lambda-terms. In *RTA'07*, volume 4533 of *LNCS*, pages 374–388, 2007.

[Vau09] Lionel Vaux. The algebraic lambda calculus. *Mathematical Structures in Computer Science*, 19(5):1029–1059, 2009.

[vT04] André van Tonder. A lambda calculus for quantum computation. *SIAM Journal on Computing*, 33:1109–1135, 2004.

[WZ82] William K. Wootters and Wojciech H. Zurek. A single quantum cannot be cloned. *Nature*, 299:802–803, 1982.

Appendix A. Detailed proofs

A.1. **Proof of Theorem 2.2.**

Theorem 2.2. If M is closed and strongly normalising in LINEAL* and $M \to_{\ell \cup \beta_v} N$, then there exists L such that $M \to_{\mathcal{L}}^* L$ and $N \to_{\mathcal{L}}^* L$.

Proof. We proceed by induction on the $\to_{\ell \cup \beta_v}$ rewrite relation. The differences between LINEAL* rules and $\lambda_{lin}^{\to}$ rules are only in the conditions of rules A_l and A_r, the three first factorisation rules and the context rule $\xi_{\lambda_{lin}}$. Hence, if $M \to_{\mathcal{L}} N$, we can just take $L = N$. So, it suffices to consider only these different rules, when they do not coincide with those in LINEAL*.

(1) $(M + N)\ V \to_{\ell \cup \beta_v} (M)\ V + (N)\ V$, with $M + N$ not normal in LINEAL* (it is closed by assumption). Let L be the normal form in LINEAL* of $M + N$. Cases:
   - $L = M' + N'$ with $M \to_L^* M'$ and $N \to_L^* N'$. Then we have $(M + N)\ V \to_L^* (M' + N')\ V \to_{\mathcal{L}} (M')\ V + (N')\ V$, and also $(M)\ V + (N)\ V \to_L^* (M')\ V + (N')\ V$.
   - $L = M'$, with $M \to_L^* M'$ and $N \to_L^* 0$. Then we have $(M + N)\ V \to_L^* (M')\ V$, and also $(M)\ V + (N)\ V \to_L^* (M')\ V + (0)\ V \to_{\mathcal{L}} (M')\ V + 0 \to_{\mathcal{L}} (M')\ V$.
   - $L = (\alpha + \beta).L'$ with $M \to_L^* \alpha.L'$ and $N \to_L^* \beta.L'$. Then we have $(M + N)\ V \to_L^* ((\alpha+\beta).L')\ V \to_{\mathcal{L}} (\alpha+\beta).(L')\ V$, and also $(M)\ V+(N)\ V \to_L^* (\alpha.L')\ V+(\beta.L')\ V \to_L^* \alpha.(L')\ V + \beta.(L')\ V \to_{\mathcal{L}} (\alpha + \beta).(L')\ V$.
   - Cases $L = (\alpha + 1).L'$ with $M \to_L^* \alpha.L'$ and $N \to_L^* L'$, and $L = (1 + 1).L'$ with $M \to_L^* L'$ and $N \to_L^* L'$ are analogous to the previous case.

(2) $(B)\ (M + N) \to_{\ell \cup \beta_v} (B)\ M + (B)\ N$, with $M + N$ not normal in LINEAL*. This case is analogous to case 1.

(3) $(\alpha.M)\ V \to_{\ell \cup \beta_v} \alpha.(M)\ V$, with M not normal in LINEAL*. Let M' be the normal form in LINEAL* of M. Then $(\alpha.M)\ V \to_L^* (\alpha.M')\ V \to_{\mathcal{L}} \alpha.(M')\ V$ and $\alpha.(M)\ V \to_L^* \alpha.(M')\ V$.

(4) $(B)\ (\alpha.M) \to_{\ell \cup \beta_v} \alpha.(B)\ M$, with M not normal in LINEAL*. This case is analogous to case 3.

(5) $(0)\ V \to_{\ell \cup \beta_v} 0$. Notice that $(0)\ V \to_{\mathcal{L}} 0$.

(6) $(B)\ 0 \to_{\ell \cup \beta_v} 0$. Notice that $(B)\ 0 \to_{\mathcal{L}} 0$.

(7) $\alpha.M + \beta.M \to_{\ell \cup \beta_v} (\alpha + \beta).M$, when M is not normal. Let L be the normal form in LINEAL* of M. Then $\alpha.M + \beta.M \to_{\mathcal{L}}^* \alpha.L + \beta.L \to_{\mathcal{L}} (\alpha + \beta).L$, and $(\alpha + \beta).M \to_{\mathcal{L}}^* (\alpha + \beta).L$.

(8) $\alpha.M + M \to_{\ell \cup \beta_v} (\alpha + 1).M$ and $M + M \to_{\ell \cup \beta_v} (1 + 1).M$, when M is not normal. Analogous to case 7.

(9) $(V)\ M \to_{\ell \cup \beta_v} (V)\ M'$, with $M \to_{\ell \cup \beta_v} M'$. By the induction hypothesis, there exists L such that $M \to_L^* L$ and $M' \to_L^* L$. Hence we have $(V)\ M \to_L^* (V)\ L$ and also $(V)\ M' \to_L^* (V)\ L$. □

A.2. **Proof of Theorem 2.3.**

Theorem 2.3. If M is closed and strongly normalising in LINEAL* and $M \to_{\mathcal{L}} N$, then there exists L such that $M \to^*_{\ell\cup\beta_v} L$ and $N \to^*_{\ell\cup\beta_v} L$.

Proof. We only need to verify the seven differing rules between the two languages. Notice that the normal form of a closed term is a value V.

(1) $(V_1 + V_2)\ L \to_{\mathcal{L}} (V_1)\ L + (V_2)\ L$, with $V_1 + V_2$ closed normal. Let $L \to_{\ell\cup\beta_v} W$, then we have $(V_1 + V_2)\ L \to^*_{\ell\cup\beta_v} (V_1 + V_2)\ W \to_{\ell\cup\beta_v} (V_1)\ W + (V_2)\ W$, and also $(V_1)\ W + (V_2)\ W \to^*_{\ell\cup\beta_v} (V_1)\ W + (V_2)\ W$.

(2) $(L)\ (V_1 + V_2) \to_{\mathcal{L}} (L)\ V_1 + (L)\ V_2$. A value V can be 0 or a linear combination of base terms.
- Let 0 be the normal form of L. Hence, $(L)\ (V_1 + V_2) \to^*_{\ell\cup\beta_v} (0)\ (V_1 + V_2) \to_{\ell\cup\beta_v} 0$ and $(L)\ V_1 + (L)\ V_2 \to^*_{\ell\cup\beta_v} (0)\ V_1 + (0)\ V_2 \to_{\ell\cup\beta_v} 0 + 0 \to_{\ell\cup\beta_v} 0$
- Let $\sum_{i=1}^{n} \alpha_i.B_i$ be the normal form of L. Then $(L)\ (V_1 + V_2) \to^*_{\ell\cup\beta_v} (\sum_{i=1}^{n} \alpha_i.B_i)\ (V_1 + V_2) \to^*_{\ell\cup\beta_v} \sum_{i=1}^{n} \alpha_i.(B_i)\ (V_1 + V_2) \to^*_{\ell\cup\beta_v} \sum_{i=1}^{n} \alpha_i.((B_i)\ V_1 + (B_i)\ V_2) \to^*_{\ell\cup\beta_v} \sum_{i=1}^{n} \alpha_i.(B_i)\ V_1 + \sum_{i=1}^{n} \alpha_i.(B_i)\ V_2$, and $(L)\ V_1 + (L)\ V_2 \to^*_{\ell\cup\beta_v} (\sum_{i=1}^{n} \alpha_i.B_i)\ V_1 + (\sum_{i=1}^{n} \alpha_i.B_i)\ V_2 \to^*_{\ell\cup\beta_v} \sum_{i=1}^{n} \alpha_i.(B_i)\ V_1 + \sum_{i=1}^{n} \alpha_i.(B_i)\ V_2$.

(3) $(\alpha.V)\ M \to_{\mathcal{L}} \alpha.(V)\ M$. Let $M \to^*_{\ell\cup\beta_v} W$, then $(\alpha.V)\ W \to^*_{\ell\cup\beta_v} (\alpha.V)\ W \to_{\ell\cup\beta_v} \alpha.(V)\ W$, and also $\alpha.(V)\ W \to^*_{\ell\cup\beta_v} \alpha.(V)\ W$.

(4) $(M)\ (\alpha.V) \to_{\mathcal{L}} \alpha.(M)\ V$.
- Let 0 be the normal form of M. Hence, $(M)\ (\alpha.V) \to^*_{\ell\cup\beta_v} (0)\ (\alpha.V) \to_{\ell\cup\beta_v} 0$ and $\alpha.(M)\ V \to^*_{\ell\cup\beta_v} \alpha.(0)\ V \to_{\ell\cup\beta_v} \alpha.0 \to_{\ell\cup\beta_v} 0$.
- Let $\sum_{i=1}^{n} \alpha_i.B_i$ be the normal form of M. Then $(M)\ (\alpha.V) \to^*_{\ell\cup\beta_v} (\sum_{i=1}^{n} \alpha_i.B_i)\ (\alpha.V) \to^*_{\ell\cup\beta_v} \sum_{i=1}^{n} \alpha_i.(B_i)\ (\alpha.V) \to^*_{\ell\cup\beta_v} \sum_{i=1}^{n} \alpha_i\alpha(B_i)\ V$ and $\alpha.(M)\ V \to^*_{\ell\cup\beta_v} \alpha.(\sum_{i=1}^{n} \alpha_i.B_i)\ V \to^*_{\ell\cup\beta_v} \alpha.(\sum_{i=1}^{n} \alpha_i.(B_i)\ V) \to^*_{\ell\cup\beta_v} \sum_{i=1}^{n} \alpha_i\alpha(B_i)\ V$.

(5) $(0)\ M \to_{\mathcal{L}} 0$. Let V be the normal form of M. Then $(0)\ M \to^*_{\ell\cup\beta_v} (0)\ V \to_{\ell\cup\beta_v} 0$.

(6) $(M)\ 0 \to_{\mathcal{L}} 0$.
- Let 0 be the normal form of M, then $(M)\ 0 \to^*_{\ell\cup\beta_v} (0)\ 0 \to_{\ell\cup\beta_v} 0$.
- Let $\sum_{i=1}^{n} \alpha_i.(B_i)$ be the normal form of M. Then $(M)\ 0 \to^*_{\ell\cup\beta_v} (\sum_{i=1}^{n} \alpha_i.(B_i))\ 0 \to^*_{\ell\cup\beta_v} \sum_{i=1}^{n} \alpha_i.(B_i)\ 0 \to^*_{\ell\cup\beta_v} \sum_{i=1}^{n} \alpha_i.0 \to^*_{\ell\cup\beta_v} \sum_{i=1}^{n} 0 \to^*_{\ell\cup\beta_v} 0$.

(7) $(M)\ N \to_{\mathcal{L}} (M)\ N'$, with $N \to_{\mathcal{L}} N'$. Let V be the normal form of M. Then $(M)\ N \to^*_{\ell\cup\beta_v} (V)\ N \to_{\ell\cup\beta_v} (V)\ N'$ and $(M)\ N' \to^*_{\ell\cup\beta_v} (V)\ N'$. □

A.3. **Proof of Theorem 3.3.**

Theorem 3.3. For any term M in a confluent fragment of $\lambda^{\to}_{lin}$, if $M \equiv\!\!\to^*_{\ell\cup\beta_v} V$, then $M \to^*_{\ell\cup\beta_v} V'$, with $V \equiv\!\!\to^*_{\ell} V'$.

