de Bruijn Indices for Metaterms

EDUARDO BONELLI, *CONICET and LIFIA, Facultad de Informática, Universidad Nacional de La Plata, Argentina.*
Email: eduardo@sol.info.unlp.edu.ar

DELIA KESNER, *PPS, CNRS and Université Paris 7, France.*
Email: kesner@pps.jussieu.fr

ALEJANDRO RIOS, *Departamento de Computación, Universidad de Buenos Aires, Argentina.*
Email: rios@dc.uba.ar

Abstract

In this paper we encode higher-order rewriting with names into higher-order rewriting in de Bruijn notation. This notation not only is defined for terms (as usually done in the literature) but also for metaterms, which are the syntactical objects used to express the rewriting rules of higher-order systems. Several examples are discussed. Fundamental properties such as confluence and normalisation are shown to be preserved.

Keywords: Higher-order rewriting, de Bruijn indices, alpha-conversion.

1 Introduction

Rewriting is a very powerful method used to model computation where rewrite rules are used to transform an expression into another simpler expression. The set of results or values of a computation may be interpreted as those expressions which cannot be rewritten any further.

There are many programming paradigms that can be modelled by rewriting, such as for example functional and logic programming, equational reasoning, object-oriented and concurrent programming and theorem provers. Rewriting is also used in mathematical reasoning, specification of security protocols, resolution constraint methods, transition systems, natural language processing, operational semantics, algebraic specifications, program transformation, etc.

One of the simplest rewriting formalisms is the so called *first-order rewriting systems* (see for example [16, 15, 19, 7, 29]), where expressions are represented using algebras, and specifications of operations and properties can be done via a set of equations. In order to obtain a computation mechanism for the expressions of the algebra, the equations are oriented as rewrite rules.

There is also a classical rewriting formalism to represent the operations used to manipulate expressions, namely the λ-calculus, where functions come with rules to evaluate them. These functions are considered as ‘first-class citizens’: we can apply a function to another function (and in particular to itself), we can yield a function as the result of another function. The λ-calculus, which is a Turing complete language, turns out to be a model of computation for the functional programming paradigm.

Both formalisms, namely first-order rewriting and λ-calculus, present some interesting characteristics, which are in some sense complementary. On the one hand, λ-calculus is a natural model for functionality, but it is often not well-adapted to deal efficiently with data structures such as integers,

Vol. 15 No. 6, © The Author, 2005. Published by Oxford University Press. All rights reserved.
For Permissions, please email: journals.permissions@oxfordjournals.org
doi:10.1093/logcom/exi051

lists or trees. In practice, even if it is known how to *encode* algebraic data types into the λ-calculus, it is preferable to introduce primitive algebraic structures into the λ-calculus [9]. On the other hand, even if some equalities can be encoded into the typed λ-calculus (such as for example those which are necessary to define addition on integers), there are some other equalities that cannot (such as for example the *surjective pairing* [2] axiom). Therefore, there is a concrete reason to enrich the expressive power of a system like the typed λ-calculus by adding either richer typing systems and/or algebraic rewrite rules.

Last but not least, first-order rewriting is not well-adapted to deal with functions as 'first-class citizens', that is, to write specifications where functions may be used as parameters as well as results. A typical example is given by a program like `map`.[1]

All these characteristics together contribute to the foundations of *higher-order rewriting systems (HORS)*. These systems combine within a unique syntax the formalisms from proof theory, such as the typed λ-calculus, with formalisms arising in algebraic specifications, such as the first-order rewriting systems. We thus obtain a language which is able to model at the same time the notion of *computation* and that of *proof*.

Many higher-order rewriting systems exist and work in the area is currently very active. This research domain may be seen to start with the pioneering work of J-W. Klop in his 1980 PhD thesis [18], where *Combinatory Reduction Systems* (CRS) were introduced. Several formalisms introduced later, of which we mention some, are: Z. Khasidashvili's *Expression Reduction Systems* (ERS) of which an early reference is [17], T. Nipkow introduces *Higher-Order Rewriting Systems* (HRS) in [22], D. Wolfram defines *Higher-Order Term Rewriting Systems* [34], V. van Oostrom and F. van Raamsdonk introduce *Higher-Order Term Rewriting Systems* [32] as a general higher-order rewriting formalism encompassing many known formalisms [31, 33] and B. Pagano defines *Explicit Reduction Systems* (XRS) [23] using de Bruijn notation. van Raamsdonk's PhD thesis provides a survey [33].

Higher-order (term) rewriting concerns the transformation of terms in the presence of binding mechanisms for variables and substitution. In order to explain the technical problems encountered when working with these systems, let us take as an example the λ-calculus, where terms are either *variables* (denoted by symbols $x, y, z, \ldots$), or *applications* having the form (MN) representing the application of the function M to an argument N, or λ-*abstractions* having the form $\lambda x.M$ representing a function with argument x and body M. The transformation rule is the β-rewrite rule, which represents the result of applying a function to an argument.

$$(\lambda x.M)N \to_\beta M[x \leftarrow N].$$

The right-hand side of this rule makes use of a special symbol to denote the *substitution* operation: $M[x \leftarrow N]$ denotes the term which results from substituting N for all free occurrences of x in M. Substitution is a metalevel notion (it lives in the world of our language of discourse) that may be seen as a consequence of the existence of special symbols called *binder symbols* that have the power to *bind* variables in terms. This entails that substitution may not be confined to usual first-order replacement, but rather has to be careful to respect the status (free or bound) of variables when doing its work. In this sense, it is fair to say that substitution is 'respectful replacement' and, as a consequence, it is a mistake to dismiss substitution as a trivial concept: the theory of higher-order rewriting is considerably more involved than that of first-order rewriting. In particular, α-conversion is needed to guarantee that substitution is a correct operation; and this comes with a

[1]The program `map` can be specified by the equations `map(f,[]) = []` and `map(f,x::l) = f(x)::map(f,l)`, where `f` represents any function, `[]` represents an empty list, and `x::l` represents a non-empty list having a first element `x` followed by a sublist `l`.

cost since automatic renaming of bound variables turns out to be an expensive operation in terms of consumption of computational resources such as memory and processing time.

This paper aims at getting rid of α-conversion in the substitution process. Although from the metalevel the execution of a substitution is atomic, the cost of computing it strongly depends on the form of the terms, especially if unwanted variable capture conflicts must be avoided by renaming bound variables. So this aim has a practical interest since any implementation of higher-order rewriting must include instructions for computing this notion of substitution. As illustrated in Section 4, there is a standard technique introduced by de Bruijn to get rid of α-conversion. De Bruijn indices take care of renaming because the representation of variables by indices completely eliminates the capture of variables. However, de Bruijn formalisms have only been studied for particular systems (and only on the term level) and no general framework of higher-order rewriting with indices has been proposed. We address this problem here by focusing not only on de Bruijn terms (as usually done in the literature for λ-calculus [21]) but also on de Bruijn metaterms, which are the syntactical objects used to express the rules of any general higher-order rewrite system formulated in a de Bruijn context. More precisely, we shall introduce a de Bruijn notation for Expression Reduction Systems, obtaining $SERS_{dB}$. In fact, we shall formulate a slightly simplified version of ERS that we shall call Simplified ERS ($SERS$), better suited for our purposes, and then consider a de Bruijn notation for this formalism. The reason for choosing the ERS formalism is that its syntax is close to the 'usual' presentation (see for example [2, 13]) of the λ-calculus. Thus, for example, the β-rewrite rule is written in the ERS formalism as $app(\lambda x.M, N) \rightarrow M[x \leftarrow N]$ where the higher-order metavariables M and N can be instantiated by any terms, while it is written in the CRS formalism as $app(lam([x].M(x)), N) \rightarrow M(N)$ where the notion of valuation for higher-order variables is a more involved operation.

Our work may be viewed as an *interface* of a programming language based on higher-order rewriting. Since the use of variable names based formalisms are necessary for humans to interact with computers in a user-friendly way, technical resources like de Bruijn indices and explicit substitutions should live behind the scene, in other words, should be implementation concerns. Moreover, it is required of whatever is behind the scene to be as faithful as possible as regards the formalism it is implementing. So a key issue is the detailed study of the relationship between $SERS$ and $SERS_{dB}$. The definitions developed in Sections 4 and 5 give formal translations from higher-order syntax with names to higher-order syntax with indices and vice versa. These translations are extensions to the higher-order setting of the translations presented in [10], also studied in [21].

We begin this paper by introducing our work and study scenario, the $SERS$ formalism. After defining notions such as pre-metaterms, metaterms and terms and their corresponding notions of substitution, we consider rewrite rules. Valuations are then introduced in order to put rewrite rules to work. Metaterms are used to specify rewrite rules, and valuations are used to instantiate them in order to rewrite terms.

The de Bruijn based formalism $SERS_{dB}$ is defined in Section 3, and analogous concepts are considered in that setting. The key idea of our de Bruijn notation for metaterms is to associate labels to metavariables in order to denote *binding contexts*. Thus for example, the metaterm with names $f(\lambda\alpha.(app(X, \alpha)), X)$ will be translated as the de Bruijn metaterm $f(\lambda(app(X_\alpha, 1)), X_\epsilon)$, where X_ϵ denotes a metavariable X, which appears in an empty binding context, and X_α denotes a metavariable X, which appears inside a binding context with a single variable, namely α. This notation turns out to be a natural tool to write higher-order rewrite rules in a de Bruijn context. Simple examples of such a fact, which we shall consider in this paper, are the η_{dB} and the C_{dB} rewrite rules, obtained from their respective name versions η and C:[2]

[2]The rewrite rule C expresses that the formula appearing as the first argument of the $imply$ function symbol implies the

$$
\begin{array}{lll@{\qquad}lll}
\lambda\alpha.(app(X,\alpha)) & \to_{\eta} & X & \lambda(app(X_\alpha,1)) & \to_{\eta_{dB}} & X_\epsilon \\
imply(\exists\alpha.\forall\beta.Y,\forall\beta.\exists\alpha.Y) & \to_{C} & true & imply(\exists\forall Y_{\beta\alpha},\forall\exists Y_{\alpha\beta}) & \to_{C_{dB}} & true
\end{array}
$$

Note that if the rightmost X_ϵ of the rule η_{dB} is instantiated by a term a, then the leftmost X_α must be instantiated by another term, say a', which is obtained by incrementing by one all the free indices in a. On the other hand, if the rightmost $Y_{\alpha\beta}$ of the rule C_{dB} is instantiated by a term b, then the leftmost $Y_{\beta\alpha}$ must be instantiated by a term, say b', which is obtained by interchanging all 1-level and 2-level indices in b. Indeed, the label $\alpha\beta$, when compared to the label $\beta\alpha$ may be seen as denoting a permutation of indices.

We undertake the task of carefully comparing the $SERS$ and $SERS_{dB}$ formalisms: Section 4 studies an encoding of $SERS$ into $SERS_{dB}$ and Section 5 considers the opposite encoding. In each case, this requires dealing with a static phase by showing how terms and rewrite rules may be encoded, and a rewrite preservation or dynamical phase in which we must show that valuations, and hence the induced rewrite relation, may also be encoded appropriately. The $SERS_{dB}$-to-$SERS$ direction shall prove to be technically more demanding than the other. The reason is that we have a choice for selecting appropriate names for variables *and* metavariables, and we must rest assured that the results are not biased by our selection.

The work presented in this paper is further extended in [6] by showing how to encode all $SERS_{dB}$ as first-order rewriting systems with the aid of explicit substitutions. As a consequence, we obtain a complete translation from higher-order rewriting within a formalism with names to first-order rewriting modulo an equational theory.

This paper is an extended version of the extended abstract published as [4].

Concerning related work, it is worth noticing that other approaches to higher-order formalisms use Nominal Logic or Higher-order Abstract Syntax. The first one is a version of many-sorted first-order logic with equality containing primitives for renaming via name-swapping, for freshness of names and for name-binding. The logic makes use of the Fraenkel-Mostowski (FM) permutation model of set theory. Some of the more representative works along this line can be found in [14, 25, 30].

Higher-order Abstract Syntax (HOAS) incorporates name binding information in a uniform and language generic way. This is done by representing object-level variables by variables in a meta-language based on typed λ-calculus. The consequence of this representation is that renaming and substitution is pushed out to the meta-level and that their properties are established once and for all. Some proposals using HOAS are [24, 11].

2 Simplified expression reduction systems

This section introduces the name based higher-order rewrite formalism $SERS$. The latter is an appropriate simplification of Khasidashvili's ERS [17] which consists in restricting binders to those which bind one variable and restricting substitution to simple substitution (in contrast to simultaneous or parallel substitution).

DEFINITION 2.1 (Signature)
A $SERS$-signature Σ consists of the following denumerable and disjoint sets.

- A set $\mathcal{V}$ of *variables* denoted $x, y, \ldots$
- A set of *pre-bound o-metavariables* (o for object) denoted $\alpha, \beta, \ldots$

one in the second argument.

- A set of *pre-free o-metavariables* denoted $\widehat{\alpha}, \widehat{\beta}, \ldots$
- A set of *t-metavariables* (t for term), denoted $X, Y, Z, \ldots$
- A set $\mathcal{F}$ of *function symbols* equipped with a fixed (possibly zero) arity, denoted $f, g, h, \ldots$
- A set $\mathcal{B}$ of *binder symbols* equipped with a fixed (non-zero) arity, denoted $\lambda, \mu, \nu, \xi, \ldots$

The o-metavariables of the signature is the union of pre-bound and pre-free o-metavariables. When speaking of metavariables without further qualifiers we refer to o- and t-metavariables. Since all these alphabets are ordered, given any symbol s we denote $\mathcal{O}(s)$ its position in the corresponding alphabet.

DEFINITION 2.2 (Pre-metaterms)
The set of $SERS$ *pre-metaterms* over Σ, denoted PMT, is defined by:

$$M \quad ::= \quad \alpha \mid \widehat{\alpha} \mid X \mid f(M, \ldots, M) \mid \xi\alpha.(M, \ldots, M) \mid M[\alpha \leftarrow M].$$

Arities are respected, i.e. a pre-metaterm $f(M_1, \ldots, M_n)$ (resp. $\xi\alpha.(M_1, \ldots, M_n)$) is generated by the grammar only if f (resp. ξ) has arity $n \geq 0$ (resp. $n > 0$).

We use $M, N, M_i, \ldots$ to denote pre-metaterms. The operator $\bullet[\bullet \leftarrow \bullet]$ in the pre-metaterm $M_1[\alpha \leftarrow M_2]$ is called the *metasubstitution operator*. The o-metavariable α in a pre-metaterm of the form $\xi\alpha.(M_1, \ldots, M_n)$ or $M_1[\alpha \leftarrow M_2]$ is referred to as the *formal parameter*. The set of binder symbols together with the metasubstitution operator are called *binder operators*, thus the metasubstitution operator is a binder operator (since it has binding power) but is *not* a binder symbol since it is not an element of $\mathcal{B}$.

$SERS$ and ERS differ in their treatment of substitution since in ERS binders and metasubstitutions are defined on *multiple* o-metavariables. Pre-metaterms like $\xi\alpha_1 \ldots \alpha_k.(M_1, \ldots, M_m)$ and $M[\alpha_1 \ldots \alpha_k \leftarrow M_1, \ldots, M_k]$ are possible in ERS, with the underlying assumption that $\alpha_1, \ldots, \alpha_k$ are all distinct and with the underlying semantics that $M[\alpha_1 \ldots \alpha_k \leftarrow M_1, \ldots, M_k]$ denotes the usual (multiple) substitution. It is well known that multiple substitution can be encoded by simple substitution. Indeed, $M[\alpha_1 \ldots \alpha_k \leftarrow M_1, \ldots, M_k]$ can be encoded as the pre-metaterm $M[\alpha_1 \leftarrow \beta_1][\alpha_2 \leftarrow \beta_2] \ldots [\alpha_k \leftarrow \beta_k][\beta_1 \leftarrow M_1][\beta_2 \leftarrow M_2] \ldots [\beta_k \leftarrow M_k]$, where $\beta_1, \ldots, \beta_k$ are *fresh* pre-bound o-metavariables. As for $\xi\alpha_1 \ldots \alpha_k.(M_1, \ldots, M_m)$ it may be encoded with the help of two binder symbols ξ and ξ' of arity 1 and m respectively, obtaining $\xi\alpha_1.(\xi\alpha_2.(\ldots \xi'\alpha_k.(M_1, \ldots, M_m)))$. There is also a notion of scope indicator in ERS, used to express in which arguments of the binder variables are bound. Scope indicators are not considered in $SERS$ since they do not seem to contribute to the expressive power of ERS.

We sometimes identify a pre-metaterm with its associated *tree*:

- The tree of a metavariable α, $\widehat{\alpha}$ or X is the tree with the single node α, $\widehat{\alpha}$ or X, respectively.
- If $T_1, \ldots, T_n$ are the trees of $M_1, \ldots, M_n$, respectively, then the tree of $f(M_1, \ldots, M_n)$ is that of Figure 1(a).
- If $T_1, \ldots, T_n$ are the trees of $M_1, \ldots, M_n$, respectively, then the tree of $\xi\alpha.(M_1, \ldots, M_n)$ is that of Figure 1(b).
- If T_1, T_2 are the trees of M_1, M_2, respectively, then the tree of $M_1[\alpha \leftarrow M_2]$ is that of Figure 1(c).

The tree of $f(M_1, \ldots, M_n)$ has the expected form, however the tree of $M_1[\alpha \leftarrow M_2]$ may seem somewhat odd since there are two nodes above the tree of M_1. The reason is that the metasubstitution operator is asymmetric in that its left argument M_1 is considered to be under a binding effect whereas M_2 is not. We would like this to be reflected in the structure of the tree, enabling us to look 'above' a position in a tree to know under which binders it occurs.

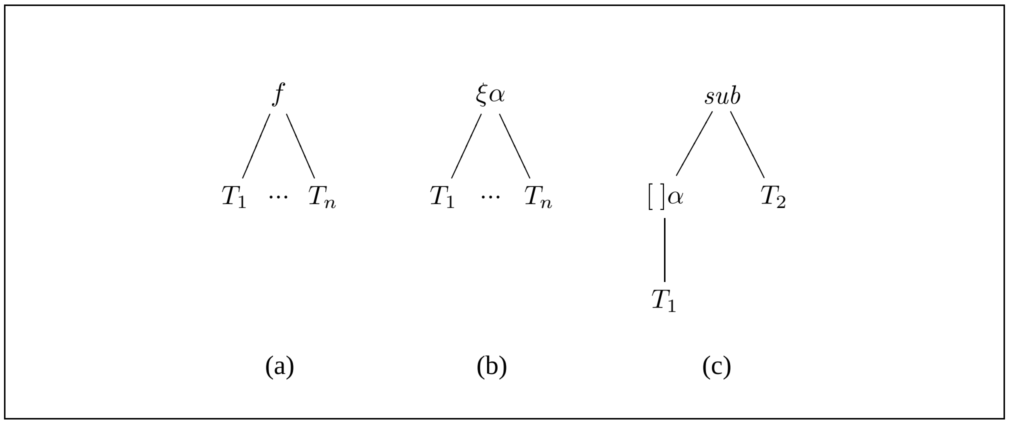

FIGURE 1. Pre-metaterms as trees

A position is a word over the alphabet $\mathbb{N}$ of natural numbers; we use ϵ to denote the empty word. Given a pre-metaterm M and a position p, we define the *subterm of M at position p* as follows:

$$\begin{array}{lcl} M|_\epsilon & = & M \\ f(M_1,\ldots,M_n)|_{i.p} & = & M_i|_p, \text{ for } i \in \{1\ldots n\} \\ \xi\alpha.(M_1,\ldots,M_n)|_{i.p} & = & M_i|_p, \text{ for } i \in \{1\ldots n\} \\ M[\alpha \leftarrow N]|_{1.1.p} & = & M|_p \\ M[\alpha \leftarrow N]|_{2.p} & = & N|_p. \end{array}$$

The set of *occurrences of N in M* is the set containing all the positions p of M such that $M|_p = N$. We say that N occurs at position p in M iff $M|_p = N$. The *parameter path* of a position p in M is the list containing all the (pre-bound) o-metavariables occurring in the path from p to the root of M, in that order. Likewise, we may define the parameter path of an occurrence of N in M.

EXAMPLE 2.3
If $M = f(\xi\alpha.X, Y)$, then X occurs at position 1.1 and Y at position 2. The parameter path of 1.1 (or just X) is α and the parameter path of 2 (or just Y) is ϵ. If $M = \mu\beta.(X[\alpha \leftarrow \lambda\gamma.(g(\beta, g(\gamma, Z)))])$, then the pre-metaterm $\lambda\gamma.(g(\beta, g(\gamma, Z)))$ occurs at position 1.2 and $g(\gamma, Z)$ occurs at position $1.2.1.2$; the parameter path of 1.2 (or just $\lambda\gamma.(g(\beta, g(\gamma, Z))))$ is β, the parameter path of $1.2.1.2$ is $\gamma\beta$.

The following definitions introduce the set of *metaterms*. Metaterms are pre-metaterms that are *well-formed* in the sense that they prevent the use of the same name for two different occurrences of a formal parameter appearing in the parameter path of a given pre-metaterm. In other words, all the formal parameters appearing in the parameter path of a pre-metaterm must be different. Also, it guarantees that pre-bound o-metavariables only occur bound.

DEFINITION 2.4 (Labels)
A *label* is a finite sequence of symbols of an alphabet. A *simple* label is a label without repeated symbols. We use $k, l, l_i, \ldots$ to denote arbitrary labels and ϵ for the empty label. If s is a symbol and

l is a label then the notation $s \in l$ means that the symbol s *appears in* the label l, and also, we use sl to denote the new label whose head is s and whose tail is l. Other notations are $|l|$ for the *length* of l (number of symbols in l) and $\mathtt{at}(l,n)$ for the nth element of l assuming $n \leq |l|$. Also, if s occurs (at least once) in l then $\mathtt{pos}(s,l)$ denotes the position of the *first occurrence* of s in l. If θ is a function defined on the alphabet of a label $l = s_1 \dots s_n$, then $\theta(l)$ denotes the label $\theta(s_1)\dots\theta(s_n)$. We may use a label as a set (e.g. if S is a set then $S \cap l$ denotes the intersection of S with the underlying set determined by l) if no confusion arises.

DEFINITION 2.5 (Metaterms)
A pre-metaterm $M \in \mathsf{PMT}$ over Σ is said to be a *metaterm* over Σ iff the predicate $\mathcal{WF}(M)$ holds, where $\mathcal{WF}(M)$ iff $\mathcal{WF}_\epsilon(M)$ holds, and $\mathcal{WF}_l(M)$ is defined by induction on the structure of the pre-metaterm M for any label l as follows:

- $\mathcal{WF}_l(\alpha)$ iff $\alpha \in l$
- $\mathcal{WF}_l(\widehat{\alpha})$ and $\mathcal{WF}_l(X)$ hold iff l is a *simple* label
- $\mathcal{WF}_l(f(M_1,\dots,M_n))$ iff for all $1 \leq i \leq n$ we have $\mathcal{WF}_l(M_i)$
- $\mathcal{WF}_l(\xi\alpha.(M_1,\dots,M_n))$ iff $\alpha \notin l$ and for all $1 \leq i \leq n$ we have $\mathcal{WF}_{\alpha l}(M_i)$
- $\mathcal{WF}_l(M_1[\alpha \leftarrow M_2])$ iff $\alpha \notin l$ and $\mathcal{WF}_l(M_2)$ and $\mathcal{WF}_{\alpha l}(M_1)$.

EXAMPLE 2.6
The pre-metaterms $f(\xi\alpha.X, \lambda\alpha.Y)$, $f(\widehat{\beta}, \lambda\alpha.Y)$ and $g(\lambda\alpha.(\xi\beta.c))$ are metaterms, however the pre-metaterms $f(\alpha, \xi\alpha.X)$ and $f(\widehat{\beta}, \lambda\alpha.(\xi\alpha.X))$ are not.

In the sequel, pre-bound (resp. pre-free) o-metavariables occurring in metaterms are simply referred to as bound (resp. free) o-metavariables. Also, we assume whenever possible, some fixed signature Σ and hence speak of pre-metaterms or metaterms instead of pre-metaterms over Σ or metaterms over Σ. As we shall see, metaterms are used to specify rewrite rules.

DEFINITION 2.7 (Free metavariables of pre-metaterms)
If M is a pre-metaterm, then $\mathrm{FMV}\textsc{ar}(M)$ denotes the *set of free metavariables* of M, which is defined as follows:

$$\begin{array}{lll}
\mathrm{FMV}\textsc{ar}(X) \overset{\text{def}}{=} \{X\} & \mathrm{FMV}\textsc{ar}(\alpha) \overset{\text{def}}{=} \{\alpha\} & \mathrm{FMV}\textsc{ar}(\widehat{\alpha}) \overset{\text{def}}{=} \{\widehat{\alpha}\} \\
\mathrm{FMV}\textsc{ar}(f(M_1,\dots,M_n)) & \overset{\text{def}}{=} \bigcup_{i=1}^{n} \mathrm{FMV}\textsc{ar}(M_i) & \\
\mathrm{FMV}\textsc{ar}(\xi\alpha.(M_1,\dots,M_n)) & \overset{\text{def}}{=} (\bigcup_{i=1}^{n} \mathrm{FMV}\textsc{ar}(M_i)) \setminus \{\alpha\} & \\
\mathrm{FMV}\textsc{ar}(M_1[\alpha \leftarrow M_2]) & \overset{\text{def}}{=} (\mathrm{FMV}\textsc{ar}(M_1) \setminus \{\alpha\}) \cup \mathrm{FMV}\textsc{ar}(M_2). &
\end{array}$$

The set of *bound* metavariables of a pre-metaterm M, written $\mathrm{BMV}\textsc{ar}(M)$, is defined as expected. Note that only pre-bound o-metavariables may occur bound in a metaterm, metavariables of the form $\widehat{\alpha}$ or X_i always occur free (if they occur at all) in a metaterm. We denote the set of *all* the metavariables of a metaterm or a pre-metaterm M by $\mathrm{MV}\textsc{ar}(M)$. So we have $\mathrm{MV}\textsc{ar}(M) = \mathrm{FMV}\textsc{ar}(M) \cup \mathrm{BMV}\textsc{ar}(M)$.

