A de Bruijn notation for higher-order rewriting

Eduardo Bonelli[1,2], Delia Kesner[2], Alejandro Ríos[1]

[1] Departamento de Computación - Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Pabellón I, Ciudad Universitaria (1428), Buenos Aires, Argentina. {ebonelli,rios}@dc.uba.ar
[2] LRI (UMR 8623) - Bât 490, Université de Paris-Sud, 91405 Orsay Cedex, France. kesner@lri.fr

Abstract. We propose a formalism for higher-order rewriting in de Bruijn notation. This notation not only is used for terms (as usually done in the literature) but also for metaterms, which are the syntactical objects used to express general higher-order rewrite systems. We give formal translations from higher-order rewriting with names to higher-order rewriting with de Bruijn indices, and vice-versa. These translations can be viewed as an *interface* in programming languages based on higher-order rewrite systems, and they are also used to show some properties, namely, that both formalisms are operationally equivalent, and that confluence is preserved when translating one formalism into the other.

1 Introduction

Higher-order (term) rewriting concerns the transformation of terms in the presence of binding mechanisms for variables. Implementing higher-order rewriting requires, beforehand, taking care of a complex notion of substitution operation and of renaming of bound variables (α-conversion). As a paradigmatic example, the β-reduction axiom of λ-calculus [12], expressed $(\lambda x.M)N \longrightarrow_\beta M\{x \leftarrow N\}$, may be interpreted as: the result of executing function $\lambda x.M$ over argument N is obtained by substituting N for all (free) occurrences of x in M. Any implementation of higher-order rewriting must include instructions for computing this substitution. Although from the meta-level the execution of a substitution is atomic, the cost of computing it highly depends on the form of the terms, specially if unwanted variable capture conflicts must be avoided by renaming bound variables. De Bruijn indices take care of renaming because the representation of variables by indices completely eliminates unwanted 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 [13]) but also on de Bruijn metaterms, which are the syntactical objects used to express any general higher-order rewrite system formulated in a de Bruijn context.

Many higher-order rewrite systems (HORS) exist and work in the area is currently very active.Klop introduced in [15] the Combinatory Reduction Systems (*CRS*), an early reference for Khasidashvili's Expression Reduction Systems (*ERS*) is [14], Nipkow introduces Higher-Order Rewrite Systems (*HRS*) in [16], Wolfram defines Higher-Order Term Rewrite Systems (see [23]), van Oostrom and van Raamsdonk introduce Higher-Order term Rewriting Systems in [21]. We choose in this work to use *ERS* because their syntax and semantics are simple and natural (they allow for example to write β-reduction in λ-calculus as usual while *CRS* do not) and the correspondence between *ERS* and *HRS* has already been established [22]. We shall begin with (a slightly simplified version of) the *ERS* formalism, that we shall call *SERS* (S for simplified) and introduce the de Bruijn index based higher-order rewrite system $SERS_{DB}$.

Our work is the first step in the construction of a formal interpretation of higher-order rewriting via a *first-order* theory. This kind of simulation would be possible with the aid of *explicit substitutions* which implement the complex substitution operation at the object level, i.e. the substitution operation is incorporated into the calculus and additional reduction rules cater for its behavior. Thus the study of the (non trivial) process of substitution may be developed in a uniform and simple manner via a first-order mechanism when de Bruijn notation is considered. Indeed, this work follows, in some sense, the lines of [9] which interprets higher-order formalisms/problems into their respective first-order ones.

Our formalism is developed in order to be used as an *interface* of a programming language based on higher-order rewriting. Of course, the use of variable name based formalisms are necessary for humans to interact with computers in a user-friendly way. Clearly 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 shall be the detailed study of the relationship between SERS and $SERS_{DB}$. The definitions developed in this paper give formal translators from higher-order syntax with names to higher-order syntax with indices and vice-versa. These translators are extensions to higher-order of the translations studied in [13] and presented in [5].

As regards existing higher-order rewrite formalisms based on de Bruijn index notation and/or explicit substitutions to the best of the authors' knowledge there are but two: *Explicit CRS* [2] and Explicit Reduction Systems (*XRS*) [17]. In [2] explicit substitutions a la λx [20, 1] are added to the *CRS* formalism as a first step towards using higher-order rewriting with explicit substitutions for modeling the evaluation of functional programs in a faithful way. Since this is done in a variable name setting α-conversion must be dealt with as in *CRS*. Pagano's *XRS* constitutes the first HORS which fuses de Bruijn index notation and explicit substitutions. It is presented as a generalization of the $\lambda\sigma_{\Uparrow}$-calculus [6] but no connection has been established between *XRS* and well-known systems such as *CRS*, *ERS* and *HRS*. Indeed, it is not clear at all how some seemingly natural rules expressible, say, in the *ERS* formalism, may be written in an *XRS*. As an example, consider a rewrite system for logical expressions such that if $imply(e_1, e_2)$ reduces to the constant *true* then e_1 logically implies e_2 in classical first-order predicate logic. A possible rewrite rule could be:

$$(imp) \quad imply(\exists x \forall y M, \forall y \exists x M) \longrightarrow true$$

A naïve attempt might consider the rewrite rule

$$(imp_{db}) \quad imply(\exists \forall M, \forall \exists M) \longrightarrow true$$

as a possible representation of this rule in the *XRS* formalism, but it does not have the desired effect since $\exists\forall M$ and $\forall\exists M$ correspond to $\exists x \forall y M$ and $\forall x \exists y M$ but $\forall x \exists y M$ and $\forall y \exists x M$ are not equivalent. Note that regardless of the fact that *XRS* incorporate explicit substitutions, this problem arises already at the level of de Bruijn notation. Another example of interest is:

$$(\eta) \quad \lambda x.(Mx) \longrightarrow M \text{ if } x \text{ is not free in } M$$

which is usually expressed in a de Bruijn based system with explicit substitutions

$$(\eta_{db}) \quad \lambda(M1) \longrightarrow N \text{ if } M =_{\mathcal{C}} N[\uparrow]$$

where $M =_{\mathcal{C}} N$ means that M and N are equivalent modulo the theory of explicit substitutions $\mathcal{C}$. Neither the (imp) rule nor (η_{db}) is possible in the *XRS* formalism so that they do not have in principle the same expressive power as *ERS*. We shall propose de Bruijn based HORS that will allow such rules to be faithfully represented.

We can then summarize the contribution of this paper by the following points:

- We propose a general de Bruijn notation for higher-order syntax.
- We bridge the gap between higher-order rewriting with names and higher-order rewriting with indices showing that they are operationally equivalent. This is the first step in establishing a connection between higher and first-order rewrite formalisms.
- Our formalism suggests a first-order tool to implement HORS. This tool would incorporate de Bruijn notation and explicit substitutions in a very general form but in contrast to [17] it would be possible to represent *all* the higher-order systems used in practice.

The rest of the paper is organized as follows. Section 2 introduces our work and study scenario, the *SERS* formalism. The de Bruijn based formalism $SERS_{DB}$ is defined in Section 3. Section 4 takes a close-up view of the relationship, via appropriate translations, between the formalisms *SERS* and $SERS_{DB}$. Also, preservation of confluence is considered. Finally, we conclude.

By lack of space we only present here an extended abstract, and therefore proofs, auxiliary lemmas and standard definitions are only hinted or just omitted, but the interested reader will find full details in [4].

2 Simplified Expression Reduction Systems

We introduce the variable name based higher-order rewrite formalism *SERS*.

2.1 Metaterms and terms

Definition 1 (Signature). *Consider the denumerable and disjoint infinite sets:*

- $\mathcal{V} = \{x_1, x_2, x_3, \ldots\}$ *a set of* variables*, arbitrary variables are denoted* $x, y, \ldots$
- $\mathcal{B}_{\mathcal{MV}} = \{\alpha_1, \alpha_2, \alpha_3, \ldots\}$ *a set of* pre-bound o-metavariables *(o for object), denoted* $\alpha, \beta, \ldots$
- $\mathcal{F}_{\mathcal{MV}} = \{\widehat{\alpha_1}, \widehat{\alpha_2}, \widehat{\alpha_3}, \ldots\}$ *a set of* pre-free o-metavariables*, denoted* $\widehat{\alpha}, \widehat{\beta}, \ldots$
- $\mathcal{T}_{\mathcal{MV}} = \{X_1, X_2, X_3, \ldots\}$ *a set of* t-metavariables *(t for term), denoted* $X, Y, Z, \ldots$
- $\mathcal{F} = \{f_1, f_2, f_3, \ldots\}$ *a set of* function symbols *equipped with a fixed (possibly zero) arity, denoted* $f, g, h, \ldots$
- $\mathcal{B} = \{\lambda_1, \lambda_2, \lambda_3, \ldots\}$ *a set of* binder symbols *equipped with a fixed (non-zero) arity, denoted* $\lambda, \mu, \nu, \xi, \ldots$

The union of $\mathcal{B}_{\mathcal{MV}}$ *and* $\mathcal{F}_{\mathcal{MV}}$ *is the set of o-metavariables of the signature. We shall use* $\delta, \delta_i, \ldots$ *to denote o-metavariables. When speaking of metavariables without further qualifiers we refer to o-metavariables and t-metavariables. Since all these alphabets are ordered, given any symbol* s *we shall denote* $\mathcal{O}(s)$ *its position in the corresponding alphabet. Thus for example if the variables are ordered* $x_1, x_2, \ldots$ *and if* $x \equiv x_i$ *then* $\mathcal{O}(x) = i$*, where the symbol* $\equiv$ *denotes syntactical identity.*

Definition 2 (Labels). *A* label *is a finite sequence of symbols of an alphabet. We shall 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* n*-th 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 $l \equiv s_1 \ldots s_n$, then $\theta(l)$ denotes the label $\theta(s_1)\ldots\theta(s_n)$. In the sequel, we may use a label as a set (e.g. $S \cap l$ denotes the intersection of a set S with the set containing the elements of l i.e. the underlying set determined by l) if no confusion arises. A* simple label *is a label without repeated symbols.*

Definition 3 (Pre-metaterms). *The set of SERS* pre-metaterms*, denoted $\mathcal{PMT}$, is defined by:*

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

Arities are supposed to be respected, i.e. a pre-metaterm like $f(M_1,\ldots,M_n)$ (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 shall use $M, N, M_i, \ldots$ to denote pre-metaterms. The symbol $.[. \leftarrow .]$ in the pre-metaterm $M[\alpha \leftarrow M]$ is called metasubstitution operator*. The o-metavariable α in a pre-metaterm of the form $\xi\alpha.(M,\ldots,M)$ or $M[\alpha \leftarrow M]$ 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* operates *as a binder in that ithas binding power) but* not *a binder symbol since it is not an element of $\mathcal{B}$.*

The main difference between *SERS* and *ERS* is that in the latter binders and metasubstitutions are defined on *multiple* o-metavariables. Indeed, 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 hypothesis 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 usual (parallel) substitution. It is well known that multiple substitution can be simulated by simple substitution. Indeed, $M[\alpha_1\ldots\alpha_k \leftarrow M_1,\ldots,M_k]$ can be simulated by 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 simulated with the pre-metaterm $\xi\alpha_1.(\xi\alpha_2.(\ldots\xi\alpha_k.(M_1,\ldots,M_m)))$. There is also a notion of scope indicator in *ERS*, used to express in which arguments of the quantifier variables are bound. Scope indicators shall not be considered in *SERS* since they do not seem to contribute to the expressive power of *ERS* [1]

A pre-metaterm M has an associated *tree*, denoted *tree*(M), in the following way:

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

A position is a label over the alphabet $\mathbb{N}$. As for the empty label, we use ϵ to denote the empty word in $\mathbb{N}^*$. Given a pre-metaterm N appearing in M, the set of *occurrences* of N in M is the set of positions of *tree*(M) where N occurs (positions in trees are defined as usual). The *parameter path* of an occurrence p in a tree T is the list containing all the (pre-bound) o-metavariables occuring in the path from p to the root of T.

Example 1. Consider the pre-metaterm $M \equiv f((\xi\alpha.(X))Y)$. Then X occurs at position 00 and Y at position 1. The parameter path of 00 (or just X) is α and the parameter path of 1 (or just

[1] Indeed, if we consider the *ERS*-term $\xi\alpha.(M_1, M_2, M_3)$ with scope indicator $(1,3)$, which specifies that α is only bound by M_1 and M_3, then it can be simulated by the *SERS*-term $f(\mu\alpha.(M_1), M_2, \mu\alpha.(M_3))$, with $f \in \mathcal{F}$ and $\mu \in \mathcal{B}$.

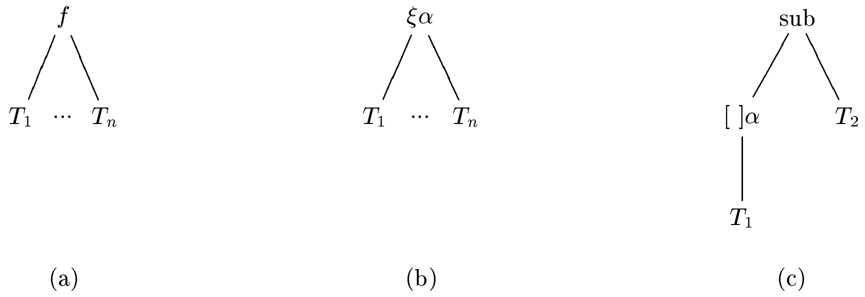


Fig. 1. Pre-metaterms as Trees

Y) is ϵ. Consider the pre-metaterm $M \equiv \mu\beta.(X[\alpha \leftarrow \lambda\gamma.(g(\beta, g(\gamma, Z)))])$. Then the submetaterm $\lambda\gamma.(g(\beta, g(\gamma, Z)))$ occurs at position 01 and $g(\gamma, Z)$ occurs at position 0101; the parameter path of 01 (or just $\lambda\gamma.(g(\beta, g(\gamma, Z))))$ is β, the parameter path of 0101 is $\gamma\beta$.

The following definition introduces the set of *metaterms*, which 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, i.e., all the formal parameters appearing in the same path of a pre-metaterm must be different. Also, it guarantees that metavariables in $\mathcal{B}_{\mathcal{MV}}$ only occur bound.

Definition 4 (Metaterms). *A pre-metaterm M is a metaterm, denoted by $\mathcal{WF}(M)$, iff the predicate $\mathcal{WF}_\epsilon(M)$ holds, where $\mathcal{WF}_l(M)$ is defined as follows:*

- $\mathcal{WF}_l(\alpha)$ *iff* $\alpha \in l$
- $\mathcal{WF}_l(\widehat{\alpha})$ *and* $\mathcal{WF}_l(X)$ *are always true*
- $\mathcal{WF}_l(f(M_1, \ldots, M_n))$ *iff for all* $1 \leq i \leq n$ *we have* $\mathcal{WF}_l(M_i)$
- $\mathcal{WF}_l(\xi\alpha.(M_1, \ldots, M_n))$ *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. The pre-metaterms $f(\xi\alpha.(X), \lambda\alpha.(Y))$, $f(\widehat{\beta}, \lambda\alpha.(Y))$ and $g(\lambda\alpha.(\xi\beta.(h)))$ are metaterms, while the pre-metaterms $f(\alpha, \xi\alpha.(X))$ and $f(\widehat{\beta}, \lambda\alpha.(\xi\alpha.(X)))$ are not.

In the sequel, pre-bound (free) o-metavariables occurring in metaterms shall simply be referred to as bound (free) o-metavariables. As we shall see, metaterms are used to specify rewrite rules.