Proof. First note that a value can only reduce to another value. This follows by direct inspection of the rewriting rules. We proceed by induction on the length of the reduction.
- If $M \Rrightarrow^*_{\ell\cup\beta_v} M$, then choose $V' = M$ and note that $M \to^*_{\ell\cup\beta_v} M$.
- Assume the result true for $M \Rrightarrow^*_{\ell\cup\beta_v} V$: there is a value V' such that $M \to^*_{\ell\cup\beta_v} V'$ and $V \Rrightarrow^*_{\ell} V'$. Let $N \Rrightarrow_{\ell\cup\beta_v} M$. Case distinction:
  - $N \to_{\beta_v} M$, then $N \to_{\beta_v} M \to^*_{\ell\cup\beta_v} V'$ which implies $N \to^*_{\ell\cup\beta_v} V'$.
  - $N \Rrightarrow_{\ell} M$, then either $N \to_{\ell} M$, and then this case is analogous to the previous one, or $M \to_{\ell} N$. Due to the confluence of the subset, there exists a term L such that $N \to^*_{\ell\cup\beta_v} L$ and $V' \to^*_{\ell} L$, implying that L is a value, thus $V' \Rrightarrow^*_{\ell} L$. Then we have $V' \Rrightarrow^*_{\ell} L$ and $V \Rrightarrow^*_{\ell} V'$, so $L \Rrightarrow^*_{\ell} V$, closing the case. □

A.4. **Proof of Lemma 3.5.**

Lemma 3.5. $[\![M[x := B]]\!] = [\![M]\!][x := \Psi(B)]$ with B a base term.

Proof. Structural induction on M.
- $M = x$. Cases:
  - $B = y$. Then $M[x := B] = y$, and so $[\![M[x := B]]\!] = \lambda f\ (f)\ \ y = \lambda f\ (f)\ \ x[y/x]$ and this is equal to $[\![M]\!][x := \Psi(B)]$.
  - $B = \lambda y\, N$. Then $[\![M[x := B]]\!] = \lambda f\ (f)\ \ \lambda y\, [\![N]\!] = \lambda f\ (f)\ \ x[\lambda y\, [\![N]\!]/x] = [\![M]\!][x := \Psi(B)]$.
- $M = y$. Then $[\![M[x := B]]\!] = [\![M]\!][x := \Psi(B)] = [\![M]\!]$.
- $M = 0$. Analogous to previous case.
- $M = \lambda y\, N$. Then $[\![(\lambda y\, N)[x := B]]\!] = [\![\lambda y\, (N[x := B])]\!] = \lambda f\ (f)\ \ \lambda y\, [\![N[x := B]]\!]$, which by the induction hypothesis is $\lambda f\ (f)\ \ \lambda y\, [\![N]\!][x := \Psi(B)] = (\lambda f\ (f)\ \ \lambda y\, [\![N]\!])[x := \Psi(B)]$ $= [\![M]\!][x := \Psi(B)]$.
- $M = (N_1)\ N_2$. Then $[\![M[x := B]]\!] = [\![((N_1)\ N_2)[x := b]]\!] = [\![(N_1[x := B])\ N_2[x := B]]\!]$ $= \lambda f\, ([\![N_1[x := B]]\!])\ \lambda g\, ([\![N_2[x := B]]\!])\ \lambda h\, ((g)\ h)\ f$, which, by the induction hypothesis, is equal to $\lambda f\, ([\![N_1]\!][x := \Psi(B)])\ \lambda g\, ([\![N_2]\!][x := \Psi(B)])\ \lambda h\, ((g)\ h)\ f$, which can be rewritten as $\lambda f\, ([\![N_1]\!])\ \lambda g\, ([\![N_2]\!])\ \lambda h\, ((g)\ h)\ f[x := \Psi(B)] = [\![(N_1)\ N_2]\!][x := \Psi(B)] = [\![M]\!][x := \Psi(B)]$.
- $M = \alpha.N$. Then $[\![M[x := B]]\!] = [\![(\alpha.N)[x := B]]\!] = [\![\alpha.(N[x := B])]\!]$ which is equal to $\lambda f\, (\alpha.[\![N[x := B]]\!]\, f)$, and this, by the induction hypothesis is $\lambda f\, (\alpha.[\![N]\!][x := \Psi(B)]\, f)$ $= (\lambda f\, (\alpha.[\![N]\!]\, f))[x := \Psi(B)] = [\![\alpha.N]\!][x := \Psi(B)] = [\![M]\!][x := \Psi(B)]$.
- $M = N_1 + N_2$. Then $[\![M[x := B]]\!] = [\![(N_1 + N_2)[x := B]]\!] = [\![N_1[x := B] + N_2[x := B]]\!]$ $= \lambda f\, (([\![N_1[x := B]]\!] + [\![N_2[x := B]]\!])\, f)$, which, by the induction hypothesis, is equal to $\lambda f\, (([\![N_1]\!][x := \Psi(B)] + [\![N_2]\!][x := \Psi(B)])\, f) = (\lambda f\, (([\![N_1]\!] + [\![N_2]\!])\, f))[x := \Psi(B)] =$ $[\![N_1 + N_2]\!][x := \Psi(B)] = [\![M]\!][x := \Psi(B)]$. □

A.5. **Proof of Lemma 3.13.**

Lemma 3.13. If K is a base term, then for any term M, $([\![M]\!])\ K \to^*_{a\cup\beta_n} M : K$.

Proof. Structural induction on M.
- $M = \lambda x\, N$. Then $([\![\lambda x\, N]\!])\ K = (\lambda f\ (f)\ \ \lambda x\, [\![N]\!])\ K$ and by definition of Ψ this is equal to $(\lambda f\ (f)\ \ \Psi(\lambda x\, N))\ K \to_{a\cup\beta_n} (K)\ \Psi(\lambda x\, N) = \lambda x\, N : K$.
- $M = 0$. Then $([\![0]\!])\ K = (0)\ K \to_{a\cup\beta_n} 0 = 0 : K$.

- $M = M' + N$. Then $(\llbracket M' + N \rrbracket)\ K = (\lambda f\,(\llbracket M' \rrbracket + \llbracket N \rrbracket)\ f)\ K$ which $\to_{a\cup\beta_n}$-reduces to $(\llbracket M' \rrbracket + \llbracket N \rrbracket)\ K \to_{a\cup\beta_n} (\llbracket M' \rrbracket)\ K + (\llbracket N \rrbracket)\ K$ which $\to_{a\cup\beta_n}$-reduces by the induction hypothesis to $M' : K + N : K = M' + N : K$.
- $M = \alpha.N$. Then $(\llbracket \alpha.N \rrbracket)\ K = (\lambda f\,(\alpha.\llbracket N \rrbracket)\ f)\ K \to_{a\cup\beta_n} (\alpha.\llbracket N \rrbracket)\ K$ which $\to_{a\cup\beta_n}$-reduces to $\alpha.(\llbracket N \rrbracket)\ K)$ and this, by the induction hypothesis, $\to_{a\cup\beta_n}$-reduces to $\alpha.(N : K) = \alpha.N : K$.
- $M = (M')\ N$. Then $(\llbracket (M')\ N \rrbracket)\ K = (\lambda f\,(\llbracket M' \rrbracket)\ \lambda g\,(\llbracket N \rrbracket)\ \lambda h\,((g)\ h)\ f)\ K$ which $\to_{a\cup\beta_n}$-reduces to $(\llbracket M' \rrbracket)\ \lambda g\,(\llbracket N \rrbracket)\ \lambda h\,((g)\ h)\ K$. Since $\lambda g\,(\llbracket N \rrbracket)\ \lambda h\,((g)\ h)\ K$ is a value, by the induction hypothesis $(\llbracket M' \rrbracket)\ \lambda g\,(\llbracket N \rrbracket)\ \lambda h\,((g)\ h)\ K$ reduces to $M' : \lambda g\,(\llbracket N \rrbracket)\ \lambda h\,((g)\ h)\ K$. We do a second induction, over M', to prove that $M' : \lambda g\,(\llbracket N \rrbracket)\ \lambda h\,((g)\ h)\ K \to_{a\cup\beta_n} (M')\ N : K$.
  - If $M' = (M_1)\ M_2$, then $M' : \lambda g\,(\llbracket N \rrbracket)\ \lambda h\,((g)\ h)\ K = ((M_1)\ M_2)\ N : K = (M')\ N : K$.
  - If M' is a base term, then $M' : \lambda g\,(\llbracket N \rrbracket)\ \lambda h\,((g)\ h)\ K$ is by definition equal to the term $(\lambda g\,(\llbracket N \rrbracket)\ \lambda h\,((g)\ h)\ K)\ \Psi(M') \to_{a\cup\beta_n} (\llbracket N \rrbracket)\ \lambda h\,((\Psi(M'))\ h)\ K$ which by the main induction hypothesis $\to_{a\cup\beta_n}$-reduces to $N : \lambda h\,((\Psi(M'))\ h)\ K$, and this is equal to $(M')\ N : K$.
  - If $M' = \alpha.M_1$, then the term $M' : \lambda g\,(\llbracket N \rrbracket)\ \lambda h\,((g)\ h)\ K$ is equal to $\alpha.M_1 : \lambda g\,(\llbracket N \rrbracket)\ \lambda h\,((g)\ h)\ K = \alpha.(M_1 : \lambda g\,(\llbracket N \rrbracket)\ \lambda h\,((g)\ h)\ K)$ which by the second induction hypothesis $\to_{a\cup\beta_n}$-reduces to $\alpha.((M_1)\ N : K) = (\alpha.M_1)\ N : K = (M')\ N : K$.
  - If $M' = M_1 + M_2$, then $M' : \lambda g\,(\llbracket N \rrbracket)\ \lambda h\,((g)\ h)\ K$ is equal to the term $M_1 + M_2 : \lambda g\,(\llbracket N \rrbracket)\ \lambda h\,((g)\ h)\ K$ which is equal to $M_1 : \lambda g\,(\llbracket N \rrbracket)\ \lambda h\,((g)\ h)\ K + M_2 : \lambda g\,(\llbracket N \rrbracket)\ \lambda h\,((g)\ h)\ K$ which $\to_{a\cup\beta_n}$-reduces by the second induction hypothesis to $(M_1)\ N : K + (M_2)\ N : K = (M_1 + M_2)\ N : K = (M')\ N : K$.
  - If $M' = 0$ then $M : \lambda g\,(\llbracket N \rrbracket)\ \lambda h\,((g)\ h)\ K = 0 : \lambda g\,(\llbracket N \rrbracket)\ \lambda h\,((g)\ h)\ K = 0 = (0)\ N : K = (M')\ N : K$. □

A.6. **Proof of Lemma 3.14.**

Lemma 3.14. If $M \to_\ell N$ then for all K base term, $M : K \to_a^* N : K$.

Proof. Case by case on the rules $\to_\ell$.

Rules A_r:

- $(B)\ (M + N) \to_\ell (B)\ M + (B)\ N$, with B being a base term. Then $(B)\ (M + N) : K = M + N : \lambda f\,((\Psi(B))\ f)\ K = M : \lambda f\,((\Psi(B))\ f)\ K + N : \lambda f\,((\Psi(B))\ f)\ K = (B)\ M : K + (B)\ N : K = (B)\ M + (B)\ N : K$.
- $(B)\ \alpha.M \to_\ell \alpha.(B)\ M$, with B a base term. Then $(B)\ \alpha.M : K$ is equal to the term $\alpha.M : \lambda f\,((\Psi(B))\ f)\ K = \alpha.(M : \lambda f\,((\Psi(B))\ f)\ K) = \alpha.((B)\ M : K)$, which is $\alpha.(B)\ M : K$.
- $(B)\ 0 \to_\ell 0$, with B a base term. Then $(B)\ 0 : K = 0 : \lambda f\,((\Psi(B))\ f)\ K = 0 = 0 : K$.

Rules A_l:

- $(M + N)\ V \to_\ell (M)\ V + (N)\ V$, with V being a value. Then $(M + N)\ V : K = (M)\ V + (N)\ V : K$.
- $(\alpha.M)\ V \to_\ell \alpha.(M)\ V)$, with V being a value. Then $(\alpha.M)\ V : K = \alpha.(M)\ V : K$.
- $(0)\ V \to_\ell 0$, with V a value. Then $(0)\ V : K = 0 = 0 : K$.

Rules F and S:

- $\alpha.(M + N) \to_\ell \alpha.M + \alpha.N$. Then $\alpha.(M + N) : K = \alpha.(M : K + N : K) \to_a \alpha.(M : K) + \alpha.(N : K) = \alpha.M + \alpha.N : K$.