EXAMPLE 2.8
Let M be the metaterm $f(\widehat{\beta}, \lambda\alpha.Y)$. Then $\mathrm{FMV}\textsc{ar}(M) = \{\widehat{\beta}, Y\}$, $\mathrm{BMV}\textsc{ar}(M) = \{\alpha\}$, and $\mathrm{MV}\textsc{ar}(M) = \{\widehat{\beta}, Y, \alpha\}$. If M is the metaterm $f(\widehat{\beta}, \lambda\alpha.\alpha)$ then $\mathrm{FMV}\textsc{ar}(M) = \{\widehat{\beta}\}$, $\mathrm{BMV}\textsc{ar}(M) = \{\alpha\}$ and $\mathrm{MV}\textsc{ar}(M) = \{\alpha, \widehat{\beta}\}$.

DEFINITION 2.9 (Terms and contexts)
The set of *SERS terms* over Σ, denoted T, and *contexts* are defined by:

$$\begin{array}{llcl} \text{Terms} & t & ::= & x \mid f(t,\ldots,t) \mid \xi x.(t,\ldots,t) \\ \text{Contexts} & C & ::= & \Box \mid f(t,\ldots,C,\ldots,t) \mid \xi x.(t,\ldots,C,\ldots,t) \end{array}$$

where $\Box$ denotes a 'hole'. We use $s, t, t_i, \ldots$ for terms and C, D for contexts. Contexts are just terms with exactly one occurrence of a hole. The x in ξx is called a *binding variable*. We remark that in contrast to other formalisms dealing with higher-order rewriting such as CRS, the set of terms is not contained in the set of pre-metaterms since the set of variables and the set of o-metavariables are disjoint. Terms are obtained from metaterms by suitable instantiation of t-metavariables *and* o-metavariables.

With $C[t]$ we denote the term obtained by replacing t for the hole $\Box$ in the context C. Note that this operation may introduce variable capture. The notion of parameter path makes also sense for contexts, where it is defined as expected and where the only occurrence considered is that of the hole. Thus, we define the *parameter path* of a context as the list containing all the variables occurring in the path from the hole $\Box$ to the root of the context. For example, the parameter path of the context $f(\lambda x.(z, \xi y.(h(y, \Box))))$ is the sequence yx.

The set of free and bound variables of terms and contexts are defined as expected. We write $\text{FV}(t)$ and $\text{BV}(t)$ for the set of free and bound variables, respectively, of the term t. Similar notation is used for the free and bound variables of a context. Substitution on terms can be defined as follows:

DEFINITION 2.10 ((Restricted) substitution on terms)
The *(restricted) substitution* of a term t for a variable x in a term s, denoted $s\{x \leftarrow t\}$, is defined:

$$\begin{array}{lll} x\{x \leftarrow t\} & \stackrel{\text{def}}{=} & t \\ y\{x \leftarrow t\} & \stackrel{\text{def}}{=} & y, \quad \text{if } x \neq y \\ f(s_1,\ldots,s_n)\{x \leftarrow t\} & \stackrel{\text{def}}{=} & f(s_1\{x \leftarrow t\},\ldots,s_n\{x \leftarrow t\}) \\ \xi y.(s_1,\ldots,s_n)\{x \leftarrow t\} & \stackrel{\text{def}}{=} & \xi y.(s_1\{x \leftarrow t\},\ldots,s_n\{x \leftarrow t\}) \\ & & \text{if } x \neq y, \text{ and } (y \notin \text{FV}(t) \text{ or } x \notin \bigcup_i \text{FV}(s_i)). \end{array}$$

Note that this notion of substitution does *not* appeal to α-conversion (renaming of bound variables as defined below) in order to avoid variable capture. Therefore this notion of restricted substitution is not defined for all terms (hence its name). For example $(\xi y.x)\{x \leftarrow y\}$ is not defined and neither is $(\xi x.z)\{x \leftarrow y\}$. When defining the rewrite relation on terms induced by rewrite rules we take α-conversion into consideration in order to guarantee that any substitution to be performed may be completed with restricted substitution. This allows us to 'localize' α-conversion when applying rewrite rules.

α-conversion on terms is the smallest reflexive, symmetric and transitive relation closed by contexts verifying the following equality:

$$(\alpha) \quad \xi x.(s_1,\ldots,s_n) \quad =_\alpha \quad \xi y.(s_1\{x \leftarrow y\},\ldots,s_n\{x \leftarrow y\})$$

where $x \neq y$ and y is fresh in $\xi x.(s_1,\ldots,s_n)$.

Note that since y does not occur in $s_1,\ldots,s_n$ substitution is always defined. We use $s =_\alpha t$ to say that the terms s and t are α-convertible. This conversion is sound in the sense that $s =_\alpha t$ implies $\text{FV}(s) = \text{FV}(t)$.

The notion of α-conversion for terms has a corresponding one for pre-metaterms which we call *v-equivalence* (v for variant). However, this requires first introducing restricted substitution for pre-metaterms.

DEFINITION 2.11 ((Restricted) substitution on pre-metaterms)
The (restricted) substitution of a pre-metaterm Q for an o-metavariable α in a pre-metaterm P, denoted $P \ll\alpha\leftarrow Q\gg$, is defined as:

$$\begin{array}{ll}
\alpha \ll\alpha\leftarrow Q\gg & \stackrel{\text{def}}{=} Q \\
\beta \ll\alpha\leftarrow Q\gg & \stackrel{\text{def}}{=} \beta, \text{ if } \alpha \neq \beta \\
\widehat{\beta} \ll\alpha\leftarrow Q\gg & \stackrel{\text{def}}{=} \widehat{\beta} \\
X \ll\alpha\leftarrow Q\gg & \stackrel{\text{def}}{=} X \\
f(M_1, \ldots, M_n) \ll\alpha\leftarrow Q\gg & \stackrel{\text{def}}{=} f(M_1 \ll\alpha\leftarrow Q\gg, \ldots, M_n \ll\alpha\leftarrow Q\gg) \\
(\xi\beta.(M_1, \ldots, M_n)) \ll\alpha\leftarrow Q\gg & \stackrel{\text{def}}{=} \xi\beta.(M_1 \ll\alpha\leftarrow Q\gg, \ldots, M_n \ll\alpha\leftarrow Q\gg) \\
& \quad \text{if } \alpha \neq \beta \text{ and } (\beta \notin \text{FMVAR}(Q) \text{ or } \alpha \notin \bigcup_{i=1}^{n} \text{FMVAR}(M_i)) \\
(M_1[\beta \leftarrow M_2]) \ll\alpha\leftarrow Q\gg & \stackrel{\text{def}}{=} (M_1 \ll\alpha\leftarrow Q\gg)[\beta \leftarrow M_2 \ll\alpha\leftarrow Q\gg] \\
& \quad \text{if } \alpha \neq \beta \text{ and } (\beta \notin \text{FMVAR}(Q) \text{ or } \alpha \notin \text{FMVAR}(M_1)).
\end{array}$$

The intuitive meaning of two v-equivalent pre-metaterms is that they are able to receive the *same* set of potential 'valuations' (Definition 2.19). Thus for example, as one would expect, $\lambda\alpha.X \neq_v \lambda\beta.X$ because when α and X are replaced by x, and β is replaced by y, one obtains $\lambda x.x$ and $\lambda y.x$, which are not α-convertible. However, since pre-metaterms contain t-metavariables, the notion of v-equivalence is not straightforward as the notion of α-conversion in the case of terms. More on the intuitive idea of v-equivalence will be said below.

DEFINITION 2.12 (v-equivalence for pre-metaterms)
Given pre-metaterms M and N, we say that M is *v-equivalent* to N, iff $M =_v N$ where $=_v$ is the smallest reflexive, symmetric and transitive relation closed by metacontexts[3] verifying:

$$\begin{array}{llll}
(v1) & \xi\alpha.(P_1, \ldots, P_n) & =_v & \xi\beta.(P_1 \ll\alpha\leftarrow\beta\gg \ldots P_n \ll\alpha\leftarrow\beta\gg) \\
(v2) & P_1[\alpha \leftarrow P_0] & =_v & P_1 \ll\alpha\leftarrow\beta\gg [\beta \leftarrow P_0]
\end{array}$$

where P_i, for $1 \leq i \leq n$, does not contain t-metavariables for $1 \leq i \leq n$, and

$(v1)$ α, β are pre-bound o-metavariable s.t. $\alpha \neq \beta$ and β does not occur in $P_1, \ldots, P_n$, and
$(v2)$ α, β are pre-bound o-metavariable s.t. $\alpha \neq \beta$ and β does not occur in P_1.

EXAMPLE 2.13
$\lambda\alpha.\alpha =_v \lambda\beta.\beta$, $\lambda\alpha.f =_v \lambda\beta.f$, but $\lambda\alpha.X \neq_v \lambda\beta.X$, and $\lambda\beta.\lambda\alpha.X \neq_v \lambda\alpha.\lambda\beta.X$.

Note that pre-metaterms may be seen as contexts where the holes of a context are represented by t-metavariables. However, metaterms are not treated as first class citizens as in [3].

We now address the rewrite rules of a *SERS*. The rewrite rules are specified using metaterms, whereas the rewrite relation is defined on terms.

DEFINITION 2.14 (*SERS*)
A *SERS-rewrite rule* over Σ is a pair of metaterms (G, D) over Σ (also written $G \rightarrow D$) such that:

- the first symbol (called *head* symbol) in G is a function symbol or a binder symbol,
- $\text{FMVAR}(D) \subseteq \text{FMVAR}(G)$, and
- G contains no occurrence of the metasubstitution operator.

Finally, we define a *SERS* as a pair $(\Sigma, \mathcal{R})$ where Σ is a *SERS*-signature and $\mathcal{R}$ is a set of *SERS*-rewrite rules over Σ. We often omit Σ and write $\mathcal{R}$ instead of $(\Sigma, \mathcal{R})$, if no confusion arises.

[3]Metacontexts are defined analogously to contexts. The notion of 'parameter path of a context' is extended to metacontexts as expected.

EXAMPLE 2.15
The λ-calculus is defined by considering the signature containing the function symbols $\mathcal{F} = \{app\}$ and binder symbols $\mathcal{B} = \{\lambda\}$, together with the $SERS$-rewrite rule:

$$app(\lambda\alpha.X, Y) \rightarrow_{\beta} X[\alpha \leftarrow Y].$$

The $\lambda\eta$-calculus is obtained by adding the following $SERS$-rewrite rule: $\lambda\alpha.(app(X, \alpha)) \rightarrow_{\eta} X$.

EXAMPLE 2.16
The $\lambda\mathtt{x}$-calculus [8, 27] is defined by considering the signature containing the function symbols $\mathcal{F} = \{app, subs\}$ and binder symbols $\mathcal{B} = \{\lambda, \sigma\}$, together with the following $SERS$-rewrite rules:

$$\begin{array}{lll}
app(\lambda\alpha.X, Y) & \rightarrow_{Beta} & subs(\sigma\alpha.X, Y) \\
subs(\sigma\alpha.(app(X, Y)), Z) & \rightarrow_{App} & app(subs(\sigma\alpha.X, Z), subs(\sigma\alpha.Y, Z)) \\
subs(\sigma\alpha.\lambda\beta.(X), Z) & \rightarrow_{Lam} & \lambda\beta.(subs(\sigma\alpha.X, Z)) \\
subs(\sigma\alpha.\alpha, Z) & \rightarrow_{Var} & Z \\
subs(\sigma\alpha.\widehat{\beta}, Z) & \rightarrow_{rGc} & \widehat{\beta}.
\end{array}$$

EXAMPLE 2.17
The $\lambda\Delta$-calculus [28] is defined by considering the signature containing the function symbols $\mathcal{F} = \{app\}$ and binder symbols $\mathcal{B} = \{\lambda, \Delta\}$, together with the following $SERS$-rewrite rules:

$$\begin{array}{lll}
app(\lambda\alpha.X, Z) & \rightarrow_{Beta} & X[\alpha \leftarrow Z] \\
app(\Delta\alpha.X, Z) & \rightarrow_{\Delta 1} & \Delta\beta.(X[\alpha \leftarrow \lambda\gamma.(app(\beta, app(\gamma, Z)))]) \\
\Delta\alpha.(app(\alpha, X)) & \rightarrow_{\Delta 2} & X \\
\Delta\alpha.(app(\alpha, (\Delta\beta.(app(\alpha, X))))) & \rightarrow_{\Delta 3} & X.
\end{array}$$

EXAMPLE 2.18
A further example is the system for foldl recursion scheme over lists, containing the function symbols $\mathcal{F} = \{foldl, nil, cons\}$, the binder symbol $\mathcal{B} = \{\xi\}$ and the rewrite rules:

$$\begin{array}{lll}
foldl(\xi\alpha.(\xi\beta.X), Y, nil) & \rightarrow_{f1} & Y \\
foldl(\xi\alpha.(\xi\beta.X), Y, cons(Z, W)) & \rightarrow_{f2} & foldl(\xi\alpha.(\xi\beta.X), X[\alpha \leftarrow Y][\beta \leftarrow Z], W).
\end{array}$$

We now proceed to define the way in which rewrite rules are instantiated in order to obtain the induced rewrite relation on terms. This implies defining how the ‘holes’ in the metaterms of the rule, represented by t-metavariables and o-metavariables, are replaced by terms and variables, respectively. Thus *valuations* are introduced followed by some additional conditions imposed on these valuations in order to single out the ‘good’ valuations (referred to as *admissible valuations*) from the ‘bad’ ones.

DEFINITION 2.19 (Valuation)
A *variable assignment* is a (partial) function θ_v from o-metavariables to variables[4] with finite domain, such that for every pair of o-metavariables $\alpha, \widehat{\beta}$ we have $\theta_v\alpha \neq \theta_v\widehat{\beta}$ (pre-bound and pre-free o-metavariables are assigned different variables). A *term assignment* is a (partial) function θ_t from t-metavariables to terms with finite domain.

A *valuation* θ over Σ is a pair of (partial) functions (θ_v, θ_t) where θ_v is a variable assignment and θ_t is a term assignment. It defines a function on metavariables, also denoted θ, as expected:

$$\begin{array}{lll}
\theta\alpha & \stackrel{\text{def}}{=} & \theta_v\alpha \\
\theta\widehat{\alpha} & \stackrel{\text{def}}{=} & \theta_v\widehat{\alpha} \\
\theta X & \stackrel{\text{def}}{=} & \theta_t X.
\end{array}$$

[4]We write indistinctly $\theta_v(\alpha)$ or $\theta_v\alpha$ to denote application of θ_v to an o-metavariable α.

A valuation θ may be extended in a unique way to the set of pre-metaterms M such that $\text{MV}\textsc{ar}(M) \subseteq Dom(\theta)$, where $Dom(\theta)$ denotes the domain of θ, as follows:

$$\begin{array}{lll} \overline{\theta}(f(M_1,\ldots,M_n)) & \stackrel{\text{def}}{=} & f(\overline{\theta}M_1,\ldots,\overline{\theta}M_n) \\ \overline{\theta}(\xi\alpha.(M_1,\ldots,M_n)) & \stackrel{\text{def}}{=} & \xi\theta_v\alpha.(\overline{\theta}M_1,\ldots,\overline{\theta}M_n) \\ \overline{\theta}(M_1[\alpha \leftarrow M_2]) & \stackrel{\text{def}}{=} & \overline{\theta}(M_1)\{\theta_v\alpha \leftarrow \overline{\theta}M_2\}. \end{array}$$

We shall not distinguish between θ and $\overline{\theta}$ if no ambiguities arise. Also, we sometimes write $\theta(M)$ thereby implicitly assuming that $\text{MV}\textsc{ar}(M) \subseteq Dom(\theta)$.

Returning to the intuition behind v-*equivalence* the idea is that it can be translated into α-conversion in the sense that $M =_v N$ implies $\theta M =_\alpha \theta N$ for any valuation θ such that θM and θN are defined. Indeed, coming back to Example 2.13 and taking $\theta = \{\alpha/x, \beta/y, X/x\}$, we have $\theta(\lambda\alpha.\alpha) = \lambda x.x =_\alpha \lambda y.y = \theta(\lambda\beta.\beta)$ and $\theta(\lambda\alpha.f) = \lambda x.f =_\alpha \lambda y.f = \theta(\lambda\beta.f)$. However $\theta(\lambda\alpha.X) = \lambda x.x \neq_\alpha \lambda y.x = \theta(\lambda\beta.X)$ and $\theta(\lambda\beta.\lambda\alpha.X) = \lambda y.\lambda x.x \neq_\alpha \lambda x.\lambda y.x = \theta(\lambda\alpha.\lambda\beta.X)$.

As the reader may have observed, a valuation computes a metasubstitution operator by executing metalevel substitution. However, since metalevel substitution is restricted in that no α-conversion is allowed to take place, we must require the valuation to be capable of executing all metasubstitution operators in a given pre-metaterm.

DEFINITION 2.20 (Safe valuations)
Let M be a pre-metaterm over Σ and θ a valuation over Σ. We say that θ *is safe for* M if $\text{MV}\textsc{ar}(M) \subseteq Dom(\theta)$ and $\overline{\theta}M$ is defined, i.e. the substitutions generated by the last clause of Definition 2.19 can be computed. Likewise, if (G, D) is a rewrite rule, we say that θ is *safe for* (G, D) if $\overline{\theta}D$ is defined and $\text{MV}\textsc{ar}(G) \subseteq Dom(\theta)$.

Note that if the notion of substitution we are dealing with were not restricted, then α-conversion could be required in order to apply a valuation to a pre-metaterm. Also, for any valuation θ and pre-metaterm M with $\text{MV}\textsc{ar}(M) \subseteq Dom(\theta)$ that contains no occurrences of the metasubstitution operator, it turns out that θ is safe for M. Thus, we only ask θ to be safe for D (not G) in the previous definition.

The following condition is the classical notion of *admissibility* used in higher-order rewriting [33] to avoid inconsistencies in rewrite steps. It runs under the name 'variable-capture-freeness' in the case of ERS [20] and aims at ruling out certain valuations which after instantiating a rewrite rule leave some free and bound occurrences of the same variable. An example is the rewrite rule $\lambda\alpha.X \rightarrow X$ and the valuation which assigns x to α and X. The resulting rewrite step is $\lambda x.x \rightarrow x$ which has an occurrence of x that is bound on the left and an occurrence of x that is free on the right.

DEFINITION 2.21 (Path condition for t-metavariables)
Let X be a t-metavariable. Consider all the occurrences $p_1, \ldots, p_n$ of X in (G, D), and their respective parameter paths $l_1, \ldots, l_n$ in the trees corresponding to G and D. A valuation θ verifies *the path condition* for X in (G, D) if for every $x \in \text{FV}(\theta X)$

- either: for all $1 \leq i \leq n$ we have $x \in \theta(l_i)$
- or: for all $1 \leq i \leq n$ we have $x \notin \theta(l_i)$.

This definition may be read as: one occurrence of $x \in \text{FV}(\theta X)$ with X in (G, D) is in the scope of some binding occurrence of x iff every occurrence of X in (G, D) is in the scope of a bound o-metavariable α with $\theta\alpha = x$. For example, consider the $SERS$ rule $\lambda\alpha.(\xi\beta.X) \rightarrow \xi\beta.X$ and the valuations $\theta_1 = \{\alpha/x, \beta/y, X/z\}$ and $\theta_2 = \{\alpha/x, \beta/y, X/x\}$. Then θ_1 verifies the path condition for X, but θ_2 does not since when instantiating the rewrite rule with θ_2 the variable x shall occur both bound (on the LHS) and free (on the RHS).

Note that our formalism allows us to specify the restricted garbage collection rule rGc of $\lambda\mathtt{x}$ (Example 2.16) as originally done in [27], while formalisms such as CRS force one to change this rule to a stronger one, namely Gc, written as $subs(\sigma\alpha.X, Z)\rightarrow_{Gc} X$, where the path condition (Definition 2.21) on valuations guarantees that if $\theta(X) = t$, then $\theta(\alpha)$ cannot be in $\mathrm{FV}(t)$.

We may then single out the 'good' valuations by the following notion of admissible valuations.

DEFINITION 2.22 (Admissible valuations)
A valuation θ over Σ is *admissible for a rewrite rule* (G, D) over Σ iff the following conditions hold:

- θ is safe for (G, D),
- if α and β occur in (G, D) with $\alpha \neq \beta$, then $\theta_v\alpha \neq \theta_v\beta$, and
- θ verifies the path condition for every t-metavariable in (G, D).

Note that an admissible valuation is safe by definition, but a safe valuation may not be admissible. As an example, consider the η-contraction rule $\lambda x.app(X, x)\rightarrow X$ if $x \notin \mathrm{FV}(X)$. As mentioned in Example 2.15, it can be expressed in the $SERS$ formalism as the rule $\lambda\alpha.app(X, \alpha)\rightarrow_\eta X$. All valuations are (trivially) safe for η since there is no metasubstitution operator on the RHS. However, the path condition could fail. Indeed, consider the valuation $\theta = \{\alpha/x, X/x\}$: although trivially safe, it is not admissible since the path condition is not verified: $x \in \theta(\alpha)$ but $x \notin \theta(\epsilon)$ (x occurs bound on the LHS and free on the RHS). In the case of the $SERS$ representation of η, the path condition is in charge of verifying the above-mentioned well-known condition on η-contraction, namely that the variable instantiated for α not occur free in the term instantiated for X.

Having defined rewrite rules and (admissible) valuations we find ourselves ready to present the rewrite relation induced on terms by a rewrite rule.

DEFINITION 2.23 (Rewriting relation)
Let $(\Sigma, \mathcal{R})$ be a $SERS$ and s, t terms over Σ. We say that s *$\mathcal{R}$-rewrites* or *$\mathcal{R}$-reduces to* t, written $s\rightarrow_{\mathcal{R}} t$, iff there exists a rewrite rule $(G, D) \in \mathcal{R}$, an admissible valuation θ for (G, D) and a context C such that $s =_\alpha C[\theta G]$ and $t =_\alpha C[\theta D]$.

We shall occasionally drop the subscript in the rewrite relation when it is clear from the context. As in first-order rewriting, rewriting does not create new variables.

LEMMA 2.24
Let $s, t \in \mathsf{T}$. If $s\rightarrow_{\mathcal{R}} t$, then $\mathrm{FV}(t) \subseteq \mathrm{FV}(s)$.

3 Simplified expression reduction systems with indices

This section introduces the de Bruijn indices based higher-order rewrite formalism $SERS_{dB}$. We follow Section 2 and introduce de Bruijn metaterms, de Bruijn terms, de Bruijn valuation, and finally, de Bruijn rewriting. In order to distinguish a concept defined for the $SERS$ formalism from its corresponding version (if it exists) in the $SERS_{dB}$ formalism we may prefix it using the qualifying term 'de Bruijn', e.g. 'de Bruijn metaterms'.

DEFINITION 3.1 (de Bruijn signature)
A $SERS_{dB}$ signature Σ consists of the following denumerable and disjoint sets.

- A set of *binder indicators* denoted $\alpha, \beta, \ldots$

- A set of *i-metavariables* (i for index) denoted $\widehat{\alpha}, \widehat{\beta}, \ldots$
- A set of *t-metavariables* (t for term), denoted $X_l, Y_l, Z_l, \ldots$, where l ranges over the set of labels built over binder indicators.
- A set $\mathcal{F}$ of *function symbols* equipped with a fixed (possibly zero) arity, denoted $f, g, h, \ldots$
- A set $\mathcal{B}$ of *binder symbols* equipped with a fixed (non-zero) arity, denoted $\lambda, \mu, \nu, \xi, \ldots$

We remark that the set of binder indicators is exactly the set of pre-bound o-metavariables introduced in Definition 2.1. The reason for using the same alphabet in both formalisms will become clear in Section 4, but intuitively, we need a mechanism to annotate binding paths in the de Bruijn setting to distinguish metaterms like $\xi\beta.(\xi\alpha.X)$ and $\xi\alpha.(\xi\beta.X)$ appearing in the same rule when translated into a $SERS_{dB}$ system.

DEFINITION 3.2 (de Bruijn pre-metaterms)
The set of *de Bruijn pre-metaterms* over the $SERS_{dB}$ signature Σ, denoted PMT_{dB}, is defined by the following two-sorted grammar:

$$\begin{array}{llll} \text{metaindices} & I & ::= & 1 \mid \mathtt{S}(I) \mid \widehat{\alpha} \\ \text{pre-metaterms} & A & ::= & I \mid X_l \mid f(A, \ldots, A) \mid \xi(A, \ldots, A) \mid A[A]. \end{array}$$

The operator $\bullet[\bullet]$ in a pre-metaterm $A[A]$ is called the *de Bruijn metasubstitution operator*. The binder symbols together with the de Bruijn metasubstitution operator are called *binder operators*. Thus the de Bruijn metasubstitution operator is a binder operator (since it has binding power) but is *not* a binder symbol since it is not an element of $\mathcal{B}$.