Definition 5 (Free metavariables of pre-metaterms). *Let M be a pre-metaterm, then $FMVar(M)$ denotes the* set of free metavariables *of M, which is defined as follows:*

$$\begin{array}{ll}
FMVar(X) \stackrel{\text{def}}{=} \{X\} \quad FMVar(\alpha) \stackrel{\text{def}}{=} \{\alpha\} \quad FMVar(\widehat{\alpha}) \stackrel{\text{def}}{=} \{\widehat{\alpha}\} & \\
FMVar(f(M_1, \ldots, M_n)) & \stackrel{\text{def}}{=} \bigcup_{i=1}^{n} FMVar(M_i) \\
FMVar(\xi\alpha.(M_1, \ldots, M_n)) & \stackrel{\text{def}}{=} (\bigcup_{i=1}^{n} FMVar(M_i)) \setminus \{\alpha\} \\
FMVar(M_1[\alpha \leftarrow M_2]) & \stackrel{\text{def}}{=} (FMVar(M_1) \setminus \{\alpha\}) \cup FMVar(M_2)
\end{array}$$

All metavariables which are not free are bound. We use $BMVar(M)$ to denote the bound metavariables of a metaterm M. Note that only o-metavariables may occur bound in a metaterm. We denote the set of metavariables of a metaterm or a pre-metaterm M by $MVar(M)$. Note that if M is a metaterm, then $FMVar(M)$ does not contain pre-bound o-metavariables.

Definition 6 (Terms and contexts). *The set of SERS* terms, *denoted* $\mathcal{T}$, *and contexts are defined by:*

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

where $\Box$ *denotes a "hole". We shall use* $s, t, t_i, \ldots$ *for terms and* C, D *for contexts. We remark that in contrast to other formalisms dealing with higher-order rewriting, here 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.*

With $C[t]$ *we denote the term obtained by replacing the term* t *for the hole* $\Box$ *in the context* C. *Note that this operation may introduce variable capture. We define the* label of a context *as a sequence of variables as follows:*

$$\begin{array}{ll} \texttt{label}(\Box) & \stackrel{\text{def}}{=} \epsilon \\ \texttt{label}(f(t_1,\ldots,C,\ldots,t_n)) & \stackrel{\text{def}}{=} \texttt{label}(C) \\ \texttt{label}(\xi x.(t_1,\ldots,C,\ldots,t_n)) & \stackrel{\text{def}}{=} \texttt{label}(C)x \end{array}$$

For example, the label of the context $C \equiv f(\lambda x.(z, \xi y.(h(y, \Box))))$ is the sequence yx. The label of a context is a notion analogous to that of a parameter path of an occurrence, but defined for terms instead of pre-metaterms and where the only occurrence considered is that of the hole.

Definition 7 (Free variables of terms). *Let* $t \in \mathcal{T}$, *the set* $FV(t)$ *of* free variables *in* t *is defined by:*

$$\begin{array}{ll} FV(x) & \stackrel{\text{def}}{=} \{x\} \\ FV(f(t_1,\ldots,t_n)) & \stackrel{\text{def}}{=} \bigcup_{i=1}^{n} FV(t_i) \\ FV(\xi x.(t_1,\ldots,t_n)) & \stackrel{\text{def}}{=} (\bigcup_{i=1}^{n} FV(t_i)) \setminus \{x\} \end{array}$$

This definition can be extended to contexts by adding the clause $FV(\Box) \stackrel{\text{def}}{=} \emptyset$.

Substitution on terms can be defined as follows:

Definition 8 ((Restricted) substitution of terms). *The* (restricted) substitution *of a term* t *for a variable* x *in a term* s, *denoted* $s\{x \leftarrow t\}$, *is defined:*

$$\begin{array}{lll} x\{x \leftarrow t\} & \stackrel{\text{def}}{=} t & \\ y\{x \leftarrow t\} & \stackrel{\text{def}}{=} y & \textit{if } x \not\equiv y \\ f(s_1,\ldots,s_n)\{x \leftarrow t\} & \stackrel{\text{def}}{=} f(s_1\{x \leftarrow t\},\ldots,s_n\{x \leftarrow t\}) & \\ \xi x.(s_1,\ldots,s_n)\{x \leftarrow t\} & \stackrel{\text{def}}{=} \xi x.(s_1,\ldots,s_n) & \\ \xi y.(s_1,\ldots,s_n)\{x \leftarrow t\} & \stackrel{\text{def}}{=} \xi y.(s_1\{x \leftarrow t\},\ldots,s_n\{x \leftarrow t\}) & \\ & \quad \textit{if } x \not\equiv y, \textit{ and } (y \notin FV(t) \textit{ or } x \notin FV(s)) & \end{array}$$

Thus $.\{. \leftarrow .\}$ denotes the substitution operator on terms but it may *not* apply α-conversion (renaming of bound variables) in order to avoid unwanted variable captures. Therefore this notion of substitution is not defined for all terms (hence its name). When defining the notion of reduction relation on terms induced by rewrite rules we shall take α-conversion into consideration. We may define α-conversion on terms as the smallest reflexive, symmetric and transitive relation closed by contexts verifying the following equality:

$$(\alpha)\ \xi x.(s_1,\ldots,s_n) \equiv_\alpha \xi y.(s_1\{x\leftarrow y\},\ldots,s_n\{x\leftarrow y\}) \quad y \text{ not in } s_1,\ldots,s_n$$

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

The notion of α-conversion for terms has a symmetrical one for pre-metaterms which we call v-*equivalence* (v for variant). The intuitive meaning of two v-equivalent pre-metaterms is that they are able to receive the *same* set of potential "valuations" (c.f. Definition 11). 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 shall be said below.

Definition 9 **(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*[2] *verifying:*

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

where β is a pre-bound o-metavariable which does not occur in $P_1,\ldots,P_n$ in $(v1)$ and does not occur in P_1 in $(v2)$, P_i does not contain t-metavariables for $1\leq i\leq n$, and $P \ll\alpha\leftarrow Q\gg$ is the restricted substitution for pre-metaterms:

$$\begin{array}{lll}
\alpha \ll\alpha\leftarrow Q\gg & \stackrel{\text{def}}{=} Q & \\
\alpha' \ll\alpha\leftarrow Q\gg & \stackrel{\text{def}}{=} \alpha' & \alpha\not\equiv\alpha' \\
\widehat{\alpha'} \ll\alpha\leftarrow Q\gg & \stackrel{\text{def}}{=} \widehat{\alpha'} & \\
X \ll\alpha\leftarrow Q\gg & \stackrel{\text{def}}{=} X & \\
f(M_1,\ldots,M_n) \ll\alpha\leftarrow Q\gg & \stackrel{\text{def}}{=} f(M_1 \ll\alpha\leftarrow Q\gg,\ldots,M_n \ll\alpha\leftarrow Q\gg) & \\
(\xi\alpha.(M_1,\ldots,M_n)) \ll\alpha\leftarrow Q\gg & \stackrel{\text{def}}{=} \xi\alpha.(M_1,\ldots,M_n) & \\
(\xi\alpha'.(M_1,\ldots,M_n)) \ll\alpha\leftarrow Q\gg & \stackrel{\text{def}}{=} \xi\alpha'.(M_1 \ll\alpha\leftarrow Q\gg,\ldots,M_n \ll\alpha\leftarrow Q\gg) & \\
 & \quad \alpha\not\equiv\alpha',\ (\alpha'\notin FMVar(Q) \text{ or } \alpha\notin FMVar(P)) & \\
(M_1[\alpha\leftarrow M_2]) \ll\alpha\leftarrow Q\gg & \stackrel{\text{def}}{=} M_1[\alpha\leftarrow M_2 \ll\alpha\leftarrow Q\gg] & \\
(M_1[\alpha'\leftarrow M_2]) \ll\alpha\leftarrow Q\gg & \stackrel{\text{def}}{=} (M_1 \ll\alpha\leftarrow Q\gg)[\alpha'\leftarrow M_2 \ll\alpha\leftarrow Q\gg] & \\
 & \quad \alpha\not\equiv\alpha',\ (\alpha'\notin FMVar(Q) \text{ or } \alpha\notin FMVar(M_1)) &
\end{array}$$

Example 3. $\lambda\alpha.(\alpha) =_v \lambda\beta.(\beta)$, $\lambda\alpha.(f) =_v \lambda\beta.(f)$, but $\lambda\alpha.(X) \neq_v \lambda\beta.(X)$, $\lambda\beta.(\lambda\alpha.(X)) \neq_v \lambda\alpha.(\lambda\beta.(X))$.

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].

2.2 Reduction

Whereas the rewrite rules are specified by using metaterms, the reduction relation is defined on terms.

[2] Metacontexts are defined analogously to contexts. The notion of "label of a context" is extended to metacontexts as expected.

Definition 10 (*SERS* **rewrite rule).** *An SERS* rewrite rule *is a pair of metaterms* (G, D) *(also written* $G \longrightarrow D$*) such that*

- *the first symbol in* G *is a function symbol or a binder symbol*
- $FMVar(D) \subseteq FMVar(G)$
- G *contains no occurrence of the metasubstitution operator*

Example 4. The $\lambda\mathtt{x}$-calculus [2, 20] is defined by the following *SERS* rewrite rules:

$$\begin{array}{lll}
@(\lambda\alpha.(X), Z) & \longrightarrow_{Beta} & \Sigma(\sigma\alpha.(X), Z) \\
\Sigma(\sigma\alpha.(@(X, Y)), Z) & \longrightarrow_{App} & @(\Sigma(\sigma\alpha.(X), Z), \Sigma(\sigma\alpha.(Y), Z)) \\
\Sigma(\sigma\alpha.(\lambda\beta.(X)), Z) & \longrightarrow_{Lambda} & \lambda\beta.(\Sigma(\sigma\alpha.(X), Z)) \\
\Sigma(\sigma\alpha.(\alpha), Z) & \longrightarrow_{Var1} & Z \\
\Sigma(\sigma\alpha.(\widehat{\beta}), Z) & \longrightarrow_{Var2} & \widehat{\beta}
\end{array}$$

Note that our formalism allows us to specify the Var2 rule as originally done in [20], while formalisms such as *CRS* force one to change this rule to a stronger one, called *gc*, written as $\Sigma(\sigma\alpha.(X), Z) \longrightarrow_{gc} X$, where the admissibility condition on valuations guarantees that if X/t is part of the valuation θ, then $\theta(\alpha)$ cannot be in $FV(t)$.

Example 5. The $\lambda\Delta$-calculus [19] is defined by the following *SERS* rewrite rules:

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

We shall now proceed to define the way in which rewrite rules are instantiated in order to obtain the rewrite relation. 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 the notion of valuation shall be 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. A word on notation: if θ is a (partial) function from a set S_1 to a set S_2 then we use $Dom(\theta)$ to denote its domain, i.e. the subset of S_1 for which it is defined.

Definition 11 (Valuation). *A* variable assignment *is a (partial) function* θ_v *from o-metavariables to variables with finite domain, such that for every pair of o-metavariables* $\alpha, \widehat{\beta}$ *we have* $\theta_v\alpha \not\equiv \theta_v\widehat{\beta}$ *(pre-bound and pre-free o-metavariables are assigned different values).*

A valuation θ *is a pair of (partial) functions* (θ_v, θ_t) *where* θ_v *is a variable assignment and* θ_t *maps t-metavariables to terms. We write* $Dom(\theta)$ *for* $Dom(\theta_v) \cup Dom(\theta_t)$*. A valuation* θ *may be extended in a unique way to the set of pre-metaterms* M *such that* $MVar(M) \subseteq Dom(\theta)$ *as follows:*

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

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

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 $\theta M \equiv_\alpha \theta N$ for any valuation θ such that θM and θN are defined. Indeed, coming back to Example 3 and taking $\theta = \{\alpha/x, \beta/y, X/x\}$, we have $\theta\lambda\alpha.(\alpha) \equiv \lambda x.(x) \equiv_\alpha \lambda y.(y) \equiv \theta\lambda\beta.(\beta)$, $\theta\lambda\alpha.(f) \equiv \lambda x.(f) \equiv_\alpha \lambda y.(f) \equiv \theta\lambda\beta.(f)$, $\theta\lambda\alpha.(X) \equiv \lambda x.(x) \not\equiv_\alpha \lambda y.(x) \equiv \theta\lambda\beta.(X)$, $\theta\lambda\beta.(\lambda\alpha.(X)) \equiv \lambda y.(\lambda x.(x)) \not\equiv_\alpha \lambda x.(\lambda y.(x)) \equiv \theta\lambda\alpha.(\lambda\beta.(X))$.

Definition 12 (Safe valuations). *Let $M \in \mathcal{PMT}$ and θ a valuation with $MVar(M) \subseteq Dom(\theta)$. We say that θ* is safe for *M if $\overline{\theta}M$ is defined, i.e. the substitutions generated by the last clause of Definition 11 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.*

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 $MVar(M) \subseteq Dom(\theta)$ that contains no occurrences of the metasubstitution operator 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 [22] to avoid inconsistencies in rewrite steps.

Definition 13 (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 $x \in FV(\theta X)$, either ($\forall 1 \leq i \leq n$ we have $x \in \theta l_i$) or ($\forall 1 \leq i \leq n$ we have $x \notin \theta l_i$).*

This definition may be read as: one occurrence of $x \in FV(\theta X)$ 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 $\theta\alpha \equiv x$. For example, consider the *SERS* rule $\lambda\alpha.(\xi\beta.(X)) \longrightarrow \xi\beta.(X)$ 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).

Definition 14 (Admissible valuations). *A valuation θ is said to be* admissible for a rewrite rule *(G, D) iff*

- *θ is safe for (G, D)*
- *if α and β occur in (G, D) with $\alpha \not\equiv \beta$ then $\theta_v\alpha \not\equiv \theta_v\beta$*
- *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: consider the rule $\lambda\alpha.app(X, \alpha) \longrightarrow X$, the valuation $\theta = \{\alpha/x, X/x\}$ is trivially safe but 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).

Before introducing the following definition let us discuss about the two possible and equivalent ways to define reduction in a higher-order framework. One can either define reduction via a notion of substitution which makes explicit use of α-conversion, as it is usually done [11], or, as it is done here, reduction is explicitly defined as reduction modulo α-conversion and using a notion of restricted substitution which does not make use of α-conversion. We choose this second (and more involved) approach since we prefer to have a notion of reduction on terms in both formalisms (with names and de Bruijn indices), which is similar enough to make technical proofs work easily.

Definition 15 (Reduction on terms). *Let $\mathcal{R}$ be a set of SERS rewrite rules and s, t terms. We say that s* $\mathcal{R}$-reduces to *t, written $s \longrightarrow_{\mathcal{R}} t$, iff there exists a rewrite rule $(G, D) \in \mathcal{R}$, an admissible valuation θ for (G, D) and a context C such that $s \equiv_\alpha C[\theta G]$ and $t \equiv_\alpha C[\theta D]$.*

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

Lemma 1. *Let θ be an admissible valuation for a rewrite rule (G, D). Then $FV(\theta D) \subseteq FV(\theta G)$.*

Proof. Suppose $x \in FV(\theta D)$. Then

- if x comes from a free o-metavariable $\widehat{\beta}$ occurring in D with $\theta\widehat{\beta} \equiv x$, then since $FMVar(D) \subseteq FMVar(G)$ we also have that $\widehat{\beta}$ occurs in G. Moreover, by definition of valuation, variables assigned to free o-metavariables cannot be captured, so that we necessarily have $x \in FV(\theta G)$.
- if x comes from instantiating a t-metavariable Z occurring in D, then Z occurs in D at position p_D, $x \in FV(\theta Z)$ and x does not appear in $\theta(l_D)$, where l_D is the parameter path of p_D in D. Now, since $FMVar(D) \subseteq FMVar(G)$, we also have that Z occurs in G, let us say at position l_G. Suppose that $x \notin FV(\theta G)$. The only possible case is that x was captured in θG so that $x \in BV(\theta G)$. Therefore x appears in $\theta(l_G)$ which contradicts the fact that θ verifies the path condition for Z in (G, D).

The reduction relation on terms satisfies the following property:

Corollary 1. *Let $s, t \in \mathcal{T}$, if $s \longrightarrow_{\mathcal{R}} t$, then $FV(t) \subseteq FV(s)$.*

Proof. If $s \longrightarrow_{\mathcal{R}} t$, then there exists a rule $(G, D) \in \mathcal{R}$ and a context C such that $s \equiv_\alpha C[\theta G]$ and $t \equiv_\alpha C[\theta D]$ where θ is an admissible valuation for (G, D). Since $FV(C[\theta G]) = FV(s)$ and $FV(C[\theta D]) = FV(t)$ we can reason directly on $C[\theta G]$ and $C[\theta D]$. Suppose $x \in FV(C[\theta D])$. Then

- if $x \in FV(C)$ we trivially have $x \in FV(C[\theta G])$.
- if $x \in FV(\theta D)$ then by Lemma 1 $x \in FV(\theta G)$ and since x is not captured in C, we also have $x \in FV(C[\theta G])$.