- $\alpha.M + \beta.M \to_\ell (\alpha+\beta).M$. Then $\alpha.M + \beta.M : K = \alpha.(M : K) + \beta.(M : K) \to_a (\alpha+\beta).(M : K) = (\alpha+\beta).M : K$.
- $\alpha.M + M \to_\ell (\alpha+1).M$. Then $\alpha.M + M : K = \alpha.M : K + M : K = \alpha.(M : K) + M : K \to_a (\alpha+1).(M : K) = (\alpha+1).M : K$.
- $M + M \to_\ell (1+1).M$. Then $M + M : K = M : K + M : K \to_a (1+1).(M : K) = (1+1).M : K$.
- $0 + M \to_\ell M$. Then $0 + M : K = (0 : K) + (M : K) = 0 + (M : K) \to_a M : K$.
- $\alpha.(\beta.M) \to_\ell (\alpha\beta).M$. Then $\alpha.(\beta.M) : K = \alpha.(\beta.M : K) = \alpha.(\beta.(M : K))$ which $\to_a$-reduces to $(\alpha\beta).(M : K) = (\alpha\beta).M : K$.
- $1.M \to_\ell M$. Then $1.M : K = 1.(M : K) \to_a M : K$.
- $0.M \to_\ell 0$. Then $0.M : K = 0.(M : K) \to_a 0 = 0 : K$.
- $\alpha.0 \to_\ell 0$. Then $\alpha.0 : K = \alpha.(0 : K) = \alpha.0 \to_a 0 = 0 : K$.

Rules *Asso* **and** *Com***:**

- $M + (N + L) \to_\ell (M + N) + L$. Then $M + (N + L) : K = M : K + (N + L : K) = M : K + (N : K + L : K) \to_a (M : K + N : K) + L : K = M + N : K + L : K = (M + N) + L : K$.
- $M + N \to_\ell N + M$. Then $M + N : K = M : K + N : K \to_a N : K + M : K = N + M : K$.

Rules ξ **and** $\xi_{\lambda_{lin}}$**:** Assume $M \to_\ell M'$, and assume that for all K base term, $M : K \to_a^* M' : K$. We show that the result also holds for each contextual rule.

- $M + N \to_\ell M' + N$. Then $M + N : K = M : K + N : K \to_a^* M' : K + N : K = M' + N : K$.
- $N + M \to_\ell N + M'$, analogous to previous case.
- $\alpha.M \to_\ell \alpha.M'$. Then $\alpha.M : K = \alpha.(M : K) \to_a^* \alpha.(M' : K) = \alpha.M' : K$.
- $(V)\ M \to_\ell (V)\ M'$. Case by case:
  - $V = B$. Then $(B)\ M : K = M : \lambda f\,((\Psi(B))\ f)\ K$ which $\to_a$-reduces by the induction hypothesis to $M' : \lambda f\,((\Psi(B))\ f)\ K = (B)\ M' : K$.
  - $V = 0$. Then $(0)\ M : K = 0 = (0)\ M' : K$.
  - $V = \alpha.W$. Then $(\alpha.W)\ M : K = \alpha.(W)\ M : K = \alpha.((W)\ M : K)$ which $\to_a$-reduces by the induction hypothesis to $\alpha.((W)\ M' : K) = \alpha.(W)\ M' : K = (\alpha.W)\ M' : K$.
  - $V = V_1 + V_2$. Then $(V_1+V_2)\ M : K = (V_1)\ M + (V_2)\ M : K = (V_1)\ M : K + (V_2)\ M : K$ which $\to_a$-reduces by the induction hypothesis to $(V_1)\ M' : K + (V_2)\ M' : K = (V_1)\ M' + (V_2)\ M' : K = (V_1 + V_2)\ M' : K$.
- $(M)\ N \to_\ell (M')\ N$ Case by case:
  - $M = B$. Absurd since a base term cannot reduce.
  - $M = \alpha.M_1$. Case by case on the possible $\to_\ell$-reductions of M:
    - $M' = \alpha.M_1'$ with $M_1 \to_\ell M_1'$. Then $(\alpha.M_1)\ N : K = \alpha.(M_1)\ N : K = \alpha.((M_1)\ N : K)$ which by the induction hypothesis $\to_a$-reduces to $\alpha.((M_1')\ N : K) = \alpha.(M_1')\ N : K = (\alpha.M_1')\ N : K$.
    - $M = \alpha.(\beta.M_3)$ and $M' = (\alpha\beta).M_3$. Then $(\alpha.(\beta.M_3))\ N : K = \alpha.(\beta.((M_3)\ N : K)) \to_a (\alpha\beta).((M_3)\ N : K) = ((\alpha\beta).M_3)\ N : K$.
    - $M = \alpha.(L_1 + L_2)$ and $M' = \alpha.L_1 + \alpha.L_2$. Then $(\alpha.(L_1 + L2))\ N : K = \alpha.((L_1)\ N : K + (L_2)\ N : K) \to_a \alpha.((L_1)\ N : K) + \alpha.((L_2)\ N : K) = (\alpha.L_1 + \alpha.L2)\ N : K$.
    - $\alpha = 1$ and $M' = M_1$. Then $(1.M_1)\ N : K = 1.((M_1)\ N : K)$ which $\to_a$-reduces to $(M_1)\ N : K$.
    - $\alpha = 0$ and $M' = 0$. Then $(0.M_1)\ N : K = 0.((M_1)\ N : K) \to_a 0 = (0)\ N : K$.

  - $M_1 = 0$ and $M' = 0$. Then $(\alpha.0)\ N : K = \alpha.((0)\ N : K) = \alpha.0 \to_a 0 = (0)\ N : K$.
- $M = M_1 + M_2$. Case by case on the possible $\to_\ell$-reductions of M:
  - $M' = M_1' + M_2$ with $M_1 \to_\ell M_1'$. Then $(M_1 + M_2)\ N : K = (M_1)\ N : K + (M_2)\ N : K$ which by the induction hypothesis $\to_a$-reduces to $(M_1')\ N : K + (M_2)\ N : K = (M_1' + M_2)\ N : K$.
  - $M' = M_1 + M_2'$ with $M_2 \to_\ell M_2'$. Analogous to previous case.
  - $M_2 = L_1 + L_2$ and $M' = (M_1 + L_1) + L_2$. Then $(M_1 + (L_1 + L_2))\ N : K = (M_1)\ N : K + ((L_1)\ N : K + (L_2)\ N : K)$ and this $\to_a$-reduces to $((M_1)\ N : K + (L_1)\ N : K) + (L_2)\ N : K = ((M_1 + L_1) + L_2)\ N : K$.
  - $M_1 = L_1 + L_2$ and $M' = L_1 + (L_2 + M_2)$. Analogous to previous case.
  - $M' = M_2 + M_1$. Then $(M_1 + M_2)\ N : K = (M_1)\ N : K + (M_2)\ N : K \to_a (M_2)\ N : K + (M_1)\ N : K = (M_2 + M_1)\ N : K$.
  - $M_1 = \alpha.M_3$, $M_2 = \beta.M_3$ and $M' = (\alpha + \beta).M_3$. Then $(\alpha.M_3 + \beta.M_3)\ N : K = \alpha.((M_3)\ N : K) + \beta.((M_3)\ N : K)$ which $\to_a$-reduces to $(\alpha+\beta).((M_3)\ N : K) = ((\alpha + \beta).M_3)\ N : K$.
  - $M_1 = \alpha.M_3$, $M_2 = M_3$ and $M' = (\alpha + 1).M_3$. Analogous to previous case.
  - $M_1 = M_2$ and $M' = (1 + 1).M_1$. Analogous to previous case.
- $M = 0$. Absurd since 0 does not reduce.
- $M = (M_1)\ M_2$. Then the term $((M_1)\ M_2)\ N : K$ is equal to $(M_1)\ M_2 : \lambda g\,([\![N]\!])\ \lambda h\,((g)\ h)\ K$, which by the induction hypothesis $\to_a$-reduces to $M' : \lambda g\,([\![N]\!])\ \lambda h\,((g)\ h)\ K$. We do a second induction, over M', to prove that $M' : \lambda g\,([\![N]\!])\ \lambda h\,((g)\ h)\ K \to_a (M')\ N : K$.
  - If $M' = (M_1')\ M_2'$, then $M' : \lambda g\,([\![N]\!])\ \lambda h\,((g)\ h)\ K$ is equal to $((M_1')\ M_2')\ N : K = (M')\ N : K$.
  - M' cannot be a base term since from $(M_1)\ M_2$ it is not possible to arrive to a base term using only $\to_\ell$.
  - If $M' = \alpha.M_1'$, then $M' : \lambda g\,([\![N]\!])\ \lambda h\,((g)\ h)\ K$ is equal to the term $\alpha.M_1' : \lambda g\,([\![N]\!])\ \lambda h\,((g)\ h)\ K = \alpha.(M_1' : \lambda g\,([\![N]\!])\ \lambda h\,((g)\ h)\ K)$ which $\to_a$-reduces by the induction hypothesis to $\alpha.((M_1')\ N : K) = (\alpha.M_1')\ N : K = (M')\ N : K$.
  - If $M' = M_1' + M_2'$, then the term $M' : \lambda g\,([\![N]\!])\ \lambda h\,((g)\ h)\ K$ is equal to $M_1' + M_2' : \lambda g\,([\![N]\!])\ \lambda h\,((g)\ h)\ K$ which is equal to $M_1' : \lambda g\,([\![N]\!])\ \lambda h\,((g)\ h)\ K + M_2' : \lambda g\,([\![N]\!])\ \lambda h\,((g)\ h)\ K$ which $\to_a$-reduces by the induction hypothesis to $(M_1')\ N : K + (M_2')\ N : K = (M_1' + M_2')\ N : K = (M')\ N : K$.
  - If $M' = 0$ then $M' : \lambda g\,([\![N]\!])\ \lambda h\,((g)\ h)\ K$ is equal $0 : \lambda g\,([\![N]\!])\ \lambda h\,((g)\ h)\ K$ and this to $0 = (0)\ N : K = (M')\ N : K$. □

A.7. **Proof of Lemma 3.15.**

Lemma 3.15. If $M \to_{\ell\cup\beta_v} N$ then for all K base term, $M : K \to^*_{a\cup\beta_n} N : K$.

Proof. Case by case on the rules $\to_{\ell\cup\beta_v}$.

Rule β_v: $(\lambda x\ M)\ B : K = B : \lambda f\,((\Psi(\lambda x\ M))\ f)\ K = (\lambda f\,((\Psi(\lambda x\ M))\ f)\ K)\ \Psi(B) \to_{\beta_n} ((\Psi(\lambda x\,M))\,\Psi(B))\,K = ((\lambda x\,[\![M]\!])\,\Psi(B))\,K \to_{\beta_n} [\![M]\!][x := \Psi(B)]\,K$, which, by Lemma 3.5, is equal to $[\![M[x := B]]\!]\,K$, and this, by Lemma 3.13, $\to^*_{a\cup\beta_n}$-reduces to $M[x := B] : K$.

Algebraic rules: If $M \to_\ell N$, then by Lemma 3.14 $M : K \to_a^* N : K$ which implies that $M : K \to_{a\cup\beta_n}^* N : K$.

Rules ξ and $\xi_{\lambda_{lin}}$: If $M \to_\ell M'$, then we use Lemma 3.14 to close the case. Assume $M \to_{\beta_v} M'$, and assume that for all K base term, $M : K \to_{a\cup\beta_n}^* M' : K$. We show that the result also holds for each contextual rule.