We use $A, B, A_i, \ldots$ to denote de Bruijn pre-metaterms and the convention that $\mathtt{S}^0(1) = 1$, $\mathtt{S}^0(\widehat{\alpha}) = \widehat{\alpha}$ and $\mathtt{S}^{j+1}(n) = \mathtt{S}(\mathtt{S}^j(n))$. As usually done for indices, we shall abbreviate $\mathtt{S}^{j-1}(1)$ as j. Positions may be defined by associating a tree to each de Bruijn pre-metaterm, as was done in the case of $SERS$. As one might expect, the tree associated to A must have one of the forms depicted in Figure 2. The 'sub' in the rightmost tree may be seen as a dummy function symbol.

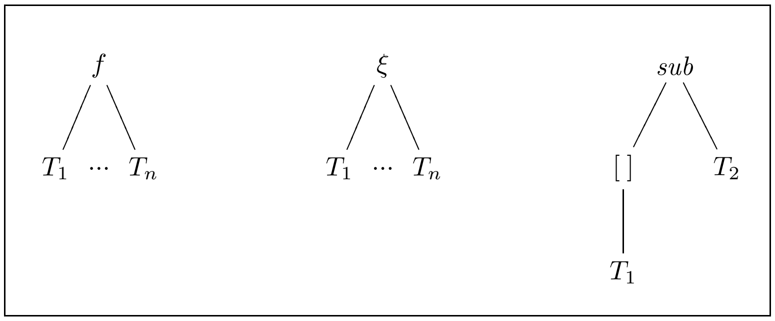

FIGURE 2. de Bruijn pre-metaterms as trees

Although the formal mechanism used to translate pre-metaterms with names into pre-metaterms with de Bruijn indices will be given in Section 4, let us introduce some intuitive ideas in order to justify the syntax used for i-metavariables. In the formalism $SERS$ there is a clear distinction between free and bound o-metavariables. This fact must also be reflected in $SERS_{dB}$, where bound o-metavariables are represented with indices and free o-metavariables are represented with i-metavariables (this distinction between free and bound variables is also used in some formalizations of λ-calculus [26]). However, free variables (or indices) in $SERS_{dB}$ appear always in a binding context, so that a de Bruijn valuation of such variables has to reflect the adjustment needed to represent the same variables but in a different context. This can be done by prefixing the i-metavariable by as many operators $\mathtt{S}$ as necessary. As an example consider the pre-metaterm $\xi\alpha.\widehat{\beta}$. If we translate it to $\xi\widehat{\beta}$, then a de Bruijn valuation like $\kappa = \{\widehat{\beta}/1\}$ binds the variable whereas this is completely impossible in the name formalism thanks to the conditions imposed on a name valuation (condition on variable assignments in Definition 2.19). Our solution is then to translate the pre-metaterm $\xi\alpha.\widehat{\beta}$ by $\xi(\mathtt{S}(\widehat{\beta}))$ in such a way that when instantiating some index for $\widehat{\beta}$ there is no capture of variables in the resulting term. The solution adopted here for translating pre-free o-metavariables into the de Bruijn formalism is in some sense what is called *pre-cooking* [5] in [12].

As in the $SERS$ formalism, we also need here a notion of well-formed pre-metaterm. The first motivation is to guarantee that labels of t-metavariables are correct w.r.t the context in which they appear, the second one is to ensure that indices like j (resp. $\mathtt{S}^j(\widehat{\alpha})$) correspond to bound (resp. free) variables. Indeed, the pre-metaterms $\xi(X_{\alpha\beta})$, $\xi(\xi(4))$ and $\xi(\widehat{\alpha})$ shall not make sense for us, and hence shall not be considered well-formed.

DEFINITION 3.3 (de Bruijn metaterms)
A pre-metaterm $A \in \mathsf{PMT}_{dB}$ over Σ is said to be a *metaterm* over Σ iff the predicate $\mathcal{WF}(A)$ holds, where $\mathcal{WF}(A)$ iff $\mathcal{WF}_\epsilon(A)$, and $\mathcal{WF}_l(A)$ is defined by induction on the structure of the pre-metaterm A for any label l as follows:

- $\mathcal{WF}_l(\mathtt{S}^j(1))$ iff $j+1 \leq |l|$
- $\mathcal{WF}_l(\mathtt{S}^j(\widehat{\alpha}))$ iff $j = |l|$ and l is a simple label
- $\mathcal{WF}_l(X_k)$ iff $l = k$ and l is a simple label
- $\mathcal{WF}_l(f(A_1,\ldots,A_n))$ iff for all $1 \leq i \leq n$ we have $\mathcal{WF}_l(A_i)$
- $\mathcal{WF}_l(\xi(A_1,\ldots,A_n))$ iff there exists $\alpha \notin l$ such that for all $1 \leq i \leq n$ we have $\mathcal{WF}_{\alpha l}(A_i)$
- $\mathcal{WF}_l(A_1[A_2])$ iff $\mathcal{WF}_l(A_2)$ and there exists $\alpha \notin l$ such that $\mathcal{WF}_{\alpha l}(A_1)$.

Therefore indices of the form $\mathtt{S}^j(1)$ may only occur in metaterms if they represent bound variables and well-formed metaindices of the form $\mathtt{S}^j(\widehat{\alpha})$ always represent a free variable. Also, if $\mathcal{WF}_k(A)$, then any metavariable occurring in A must be of the form X_{lk} for some label l (moreover, lk is a simple label). Note that when considering $\mathcal{WF}_l(M)$ and $\mathcal{WF}_l(A)$ it is Definitions 2.5 and 3.3 which are referenced, respectively.

EXAMPLE 3.4
The pre-metaterms $\xi(X_\alpha, \lambda(Y_{\beta\alpha}, 2))$, $f(\widehat{\beta}, \lambda(Y_\alpha, \mathtt{S}(\widehat{\alpha})))$ and $g(\lambda(\xi c))$ are metaterms, however the pre-metaterms $f(\mathtt{S}(\widehat{\alpha}), \xi(X_\beta))$, $\lambda(\xi(X_{\alpha\alpha}))$, $f(\widehat{\beta}, \lambda(\xi(\mathtt{S}(\widehat{\beta}))))$ are not.

DEFINITION 3.5 (Free de Bruijn metavariables)
Let A be a de Bruijn pre-metaterm. The set of *free metavariables* of A, written $\text{FMVAR}(A)$, is defined as:

$$\begin{array}{lcl} \text{FMVAR}(1) & \overset{\text{def}}{=} & \emptyset \\ \text{FMVAR}(\mathtt{S}(I)) & \overset{\text{def}}{=} & \text{FMVAR}(I) \\ \text{FMVAR}(\widehat{\alpha}) & \overset{\text{def}}{=} & \{\widehat{\alpha}\} \\ \text{FMVAR}(X_l) & \overset{\text{def}}{=} & \{X_l\} \end{array} \qquad \begin{array}{lcl} \text{FMVAR}(f(A_1,\ldots,A_n)) & \overset{\text{def}}{=} & \bigcup_{i=1}^n \text{FMVAR}(A_i) \\ \text{FMVAR}(\xi(A_1,\ldots,A_n)) & \overset{\text{def}}{=} & \bigcup_{i=1}^n \text{FMVAR}(A_i) \\ \text{FMVAR}(A_1[A_2]) & \overset{\text{def}}{=} & \text{FMVAR}(A_1) \cup \text{FMVAR}(A_2) \end{array}$$

[5]The pre-cooking function takes a $\lambda\sigma$-term with t-metavariables and suffixes them with as many explicit shift operators [1] as the number of binders present in its parameter path. This avoids variable capture when the higher-order unification procedure finds solutions for the t-metavariables.

The set of *names* of free metavariables of A is the set $\mathrm{FMV}\textsc{ar}(A)$ where each X_l is replaced simply by X. We also write, by abuse of notation, $\mathrm{FMV}\textsc{ar}(A)$ to denote such a set of names. For example, $\mathrm{FMV}\textsc{ar}(f(\lambda X_\alpha, Y_\epsilon, \widehat{\alpha})) = \{X, Y, \widehat{\alpha}\}$. We use $\mathrm{MV}\textsc{ar}(A)$ to denote the set of all metavariables of the de Bruijn pre-metaterm A.

DEFINITION 3.6 (de Bruijn terms and de Bruijn contexts)
The set of *de Bruijn terms* over Σ, denoted T_{dB}, and the set of de Bruijn contexts over Σ are defined by:

$$\begin{array}{lccl} \text{de Bruijn indices} & n & ::= & 1 \mid \mathtt{S}(n) \\ \text{de Bruijn terms} & a & ::= & n \mid f(a, \ldots, a) \mid \xi(a, \ldots, a) \\ \text{de Bruijn contexts} & E & ::= & \Box \mid f(a, \ldots, E, \ldots, a) \mid \xi(a, \ldots, E, \ldots, a). \end{array}$$

We use $a, b, a_i, b_i, \ldots$ for de Bruijn terms and $E, F, \ldots$ for de Bruijn contexts. The notion of the tree associated to a may be defined as for de Bruijn pre-metaterms. We may refer to the *binder path number* of a context, which is the number of binders between the $\Box$ and the root. In contrast to Definition 2.9, we have here that de Bruijn terms are also de Bruijn pre-metaterms, that is, $\mathsf{T}_{dB} \subset \mathsf{PMT}_{dB}$, although note that some de Bruijn terms may not be de Bruijn metaterms, i.e. may not be well-formed de Bruijn pre-metaterms. Indeed, the valid term $\xi(\xi(4))$ is not a metaterm, however, the index 4 may be seen as a constant in the pre-metaterm $\xi(\xi(4))$. If an arbitrary free variable is wished to be represented in a metaterm, then i-metavariables should be used.

DEFINITION 3.7 (Free de Bruijn indices)
The set of *free indices* of a de Bruijn term a, written $\mathrm{FI}(a)$, is defined as follows:

$$\begin{array}{lcl} \mathrm{FI}(n) & \stackrel{\text{def}}{=} & \{n\} \\ \mathrm{FI}(f(a_1, \ldots, a_n)) & \stackrel{\text{def}}{=} & \bigcup_{i=1}^{n} \mathrm{FI}(a_i) \\ \mathrm{FI}(\xi(a_1, \ldots, a_n)) & \stackrel{\text{def}}{=} & (\bigcup_{i=1}^{n} \mathrm{FI}(a_i)) \setminus\!\!\setminus 1 \end{array}$$

where for every set of indices S, the operation $S \setminus\!\!\setminus j$ is defined as $\{n - j \mid n \in S \text{ and } n > j\}$.

When encoding SERS_{dB} systems as SERS systems we shall need to speak of the free variable names (objects in $\mathcal{V}$, from the definition of a SERS signature) associated to the free de Bruijn indices. For example, if $a = \xi(1, 2, 3)$, then $\mathrm{FI}(a) = \{1, 2\}$. The named variable associated to the free index 1 is x_1, and likewise for 2 it is x_2. In general, we write $\textsc{Names}(S)$ for the names of the variables whose indices are in the set S. For example, $\textsc{Names}(\mathrm{FI}(a)) = \{x_1, x_2\}$.

DEFINITION 3.8 (de Bruijn substitution and de Bruijn updating function)
The result of substituting a term b for the index $n \geq 1$ in a term a is denoted $a\{\!\{n \leftarrow b\}\!\}$ and defined as:

$$\begin{array}{lcl} f(a_1, \ldots, a_n)\{\!\{n \leftarrow b\}\!\} & \stackrel{\text{def}}{=} & f(a_1\{\!\{n \leftarrow b\}\!\}, \ldots, a_n\{\!\{n \leftarrow b\}\!\}) \\ \xi(a_1, \ldots, a_n)\{\!\{n \leftarrow b\}\!\} & \stackrel{\text{def}}{=} & \xi(a_1\{\!\{n+1 \leftarrow b\}\!\}, \ldots, a_n\{\!\{n+1 \leftarrow b\}\!\}) \\ m\{\!\{n \leftarrow b\}\!\} & \stackrel{\text{def}}{=} & \begin{cases} m - 1 & \text{if } m > n \\ \mathcal{U}_0^n(b) & \text{if } m = n \\ m & \text{if } m < n \end{cases} \end{array}$$

where for $i \geq 0$ and $n \geq 1$ we define the *updating functions* $\mathcal{U}_i^n(\bullet)$ as follows:

$$\begin{array}{lcl}
\mathcal{U}_i^n(f(a_1,\ldots,a_n)) & \overset{\text{def}}{=} & f(\mathcal{U}_i^n(a_1),\ldots,\mathcal{U}_i^n(a_n)) \\
\mathcal{U}_i^n(\xi(a_1,\ldots,a_n)) & \overset{\text{def}}{=} & \xi(\mathcal{U}_{i+1}^n(a_1),\ldots,\mathcal{U}_{i+1}^n(a_n)) \\
\mathcal{U}_i^n(m) & \overset{\text{def}}{=} & \begin{cases} m+n-1 & \text{if } m > i \\ m & \text{if } m \leq i. \end{cases}
\end{array}$$

Due to the various notions of substitution and replacement introduced so far, we give in Figure 3 a brief synopsis of the situation.

Operator	Names	de Bruijn	Terms	Metaterms	Explicit	Implicit	Def.
$M_1[\alpha \leftarrow M_2]$	$\surd$			$\surd$	$\surd$		Def. 2.2
$t_1\{x \leftarrow t_2\}$	$\surd$		$\surd$			$\surd$	Def. 2.10
$M_1 \ll\alpha\leftarrow M_2\gg$	$\surd$			$\surd$		$\surd$	Def. 2.11
$A_1[A_2]$		$\surd$		$\surd$	$\surd$		Def. 3.2
$a_1\{\!\{n \leftarrow a_2\}\!\}$		$\surd$	$\surd$			$\surd$	Def. 3.8

FIGURE 3. Notions of substitution

We now consider the rewrite rules of a $SERS_{dB}$. This includes defining valuations, their validity, and the term rewrite relation in $SERS_{dB}$. Rewrite rules are specified with de Bruijn metaterms, whereas the induced rewrite relation is on de Bruijn terms.

DEFINITION 3.9 ($SERS_{dB}$)
A *de Bruijn rewrite rule* over Σ is a pair of de Bruijn metaterms (L, R) over Σ (also written $L \rightarrow R$) such that:

- the first symbol (called head symbol) in L is a function symbol or a binder symbol,
- $\text{FMVAR}(R) \subseteq \text{FMVAR}(L)$, and
- the metasubstitution operator does not occur in L.

Finally, we define a $SERS_{dB}$ to be a pair $(\Sigma, \mathcal{R})$ where Σ is a $SERS_{dB}$-signature and $\mathcal{R}$ is a set of $SERS_{dB}$-rewrite rules over Σ.

As in the case of $SERS$, we shall often omit Σ and write $\mathcal{R}$ instead of $(\Sigma, \mathcal{R})$, if no confusion arises.

EXAMPLE 3.10
The λ_{dB}-calculus is defined by considering the signature containing the function symbols $\mathcal{F} = \{app\}$ and binder symbols $\mathcal{B} = \{\lambda\}$, together with the $SERS_{dB}$-rewrite rule:

$$app(\lambda X_\alpha, Y_\epsilon) \rightarrow_{\beta_{dB}} X_\alpha[Y_\epsilon].$$

The $\lambda_{dB}\eta_{dB}$-calculus is obtained by adding the $SERS_{dB}$-rewrite rule: $\lambda(app(X_\alpha, 1)) \rightarrow_{\eta_{dB}} X_\epsilon$.

See also Examples 4.7 and 4.8.

DEFINITION 3.11 (de Bruijn valuation)
A *de Bruijn valuation* κ over Σ is a pair of (partial) functions (κ_i, κ_t) where κ_i is a function from i-metavariables to positive integers greater than 0, and κ_t is a function from t-metavariables to de

Bruijn terms. It defines a function on metaindices and t-metavariables, also denoted κ, as expected:

$$\begin{array}{lcl}
\kappa 1 & \stackrel{\text{def}}{=} & 1 \\
\kappa \mathtt{S}(I) & \stackrel{\text{def}}{=} & \mathtt{S}(\kappa I) \\
\kappa \widehat{\alpha} & \stackrel{\text{def}}{=} & \kappa_i \widehat{\alpha} \\
\kappa X_l & \stackrel{\text{def}}{=} & \kappa_t X_l .
\end{array}$$

A valuation κ determines in a unique way a function $\overline{\kappa}$ from the set of pre-metaterms A with $\text{FMV}{\small\text{AR}}(A) \subseteq \mathit{Dom}(\kappa)$, where $\mathit{Dom}(\kappa)$ denotes the domain of κ, to the set of terms as follows:

$$\begin{array}{lcl}
\overline{\kappa}(f(A_1, \ldots, A_n)) & \stackrel{\text{def}}{=} & f(\overline{\kappa}A_1, \ldots, \overline{\kappa}A_n) \\
\overline{\kappa}(\xi(A_1, \ldots, A_n)) & \stackrel{\text{def}}{=} & \xi(\overline{\kappa}A_1, \ldots, \overline{\kappa}A_n) \\
\overline{\kappa}(A_1[A_2]) & \stackrel{\text{def}}{=} & \overline{\kappa}(A_1)\{\!\{1 \leftarrow \overline{\kappa}A_2\}\!\}.
\end{array}$$

Note that in the above definition the substitution operator $\bullet\{\!\{\bullet \leftarrow \bullet\}\!\}$ refers to the usual substitution defined on terms with de Bruijn indices (Definition 3.8).

In order to motivate the notion of valid de Bruijn valuation consider the following rule:

$$\xi\alpha.(\xi\beta.X) \rightarrow_r \xi\beta.(\xi\alpha.X).$$

Even if translation of rewrite rules into de Bruijn rewrite rules has not been defined yet (Section 4), one may guess that a reasonable translation would be the following rule:

$$\xi(\xi(X_{\beta\alpha})) \rightarrow_{r_{dB}} \xi(\xi(X_{\alpha\beta})),$$

which indicates that β (resp. α) is the first bound occurrence in the LHS (resp. RHS) while α (resp. β) is the second bound occurrence in the LHS (resp. RHS). Now, if X is instantiated by x, α by x and β by y in the SERS system, then we have an r-rewrite step $\xi x.(\xi y.x) \rightarrow \xi y.(\xi x.x)$. However, to reflect this fact in the corresponding SERS_{dB} system we need to instantiate $X_{\beta\alpha}$ by 2 and $X_{\alpha\beta}$ by 1, thus obtaining an r_{dB}-rewrite step $\xi(\xi\ 2) \rightarrow \xi(\xi\ 1)$. This clearly shows that de Bruijn t-metavariables having the same name but different label cannot be instantiated arbitrarily as they have to reflect the renaming of variables which is indicated by their labels. Indeed, the goal pursued by the labels of metavariables is that of incorporating 'context' information as a defining part of a metavariable. As a consequence, we must verify that the terms substituted for every occurrence of a fixed metavariable coincide 'modulo' their corresponding context. Dealing with such notion of 'coherence' of substitutions in a de Bruijn formalism is also present in other formalisms but in a more restricted form. Thus for example, as mentioned before, a pre-cooking function is used in [12] in order to avoid variable capture in the higher-order unification procedure. In *XRS* [23] the notions of binding arity and pseudo-binding arity are introduced in order to take into account the parameter path of the different occurrences of t-metavariables appearing in a rewrite rule. Then (roughly) it is required that the binding arity of a t-metavariable on the LHS of a rewrite rule (rewrite rules are required to be left-linear) equals the pseudo-binding arity of the same t-metavariable occurring on the RHS of the rule. Our notion of 'coherence' is implemented with *valid valuations* (Definition 3.13) and it turns out to be more general than the solutions proposed in [12] and [23].

DEFINITION 3.12 (Value function)
Let $a \in \mathsf{T}_{dB}$ and l be a label of binder indicators. We define the *value function* $\mathit{Value}(l, a)$ as $\mathit{Value}^0(l, a)$ where:

$$
\begin{array}{lcl}
\mathit{Value}^i(l,n) & \overset{\text{def}}{=} & \begin{cases} n & \text{if } n \leq i \\ \mathtt{at}(l, n-i) & \text{if } 0 < n-i \leq |l| \\ x_{n-i-|l|} & \text{if } n-i > |l| \end{cases} \\
\mathit{Value}^i(l, f(a_1,\ldots,a_n)) & \overset{\text{def}}{=} & f(\mathit{Value}^i(l,a_1),\ldots \mathit{Value}^i(l,a_n)) \\
\mathit{Value}^i(l, \xi(a_1,\ldots,a_n)) & \overset{\text{def}}{=} & \xi(\mathit{Value}^{i+1}(l,a_1),\ldots, \mathit{Value}^{i+1}(l,a_n)).
\end{array}
$$

The function $\mathit{Value}(l,a)$ interprets the de Bruijn term a in an l-context: bound indices are left untouched, free indices referring to the l-context are replaced by the corresponding binder indicator and the remaining free indices are replaced by their corresponding variable names. It might be observed that if repeated binder indicators are allowed in the label l of Definition 3.12, then this intuition would not seem to hold. Indeed, for our purposes the case of interest is when the label l is simple. Nevertheless, many auxiliary results may be proved without this requirement, thus we prefer not to restrict this definition prematurely (by requiring l to be simple). Finally, note also that $\mathit{Value}^i(l,n)$ may return three different kinds of results. This is just a technical resource to make easier later proofs. Indeed, we have for example $\mathit{Value}(\alpha\beta, \xi(f(3,1))) = \xi(f(\beta,1)) = \mathit{Value}(\beta\alpha, \xi(f(2,1)))$ and $\mathit{Value}(\epsilon, f(\xi 1, \lambda 2)) = f(\xi 1, \lambda x_1) \neq f(\xi 1, \lambda\alpha) = \mathit{Value}(\alpha, f(\xi 1, \lambda 2))$.

DEFINITION 3.13 (Valid de Bruijn valuation)
A de Bruijn valuation κ over Σ is said to be ***valid*** if for every pair of t-metavariables X_l and $X_{l'}$ in $\mathit{Dom}(\kappa)$ we have $\mathit{Value}(l, \kappa X_l) = \mathit{Value}(l', \kappa X_{l'})$. Likewise, we say that a de Bruijn valuation κ is ***valid for a rewrite rule*** (L,R) if every metavariable in (L,R) is in $\mathit{Dom}(\kappa)$ and for every pair of t-metavariables X_l and $X_{l'}$ in (L,R) we have $\mathit{Value}(l, \kappa X_l) = \mathit{Value}(l', \kappa X_{l'})$.

It is interesting to note that there is no concept analogous to safeness (Definition 2.20) as used for named SERS due to the use of de Bruijn indices. Also, the last condition in the definition of an admissible valuation (Definition 2.22) is subsumed by the above Definition 3.13 in the setting of SERS_{dB}.

EXAMPLE 3.14
Returning to the above mentioned example we have that $\kappa = \{X_{\beta\alpha}/2, X_{\alpha\beta}/1\}$ is valid for the rule r_{dB} since $\mathit{Value}(\beta\alpha, 2) = \alpha = \mathit{Value}(\alpha\beta, 1)$.

Another interesting example is the η-contraction rule $\lambda x.app(X,x) \rightarrow X$ if $x \notin \mathrm{FV}(X)$. Recall from Section 2 that it can be expressed in the SERS formalism, without conditions, as the rule $\lambda\alpha.app(X,\alpha) \rightarrow_\eta X$. In the SERS_{dB} formalism it may be expressed as $\lambda(app(X_\alpha, 1)) \rightarrow_{\eta_{dB}} X_\epsilon$.

Observe that this kind of rule cannot be expressed in the XRS formalism since it does not verify the binding arity condition. Our formalism allows us to write rules like η_{dB} because valid valuations will test for coherence of values. Indeed, an admissible valuation for η is a valuation θ such that θX does not contain a free occurrence of $\theta(\alpha)$. This is exactly the condition used in any usual formalization of the η-rule. A valid valuation κ for η_{dB} could, for example, be a valuation $\kappa = \{X_\alpha/m, X_\epsilon/n\}$ such that $\mathit{Value}(\alpha, \kappa X_\alpha) = \mathit{Value}(\epsilon, \kappa X_\epsilon)$, that is, $m = 1$ is not possible, and n is necessarily $m-1$.

To summarize, valid valuations guarantee that the unique value assigned to a t-metavariable X in the framework with names is translated accordingly in the de Bruijn framework w.r.t the different parameter paths of all the occurrences of X in the rewrite rule. This is, in some sense, an updating of X w.r.t the different parameter paths where it appears, and it gives us the right notion of coherence for valuations.

DEFINITION 3.15 (Rewriting de Bruijn terms)
Let $\mathcal{R}$ be a set of de Bruijn rules over Σ and a, b de Bruijn terms, over Σ. We say that a *$\mathcal{R}$-rewrites* or *$\mathcal{R}$-reduces to* b, written $a \rightarrow_{\mathcal{R}} b$, iff there is a de Bruijn rule $(L, R) \in \mathcal{R}$ and a de Bruijn valuation κ valid for (L, R) such that $a = E[\kappa L]$ and $b = E[\kappa R]$, where E is a de Bruijn context.

Thus, the term $\lambda(app(\lambda(app(1,3)),1))$ rewrites by the η_{dB} rule to $\lambda(app(1,2))$, using the (valid) valuation $\kappa = \{X_\alpha/\lambda(app(1,3), X_\epsilon/\lambda(app(1,2))\}$.