Lemma 2. *Let $s, t \in \mathcal{T}$, if $s \longrightarrow_{\mathcal{R}} t$, then $FV(t) \subseteq FV(s)$.*
Proof. It follows from the fact that if θ is admissible for (G, D) then $FV(\theta D) \subseteq FV(\theta G)$. See [4] for details.

3 Simplified Expression Reduction Systems with Indices

We introduce de Bruijn indices based higher-order rewrite formalism $SERS_{DB}$.

3.1 De Bruijn metaterms and terms

A classical way to avoid α-conversion is to use de Bruijn index notation [7, 8], where names of variables are replaced by natural numbers. Indeed, a given occurrence p of a variable in a term, say x, is replaced by the number of binder operators whose occurrences are between the binder of this x and p. For example, $\xi x.(\lambda y.(f(x, \xi z.(g(z, x)), y)))$ is written $\xi(\lambda(f(2, \xi(g(1, 3)), 1)))$. Remark that a same variable can be translated in many different ways (as the variable x is translated into 2 and 3), and that different variables can be translated into the same number (as the variables z and y are both translated into 1). Representation of free variables is retrieved from a given context of variables in such a way that an occurrence of a variable n represents the $(n-m)$-th free variable of the context when $n > m$ and there are m binder operators surrounding this occurrence n. For example, the term $\xi x.(f(y, z))$ is written as $\xi(f(2, 3))$ with respect to the context y, z. When talking about a set N of de Bruijn indices we may refer to $\mathtt{Names}(N)$ as the set of names of N given by the order on the set of variables $\mathcal{V}$ introduced in Section 2. Indeed, if $N = \{n_1, \ldots, n_m\}$, then $\mathtt{Names}(N) = \{x_{n_1}, \ldots, x_{n_m}\}$.

In the sequel, 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", eg. "de Bruijn metaterms".

Definition 16 (de Bruijn signature). *Consider the denumerable and disjoint infinite sets:*

- $\{\alpha_1, \alpha_2, \alpha_3, \ldots\}$ *a set of symbols called* binder indicators*, denoted* $\alpha, \beta, \ldots$*,*
- $\mathcal{I}_{\mathcal{MV}} = \{\widehat{\alpha_1}, \widehat{\alpha_2}, \ldots\}$ *a set of* i-metavariables *(i for index), denoted* $\widehat{\alpha}, \widehat{\beta}, \ldots$*,*
- $\mathcal{T}_{\mathcal{MV}} = \{X_l^1, X_l^2, X_l^3, \ldots\}$ *a set of* t-metavariables *(t for term), where* l *ranges over the set of labels built over binder indicators, denoted* $X_l, Y_l, Z_l, \ldots$*,*
- $\mathcal{F} = \{f_1, f_2, f_3, \ldots\}$ *a set of* function symbols *equipped with a fixed (possibly zero) arity, denoted* $f, g, h, \ldots$*,*
- $\mathcal{B} = \{\lambda_1, \lambda_2, \lambda_3, \ldots\}$ *a set 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 1. The reason for using the same alphabet in both formalisms shall 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 an $SERS_{DB}$ system.

Definition 17 (de Bruijn pre-metaterms). *The set of* de Bruijn pre-metaterms*, denoted* $\mathcal{PMT}_{db}$*, is defined by the following two-sorted grammar:*

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

The symbol $.[\![.]\!]$ *in a pre-metaterm* $A[\![A]\!]$ *is called* 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* operates *as a binder in that it has binding power) but* not *a binder symbol since it is not an element of* $\mathcal{B}$*.*

We shall 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.

Even if 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 intuitively some 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 the formalism $SERS_{DB}$, 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 [18]). However, free variables in $SERS_{DB}$ appear always in a binding context, so that a de Bruijn valuation of such kind of variables has to reflect the adjustment needed to represent the same variables but in a different context. This can be done by surrounding 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 (c.f. condition on variable assignments in Definition 11). Our solution is then to translate the pre-metaterm $\xi\alpha.(\widehat{\beta})$ by $\xi(\mathtt{S}(\widehat{\beta}))$ in such a way that there is no capture of variables since $\kappa(\xi(\mathtt{S}(\widehat{\beta})))$ is exactly $\xi(2)$. The solution adopted here for translating pre-free o-metavariables into the de Bruijn formalism is in some sense what is called *pre-cooking* in [10]: the pre-cooking function takes a $\lambda\sigma$ term with t-metavariables and suffixes them with as many explicit shift operators 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.

We use $MVar(A)$ (resp. $MVar_i(A)$ and $MVar_t(A)$) to denote the set of all metavariables (resp. i- and t-metavariables) of the de Bruijn pre-metaterm A.

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 $\mathtt{S}^i(1)$ (resp. $\mathtt{S}^i(\widehat{\alpha})$) 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 18 (de Bruijn metaterms). *A pre-metaterm $A \in \mathcal{PMT}_{db}$ is said to be a* metaterm *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 as follows:*

- $\mathcal{WF}_l(\mathtt{S}^j(1))$ *iff* $j+1 \leq |l|$
- $\mathcal{WF}_l(\mathtt{S}^j(\widehat{\alpha}))$ *iff* $j = |l|$
- $\mathcal{WF}_l(X_k)$ *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. Note that when considering $\mathcal{WF}_l(M)$ and $\mathcal{WF}_l(A)$ it is Definitions 4 and 18 which are referenced, respectively.

Example 6. Pre-metaterms $\xi(X_\alpha, \lambda(Y_{\beta\alpha}, \mathtt{S}(1)))$, $f(\widehat{\beta}, \lambda(Y_\alpha, \mathtt{S}(\widehat{\alpha})))$, $g(\lambda(\xi(h)))$ are metaterms, while $f(\mathtt{S}(\widehat{\alpha}), \xi(X_\beta))$, $\lambda(\xi(X_{\alpha\alpha}))$, $f(\widehat{\beta}, \lambda(\xi(\mathtt{S}(\widehat{\beta}))))$ are not.

Definition 19 (de Bruijn terms and de Bruijn contexts). *The set of* de Bruijn terms*, denoted $\mathcal{T}_{db}$, and the set of de Bruijn contexts are defined by:*

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

We use $a, b, a_i, b_i, \ldots$ for de Bruijn terms and $E, F, \ldots$ for de Bruijn contexts. 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 6, we have here that de Bruijn terms are also de Bruijn pre-metaterms, that is, $\mathcal{T}_{db} \subset \mathcal{PMT}_{db}$, 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 metaterm $\xi(\xi(4))$. If an arbitrary free variable is wished to be represented in a metaterm, then i-metavariables should be used.

Definition 20 (Free de Bruijn variables). *We denote by $FV(a)$ the set of* free variables *of a de Bruijn term a, which is defined as follows:*

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

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

Definition 21 (de Bruijn substitution). *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:*

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

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

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

Due to the various notions of substitution and replacement introduced so far, in Figure 2 we give a brief synopsis of the situation.

Operator	Names	de Bruijn	Terms	Metaterms	Explicit	Implicit
$M_1[\alpha \leftarrow M_2]$	√			√	√	
$t_1\{x \leftarrow t_2\}$	√		√			√
$A_1[\![A_2]\!]$		√		√	√	
$a_1\{\!\{n \leftarrow a_2\}\!\}$		√	√			√
$M \ll\alpha \leftarrow \beta\gg$	√			√		√

Fig. 2. Notions of substitution

Definition 22 (Free de Bruijn metavariables). *Let A be a de Bruijn pre-metaterm. The set of* free metavariables *of A, $FMVar(A)$, is defined as:*

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

Note that this definition also applies to de Bruijn metaterms. The set of names of free metavariables of A is the set of free metavariables of A where each X_l is replaced simply by X. This notion will be used in Definition 23.

3.2 Reduction

We define rewrite rules, valuations, their validity, and reduction in Simplified Expression Reduction Systems with de Bruijn indices.

Definition 23 (de Bruijn rewrite rule). *A* de Bruijn rewrite rule *is a pair of de Bruijn metaterms (L, R) (also written $L \longrightarrow R$) such that*

- *the first symbol in L is a function symbol or a binder symbol*
- *the set of names of $FMVar(R)$ is included in the set of names of $FMVar(L)$*
- *the metasubstitution operator does not occur in L*

Definition 24 (de Bruijn valuation). *A* de Bruijn valuation *κ is a pair of (partial) functions (κ_i, κ_t) where κ_i is a function from i-metavariables to integers, and κ_t is a function from t-metavariables to de Bruijn terms. We denote by $Dom(\kappa)$ the set $Dom(\kappa_i) \cup Dom(\kappa_t)$. A valuation κ determines in a unique way a function $\overline{\kappa}$ from the set of pre-metaterms A with $FMVar(A) \subseteq Dom(\kappa)$ to the set of terms as follows:*

$$\begin{array}{llll}
\overline{\kappa}1 \stackrel{\text{def}}{=} 1 & & \overline{\kappa}f(A_1, \ldots, A_n) \stackrel{\text{def}}{=} f(\overline{\kappa}A_1, \ldots, \overline{\kappa}A_n) \\
\overline{\kappa}\mathtt{S}(I) \stackrel{\text{def}}{=} \mathtt{S}(\overline{\kappa}I) & & \overline{\kappa}\xi(A_1, \ldots, A_n) \stackrel{\text{def}}{=} \xi(\overline{\kappa}A_1, \ldots, \overline{\kappa}A_n) \\
\overline{\kappa}\widehat{\alpha} \stackrel{\text{def}}{=} \kappa_i\widehat{\alpha} & & \overline{\kappa}(A_1[\![A_2]\!]) \stackrel{\text{def}}{=} \overline{\kappa}(A_1)\{\!\{1 \leftarrow \overline{\kappa}A_2\}\!\} \\
\overline{\kappa}X_l \stackrel{\text{def}}{=} \kappa_t X_l & &
\end{array}$$

Note that in the above definition the substitution operator $.\{\!\{. \leftarrow .\}\!\}$ refers to the usual substitution defined on terms with de Bruijn indices (cf. Definition 21).

We now introduce the notion of *value function* which is used to give semantics to metavariables with labels in the $SERS_{DB}$ formalism. 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 [10] in order to avoid variable capture in the higher-order unification procedure. In *XRS* [17] 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 it is required (roughly) 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* (cf. Definition 26) and it turns out to be more general than the solutions proposed in [10] and [17].

Definition 25 (Value function). *Let $a \in \mathcal{T}_{db}$ and l be a label of binder indicators. Then we define the value function $Value(l, a)$ as $Value^0(l, a)$ where*

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

It is worth noting that $\mathit{Value}^i(l,n)$ may give three different kinds of results. This is just a technical trick to make easier later proofs. Indeed, we have for example $\mathit{Value}(\alpha\beta, \xi(f(3,1))) \equiv \xi(f(\beta,1)) \equiv \mathit{Value}(\beta\alpha, \xi(f(2,1)))$ and $\mathit{Value}(\epsilon, f(\xi(1),\lambda(2))) \equiv f(\xi(1),\lambda(x_1)) \not\equiv f(\xi(1),\lambda(\alpha)) \equiv \mathit{Value}(\alpha, f(\xi(1),\lambda(2)))$. Thus 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 25 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.

In order to introduce the notion of valid de Bruijn valuations let us consider the following rule:

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

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 (called r_{DB}):

$$\xi(\xi(X_{\beta\alpha})) \longrightarrow_{r_{DB}} \xi(\xi(X_{\alpha\beta}))$$

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 a r-reduction step $\xi x.(\xi y.(x)) \longrightarrow \xi y.(\xi x.(x))$. 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 a r_{DB}-reduction step $\xi(\xi(2)) \longrightarrow \xi(\xi(1))$. 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. This is exactly the role of the property of validity:

Definition 26 (Valid de Bruijn valuation). *A de Bruijn valuation κ is said to be* valid *if for every pair of t-metavariables X_l and $X_{l'}$ in $Dom(\kappa)$ we have $\mathit{Value}(l,\kappa X_l) \equiv \mathit{Value}(l',\kappa X_{l'})$. Likewise, we say that a de Bruijn valuation κ is* valid for a rewrite rule (L,R) *if for every pair of t-metavariables X_l and $X_{l'}$ in (L,R) we have $\mathit{Value}(l,\kappa X_l) \equiv \mathit{Value}(l',\kappa X_{l'})$.*

It is interesting to note that there is no concept analogous to safeness (cf. Definition 12) as used for named *SERS* due to the use of de Bruijn indices. Also, the last condition in the definition of an admissible valuation (cf. Definition 14) is subsumed by the above Definition 26 in the setting of SERS_{DB}.

Example 7. Returning to the example just after Definition 25 we have that $\kappa = \{X_{\beta\alpha}/2, X_{\alpha\beta}/1\}$ is valid since $\mathit{Value}(\beta\alpha, 2) \equiv \alpha \equiv \mathit{Value}(\alpha\beta, 1)$.

Another interesting example is the well-known η-contraction rule $\lambda x.@(X,x) \longrightarrow X$ if $x \notin FV(X)$. It can be expressed in the *SERS* formalism as the rule $(\eta_n)\quad \lambda\alpha.@(X,\alpha) \longrightarrow X$, and in the *SERS$_{DB}$* formalism as the rule $(\eta_{db})\quad \lambda(@(X_\alpha,1)) \longrightarrow X_\epsilon$.

Remark that this kind of rule cannot be expressed in the *XRS* formalism [17] 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_n 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 $\mathit{Value}(\alpha,\kappa X_\alpha) \equiv \mathit{Value}(\epsilon,\kappa X_\epsilon)$, that is, $m \equiv 1$ 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 label contexts of all the occurrences of X in the rewrite rule. This is, in some sense, an updating of X w.r.t the different label contexts where it appears, and it gives us the right notion of coherence for valuations.

Definition 27 (Reduction on de Bruijn terms). *Let $\mathcal{R}$ be a set of de Bruijn rules and a,b de Bruijn terms. We say that a $\mathcal{R}$-reduces to b, written $a \longrightarrow_{\mathcal{R}} b$, iff there is a de Bruijn rule $(L,R) \in \mathcal{R}$ and a de Bruijn valuation κ valid for (L,R) such that $a \equiv E[\kappa L]$ and $b \equiv E[\kappa R]$, 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 $\kappa = \{X_\alpha/\lambda(app(1,3)), X_\epsilon/\lambda(app(1,2))\}$.

Lemma 3. *Let κ be a valid valuation for a rewrite rule (L,R). Then $FV(\kappa R) \subseteq FV(\kappa L)$.*

Proof. Suppose $n \in FV(\kappa R)$. Then

- if n comes from a free o-metavariable $\widehat{\beta}$ occurring in R, then there is an m such that $\mathtt{S}^m(\widehat{\beta})$ occurs in R and $\kappa\widehat{\beta} \equiv n$, where m is the number of binders "above" $\widehat{\beta}$ in R. Since L and R are de Bruijn metaterms and the names of $\mathit{FMVar}(R)$ are included in the names of $\mathit{FMVar}(L)$ there is an m' such that $\mathtt{S}^{m'}(\widehat{\beta})$ occurs in L where m' is the number of binders "above" this occurrence of $\widehat{\beta}$. Hence by definition of free variable $n \in FV(\kappa L)$.
- if n comes from instantiating a t-metavariable Z_l occurring in R then $n+|l| \in FV(\kappa Z_l)$. Since L and R are de Bruijn metaterms and the names of $\mathit{FMVar}(R)$ are included in the names of $\mathit{FMVar}(L)$ then there is an occurrence of $Z_{l'}$ in L for some label of binder indicators l'. Since κ is valid for (L,R) then $\mathit{Value}(l,\kappa Z_l) \equiv \mathit{Value}(l',\kappa Z_{l'})$. Now as $n+|l| \in FV(\kappa Z_l)$ we have that x_n occurs in $\mathit{Value}(l,\kappa Z_l)$ and hence also in $\mathit{Value}(l',\kappa Z_{l'})$. Therefore $n+|l'| \in FV(\kappa Z_{l'})$. Finally, by well-formedness of the pre-metaterm L we have $n \in FV(\kappa L)$.