- $M + N \to_{\beta_v} M' + N$. Then $M + N : K = M : K + N : K \to_{a\cup\beta_n}^* M' : K + N : K = M' + N : K$.
- $N + M \to_{\beta_v} N + M'$, analogous to previous case.
- $\alpha.M \to_{\beta_v} \alpha.M'$. Then $\alpha.M : K = \alpha.(M : K) \to_{a\cup\beta_n}^* \alpha.(M' : K) = \alpha.M' : K$.
- $(V)\ M \to_{\beta_v} (V)\ M'$. Case by case:
  - $V = B$. Then $(B)\ M : K = M : \lambda f\,((\Psi(B))\ f)\ K$ which $\to_{a\cup\beta_n}$-reduces by the induction hypothesis to $M' : \lambda f\,((\Psi(B))\ f)\ K = (B)\ M' : K$.
  - $V = 0$. Then $(0)\ M : K = 0 = (0)\ M' : K$.
  - $V = \alpha.W$. Then $(\alpha.W)\ M : K = \alpha.(W)\ M : K = \alpha.((W)\ M : K)$ which $\to_{a\cup\beta_n}$-reduces by the induction hypothesis to $\alpha.((W)\ M' : K) = \alpha.(W)\ M' : K = (\alpha.W)\ M' : K$.
  - $V = V_1 + V_2$. Then $(V_1 + V_2)\ M : K = (V_1)\ M + (V_2)\ M : K = (V_1)\ M : K + (V_2)\ M : K$ which $\to_{a\cup\beta_n}$-reduces by the induction hypothesis to $(V_1)\ M' : K + (V_2)\ M' : K = (V_1)\ M' + (V_2)\ M' : K = (V_1 + V_2)\ M' : K$.
- $(M)\ N \to_{\beta_v} (M')\ N$ Case by case:
  - $M = B$. Absurd since a base term cannot reduce.
  - $M = \alpha.M_1$. The only possible $\to_{\beta_v}$-reduction from M is $M' = \alpha.M_1'$ with $M_1 \to_{\beta_v} M_1'$. Then $(\alpha.M_1)\ N : K = \alpha.(M_1)\ N : K = \alpha.((M_1)\ N : K)$ which by the induction hypothesis $\to_{a\cup\beta_n}$-reduces to $\alpha.((M_1')\ N : K) = \alpha.(M_1')\ N : K = (\alpha.M_1')\ N : K$.
  - $M = M_1 + M_2$. Case by case on the possible $\to_{\beta_v}$-reductions of M:
    - $M' = M_1' + M_2$ with $M_1 \to_{\beta_v} M_1'$. Then $(M_1 + M_2)\ N : K = (M_1)\ N : K + (M_2)\ N : K$ which by the induction hypothesis $\to_{a\cup\beta_n}$-reduces to $(M_1')\ N : K + (M_2)\ N : K = (M_1' + M_2)\ N : K$.
    - $M' = M_1 + M_2'$ with $M_2 \to_{\beta_v} M_2'$. Analogous to previous case.
  - $M = 0$. Absurd since 0 does not reduce.
  - $M = (M_1)\ M_2$. Then the term $((M_1)\ M_2)\ N : K$ is equal to $(M_1)\ M_2 : \lambda g\,([\![N]\!])\ \lambda h\,((g)\ h)\ K$, which $\to_{a\cup\beta_n}$-reduces, by the induction hypothesis, to $M' : \lambda g\,([\![N]\!])\ \lambda h\,((g)\ h)\ K$. We do a second induction, over M', to prove that $M' : \lambda g\,([\![N]\!])\ \lambda h\,((g)\ h)\ K \to_{a\cup\beta_n}$-reduces to $(M')\ N : K$.
    - If $M' = (M_1')\ M_2'$, then $M' : \lambda g\,([\![N]\!])\ \lambda h\,((g)\ h)\ K$ is equal to $((M_1')\ M_2')\ N : K = (M')\ N : K$.
    - If M' is a base term, then the term $M' : \lambda g\,([\![N]\!])\ \lambda h\,((g)\ h)\ K$ is equal to $(\lambda g\,([\![N]\!])\ \lambda h\,((g)\ h)\ K)\ \Psi(M')$ which $\to_{a\cup\beta_n}$-reduces to $([\![N]\!])\ \lambda h\,((\Psi(M'))\ h)\ K$ which, by Lemma 3.13, $\to_{a\cup\beta_n}$-reduces $N : \lambda h\,((\Psi(M'))\ h)\ K = (M')\ N : K$.
    - If $M' = \alpha.M_1'$, then $M' : \lambda g\,([\![N]\!])\ \lambda h\,((g)\ h)\ K$ is equal to the term $\alpha.M_1' : \lambda g\,([\![N]\!])\ \lambda h\,((g)\ h)\ K = \alpha.(M_1' : \lambda g\,([\![N]\!])\ \lambda h\,((g)\ h)\ K)$ which $\to_{a\cup\beta_n}$-reduces by the induction hypothesis to $\alpha.((M_1')\ N : K) = (\alpha.M_1')\ N : K = (M')\ N : K$.
    - If $M' = M_1' + M_2'$, then the term $M' : \lambda g\,([\![N]\!])\ \lambda h\,((g)\ h)\ K$ is equal to $M_1' + M_2' : \lambda g\,([\![N]\!])\ \lambda h\,((g)\ h)\ K$ which is equal to $M_1' : \lambda g\,([\![N]\!])\ \lambda h\,((g)\ h)\ K +$

$M'_2 : \lambda g\,([\![N]\!])\ \lambda h\,((g)\ h)\ K$ which $\to_{a\cup\beta_n}$-reduces by the induction hypothesis to $(M'_1)\ N : K + (M'_2)\ N : K = (M'_1 + M'_2)\ N : K = (M')\ N : K$.

∗ If $M' = 0$ then $M' : \lambda g\,([\![N]\!])\ \lambda h\,((g)\ h)\ K$ is equal to $0 : \lambda g\,([\![N]\!])\ \lambda h\,((g)\ h)\ K = 0 = (0)\ N : K = (M')\ N : K$. □

A.8. **Proof of Lemma 3.18.**

Lemma 3.18. For any term M, $\overline{([\![M]\!])\ k} = M$.

Proof. By induction on M.

- Case x. Then $\sigma([\![x]\!]) = \sigma(\lambda k\,(k)\ x) = \overline{(k)\ x} = \underline{k}[\psi(x)] = \psi(x) = x$.
- Case $\lambda x\,M$. Then $\sigma([\![\lambda x\,M]\!]) = \sigma(\lambda k\,(k)\ \lambda x\,[\![M]\!]) = \overline{k(\lambda x\,[\![M]\!])} = \underline{k}[\psi(\lambda x\,[\![M]\!])] = \psi(\lambda x\,[\![M]\!]) = \lambda x\,\sigma([\![M]\!])$, which, by the induction hypothesis, is equal to $= \lambda x\,M$.
- Case MN. Then $\sigma([\![(M)\ N]\!]) = \sigma(\lambda k\,([\![M]\!])\ \lambda b_1\,([\![N]\!])\ \lambda b_2\,((b_1)\ b_2)\ k)$ which is equal to $\overline{([\![M]\!])\ \lambda b_1\,([\![N]\!])\ \lambda b_2\,((b_1)\ b_2)\ k} = \underline{\lambda b_1\,([\![N]\!])\ \lambda b_2\,((b_1)\ b_2)\ k}[\sigma([\![M]\!])]$ which is equal to $\underline{k}[(\sigma([\![M]\!]))\ \sigma([\![N]\!])] = (\sigma([\![M]\!]))\ \sigma([\![N]\!])$, which, by the induction hypothesis, is equal to $= (M)\ N$.
- Case 0. Then $\sigma([\![0]\!]) = \sigma(0) = 0$
- Case $\alpha.M$. Then $\sigma([\![\alpha.M]\!]) = \sigma(\lambda k\,(\alpha.[\![M]\!])\ k) = \overline{(\alpha.[\![M]\!])\ k} = \underline{k}[\sigma(\alpha.[\![M]\!])] = \sigma(\alpha.[\![M]\!]) = \alpha.\sigma([\![M]\!])$, which, by the induction hypothesis, is equal to $= \alpha.M$.
- Case $M + N$. Then $\sigma([\![M + N]\!]) = \sigma(\lambda k\,([\![M]\!] + [\![N]\!])\ k) = \overline{([\![M]\!] + [\![N]\!])\ k} = \underline{k}[\sigma([\![M]\!] + [\![N]\!])] = \sigma([\![M]\!] + [\![N]\!]) = \sigma([\![M]\!]) + \sigma([\![N]\!])$, which, by the induction hypothesis, is equal to $= M + N$. □

A.9. **Proof of Lemma 3.19.**

Lemma 3.19. For any value V, $\overline{V : k} = V$.

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 3.18.
- Case $\alpha.V$. Then $\overline{\alpha.V : k} = \overline{\alpha.(V : k)} = \alpha.\overline{V : k} = \alpha V$ by the induction hypothesis.
- Case $U + V$. Then $\overline{U + V : k} = \overline{U : k + V : k} = \overline{U : k} + \overline{V : k} = U + V$ by the induction hypothesis. □

A.10. **Proof of Lemma 3.20.**

Lemma 3.20. For any computation D, if $D \to_{a\cup\beta_n} D'$ then $\overline{D} \to^*_{\ell\cup\beta_v} \overline{D'}$. Also, if $D \to_a D'$ then $\overline{D} \to^*_\ell \overline{D'}$.

To prove this lemma, we need several intermediary results.

Lemma A.1. The following equalities hold.

(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. We prove simultaneously the four properties by induction on the structure of 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 $\psi(\lambda x\, S)[x := \psi(B)] = (\lambda y\, \sigma(S))[x := \psi(B)]$, which, by the induction hypothesis, is equal to $\lambda y\, \sigma(S[x := B]) = \psi((\lambda y\, S)[x := B])$.

(2) Cases for T.
- Case $\lambda k\, C$. Then $\sigma(\lambda k\, C)[x := \psi(B)] = \overline{C}[x := \psi(B)]$, which, by the induction hypothesis, is equal to $\overline{C[x := B]} = \sigma((\lambda k\, C)[x := B])$.
- Case 0. Then $\sigma(0)[x := \psi(B)] = 0 = \sigma(0[x := B])$.
- Case $\alpha.T$. Then $\sigma(\alpha.T)[x := \psi(B)] = (\alpha.\sigma(T))[x := \psi(B)]$, which, by the induction hypothesis, is equal to $\alpha.\sigma(T[x := B]) = \sigma((\alpha.T)[x := B])$.
- Case $T_1 + T_2$. Then $\sigma(T_1 + T_2)[x := \psi(B)] = (\sigma(T_1) + \sigma(T_2))[x := \psi(B)]$, which, by the induction hypothesis, is equal to $\sigma(T_1[x := B]) + \sigma(T_2[x := B]) = \sigma((T_1 + T_2)[x := B])$.

(3) Cases for C.
- Case $(K)\ B_1$. Then $\overline{(K)\ B_1}[x := \psi(B)] = \underline{K}[\psi(B_1)][x := \psi(B)]$, which, by the induction hypothesis, is equal to $\underline{K[x := B]}[\psi(B_1)[x := \psi(B)]]$, which, by the induction hypothesis, is $\underline{K[x := B]}[\psi(B_1[x := B])] = \overline{((K)\ B_1)[x := B]}$.
- Case $((B_1)\ B_2)\ K$. Then $\overline{((B_1)\ B_2)\ K}[x := \psi(B)] = \underline{K}[(\psi(B_1))\ \psi(B_2)][x := \psi(B)]$, which, by the induction hypothesis, is $\underline{K[x := B]}[((\psi(B_1))\ \psi(B_2))[x := \psi(B)]]$, which, by the induction hypothesis, is equal to $\underline{K[x := B]}[(\psi(B_1[x := B]))\ \psi(B_2[x := B])] = \overline{(((B_1)\ B_2)\ K)[x := B]}$.
- Case $(T)\ K$. Then $\overline{(T)\ K}[x := \psi(B)] = \underline{K}[\sigma(T)][x := \psi(B)]$, which, by the induction hypothesis, is equal to $\underline{K[x := B]}[\sigma(T)[x := \psi(B)]]$, which, by the induction hypothesis, is equal to $\underline{K[x := B]}[\sigma(T[x := B])] = \overline{((T)\ K)[x := B]}$.