The rewrite relation on de Bruijn terms satisfies the following property:

LEMMA 3.16
Let $a \in \mathsf{T}_{dB}$. If $a \rightarrow_{\mathcal{R}} b$, then $\mathrm{FI}(b) \subseteq \mathrm{FI}(a)$.

4 From names to indices

In this section we show how rewriting in the $SERS$ formalism may be simulated in the $SERS_{dB}$ formalism. This requires two well-distinguished phases, which we can refer to as the *definition phase* and the *rewrite-preservation phase*. The definition phase consists in defining appropriate translations from pre-metaterms, terms and valuations in the $SERS$ setting into the corresponding notions in the $SERS_{dB}$ setting, work which is carried out in the first part of this section. The second part deals with the rewrite-preservation phase, that is, showing how $SERS$ rewrite steps can be simulated via $SERS_{dB}$ rewrite steps. The rewrite-preservation phase shall use the results developed in the definition phase.

4.1 The definition phase

We begin by showing how to translate terms into de Bruijn terms.

DEFINITION 4.1 (From terms (and contexts) to de Bruijn terms (and contexts))
The translation of a term t, denoted $T(t)$, is defined as $T_\epsilon(t)$ where

$$\begin{array}{lcl}
T_k(x) & \overset{\text{def}}{=} & \begin{cases} \mathtt{pos}(x,k) & \text{if } x \in k \\ \mathcal{O}(x) + |k| & \text{if } x \notin k \end{cases} \\
T_k(f(t_1,\ldots,t_n)) & \overset{\text{def}}{=} & f(T_k(t_1),\ldots,T_k(t_n)) \\
T_k(\xi x.(t_1,\ldots,t_n)) & \overset{\text{def}}{=} & \xi(T_{xk}(t_1),\ldots,T_{xk}(t_n)).
\end{array}$$

The translation of a context, denoted $T(C)$, is defined as above but adding the clause $T_k(\Box) \overset{\text{def}}{=} \Box$.

As a consequence of the previous definition, there is a clear bijection between the set of free variables of a term t and the set of free variables of its de Bruijn representation $T(t)$.

The following lemma will be used to prove the main results of this section; it states that variable renaming commutes with translation.

LEMMA 4.2
Let $s \in \mathsf{T}$, let l, k be labels of variables and x, y variables such that y does not occur at all in s and $x, y \notin l$. If $s\{x \leftarrow y\}$ is defined then $T_{lyk}(s\{x \leftarrow y\}) = T_{lxk}(s)$.

PROOF. By induction on s.

- $s = x$. Then we reason as follows:

$$\begin{array}{rcll} T_{lyk}(x\{x \leftarrow y\}) & = & T_{lyk}(y) & \\ & = & \mathtt{pos}(y, lyk) & \\ & = & |l| + 1 & (y \notin l) \\ & = & \mathtt{pos}(x, lxk) & (x \notin l) \\ & = & T_{lxk}(s). & \end{array}$$

- $s = z \neq x$. Then $T_{lyk}(z\{x \leftarrow y\}) = T_{lyk}(z)$. We consider two further cases:
  - $z \in lyk$. Since $y \neq z$ (y does not occur in s) $T_{lyk}(z) = \mathtt{pos}(z, lyk) = \mathtt{pos}(z, lxk) = T_{lxk}(z)$.
  - $z \notin lyk$. Then $T_{lyk}(z) = \mathcal{O}(z) + |lyk| = \mathcal{O}(z) + |lxk| = T_{lxk}(z)$.
- $s = f(s_1, \dots, s_n)$. We use the induction hypothesis.
- $s = \xi z.(s_1, \dots, s_n)$ with $z \neq x$. Note that $z \neq y$ by hypothesis. We reason as follows:

$$\begin{array}{rcll} T_{lyk}(s\{x \leftarrow y\}) & = & \xi(T_{zlyk}(s_1\{x \leftarrow y\}), \dots, T_{zlyk}(s_n\{x \leftarrow y\})) & \\ & = & \xi(T_{zlxk}(s_1), \dots, T_{zlxk}(s_n)) & (i.h.) \\ & = & T_{lxk}(s). & \end{array}$$

∎

As expected, the translation satisfies:

LEMMA 4.3 (T is compatible with α-conversion)
Given two terms $s, t \in \mathsf{T}$ such that $s =_\alpha t$ we have $T_k(s) = T_k(t)$ for any label of variables k.

PROOF. By induction on the derivation of $s =_\alpha t$.

- Base cases. If $s = t$ then the result holds trivially, so suppose $s =_\alpha t$ and the conversion takes place at the root. Then $s = \xi x.(s_1, \dots, s_n) =_\alpha \xi y.(s_1\{x \leftarrow y\}, \dots, s_n\{x \leftarrow y\}) = t$ where x and y are variables not occurring in $s_1, \dots s_n$. Then by Lemma 4.2 we have $T_k(s) = \xi(T_{xk}(s_1), \dots, T_{xk}(s_n)) = \xi(T_{yk}(s_1\{x \leftarrow y\}), \dots, T_{yk}(s_n\{x \leftarrow y\})) = T_k(t)$.
- Inductive cases:
  - $s =_\alpha t$ follows from $t =_\alpha s$. We use the induction hypothesis.
  - $s =_\alpha t$ follows from $s =_\alpha s'$ and $s' =_\alpha t$. We use the induction hypothesis.
  - the conversion is internal. Then two further cases are considered:
    - $s = f(s_1, \dots, s_i, \dots, s_n)$ and $t = f(s_1, \dots, s'_i, \dots, s_n)$ where $s_i =_\alpha s'_i$. We conclude by using the induction hypothesis.
    - $s = \xi x.(s_1, \dots, s_i, \dots, s_n)$ and $t = \xi x.(s_1, \dots, s'_i, \dots, s_n)$ where $s_i =_\alpha s'_i$. Then we have

$$\begin{array}{l} T_k(s) = \xi(T_{xk}(s_1), \dots, T_{xk}(s_i), \dots, T_{xk}(s_n)) =_{i.h.} \\ \xi(T_{xk}(s_1), \dots, T_{xk}(s'_i), \dots, T_{xk}(s_n)) = T_k(t). \end{array}$$

∎

We now consider a translation from pre-metaterms to de Bruijn pre-metaterms. We shall also use the letter T for this translation to avoid having to introduce yet another symbol.

DEFINITION 4.4 (From pre-metaterms to de Bruijn pre-metaterms)
Let M be a pre-metaterm. Its translation, denoted $T(M)$, is defined as $T_\epsilon(M)$ where $T_k(M)$ is defined by

$$\begin{array}{llllll} T_k(\alpha) & \overset{\text{def}}{=} & \texttt{pos}(\alpha,k), \text{ if } \alpha \in k & T_k(f(M_1,\ldots,M_n)) & \overset{\text{def}}{=} & f(T_k(M_1),\ldots,T_k(M_n)) \\ T_k(\widehat{\alpha}) & \overset{\text{def}}{=} & \texttt{S}^{|k|}(\widehat{\alpha}) & T_k(\xi\alpha.(M_1,\ldots,M_n)) & \overset{\text{def}}{=} & \xi(T_{\alpha k}(M_1),\ldots,T_{\alpha k}(M_n)) \\ T_k(X) & \overset{\text{def}}{=} & X_k & T_k(M_1[\alpha \leftarrow M_2]) & \overset{\text{def}}{=} & T_{\alpha k}(M_1)[T_k(M_2)]. \end{array}$$

Note that if M is a metaterm, then $T(M)$ will be defined and will only have t-metavariables with simple labels. Note also that for some pre-metaterms, such as $\xi\alpha.\beta$, the translation $T(\bullet)$ is not defined. Moreover, if M is a metaterm then $T(M)$ is a de Bruijn metaterm.[6]

EXAMPLE 4.5
Let $M = \xi\alpha.(X, \lambda\beta.(Y,\alpha))$, $M' = f(\widehat{\beta}, \lambda\alpha.(Y,\widehat{\alpha}))$ and $M'' = g(\lambda\alpha.(\xi\beta.c))$. Then their respective translations are $A = \xi(X_\alpha, \lambda(Y_{\beta\alpha}, \texttt{S}(1)))$, $A' = f(\widehat{\beta}, \lambda(Y_\alpha, \texttt{S}(\widehat{\alpha})))$ and $A'' = g(\lambda(\xi c))$, which are metaterms as remarked in Example 3.4.

DEFINITION 4.6 (From $SERS$ rewrite rules to $SERS_{dB}$ rewrite rules)
Let (G,D) be a rewrite rule in the $SERS$ formalism. Then $T(G,D)$ denotes the translation of the rewrite rule, defined as $(T(G), T(D))$.

As an immediate consequence of Definitions 4.4 and 4.6, if (G,D) is an $SERS$ rewrite rule, then $T(G,D)$ is an $SERS_{dB}$ rewrite rule.

EXAMPLE 4.7 ($\lambda\mathtt{x}$ continued)
Following Example 2.16, the specification of $\lambda\mathtt{x}$ in the $SERS_{dB}$ formalism is given below. It results from translating the rewrite rules of Example 2.16.

$$\begin{array}{lll} app(\lambda X_\alpha, Z_\epsilon) & \to & subs(\sigma X_\alpha, Z_\epsilon) \\ subs(\sigma(app(X_\alpha, Y_\alpha)), Z_\epsilon) & \to & app(subs(\sigma X_\alpha, Z_\epsilon), subs(\sigma Y_\alpha, Z_\epsilon)) \\ subs(\sigma(\lambda(X_{\beta\alpha})), Z_\epsilon) & \to & \lambda(subs(\sigma(X_{\alpha\beta}), Z_\beta)) \\ subs(\sigma(1), Z_\epsilon) & \to & Z_\epsilon \\ subs(\sigma(\texttt{S}(\widehat{\beta})), Z_\epsilon) & \to & \widehat{\beta}. \end{array}$$

The rule $subs(\sigma(\lambda X_{\beta\alpha}), Z_\epsilon) \to \lambda(subs(\sigma X_{\alpha\beta}, Z_\beta))$ is interesting since it illustrates the use of binder commutation from $X_{\beta\alpha}$ to $X_{\alpha\beta}$ and shows how some index adjustment is necessary when going from Z_ϵ to Z_β.

EXAMPLE 4.8 (The $\lambda\Delta$-calculus continued)
The translation of the $\lambda\Delta$-calculus (Example 2.17) yields the following rewrite rules in the $SERS_{dB}$ formalism:

$$\begin{array}{lll} app(\lambda X_\alpha, Z_\epsilon) & \to_{\beta_{dB}} & X_\alpha[Z_\epsilon] \\ app(\Delta X_\alpha, Z_\epsilon) & \to_{\Delta_1} & \Delta(X_{\alpha\beta}[\lambda(app(\texttt{S}(1), app(1, Z_{\gamma\beta})))]) \\ \Delta(app(1, X_\alpha)) & \to_{\Delta_2} & X_\epsilon \\ \Delta(app(1, (\Delta(app(\texttt{S}(1), X_{\beta\alpha}))))) & \to_{\Delta_3} & X_\epsilon. \end{array}$$

We remark that the translation of the Δ_1, Δ_2 and Δ_3 rules would not be possible in XRS.

Suppose some rewrite rule (L,R) is used to rewrite a term s. Then $s =_\alpha C[\theta(L)]$ for some context C and admissible valuation θ. When encoding this rewrite step in the $SERS_{dB}$ setting we have to encode not only terms and metaterms, but also the valuation θ. Definition 4.9 below shows how one may encode valuations. This definition is parameterized over a label k, an issue which we would like to clarify. Suppose the metavariable X_l occurs in L, then when θ instantiates X_l the status of any variable x in the resulting term, $\theta(X_l)$, can be of one of four classes (see also Figure 4):

[6]This can be proved by showing a more general property, namely, for every pre-metaterm M, if $\mathcal{WF}_l(M)$, then $\mathcal{WF}_l(T_l(M))$.

a) either, x is bound in $\theta(X_l)$,

b) or, x is free in $\theta(X_l)$ but is bound by some binder above X_l in L, in other words, there is a binder indicator $\alpha \in l$ such that $\theta(\alpha) = x$,

c) or, x is free in $\theta(X_l)$ and x is not bound by the binders above X_l in the rule, i.e. $x \notin \theta(l)$, but x is bound by a binder above the $\Box$ in the context C,

d) or, x is free in $\theta(X_l)$, it is not bound by the binders above X_l in the rewrite rule, and it is not bound by a binder in the context C above the $\Box$. Thus x is free in s.

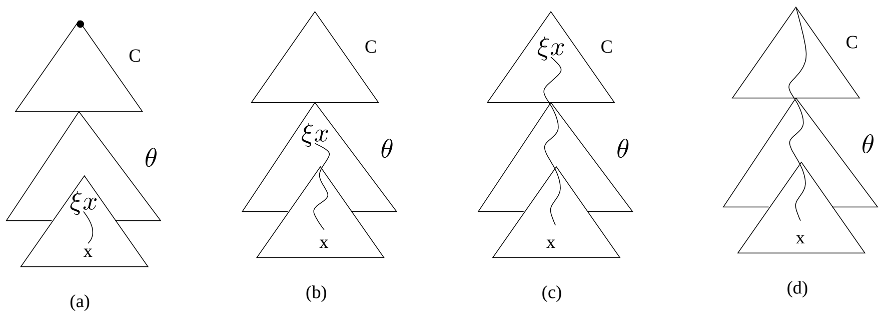

FIGURE 4. Instantiating a metavariable X_l of a metaterm L in some context C

Therefore, when translating a valuation to the $SERS_{dB}$ setting we need to know what the names of the variables of the binders above the $\Box$ are. For example, if the third case holds then when translating to indices we must assign x an index which avoids being captured in the context. This is the rôle of the label k in the following definition.

DEFINITION 4.9 (From valuations to de Bruijn valuations)
Let θ be a valuation and k be a label of variables. Then the translation of θ w.r.t the label k (referred to as the context label) is defined as the de Bruijn valuation:

$$\begin{array}{lcl} T_k(\theta)(X_l) & \overset{\text{def}}{=} & T_{\theta(l)k}(\theta(X)) \text{ if } \theta(l) \text{ is defined} \\ T_k(\theta)(\widehat{\alpha}) & \overset{\text{def}}{=} & T_k(\theta(\widehat{\alpha})) \end{array}$$

where $X, \widehat{\alpha} \in Dom(\theta)$.

4.2 The rewrite-preservation phase

In this section we study the rewrite-preservation phase, that is, we show that the translations of the definition phase ensure that the notion of rewriting in the formalism with de Bruijn indices has the same semantics as the corresponding one with names.

The following lemmas will be used in the proof of the main result of this section, namely Proposition 4.15, which states that $SERS$-rewriting may be simulated as $SERS_{dB}$-rewriting.

LEMMA 4.10 (T and Updating Functions)
Let $s \in \mathsf{T}$, let l_1, l_2, k be labels of variables such that $|l_1| = j$, $|l_2| = i - 1$ and FV$(s) \cap l_2 = \emptyset$. Then $T_{l_1 l_2 k}(s) = \mathcal{U}^i_j(T_{l_1 k}(s))$.

PROOF. By induction on s. We shall consider the case that s is a variable, the others follow from the induction hypothesis. Suppose $s = x$. Note that since by hypothesis $x \notin l_2$ we have three cases to consider:

- $x \in l_1$. Then $T_{l_1 l_2 k}(x) = \mathtt{pos}(x, l_1 l_2 k) = \mathtt{pos}(x, l_1 k) = \mathcal{U}^i_j(\mathtt{pos}(x, l_1 k))$. The last equality holds since $\mathtt{pos}(x, l_1 k) \leq j$.
- $x \notin l_1$ and $x \in k$. Then $T_{l_1 l_2 k}(x) = \mathtt{pos}(x, l_1 l_2 k) = \mathtt{pos}(x, l_1 k) + i - 1 = \mathcal{U}^i_j(\mathtt{pos}(x, l_1 k))$.
- $x \notin l_1 k$. Then $T_{l_1 l_2 k}(x) = \mathcal{O}(x) + |l_1 l_2 k| = \mathcal{O}(x) + |l_1 k| + i - 1 = \mathcal{U}^i_j(T_{l_1 k}(x))$.

∎

LEMMA 4.11 (T is compositional w.r.t substitution)
Let $s, t \in \mathsf{T}$, l, k be labels of variables with $|l| = i - 1$, let x be a variable such that $x \notin l$, and suppose FV$(t) \cap l = \emptyset$. If $s\{x \leftarrow t\}$ is defined then $T_{lk}(s\{x \leftarrow t\}) = T_{lxk}(s)\{\!\{i \leftarrow T_k(t)\}\!\}$.

PROOF. By induction on s. See the Appendix. ∎

As expected, the translation is well-behaved w.r.t contexts and valuations. We take the opportunity to remind the reader that the notion of a parameter path is given in Definition 2.9. Induction on the context C may be used for proving the following result.

LEMMA 4.12 (T is compositional w.r.t contexts)
Let C be a context, l the parameter path of C and $t \in \mathsf{T}$. Then for every label k we have, $T_k(C[t]) = T_k(C)[T_{lk}(t)]$.

LEMMA 4.13 (T is compositional w.r.t valuations)
Let M be a pre-metaterm, l a label of binder indicators, and suppose

1. $\mathcal{WF}_l(M)$,
2. $\theta = (\theta_v, \theta_t)$ is a valuation such that θ_v is injective on the bound o-metavariables, and
3. θ is safe for M.

Then for every label k we have $T_{\theta(l)k}(\theta M) = T_k(\theta)(T_l(M))$.

PROOF. By induction on the pre-metaterm M. See the Appendix. ∎

LEMMA 4.14
Let k, k' be labels of binder indicators, l a label of variables and θ be an injective function on the set of binder indicators. Then for every $t \in \mathsf{T}$, every $p \geq 0$, every $x_1, \ldots, x_p$, if for every $z \in$ FV$(t) \setminus \{x_1, \ldots, x_p\}$ we have $z \in \theta(k)$ iff $z \in \theta(k')$, then

$$Value^p(k, T_{x_1 \ldots x_p \theta(k) l}(t)) = Value^p(k', T_{x_1 \ldots x_p \theta(k') l}(t)).$$

PROOF. By induction on t. See the Appendix. ∎

We can finally conclude with the main result of this section, which ensures that the $SERS_{dB}$ formalism preserves $SERS$-rewriting.

PROPOSITION 4.15 (Simulating $SERS$-rewriting via $SERS_{dB}$-rewriting)
Suppose that $s \rightarrow t$ in the $SERS$ formalism using the rewrite rule (G, D). Then $T(s) \rightarrow T(t)$ in the $SERS_{dB}$ formalism using the de Bruijn rewrite rule $T(G, D)$.

PROOF. By definition of the rewrite relation (Definition 2.23) there is an admissible valuation θ for (G, D) and there is a context C such that $s =_\alpha C[\theta G]$ and $t =_\alpha C[\theta D]$. By Lemma 4.3 $T(s) = T(C[\theta G])$ and $T(t) = T(C[\theta D])$. Note that $T(G, D) = (T(G), T(D))$ is a de Bruijn rewrite rule as remarked just after Definitions 4.4 and 4.6. The proof thus proceeds in two steps: in Step 1 we show that there exists a de Bruijn valuation κ and a de Bruijn context E such that $T(s) = E[\kappa T(G)]$ and $T(t) = E[\kappa T(D)]$; Step 2 consists in showing that κ is a valid de Bruijn valuation for $(T(G), T(D))$.

- Step 1. By Lemma 4.12 we have $T(s) = T(C)[T_k(\theta G)]$ and $T(t) = T(C)[T_k(\theta D)]$, where k is the parameter path of the context C, and $T(C)$ is a de Bruijn context. By hypothesis G is well-formed and θ is safe for G (so that θ_v is injective on the set of bound o-metavariables). As a consequence we can apply Lemma 4.13 so that $T_k(\theta G) = T_k(\theta)(T(G))$ and $T_k(\theta D) = T_k(\theta)(T(D))$, where $T_k(\theta)$ is a de Bruijn valuation. Thus we may take $\kappa \overset{\text{def}}{=} T_k(\theta)$ and $E \overset{\text{def}}{=} T(C)$.
- Step 2. We have still to show that $T_k(\theta)$ is valid for $(T(G), T(D))$. By Definition 3.13 we have to check that $\mathit{Value}(l, T_k(\theta)(X_l)) = \mathit{Value}(l', T_k(\theta)(X_{l'}))$ for every pair of t-metavariables X_l and $X_{l'}$ appearing in the de Bruijn rewrite rule $(T(G), T(D))$, that is, by Definition 4.9, that $\mathit{Value}(l, T_{\theta(l)k}(\theta(X))) = \mathit{Value}(l', T_{\theta(l')k}(\theta(X)))$. Finally, verifying the following conditions allows us to conclude from Lemma 4.14 with $p = 0$:
  - $\theta(X)$ is a term in T by definition of valuations,
  - θ is injective on bound o-metavariables since it is admissible,
  - finally, we need to show that for every variable $z \in \mathrm{FV}(\theta X)$ we have $z \in \theta(l)$ iff $z \in \theta(l')$. But this immediately follows from the fact that θ verifies the path condition for X in (G, D) because it is admissible.

∎

5 From indices to names

In this section we show that SERS are operationally equivalent to SERS_{dB}. For that, we show how the notion of rewriting in the SERS_{dB} formalism may be simulated in the SERS. As in Section 4 we develop the required results by distinguishing the *definition phase* and the *rewrite-preservation phase*.

5.1 The definition phase

We begin with a translation from de Bruijn terms to terms with variable names. This makes use of the NAMES($\bullet$) function given after Definition 3.7.

DEFINITION 5.1 (From de Bruijn terms (and contexts) to terms (and contexts))
The translation of $a \in \mathsf{T}_{dB}$, denoted $U(a)$, is defined as $U_\epsilon^{\text{NAMES}(\mathrm{FI}(a))}(a)$ where, for every finite set of variables S, and every label of variables k, $U_k^S(a)$ is defined as follows:

$$U_k^S(n) \overset{\text{def}}{=} \begin{cases} \mathtt{at}(k, n) & \text{if } n \leq |k| \\ x_{n-|k|} & \text{if } n > |k| \text{ and } x_{n-|k|} \in S \end{cases}$$

$$U_k^S(f(a_1, \ldots, a_n)) \overset{\text{def}}{=} f(U_k^S(a_1), \ldots, U_k^S(a_n))$$
$$U_k^S(\xi(a_1, \ldots, a_n)) \overset{\text{def}}{=} \xi x.(U_{xk}^S(a_1), \ldots, U_{xk}^S(a_n)) \text{ for any } x \notin k \cup S$$

The translation of a de Bruijn context E, denoted $U(E)$, is defined as above but adding the clause $U_k^S(\Box) \overset{\text{def}}{=} \Box$. We remark that we can always choose $x \notin k \cup S$ since both k and S are finite.

Note that $U(\bullet)$ is not a function since the choice of bound variables is non-deterministic. However, one can show that if t and t' belong both to $U(a)$, then $t =_\alpha t'$. Thus, $U(\bullet)$ can be seen as a function from de Bruijn terms to α-equivalence classes.

REMARK 5.2
We remark that given a set of variables S, a de Bruijn term a and a label k, the translation $U_k^S(a)$ is always defined if $\text{NAMES}(\text{FI}(a)\backslash\backslash|k|) \subseteq S$. It is quite evident that $\text{FI}(\xi(a))\backslash\backslash n$ is exactly $\text{FI}(a)\backslash\backslash(n+1)$. Also, if $U_k^S(C[a])$ is defined and $|l|$ is the binder path number of C (see Definition 3.6), then $U_{lk}^S(a)$ is also defined. Note, moreover, that if $x \in \text{FV}(U_k^S(a))$ then $x \in S \cup k$.

DEFINITION 5.3 (From de Bruijn pre-metaterms to pre-metaterms)
The translation of a de Bruijn pre-metaterm A, denoted $U(A)$, is defined as $U_\epsilon(A)$, where $U_l(A)$ is defined as follows:

$$
\begin{array}{ll}
U_l(\mathtt{S}^i(1)) & \overset{\text{def}}{=} \mathtt{at}(l, i+1), \text{if } i+1 \leq |l| \\
U_l(\mathtt{S}^{|l|}(\widehat{\alpha})) & \overset{\text{def}}{=} \widehat{\alpha} \\
U_l(X_l) & \overset{\text{def}}{=} X \\
U_l(f(A_1, \ldots, A_n)) & \overset{\text{def}}{=} f(U_l(A_1), \ldots, U_l(A_n)) \\
U_l(\xi(A_1, \ldots, A_n)) & \overset{\text{def}}{=} \xi\alpha.(U_{\alpha l}(A_1), \ldots, U_{\alpha l}(A_n)), \text{ if } \quad \mathcal{WF}_{\alpha l}(A_i) \text{ for some } \alpha \notin l \text{ s.t. } 1 \leq i \leq n \\
U_l(A_1[A_2]) & \overset{\text{def}}{=} U_{\alpha l}(A_1)[\alpha \leftarrow U_l(A_2)], \text{ if } \mathcal{WF}_{\alpha l}(A_1) \text{ for some } \alpha \notin l.
\end{array}
$$

As in Definition 5.1 the translation of a de Bruijn pre-metaterm is not a function since it depends on the choice of the names for o-metavariables. Indeed, two different pre-metaterms obtained by this translation will be v-equivalent. Also, for some de Bruijn pre-metaterms such as $\xi(2)$, the translation may not be defined. However, it is defined on de Bruijn metaterms.