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

Corollary 2. *Let $a \in \mathcal{T}_{db}$. If $a \longrightarrow_{\mathcal{R}} b$, then $FV(b) \subseteq FV(a)$.*

Proof. If $a \longrightarrow_{\mathcal{R}} b$, then there exists a valid valuation κ for a rewrite rule (L,R) in $\mathcal{R}$ such that $a \equiv E[\kappa L]$ and $b \equiv E[\kappa R]$. Then

- if $n \in FV(E)$ we trivially have $n \in FV(a)$.
- if $n \in FV(\kappa R)$ then by Lemma 3 also $n \in FV(\kappa L)$ and since n is not captured in E, we also have $n \in FV(a)$.

4 From *SERS* to $SERS_{DB}$

In this section we show how reduction 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 *preservation-of-reduction 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 preservation-of-reduction phase, that is, showing how *SERS* rewrite steps can be simulated via $SERS_{DB}$ rewrite steps. The preservation-of-reduction phase shall use the results developed in the definition phase.

4.1 From *SERS* to $SERS_{DB}$: the definition phase

In this section we study the definition phase, that is, we define translations from the *SERS* to the $SERS_{DB}$ formalism for all the objects used in the reduction process.

Definition 28 (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*

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

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

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 in the main statements of this section.

Lemma 4. *Let $s \in \mathcal{T}$, let l,k be labels of variables and x,y variables such that y does not occur in s and $x,y \notin l$. If $s\{x \leftarrow t\}$ is defined then $T_{lyk}(s\{x \leftarrow y\}) \equiv T_{lxk}(s)$.*

Proof. By induction on s.

- $s \equiv x$. Then we reason as follows:

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

- $s \equiv z \not\equiv x$. Then $T_{lyk}(z\{x \leftarrow y\}) \equiv T_{lyk}(z)$. We consider two further cases:
  - $z \in lyk$. Recall that y does not occur in z and hence $y \not\equiv z$. Then $T_{lyk}(z) \equiv \mathtt{pos}(z,lyk) \equiv \mathtt{pos}(z,lxk) \equiv T_{lxk}(z)$.
  - $z \notin lyk$. Then $T_{lyk}(z) \equiv \mathcal{O}(z)+|lyk| \equiv \mathcal{O}(z)+|lxk| \equiv T_{lxk}(z)$.
- $s \equiv f(s_1,\ldots,s_n)$. We use the induction hypothesis.

– $s \equiv \xi x.(s_1, \ldots, s_n)$. We reason as follows:

$$\begin{aligned} T_{lyk}(s\{x \leftarrow y\}) &\equiv T_{lyk}(\xi x.(s_1, \ldots, s_n)) \\ &\equiv T_{lxk}(\xi x.(s_1, \ldots, s_n)) \end{aligned}$$

The last equality follows from the observation that since $x \notin FV(\xi x.(s_1, \ldots, s_n))$ then $T_{lxk}(\xi x.(s_1, \ldots, s_n)) \equiv T_{lzk}(\xi x.(s_1, \ldots, s_n))$ for any $z \notin FV(\xi x.(s_1, \ldots, s_n))$, in particular, this holds for $z \equiv y$.

– $s \equiv \xi z.(s_1, \ldots, s_n)$ with $z \not\equiv x$. Note that $z \not\equiv y$ by hypothesis. We reason as follows:

$$\begin{aligned} T_{lyk}(s\{x \leftarrow y\}) &\equiv \xi(T_{zlyk}(s_1\{x \leftarrow y\}), \ldots, T_{zlyk}(s_n\{x \leftarrow y\})) \\ &\equiv \xi(T_{zlxk}(s_1), \ldots, T_{zlxk}(s_n)) && \text{(i.h.)} \\ &\equiv T_{lxk}(s) \end{aligned}$$

As expected, the translation satisfies:

Lemma 5 (T is compatible with α-conversion). *Given two terms $s, t \in \mathcal{T}$ such that $s \equiv_\alpha t$ we have $T_k(s) \equiv T_k(t)$ for any label of variables k.*

Proof. By induction on the derivation of $s \equiv_\alpha t$.

– Base cases:
  - $s \equiv t$. Trivial.
  - $s \equiv_\alpha t$ and the conversion takes place at the root. Then $s \equiv \xi x.(s_1, \ldots, s_n) \equiv_\alpha \xi y.(s_1\{x \leftarrow y\}, \ldots, s_n\{x \leftarrow y\}) \equiv t$ where y is a variable not occurring in $s_1, \ldots s_n$. Then by Lemma 4 we have $T_k(s) \equiv \xi(T_{xk}(s_1), \ldots, T_{xk}(s_n)) \equiv \xi(T_{yk}(s_1\{x \leftarrow y\}), \ldots, T_{yk}(s_n\{x \leftarrow y\})) \equiv T_k(t)$.

– Inductive cases:
  - $s \equiv_\alpha t$ follows from $t \equiv_\alpha s$. We use the induction hypothesis.
  - $s \equiv_\alpha t$ follows from $s \equiv_\alpha s'$ and $s' \equiv_\alpha t$. We use the induction hypothesis.
  - the conversion is internal. Then two further cases are considered:
    * $s \equiv f(s_1, \ldots, s_i, \ldots, s_n)$ and $t \equiv f(s_1, \ldots, s_i', \ldots, s_n)$ where $s_i \equiv_\alpha s_i'$. We conclude by using the induction hypothesis.
    * $s \equiv \xi x.(s_1, \ldots, s_i, \ldots, s_n)$ and $t \equiv \xi x.(s_1, \ldots, s_i', \ldots, s_n)$ where $s_i \equiv_\alpha s_i'$. Then we have $T_k(s) \equiv \xi(T_{xk}(s_1), \ldots, T_{xk}(s_i), \ldots, T_{xk}(s_n)) \equiv_{i.h.} \xi(T_{xk}(s_1), \ldots, T_{xk}(s_i'), \ldots, T_{xk}(s_n)) \equiv T_k(t)$.

Definition 29 (From pre-metaterms to de Bruijn pre-metaterms). *The translation of a pre-metaterm M, denoted $T(M)$, is defined as $T_\epsilon(M)$ where $T_k(M)$ is defined by*

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

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 $T(.)$ is not defined. Moreover, if M is a metaterm then $T(M)$ is a de Bruijn metaterm.

Lemma 6 (T preserves well-formedness). *If M is a metaterm, then $T(M)$ is a de Bruijn metaterm.*

Proof. We need to prove a more general result: let M be a pre-metaterm, if $\mathcal{WF}_l(M)$, then $\mathcal{WF}_l(T_l(M))$. This is proved by induction on M.

- $M \equiv \alpha$. Then $\alpha \in l$ and we have $\mathcal{WF}_l(T_l(\alpha))$ iff $\mathcal{WF}_l(\mathtt{pos}(\alpha, l))$. And the latter holds trivially.
- $M \equiv \widehat{\alpha}$. Then $\mathcal{WF}_l(T_l(\widehat{\alpha}))$ iff $\mathcal{WF}_l(\mathtt{S}^{|l|}(\widehat{\alpha}))$. And the latter holds trivially.
- $M \equiv X$. Then we have $\mathcal{WF}_l(T_l(X))$ iff $\mathcal{WF}_l(X_l)$. And the latter holds trivially.
- $M \equiv f(M_1, \ldots, M_n)$. By induction hypothesis.
- $M \equiv \xi\alpha.(M_1, \ldots, M_n)$. Then $\mathcal{WF}_l(T_l(\xi\alpha.(M_1, \ldots, M_n)))$ iff $\mathcal{WF}_{\beta l}(T_{\alpha l}(M_i))$ with $1 \leq i \leq n$ for some $\beta \notin l$. Since $\mathcal{WF}_l(M)$ we have $\alpha \notin l$ and $\mathcal{WF}_{\alpha l}(M_i)$, so that we may take $\beta \equiv \alpha$ and conclude by the induction hypothesis.
- $M \equiv M_1[\alpha \leftarrow M_2]$. Similar to the previous case.

Example 8. Let $M \equiv \xi\alpha.(X, \lambda\beta.(Y, \alpha))$, $M' \equiv f(\widehat{\beta}, \lambda\alpha.(Y, \widehat{\alpha}))$ and $M'' \equiv g(\lambda\alpha.(\xi\beta.(h)))$. Then their respective translations are $A \equiv \xi(X_\alpha, \lambda(Y_{\beta\alpha}, \mathtt{S}(1)))$, $A' \equiv f(\widehat{\beta}, \lambda(Y_\alpha, \mathtt{S}(\widehat{\alpha})))$ and $A'' \equiv g(\lambda(\xi(h)))$, which are metaterms as remarked in Example 6.

Definition 30 (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 Lemma 6 and Definition 30, if (G, D) is an *SERS* rewrite rule, then $T(G, D)$ is an *SERS$_{DB}$* rewrite rule.

Example 9 ($\lambda\mathtt{x}$ continued). Following Example 4, the specification of $\lambda\mathtt{x}$ in the *SERS$_{DB}$* formalism is given below.

$$\begin{array}{lll}
@(\lambda(X_\alpha), Z_\epsilon) & \longrightarrow & \Sigma(\sigma(X_\alpha), Z_\epsilon) \\
\Sigma(\sigma(@(X_\alpha, Y_\alpha)), Z_\epsilon) & \longrightarrow & @(\Sigma(\sigma(X_\alpha), Z_\epsilon), \Sigma(\sigma(Y_\alpha), Z_\epsilon)) \\
\Sigma(\sigma(\lambda(X_{\beta\alpha})), Z_\epsilon) & \longrightarrow & \lambda(\Sigma(\sigma(X_{\alpha\beta}), Z_\beta)) \\
\Sigma(\sigma(1), Z_\epsilon) & \longrightarrow & Z_\epsilon \\
\Sigma(\sigma(\mathtt{S}(\widehat{\beta})), Z_\epsilon) & \longrightarrow & \widehat{\beta}
\end{array}$$

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

Example 10 (The $\lambda\Delta$-calculus continued). The translation of the $\lambda\Delta$-calculus (Example 5) yields the following rewrite rules in the *SERS$_{DB}$* formalism

$$\begin{array}{lll}
@(\lambda(X_\alpha), Z_\epsilon) & \longrightarrow & X_\alpha[\![Z_\epsilon]\!] \\
@(\Delta(X_\alpha), Z_\epsilon) & \longrightarrow & \Delta(X_{\alpha\beta}[\![\lambda(@(\mathtt{S}(1), @(1, Z_{\gamma\beta})))]\!]) \\
\Delta(@(1, X_\alpha)) & \longrightarrow & X_\epsilon \\
\Delta(@(1, (\Delta(@(\mathtt{S}(1), X_{\beta\alpha}))))) & \longrightarrow & X_\epsilon
\end{array}$$

We remark that the translation of Δ_1, Δ_2 and Δ_3 would not be possible in the formalism proposed in [17].

Definition 31 (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*

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

4.2 From *SERS* to $SERS_{DB}$: the preservation-of-reduction phase

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

The following lemmas will be used in the proof of the main result of this section.

Lemma 7. *Let $s \in \mathcal{T}$ and $a \in \mathcal{T}_{db}$, let x be a variable and l, k be labels of variables with $|l| = i - 1$ and $x \in l$. Then $T_{lk}(s) \equiv T_{lxk}(s)\{\!\{i \leftarrow a\}\!\}$.*

Proof. By induction on s.

- $s \equiv x$. Then $T_{lk}(x) \equiv \mathtt{pos}(x, lk) \equiv \mathtt{pos}(x, lxk) \equiv \mathtt{pos}(x, lxk)\{\!\{i \leftarrow a\}\!\}$. The last equality holds since $\mathtt{pos}(x, lxk) < i$.
- $s \equiv y \not\equiv x$. We have three cases to consider:
  - $y \in l$. Then $T_{lk}(y) \equiv \mathtt{pos}(y, lk) \equiv \mathtt{pos}(y, lxk) \equiv \mathtt{pos}(y, lxk)\{\!\{i \leftarrow a\}\!\}$. The last equality holds since $\mathtt{pos}(y, lxk) < i$.
  - $y \notin l$ and $y \in k$. Then $T_{lk}(y) \equiv \mathtt{pos}(y, lk) \equiv \mathtt{pos}(y, lxk) - 1 \equiv \mathtt{pos}(y, lxk)\{\!\{i \leftarrow a\}\!\}$. The last equality holds since $\mathtt{pos}(y, lxk) > i$.
  - $y \notin lk$. Then $T_{lk}(y) \equiv \mathcal{O}(y) + |lk| \equiv \mathcal{O}(y) + |lxk| - 1 \equiv T_{lxk}(y)\{\!\{i \leftarrow a\}\!\}$.
- $s \equiv f(s_1, \ldots, s_n)$. Then we use the induction hypothesis.
- $s \equiv \xi y.(s_1, \ldots, s_n)$. Then we have:

$$\begin{aligned} T_{lk}(\xi y.(s_1, \ldots, s_n)) \equiv & \quad \xi(T_{ylk}(s_1), \ldots, T_{ylk}(s_n)) \\ \equiv_{i.h.} & \; \xi(T_{ylxk}(s_1)\{\!\{i+1 \leftarrow a\}\!\}, \ldots, T_{ylxk}(s_n)\{\!\{i+1 \leftarrow a\}\!\}) \\ \equiv & \quad T_{lxk}(s)\{\!\{i \leftarrow a\}\!\} \end{aligned}$$

Lemma 8. *Let $s \in \mathcal{T}$, 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 $T_{l_1 l_2 k}(s) \equiv \mathcal{U}^i_j(T_{l_1 k}(s))$.*

Proof. By induction on s.

- $s \equiv x$. Note that since by hypothesis $x \notin l_2$ we have three cases to consider:
  - $x \in l_1$. Then $T_{l_1 l_2 k}(x) \equiv \mathtt{pos}(x, l_1 l_2 k) \equiv \mathtt{pos}(x, l_1 k) \equiv \mathcal{U}^i_j(\mathtt{pos}(x, l_1 k))$. The last equality holds since $\mathtt{pos}(x, l_1 k) \leq j$.
  - $x \notin l_1$ and $x \in k$. Then $T_{l_1 l_2 k}(x) \equiv \mathtt{pos}(x, l_1 l_2 k) \equiv \mathtt{pos}(x, l_1 k) + i - 1 \equiv \mathcal{U}^i_j(\mathtt{pos}(x, l_1 k))$.
  - $x \notin l_1 k$. Then $T_{l_1 l_2 k}(x) \equiv \mathcal{O}(x) + |l_1 l_2 k| \equiv \mathcal{O}(x) + |l_1 k| + i - 1 \equiv \mathcal{U}^i_j(T_{l_1 k}(x))$.
- $s \equiv f(s_1, \ldots, s_n)$. The we use the induction hypothesis.
- $s \equiv \xi y.(s_1, \ldots, s_n)$. Then we have:

$$\begin{aligned} T_{l_1 l_2 k}(s) \equiv & \quad \xi(T_{y l_1 l_2 k}(s_1) \ldots, T_{y l_1 l_2 k}(s_1)) \\ \equiv_{i.h} & \; \xi(\mathcal{U}^i_{j+1}(T_{y l_1 k}(s_1)), \ldots, \mathcal{U}^i_{j+1}(T_{y l_1 k}(s_1))) \\ \equiv & \quad \mathcal{U}^i_j(T_{l_1 k}(s)) \end{aligned}$$

Lemma 9. *Let $s, t \in \mathcal{T}$, 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 $T_{lk}(s\{x \leftarrow t\}) \equiv T_{lxk}(s)\{\!\{i \leftarrow T_k(t)\}\!\}$.*

Proof. By induction on s.