(4) Cases for K.
- Case k. Then $\underline{k}[M][x := \psi(B)] = M[x := \psi(B)] = k[x := B][M[x := \psi(B)]]$.
- Case $\lambda b\, B_1 b K$. Then $\underline{\lambda b\, ((B_1)\ b)\ K}[M][x := \psi(B)] = \underline{K}[(\psi(B_1))\ M][x := \psi(B)]$, which, by the induction hypothesis, is $\underline{K[x := B]}[((\psi(B_1))\ M)[x := \psi(B)]]$, which, by the induction hypothesis, is equal to $\underline{K[x := B]}[(\psi(B_1[x := B]))\ M[x := \psi(B)]] = \underline{(\lambda b\, ((B_1)\ b)\ K)[x := B]}[M[x := \psi(B)]]$.
- Case $\lambda b_1\, (T)\ \lambda b_2\, ((b_1)\ b_2)\ K$. Then $\underline{\lambda b_1\, (T)\ \lambda b_2\, ((b_1)\ b_2)\ K}[M][x := \psi(B)]$ is the term $\underline{K}[(M)\ \sigma(T)][x := \psi(B)]$, which, by the induction hypothesis, is equal to $\underline{K[x := B]}[((M)\ \sigma(T))[x := \psi(B)]]$, which, by the induction hypothesis, is equal to $\underline{K[x := B]}[(M[x := \psi(B)])\ \sigma(T[x := B])]$ and this is finally equal to the term $\underline{(\lambda b_1\, (T)\ \lambda b_2\, ((b_1)\ b_2)\ K)[x := B]}[M[x := \psi(B)]]$. □

Lemma A.2. 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.
- Case k. Then $\underline{K_1}[\underline{k}[M]] = \underline{K_1}[M] = \underline{k[k := K_1]}[M]$.
- Case $\lambda b\, ((B)\ b)\ K$. Then $\underline{K_1}[\underline{\lambda b\, ((B)\ b)\ K}[M]] = \underline{K_1}[\underline{K}[(\psi(B))\ M]]$, which, by the induction hypothesis, is $\underline{K[k := K_1]}[(\psi(B))\ M] = \underline{(\lambda b\, ((B)\ b)\ K)[k := K_1]}[M]$.

- Case $\lambda b_1\,(T)\ \lambda b_2\,((b_1)\ b_2)\ K$. Then $\underline{K_1}[\underline{\lambda b_1\,(T)\ \lambda b_2\,((b_1)\ b_2)\ K}[M]]$ which is equal to $\underline{K_1}[\underline{K}[(M)\ \sigma(T)]]$, which, by the induction hypothesis, is $\underline{K[k := K_1]}[M\sigma(T)]$, which is equal to the term $\underline{(\lambda b_1\,(T)\ \lambda b_2\,((b_1)\ b_2)\ K)[k := K_1]}[M]$. □

Lemma A.3. For all K and C, $\underline{K}[\overline{C}] = \overline{C[k := K]}$.

Proof. By induction on the structure of C, using Lemma A.2 where necessary.
- Case $(K_2)\ B$. Then $\underline{K}[\overline{(K_2)\ B}] = \underline{K}[\underline{K_2}[\psi(B)]]$, which, by Lemma A.2, is equal to $\underline{K_2[k := K]}[\psi(B)] = \overline{((K_2)\ B)[k := K]}$.
- Case $((B_1)\ B_2)\ K_2$. Then $\underline{K}[\overline{((B_1)\ B_2)\ K_2}] = \underline{K}[\underline{K_2}[(\psi(B_1))\ \psi(B_2)]]$, which, by Lemma A.2, is equal to $\underline{K_2[k := K]}[(\psi(B_1))\ \psi(B_2)] = \overline{(((B_1)\ B_2)\ K_2)[k := K]}$.
- Case $(T)\ K_2$. Then $\underline{K}[\overline{(T)\ K_2}] = \underline{K}[\underline{K_2}[\sigma(T)]]$, which, by Lemma A.2, is equal to $= \underline{K_2[k := K]}[\sigma(T)] = \overline{((T)\ K_2)[k := K]}$. □

Lemma A.4. For any continuation K and term M, if $M \to_{\ell\cup\beta_v} M'$, then $\underline{K}[M] \to_{\ell\cup\beta_v} \underline{K}[M']$.

Proof. By induction on the structure of K.
- Case k. Then $\underline{k}[M] = M \to_{\ell\cup\beta_v} M' = \underline{k}[M']$.
- Case $\lambda b\,((B)\ b)\ K$. Then $\psi(B)M \to_{\ell\cup\beta_v} \psi(B)M'$ since $\psi(B)$ is a base term, and $\underline{(\lambda b\,((B)\ b)\ K)}[M] = \underline{K}[(\psi(B))\ M]$, which, by the induction hypothesis, $\to_{\ell\cup\beta_v}$-reduces to $\underline{K}[(\psi(B))\ M'] = \underline{\lambda b\,((B)\ b)\ K}[M']$.
- Case $\lambda b_1\,(T)\ \lambda b_2\,((b_1)\ b_2)\ K$. Then we have $M\sigma(T) \to_{\ell\cup\beta_v} M'\sigma(T)$. Hence, we have $\underline{\lambda b_1\,(T)\ \lambda b_2\,((b_1)\ b_2)\ K}[M] = \underline{K}[(M)\ \sigma(T)]$, which, by the induction hypothesis, $\to_{\ell\cup\beta_v}$-reduces to $K[(M')\ \sigma(T)] = \underline{\lambda b_1\,(T)\ \lambda b_2\,((b_1)\ b_2)\ K}[M']$. □

Lemma A.5. The following relations hold.
- $\underline{K}[M_1 + M_2] \to_\ell^* \underline{K}[M_1] + \underline{K}[M_2]$
- $\underline{K}[\alpha.M] \to_\ell^* \alpha.\underline{K}[M]$
- $\underline{K}[0] \to_\ell^* 0$

Proof. We prove each statement by induction on K, using Lemma A.4 where necessary. 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 $\lambda b\,((B)\ b)\ K$. Then $\underline{\lambda b\,((B)\ b)\ K}[M_1 + M_2] = \underline{K}[(\psi(B))\ (M_1 + M_2)]$, which, by Lemma A.4, $\to_\ell$-reduces to $\underline{K}[(\psi(B))\ M_1 + (\psi(B))\ M_2]$, which, by the induction hypothesis, $\to_\ell^*$-reduces to $\underline{K}[(\psi(B))\ M_1] + \underline{K}[(\psi(B))\ M_2] = \underline{\lambda b\,((B)\ b)\ K}[M_1] + \underline{\lambda b\,((B)\ b)\ K}[M_2]$.
- Case $\lambda b_1\,(S)\ \lambda b_2\,((b_1)\ b_2)\ K$. Then the term $\underline{\lambda b_1\,(S)\lambda b_2\,((b_1)\ b_2)\ K)}[M_1 + M_2]$ is equal to $\underline{K}[(M_1 + M_2)\ \sigma(S)]$ which, by Lemma A.4, $\to_\ell$-reduces to $\underline{K}[(M_1)\ \sigma(S) + (M_2)\ \sigma(S)]$, which, by the induction hypothesis, $\to_\ell^*$-reduces to $\underline{K}[(M_1)\ \sigma(S)] + \underline{K}[(M_2)\ \sigma(S)]$ which is equal to $\underline{\lambda b_1\,(S)\ \lambda b_2\,((b_1)\ b_2)\ K}[M_1] + \underline{\lambda b_1\,(S)\ \lambda b_2\,((b_1)\ b_2)\ K}[M_2]$. □

Lemma A.6. For any suspension T, if $T \to_a T'$ then $\sigma(T) \to_\ell \sigma(T')$.

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.
- Case L. Using linearity of σ. We give the following example. $\sigma(T_1 + (T_2 + T_3)) = \sigma(T_1) + (\sigma(T_2) + \sigma(T_3)) \to_\ell (\sigma(T_1) + \sigma(T_2)) + \sigma(T_3) = \sigma((T_1 + T_2) + T_3)$.

- Case ξ. 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 $\sigma(T_1 + T_2)\ \sigma(T_1) + \sigma(T_2)$, which, by the induction hypothesis, $\to_\ell$-reduces to $\sigma(T_1') + \sigma(T_2) = \sigma(T_1' + T_2)$. □

We now have the tools to prove the Lemma 3.20.

Proof of Lemma 3.20. By induction on the reduction rule, using Lemmas A.1, A.3, A.4, A.5 and A.6 where necessary.

- Case β_n. There are several sub-cases.
  - Case $(\lambda b\,((B_1)\ b)\ K)\ B_2 \to_{\beta_n} ((B_1)\ B_2)\ K$. Then $\overline{(\lambda b\,((B_1)\ b)\ K)\ B_2}$ is equal to $\lambda b\,((B_1)\ b)\ \underline{K}[\psi(B_2)] = \underline{K}[(\psi(B_1))\ \psi(B_2)] = \overline{((B_1)\ B_2)\ K}$.
  - Case $\overline{(\lambda b_1\,(S)\ \lambda b_2\,((b_1)\ b_2)\ K)\ B_1} \to_{\beta_n} (S)\ \lambda b_2\,((B_1)\ b_2)\ K$. Then we have that $\overline{(\lambda b_1\,(S)\ \lambda b_2\,((b_1)\ b_2)\ K)\ B_1} = \underline{\lambda b_1\,(S)\,\lambda b_2\,((b_1)\ b_2)\ K}[\psi(B_1)] = \underline{K}[(\psi(B_1))\ \sigma(S)] = \underline{\lambda b_2\,((B_1)\ b_2)\ K}[\sigma(S)] = \overline{(S)\ \lambda b_2\,((B_1)\ b_2)\ K}$.
  - Case $(\lambda k\,C)\ K \to_{\beta_n} C[k := K]$. Then $\overline{(\lambda k\,C)\ K} = \underline{K}[\overline{C}]$, which, by Lemma A.3, is equal to $\overline{C[k := K]}$
  - Case $((\lambda x\,S)\ B)\ K \to_{\beta_n} S[x := B]K$. Then $\psi(B)$ is a base term, and hence $(\lambda x\,\sigma(S))\ \psi(B) \to_{\ell\cup\beta_v} \sigma(S)[x := \psi(B)]$, so $\overline{((\lambda x\,S)\ B)\ K} = \underline{K}[(\lambda x\,\sigma(S))\ \psi(B)]$, which, by Lemma A.4, $\to_{\ell\cup\beta_v}$-reduces to $\underline{K}[\sigma(S)[x := \psi(B)]]$, which, by Lemma A.1, is equal to $\underline{K}[\sigma(S[x := B])] = \overline{(S[x := B])\ K}$.
- Case A. Since B and K are base terms, the only term that can match the rules is $(T)\ K$. There are three sub-cases.
  - Case $(T_1 + T_2)\ K \to_a (T_1)\ K + (T_2)\ K$. Then $\overline{(T_1 + T_2)\ K} = \underline{K}[\sigma(T_1) + \sigma(T_2)]$, which, by Lemma A.5, $\to_\ell^*$-reduces to $\underline{K}[\sigma(T_1)] + \underline{K}[\sigma(T_2)] = \overline{(T_1)\ K + (T_2)\ K}$.
  - Case $(\alpha.T)\ K \to_a \alpha.((T)\ K)$. Then $\overline{(\alpha.T)\ K}$, $= \underline{K}[\alpha.\sigma(T)]$ which, by Lemma A.5, $\to_\ell^*$-reduces to $\alpha.\underline{K}[\sigma(T)] = \overline{\alpha.((T)\ K)}$.
  - Case $(0)\ K \to_a 0$. Then $\overline{(0)\ K}$, $= \underline{K}[0]$ which, by Lemma A.5, $\to_\ell^*$-reduces to $0 = \overline{0}$.
- 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 $\overline{D_1 + (D_2 + D_3)} = \overline{D_1} + (\overline{D_2} + \overline{D_3})$, and this $\to_\ell$-reduces to $(\overline{D_1} + \overline{D_2}) + \overline{D_3}$ $= \overline{(D_1 + D_2) + D_3}$.
- Case ξ. There are 4 sub-cases.
  - Case $(T)\ K \to_a (T')\ K$ and $T \to_a T'$. Then $\sigma(T) \to_\ell \sigma(T')$ by Lemma A.6, therefore $\overline{(T)\ K} = \underline{K}[\sigma(T)]$, by Lemma A.4, $\to_\ell$-reduces to $\underline{K}[\sigma(T')] = \overline{(T')\ K}$.
  - 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 the induction hypothesis $\overline{D_1} \to_\ell \overline{D_1'}$, therefore $\overline{D_1 + D_2} = \overline{D_1} + \overline{D_2} \to_\ell \overline{D_1}' + \overline{D_2} = \overline{D_1' + D_2}$. □

A.11. **Proof of Lemma 3.22.**

Lemma 3.22. $\{\!|M[x := N]|\!\} = \{\!|M|\!\}[x := \{\!|N|\!\}]$.