DEFINITION 5.4 (From $SERS_{dB}$ rewrite rules to $SERS$ rewrite rules)
Let (L, R) be a de Bruijn rewrite rule then its translation, denoted $U(L, R)$, is the pair of metaterms $(U(L), U(R))$.

Note that if A is such that $\mathcal{WF}_l(A)$ holds then its translation $U_l(A)$ is also a named metaterm, that is, $\mathcal{WF}_l(U_l(A))$ also holds. Therefore, by Definition 2.14 the translation of a de Bruijn rewrite rule is a rewrite rule in the $SERS$ formalism. As mentioned above, if a de Bruijn pre-metaterm A is not a de Bruijn metaterm then $U_l(A)$ may not be defined.

EXAMPLE 5.5
Consider the de Bruijn rule $app(\Delta X_\alpha, Z_\epsilon) \rightarrow \Delta(X_{\alpha\beta}[\lambda(app(2, app(1, Z_{\gamma\beta})))])$ from Example 4.8. The rule obtained by the translation of Definition 5.3 is

$$app(\Delta\alpha.X, Z) \rightarrow \Delta\beta.(X[\alpha \leftarrow \lambda\gamma.(app(\beta, app(\gamma, Z)))]),$$

whereas, for the rule $subs(\sigma(\mathtt{S}(\widehat{\beta})), Z_\epsilon) \rightarrow \widehat{\beta}$ we obtain $subs(\sigma\gamma.\widehat{\beta}, Z) \rightarrow \widehat{\beta}$ for some bound o-metavariable γ.

DEFINITION 5.6 (From de Bruijn valuations to valuations)
Let $\kappa = (\kappa_i, \kappa_t)$ be a de Bruijn valuation, S be a finite set of variables and k a label of variables, and θ_v be a variable assignment such that:

1. $\theta_v(\alpha) \notin S \cup k$, for any $\alpha \in Dom(\theta_v)$, and
2. for every $\widehat{\alpha} \in Dom(\kappa_i)$, $\theta_v(\widehat{\alpha}) = \begin{cases} \mathtt{at}(k, \kappa_i(\widehat{\alpha})) & \text{if } \kappa_i(\widehat{\alpha}) \leq |k| \\ x_{\kappa_i(\widehat{\alpha})-|k|} & \text{otherwise with } x_{\kappa_i(\widehat{\alpha})-|k|} \in S. \end{cases}$

The translation of κ is the valuation $U_{(\theta_v,S,k)}(\kappa) \overset{\text{def}}{=} (\theta_v, \theta_t)$, where

$$\theta_t X \quad \overset{\text{def}}{=} \quad U^S_{\theta_v(l)k}(\kappa X_l) \qquad \text{for any } X_l \text{ in } Dom(\kappa).$$

Condition 2 on θ_v says that if an i-metavariable in A is bound (or free) in the context label k as interpreted via κ then the new valuation $U_{(\theta_v,S,k)}(\kappa)$ must reflect this fact. We will now show that if κ is a valid de Bruijn valuation then this definition is *correct*, that is, the definition does not depend on the choice of the t-metavariable X_l in $Dom(\kappa)$. For that, we need some lemmas, which are developed in the Appendix.

LEMMA 5.7 (Translation of de Bruijn valuations is correct)
The valuation $U_{(\theta,S,k)}(\kappa)$ of Definition 5.6 is correct if κ is valid, where correct means that for every X_l and $X_{l'}$ in $Dom(\kappa)$ we have $U^S_{\theta_v(l)k}(\kappa X_l) =_\alpha U^S_{\theta_v(l')k}(\kappa X_{l'})$, whenever both terms are defined.

PROOF. Since κ is valid we have $Value(l, \kappa X_l) = Value(l', \kappa X_{l'})$ for every X_l and X'_l in $Dom(\kappa)$. Then by Lemma C.3 we may conclude $U^S_{\theta_v(l)k}(\kappa X_l) =_\alpha U^S_{\theta_v(l')k}(\kappa X_{l'})$. ■

5.2 The rewrite-preservation phase

In this section we study the rewrite-preservation phase, that is, we show that the translations of the definition phase ensure that the notion of rewriting in the formalism with names has the same semantics as the corresponding one with de Bruijn indices. More precisely, we seek to prove Proposition 5.12.

PROPOSITION 5.12 (Simulating $SERS_{dB}$-rewriting via $SERS$-rewriting)
Assume $a \rightarrow b$ in the $SERS_{dB}$ formalism using rewrite rule (L, R). Then $U(a) \rightarrow U(b)$ in the $SERS$ formalism using rule $U(L, R)$.

For that we need to develop some intermediate results. The following lemma states that the translation is well-behaved w.r.t de Bruijn contexts. Induction on the context E and Lemma C.1 (see the Appendix) may be used for proving it.

LEMMA 5.8 (U is compositional w.r.t de Bruijn contexts)
Let E be a de Bruijn context, l, k labels where l is the parameter path of $U^S_k(E)$ and $a \in \mathsf{T}_{dB}$. If $U^S_k(E[a])$ is defined, then $U^S_k(E[a]) =_\alpha U^S_k(E)[U^S_{lk}(a)]$.

LEMMA 5.9 (U and Updating Functions)
Let $a \in \mathsf{T}_{dB}$, l_1, l_2 and k be labels of variables with $|l_1| = j$ and $|l_2| = i - 1$. Then $U^S_{l_1 l_2 k}(\mathcal{U}^i_j(a)) =_\alpha U^S_{l_1 k}(a)$ if $U^S_{l_1 k}(a)$ is defined.

PROOF. We proceed by induction on a. The case $a = f(a_1, \ldots, a_n)$ is straightforward by the induction hypothesis, so we consider the other ones.

- $a = n$. We have two cases to consider:
  - $n \leq j$. Then $U^S_{l_1 l_2 k}(\mathcal{U}^i_j(n)) = U^S_{l_1 l_2 k}(n) = \mathtt{at}(l_1 l_2 k, n) = \mathtt{at}(l_1 k, n) = U^S_{l_1 k}(n)$.

  - $n > j$. Then $U^S_{l_1 l_2 k}(\mathcal{U}^i_j(n)) = U^S_{l_1 l_2 k}(n+i-1)$ and we have two further subcases to consider:
    - $n+i-1 \leq |l_1 l_2 k|$. Then since $n \leq |l_1 k|$ we have $U^S_{l_1 l_2 k}(n+i-1) = \mathtt{at}(l_1 l_2 k, n+i-1) = \mathtt{at}(l_1 k, n) = U^S_{l_1 k}(n)$.
    - $n+i-1 > |l_1 l_2 k|$. Then since $n > |l_1 k|$, $U^S_{l_1 l_2 k}(n+i-1) = x_{n+i-1-|l_1 l_2 k|} = x_{n-|l_1 k|} = U^S_{l_1 k}(n)$.
- $a = \xi(a_1, \ldots, a_n)$. Then we reason as follows:

$$\begin{array}{rcll}
U^S_{l_1 l_2 k}(\mathcal{U}^i_j(a)) & = & \xi z.(U^S_{zl_1 l_2 k}(\mathcal{U}^i_{j+1}(a_1)), \ldots, U^S_{zl_1 l_2 k}(\mathcal{U}^i_{j+1}(a_n))) & \\
& =_\alpha & \xi z.(U^S_{zl_1 k}(a_1), \ldots, U^S_{zl_1 k}(a_n)) & (i.h.) \\
& =_\alpha & \xi z'.(U^S_{zl_1 k}(a_1)\{z \leftarrow z'\}, \ldots, U^S_{zl_1 k}(a_n)\{z \leftarrow z'\}) & (z' \text{ fresh}) \\
& =_\alpha & \xi z'.(U^S_{z' l_1 k}(a_1), \ldots, U^S_{z' l_1 k}(a_n)) & (Lemma\ C.1) \\
& =_\alpha & \xi z'.(U^S_{yl_1 k}(a_1)\{y \leftarrow z'\}, \ldots, U^S_{yl_1 k}(a_n)\{y \leftarrow z'\}) & (Lemma\ C.1) \\
& =_\alpha & \xi y.(U^S_{yl_1 k}(a_1), \ldots, U^S_{yl_1 k}(a_n)) & \\
& = & U^S_{l_1 k}(a). &
\end{array}$$

The phrase 'z' fresh' should be read, in full rigor, as 'z' does not occur in $U^S_{zl_1 k}(a_i)$ nor in $U^S_{yl_1 k}(a_i)$ for $1 \leq i \leq n$'. The definition of $U^{\bullet}_{\bullet}(\bullet)$ and the hypothesis that $U^S_{l_1 k}(a)$ is defined, allows us to apply Lemma C.1 above.

∎

LEMMA 5.10 (U is compositional w.r.t de Bruijn substitution)
Let $a, b \in \mathsf{T}_{dB}$, l and k labels of variables with $|l| = i-1$, x a variable such that $x \notin lk \cup S$. Then $U^S_{lk}(a\{\!\{i \leftarrow b\}\!\}) =_\alpha U^S_{lxk}(a)\{x \leftarrow U^S_k(b)\}$, assuming both sides of the equation are defined.

PROOF. The proof is by induction on a. See the Appendix. ∎

LEMMA 5.11 (U is compositional w.r.t valuations)
Consider a de Bruijn valuation $\kappa = (\kappa_i, \kappa_t)$, a de Bruijn pre-metaterm A, a finite set of variables S, a variable assignment θ_v verifying the hypothesis in Definition 5.6, a label of binder indicators l and a label of variables k. If the following conditions hold:

1. κ is valid,
2. κA is defined,
3. θ_v is defined over l and the bound o-metavariables in $U_l(A)$,
4. θ_v is injective on the bound o-metavariables,
5. $\textsc{Names}(\text{FI}(\kappa A) \setminus\!\!\setminus |\theta_v(l)k|) \subseteq S$, and
6. $\mathcal{WF}_l(A)$.

Then, $U^S_{\theta_v(l)k}(\kappa A) =_\alpha U_{(\theta_v, S, k)}(\kappa) U_l(A)$.

Intuitively k represents the context information where the reduction is performed (thus k is a label of variables). We also require $\textsc{Names}(\text{FI}(\kappa A) \setminus\!\!\setminus |\theta_v(l)k|) \subseteq S$ to ensure that $U^S_{\theta_v(l)k}(\kappa A)$ is defined.

PROOF. By induction on A. See the Appendix. ∎

The reader should note that the translation of a valid de Bruijn valuation is an admissible named valuation. Recall that a valuation is admissible for a rewrite rule (G, D) iff the following conditions hold:

- θ is safe for (G, D) (Definition 2.20),
- if α and β occur in (G, D) with $\alpha \neq \beta$ then $\theta_v\alpha \neq \theta_v\beta$, and
- θ verifies the path condition (Definition 2.21) for every t-metavariable in (G, D).

Safeness is considered in Lemma D.1 and Lemma D.3 goes on to consider admissibility. Both results are developed in the Appendix. So we move on directly to the main result of this section, i.e. that the $SERS$ formalism preserves $SERS_{dB}$-rewriting.

PROPOSITION 5.12 (Simulating $SERS_{dB}$-rewriting via $SERS$-rewriting)
Assume $a \rightarrow b$ is a reduction step in the $SERS_{dB}$ formalism using rewrite rule (L, R). Then $U(a) \rightarrow U(b)$ in the $SERS$ formalism using rule $U(L, R)$.

PROOF. Let us consider the de Bruijn rewrite step $a \rightarrow b$ using a de Bruijn valuation κ which is valid for (L, R). Without loss of generality we can suppose that κ is only defined on the metavariables of (L, R). And, let $U(L, R) = (G, D)$. By definition of the rewrite relation we have a de Bruijn context E such that $a = E[\kappa L]$ and $b = E[\kappa R]$. We proceed as follows:

- Take S as the set of variables $\text{NAMES}(\text{FI}(a))$ so that $U^S_\epsilon(a)$ is defined. Note that since $\text{FI}(b) \subseteq \text{FI}(a)$ holds by Corollary 3.16, $U^S_\epsilon(b)$ is also defined.
- Let k be the parameter path of $U^S_\epsilon(E)$.
- Now, to apply Lemma 5.8 we need to show that $U^S_k(\kappa L)$ and $U^S_k(\kappa R)$ are defined, which follows from the first and second items. Therefore, $U^S_\epsilon(E[\kappa L]) =_\alpha U^S_\epsilon(E)[U^S_k(\kappa L)]$ and $U^S_\epsilon(E[\kappa R]) =_\alpha U^S_\epsilon(E)[U^S_k(\kappa R)]$.
- The next step is to apply Lemma 5.11 in order to decompose $U^S_k(\kappa L)$ and $U^S_k(\kappa R)$. First of all, let us fix any variable assignment θ_v such that it verifies the following requirements:
  - it is defined over all the o-metavariables in $U_\epsilon(L)$ and $U_\epsilon(R)$ and only on these,
  - it is injective on the bound o-metavariables,
  - $\theta_v(\alpha) \notin S \cup k$ for any bound o-metavariable $\alpha \in Dom(\theta_v)$ (i.e. the variables assigned to bound o-metavariables in the rewrite rule $(U(L), U(R))$ are not confused with the free variables in a and b, that is, with the variables in S, nor with the variables bound in (the parameter path of) the context where the rewrite-step takes place, that is, the variables in k).
  - We also define θ_v on the free o-metavariables of the rewrite rule $(U(L), U(R))$ as the hypothesis dictates, i.e. for all $\widehat{\alpha}$ we define

$$\theta_v(\widehat{\alpha}) \stackrel{\text{def}}{=} \begin{cases} \texttt{at}(k, \kappa_i(\widehat{\alpha})) & \text{if } \kappa_i(\widehat{\alpha}) \leq |k| \\ x_{\kappa_i(\widehat{\alpha})-|k|} & \text{otherwise with } x_{\kappa_i(\widehat{\alpha})-|k|} \in S. \end{cases}$$

  We shall now consider the case of $U^S_k(\kappa L)$, the other one being similar. We must thus meet the conditions of the previous lemma in order to resolve $U^S_k(\kappa L)$. Let $l = \epsilon$.
  1. κ is valid by hypothesis.
  2. κL is defined since $a = E[\kappa L]$.
  3. We also have that θ_v is defined over ϵ and the bound o-metavariables in $U_\epsilon(L)$ and $U_\epsilon(R)$.
  4. The assignment θ_v is injective over the bound o-metavariables.
  5. $\text{NAMES}(\text{FI}(\kappa L) \setminus\!\!\setminus |k|) \subseteq S$ holds since by definition we set $S = \text{NAMES}(\text{FI}(a))$ (Note that by Corollary 3.16 we have $\text{NAMES}(\text{FI}(b)) \subseteq S$).
  6. Finally, $\mathcal{WF}_\epsilon(L)$ holds since (L, R) is a de Bruijn rewrite rule and hence L and R are well-formed de Bruijn pre-metaterms.

  We may thus apply Lemma 5.11.

Let us summarize our situation:

$$\begin{array}{lll} U^S_\epsilon(E[\kappa L]) & =_\alpha & U^S_\epsilon(E)[U^S_k(\kappa L)] \qquad (Lemma\ 5.8) \\ & =_\alpha & U^S_\epsilon(E)[U_{(\theta_v,S,k)}(\kappa)(U_\epsilon(L))] \quad (Lemma\ 5.11) \\ & = & U^S_\epsilon(E)[U_{(\theta_v,S,k)}(\kappa)G] \end{array}$$

and

$$\begin{array}{lll} U^S_\epsilon(E[\kappa R]) & =_\alpha & U^S_\epsilon(E)[U^S_k(\kappa R)] \qquad (Lemma\ 5.8) \\ & =_\alpha & U^S_\epsilon(E)[U_{(\theta_v,S,k)}(\kappa)(U_\epsilon(R))] \quad (Lemma\ 5.11) \\ & = & U^S_\epsilon(E)[U_{(\theta_v,S,k)}(\kappa)D]. \end{array}$$

So we now define the named context $C \overset{\text{def}}{=} U^S_\epsilon(E)$ and we also define the named valuation $\theta' \overset{\text{def}}{=} U_{(\theta_v,S,k)}(\kappa)$. Then we have $U(a) = C[\theta' G]$ and $U(b) = C[\theta' D]$. In order to conclude that $U(a) \to U(b)$, by definition of $SERS$-rewriting, we are left to verify that θ' is admissible for (G, D). Now,

1. $U_{(\theta_v,S,k)}(\kappa)$ is defined for *all* the metavariables of G and D since both $U_{(\theta_v,S,k)}(\kappa)(G)$ and $U_{(\theta_v,S,k)}(\kappa)(D)$ are defined.
2. θ_v is injective on *all* the bound o-metavariables in (G, D) by definition of θ_v.

We can then apply Lemma D.3 and conclude that θ' is admissible for (G, D).

∎

6 Preserving properties

This section studies the relationship between the translation functions over pre-metaterms and terms introduced above. It gives rise to two results stating, respectively, that given a metaterm M then $U(T(M))$ is v-equivalent (Definition 2.12) to M (see Figure 5), and that given a de Bruijn metaterm A then $T(U(A))$ is identical to A. These results are listed below and proved in the Appendix. They are used to show that properties such as confluence, local confluence, the diamond property and strong and weak normalization are preserved when translating an $SERS$ rewrite system into a $SERS_{dB}$ rewrite system and vice versa.

LEMMA 6.1
1. Let $M \in \mathsf{PMT}$ such that $\mathcal{WF}(M)$. Then $U(T(M)) =_v M$.
2. Let $t \in \mathsf{T}$. Then $U(T(t)) =_\alpha t$.
3. Let $A \in \mathsf{PMT}_{dB}$. If $\mathcal{WF}(A)$ then $T(U(A)) = A$.
4. Let $a \in \mathsf{T}_{dB}$. Then $T(U(a)) = a$.

The following lemma ensures that rewrite rules described with v-equivalent metaterms generate the same rewrite relation. This lemma together with the above mentioned results and the simulation propositions (Proposition 4.15 and Proposition 5.12) are exactly what is needed to preserve properties in both senses.

LEMMA 6.2
Let (G, D) and (G', D') be $SERS$ rewrite rules such that $G =_v G'$ and $D =_v D'$. Then $\to_{(G,D)} = \to_{(G',D')}$.

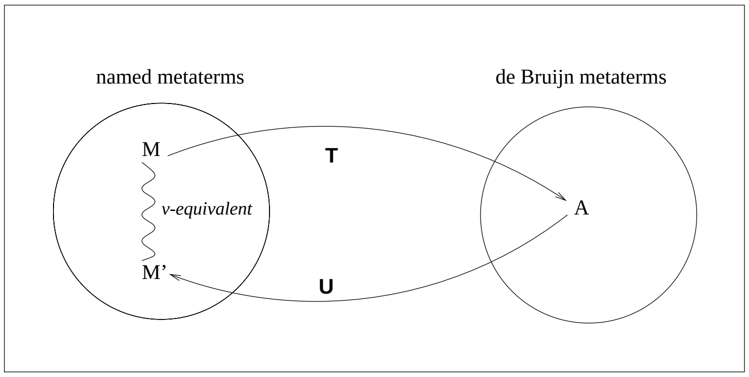

FIGURE 5. v-equivalence and translations

PROOF. Without loss of generality we prove that if $s \rightarrow_{(G,D)} t$ then $s \rightarrow_{(G',D')} t$. Thus let us assume that there is an admissible valuation θ for (G,D) and a context C such that $s =_\alpha C[\theta G]$ and $C[\theta D] =_\alpha t$.

The set of bound o-metavariables occurring in (G', D') may be divided into two (not necessarily disjoint) sets $\mathcal{B}_1$ and $\mathcal{B}_2$. In $\mathcal{B}_1$ we find those bound o-metavariables which occur in the parameter path of some t-metavariable in (G', D'), and in $\mathcal{B}_2$ the other bound variables occurring in (G', D'). The o-metavariables in $\mathcal{B}_1$ are not renamed in any way by the v-equivalence relation (Definition 2.12). We define the valuation $\theta' = (\theta'_t, \theta'_v)$ as follows:

$$\begin{array}{lcll} \theta'_t X & \overset{\text{def}}{=} & \theta_t X & \\ \theta'_v \alpha & \overset{\text{def}}{=} & \theta_v \alpha & \text{if } \alpha \in \mathcal{B}_1 \\ \theta'_v \widehat{\alpha} & \overset{\text{def}}{=} & \theta_v \widehat{\alpha}. & \end{array}$$

In order to fully define θ' we must consider the value it assigns to those o-metavariables in $\mathcal{B}_2$ which are not in $\mathcal{B}_1$. For these we simply require θ' to assign any variables such that the resulting valuation is safe for (G', D'), and θ'_v is injective on the bound o-metavariables.

Observe the following:

1. MVAR$(G', D') \subseteq Dom(\theta')$,
2. θ' is by construction an admissible valuation for (G', D'), and
3. $s =_\alpha C[\theta G] =_\alpha C[\theta' G']$ and $t =_\alpha C[\theta D] =_\alpha C[\theta' D']$.

Hence $s \rightarrow_{(G',D')} t$. ∎

COROLLARY 6.3
Let (G, D) be an *SERS* rewrite rule. Then the rewrite relations generated by (G, D) and $U(T(G, D))$ are identical.

PROOF. Use Lemma 6.1(1), Lemma 6.2 and the fact that the translations preserve well-formedness. ∎

DEFINITION 6.4 (Local Confluence, Confluence, Diamond Property)
Let $\mathcal{S}$ be a *reduction relation* defined on a set $\mathcal{O}$ and let $\mathcal{S}^*$ be its reflexive-transitive closure. The relation $\mathcal{S}$ is said to have the *local confluence* (resp. *confluence*) property iff for every $a, b, c \in \mathcal{O}$

such that $a\mathcal{S}b$ and $a\mathcal{S}c$ (resp. $a\mathcal{S}^*b$ and $a\mathcal{S}^*c$), there is $d \in \mathcal{O}$ such that $b\mathcal{S}^*d$ and $c\mathcal{S}^*d$. The relation $\mathcal{S}$ is said to have the *diamond* property if for every $a, b, c \in \mathcal{O}$ such that $a\mathcal{S}b$ and $a\mathcal{S}c$ there is $d \in \mathcal{O}$ such that $b\mathcal{S}d$ and $c\mathcal{S}d$.

Preservation of confluence is stated in the two following theorems. Preservation of local confluence and of the diamond property are stated and proved analogously.

THEOREM 6.5
If $\mathcal{R}$ is a confluent $SERS$ then $T(\mathcal{R})$ is a confluent $SERS_{dB}$.

PROOF. Suppose $a \twoheadrightarrow_{T(\mathcal{R})} b$ and $a \twoheadrightarrow_{T(\mathcal{R})} c$ for some de Bruijn terms a, b, c. Applying the translation mapping $U(\bullet)$ and using Proposition 5.12 we obtain the diagram (b) of Figure 6. The reductions denoted by the dotted lines are obtained by Corollary 6.3 and the confluence of $\mathcal{R}$.

Now applying the translation mapping $T(\bullet)$ and using Proposition 4.15 we obtain the diagram (c) of Figure 6. Finally, Lemma 6.1(4) and Lemma 6.1(3) yield the desired diagram illustrated as diagram (a) in Figure 6.

$$\begin{tikzcd}
 & a \arrow[dl, two heads, "T(\mathcal{R})"'] \arrow[dr, two heads, "T(\mathcal{R})"] & \\
b \arrow[dr, dotted, two heads, "T(\mathcal{R})"'] & & c \arrow[dl, dotted, two heads, "T(\mathcal{R})"] \\
 & T(s) &
\end{tikzcd}$$

(a)

$$\begin{tikzcd}
 & U(a) \arrow[dl, two heads, "U(T(\mathcal{R}))"'] \arrow[dr, two heads, "U(T(\mathcal{R}))"] & \\
U(b) \arrow[dr, dotted, two heads, "U(T(\mathcal{R}))"'] & & U(c) \arrow[dl, dotted, two heads, "U(T(\mathcal{R}))"] \\
 & s &
\end{tikzcd}$$

(b)

$$\begin{tikzcd}
 & T(U(a)) \arrow[dl, two heads, "T(U(T(\mathcal{R})))"'] \arrow[dr, two heads, "T(U(T(\mathcal{R})))"] & \\
T(U(b)) \arrow[dr, dotted, two heads, "T(U(T(\mathcal{R})))"'] & & T(U(c)) \arrow[dl, dotted, two heads, "T(U(T(\mathcal{R})))"] \\
 & T(s) &
\end{tikzcd}$$

(c)

FIGURE 6. Preservation of confluence from names to indices

∎

THEOREM 6.6
If $\mathcal{R}$ is a confluent $SERS_{dB}$ then $U(\mathcal{R})$ is a confluent $SERS$.

PROOF. Suppose $s \twoheadrightarrow_{U(\mathcal{R})} t_1$ and $s \twoheadrightarrow_{U(\mathcal{R})} t_2$ for some terms s, t_1, t_2. Applying the translation mapping $T(\bullet)$ and using Proposition 4.15 we may obtain the diagram (b) of Figure 7. The reductions denoted by the dotted lines are obtained by the confluence of $\mathcal{R}$. Note also that Lemma 6.1(3) has been used.

Now applying the translation mapping $U(\bullet)$ and using Proposition 5.12 we obtain the diagram (c) of Figure 7. Finally, Lemma 6.1(2) and the definition of reduction Definition 2.23 yield the desired diagram illustrated as diagram (a) in Figure 7.

∎

We focus now on preservation of normalization properties.