- $s \equiv x$. Then we have:
$$\begin{aligned} T_{lk}(t) &\equiv \mathcal{U}_0^i(T_k(t)) && \text{(L.8)} \\ &\equiv i\{\!\{i \leftarrow T_k(t)\}\!\} \\ &\equiv T_{lxk}(s)\{\!\{i \leftarrow T_k(t)\}\!\} \ (x \notin l) \end{aligned}$$
- $s \equiv y \not\equiv x$. Then we have three subcases to consider:
  - $y \in l$. Then $T_{lk}(y) \equiv \mathtt{pos}(y, lk) \equiv \mathtt{pos}(y, lxk) \equiv T_{lxk}(y) \equiv T_{lxk}(y)\{\!\{i \leftarrow T_k(t)\}\!\}$. The last equality holds because $\mathtt{pos}(y, l) < i$.
  - $y \notin l$ and $y \in k$. Then $T_{lk}(y) \equiv \mathtt{pos}(y, lk) \equiv \mathtt{pos}(y, lxk) - 1 \equiv T_{lxk}(y) - 1 \equiv T_{lxk}(y)\{\!\{i \leftarrow T_k(t)\}\!\}$. The last equality holds because $\mathtt{pos}(y, lxk) > i$.
  - $y \notin lk$. Then $T_{lk}(y) \equiv \mathcal{O}(y) + |lk| \equiv \mathcal{O}(y) + |lxk| - 1 \equiv T_{lxk}(y) - 1 \equiv T_{lxk}(y)\{\!\{i \leftarrow T_k(t)\}\!\}$. The last equality holds because $\mathcal{O}(y) + |lxk| > i$.
- $s \equiv f(s_1, \ldots, s_n)$. The we have
$$\begin{aligned} & T_{lk}(s\{x \leftarrow t\}) \\ \equiv \ & f(T_{lk}(s_1\{x \leftarrow t\}), \ldots, T_{lk}(s_n\{x \leftarrow t\})) \\ \equiv_{i.h.} \ & f(T_{lxk}(s_1)\{\!\{i \leftarrow T_k(t)\}\!\}, \ldots, T_{lxk}(s_n)\{\!\{i \leftarrow T_k(t)\}\!\}) \\ \equiv \ & T_{lxk}(s)\{\!\{i \leftarrow T_k(t)\}\!\} \end{aligned}$$
- $s \equiv \xi y.(s_1, \ldots, s_n)$. Note that since $s\{x \leftarrow t\}$ is defined by hypothesis we know that $y \notin FV(t)$ for otherwise $s\{x \leftarrow t\}$ would not be defined. We consider two further cases:
  - $y \not\equiv x$. Then we have
$$\begin{aligned} & T_{lk}(s\{x \leftarrow t\}) \\ \equiv \ & \xi(T_{ylk}(s_1\{x \leftarrow t\}, \ldots, s_n\{x \leftarrow t\})) \\ \equiv_{i.h.} \ & \xi(T_{ylxk}(s_1)\{\!\{i+1 \leftarrow T_k(t)\}\!\}, \ldots, T_{ylxk}(s_n)\{\!\{i+1 \leftarrow T_k(t)\}\!\}) \\ \equiv \ & T_{lxk}(s)\{\!\{i \leftarrow T_k(t)\}\!\} \end{aligned}$$
  - $y \equiv x$. Then we have
$$\begin{aligned} & T_{lk}(s\{x \leftarrow t\}) \\ & \equiv T_{lk}(\xi x.(s_1, \ldots, s_n)) \\ & \equiv \xi(T_{xlk}(s_1), \ldots, T_{xlk}(s_n)) \\ & \equiv \xi(T_{xlxk}(s_1)\{\!\{i+1 \leftarrow T_k(t)\}\!\}, \ldots, T_{xlxk}(s_n)\{\!\{i+1 \leftarrow T_k(t)\}\!\}) \ \text{(L.7)} \\ & \equiv T_{lxk}(\xi x.(s_1, \ldots, s_n))\{\!\{i \leftarrow T_k(t)\}\!\} \end{aligned}$$

As expected, the translation is well-behaved w.r.t contexts and valuations.

Lemma 10 (T is modular w.r.t contexts). *Let C be a context, l the label of C and $t \in \mathcal{T}$. Then for every label k we have, $T_k(C[t]) \equiv T_k(C)[T_{lk}(t)]$.*

Proof. By induction on the context C.

- $C \equiv \Box$. Then $l \equiv \epsilon$ and the result holds trivially.
- $C \equiv f(s_1, \ldots, C', \ldots, s_n)$. We use the induction hypothesis and the fact that the label for C is also the label for C'.
- $C \equiv \xi x.(s_1, \ldots, C', \ldots, s_n)$. Then $l \equiv l'x$ for some label l' with l' the label for C'. We reason as follows:
$$\begin{aligned} & T_k(\xi x.(s_1, \ldots, C', \ldots, s_n)[t]) \\ \equiv \ & \xi(T_{xk}(s_1), \ldots, T_{xk}(C'[t]), \ldots T_{xk}(s_n)) \\ \equiv_{i.h.} \ & \xi(T_{xk}(s_1), \ldots, T_{xk}(C')[T_{l'xk}(t)], \ldots T_{xk}(s_n)) \\ \equiv \ & T_k(C)[T_{lk}(t)] \end{aligned}$$

Lemma 11 (T is modular w.r.t valuations). *Let M be a pre-metaterm, l a label of binder indicators, and suppose*

1. $\mathcal{WF}_l(M)$,
2. $\theta = (\theta_v, \theta_t)$ *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 $T_{\theta(l)k}(\theta M) \equiv T_k(\theta)(T_l(M))$.

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

– $M \equiv \alpha \in l$. Then we have

$$\begin{array}{rll} T_{\theta(l)k}(\theta\alpha) \equiv & & \mathtt{pos}(\theta\alpha, \theta(l)) \\ & \equiv_{hyp.2} & \mathtt{pos}(\alpha, l) \\ & \equiv & T_k(\theta)(\mathtt{pos}(\alpha, l)) \\ & \equiv & T_k(\theta)(T_l(\alpha)) \end{array}$$

– $M \equiv \widehat{\alpha}$. Then, by the definition of valuation and since $\widehat{\alpha} \notin l$ because l is a label of binder indicators, we have

$$T_{\theta(l)k}(\theta\widehat{\alpha}) \equiv \begin{cases} \mathtt{pos}(\theta\widehat{\alpha}, k) + |\theta(l)| & \text{if } \theta\widehat{\alpha} \in k \\ \mathcal{O}(\theta\widehat{\alpha}) + |\theta(l)k| & \text{otherwise.} \end{cases}$$

$$\equiv \begin{cases} \mathtt{S}^{|l|}(\mathtt{pos}(\theta\widehat{\alpha}, k)) & \text{if } \theta\widehat{\alpha} \in k \\ \mathtt{S}^{|l|}(\mathcal{O}(\theta\widehat{\alpha}) + |k|) & \text{otherwise.} \end{cases}$$

$$\begin{array}{l} \equiv T_k(\theta)(\mathtt{S}^{|l|}(\widehat{\alpha})) \\ \equiv T_k(\theta)(T_l(\widehat{\alpha})) \end{array}$$

– $M \equiv X$. Then we have,

$$\begin{array}{rl} T_{\theta(l)k}(\theta X) & \equiv T_k(\theta)(X_l) \\ & \equiv T_k(\theta)(T_l(X)) \end{array}$$

– $M \equiv f(M_1, \ldots, M_n)$. Then we have

$$\begin{array}{rll} T_{\theta(l)k}(\theta f(M_1, \ldots, M_n)) \equiv & & f(T_{\theta(l)k}(\theta M_1), \ldots, T_{\theta(l)k}(\theta M_n)) \\ & \equiv_{i.h.} & f(T_k(\theta)(T_l(M_1)), \ldots, T_k(\theta)(T_l(M_n))) \\ & \equiv & T_k(\theta)(T_l(f(M_1, \ldots, M_n))) \end{array}$$

– $M \equiv \xi\alpha.(M_1, \ldots, M_n)$. Then we have

$$\begin{array}{rll} T_{\theta(l)k}(\theta\xi\alpha.(M_1, \ldots, M_n)) \equiv & & \xi(T_{\theta(\alpha)\theta(l)k}(\theta M_1), \ldots, T_{\theta(\alpha)\theta(l)k}(\theta M_n)) \\ & \equiv_{i.h.} & \xi(T_k(\theta)(T_{\alpha l}(M_1)), \ldots, T_k(\theta)(T_{\alpha l}(M_n))) \\ & \equiv & T_k(\theta)(T_l(\xi\alpha.(M_1, \ldots, M_n))) \end{array}$$

– $M \equiv M_1[\alpha \leftarrow M_2]$. Then we have

$$\begin{array}{rll} T_{\theta(l)k}(\theta(M_1[\alpha \leftarrow M_2])) \equiv & & T_{\theta(l)k}(\theta M_1\{\theta_v(\alpha) \leftarrow \theta M_2\}) \\ & \equiv_{L.\ 9} & T_{\theta(\alpha)\theta(l)k}(\theta M_1)\{\!\{1 \leftarrow T_{\theta(l)k}(\theta M_2)\}\!\} \\ & \equiv_{i.h.} & T_k(\theta)(T_{\alpha l}(M_1))\{\!\{1 \leftarrow T_k(\theta)(T_l(M_2))\}\!\} \\ & \equiv & T_k(\theta)(T_{\alpha l}(M_1)[\![T_l(M_2)]\!]) \\ & \equiv & T_k(\theta)(T_l(M_1[\alpha \leftarrow M_2])) \end{array}$$

Note that since θ is safe for M we may apply Lemma 9 above with $l \equiv \epsilon$. Indeed, $\theta_v(\alpha) \notin \theta(l)$ and $FV(\theta M_2) \cap \theta(l) = \emptyset$ for $l \equiv \epsilon$.

Lemma 12. *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 $t \in \mathcal{T}$, 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*

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

Proof. We use induction on t.

- $t \equiv x$. We have the following further cases to consider:
  - $x \equiv x_i$ with $1 \leq i \leq p$. Then
  $$Value^p(k, T_{x_1 \ldots x_p \theta(k) l}(x)) \equiv Value^p(k, i) \equiv i \equiv Value^p(k', i) \equiv Value^p(k', T_{x_1 \ldots x_p \theta(k') l}(x))$$
  - $x \in \theta(k) \cap \theta(k')$ and the previous case does not hold. Let $i = \mathtt{pos}(x, \theta(k))$ and $j = \mathtt{pos}(x, \theta(k'))$. Then $\mathtt{at}(k, i) \equiv \mathtt{at}(k', j)$ by injectivity of θ and we have
  $$\begin{aligned} & Value^p(k, T_{x_1 \ldots x_p \theta(k) l}(x)) \\ \equiv\ & Value^p(k, p + i) \\ \equiv\ & \mathtt{at}(k, i) \\ \equiv\ & \mathtt{at}(k', j) \\ \equiv\ & Value^p(k', p + j) \\ \equiv\ & Value^p(k', T_{x_1 \ldots x_p \theta(k') l}(x)) \end{aligned}$$
  - $x \in l$ and the previous cases do not hold. Then for some i with $1 \leq i \leq |l|$ we have
  $$\begin{aligned} & Value^p(k, T_{x_1 \ldots x_p \theta(k) l}(x)) \\ \equiv\ & Value^p(k, p + |\theta(k)| + i) \\ \equiv\ & x_i \\ \equiv\ & Value^p(k', p + |\theta(k')| + i) \\ \equiv\ & Value^p(k', T_{x_1 \ldots x_p \theta(k') l}(x)) \end{aligned}$$
  - $x \notin \{x_1, \ldots, x_p\}$, $x \notin \theta(k) \cup \theta(k')$, $x \notin l$. Then we have
  $$\begin{aligned} & Value^p(k, T_{x_1 \ldots x_p \theta(k) l}(x)) \\ \equiv\ & Value^p(k, \mathcal{O}(x) + p + |\theta(k) l|) \\ \equiv\ & x_{\mathcal{O}(x) + |l|} \\ \equiv\ & Value^p(k', \mathcal{O}(x) + p + |\theta(k') l|) \\ \equiv\ & Value^p(k', T_{x_1 \ldots x_p \theta(k') l}(x)) \end{aligned}$$
- $t \equiv f(t_1, \ldots, t_n)$. By induction hypothesis we have $Value^p(k, T_{x_1 \ldots x_p \theta(k) l}(t_i)) \equiv Value^p(k', T_{x_1 \ldots x_p \theta(k') l}(t_i))$ so the property trivially holds.
- $t \equiv \xi x.(t_1, \ldots, t_n)$. Then we have
$$\begin{aligned} & Value^p(k, T_{x_1 \ldots x_p \theta(k) l}(\xi x.(t_1, \ldots, t_n))) \\ \equiv\ & Value^p(k, \xi(T_{x x_1 \ldots x_p \theta(k) l}(t_1), \ldots, T_{x x_1 \ldots x_p \theta(k) l}(t_n))) \\ \equiv_{i.h.}\ & \xi(Value^{p+1}(k, T_{x x_1 \ldots x_p \theta(k) l}(t_1)), \ldots, Value^{p+1}(k, T_{x x_1 \ldots x_p \theta(k) l}(t_n))) \\ \equiv\ & \xi(Value^{p+1}(k', T_{x x_1 \ldots x_p \theta(k') l}(t_1)), \ldots, Value^{p+1}(k', T_{x x_1 \ldots x_p \theta(k') l}(t_n))) \\ \equiv\ & Value^p(k', T_{x_1 \ldots x_p \theta(k') l}(\xi x.(t_1, \ldots, t_n))) \end{aligned}$$

We can finally conclude with the main result of this section which ensures that the $SERS_{DB}$ formalism preserves the semantics of *SERS* reduction.

Proposition 1 (Simulating *SERS* reduction via $SERS_{DB}$ reduction). *Suppose $s \longrightarrow t$ in the SERS formalism using the rewrite rule (G, D). Then $T(s) \longrightarrow T(t)$ in the $SERS_{DB}$ formalism using the de Bruijn rewrite rule $T(G, D)$.*

Proof. By definition of reduction (Definition 15) there is an admissible valuation θ for (G, D) and there is a context C such that $s \equiv_\alpha C[\theta G]$ and $t \equiv_\alpha C[\theta D]$. By Lemma 5 $T(s) \equiv T(C[\theta G])$ and $T(t) \equiv T(C[\theta D])$. Note that $T(G, D) = (T(G), T(D))$ is a de Bruijn rewrite rule by Lemma 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 $T(s) \equiv E[\kappa T(G)]$ and $T(t) \equiv E[\kappa T(D)]$; Step 2 consists in showing that κ is a valid de Bruijn valuation for $(T(G), T(D))$.

- Step 1. By Lemma 10 we have $T(s) \equiv T(C)[T_k(\theta G)]$ and $T(t) \equiv T(C)[T_k(\theta D)]$, where k is the label 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 11 so that $T_k(\theta G) \equiv T_k(\theta)(T(G))$ and $T_k(\theta D) \equiv T_k(\theta)(T(D))$, where $T_k(\theta)$ is a de Bruijn valuation. Thus we may take $\kappa \stackrel{\text{def}}{=} T_k(\theta)$ and $E \stackrel{\text{def}}{=} T(C)$.
- Step 2. We have still to show that $T_k(\theta)$ is valid for $(T(G), T(D))$. By Definition 26 we have to check that $Value(l, T_k(\theta)(X_l)) \equiv Value(l', T_k(\theta)(X_{l'}))$ 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 31, that $Value(l, T_{\theta(l)k}(\theta(X))) \equiv Value(l', T_{\theta(l')k}(\theta(X)))$. In order to apply Lemma 12 with $p = 0$ we must verify its conditions:
  - $\theta(X)$ is a term in $\mathcal{T}$ by definition of valuations.
  - θ is injective on bound o-metavariables since it is admissible,
  - finally, we need to show that for every variable $z \in FV(\theta X)$ 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.

  Thus by Lemma 12 we are done.

5 From $SERS_{DB}$ to *SERS*

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

5.1 From $SERS_{DB}$ to *SERS*: the definition phase

In this section we study the definition phase, that is, we define translations from the $SERS_{DB}$ to the *SERS* formalism for all the objects used in the reduction process.