Proof. Structural induction on M.

- $M = x$. Then $\{\!|x[x := N]|\!\} = \{\!|N|\!\} = x[x := \{\!|N|\!\}] = \{\!|x|\!\}[x := \{\!|N|\!\}]$.
- $M = y$. Then $\{\!|y[x := N]|\!\} = y = \{\!|y|\!\}[x := \{\!|N|\!\}]$.

- $M = 0$. Analogous to previous case.
- $M = \lambda y\, M'$. Then $\{\!|(\lambda y\, M')[x := N]|\!\} = \{\!|\lambda y\, (M'[x := N])|\!\}$, which, by the induction hypothesis, is equal to $\lambda f\ (f)\ \lambda y\, \{\!|M'[x := N]|\!\} = \lambda f\ (f)\ \lambda y\, \{\!|M'|\!\}[x := \{\!|N|\!\}] = (\lambda f\ (f)\ \lambda y\, \{\!|M'|\!\})[x := \{\!|N|\!\}] = \{\!|M|\!\}[x := \{\!|N|\!\}]$.
- $M = (N_1)\ N_2$. Then $\{\!|M[x := N]|\!\} = \{\!|((N_1)\ N_2)[x := N]|\!\}$, which is equal to the term $\{\!|(N_1[x := N])\ N_2[x := N]|\!\}$, and this, by the induction hypothesis, is equal to $\lambda f\, (\{\!|N_1[x := N]|\!\})\ \lambda g\, ((g)\ \{\!|N_2[x := N]|\!\})\ f = \lambda f\, (\{\!|N_1|\!\}[x := \{\!|N|\!\}])\ \lambda g\, ((g)\ \{\!|N_2|\!\}[x := \{\!|N|\!\}])\ f = (\lambda f\, (\{\!|N_1|\!\})\ \lambda g\, ((g)\ \{\!|N_2|\!\})\ f)[x := \{\!|N|\!\}] = \{\!|(N_1)\ N_2|\!\}[x := \{\!|N|\!\}] = \{\!|M|\!\}[x := \{\!|N|\!\}]$.
- $M = \alpha.M'$. Then $\{\!|M[x := N]|\!\} = \{\!|(\alpha.M')[x := N]|\!\} = \{\!|\alpha.(M'[x := N])|\!\}$, which, by the induction hypothesis, is equal to $\lambda f\, (\alpha.\{\!|M'[x := N]|\!\}))\ f = \lambda f\, (\alpha.\{\!|M'|\!\}[x := \{\!|N|\!\}])\ f = \lambda f\, (\alpha.\{\!|M'|\!\})\ f[x := \{\!|N|\!\}] = \{\!|\alpha.M'|\!\}[x := \{\!|N|\!\}] = \{\!|M|\!\}[x := \{\!|N|\!\}]$.
- $M = N_1 + N_2$. Then $\{\!|M[x := N]|\!\} = \{\!|(N_1 + N_2)[x := N]|\!\}$ which is equal to the term $\{\!|N_1[x := N] + N_2[x := N]|\!\}$, which, by the induction hypothesis, is $\lambda f\, (\{\!|N_1[x := N]|\!\} + \{\!|N_2[x := N]|\!\})\ f = \lambda f\, (\{\!|N_1|\!\}[x := \{\!|N|\!\}] + \{\!|N_2|\!\}[x := \{\!|N|\!\}])\ f$, which is $\lambda f\, ((\{\!|N_1|\!\} + \{\!|N_2|\!\})\ f[x := \{\!|N|\!\}] = \{\!|N_1 + N_2|\!\}[x := \{\!|N|\!\}] = \{\!|M|\!\}[x := \{\!|N|\!\}]$. □

A.12. **Proof of Lemma 3.28.**

Lemma 3.28. If K is a base term, for any term M $(\{\!|M|\!\})\ K \to^*_{\ell\cup\beta_v} M : K$.

Proof. Structural induction on M.

- $M = x$. Then $(\{\!|x|\!\})\ K = (x)\ K = x : K$.
- $M = \lambda x\, N$. Then $(\{\!|\lambda x\, N|\!\})\ K = (\lambda f\ (f)\ \lambda x\, \{\!|N|\!\})\ K$ and by definition of Φ this is equal to $(\lambda f\ (f)\ \Phi(\lambda x\, N))\ K \to_{\ell\cup\beta_v} (K)\ \Phi(\lambda x\, N) = \lambda x\, N : K$.
- $M = 0$. Then $(\{\!|0|\!\})\ K = (\lambda f\, (0)\ f)\ K \to_{\ell\cup\beta_v} (0)\ K \to_{\ell\cup\beta_v} 0 = 0 : K$.
- $M = M' + N$. Then $(\{\!|M' + N|\!\})\ K = (\lambda f\, (\{\!|M'|\!\} + \{\!|N|\!\})\ f)\ K \to_{\ell\cup\beta_v} (\{\!|M'|\!\} + \{\!|N|\!\})\ K$ which $\to_{\ell\cup\beta_v}$-reduces by the induction hypothesis to $M' : K + N : K = M' + N : K$.
- $M = \alpha.N$. Then $(\{\!|\alpha.N|\!\})\ K = (\lambda f\, (\alpha.\{\!|N|\!\})\ f)\ K \to_{\ell\cup\beta_v} (\alpha.\{\!|N|\!\})\ K$ which $\to_{\ell\cup\beta_v}$-reduces to $\alpha.(\{\!|N|\!\})\ K)$ and this, by the induction hypothesis, $\to_{\ell\cup\beta_v}$-reduces to $\alpha.(N : K) = \alpha.N : K$.
- $M = (M')\ N$. Then $(\{\!|(M')\ N|\!\})\ K = (\lambda f\, (\{\!|M'|\!\})\ \lambda g\, ((g)\ \{\!|N|\!\})\ f)\ K$ which $\to_{\ell\cup\beta_v}$-reduces to $(\{\!|M'|\!\})\ \lambda g\, ((g)\ \{\!|N|\!\})\ K$. Note that $\lambda g\, ((g)\ \{\!|N|\!\})\ K$ is a value, so by the induction hypothesis the above term reduces to $M' : \lambda g\, ((g)\ \{\!|N|\!\})\ K$. We do a second induction, over M', to prove that $M' : \lambda g\, ((g)\ \{\!|N|\!\})\ K \to^*_{\ell\cup\beta_v} (M')\ N : K$.
  - If $M' = (M_1)\ M_2$, then $M' : \lambda g\, ((g)\ \{\!|N|\!\})\ K = ((M_1)\ M_2)\ N : K = (M')\ N : K$.
  - If $M' = x$ then $M' : \lambda g\, ((g)\ \{\!|N|\!\})\ K = (x)\ \lambda g\, ((g)\ \{\!|N|\!\})\ K = (M')\ N : K$.
  - If $M' = \lambda x\, M_1$ then $M' : \lambda g\, ((g)\ \{\!|N|\!\})\ K = (\lambda g\, ((g)\ \{\!|N|\!\})\ K)\ \Phi(M')$ which $\to_{\ell\cup\beta_v}$-reduces to $((\Phi(M'))\ \{\!|N|\!\})\ K = (M')\ N : K$.
  - If $M' = \alpha.M_1$, then $\alpha.M_1 : \lambda g\, ((g)\ \{\!|N|\!\})\ K = \alpha.(M_1 : \lambda g\, ((g)\ \{\!|N|\!\})\ K)$ which $\to^*_{\ell\cup\beta_v}$-reduces by the induction hypothesis to $\alpha.((M_1)\ N : K) = (\alpha.M_1)\ N : K = (M')\ N : K$.
  - If $M' = M_1 + M_2$, then $M' : \lambda g\, ((g)\ \{\!|N|\!\})\ K = M_1 + M_2 : \lambda g\, ((g)\ \{\!|N|\!\})\ K$ which is equal to $M_1 : \lambda g\, ((g)\ \{\!|N|\!\})\ K + M_2 : \lambda g\, ((g)\ \{\!|N|\!\})\ K$ which $\to^*_{\ell\cup\beta_v}$-reduces by the induction hypothesis to $(M_1)\ N : K + (M_2)\ N : K = (M_1 + M_2)\ N : K = (M')\ N : K$.
  - If $M' = 0$ then $M' : \lambda g\, ((g)\ \{\!|N|\!\})\ K = 0 : \lambda g\, ((g)\ \{\!|N|\!\})\ K = 0 = (0)\ N : K = (M')\ N : K$. □

A.13. **Proof of Lemma 3.29.**

Lemma 3.29. If $M \to_{a\cup\beta_n} N$ then for all K base term, $M : K \to^*_{\ell\cup\beta_v} N : K$

Proof. Case by case on the rules of λ_{alg}.

Rule β_v: $(\lambda x\ M)\ N : K = ((\Phi(\lambda x\ M))\ \{\!|N|\!\})\ K = ((\lambda x\ (\{\!|M|\!\}))\ \{\!|N|\!\})\ K$. Since $\{\!|N|\!\}$ is a base term, this last term $\to_{\ell\cup\beta_v}$-reduces to $\{\!|M|\!\}[x := (\{\!|N|\!\})]\ K$, which, by Lemma 3.22, is equal to $\{\!|M[x := N]|\!\}\ K$, and this, by Lemma 3.28, $\to^*_{\ell\cup\beta_v}$-reduces to $M[x := N] : K$.

Rules A:

- Let $(M + N)\ L \to_{a\cup\beta_n} (M)\ L + (N)\ L$. $(M + N)\ L : K = ((M)\ L + (N)\ L) : K$.
- Let $(\alpha.M)\ N \to_{a\cup\beta_n} \alpha.(M)\ N$. $(\alpha.M)\ N : K = \alpha.(M)\ N : K$
- Let $(0)\ N \to_{a\cup\beta_n} 0$. $(0)\ N : K = 0 = 0 : K$

Rules F and S:

- $\alpha.(M + N) \to_{a\cup\beta_n} \alpha.M + \alpha.N$. Then $\alpha.(M + N) : K = \alpha.(M : K + N : K) \to_{\ell\cup\beta_v} \alpha.(M : K) + \alpha.(N : K) = \alpha.M + \alpha.N : K$.
- $\alpha.M+\beta.M \to_{a\cup\beta_n} (\alpha+\beta).M$. Then $\alpha.M+\beta.M : K = \alpha.(M : K)+\beta.(M : K) \to_{\ell\cup\beta_v} (\alpha + \beta).(M : K) = (\alpha + \beta).M : K$.
- $\alpha.M + M \to_{a\cup\beta_n} (\alpha + 1).M$. Then $\alpha.M + M : K = \alpha.M : K + M : K = \alpha.(M : K) + M : K \to_{\ell\cup\beta_v} (\alpha + 1).(M : K) = (\alpha + 1).M : K$.
- $M + M \to_{a\cup\beta_n} (1 + 1).M$. Then $M + M : K = M : K + M : K \to_{\ell\cup\beta_v} (1 + 1).(M : K) = (1 + 1).M : K$.
- $0 + M \to_{a\cup\beta_n} M$. Then $0 + M : K = (0 : K) + (M : K) = 0 + (M : K) \to_{\ell\cup\beta_v} M : K$.
- $\alpha.(\beta.M) \to_{a\cup\beta_n} (\alpha\beta).M$. Then $\alpha.(\beta.M) : K = \alpha.(\beta.M : K) = \alpha.(\beta.(M : K))$ which $\to_{\ell\cup\beta_v}$-reduces to $(\alpha\beta).(M : K) = (\alpha\beta).M : K$.
- $1.M \to_{a\cup\beta_n} M$. Then $1.M : K = 1.(M : K) \to_{\ell\cup\beta_v} M : K$.
- $0.M \to_{a\cup\beta_n} 0$. Then $0.M : K = 0.(M : K) \to_{\ell\cup\beta_v} 0 = 0 : K$.
- $\alpha.0 \to_{a\cup\beta_n} 0$. Then $\alpha.0 : K = \alpha.(0 : K) = \alpha.0 \to_{\ell\cup\beta_v} 0 = 0 : K$.