DEFINITION 6.7 (Weak and Strong Normalization)
Let us consider a reduction relation $\mathcal{S}$ on a set $\mathcal{O}$. The relation $\mathcal{S}$ is said to have the *weak normalization* property iff for every $a \in \mathcal{O}$ there is at least one finite $\mathcal{S}$-reduction chain $a\mathcal{S} \ldots \mathcal{S}b$ such that

$$\begin{tikzcd}
& s \arrow[dl, ->>, "U(\mathcal{R})"'] \arrow[dr, ->>, "U(\mathcal{R})"] & \\
t_1 \arrow[dr, dotted, ->>, "U(\mathcal{R})"'] & & t_2 \arrow[dl, dotted, ->>, "U(\mathcal{R})"] \\
& U(a) &
\end{tikzcd}$$

(a)

$$\begin{tikzcd}
& T(s) \arrow[dl, ->>, "T(U(\mathcal{R}))"'] \arrow[dr, ->>, "T(U(\mathcal{R}))"] & \\
T(t_1) \arrow[dr, dotted, ->>, "T(U(\mathcal{R}))"'] & & T(t_2) \arrow[dl, dotted, ->>, "T(U(\mathcal{R}))"] \\
& a &
\end{tikzcd}$$

(b)

$$\begin{tikzcd}
& U(T(s)) \arrow[dl, ->>, "U(T(U(\mathcal{R})))"'] \arrow[dr, ->>, "U(T(U(\mathcal{R})))"] & \\
U(T(t_1)) \arrow[dr, dotted, ->>, "U(T(U(\mathcal{R})))"'] & & U(T(t_2)) \arrow[dl, dotted, ->>, "U(T(U(\mathcal{R})))"] \\
& U(a) &
\end{tikzcd}$$

(c)

FIGURE 7. Preservation of confluence from indices to names

b cannot be further reduced. The relation $\mathcal{S}$ is said to have the *strong normalization* property iff for every $a \in \mathcal{O}$ there is no infinite $\mathcal{S}$-reduction chain starting at a.

THEOREM 6.8

1. If $\mathcal{R}$ is a strongly normalizing $SERS$, then $T(\mathcal{R})$ is a strongly normalizing $SERS_{dB}$.
2. If $\mathcal{R}$ is a strongly normalizing $SERS_{dB}$, then $U(\mathcal{R})$ is a strongly normalizing $SERS$.

PROOF. Both items are proved in a similar way. As an example, we prove the first one. Suppose an infinite derivation for some de Bruijn terms $a_0, a_1, a_2, \ldots$:

$$a_0 \to_{T(\mathcal{R})} a_1 \to_{T(\mathcal{R})} a_2 \to_{T(\mathcal{R})} \cdots$$

Applying the translation mapping $U(\bullet)$ and using Proposition 5.12 we obtain:

$$U(a_0) \to_{U(T(\mathcal{R}))} U(a_1) \to_{U(T(\mathcal{R}))} U(a_2) \to_{U(T(\mathcal{R}))} \cdots$$

which is an infinite derivation in $\mathcal{R}$ by Lemma 6.1(1) and Corollary 6.3, contradicting that $\mathcal{R}$ is strongly normalizing.

∎

Theorem 6.8 also holds if all occurrences of 'strongly normalizing' are replaced by 'weakly normalizing'.

7 Conclusions

We have proposed a formalism for higher-order rewriting with de Bruijn notation and we have shown that rewriting with names and rewriting with indices are operationally equivalent. We have given formal translations from one formalism into the other which can be viewed as an *interface* in programming languages based on higher-order rewriting systems.

In a sequel article [5, 6] we introduce a conversion procedure from the Simplified Expression Reduction Systems with de Bruijn indices into first-order rewriting. Composing the translation presented in this work with the aforementioned conversion procedure yields a framework for establishing a precise relation between higher-order rewriting with names and first-order rewriting.

References

[1] M. Abadi, L. Cardelli, P. L. Curien, and J.-J. Lévy. Explicit substitutions. *Journal of Functional Programming*, **4**, 375–416, 1991.

[2] H. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1984. Revised edition.

[3] M. Bognar and R. de Vrijer. The context calculus lambda-c. In *Workshop on Logical Frameworks and Meta-languages (LFM)*, Paris, France, 1999.

[4] E. Bonelli, D. Kesner, and A. Ríos. A de Bruijn notation for higher-order rewriting. In *11th International Conference on Rewriting Techniques and Applications*, L. Bachmair, ed., pp. 62–79. Volume 1833 of *Lecture Notes in Computer Science*, Springer-Verlag, 2000.

[5] E. Bonelli, D. Kesner, and A. Ríos. From higher-order to first-order rewriting (extended abstract). In *12th International Conference on Rewriting Techniques and Applications*, A. Middeldorp, ed., pp. 47–62. Volume 2051 of *Lecture Notes in Computer Science*, Springer-Verlag, 2001.

[6] E. Bonelli, D. Kesner, and A. Ríos. Relating higher-order and first-order rewriting. *Journal of Logic and Computation*, 2005.

[7] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[8] R. Bloo and K. Rose. Combinatory reduction systems with explicit substitution that preserve strong normalisation. In *7th International Conference on Rewriting Techniques and Applications*, H. Ganzinger, ed., Vvolume 1103 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996.

[9] V. Breazu-Tannen. Combining algebra and higher order types. In *3rd Annual IEEE Symposium on Logic in Computer Science (LICS)*, pp. 82–90. IEEE Computer Society Press, 1988.

[10] P.-L. Curien. *Categorical Combinators, Sequential Algorithms and Functional Programming*. Progress in Theoretical Computer Science. Birkhaüser, 1993. Second edition.

[11] J. Despeyroux, A. Felty, and A. Hirschowitz. Higher-order abstract syntax in coq. Technical Report 2556, INRIA, May 1995.

[12] G. Dowek, T. Hardin, and C. Kirchner. Higher-order unification via explicit substitutions. In *Proceedings of Tenth Annual IEEE Symposium on Logic in Computer Science (LICS)*, D. Kozen, ed. San Diego, USA, 1995.

[13] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge University Press, 1990.

[14] M. Gabbay and A. Pitts. A new approach to abstract syntax involving binders. In *14th Annual IEEE Symposium on Logic in Computer Science (LICS)*, G. Longo, ed., pp. 214–224. IEEE Computer Society Press, 1999.

[15] J. Goguen, J. Thatcher and E. Wagner. An initial algebra approach to the specification, correctness and implementation of abstract data types. In *Current Trends in Programming Methodology*, volume 4, pp. 80–149. Prentice Hall, 1978.

[16] G. Huet and D. Oppen. Equations and rewrite rules: A survey. In *Formal Language Theory: Perspectives and Open Problems*, R. V. Book, ed., pp. 349–405. Academic Press, 1980.

[17] Z. Khasidashvili. Expression reduction systems. In *Proceedings of IN Vekua Institute of Applied Mathematics*, volume 36, Tbilisi, 1990.

[18] J.-W. Klop. *Combinatory Reduction Systems*. PhD thesis, Mathematical Centre Tracts 127, CWI, Amsterdam, 1980.

[19] J.-W. Klop. *Term Rewriting Systems*, volume 2. Oxford University Press, 1992.

[20] Z. Khasidashvili, M. Ogawa, and V. van Oostrom. Perpetuality and uniform normalization in orthogonal rewrite systems. *Information and Computation*, **164**, 118–151, 2001.

[21] F. Kamareddine and A. Ríos. Bridging de Bruijn indices and variable names in explicit substitutions calculi. *Logic Journal of the Interest Group of Pure and Applied Logic*, **6**, 843–874, 1998.

[22] T. Nipkow. Higher-order critical pairs. In *6th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pp. 342–349. IEEE Computer Society Press, 1991.

[23] B. Pagano. *Des calculs de substitution explicite et de leur application à la compilation des langages fonctionnels*. Thèse de doctorat, Université Pierre et Marie Curie, 1998.

[24] F. Pfenning and C. Elliot. Higher-order abstract syntax. In *ACM Symposium on Programming Language Design and Implementation*, pp. 199–208. ACM, 1988.

[25] A. Pitts. Nominal logic, a first order theory of names and binding. *Information and Computation*, **186**, 165–193, 2003.

[26] R. Pollack. Closure under alpha-conversion. In *Types for Proofs and Programs (TYPES)*, H. Barendregt and T. Nipkow, eds. Number 806 in Lecture Notes in Computer Science, Nijmegen, The Netherlands, 1993.

[27] K. Rose. Explicit cyclic substitutions. In *Proceedings of the Third International Workshop on Conditional Term Rewriting Systems (CTRS)*, M. Rusinowitch and J.-L. Rémy, eds., pp. 36–50. Number 656 in Lecture Notes in Computer Science, 1992.

[28] J. Rehof and M. H. Sørensen. The λ_Δ calculus. In *Theoretical Aspects of Computer Software (TACS)*, M. Hagiya and J. Mitchell, eds., pp. 516–542. Number 789 in *Lecture Notes in Computer Science*, Springer-Verlag, 1994.

[29] TERESE. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, March 2003.

[30] C. Urban, A. Pitts, and M. Gabbay. Nominal unification. *Theoretical Computer Science*, 2004.

[31] V. van Oostrom. *Confluence for Abstract and Higher-order Rewriting*. PhD thesis, Vrije University, Amsterdam, Netherlands, 1994.

[32] V. van Oostrom and F. van Raamsdonk. Weak orthogonality implies confluence: the higher-order case. In *Proceedings of the 3rd International Symposium on Logical Foundations of Computer Science*, volume 813 of *Lecture Notes in Computer Science*, A. Nerode and Y. Matiyasevich, eds, pp. 379–392. Springer-Verlag, 1994.

[33] F. van Raamsdonk. *Confluence and Normalization for Higher-Order Rewriting*. PhD thesis, Amsterdam University, Netherlands, 1996.

[34] D. Wolfram. *The Clausal Theory of Types*. Volume 21 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1993.

Appendix

A Compositionality of T

We show that the translation T from names to indices is compositional w.r.t substitution and valuations.

LEMMA A.1 (T is compositional w.r.t. substitution)
Let $s, t \in \mathsf{T}$, l, k be labels of variables with $|l| = i - 1$, let x be a variable such that $x \notin l$, and suppose $\mathrm{FV}(t) \cap l = \emptyset$. If $s\{x \leftarrow t\}$ is defined then $T_{lk}(s\{x \leftarrow t\}) = T_{lxk}(s)\{\!\{i \leftarrow T_k(t)\}\!\}$.

PROOF. By induction on s.

- $s = x$. Then we have

$$\begin{array}{rcll} T_{lk}(t) & = & \mathcal{U}_0^i(T_k(t)) & (\mathit{Lemma}\ 4.10) \\ & = & i\{\!\{i \leftarrow T_k(t)\}\!\} & \\ & = & T_{lxk}(s)\{\!\{i \leftarrow T_k(t)\}\!\} & (x \notin l). \end{array}$$

- $s = y \neq x$. Then we have three subcases to consider
  - $y \in l$. Then $T_{lk}(y) = \mathtt{pos}(y, lk) = \mathtt{pos}(y, lxk) = T_{lxk}(y) = T_{lxk}(y)\{\!\{i \leftarrow T_k(t)\}\!\}$. The last equality holds because $\mathtt{pos}(y, l) < i$.
  - $y \notin l$ and $y \in k$. Then $T_{lk}(y) = \mathtt{pos}(y, lk) = \mathtt{pos}(y, lxk) - 1 = T_{lxk}(y) - 1 = T_{lxk}(y)\{\!\{i \leftarrow T_k(t)\}\!\}$. The last equality holds because $\mathtt{pos}(y, lxk) > i$.
  - $y \notin lk$. Then $T_{lk}(y) = \mathcal{O}(y) + |lk| = \mathcal{O}(y) + |lxk| - 1 = T_{lxk}(y) - 1 = T_{lxk}(y)\{\!\{i \leftarrow T_k(t)\}\!\}$. The last equality holds because $\mathcal{O}(y) + |lxk| > i$.
- $s = f(s_1, \ldots, s_n)$. Then we have

$$\begin{array}{rcl} T_{lk}(s\{x \leftarrow t\}) & = & f(T_{lk}(s_1\{x \leftarrow t\}), \ldots, T_{lk}(s_n\{x \leftarrow t\})) \\ & =_{i.h.} & f(T_{lxk}(s_1)\{\!\{i \leftarrow T_k(t)\}\!\}, \ldots, T_{lxk}(s_n)\{\!\{i \leftarrow T_k(t)\}\!\}) \\ & = & T_{lxk}(s)\{\!\{i \leftarrow T_k(t)\}\!\}. \end{array}$$

- $s = \xi y.(s_1, \ldots, s_n)$. Note that since $s\{x \leftarrow t\}$ is defined by hypothesis we know that $y \notin \mathrm{FV}(t)$ and $y \neq x$ for otherwise $s\{x \leftarrow t\}$ would not be defined. Then we have

$$\begin{array}{rcl} T_{lk}(s\{x \leftarrow t\}) & = & \xi(T_{ylk}(s_1\{x \leftarrow t\}), \ldots, T_{ylk}(s_n\{x \leftarrow t\})) \\ & =_{i.h.} & \xi(T_{ylxk}(s_1)\{\!\{i+1 \leftarrow T_k(t)\}\!\}, \ldots, T_{ylxk}(s_n)\{\!\{i+1 \leftarrow T_k(t)\}\!\}) \\ & = & T_{lxk}(s)\{\!\{i \leftarrow T_k(t)\}\!\}. \end{array}$$

■

LEMMA A.2 (T is compositional w.r.t. valuations)
Let M be a pre-metaterm, l a label of binder indicators, and suppose

1. $\mathcal{WF}_l(M)$,
2. $\theta = (\theta_v, \theta_t)$ is a valuation such that θ_v is injective on the bound o-metavariables, and

3. θ is safe for M.
   Then for every label k we have $T_{\theta(l)k}(\theta M) = T_k(\theta)(T_l(M))$.

PROOF. By induction on the pre-metaterm M. Since $\mathcal{WF}_l(M)$ we have the following cases to consider:

- $M = \alpha \in l$. Then $T_{\theta(l)k}(\theta\alpha) = \texttt{pos}(\theta\alpha, \theta(l)) =_{hyp.2} \texttt{pos}(\alpha, l) = T_k(\theta)(\texttt{pos}(\alpha, l)) = T_k(\theta)(T_l(\alpha))$.
- $M = \widehat{\alpha}$. Then by the definition of valuation and since $\widehat{\alpha} \notin l$ because l is a label of binder indicators:

$$\begin{array}{rcl} T_{\theta(l)k}(\theta\widehat{\alpha}) & = & \left\{\begin{array}{ll} \texttt{pos}(\theta\widehat{\alpha}, k) + |\theta(l)| & \text{if } \theta\widehat{\alpha} \in k \\ \mathcal{O}(\theta\widehat{\alpha}) + |\theta(l)k| & \text{otherwise} \end{array}\right. \\ & = & \left\{\begin{array}{ll} \texttt{S}^{|l|}(\texttt{pos}(\theta\widehat{\alpha}, k)) & \text{if } \theta\widehat{\alpha} \in k \\ \texttt{S}^{|l|}(\mathcal{O}(\theta\widehat{\alpha}) + |k|) & \text{otherwise} \end{array}\right. \\ & = & T_k(\theta)(\texttt{S}^{|l|}(\widehat{\alpha})) \\ & = & T_k(\theta)(T_l(\widehat{\alpha})). \end{array}$$

- $M = X$. Then $T_{\theta(l)k}(\theta X) = T_k(\theta)(X_l) = T_k(\theta)(T_l(X))$.
- $M = \xi\alpha.(M_1, \dots, M_n)$ (the case where $M = f(M_1, \dots, M_n)$ is similar). Then we have

$$\begin{array}{rcl} T_{\theta(l)k}(\theta\xi\alpha.(M_1, \dots, M_n)) & = & \xi(T_{\theta(\alpha)\theta(l)k}(\theta M_1), \dots, T_{\theta(\alpha)\theta(l)k}(\theta M_n)) \\ & =_{i.h.} & \xi(T_k(\theta)(T_{\alpha l}(M_1)), \dots, T_k(\theta)(T_{\alpha l}(M_n))) \\ & = & T_k(\theta)(T_l(\xi\alpha.(M_1, \dots, M_n))). \end{array}$$

- $M = M_1[\alpha \leftarrow M_2]$. Then we have

$$\begin{array}{rcl} T_{\theta(l)k}(\theta(M_1[\alpha \leftarrow M_2])) & = & T_{\theta(l)k}(\theta M_1\{\theta_v(\alpha) \leftarrow \theta M_2\}) \\ & =_{Lemma\ 4.11} & T_{\theta(\alpha)\theta(l)k}(\theta M_1)\{\!\{1 \leftarrow T_{\theta(l)k}(\theta M_2)\}\!\} \\ & =_{i.h.} & T_k(\theta)(T_{\alpha l}(M_1))\{\!\{1 \leftarrow T_k(\theta)(T_l(M_2))\}\!\} \\ & = & T_k(\theta)(T_{\alpha l}(M_1)[T_l(M_2)]) \\ & = & T_k(\theta)(T_l(M_1[\alpha \leftarrow M_2])). \end{array}$$

  Note that since θ is safe for M we may apply Lemma 4.11 with $l = \epsilon$. Indeed, $\theta_v(\alpha) \notin \theta(l)$ and $\text{FV}(\theta M_2) \cap \theta(l) = \emptyset$ for $l = \epsilon$.

∎

LEMMA A.3
Let k, k' be labels of binder indicators, l a label of variables and θ be an injective function on the set of binder indicators. Then for every $t \in \mathsf{T}$, every $p \geq 0$, every $x_1, \dots, x_p$, if for every $z \in \text{FV}(t) \setminus \{x_1, \dots, x_p\}$ we have $z \in \theta(k)$ iff $z \in \theta(k')$, then

$$\mathit{Value}^p(k, T_{x_1 \dots x_p \theta(k) l}(t)) = \mathit{Value}^p(k', T_{x_1 \dots x_p \theta(k') l}(t)).$$

PROOF. We use induction on t.

- $t = x$. We have the following further cases to consider:
  – $x = x_i$ with $1 \leq i \leq p$. Then

$$\begin{array}{l} \mathit{Value}^p(k, T_{x_1 \dots x_p \theta(k) l}(x)) = \\ \mathit{Value}^p(k, i) = i = \mathit{Value}^p(k', i) = \\ \mathit{Value}^p(k', T_{x_1 \dots x_p \theta(k') l}(x)). \end{array}$$

  – $x \in \theta(k) \cap \theta(k')$ and the previous case does not hold. Let $i = \texttt{pos}(x, \theta(k))$ and $j = \texttt{pos}(x, \theta(k'))$. Then $\texttt{at}(k, i) = \texttt{at}(k', j)$ by injectivity of θ and we have

$$\begin{array}{rcl} \mathit{Value}^p(k, T_{x_1 \dots x_p \theta(k) l}(x)) & = & \mathit{Value}^p(k, p + i) \\ & = & \texttt{at}(k, i) \\ & = & \texttt{at}(k', j) \\ & = & \mathit{Value}^p(k', p + j) \\ & = & \mathit{Value}^p(k', T_{x_1 \dots x_p \theta(k') l}(x)). \end{array}$$

– $x \in l$ and the previous cases do not hold. Then for some i with $1 \leq i \leq |l|$ we have

$$\begin{array}{rcl} \mathit{Value}^p(k, T_{x_1 \dots x_p \theta(k) l}(x)) & = & \mathit{Value}^p(k, p + |\theta(k)| + i) \\ & = & x_i \\ & = & \mathit{Value}^p(k', p + |\theta(k')| + i) \\ & = & \mathit{Value}^p(k', T_{x_1 \dots x_p \theta(k') l}(x)). \end{array}$$

– $x \notin \{x_1, \dots, x_p\}$, $x \notin \theta(k) \cup \theta(k')$, $x \notin l$. Then we have

$$\begin{array}{rcl} \mathit{Value}^p(k, T_{x_1 \dots x_p \theta(k) l}(x)) & = & \mathit{Value}^p(k, \mathcal{O}(x) + p + |\theta(k) l|) \\ & = & x_{\mathcal{O}(x) + |l|} \\ & = & \mathit{Value}^p(k', \mathcal{O}(x) + p + |\theta(k') l|) \\ & = & \mathit{Value}^p(k', T_{x_1 \dots x_p \theta(k') l}(x)). \end{array}$$

- $t = f(t_1, \dots, t_n)$. We have $\mathit{Value}^p(k, T_{x_1 \dots x_p \theta(k) l}(t_i)) = \mathit{Value}^p(k', T_{x_1 \dots x_p \theta(k') l}(t_i))$ by induction hypothesis so the property trivially holds.
- $t = \xi x.(t_1, \dots, t_n)$. Then we have

$$\begin{array}{rl} & \mathit{Value}^p(k, T_{x_1 \dots x_p \theta(k) l}(\xi x.(t_1, \dots, t_n))) \\ = & \mathit{Value}^p(k, \xi(T_{x x_1 \dots x_p \theta(k) l}(t_1), \dots, T_{x x_1 \dots x_p \theta(k) l}(t_n))) \\ = & \xi(\mathit{Value}^{p+1}(k, T_{x x_1 \dots x_p \theta(k) l}(t_1)), \dots, \mathit{Value}^{p+1}(k, T_{x x_1 \dots x_p \theta(k) l}(t_n))) \\ =_{i.h.} & \xi(\mathit{Value}^{p+1}(k', T_{x x_1 \dots x_p \theta(k') l}(t_1)), \dots, \mathit{Value}^{p+1}(k', T_{x x_1 \dots x_p \theta(k') l}(t_n))) \\ = & \mathit{Value}^p(k', T_{x_1 \dots x_p \theta(k') l}(\xi x.(t_1, \dots, t_n))). \end{array}$$

∎

B Compositionality of U

We show that the translation U from indices to names is compositional w.r.t. de Bruijn substitutions and valuations.

LEMMA B.1 (U is compositional w.r.t. de Bruijn substitution)
Let $a, b \in \mathsf{T}_{dB}$, l and k labels of variables with $|l| = i - 1$, x a variable such that $x \notin lk \cup S$. Then $U^S_{lk}(a\{\!\{i \leftarrow b\}\!\}) =_\alpha U^S_{lxk}(a)\{x \leftarrow U^S_k(b)\}$, assuming both sides of the equation are defined.

PROOF. The proof is by induction on a.

- $a = n$. We have three further cases to consider:

– $n < i$. Then we reason as follows:

$$\begin{array}{rcll} U^S_{lk}(a\{\!\{i \leftarrow b\}\!\}) & = & U^S_{lk}(n) & \\ & = & \mathtt{at}(lk, n) & \\ & = & \mathtt{at}(lxk, n) & \\ & = & \mathtt{at}(lxk, n)\{x \leftarrow U^S_k(b)\} & (x \notin l) \\ & = & U^S_{lxk}(n)\{x \leftarrow U^S_k(b)\}. & \end{array}$$

– $n > i$. Then since $U^S_{lk}(a\{\!\{i \leftarrow b\}\!\}) = U^S_{lk}(n-1)$ we consider two further cases:

∗ $n - 1 \leq |lk|$. We reason as follows:

$$\begin{array}{rcll} U^S_{lk}(n-1) & = & \mathtt{at}(lk, n-1) & \\ & = & \mathtt{at}(lxk, n) & \\ & = & \mathtt{at}(lxk, n)\{x \leftarrow U^S_k(b)\} & (x \notin k) \\ & = & U^S_{lxk}(n)\{x \leftarrow U^S_k(b)\}. & \end{array}$$

∗ $n - 1 > |lk|$. We reason as follows:

$$\begin{array}{rcll} U^S_{lk}(n-1) & = & x_{n-1-|lk|} & \\ & = & x_{n-|lxk|} & \\ & = & x_{n-|lxk|}\{x \leftarrow U^S_k(b)\} & (x \notin S) \\ & = & U^S_{lxk}(n)\{x \leftarrow U^S_k(b)\}. & \end{array}$$

– $n = i$. Then on the one hand we have $U^S_{lk}(i\{\!\{i \leftarrow b\}\!\}) = U^S_{lk}(\mathcal{U}^i_0(b))$. And on the other $U^S_{lxk}(i)\{x \leftarrow U^S_k(b)\} = x\{x \leftarrow U^S_k(b)\} = U^S_k(b)$. Lemma 5.9 concludes this case.