Definition 32 (From de Bruijn terms (and contexts) to terms (and contexts)). *We define the translation of $a \in \mathcal{T}_{db}$, denoted $U(a)$, as $U_\epsilon^{\mathtt{Names}(FV(a))}(a)$ where, for every finite set of variables S, and every label of variables k, $U_k^S(a)$ is defined as follows:*

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

$$\begin{array}{lll} U_k^S(f(a_1,\ldots,a_n)) & \stackrel{\text{def}}{=} & f(U_k^S(a_1),\ldots,U_k^S(a_n)) \\ U_k^S(\xi(a_1,\ldots,a_n)) & \stackrel{\text{def}}{=} & \xi x.(U_{xk}^S(a_1),\ldots,U_{xk}^S(a_n)) \quad \textit{for any } x \notin k \cup S \end{array}$$

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

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

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 $\mathtt{Names}(FV(a) \setminus\!\!\setminus |k|) \subseteq S$. It is quite evident that $FV(\xi(a)) \setminus\!\!\setminus n$ is exactly $FV(a) \setminus\!\!\setminus (n+1)$. Also, if $U_k^S(C[a])$ is defined and $|l|$ is the binder path number of C (see Definition 19), then $U_{lk}^S(a)$ is also defined. Note, moreover, that if $x \in FV(U_k^S(a))$ then $x \in S \cup k$.

Definition 33 (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:*

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

As in Definition 32 we remark that 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 34 (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 $(U_\epsilon(L), U_\epsilon(R))$.*

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 10 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 11. Consider the de Bruijn rule $@(\Delta(X_\alpha), Z_\epsilon) \longrightarrow \Delta(X_{\alpha\beta}[\![\lambda(@(\mathtt{S}(1), @(1, Z_{\gamma\beta})))]\!])$ from Example 10. The rule obtained by the translation of Definition 33 is $@(\Delta\alpha.(X), Z) \longrightarrow \Delta\beta.(X[\alpha \leftarrow \lambda\gamma.(@(\beta, @(\gamma, Z)))])$. Whereas, for the rule $\Sigma(\sigma(\mathtt{S}(\widehat{\beta})), Z_\epsilon) \longrightarrow \widehat{\beta}$ we obtain $\Sigma(\sigma\gamma.(\widehat{\beta}), Z) \longrightarrow \widehat{\beta}$ for some bound o-metavariable γ.

Definition 35 (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*

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

The translation of κ is the valuation $U_{(\theta_v,S,k)}(\kappa) \stackrel{\text{def}}{=} (\theta_v, \theta_t)$, where $\theta_t X \stackrel{\text{def}}{=} U^S_{\theta_v(l)k}(\kappa X_l)$ for X_l in $Dom(\kappa)$.

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

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

The last condition on θ_v says that if an i-metavariable in A is bound (or free) in the context 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 first need the following.

Lemma 13. *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. $\mathtt{Names}(FV(a) \setminus\!\!\setminus |l|) \subseteq S$.

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

Proof. The condition $\mathtt{Names}(FV(a) \setminus\!\!\setminus |l|) \subseteq S$ is required for $U^S_l(a)$ and $U^S_{l\{z\leftarrow y\}}(a)$ to be defined. The proof proceeds by induction on t.

- $a \equiv n$. We have two further cases to consider:
  - $1 \leq n \leq |l|$. Then on one hand $U^S_l(n)\{z \leftarrow y\} \equiv \mathtt{at}(l, n)\{z \leftarrow y\} \equiv \mathtt{at}(l\{z \leftarrow y\}, n) \equiv U^S_{l\{z\leftarrow y\}}(n)$.
  - $n > |l|$. Then since $z \notin S$ we have $U^S_l(n)\{z \leftarrow y\} \equiv x_{n-|l|}\{z \leftarrow y\} \equiv x_{n-|l|} \equiv x_{n-||\{l\leftarrow z\}y} \equiv U^S_{l\{z\leftarrow y\}}(n)$.
- $a \equiv f(a_1, \ldots, a_n)$. We use the induction hypothesis.

– $a \equiv \xi(a_1, \ldots, a_n)$. Then we reason as follows:

$$\begin{array}{rll}
U^S_l(a)\{z \leftarrow y\} \equiv & \xi x.(U^S_{xl}(a_1), \ldots, U^S_{xl}(a_n))\{z \leftarrow y\} & \\
\equiv & \xi x.(U^S_{xl}(a_1)\{z \leftarrow y\}, \ldots, U^S_{xl}(a_n)\{z \leftarrow y\}) & (z \in l \text{ hence } z \not\equiv x \\
& & \text{and } y \text{ not in } U^S_l(a)) \\
\equiv_\alpha & \xi x.(U^S_{xl\{z\leftarrow y\}}(a_1), \ldots, U^S_{xl\{z\leftarrow y\}}(a_n)) & \text{(i.h.)} \\
\equiv_\alpha & \xi v.(U^S_{xl\{z\leftarrow y\}}(a_1)\{x \leftarrow v\}, \ldots, U^S_{xl\{z\leftarrow y\}}(a_n)\{x \leftarrow v\}) & (v \text{ fresh}) \\
\equiv_\alpha & \xi v.(U^S_{vl\{z\leftarrow y\}\{x\leftarrow v\}}(a_1), \ldots, U^S_{vl\{z\leftarrow y\}\{x\leftarrow v\}}(a_n)) & \text{(i.h.)} \\
\equiv_\alpha & \xi v.(U^S_{vl\{z\leftarrow y\}}(a_1), \ldots, U^S_{vl\{z\leftarrow y\}}(a_n)) & (x \not\equiv y \text{ and } x \notin l) \\
\equiv_\alpha & \xi v.(U^S_{(wl\{z\leftarrow y\})\{w\leftarrow v\}}(a_1), \ldots, U^S_{(wl\{z\leftarrow y\})\{w\leftarrow v\}}(a_n)) & \\
\equiv_\alpha & \xi v.(U^S_{wl\{z\leftarrow y\}}(a_1)\{w \leftarrow v\}, \ldots, U^S_{wl\{z\leftarrow y\}}(a_n)\{w \leftarrow v\}) & \text{(i.h.)} \\
\equiv_\alpha & \xi w.(U^S_{wl\{z\leftarrow y\}}(a_1), \ldots, U^S_{wl\{z\leftarrow y\}}(a_n)) & (w \in l\{z \leftarrow y\} \cup S) \\
\equiv & U^S_{l\{z\leftarrow y\}}(a) &
\end{array}$$

Lemma 14 **(U is modular w.r.t de Bruijn contexts).** *Let E be a de Bruijn context, l, k labels where $|l|$ is the binder path number of E and $a \in \mathcal{T}_{db}$. If $U^S_k(E[a])$ and $U^S_{lk}(a)$ are defined, then $U^S_k(E[a]) \equiv_\alpha U^S_k(E)[U^S_{lk}(a)]$.*

Proof. By induction on the context E.

– $E \equiv \Box$. Then $l \equiv \epsilon$ and the result holds trivially.
– $E \equiv f(a_1, \ldots, E', \ldots, a_n)$. Note that the binder path number of E and E' are the same. We reason as follows:

$$\begin{array}{rll}
U^S_k(f(a_1, \ldots, E'[a], \ldots, a_n)) \equiv & f(U^S_k(a_1), \ldots, U^S_k(E'[a]), \ldots, U^S_k(a_n)) & \\
\equiv_\alpha & f(U^S_k(a_1), \ldots, U^S_k(E')[U^S_{lk}(a)], \ldots, U^S_k(a_n)) & \text{(i.h.)} \\
\equiv & U^S_k(E)[U^S_{lk}(a)] &
\end{array}$$

– $E \equiv \xi(a_1, \ldots, E', \ldots, a_n)$. Since l is a label such that $|l|$ is the binder path number of E, we have $l \equiv l'y$ for some label l' and variable y. We now reason as follows:

$$\begin{array}{rll}
& U^S_k(\xi(a_1, \ldots, E'[a], \ldots, a_n)) & \\
\equiv & \xi x.(U^S_{xk}(a_1), \ldots, U^S_{xk}(E'[a]), \ldots, U^S_{xk}(a_n)) & \\
\equiv_\alpha & \xi z.(U^S_{xk}(a_1)\{x \leftarrow z\}, \ldots, U^S_{xk}(E'[a])\{x \leftarrow z\}, \ldots, U^S_{xk}(a_n)\{x \leftarrow z\}) & (z \text{ fresh}) \\
\equiv_\alpha & \xi z.(U^S_{zk}(a_1), \ldots, U^S_{zk}(E'[a]), \ldots, U^S_{zk}(a_n)) & \text{(L.13)} \\
\equiv_\alpha & \xi z.(U^S_{yk}(a_1)\{y \leftarrow z\}, \ldots, U^S_{yk}(E'[a])\{y \leftarrow z\}, \ldots, U^S_{yk}(a_n)\{y \leftarrow z\}) & \text{(L.13)} \\
\equiv_\alpha & \xi y.(U^S_{yk}(a_1), \ldots, U^S_{yk}(E'[a]), \ldots, U^S_{yk}(a_n)) & \\
\equiv_\alpha & \xi y.(U^S_{yk}(a_1), \ldots, U^S_{yk}(E')[U^S_{l'yk}(a)], \ldots, U^S_{yk}(a_n)) & \text{(i.h.)} \\
\equiv & U^S_k(E)[U^S_{lk}(a)] &
\end{array}$$

Note that $x \notin k$ and $y \notin k$ for otherwise they could not have been chosen by the $U^{\cdot}_{\cdot}(.)$ translation mapping as candidate variables for binding.

Lemma 15. *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: $Value^{j+1}(l, a) \equiv Value^{j+1}(l', b)$ implies $Value^j(\alpha l, a) \equiv Value^j(\alpha l', b)$.*

Proof. By induction on a.

- $a \equiv m$. Since $Value^{j+1}(l,m) \equiv Value^{j+1}(l',b)$ we have $b \equiv n$ for some index n. We proceed by cases:
  - $m \leq j+1$. Then since $Value^{j+1}(l,m) \equiv m \equiv Value^{j+1}(l',n)$ by Definition 25 we have $n \equiv m$ and therefore $Value^{j}(\alpha l,m) \equiv Value^{j}(\alpha l',n)$.
  - $m > j+1$. We have two different cases:
    * $m-(j+1) \leq |l|$. Then by hypothesis we have $Value^{j+1}(l,m) \equiv \mathtt{at}(l,m-(j+1)) \equiv Value^{j+1}(l',n)$, and hence $0 < n-(j+1) \leq |l'|$ and $\mathtt{at}(l,m-(j+1)) \equiv \mathtt{at}(l',n-(j+1))$. Therefore $Value^{j}(\alpha l,m) \equiv Value^{j}(\alpha l',n)$ since $1 < m-j \leq |\alpha l|$ and $1 < n-j \leq |\alpha l'|$.
    * $m-(j+1) > |l|$. Then by hypothesis we have $Value^{j+1}(l,m) \equiv x_{m-(j+1)-|l|} \equiv Value^{j+1}(l',n)$, and hence $n-(j+1) > |l'|$ and $m-(j+1)-|l| \equiv n-(j+1)-|l'|$. Therefore $Value^{j}(\alpha l,m) \equiv x_{m-j-|\alpha l|} \equiv Value^{j}(\alpha l',n)$.
- $a = f(a_1,\ldots,a_n)$. By Definition 25 and the hypothesis we have necessarily that $b \equiv f(b_1,\ldots,b_n)$ and $Value^{j+1}(l,a_i) \equiv Value^{j+1}(l',b_i)$ for $1 \leq i \leq n$ so that by induction hypothesis we can conclude $Value^{j}(\alpha l,a_i) \equiv Value^{j}(\alpha l',b_i)$ and thus $Value^{j}(\alpha l,a) \equiv Value^{j}(\alpha l',b)$.
- $a = \xi(a_1,\ldots,a_n)$. By Definition 25 and the hypothesis we have necessarily that $b \equiv \xi(b_1,\ldots,b_n)$ and $Value^{j+2}(l,a_i) \equiv Value^{j+2}(l',b_i)$ for $1 \leq i \leq n$ so that by induction hypothesis we can conclude $Value^{j+1}(\alpha l,a_i) \equiv Value^{j+1}(\alpha l',b_i)$ and thus $Value^{j}(\alpha l,a) \equiv Value^{j}(\alpha l',b)$.

Note that the converse of Lemma 15 does not hold (for α may already be present in l or l'). Indeed, $Value^{0}(\alpha\alpha,2) \equiv Value^{0}(\alpha\alpha,1)$, yet $Value^{1}(\alpha\alpha,2) \not\equiv Value^{1}(\alpha\alpha,1)$. 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 $U_{\cdot}^{\cdot}(.)$ translation in the $SERS$ formalism. The following lemma investigates this matter.

Lemma 16. *Let $a,b \in \mathcal{T}_{db}$, S be a set of name 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 $Value(l,a) \equiv Value(l',b)$ implies $U^S_{\theta_v(l)k}(a) \equiv_\alpha U^S_{\theta_v(l')k}(b)$.*

Proof. By induction on a.

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

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

  We now consider the following cases:
  - $m \leq |l|$. Then since $Value^{0}(l,m) \equiv \mathtt{at}(l,m) \equiv Value^{0}(l',n)$ we have $n \leq |l'|$ and $\mathtt{at}(l,m) \equiv \mathtt{at}(l',n)$. Then $U^S_{\theta_v(l')k}(n) \equiv \mathtt{at}(\theta_v(l'),n) \equiv \theta_v(\mathtt{at}(l',n)) \equiv \theta_v(\mathtt{at}(l,m)) \equiv U^S_{\theta_v(l)k}(m)$.
  - $|l| < m \leq |lk|$. Then since $Value^{0}(l,m) \equiv x_{m-|l|} \equiv Value^{0}(l',n)$ we have $n > |l'|$ and $x_{m-|l|} \equiv x_{n-|l'|}$. Then $m-|l| \equiv n-|l'|$. Thus $U^S_{\theta_v(l')k}(n) \equiv \mathtt{at}(k,n-|l'|) \equiv \mathtt{at}(k,m-|l|) \equiv U^S_{\theta_v(l)k}(m)$.
  - $m > |lk|$. Then since $Value^{0}(l,m) \equiv x_{m-|l|} \equiv Value^{0}(l',n)$ we have $n > |l'|$ and $x_{m-|l|} \equiv x_{n-|l'|}$. Then $m-|l| \equiv n-|l'|$ and since $m > |lk|$ we also have $n > |l'k|$. Thus $U^S_{\theta_v(l')k}(n) \equiv x_{n-|l'k|} \equiv x_{m-|lk|} \equiv U^S_{\theta_v(l)k}(m)$.