Rules *Asso* and *Com*:

- $M + (N + L) \to_{a\cup\beta_n} (M + N) + L$. Then $M + (N + L) : K = M : K + (N + L : K) = M : K + (N : K + L : K) \to_{\ell\cup\beta_v} (M : K + N : K) + L : K = M + N : K + L : K = (M + N) + L : K$.
- $M + N \to_{a\cup\beta_n} N + M$. Then $M + N : K = M : K + N : K \to_{\ell\cup\beta_v} N : K + M : K = N + M : K$.

Rules ξ: Assume $M \to_{a\cup\beta_n} M'$, and that for all K base term, $M : K \to^*_{\ell\cup\beta_v} M' : K$. We show that the result also holds for each contextual rule.

- $M + N \to_{a\cup\beta_n} M' + N$. Then $M + N : K = M : K + N : K \to^*_{\ell\cup\beta_v} M' : K + N : K = M' + N : K$.
- $N + M \to_{a\cup\beta_n} N + M'$, analogous to previous case.
- $\alpha.M \to_{a\cup\beta_n} \alpha.M'$. Then $\alpha.M : K = \alpha.(M : K) \to^*_{\ell\cup\beta_v} \alpha.(M' : K) = \alpha.M' : K$.
- $(M)\ N \to_{a\cup\beta_n} (M')\ N$ Case by case:
  - $M = B$. Absurd since a base term cannot reduce.
  - $M = \alpha.M_1$. Case by case on the possible $\to_{a\cup\beta_n}$-reductions of M:
    - $M' = \alpha.M'_1$ with $M_1 \to_{a\cup\beta_n} M'_1$. Then $(\alpha.M_1)\ N : K = \alpha.(M_1)\ N : K = \alpha.((M_1)\ N : K)$ which, by the induction hypothesis, $\to_{\ell\cup\beta_v}$-reduces to $\alpha.((M'_1)\ N : K) = \alpha.(M'_1)\ N : K = (\alpha.M'_1)\ N : K$.
    - $M = \alpha.(\beta.M_3)$ and $M' = (\alpha\beta).M_3$. Then $(\alpha.(\beta.M_3))\ N : K = \alpha.(\beta.((M_3)\ N : K)) \to_{\ell\cup\beta_v} (\alpha\beta).((M_3)\ N : K) = ((\alpha\beta).M_3)\ N : K$.

  - $M = \alpha.(L_1 + L_2)$ and $M' = \alpha.L_1 + \alpha.L_2$. Then $(\alpha.(L_1 + L2))\ N : K = \alpha.((L_1)\ N : K + (L_2)\ N : K) \to_{\ell \cup \beta_v} \alpha.((L_1)\ N : K) + \alpha.((L_2)\ N : K) = (\alpha.L_1 + \alpha.L2)\ N : K$.
  - $\alpha = 1$ and $M' = M_1$. Then $(1.M_1)\ N : K = 1.((M_1)\ N : K)$ and this $\to_{\ell \cup \beta_v}$-reduces to $(M_1)\ N : K$.
  - $\alpha = 0$ and $M' = 0$. Then $(0.M_1)\ N : K = 0.((M_1)\ N : K) \to_{\ell \cup \beta_v} 0 = (0)\ N : K$.
  - $M_1 = 0$ and $M' = 0$. Then $(\alpha.0)\ N : K = \alpha.((0)\ N : K) = \alpha.0 \to_{\ell \cup \beta_v} 0 = (0)\ N : K$.
- $M = M_1 + M_2$. Case by case on the possible $\to_{a \cup \beta_n}$-reductions of M:
  - $M' = M_1' + M_2$ with $M_1 \to_{a \cup \beta_n} M_1'$. Then $(M_1 + M_2)\ N : K = (M_1)\ N : K + (M_2)\ N : K$ which by the induction hypothesis $\to_{\ell \cup \beta_v}$-reduces to $(M_1')\ N : K + (M_2)\ N : K = (M_1' + M_2)\ N : K$.
  - $M' = M_1 + M_2'$ with $M_2 \to_{a \cup \beta_n} M_2'$. Analogous to previous case.
  - $M_2 = L_1 + L_2$ and $M' = (M_1 + L_1) + L_2$. Then $(M_1 + (L_1 + L_2))\ N : K = (M_1)\ N : K + ((L_1)\ N : K + (L_2)\ N : K)$ which $\to_{\ell \cup \beta_v}$-reduces to $((M_1)\ N : K + (L_1)\ N : K) + (L_2)\ N : K = ((M_1 + L_1) + L_2)\ N : K$.
  - $M_1 = L_1 + L_2$ and $M' = L_1 + (L_2 + M_2)$. Analogous to previous case.
  - $M' = M_2 + M_1$. Then $(M_1 + M_2)\ N : K = (M_1)\ N : K + (M_2)\ N : K \to_{\ell \cup \beta_v} (M_2)\ N : K + (M_1)\ N : K = (M_2 + M_1)\ N : K$.
  - $M_1 = \alpha.M_3$, $M_2 = \beta.M_3$ and $M' = (\alpha + \beta).M_3$. Then $(\alpha.M_3 + \beta.M_3)\ N : K = \alpha.((M_3)\ N : K) + \beta.((M_3)\ N : K) \to_{\ell \cup \beta_v} (\alpha + \beta).((M_3)\ N : K) = ((\alpha + \beta).M_3)\ N : K$.
  - $M_1 = \alpha.M_3$, $M_2 = M_3$ and $M' = (\alpha + 1).M_3$. Analogous to previous case.
  - $M_1 = M_2$ and $M' = (1 + 1).M_1$. Analogous to previous case.
- $M = 0$. Absurd since 0 does not reduce.
- $M = (M_1)\ M_2$. Then $((M_1)\ M_2)\ N : K = (M_1)\ M_2 : \lambda g\,((g)\ \{\!|N|\!\})\ K$, which by the induction hypothesis $\to_{\ell \cup \beta_v}$-reduces to $M' : \lambda g\,((g)\ \{\!|N|\!\})\ K$. We do a second induction, over M', to prove that $M' : \lambda g\,((g)\ \{\!|N|\!\})\ K \to^*_{\ell \cup \beta_v}$-reduces to $(M')\ N : K$.
  - If $M' = (M_1')\ M_2'$, then $M' : \lambda g\,((g)\ \{\!|N|\!\})\ K = ((M_1')\ M_2')\ N : K = (M')\ N : K$.
  - If $M' = x$ then $M' : \lambda g\,((g)\ \{\!|N|\!\})\ K = (x)\ \lambda g\,((g)\ \{\!|N|\!\})\ K = (M')\ N : K$.
  - If $M' = \lambda x\, M_1'$ then $M' : \lambda g\,((g)\ \{\!|N|\!\})\ K$ is $(\lambda g\,((g)\ \{\!|N|\!\})\ K)\ \Phi(M') \to_{\ell \cup \beta_v} ((\Phi(M'))\ \{\!|N|\!\})\ K = (M')\ N : K$.
  - If $M' = \alpha.M_1'$, then $\alpha.M_1' : \lambda g\,((g)\ \{\!|N|\!\})\ K = \alpha.(M_1' : \lambda g\,((g)\ \{\!|N|\!\})\ K)$ which $\to^*_{\ell \cup \beta_v}$-reduces by the induction hypothesis to $\alpha.((M_1')\ N : K) = (\alpha.M_1')\ N : K = (M')\ N : K$.
  - If $M' = M_1' + M_2'$, then $M' : \lambda g\,((g)\ \{\!|N|\!\})\ K = M_1' + M_2' : \lambda g\,((g)\ \{\!|N|\!\})\ K$ which is equal to $M_1' : \lambda g\,((g)\ \{\!|N|\!\})\ K + M_2' : \lambda g\,((g)\ \{\!|N|\!\})\ K$ which $\to^*_{\ell \cup \beta_v}$-reduces by the induction hypothesis to $(M_1')\ N : K + (M_2')\ N : K = (M_1' + M_2')\ N : K = (M')\ N : K$.
  - If $M' = 0$ then $M' : \lambda g\,((g)\ \{\!|N|\!\})\ K = 0 : \lambda g\,((g)\ \{\!|N|\!\})\ K = 0 = (0)\ N : K = (M')\ N : K$. □

A.14. **Proof of Lemma 3.31.**

Lemma 3.31. For any term M, $\overline{(\{\!|M|\!\})\ k} = M$.

Proof. By induction on M.

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

- Case $\lambda x\, M$. Then $\sigma(\{\!|\lambda x\, M|\!\}) = \sigma(\lambda k\,(k)\ \lambda x\,\{\!|M|\!\}) = \overline{(k)\ \lambda x\,\{\!|M|\!\}} = \underline{k}[\phi(\lambda x\,\{\!|M|\!\})] = \phi(\lambda x\,\{\!|M|\!\}) = \lambda x\,\sigma(\{\!|M|\!\})$, which, by the induction hypothesis, is equal to $\lambda x\, M$.
- Case $(M)\ N$. Then $\sigma(\{\!|(M)\ N|\!\}) = \sigma(\lambda k\,(\{\!|M|\!\})\ \lambda b.((b)\ \{\!|N|\!\})\ k)$ which is equal to $\overline{(\{\!|M|\!\})\ \lambda b.((b)\ \{\!|N|\!\})\ k} = \underline{\lambda b.((b)\ \{\!|N|\!\})\ k}[\sigma(\{\!|M|\!\})] = \underline{k}[(\sigma(\{\!|M|\!\}))\ \sigma(\{\!|N|\!\})]$, equal to $(\sigma(\{\!|M|\!\}))\ \sigma(\{\!|N|\!\})$, which, by the induction hypothesis, is equal to $(M)\ N$.
- Case 0. Then $\sigma(\{\!|0|\!\}) = \sigma(0) = 0$
- Case $\alpha.M$. Then $\sigma(\{\!|\alpha.M|\!\}) = \sigma(\lambda k\,(\alpha.\{\!|M|\!\})\ k)$, which is equal to the term $\overline{(\alpha.\{\!|M|\!\})k} = \underline{k}[\sigma(\alpha.\{\!|M|\!\})] = \sigma(\alpha.\{\!|M|\!\}) = \alpha.\sigma(\{\!|M|\!\})$, which, by the induction hypothesis, is equal to $\alpha.M$.
- Case $M + N$. Then $\sigma(\{\!|M + N|\!\}) = \sigma(\lambda k\,(\{\!|M|\!\} + \{\!|N|\!\})\ k) = \overline{(\{\!|M|\!\} + \{\!|N|\!\})\ k} = \underline{k}[\sigma(\{\!|M|\!\} + \{\!|N|\!\})] = \sigma(\{\!|M|\!\} + \{\!|N|\!\}) = \sigma(\{\!|M|\!\}) + \sigma(\{\!|N|\!\})$, which, by the induction hypothesis, is equal to $M + N$. □

A.15. **Proof of Lemma 3.32.**

Lemma 3.32. For any value V, $\overline{V : k} = V$.

Proof. By induction on V.
- Case 0. Then $\overline{0 : k} = \overline{0} = 0$.
- Case x. Then $\overline{x : k} = \overline{(x)\ k} = \underline{k}[x] = x$.
- 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 3.31.
- Case $\alpha.V$. Then $\overline{\alpha.V : k} = \overline{\alpha.(V : k)} = \alpha.\overline{V : k} = \alpha.V$ by the induction hypothesis.
- Case $U + V$. Then $\overline{U + V : k} = \overline{U : k + V : k} = \overline{U : k} + \overline{V : k} = U + V$ by the induction hypothesis. □

A.16. **Proof of Lemma 3.33.**

Lemma 3.33. For any computation D, if $D \to_{\ell \cup \beta_v} D'$ then $\overline{D} \to^*_{a \cup \beta_n} \overline{D'}$. Also, if $D \to_\ell D'$ then $\overline{D} \to^*_a \overline{D'}$.

In order to prove this lemma, we need intermediary results similar to Lemmas A.1, A.2, A.3, A.4, A.5, and A.6.