- $a = f(a_1, \ldots, a_n)$. We use the induction hypothesis.
- $a = \xi(a_1, \ldots, a_n)$. Then we reason as follows:

$$\begin{array}{l}
U^S_{lk}(a\{\!\{i \leftarrow b\}\!\}) \\
= \quad (z \notin lk \cup S \text{ by Definition 5.1}) \\
\xi z.(U^S_{zlk}(a_1\{\!\{i+1 \leftarrow b\}\!\}), \ldots, U^S_{zlk}(a_n\{\!\{i+1 \leftarrow b\}\!\})) \\
=_\alpha \quad (y \text{ fresh}) \\
\xi y.(U^S_{zlk}(a_1\{\!\{i+1 \leftarrow b\}\!\})\{z \leftarrow y\}, \ldots, U^S_{zlk}(a_n\{\!\{i+1 \leftarrow b\}\!\})\{z \leftarrow y\}) \\
=_\alpha \quad (Lemma\ C.1) \\
\xi y.(U^S_{ylk}(a_1\{\!\{i+1 \leftarrow b\}\!\}), \ldots, U^S_{ylk}(a_n\{\!\{i+1 \leftarrow b\}\!\})) \\
=_\alpha \quad (i.h.) \\
\xi y.(U^S_{ylxk}(a_1)\{x \leftarrow U^S_k(b)\}, \ldots, U^S_{ylxk}(a_n)\{x \leftarrow U^S_k(b)\}) \\
=_\alpha \quad (\text{see below}) \\
\xi y.(U^S_{(z'lxk)\{z' \leftarrow y\}}(a_1)\{x \leftarrow U^S_k(b)\}, \ldots, U^S_{(z'lxk)\{z' \leftarrow y\}}(a_n)\{x \leftarrow U^S_k(b)\}) \\
=_\alpha \quad (Lemma\ C.1) \\
\xi y.(U^S_{z'lxk}(a_1)\{z' \leftarrow y\}\{x \leftarrow U^S_k(b)\}, \ldots, U^S_{z'lxk}(a_n)\{z' \leftarrow y\}\{x \leftarrow U^S_k(b)\}) \\
=_\alpha \quad (\text{Subst.Lemma}) \\
\xi y.(U^S_{z'lxk}(a_1)\{x \leftarrow U^S_k(b)\}\{z' \leftarrow y\}, \ldots, U^S_{z'lxk}(a_n)\{x \leftarrow U^S_k(b)\}\{z' \leftarrow y\}) \\
=_\alpha \\
\xi z'.(U^S_{z'lxk}(a_1)\{x \leftarrow U^S_k(b)\}, \ldots, U^S_{z'lxk}(a_n)\{x \leftarrow U^S_k(b)\}) \\
= \\
U^S_{lxk}(a)\{x \leftarrow U^S_k(b)\}.
\end{array}$$

Note that since the RHS of the equation to prove is defined, from the last line above we learn that $z' \notin lxk \cup S$ and $z' \notin \text{FV}(U^S_k(b))$.

'Subst.Lemma' refers to the substitution lemma for the λ-calculus [2], which is also valid for our restricted notion of substitution and reads as follows: $s\{x \leftarrow t\}\{y \leftarrow u\} = s\{y \leftarrow u\}\{x \leftarrow t\{y \leftarrow u\}\}$ if $x \notin \text{FV}(u)$ for distinct variables x and y, and both sides of the equation together with the term $t\{y \leftarrow u\}$ are defined.

∎

LEMMA B.2 (U is compositional w.r.t. valuations)
Consider a de Bruijn valuation $\kappa = (\kappa_i, \kappa_t)$, a de Bruijn pre-metaterm A, a finite set of variables S, a variable assignment θ_v verifying the hypothesis in Definition 5.6, a label of binder indicators l and a label of variables k. If the following conditions hold:

1. κ is valid,
2. κA is defined,
3. θ_v is defined over l and the bound o-metavariables in $U_l(A)$,
4. θ_v is injective on the bound o-metavariables,
5. $\text{NAMES}(\text{FI}(\kappa A) \backslash\!\backslash |\theta_v(l)k|) \subseteq S$, and
6. $\mathcal{WF}_l(A)$.

Then, $U^S_{\theta_v(l)k}(\kappa A) =_\alpha U_{(\theta_v,S,k)}(\kappa)U_l(A)$.

PROOF. By induction on A. Below we shall use LHS and RHS to denote the left- and right-hand side respectively, of the equation to prove.

- $A = X_h$. Since $\mathcal{WF}_l(X_h)$ by Hypothesis 6, we have that $h = l$ and so $LHS = U^S_{\theta_v(l)k}(\kappa X_l)$. And on the other hand

$$\begin{array}{rcll}
RHS & = & U_{(\theta_v,S,k)}(\kappa)(U_l(X_l)) & \\
& = & U_{(\theta_v,S,k)}(\kappa)X & \\
& = & U^S_{\theta_v(l')k}(\kappa X_{l'}) & (\text{ with } X_{l'} \in Dom(\kappa)).
\end{array}$$

Then since κ is valid (Hypothesis 1) we may apply Lemma 5.7 to conclude.

- $A = \mathtt{S}^j(\widehat{\alpha})$. Since $\mathcal{WF}_l(\mathtt{S}^j(\widehat{\alpha}))$ holds by Hypothesis 6, then $j = |l|$. We have

$$\begin{array}{rcl} LHS & = & U^S_{\theta_v(l)k}(\kappa\mathtt{S}^{|l|}(\widehat{\alpha})) \\ & = & U^S_{\theta_v(l)k}(\mathtt{S}^{|l|}(\kappa_i(\widehat{\alpha}))) \\ & = & \begin{cases} \mathtt{at}(k, \kappa_i(\widehat{\alpha})) & \text{if } \kappa_i(\widehat{\alpha}) \leq |k| \\ x_{\kappa_i(\widehat{\alpha})-|k|} & \text{otherw. with } x_{\kappa_i(\widehat{\alpha})-|k|} \in S. \end{cases} \end{array}$$

On the other hand we have

$$\begin{array}{rcl} RHS & = & U_{(\theta_v,S,k)}(\kappa)(U_l(\mathtt{S}^{|l|}(\widehat{\alpha}))) \\ & = & U_{(\theta_v,S,k)}(\kappa)(\widehat{\alpha}) \\ & = & \theta_v(\widehat{\alpha}). \end{array}$$

We can conclude $LHS = RHS$ because θ_v satisfies the requirements of Definition 5.6.

- $A = \mathtt{S}^j(1)$. Since $\mathcal{WF}_l(\mathtt{S}^j(1))$ holds by the Hypothesis 6, $j + 1 \leq |l|$. Thus,

$$\begin{array}{rcl} LHS & = & U^S_{\theta_v(l)k}(\kappa\mathtt{S}^j(1)) \\ & = & U^S_{\theta_v(l)k}(\mathtt{S}^j(1)) \\ & = & \mathtt{at}(\theta_v(l), j+1) \\ & = & \theta_v(\mathtt{at}(l, j+1)). \end{array} \qquad \begin{array}{rcl} RHS & = & U_{(\theta_v,S,k)}(\kappa)(U_l(\mathtt{S}^j(1))) \\ & = & U_{(\theta_v,S,k)}(\kappa)(\mathtt{at}(l, j+1)) \\ & = & \theta_v(\mathtt{at}(l, j+1)). \end{array}$$

- $A = \xi(A_1, \ldots, A_n)$. Then we reason as follows

$$\begin{array}{rl} & RHS \\ = & U_{(\theta_v,S,k)}(\kappa)(U_l(\xi(A_1, \ldots, A_n))) \\ = & U_{(\theta_v,S,k)}(\kappa)(\xi\alpha.(U_{\alpha l}(A_1), \ldots, U_{\alpha l}(A_n)))\ (\alpha \notin l \text{ and } \mathcal{WF}_{\alpha l}(A_i) \text{ for all } 1 \leq i \leq n) \\ = & \xi\theta_v(\alpha).(U_{(\theta_v,S,k)}(\kappa)(U_{\alpha l}(A_1)), \ldots, U_{(\theta_v,S,k)}(\kappa)(U_{\alpha l}(A_n)))). \end{array}$$

In order to apply the induction hypothesis we need to verify the hypothesis for A_i. The Hypothesis 1 holds by definition and the Hypothesis 2 is evident since κA is defined. Hypothesis 3 holds since by hypothesis, θ_v is defined over l and the bound o-metavariables in $U_l(A) = \xi\alpha.(U_{\alpha l}(A_1), \ldots, U_{\alpha l}(A_n))$, hence it is defined over the bound o-metavariables in $U_{\alpha l}(A_i) \cup \alpha l$ for $1 \leq i \leq n$. Regarding Hypothesis 5, namely $\textsc{Names}(\mathrm{FI}(\kappa A_i) \setminus\!\!\setminus |\theta_v(\alpha l)k|) \subseteq S$; but this is evident by Hypothesis 5 for A and the general fact that $\mathrm{FI}(\xi(a_1, \ldots, a_n)) \setminus\!\!\setminus n = \mathrm{FI}(a_1, \ldots, a_n) \setminus\!\!\setminus n + 1$. Hypothesis 6 is also true because when translating the de Bruijn pre-metaterm A we choose α verifying the condition $\mathcal{WF}_{\alpha l}(A_i)$. Thus applying the induction hypothesis we have:

$$\begin{array}{l} RHS \\ = \\ \xi\theta_v(\alpha).(U^S_{\theta_v(\alpha l)k}(\kappa A_1), \ldots, U^S_{\theta_v(\alpha l)k}(\kappa A_n)) \\ =_\alpha \quad (z' \text{ does not occur in } U^S_{\theta_v(\alpha l)k}(\kappa A_i)) \\ \xi z'.((U^S_{\theta_v(\alpha l)k}(\kappa A_1))\{\theta_v(\alpha) \leftarrow z'\}, \ldots, (U^S_{\theta_v(\alpha l)k}(\kappa A_n))\{\theta_v(\alpha) \leftarrow z'\}) \\ =_\alpha \quad (Lemma\ C.1) \\ \xi z'.(U^S_{(\theta_v(\alpha l)k)\{\theta_v(\alpha)\leftarrow z'\}}(\kappa A_1), \ldots, U^S_{(\theta_v(\alpha l)k)\{\theta_v(\alpha)\leftarrow z'\}}(\kappa A_n)) \\ =_\alpha \quad (\theta_v \text{ injective and Definition 5.6}) \\ \xi z'.(U^S_{z'\theta_v(l)k}(\kappa A_1), \ldots, U^S_{z'\theta_v(l)k}(\kappa A_n)) \\ = \quad (z \notin \theta_v(l)k \cup S) \\ \xi z'.(U^S_{(z\theta_v(l)k)\{z\leftarrow z'\}}(\kappa A_1), \ldots, U^S_{(z\theta_v(l)k)\{z\leftarrow z'\}}(\kappa A_n)) \\ =_\alpha \quad (Lemma\ C.1) \\ \xi z'.(U^S_{z\theta_v(l)k}(\kappa A_1)\{z \leftarrow z'\}, \ldots, U^S_{z\theta_v(l)k}(\kappa A_n)\{z \leftarrow z'\}) \\ =_\alpha \\ \xi z.(U^S_{z\theta_v(l)k}(\kappa A_1), \ldots, U^S_{z\theta_v(l)k}(\kappa A_n)). \end{array}$$

On the other hand we have

$$\begin{array}{rcl} LHS & = & U^S_{\theta_v(l)k}(\kappa(\xi(A_1, \ldots, A_n))) \\ & = & U^S_{\theta_v(l)k}(\xi(\kappa A_1, \ldots, \kappa A_n)) \\ & = & \xi z.(U^S_{z\theta_v(l)k}(\kappa A_1), \ldots, U^S_{z\theta_v(l)k}(\kappa A_n)) \quad (z \notin \theta_v(l)k \cup S). \end{array}$$

- $A = f(A_1, \ldots, A_n)$. Then we have

$$\begin{array}{rcl} LHS & = & U^S_{\theta_v(l)k}(\kappa(f(A_1, \ldots, A_n))) \\ & = & U^S_{\theta_v(l)k}(f(\kappa A_1, \ldots, \kappa A_n)) \\ & = & f(U^S_{\theta_v(l)k}(\kappa A_1), \ldots, U^S_{\theta_v(l)k}(\kappa A_n)). \end{array}$$

$$\begin{array}{rcl} RHS & = & U_{(\theta_v,S,k)}(\kappa)(U_l(f(A_1, \ldots, A_n))) \\ & = & U_{(\theta_v,S,k)}(\kappa)(f(U_l(A_1), \ldots, U_l(A_n))) \\ & = & f(U_{(\theta_v,S,k)}(\kappa)U_l(A_1), \ldots, U_{(\theta_v,S,k)}(\kappa)U_l(A_n)). \end{array}$$

We can immediately conclude by induction hypothesis.

- $A = A_1[A_2]$. Then we have

$$\begin{array}{rcl} LHS & = & U^S_{\theta_v(l)k}(\kappa(A_1[A_2])) \\ & = & U^S_{\theta_v(l)k}(\kappa A_1 \{\!\{1 \leftarrow \kappa A_2\}\!\}). \end{array}$$

On the other hand we have

$$\begin{array}{lll} & RHS & \\ = & U_{(\theta_v,S,k)}(\kappa)(U_l(A_1[A_2])) & \\ = & U_{(\theta_v,S,k)}(\kappa)(U_{\alpha l}(A_1)[\alpha \leftarrow U_l(A_2)]) & (\star) \\ = & (U_{(\theta_v,S,k)}(\kappa)(U_{\alpha l}(A_1)))\{\theta_v(\alpha) \leftarrow U_{(\theta_v,S,k)}(\kappa)(U_l(A_2))\} & \\ =_\alpha & U^S_{\theta_v(\alpha l)k}(\kappa A_1)\{\theta_v(\alpha) \leftarrow U^S_{\theta_v(l)k}(\kappa A_2)\}. & (i.h.) \end{array}$$

where in $(\star)$, $\alpha \notin l$ is such that $\mathcal{WF}_{\alpha l}(A_1)$.

Note that in the last step the inductive hypothesis may be applied by the same reasons we used in the case of the binder.

Now, since θ_v is injective and satisfies the conditions of Definition 5.6, then $\theta_v(\alpha) \notin S \cup \theta_v(l)k$ and we can then conclude by applying Lemma 5.10.

∎

C From de Bruijn valuations to correct valuations

In this section we prove that the translation of a valid de Bruijn valuation (Definition 5.6) does not depend on the choice of the t-metavariable.

LEMMA C.1 (Renaming and the $U^\bullet_\bullet(\bullet)$ translation)
Let l be a label of variables, z and y be two variables, S be a set of variables and a be a de Bruijn term such that:

1. $z \in l$ and $z \notin S$,
2. y does not occur in $U^S_l(a)$, and
3. $\text{NAMES}(\text{FI}(a) \setminus\!\!\setminus |l|) \subseteq S$.

Then we have $U^S_l(a)\{z \leftarrow y\} =_\alpha U^S_{l\{z \leftarrow y\}}(a)$.

PROOF. The condition $\text{NAMES}(\text{FI}(a) \setminus\!\!\setminus |l|) \subseteq S$ is required for $U^S_l(a)$ and $U^S_{l\{z\leftarrow y\}}(a)$ to be defined. The proof proceeds by induction on a. The case where a is of the form $f(a_1, \ldots, a_n)$ follows from the induction hypothesis so we consider the remaining ones.

- $a = n$. We have two further cases to consider:
  - $1 \leq n \leq |l|$. Then $U^S_l(n)\{z \leftarrow y\} = \mathtt{at}(l, n)\{z \leftarrow y\} = \mathtt{at}(l\{z \leftarrow y\}, n) = U^S_{l\{z\leftarrow y\}}(n)$.
  - $n > |l|$. Then since $z \notin S$ we have $U^S_l(n)\{z \leftarrow y\} = x_{n-|l|}\{z \leftarrow y\} = x_{n-|l|} = x_{n-|l\{z\leftarrow y\}|} = U^S_{l\{z\leftarrow y\}}(n)$.
- $a = \xi(a_1, \ldots, a_n)$. Then we reason as follows:

$$
\begin{array}{l}
U^S_l(a)\{z \leftarrow y\} \\
= \\
\xi x.(U^S_{xl}(a_1), \ldots, U^S_{xl}(a_n))\{z \leftarrow y\} \\
= \;\; (z \in l \text{ hence } z \neq x \text{ and } y \notin U^S_l(a)) \\
\xi x.(U^S_{xl}(a_1)\{z \leftarrow y\}, \ldots, U^S_{xl}(a_n)\{z \leftarrow y\}) \\
=_\alpha \;\; (i.h.) \\
\xi x.(U^S_{xl\{z\leftarrow y\}}(a_1), \ldots, U^S_{xl\{z\leftarrow y\}}(a_n)) \\
=_\alpha \;\; (v \text{ fresh}) \\
\xi v.(U^S_{xl\{z\leftarrow y\}}(a_1)\{x \leftarrow v\}, \ldots, U^S_{xl\{z\leftarrow y\}}(a_n)\{x \leftarrow v\}) \\
=_\alpha \;\; (i.h.) \\
\xi v.(U^S_{vl\{z\leftarrow y\}\{x\leftarrow v\}}(a_1), \ldots, U^S_{vl\{z\leftarrow y\}\{x\leftarrow v\}}(a_n)) \\
=_\alpha \;\; (x \neq y \text{ and } x \notin l) \\
\xi v.(U^S_{vl\{z\leftarrow y\}}(a_1), \ldots, U^S_{vl\{z\leftarrow y\}}(a_n)) \\
=_\alpha \\
\xi v.(U^S_{(wl\{z\leftarrow y\})\{w\leftarrow v\}}(a_1), \ldots, U^S_{(wl\{z\leftarrow y\})\{w\leftarrow v\}}(a_n)) \\
=_\alpha \;\; (i.h.) \\
\xi v.(U^S_{wl\{z\leftarrow y\}}(a_1)\{w \leftarrow v\}, \ldots, U^S_{wl\{z\leftarrow y\}}(a_n)\{w \leftarrow v\}) \\
=_\alpha \;\; (w \notin l\{z \leftarrow y\} \cup S) \\
\xi w.(U^S_{wl\{z\leftarrow y\}}(a_1), \ldots, U^S_{wl\{z\leftarrow y\}}(a_n)) \\
= \\
U^S_{l\{z\leftarrow y\}}(a).
\end{array}
$$

Since the translation function on the LHS and RHS of the equation to prove may have chosen different variables for the ξ binder we relate them through a fresh variable v.

∎

LEMMA C.2
Let a and b be de Bruijn terms, l and l' labels of binder indicators and α a binder indicator. Then for $j \geq 0$ we have: $\mathit{Value}^{j+1}(l, a) = \mathit{Value}^{j+1}(l', b)$ implies $\mathit{Value}^j(\alpha l, a) = \mathit{Value}^j(\alpha l', b)$.

PROOF. By induction on a.

- $a = m$. Since $\mathit{Value}^{j+1}(l, m) = \mathit{Value}^{j+1}(l', b)$ we have $b = n$ for some index n. We proceed by cases:
  - $m \leq j + 1$. Then since $\mathit{Value}^{j+1}(l, m) = m = \mathit{Value}^{j+1}(l', n)$ by Definition 3.12 we have $n = m$ and therefore $\mathit{Value}^j(\alpha l, m) = \mathit{Value}^j(\alpha l', n)$.
  - $m > j + 1$. We have two different cases:
    - $m-(j+1) \leq |l|$. Then by hypothesis we have $\mathit{Value}^{j+1}(l, m) = \mathtt{at}(l, m-(j+1)) = \mathit{Value}^{j+1}(l', n)$, and hence $0 < n-(j+1) \leq |l'|$ and $\mathtt{at}(l, m-(j+1)) = \mathtt{at}(l', n-(j+1))$. Therefore $\mathit{Value}^j(\alpha l, m) = \mathit{Value}^j(\alpha l', n)$ since $1 < m - j \leq |\alpha l|$ and $1 < n - j \leq |\alpha l'|$.
    - $m - (j + 1) > |l|$. Then by hypothesis we have $\mathit{Value}^{j+1}(l, m) = x_{m-(j+1)-|l|} = \mathit{Value}^{j+1}(l', n)$, and hence $n - (j + 1) > |l'|$ and $m - (j + 1) - |l| = n - (j + 1) - |l'|$. Therefore $\mathit{Value}^j(\alpha l, m) = x_{m-j-|\alpha l|} = \mathit{Value}^j(\alpha l', n)$.
- $a = f(a_1, \ldots, a_n)$ or $a = \xi(a_1, \ldots, a_n)$. By Definition 3.12 and the hypothesis it must be the case that $b = f(b_1, \ldots, b_n)$ (resp. $b = \xi(b_1, \ldots, b_n)$) and $\mathit{Value}^{j+1}(l, a_i) = \mathit{Value}^{j+1}(l', b_i)$ (resp. $\mathit{Value}^{j+2}(l, a_i) = \mathit{Value}^{j+2}(l', b_i)$) for $1 \leq i \leq n$. By induction hypothesis we can conclude $\mathit{Value}^j(\alpha l, a_i) = \mathit{Value}^j(\alpha l', b_i)$ (resp. $\mathit{Value}^{j+1}(\alpha l, a_i) = \mathit{Value}^{j+1}(\alpha l', b_i)$) and thus $\mathit{Value}^j(\alpha l, a) = \mathit{Value}^j(\alpha l', b)$.

∎

Note that the converse of Lemma C.2 does not hold (for α may already be present in l or l'). Indeed, $\mathit{Value}^0(\alpha\alpha, 2) = \mathit{Value}^0(\alpha\alpha, 1)$, yet $\mathit{Value}^1(\alpha, 2) \neq \mathit{Value}^1(\alpha, 1)$. The value function is used to determine when a de Bruijn valuation is valid or not. It is defined in the $SERS_{dB}$ formalism in order to describe reduction on de Bruijn terms. A natural question which arises is that of the relationship between value equivalent de Bruijn terms considered as named terms via de $U^{\bullet}_{\bullet}(\bullet)$ translation in the $SERS$ formalism. The following lemma investigates this matter.

LEMMA C.3
Let $a, b \in \mathsf{T}_{dB}$, S be a set of variables, l, l' be labels of binder indicators, k a label of variables, and θ_v a variable assignment. If both $U^S_{\theta_v(l)k}(a)$ and $U^S_{\theta_v(l')k}(b)$ are defined, then $\mathit{Value}(l, a) = \mathit{Value}(l', b)$ implies $U^S_{\theta_v(l)k}(a) =_\alpha U^S_{\theta_v(l')k}(b)$.

PROOF. By induction on a.

- $a = m$. Since $\mathit{Value}^0(l, m) = \mathit{Value}^0(l', b)$ we have $b = n$ for some index n. The left-hand side reads

$$U^S_{\theta_v(l)k}(m) \quad = \quad \begin{cases} \mathtt{at}(\theta_v(l), m) & m \leq |l| \\ \mathtt{at}(k, m - |l|) & |l| < m \leq |kl| \\ x_{m-|lk|} & m > |kl| \text{ and } x_{m-|lk|} \in S. \end{cases}$$

  We now consider the following cases:
  - $m \leq |l|$. Then since $\mathit{Value}^0(l, m) = \mathtt{at}(l, m) = \mathit{Value}^0(l', n)$ we have $n \leq |l'|$ and $\mathtt{at}(l, m) = \mathtt{at}(l', n)$. Therefore $U^S_{\theta_v(l')k}(n) = \mathtt{at}(\theta_v(l'), n) = \theta_v(\mathtt{at}(l', n)) = \theta_v(\mathtt{at}(l, m)) = U^S_{\theta_v(l)k}(m)$.
  - $|l| < m \leq |lk|$. Then since $\mathit{Value}^0(l, m) = x_{m-|l|} = \mathit{Value}^0(l', n)$ we have $n > |l'|$ and $x_{m-|l|} = x_{n-|l'|}$. Then $m - |l| = n - |l'|$. Thus $U^S_{\theta_v(l')k}(n) = \mathtt{at}(k, n - |l'|) = \mathtt{at}(k, m - |l|) = U^S_{\theta_v(l)k}(m)$.
  - $m > |lk|$. Then since $\mathit{Value}^0(l, m) = x_{m-|l|} = \mathit{Value}^0(l', n)$ we have $n > |l'|$ and $x_{m-|l|} = x_{n-|l'|}$. Then $m - |l| = n - |l'|$ and since $m > |lk|$ we also have $n > |l'k|$. Thus $U^S_{\theta_v(l')k}(n) = x_{n-|l'k|} = x_{m-|lk|} = U^S_{\theta_v(l)k}(m)$.
- $a = f(a_1, \ldots, a_n)$. Since $\mathit{Value}^0(l, a) = \mathit{Value}^0(l', b)$ we have $b = f(b_1, \ldots, b_n)$ and $\mathit{Value}^0(l, a_i) = \mathit{Value}^0(l', b_i)$ for $1 \leq i \leq n$. Then by the induction hypothesis we have $U^S_{\theta_v(l)k}(a_i) =_\alpha U^S_{\theta_v(l')k}(b_i)$ and hence $U^S_{\theta_v(l)k}(a) =_\alpha U^S_{\theta_v(l')k}(b)$.
- $a = \xi(a_1, \ldots, a_n)$. Since $\mathit{Value}^0(l, a) = \mathit{Value}^0(l', b)$ we have $b = \xi(b_1, \ldots, b_n)$ and $\mathit{Value}^1(l, a_i) = \mathit{Value}^1(l', b_i)$ for $1 \leq i \leq n$. Then $\mathit{Value}^0(\beta l, a_i) = \mathit{Value}^0(\beta l', b_i)$ holds by Lemma C.2, where in particular we can take β to be a fresh o-metavariable such that θ_v is undefined on β. Let us extend the function θ_v to β by defining $\theta_v(\beta) \stackrel{\text{def}}{=} z$, where z is a fresh variable such that $z \notin \theta_v(l)\theta_v(l')k \cup S$. Then since $U^S_{z\theta_v(l)k}(a_i)$ and $U^S_{z\theta_v(l')k}(b_i)$ are defined we can apply the induction hypothesis to get $U^S_{z\theta_v(l)k}(a_i) = U^S_{\theta_v(\beta l)k}(a_i) =_\alpha U^S_{\theta_v(\beta l')k}(b_i) = U^S_{z\theta_v(l')k}(b_i)$ for $1 \leq i \leq n$.