– $a \equiv f(a_1, \ldots, a_n)$. Since $Value^0(l, a) \equiv Value^0(l', b)$ we have $b \equiv f(b_1, \ldots, b_n)$ and $Value^0(l, a_i) \equiv Value^0(l', b_i)$ for $1 \le i \le n$. Then by the induction hypothesis we have $U^S_{\theta_v(l)k}(a_i) \equiv_\alpha U^S_{\theta_v(l')k}(b_i)$ and hence $U^S_{\theta_v(l)k}(a) \equiv_\alpha U^S_{\theta_v(l')k}(b)$.
– $a \equiv \xi(a_1, \ldots, a_n)$. Since $Value^0(l, a) \equiv Value^0(l', b)$ we have $b \equiv \xi(b_1, \ldots, b_n)$ and $Value^1(l, a_i) \equiv Value^1(l', b_i)$ for $1 \le i \le n$. Then $Value^0(\beta l, a_i) \equiv Value^0(\beta l', b_i)$ holds by Lemma 15, 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 $U^S_{\theta_v(\beta l)k}(a_i) \equiv U^S_{z\theta_v(l)k}(a_i) \equiv_\alpha U^S_{z\theta_v(l')k}(b_i) \equiv U^S_{\theta_v(\beta l')k}(b_i)$ for $1 \le i \le n$.
We now reason as follows:

$$
\begin{array}{lll}
 & U^S_{\theta_v(l)k}(\xi(a_1, \ldots, a_n)) & \\
\equiv & \xi x.(U^S_{x\theta_v(l)k}(a_1), \ldots, U^S_{x\theta_v(l)k}(a_n)) & (x \notin \theta_v(l)k \cup S) \\
\equiv_\alpha & \xi z.(U^S_{x\theta_v(l)k}(a_1)\{x \leftarrow z\}, \ldots, U^S_{x\theta_v(l)k}(a_n)\{x \leftarrow z\}) & \\
\equiv & \xi z.(U^S_{z\theta_v(l)k}(a_1), \ldots, U^S_{z\theta_v(l)k}(a_n)) & (\text{L.13}) \\
\equiv_\alpha & \xi z.(U^S_{z\theta_v(l')k}(b_1), \ldots, U^S_{z\theta_v(l')k}(b_n)) & (\text{i.h.}) \\
\equiv & \xi z.(U^S_{y\theta_v(l')k}(b_1)\{y \leftarrow z\}, \ldots, U^S_{y\theta_v(l')k}(b_n)\{y \leftarrow z\}) & (\text{L.13}) \\
\equiv_\alpha & \xi y.(U^S_{y\theta_v(l')k}(b_1), \ldots, U^S_{y\theta_v(l')k}(b_n)) & (y \notin \theta_v(l')k \cup S \text{ by Def. 32}) \\
\equiv & U^S_{\theta_v(l')k}(\xi(b_1, \ldots, b_n)) &
\end{array}
$$

Note that, in general, the converse of Lemma 16 does not hold. Indeed it suffices to consider $k \equiv \epsilon$, $l \equiv \alpha$, $l' \equiv \beta$, $a \equiv 1$, $b \equiv 1$, $S = \emptyset$ and the variable assignment $\theta_v\alpha \equiv \theta_v\beta \equiv x$. Then $U^S_x(a) \equiv x \equiv U^S_x(b)$ but $Value(\alpha, 1) \equiv \alpha \not\equiv \beta \equiv Value(\beta, 1)$.

We can now show that the translation of de Bruijn valuations is correct in the sense mentioned above.

Lemma 17 (Translation of de Bruijn valuations is correct). *The valuation $U_{(\theta,S,k)}(\kappa)$ given in Definition 35 is correct if κ is valid, where correct means that for every X_l and $X_{l'}$ in $Dom(\kappa)$ we have $U^S_{\theta_v(l)k}(\kappa X_l) \equiv_\alpha U^S_{\theta_v(l')k}(\kappa X_{l'})$, whenever both terms are defined.*

Proof. Since κ is valid we have $Value(l, \kappa X_l) \equiv Value(l', \kappa X_{l'})$ for every X_l and X'_l in $Dom(\kappa)$. Then by Lemma 16 we may conclude $U^S_{\theta_v(l)k}(\kappa X_l) \equiv_\alpha U^S_{\theta_v(l')k}(\kappa X_{l'})$.

5.2 From $SERS_{DB}$ into $SERS$: the preservation-of-reduction phase

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

Lemma 18. *Let $a \in \mathcal{T}_{db}$, l_1, l_2 and k be labels of variables with $|l_1| = j$ and $|l_2| = i - 1$. Then $U^S_{l_1 l_2 k}(\mathcal{U}^i_j(a)) \equiv_\alpha U^S_{l_1 k}(a)$ if $U^S_{l_1 k}(a)$ is defined.*

Proof. We proceed by induction on a.

– $a \equiv n$. We have two cases to consider:
  - $n \le j$. Then $U^S_{l_1 l_2 k}(\mathcal{U}^i_j(n)) \equiv U^S_{l_1 l_2 k}(n) \equiv \mathtt{at}(l_1 l_2 k, n) \equiv \mathtt{at}(l_1 k, n) \equiv U^S_{l_1 k}(n)$.
  - $n > j$. Then $U^S_{l_1 l_2 k}(\mathcal{U}^i_j(n)) \equiv U^S_{l_1 l_2 k}(n + i - 1)$ 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 $U^S_{l_1 l_2 k}(n+i-1) \equiv \mathtt{at}(l_1 l_2 k, n+i-1) \equiv \mathtt{at}(l_1 k, n) \equiv U^S_{l_1 k}(n)$.
* $n+i-1 > |l_1 l_2 k|$. Then since $n > |l_1 k|$ we have $U^S_{l_1 l_2 k}(n+i-1) \equiv x_{n+i-1-|l_1 l_2 k|} \equiv x_{n-|l_1 k|} \equiv U^S_{l_1 k}(n)$.

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

$$\begin{array}{lll}
 & U^S_{l_1 l_2 k}(\mathcal{U}^i_j(a)) & \\
\equiv & \xi z.(U^S_{z l_1 l_2 k}(\mathcal{U}^i_{j+1}(a_1)), \ldots, U^S_{z l_1 l_2 k}(\mathcal{U}^i_{j+1}(a_1))) & \\
\equiv_\alpha & \xi z.(U^S_{z l_1 k}(a_1), \ldots, U^S_{z l_1 k}(a_n)) & \text{(i.h.)} \\
\equiv_\alpha & \xi z'.(U^S_{z l_1 k}(a_1)\{z \leftarrow z'\}, \ldots, U^S_{z l_1 k}(a_n)\{z \leftarrow z'\}) & (z' \text{ fresh}) \\
\equiv_\alpha & \xi z'.(U^S_{z' l_1 k}(a_1), \ldots, U^S_{z' l_1 k}(a_n)) & \text{(L.13)} \\
\equiv_\alpha & \xi z'.(U^S_{y l_1 k}(a_1)\{y \leftarrow z'\}, \ldots, U^S_{y l_1 k}(a_n)\{y \leftarrow z'\}) & \text{(L.13)} \\
\equiv_\alpha & \xi y.(U^S_{y l_1 k}(a_1), \ldots, U^S_{y l_1 k}(a_n)) & \\
\equiv & U^S_{l_1 k}(a) &
\end{array}$$

The phrase "z' fresh" should be read, in full rigor, as "z' does not occur in $U^S_{z l_1 k}(a_i)$ nor in $U^S_{y l_1 k}(a_i)$ for $1 \leq i \leq n$". The definition of $U^{\cdot}_{\cdot}(.)$ and the hypothesis that $U^S_{l_1 k}(a)$ is defined, allows us to apply Lemma 13 above.

Lemma 19. *Let $a, b \in \mathcal{T}_{db}$, l and k labels of variables with $|l| = i-1$, x a variable such that $x \notin lk \cup S$. Then $U^S_{lk}(a\{\!\{i \leftarrow b\}\!\}) \equiv_\alpha U^S_{lxk}(a)\{x \leftarrow U^S_k(b)\}$, assuming both sides of the equation are defined.*

Proof. The proof is by induction on a.

– $a \equiv n$. We have three further cases to consider:
• $n < i$. Then we reason as follows:

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

• $n > i$. Then since $U^S_{lk}(a\{\!\{i \leftarrow b\}\!\}) \equiv U^S_{lk}(n-1)$ we consider two further cases:
* $n-1 \leq |lk|$. We reason as follows:

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

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

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

- $n = i$. Then on one hand we have $U^S_{lk}(i\{\!\{i \leftarrow b\}\!\}) \equiv U^S_{lk}(\mathcal{U}^i_0(b))$. And on the other $U^S_{lxk}(i)\{x \leftarrow U^S_k(b)\} \equiv x\{x \leftarrow U^S_k(b)\} \equiv U^S_k(b)$. Lemma 18 concludes this case.

– $a \equiv f(a_1, \ldots, a_n)$. We use the induction hypothesis.

– $a \equiv \xi(a_1, \ldots, a_n)$. Then we reason as follows:

$$
\begin{array}{lll}
 & U^S_{lk}(a\{\!\{i \leftarrow b\}\!\}) & \\
\equiv & \xi z.(U^S_{zlk}(a_1\{\!\{i+1 \leftarrow b\}\!\}), \ldots, U^S_{zlk}(a_n\{\!\{i+1 \leftarrow b\}\!\})) & (z \notin lk \cup S \text{ by Def. 32}) \\
\equiv_\alpha & \xi y.(U^S_{zlk}(a_1\{\!\{i+1 \leftarrow b\}\!\})\{z \leftarrow y\}, \ldots, U^S_{zlk}(a_n\{\!\{i+1 \leftarrow b\}\!\})\{z \leftarrow y\}) & (y \text{ fresh}) \\
\equiv_\alpha & \xi y.(U^S_{ylk}(a_1\{\!\{i+1 \leftarrow b\}\!\}), \ldots, U^S_{ylk}(a_n\{\!\{i+1 \leftarrow b\}\!\})) & (\text{L.13}) \\
\equiv_\alpha & \xi y.(U^S_{ylxk}(a_1)\{x \leftarrow U^S_k(b)\}, \ldots, U^S_{ylxk}(a_n)\{x \leftarrow U^S_k(b)\}) & (\text{i.h.}) \\
\equiv_\alpha & \xi y.(U^S_{(z'lxk)\{z' \leftarrow y\}}(a_1)\{x \leftarrow U^S_k(b)\}, \ldots, U^S_{(z'lxk)\{z' \leftarrow y\}}(a_n)\{x \leftarrow U^S_k(b)\}) & (\text{see below}) \\
\equiv_\alpha & \xi y.(U^S_{z'lxk}(a_1)\{z' \leftarrow y\}\{x \leftarrow U^S_k(b)\}, \ldots, U^S_{z'lxk}(a_n)\{z' \leftarrow y\}\{x \leftarrow U^S_k(b)\}) & (\text{L.13}) \\
\equiv_\alpha & \xi y.(U^S_{z'lxk}(a_1)\{x \leftarrow U^S_k(b)\}\{z' \leftarrow y\}, \ldots, U^S_{z'lxk}(a_n)\{x \leftarrow U^S_k(b)\}\{z' \leftarrow y\}) & (\text{Subst.L.}) \\
\equiv_\alpha & \xi z'.(U^S_{z'lxk}(a_1)\{x \leftarrow U^S_k(b)\}, \ldots, U^S_{z'lxk}(a_n)\{x \leftarrow U^S_k(b)\}) & \\
\equiv & U^S_{lxk}(a)\{x \leftarrow U^S_k(b)\} &
\end{array}
$$

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 $z' \notin FV(U^S_k(b))$.

"Subst.L." refers to a generalization of the substitution lemma for the λ-calculus [12], which is also valid for our restricted notion of substitution, and which says that: $s\{x \leftarrow t\}\{y \leftarrow u\} \equiv s\{y \leftarrow u\}\{x \leftarrow t\{y \leftarrow u\}\}$ if $x \notin FV(u)$ for distinct variables x and y, and both sides of the equation together with the term $t\{y \leftarrow u\}$ are defined.

Lemma 20 (U is modular w.r.t valuations). *Let us 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 35, 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. *$\texttt{Names}(FV(\kappa A) \setminus\!\!\setminus |\theta_v(l)k|) \subseteq S$, and*
6. *$\mathcal{WF}_l(A)$.*

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

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

Proof. By induction on A. Below we shall use lhs and rhs to denote the left and right hand side respectively, of the equation to prove.

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

$$
\begin{array}{ll}
rhs & \equiv U_{(\theta_v,S,k)}(\kappa)(U_l(X_l)) \\
 & \equiv U_{(\theta_v,S,k)}(\kappa)X \\
 & \equiv U^S_{\theta_v(l')k}(\kappa X_{l'}) \qquad (\text{ with } X_{l'} \in Dom(\kappa))
\end{array}
$$

Then since κ is valid (Hypothesis 1) we may apply Lemma 17 to conclude.

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

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

On the other hand we have

$$\begin{aligned} rhs &\equiv U_{(\theta_v,S,k)}(\kappa)(U_l(\mathtt{S}^{|l|}(\widehat{\alpha}))) \\ &\equiv U_{(\theta_v,S,k)}(\kappa)\widehat{\alpha} \\ &\equiv \theta_v(\widehat{\alpha}) \end{aligned}$$

We can conclude $lhs \equiv rhs$ because θ_v satisfies the requirements of Definition 35.

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

$$\begin{aligned} lhs &\equiv U^S_{\theta_v(l)k}(\kappa\mathtt{S}^j(1)) \\ &\equiv U^S_{\theta_v(l)k}(\mathtt{S}^j(1)) \\ &\equiv \mathtt{at}(\theta_v(l), j+1) \\ &\equiv \theta_v(\mathtt{at}(l, j+1)) \end{aligned} \qquad \begin{aligned} rhs &\equiv U_{(\theta_v,S,k)}(\kappa)(U_l(\mathtt{S}^j(1))) \\ &\equiv U_{(\theta_v,S,k)}(\kappa)(\mathtt{at}(l, j+1)) \\ &\equiv \theta_v(\mathtt{at}(l, j+1)) \end{aligned}$$

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

$$\begin{aligned} rhs &\equiv U_{(\theta_v,S,k)}(\kappa)(U_l(\xi(A_1,\ldots,A_n))) \\ &\equiv U_{(\theta_v,S,k)}(\kappa)(\xi\alpha.(U_{\alpha l}(A_1),\ldots,U_{\alpha l}(A_n))) && \text{(where } \alpha \text{ is such that } 1 \leq i \leq n \;\; \mathcal{WF}_{\alpha l}(A_i), \text{ for } \alpha \notin l) \\ &\equiv \xi\theta_v(\alpha).(U_{(\theta_v,S,k)}(\kappa)(U_{\alpha l}(A_1)),\ldots,U_{(\theta_v,S,k)}(\kappa)(U_{\alpha l}(A_n)))) \end{aligned}$$

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 $U_l(A) \equiv \xi\alpha.(U_{\alpha l}(A_1),\ldots,U_{\alpha l}(A_n))$, hence it is defined over the bound o-metavariables in $U_{\alpha l}(A_i) \cup \alpha l$ for $1 \leq i \leq n$. We have then to verify the Hypothesis 5, that is, $\mathtt{Names}(FV(\kappa A_i) \backslash\!\!\backslash |\theta_v(\alpha l)k|) \subseteq S$; but this is evident by the Hypothesis 5 for A and the general fact that $FV(\xi(a_1,\ldots,a_n)) \backslash\!\!\backslash n$ implies $FV(a_1,\ldots,a_n) \backslash\!\!\backslash n+1$. 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 we have:

$$\begin{aligned} rhs \equiv\;& \xi\theta_v(\alpha).(U^S_{\theta_v(\alpha l)k}(\kappa A_1),\ldots,U^S_{\theta_v(\alpha l)k}(\kappa A_n)) \\ \equiv_\alpha\;& \xi z'.((U^S_{\theta_v(\alpha l)k}(\kappa A_1))\{\theta_v(\alpha) \leftarrow z'\},\ldots,(U^S_{\theta_v(\alpha l)k}(\kappa A_n))\{\theta_v(\alpha) \leftarrow z'\}) && (z' \text{ does not occur in } U^S_{\theta_v(\alpha l)k}(\kappa A_i)) \\ \equiv_\alpha\;& \xi z'.(U^S_{(\theta_v(\alpha l)k)\{\theta_v(\alpha)\leftarrow z'\}}(\kappa A_1),\ldots,U^S_{(\theta_v(\alpha l)k)\{\theta_v(\alpha)\leftarrow z'\}}(\kappa A_n)) && \text{(L.13)} \\ \equiv_\alpha\;& \xi z'.(U^S_{z'\theta_v(l)k}(\kappa A_1),\ldots,U^S_{z'\theta_v(l)k}(\kappa A_n)) && \text{(injectivity of } \theta_v) \\ \equiv\;\;& \xi z'.(U^S_{(z\theta_v(l)k)\{z\leftarrow z'\}}(\kappa A_1),\ldots,U^S_{(z\theta_v(l)k)\{z\leftarrow z'\}}(\kappa A_n)) && (z \notin \theta_v(l)k \cup S) \\ \equiv_\alpha\;& \xi z'.(U^S_{z\theta_v(l)k}(\kappa A_1)\{z \leftarrow z'\},\ldots,U^S_{z\theta_v(l)k}(\kappa A_n)\{z \leftarrow z'\}) && \text{(L.13)} \\ \equiv_\alpha\;& \xi z.(U^S_{z\theta_v(l)k}(\kappa A_1),\ldots,U^S_{z\theta_v(l)k}(\kappa A_n)) \end{aligned}$$

On the other hand we have

$$\begin{aligned} lhs &\equiv U^S_{\theta_v(l)k}(\kappa(\xi(A_1,\ldots,A_n))) \\ &\equiv U^S_{\theta_v(l)k}(\xi(\kappa A_1,\ldots,\kappa A_n)) \\ &\equiv \xi z.(U^S_{z\theta_v(l)k}(\kappa A_1),\ldots,U^S_{z\theta_v(l)k}(\kappa A_n)) \;\; (z \notin \theta_v(l)k \cup S) \end{aligned}$$

– $A \equiv f(A_1, \ldots, A_n)$. Then we have

$$\begin{aligned} lhs &\equiv U^S_{\theta_v(l)k}(\kappa(f(A_1,\ldots,A_n))) \\ &\equiv U^S_{\theta_v(l)k}(f(\kappa A_1,\ldots,\kappa A_n)) \\ &\equiv f(U^S_{\theta_v(l)k}(\kappa A_1),\ldots,U^S_{\theta_v(l)k}(\kappa A_n)) \end{aligned} \qquad \begin{aligned} rhs &\equiv U_{(\theta_v,S,k)}(\kappa)(U_l(f(A_1,\ldots,A_n))) \\ &\equiv U_{(\theta_v,S,k)}(\kappa) f(U_l(A_1),\ldots,U_l(A_n)) \\ &\equiv f(U_{(\theta_v,S,k)}(\kappa)U_l(A_1),\ldots,U_{(\theta_v,S,k)}(\kappa)U_l(A_n)) \end{aligned}$$

We can immediately conclude by induction hypothesis.