Lemma A.7. The following equalities hold.
(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. We prove simultaneously the four properties by induction on the structure of B, T, C and K.
(1) Cases for B.
  - Case $\lambda y\, S$. Then $\phi(\lambda x\, S_1)[x := \sigma(S)] = (\lambda y\, \sigma(S_1))[x := \sigma(S)]$, which, by the induction hypothesis, is equal to $\lambda y\, \sigma(S_1[x := S]) = \phi((\lambda y\, S_1)[x := S])$.
(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 $\sigma(\lambda k\, C)[x := \sigma(S)] = \overline{C}[x := \sigma(S)]$, which, by the induction hypothesis, is equal to $\overline{C[x := S]} = \sigma((\lambda k\, C)[x := S])$.
- Case 0. Then $\sigma(0)[x := \sigma(S)] = 0 = \sigma(0[x := S])$.
- Case $\alpha.T$. Then $\sigma(\alpha.T)[x := \sigma(S)] = (\alpha.\sigma(T))[x := \sigma(S)]$, which, by the induction hypothesis, is equal to $\alpha.\sigma(T[x := S]) = \sigma((\alpha.T)[x := S])$.
- Case $T_1 + T_2$. Then $\sigma(T_1 + T_2)[x := \sigma(S)] = (\sigma(T_1) + \sigma(T_2))[x := \sigma(S)]$, which, by the induction hypothesis, is equal to $\sigma(T_1[x := S]) + \sigma(T_2[x := S]) = \sigma((T_1 + T_2)[x := S])$.

(3) Cases for C.

- Case $(K)\ B$. Then $\overline{(K)\ B}[x := \sigma(S)] = \underline{K}[\phi(B)][x := \sigma(S)]$, which, by the induction hypothesis, is equal to $\underline{K[x := S]}[\phi(B)[x := \sigma(S)]]$, which, by the induction hypothesis, is equal to $\underline{K[x := S]}[\phi(B[x := S])] = \overline{((K)\ B)[x := S]}$.
- Case $((B)\ S_2)\ K$. Then $\overline{(B)\ S_2)\ K}[x := \sigma(S)] = \underline{K}[(\psi(B))\ \sigma(S_2)][x := \sigma(S)]$, which, by the induction hypothesis, is equal to $\underline{K[x := S]}[((\psi(B))\ \sigma(S_2))[x := \sigma(S)]]$, which, by the induction hypothesis, is equal to $\underline{K[x := S]}[(\psi(B[x := S]))\ \sigma(S_2[x := S])] = \overline{(((B)\ S_2)\ K)[x := S]}$.
- Case $(T)\ K$. Then $\overline{(T)\ K}[x := \sigma(S)] = \underline{K}[\sigma(T)][x := \sigma(S)]$, which, by the induction hypothesis, is equal to $\underline{K[x := S]}[\sigma(T)[x := \sigma(S)]]$, which, by the induction hypothesis, is equal to $\underline{K[x := S]}[\sigma(T[x := S])] = \overline{((T)\ K)[x := S]}$

(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 $\lambda b\,((b)\ S_2)\ K$. Then $\underline{\lambda b\,((b)\ S_2)\ K}[M][x := \sigma(S)] = \underline{K}[M\sigma(S_2)][x := \sigma(S)]$, which, by the induction hypothesis, is $\underline{K[x := S]}[((M)\ \sigma(S_2))[x := \sigma(S)]]$, which, by the induction hypothesis, is equal to $\underline{K[x := S]}[M[x := \sigma(S)]\sigma(S_2[x := S])] = \underline{(\lambda b\,((b)\ S_2)\ K)[x := S]}[M[x := \sigma(S)]]$. □

Lemma A.8. For all terms M and continuations K_1 and K_2, we have, $\underline{K_1}[\underline{K_2}[M]] = \underline{K_2[k := K_1]}[M]$.

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 $\lambda b\,((b)\ S)\ K$. Then $\underline{K_1}[\underline{\lambda b\,((b)\ S)\ K}[M]] = \underline{K_1}[\underline{K}[(M)\ \sigma(S)]]$, which, by the induction hypothesis, is equal to $\underline{K[k := K_1]}[(M)\ \sigma(S)] = \underline{(\lambda b\,((b)\ S)\ K)[k := K_1]}[M]$. □

Lemma A.9. For all K and C, $\underline{K}[\overline{C}] = \overline{C[k := K]}$.

Proof. By induction on the structure of C, using Lemma A.8 where necessary.

- Case $(K_2)\ B$. Then $\underline{K}[\overline{(K_2)\ B}] = \underline{K}[\underline{K_2}[\phi(B)]]$, which, by Lemma A.8, is equal to $\underline{K_2[k := K]}[\phi(B)] = \overline{((K_2)\ B)[k := K]}$.
- Case $((B)\ S)\ K_2$. Then $\underline{K}[\overline{((B)\ S)\ K_2}] = \underline{K}[\underline{K_2}[(\phi(B))\ \sigma(S)]]$, which, by Lemma A.8, is equal to $\underline{K_2[k := K]}[(\phi(B))\ \sigma(S)] = \overline{(((B)\ S)\ K_2)[k := K]}$.
- Case $(T)\ K_2$. Then $\underline{K}[\overline{(T)\ K_2}] = \underline{K}[\underline{K_2}[\sigma(T)]]$, which, by Lemma A.8, is equal to $\underline{K_2[k := K]}[\sigma(T)] = \overline{((T)\ K_2)[k := K]}$. □

Lemma A.10. For any continuation K and term M, if $M \to_{a \cup \beta_n} M'$ then $\underline{K}[M] \to_{a \cup \beta_n} \underline{K}[M']$.

Proof. By induction on the structure of K.

- Case k. Then $\underline{k}[M] = M \to_{a\cup\beta_n} M' = \underline{k}[M']$.
- Case $\lambda b\,((b)\ S)\ K$. Then we have $M\sigma(S) \to_{a\cup\beta_n} M'\sigma(S)$, and $\underline{\lambda b\,((b)\ S)\ K}[M] = \underline{K}[(M)\ \sigma(S)]$, and this, by the induction hypothesis, $\to_{a\cup\beta_n}$-reduces to $\underline{K}[(M')\ \sigma(S)] = \underline{\lambda b\,((b)\ S)\ K}[M']$. □

Lemma A.11. For any continuation K, scalar α and terms M, M_1 and M_2, the following relations hold.
- $\underline{K}[M_1 + M_2] \to_a^* \underline{K}[M_1] + \underline{K}[M_2]$
- $\underline{K}[\alpha.M] \to_a^* \alpha.\underline{K}[M]$
- $\underline{K}[0] \to_a^* 0$

Proof. We prove each statement by induction on K, using Lemma A.10 where necessary.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 $\lambda b\,((b)\ S)\ K$. Then $\underline{\lambda b\,((b)\ S)\ K}[M_1+M_2] = \underline{K}[(M_1+M_2)\ \sigma(S)]$, which, by Lemma A.4, $\to_a$-reduces to $\underline{K}[(M_1)\ \sigma(S) + (M_2)\ \sigma(S)]$, and this, by the induction hypothesis, $\to_a^*$-reduces to $\underline{K}[(M_1)\ \sigma(S)] + \underline{K}[(M_2)\ \sigma(S)] = \underline{\lambda b\,((b)\ S)\ K}[M_1] + \underline{\lambda b\,((b)\ S)\ K}[M_2]$ □

Lemma A.12. For any suspension T, if $T \to_\ell T'$ 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.
- Case L. Using linearity of σ. We give the following example. $\sigma(T_1 + (T_2 + T_3)) = \sigma(T_1) + (\sigma(T_2) + \sigma(T_3)) \to_a (\sigma(T_1) + \sigma(T_2)) + \sigma(T_3) = \sigma((T_1 + T_2) + T_3)$.
- Case ξ. Using linearity and the induction hypothesis. We give the following example. Consider the case $T_1 + T_2 \to_\ell T_1' + T_2$ with $T_1 \to_\ell T_1'$. Then $\sigma(T_1 + T_2) = \sigma(T_1) + \sigma(T_2)$, which, by the induction hypothesis, $\to_a$-reduces to $\sigma(T_1') + \sigma(T_2) = \sigma(T_1' + T_2)$. □

We now have the tools to prove the Lemma 3.33.

Proof of Lemma 3.33. By induction on the reduction rule, using Lemmas A.7, A.9, A.10, A.11 and A.12 where necessary. The rules $\xi_{\lambda_{lin}}$ and A_r are not applicable since arguments in the target language are always base terms.
- Case β_v. There are several sub-cases.
  - Case $(\lambda b\,((b)\ S)\ K)\ B \to_{\beta_v} ((B)\ S)\ K$. Then $\overline{(\lambda b\,((b)\ S)\ K)\ B}$ which is equal to $\underline{\lambda b\,((b)\ S)\ K}[\phi(B)] = \underline{K}[(\phi(B))\ \sigma(S)] = \overline{((B)\ S)\ K}$.
  - Case $(\lambda k\,C)\ K \to_{\beta_v} C[k := K]$. Then $\overline{(\lambda k\,C)\ K} = \underline{K}[\overline{C}]$, which, by Lemma A.9, is equal to $\overline{C[k := K]}$.
  - Case $((\lambda x\,S)\ S_2)\ K \to_{\beta_v} (S[x := S_2])\ K$. Then $(\lambda x\,\sigma(S))\ \sigma(S_2) \to_a \sigma(S)[x := \sigma(S_2)]$, and $\overline{((\lambda x\,S)\ S_2)\ K} = \underline{K}[(\lambda x\,\sigma(S))\ \sigma(S_2)]$, which, by Lemma A.10, $\to_\ell$-reduces to $\underline{K}[\sigma(S)[x := \sigma(S_2)]]$, which, by Lemma A.7, is equal to $\underline{K}[\sigma(S[x := S_2])] = \overline{(S[x := S_2])\ K}$.
- Case A_l. Since B and K are base terms, the only term that can match the rules is $(T)\ K$. There are three sub-cases.
  - Case $(T_1 + T_2)\ K \to_\ell (T_1)\ K + (T_2)\ K$. Then $\overline{(T_1 + T_2)\ K} = \underline{K}[\sigma(T_1) + \sigma(T_2)]$, which, by Lemma A.11, $\to_a^*$-reduces to $\underline{K}[\sigma(T_1)] + \underline{K}[\sigma(T_2)] = \overline{(T_1)\ K + (T_2)\ K}$.
  - Case $(\alpha.T)\ K \to_\ell \alpha.((T)\ K)$. Then $\overline{(\alpha.T)\ K}$, $= \underline{K}[\alpha.\sigma(T)]$, which, by Lemma A.11, $\to_a^*$-reduces to $\alpha.\underline{K}[\sigma(T)] = \overline{\alpha.((T)\ K)}$.
  - Case $(0)\ K \to_\ell 0$. Then $\overline{(0)\ K} = \underline{K}[0]$, which, by Lemma A.11, $\to_a^*$-reduces to $0 = \overline{0}$.

- 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 $D_1 + (D_2 + D_3) \to_\ell (D_1 + D_2) + D_3$. Then $\overline{D_1 + (D_2 + D_3)} = \overline{D_1} + (\overline{D_2} + \overline{D_3}) \to_a (\overline{D_1} + \overline{D_2}) + \overline{D_3}$, which is equal to $\overline{(D_1 + D_2) + D_3}$.
- Case ξ. There are 4 sub-cases.
  - Case $(T)\ K \to_\ell (T')\ K$ and $T \to_\ell T'$. Then $\sigma(T) \to_a \sigma(T')$ by Lemma A.12, therefore $\overline{(T)\ K} = \underline{K}[\sigma(T)]$, which, by Lemma A.10, $\to_a$-reduces to $\underline{K}[\sigma(T')] = \overline{(T')\ K}$.
  - The other three cases are similar to each other. We give the following example. Consider $D_1 + D_2 \to_\ell D_1' + D_2$ and $D_1 \to_\ell D_1'$. Then by the induction hypothesis $\overline{D_1} \to_a \overline{D_1'}$, therefore $\overline{D_1 + D_2} = \overline{D_1} + \overline{D_2} \to_a \overline{D_1}' + \overline{D_2} = \overline{D_1' + D_2}$. □