  We now reason as follows:

$$\begin{array}{lll}
 & U^S_{\theta_v(l)k}(\xi(a_1, \ldots, a_n)) & \\
= & \xi x.(U^S_{x\theta_v(l)k}(a_1), \ldots, U^S_{x\theta_v(l)k}(a_n)) & (x \notin \theta_v(l)k \cup S) \\
=_\alpha & \xi z.(U^S_{x\theta_v(l)k}(a_1)\{x \leftarrow z\}, \ldots, U^S_{x\theta_v(l)k}(a_n)\{x \leftarrow z\}) & \\
= & \xi z.(U^S_{z\theta_v(l)k}(a_1), \ldots, U^S_{z\theta_v(l)k}(a_n)) & (\mathit{Lemma\ C}.1) \\
=_\alpha & \xi z.(U^S_{z\theta_v(l')k}(b_1), \ldots, U^S_{z\theta_v(l')k}(b_n)) & (\mathit{i.h.}) \\
= & \xi z.(U^S_{y\theta_v(l')k}(b_1)\{y \leftarrow z\}, \ldots, U^S_{y\theta_v(l')k}(b_n)\{y \leftarrow z\}) & (\mathit{Lemma\ C}.1) \\
=_\alpha & \xi y.(U^S_{y\theta_v(l')k}(b_1), \ldots, U^S_{y\theta_v(l')k}(b_n)) & (y \notin \theta_v(l')k \cup S \\
 & & \text{by Definition 5.1}) \\
= & U^S_{\theta_v(l')k}(\xi(b_1, \ldots, b_n)) &
\end{array}$$

∎

Note that, in general, the converse of Lemma C.3 does not hold. Indeed it suffices to consider $k = \epsilon$, $l = \alpha$, $l' = \beta$, $a = 1$, $b = 1$, $S = \emptyset$ and the variable assignment $\theta_v\alpha = \theta_v\beta = x$. Then $U^S_x(a) = x = U^S_x(b)$ but $\mathit{Value}(\alpha, 1) = \alpha \neq \beta = \mathit{Value}(\beta, 1)$.

We can now show that the translation of de Bruijn valuations is correct in the sense mentioned above. This is completed in Section 3 as Lemma 5.7.

D From valid de Bruijn valuations to admissible valuations

This subsection shows that if we depart from a valid valuation κ in the de Bruijn indices setting and we translate this valuation as dictated by Definition 5.6 into a valuation in the SERS setting, then we obtain an admissible valuation. In other words, the resulting valuation is safe (Definition 2.20) and verifies the path condition (Definition 2.21).

A word on notation: we shall use $\delta, \delta_i, \ldots$ to denote o-metavariables (that is, δ may either be a pre-bound o-metavariable such as α, or a pre-free metavariable such as $\widehat{\alpha}$).

LEMMA D.1 (valid de Bruijn valuations translate to safe valuations)
Let κ be a valid de Bruijn valuation for a rewrite rule (L, R), θ_v a variable assignment satisfying the requirements of Definition 5.6, S a finite set of variables, k a label of variables, and $U(L, R) = (G, D)$ the translation of (L, R). If the following conditions hold

1. $U_{(\theta_v,S,k)}(\kappa)$ is defined for *all* metavariables of G and D, and
2. θ_v is injective on the set of bound o-metavariables of (G, D)
   then $U_{(\theta_v,S,k)}(\kappa)$ is safe for (G, D).

PROOF. Recall that $U_{(\theta_v,S,k)}(\kappa) \stackrel{\text{def}}{=} (\theta_v, \theta_t)$ where:

$$\theta_t X \quad \stackrel{\text{def}}{=} \quad U^S_{\theta_v(l)k}(\kappa X_l) \qquad \text{for any } X_l \in Dom(\kappa).$$

In what follows we shall abbreviate $U_{(\theta_v,S,k)}(\kappa)$ with θ' for the sake of readability. Suppose that θ' is not safe for (G, D), then unwanted variable capture arises in $\theta' D$ (since the metasubstitution operator does not occur on the LHS of a rewrite rule, no renaming problems can arise in G). Thus there exist metaterms M_1 and M_2 and a formal parameter α such that

- $M_1[\alpha \leftarrow M_2]$ occurs in D (or equivalently $D = C[M_1[\alpha \leftarrow M_2]]$ for some metacontext C). The metaterm D may be depicted as in Figure 8(a) where l_1 denotes the parameter path of the metacontext C.
- θ' is defined for M_1 and M_2.
- We have either
  Case 1: $\theta'\alpha = x$ and $x \in BV(\theta' M_1)$ for some variable x, or
  Case 2: $\theta'\alpha = x$, $x \in$ FV$(\theta' M_1)$, $y \in$ BV$(\theta' M_1)$ and $y \in$ FV$(\theta' M_2)$ for some variables $x \neq y$.

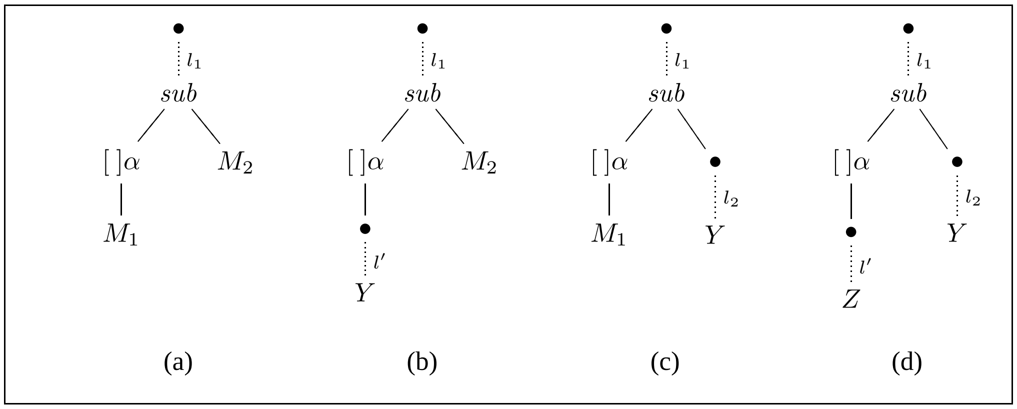

FIGURE 8. Tree form for D

Before proceeding let us show the following fact:

FACT D.2
If $z \in$ BV$(\theta' M_1)$, then z cannot be bound by a formal parameter $\beta \in \alpha l_1$ (that is, for all $\beta \in \alpha l_1$ we have $\theta'(\beta) \neq z$).

PROOF. Suppose that for some $\beta \in \alpha l_1$ we have $\theta'(\beta) = z$. Thus by definition of θ' we have $\theta_v(\beta) = z$. Let us consider the *bound* occurrence of z in $\theta' M_1$. There are two possibilities:

1. z comes from the instantiation of a bound o-metavariable, so that $z = \theta'(\beta')$ for some formal parameter β' in M_1. Now since D is a well-formed pre-metaterm we must have $\beta \neq \beta'$. But $\theta'(\beta')$ is equal to $\theta_v(\beta')$ by definition, so that θ_v assigns the same value, namely z, to two different bound o-metavariables β and β' of D, thus contradicting Hypothesis 2.
2. z comes from the instantiation of a t-metavariable, so that $z \in BV(t)$ with $t = \theta' Y$ for some t-metavariable Y occurring in M_1. By Definition 5.6 we have
$$t = U^S_{\theta_v(l)k}(\kappa Y_l)$$
   for some t-metavariable Y_l occurring in $Dom(\kappa)$ with $l = l'\alpha l_1$ (see Figure 8(b)). Therefore by definition of the term translation function $U^{\bullet}_{\bullet}(\bullet)$ (Definition 5.1) the variable z cannot be a candidate for binding in κY_l since it already occurs in the label $\theta_v(l)k$, indeed, $\beta \in l$ and $\theta_v(\beta) = z$.

∎

Now, suppose that **case 1** holds, that is, $\theta'\alpha = x$ and $x \in \mathrm{BV}(\theta' M_1)$ for some variable x. This leads immediately to a contradiction with the Fact D.2 recently proved. Therefore, let us continue with the proof of the lemma by considering that **case 2** holds, and in particular, let us see where the free variable occurrence of y comes from in $\theta' M_2$. We have two possible cases:

1. There is an occurrence of an o-metavariable δ in M_2 such that $\theta_v(\delta) = y$. As observed above (Fact D.2), since y may not be bound by a formal parameter in l_1 (i.e. there is no $\beta \in l_1$ with $\theta_v(\beta) = y$) then $\delta \notin l_1$. Thus $\delta = \widehat{\beta}'$ for some free o-metavariable $\widehat{\beta}'$ or else the pre-metaterm D would not be a metaterm. So then $\widehat{\beta}'$ is a free o-metavariable in D and thus by Hypothesis 1 $U_{(\theta_v,S,k)}(\kappa)$ is defined on $\widehat{\beta}'$. Now, the assignment θ_v satisfies the requirements of Definition 5.6, so that in particular by the second requirement we must have $\theta_v(\widehat{\beta}') = y \in S \cup k$.

   We now analyse where the bound occurrence of y comes from in $\theta' M_1$ in order to arrive at a contradiction. Here too we have two cases to consider:

   (a) $y = \theta_v(\beta'')$ for some formal parameter β'' occuring in M_1. Now, $\theta_v(\beta'') \notin S \cup k$ since θ_v satisfies the first requirement of Definition 5.6 by hypothesis, so that we arrive at a contradiction.

   (b) y comes from instantiating some t-metavariable Y in M_1, i.e. $y \in \mathrm{BV}(\theta' Y)$ for some t-metavariable Y in M_1 (Figure 8(b)). Thus there is a t-metavariable Y_l with $l = l'\alpha l_1$ in $Dom(\kappa)$, a simple label k' and an index m such that $U^S_{k'\theta_v(l'\alpha l_1)k}(m) = y = \mathtt{at}(k'\theta_v(l'\alpha l_1)k, m)$.
   Now since y is bound in $\theta' Y$ we have $m \leq |k'\theta_v(l')|$. But then by definition of $U^{\bullet}_{\bullet}(\bullet)$ we have $y \notin S \cup \theta_v(\alpha l_1)k$, in other words, y cannot have been used as a candidate variable for binding. In particular, $y \notin S \cup k$. This is a contradiction since we already know that $y \in S \cup k$.

2. There is an occurrence of a t-metavariable Y in M_2 such that $y \in \mathrm{FV}(\theta' Y)$. Then there is an occurrence of Y_l in $Dom(\kappa)$ with $l = l_2 l_1$ such that $y \in \mathrm{FV}(U^S_{\theta_v(l_2 l_1)k}(\kappa Y_l))$ where l_1 is the label 'above' M_2 (Figure 8(c)). Note that since for this occurrence of y we have $y \in \mathrm{FV}(\theta' M_2)$ then we must have that $y \in S$ or $y \in \theta_v(l_1)k$.

   We now analyse where the bound occurrence of y comes from in $\theta' M_1$. Here too we have two cases to consider:

   (a) $y = \theta_v(\beta'')$ for some formal parameter β'' occurring in M_1. If $y \in S$ or $y \in k$ we arrive at a contradiction with the fact that θ_v verifies the first requirement of Definition 5.6 (saying that $\theta_v(\beta'') \notin S \cup k$). Moreover, if $y \in \theta_v(l_1)$ we contradict Fact D.2.

   (b) y comes from instantiating some o-metavariable Z in M_1, i.e. $y \in \mathrm{BV}(\theta' Z)$ for some t-metavariable Z in M_1 (Figure 8(d)). Thus there is an o-metavariable Z_l in $Dom(\kappa)$ with $l = l'\alpha l_1$, a simple label k' and an index m such that $U^S_{k'\theta_v(l'\alpha l_1)k}(m) = y = \mathtt{at}(k'\theta_v(l'\alpha l_1)k, m)$.
   Now since y is bound in $\theta' M_1$ we have $m \leq |k'\theta_v(l')|$. But then by definition of $U^{\bullet}_{\bullet}(\bullet)$ we have $y \notin S \cup \theta_v(\alpha l_1)k$. In particular, $y \notin S \cup \theta_v(l_1)k$. This is a contradiction since we already know that $y \in S$ or $y \in \theta_v(l_1)k$.

∎

LEMMA D.3 (From valid de Bruijn valuations to admissible valuations)
Consider the following: a valid de Bruijn valuation κ for a rewrite rule (L, R), θ_v a variable assignment verifying the hypothesis in Definition 5.6, and $U(L, R) = (G, D)$ the translation of (L, R). If the following conditions hold:
1. $U_{(\theta_v,S,k)}(\kappa)$ is defined for *all* metavariables of G and D, and
2. θ_v is injective on the set of bound o-metavariables of (G, D),
then $U_{(\theta_v,S,k)}(\kappa)$ is admissible for (G, D).

PROOF. We shall abbreviate $U_{(\theta_v,S,k)}(\kappa)$ by θ' in order to improve readability. Since by Lemma D.1 we have that θ' is safe then by Definition 2.22 we have still to check the following properties:

- θ verifies the path condition for X in (G, D): if no t-metavariable occurs more than once then the property is trivial so let us suppose that there is a t-metavariable X in (G, D) occurring at two different positions p and p'. Let us take any variable $x \in \mathrm{FV}(\theta' X)$ and let l and l' be the parameter paths of p and p' in the trees corresponding to G or D. Suppose $\theta' X = U_{(\theta_v,S,k)}(\kappa) X \overset{\mathrm{def}}{=} U^S_{\theta_v(l)k}(\kappa X_l)$. Then since κ is valid by Lemma 5.7 $U^S_{\theta_v(l)k}(\kappa X_l) =_\alpha U^S_{\theta_v(l')k}(\kappa X_{l'})$. As a consequence, the set of free variables of both terms is the same. Now, to show that θ verifies the path condition for X in (G, D) let us suppose that $x \in \theta_v(l)$. Since the o-metavariables in l are bound in the rule (G, D), and θ_v is defined for all the metavariables of (G, D) by Hypothesis 1, then by the requirements of Definition 5.6 $x \notin S \cup k$. Now, since x is free in $U^S_{\theta_v(l')k}(\kappa X_{l'})$ then x must be in $S \cup \theta_v(l')k$, which implies that x is necessarily in $\theta_v(l')$. This allows us to conclude that θ verifies the path condition for X in (G, D).
- If the pre-bound o-metavariables α and β occur in (G, D) with $\alpha \neq \beta$, then $\theta_v\alpha \neq \theta_v\beta$: this property trivially holds by Hypothesis 2.

∎

E Preserving properties

We start by a technical lemma that will be used later.

LEMMA E.1
Let $M \in \mathsf{PMT}$ without occurrences of t-metavariables, and let

1. $k\alpha l$ be a simple label, k' a label such that $|k| = |k'|$, α' a pre-bound o-metavariable,
2. β a bound o-metavariable such that it does not occur in $U_{k\alpha l}(T_{k'\alpha' l}(M))$, and
3. $\mathcal{WF}_{k'\alpha' l}(M)$ hold.

Then $(U_{k\alpha l}(T_{k'\alpha' l}(M))) \ll\alpha \leftarrow \beta\gg =_v U_{k\beta l}(T_{k'\alpha' l}(M))$.

PROOF. By induction on M. Let $k = \beta_1 \dots \beta_n$ and $k' = \beta'_1 \dots \beta'_n$. By Hypothesis 3 we have the following cases to consider:

- $M = \alpha'' \in k'$ and hence $\alpha'' = \beta'_j$ for some $1 \le j \le n$. Then we have
$$U_{k\alpha l}(T_{k'\alpha' l}(M)) \ll\alpha \leftarrow \beta\gg = \beta_j \ll\alpha \leftarrow \beta\gg =_{hyp.1} \beta_j = U_{k\beta l}(T_{k'\alpha' l}(M)).$$
- $M = \alpha'' \in l$ and $\alpha'' \notin k'$. Then we have
$$U_{k\alpha l}(T_{k'\alpha' l}(M)) \ll\alpha \leftarrow \beta\gg = \alpha'' \ll\alpha \leftarrow \beta\gg =_{hyp.1} \alpha'' = U_{k\beta l}(T_{k'\alpha' l}(M)).$$
- $M = \alpha'$ and $\alpha' \notin k'$. Then we have
$$U_{k\alpha l}(T_{k'\alpha' l}(M)) \ll\alpha \leftarrow \beta\gg = \alpha \ll\alpha \leftarrow \beta\gg = \beta = U_{k\beta l}(T_{k'\alpha' l}(M)).$$
- $M = \widehat{\alpha}$. Then we have
$$U_{k\alpha l}(T_{k'\alpha' l}(M)) \ll\alpha \leftarrow \beta\gg = \widehat{\alpha} \ll\alpha \leftarrow \beta\gg = \widehat{\alpha} = U_{k\beta l}(T_{k'\alpha' l}(M)).$$
- $M = f(M_1, \dots, M_n)$. Then we use the induction hypothesis.
- $M = \xi\alpha''.(M_1, \dots, M_n)$. We reason as follows:
$$\begin{array}{l}(U_{k\alpha l}(T_{k'\alpha' l}(M)))) \ll\alpha \leftarrow \beta\gg = \\ (\xi\beta'.(U_{\beta' k\alpha l}(T_{\alpha'' k'\alpha' l}(M_1)), \dots, U_{\beta' k\alpha l}(T_{\alpha'' k'\alpha' l}(M_n)))) \ll\alpha \leftarrow \beta\gg\end{array}$$
for $\beta' \notin k\alpha l$ such that $\mathcal{WF}_{\beta' k\alpha l}(T_{\alpha'' k'\alpha' l}(M_i))$ holds for $1 \le i \le n$. Since $\beta \ne \beta'$ by Hypothesis 2, continue
$$\begin{array}{l}(\xi\beta'.(U_{\beta' k\alpha l}(T_{\alpha'' k'\alpha' l}(M_1)), \dots, U_{\beta' k\alpha l}(T_{\alpha'' k'\alpha' l}(M_n)))) \ll\alpha \leftarrow \beta\gg \\ = \\ \xi\beta'.((U_{\beta' k\alpha l}(T_{\alpha'' k'\alpha' l}(M_1))) \ll\alpha \leftarrow \beta\gg, \dots, (U_{\beta' k\alpha l}(T_{\alpha'' k'\alpha' l}(M_n))) \ll\alpha \leftarrow \beta\gg) \\ =_v (i.h.) \\ \xi\beta'.(U_{\beta' k\beta l}(T_{\alpha'' k'\alpha' l}(M_1)), \dots, U_{\beta' k\beta l}(T_{\alpha'' k'\alpha' l}(M_n))) \\ \overset{\text{def}}{=} \\ U_{k\beta l}(T_{k'\alpha' l}(\xi\alpha''.(M_1, \dots, M_n)))\end{array}$$
- $M = M_1[\alpha' \leftarrow M_2]$. Similar to the previous case.

∎

LEMMA E.2
Let $M \in \mathsf{PMT}$ and l a simple label. If $\mathcal{WF}_l(M)$ then $U_l(T_l(M)) =_v M$.

PROOF. By induction on M.

- $M = \alpha$. Then since $\mathcal{WF}_l(M)$ we have $\alpha \in l$ and thus $U_l(T_l(\alpha)) = U_l(\mathtt{pos}(\alpha, l)) = \alpha$.
- $M = \widehat{\alpha}$. Then $U_l(T_l(\widehat{\alpha})) = U_l(\mathsf{S}^{|l|}(\widehat{\alpha})) = \widehat{\alpha}$.
- $M = X$. Then $U_l(T_l(X)) = U_l(X_l) = X$.
- $M = f(M_1, \dots, M_n)$. We use the induction hypothesis.
- $M = \xi\alpha.(M_1, \dots, M_n)$. We reason as follows:
$$\begin{array}{l}U_l(T_l(\xi\alpha.(M_1, \dots, M_n))) = \\ U_l(\xi(T_{\alpha l}(M_1), \dots, T_{\alpha l}(M_n))) = \\ \xi\beta.(U_{\beta l}(T_{\alpha l}(M_1)), \dots, U_{\beta l}(T_{\alpha l}(M_n)))\end{array}$$
where $\beta \notin l$. We have two further cases to consider:

1. There are no occurrences of t-metavariables in M. Now if $\beta = \alpha$ we conclude by using the induction hypothesis so let us assume then that $\beta \neq \alpha$.

$$\begin{array}{l}
\xi\beta.(U_{\beta l}(T_{\alpha l}(M_1)), \ldots, U_{\beta l}(T_{\alpha l}(M_n))) \\
=_v \quad (\beta' \text{ not in } U_{\beta l}(T_{\alpha l}(M_i))) \\
\xi\beta'.(U_{\beta l}(T_{\alpha l}(M_1)) \ll\beta\leftarrow\beta'\gg, \ldots, U_{\beta l}(T_{\alpha l}(M_n)) \ll\beta\leftarrow\beta'\gg) \\
= \quad (\mathit{Lemma\ E}.1) \\
\xi\beta'.(U_{\beta' l}(T_{\alpha l}(M_1)), \ldots, U_{\beta' l}(T_{\alpha l}(M_n))) \\
= \quad (\mathit{Lemma\ E}.1) \\
\xi\beta'.(U_{\alpha l}(T_{\alpha l}(M_1)) \ll\alpha\leftarrow\beta'\gg, \ldots, U_{\alpha l}(T_{\alpha l}(M_n)) \ll\alpha\leftarrow\beta'\gg) \\
=_v \quad (i.h.) \\
\xi\alpha.(U_{\alpha l}(T_{\alpha l}(M_1)), \ldots, U_{\alpha l}(T_{\alpha l}(M_n))) \\
=_v \\
\xi\alpha.(M_1, \ldots, M_n).
\end{array}$$

2. There is an occurrence of a t-metavariable X in M. In this case since $U_l(T_l(M))$ is defined we observe that it must be that $\beta = \alpha$. Indeed, we have that $X_{l'\alpha l}$ occurs in $T_l(M)$ for some label l'. Hence when translating this metavariable to the de Bruijn setting we arrive at $U_{l''\beta l}(X_{l'\alpha l})$, which is defined only for $l''\beta l = l'\alpha l$. Therefore, $\beta = \alpha$ and we use the induction hypothesis.

- $M = M_1[\alpha \leftarrow M_2]$. We proceed as above.

∎

LEMMA E.3
Let $t \in \mathsf{T}$ such that $\mathrm{FV}(t) \subseteq S \cup l$ for l any label and S a finite set of variables. Then $U^S_l(T_l(t)) =_\alpha t$.

PROOF. By induction on the structure of t.

- $t = x$. Then there are two cases to consider:
  - $x \in l$. Then $U^S_l(T_l(x)) = U^S_l(\mathtt{pos}(x, l)) = x$.
  - $x \notin l$. Then $U^S_l(T_l(x)) = U^S_l(\mathcal{O}(x) + |l|)$. By hypothesis $x \in S \cup l$ so that $x \in S$ and then $U^S_l(\mathcal{O}(x) + |l|) = x_{\mathcal{O}(x)} = x$.
- $t = f(t_1, \ldots, t_n)$. Then $U^S_l(T_l(t)) = f(U^S_l(T_l(t_1)), \ldots, U^S_l(T_l(t_n))) =_\alpha f(t_1, \ldots, t_n)$. The last step holds by induction hypothesis.
- $t = \xi x.(t_1, \ldots, t_n)$. Then

$$\begin{array}{lll}
 & U^S_l(T_l(\xi x.(t_1, \ldots, t_n))) & \\
= & U^S_l(\xi(T_{xl}(t_1), \ldots, T_{xl}(t_n))) & \\
= & \xi z.(U^S_{zl}(T_{xl}(t_1)), \ldots, U^S_{zl}(T_{xl}(t_n))) & (z \notin S \cup l) \\
=_\alpha & \xi z'.(U^S_{zl}(T_{xl}(t_1))\{z \leftarrow z'\}, \ldots, U^S_{zl}(T_{xl}(t_n))\{z \leftarrow z'\}) & (z' \text{ not in } U^S_{z'l}(T_{xl}(t_i))) \\
= & \xi z'.(U^S_{z'l}(T_{xl}(t_1)), \ldots, U^S_{z'l}(T_{xl}(t_n))) & (\mathit{Lemma\ C}.1) \\
= & \xi z'.(U^S_{xl}(T_{xl}(t_1))\{x \leftarrow z'\}, \ldots, U^S_{xl}(T_{xl}(t_n))\{x \leftarrow z'\}) & (\mathit{Lemma\ C}.1) \\
=_\alpha & \xi x.(U^S_{xl}(T_{xl}(t_1)), \ldots, U^S_{xl}(T_{xl}(t_n))) & (x \notin l) \\
=_\alpha & \xi x.(t_1, \ldots, t_n) & (i.h.).
\end{array}$$

∎

LEMMA E.4
Let $A \in \mathsf{PMT}_{dB}$ and l be any simple label. If $\mathcal{WF}_l(A)$ then $T_l(U_l(A)) = A$.

PROOF. By induction on A. ∎

LEMMA E.5
Let a be a de Bruijn term and l be any simple label such that $\textsc{Names}(\mathrm{FI}(a) \setminus\!\!\setminus |l|) \subseteq S$ and $l \cap S = \emptyset$. Then $T_l(U^S_l(a)) = a$.

PROOF. By induction on the structure of a. ∎

PROOF. [Of **Lemma 6.1**]
1. Let $M \in \mathsf{PMT}$ such that $\mathcal{WF}(M)$. Then $U(T(M)) =_v M$. The proof is a direct consequence of Lemma E.2.
2. Let $t \in \mathsf{T}$. Then $U(T(t)) =_\alpha t$. The proof is a direct consequence of Lemma E.3
3. Let $A \in \mathsf{PMT}_{dB}$. If $\mathcal{WF}(A)$ then $T(U(A)) = A$. The proof is a direct consequence of Lemma E.4.
4. Let $a \in \mathsf{T}_{dB}$. Then $T(U(a)) = a$. The proof is a direct consequence of Lemma E.5.

∎

Received May 2002