– $A \equiv A_1[\![A_2]\!]$. Then we have

$$\begin{aligned} lhs &\equiv U^S_{\theta_v(l)k}(\kappa(A_1[\![A_2]\!])) \\ &\equiv U^S_{\theta_v(l)k}(\kappa A_1 \{\!\{1 \leftarrow \kappa A_2\}\!\}) \end{aligned}$$

On the other hand we have

$$\begin{aligned} rhs \equiv\;& U_{(\theta_v,S,k)}(\kappa)(U_l(A_1[\![A_2]\!])) \\ \equiv\;& U_{(\theta_v,S,k)}(\kappa)(U_{\alpha l}(A_1)[\alpha \leftarrow U_l(A_2)]) && \text{(where } \alpha \text{ is such that } 1 \le i \le n \;\; \mathcal{WF}_{\alpha l}(A_1), \\ & && \text{for } \alpha \notin l) \\ \equiv\;& (U_{(\theta_v,S,k)}(\kappa)(U_{\alpha l}(A_1)))\{\theta_v(\alpha) \leftarrow U_{(\theta_v,S,k)}(\kappa)(U_l(A_2))\} \\ \equiv_\alpha\;& U^S_{\theta_v(\alpha l)k}(\kappa A_1)\{\theta_v(\alpha) \leftarrow U^S_{\theta_v(l)k}(\kappa A_2)\} && \text{(i.h.)} \end{aligned}$$

Remark 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 35, then $\theta_v(\alpha) \notin S \cup \theta_v(l)k$ and we can then conclude by applying Lemma 19.

We show now that the translation of a valid de Bruijn valuation is an admissible named valuation. Lemma 21 considers safeness and Lemma 22 goes on to consider admissibility.

Lemma 21 (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 35, S a finite set of variables, k a label of variables, and $U(L,R) \equiv (G,D)$ 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:

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

So first we must verify that for every t-metavariable in (G,D), $U_{(\theta_v,S,k)}(\kappa)$ is indeed defined, but this is guaranteed by Hypothesis 1.

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 $D \equiv C[M_1[\alpha \leftarrow M_2]]$ for some metacontext C[3]),
– θ' is defined for M_1 and M_2,
– $\theta'\alpha \in FV(\theta' M_1)$, and
– for some variable x we have $x \in BV(\theta' M_1)$ and also $x \in FV(\theta' M_2)$.

The metaterm D may be depicted as in Figure 3(a) where l_1 denotes the label of the metacontext C.

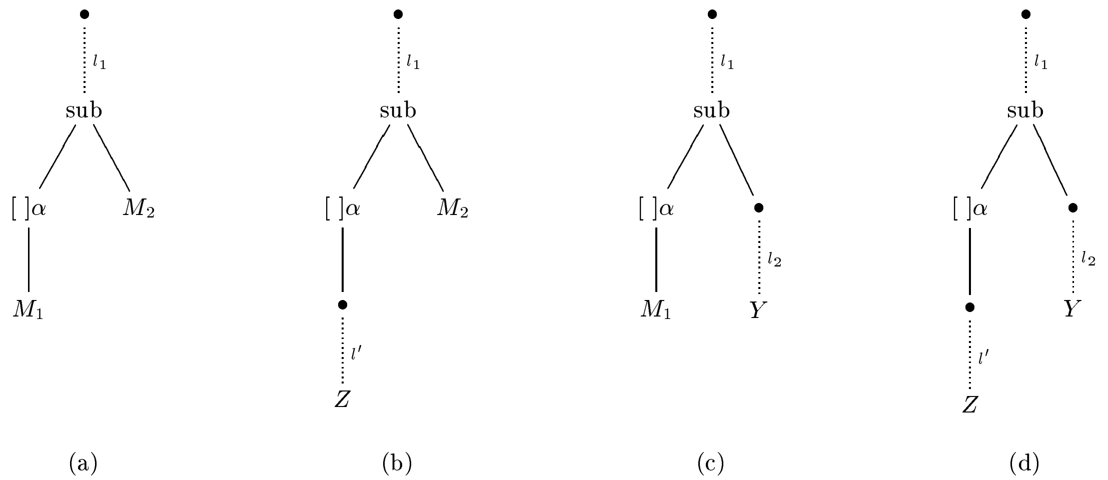


Fig. 3. Tree form for D

Before proceeding we will show the following:

Fact 36 *The free variable occurrence $x \in FV(\theta' M_2)$ cannot be bound by a formal parameter $\beta \in l_1$ (i.e. for all $\beta \in l_1$ we have $\theta'(\beta) \not\equiv x$.). This may be verified by contradiction as follows. Suppose that for some $\beta \in l_1$ we have $\theta'(\beta) \equiv x$. Thus, by definition of $U_{(\theta_v,S,k)}(\kappa)$ we have $\theta_v(\beta) \equiv x$. Let us consider the* bound *occurrence of x in $\theta' M_1$. There are two possibilities:*

1. *x comes from the instantiation of a bound o-metavariable, so that $x \equiv \theta'(\beta')$ for some formal parameter β' in M_1. Now since D is a well-formed pre-metaterm we must have $\beta \not\equiv \beta'$. But $\theta'(\beta')$ is equal to $\theta_v(\beta')$ by definition, so that θ_v assigns the same value, namely x, to two different bound o-metavariables β and β' of D, thus contradicting the Hypothesis 2.*
2. *x comes from the instantiation of a t-metavariable, so that $x \in BV(t)$ with $t \equiv \theta' Y$ for some t-metavariable Y occurring in M_1. By Definition 35 we have*

$$t \equiv U^S_{\theta_v(l)k}(\kappa Y_l)$$

for some t-metavariable Y_l occurring in $Dom(\kappa)$ with $l \equiv l'\alpha l_1$ (see Figure 3(b)). Therefore by definition of the term translation function $U^{\cdot}_{\cdot}(.)$ (Definition 32) the variable x cannot be a candidate for binding since it already occurs in the label $\theta_v(l)k$, indeed, $\beta \in l$ and $\theta_v(\beta) \equiv x$.

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

Thus we have proven that the free variable occurrence x in $\theta' M_2$ cannot be bound by a formal parameter in the label of the metacontext C.

We now return to the proof of the lemma. Let us consider where the free variable occurrence of x 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 $\theta_v(\delta) \equiv x$. As observed above, since x may not be bound by a formal parameter in l_1 (i.e. there is no $\beta \in l_1$ with $\theta_v(\beta) \equiv x$) then $\delta \notin l_1$. Thus $\delta \equiv \widehat{\beta}'$ 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 the Hypothesis 1 $U_{(\theta_v,S,k)}(\kappa)$ is defined on $\widehat{\beta}'$. Now, the assignment θ_v satisfies the requirements of Definition 35, so that in particular we must have $\theta_v(\widehat{\beta}') = x \in S \cup k$.
   We now analyse where the bound occurrence of x comes from in $\theta' M_1$ in order to arrive at a contradiction. Here too we have two cases to consider:
   (a) $x \equiv \theta_v(\beta'')$ for some formal parameter β'' occuring in M_1. Now, $\theta_v(\beta'') \notin S \cup k$ since θ_v satisfies the requirement of Definition 35 by hypothesis, so that we arrive at a contradiction.
   (b) x comes from instantiating some t-metavariable Z in M_1, i.e. $x \in BV(\theta' Z)$ for some t-metavariable Z in M_1 (Figure 3(b)). Thus there is a t-metavariable Z_l with $l \equiv l'\alpha l_1$ in $Dom(\kappa)$, a simple label k' and an index m such that $U^S_{k'\theta_v(l'\alpha l_1)k}(m) \equiv x \equiv \mathtt{at}(k'\theta_v(l'\alpha l_1)k, m)$
   .
   Now since x is bound in $\theta' Z$ we have $m \leq |k'\theta_v(l')|$. But then by definition of $U_{\cdot}(.)$ we have $x \notin S \cup \theta_v(\alpha l_1)k$, in other words, x cannot have been used as a candidate variable for binding. In particular, $x \notin S \cup k$. This is a contradiction since we already know that $x \in S \cup k$.
2. There is an occurrence of a t-metavariable Y in M_2 such that $x \in FV(\theta' Y)$. Then there is an occurrence of Y_l in $Dom(\kappa)$ with $l \equiv l_2 l_1$ such that $x \in FV(U^S_{\theta_v(l_2 l_1)k}(\kappa Y_l))$ where l_1 is the label "above" M_2 (Figure 3(c)). Note that since for this occurrence of x we have $x \in FV(\theta' M_2)$ then we must have that $x \in S$ or $x \in \theta_v(l_1)k$.
   We now analyse where the bound occurrence of x comes from in $\theta' M_1$. Here too we have two cases to consider:
   (a) $x \equiv \theta_v(\beta'')$ for some formal parameter β'' occurring in M_1. If $x \in S$ or $x \in k$ we arrive at a contradiction with the fact that θ_v verifies the requirements of Definition 35 (saying that $\theta_v(\beta'') \notin S \cup k$). Moreover, if $x \in \theta_v(l_1)$ we contradict the above mentioned fact.
   (b) x comes from instantiating some o-metavariable Z in M_1, i.e. $x \in BV(\theta' Z)$ for some t-metavariable Z in M_1 (Figure 3(d)). Thus there is a o-metavariable Z_l in $Dom(\kappa)$ with $l \equiv l'\alpha l_1$, a simple label k' and an index m such that $U^S_{k'\theta_v(l'\alpha l_1)k}(m) \equiv x \equiv \mathtt{at}(k'\theta_v(l'\alpha l_1)k, m)$.
   Now since x is bound in M_1 we have $m \leq |k'\theta_v(l')|$. But then by definition of $U_{\cdot}(.)$ we have $x \notin S \cup \theta_v(\alpha l_1)k$. In particular, $x \notin S \cup \theta_v(l_1)k$. This is a contradiction since we already know that $x \in S$ or $x \in \theta_v(l_1)k$.

Lemma 22 (valid de Bruijn valuations translate to admissible valuations). *Let κ be a valid de Bruijn valuation for a rewrite rule (L, R), θ_v a variable assignment verifying the hypothesis in Definition 35, and $U(L, R) \equiv (G, D)$ 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 21 we have that θ' is safe then by Definition 14 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) occuring at two different positions p and p'. Let us take any variable $x \in FV(\theta' X)$ and let l and l' be the parameter paths of p and p' in the trees corresponding to G or D. Suppose $\theta' X \equiv U_{(\theta_v,S,k)}(\kappa)X =_{def} U^S_{\theta_v(l)k}(\kappa X_l)$. Then since κ is valid by Lemma 17 $U^S_{\theta_v(l)k}(\kappa X_l) \equiv_\alpha U^S_{\theta_v(l')k}(\kappa X_{l'})$. 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 35 $x \notin S \cup k$. 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 bound o-metavariables α and β occur bound in (G, D) with $\alpha \not\equiv \beta$, then $\theta_v\alpha \not\equiv \theta_v\beta$: this property trivially holds by Hypothesis 2.

We can conclude with the main result of this section which ensures that the *SERS* formalism preserves the semantics of $SERS_{DB}$ reduction.

Proposition 2 (Simulating $SERS_{DB}$ reduction via *SERS* reduction). *Suppose $a \longrightarrow b$ in the $SERS_{DB}$ formalism using rewrite rule (L, R). Then $U(a) \longrightarrow U(b)$ in the SERS formalism using rule $U(L, R)$.*

Proof. Let us consider the de Bruijn rewrite step $a \longrightarrow b$ 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 $U(L, R) \equiv (G, D)$. By definition of reduction we have a de Bruijn context E such that $a \equiv E[\kappa L]$ and $b \equiv E[\kappa R]$. We proceed as follows:

- Take S as the set of variables $\mathtt{Names}(FV(a))$ so that $U^S_\epsilon(a)$ is defined. Note that since $FV(b) \subseteq FV(a)$ holds by Corollary 2, $U^S_\epsilon(b)$ is also defined.
- Take any *simple* label k of variables such that $S \cap k = \emptyset$ and $|k|$ is the binder path number of E .
- Now, to apply Lemma 14 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, $U^S_\epsilon(E[\kappa L]) \equiv_\alpha U^S_\epsilon(E)[U^S_k(\kappa L)]$ and $U^S_\epsilon(E[\kappa R]) \equiv_\alpha U^S_\epsilon(E)[U^S_k(\kappa R)]$.
- The next step is to apply Lemma 20 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 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 label of) the context where the reduction takes place, that is, the variables in k).
  - We also define θ_v on the free o-metavariables as the hypothesis dictates, i.e. for all $\widehat{\alpha}$ we define

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

We shall now consider the case of $U_k^S(\kappa L)$, the other one being similar. We must thus meet the conditions of the lemma in order to resolve $U_k^S(\kappa L)$. Let $l \equiv \epsilon$.

1. κ is valid by hypothesis.
2. κL is defined since $a \equiv E[\kappa L]$.
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. $\texttt{Names}(FV(\kappa L) \setminus\setminus |k|)$ holds since by definition we set $S = \texttt{Names}(FV(a))$ (Note that by Corollary 2 we have $\texttt{Names}(FV(b)) \subseteq S$),
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 20.

Let us summarize our situation:

$$\begin{array}{rll} U_\epsilon^S(E[\kappa L]) & \equiv_\alpha U_\epsilon^S(E)[U_k^S(\kappa L)] & \text{(L.14)} \\ & \equiv_\alpha U_\epsilon^S(E)[U_{(\theta_v,S,k)}(\kappa)(U_\epsilon(L))] & \text{(L.20)} \\ & \equiv \quad U_\epsilon^S(E)[U_{(\theta_v,S,k)}(\kappa)G] & \end{array}$$

and

$$\begin{array}{rll} U_\epsilon^S(E[\kappa R]) & \equiv_\alpha U_\epsilon^S(E)[U_k^S(\kappa R)] & \text{(L.14)} \\ & \equiv_\alpha U_\epsilon^S(E)[U_{(\theta_v,S,k)}(\kappa)(U_\epsilon(R))] & \text{(L.20)} \\ & \equiv \quad U_\epsilon^S(E)[U_{(\theta_v,S,k)}(\kappa)D] & \end{array}$$

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

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

We can then apply Lemma 21 and Lemma 22 and conclude that θ' is admissible for (G, D).

6 Preserving confluence

This section studies the relationship between the translation functions over pre-metaterms and terms introduced above. This gives rise to two results stating, respectively, that given a metaterm M then $U(T(M))$ is v-equivalent to M(see Figure 4), and that given a de Bruijn metaterm A then $T(U(A))$ is identical to A. These results are used to show that confluence is preserved when translating an *SERS* rewrite system into an *SERS*$_{DB}$ rewrite system.

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

Lemma 23. *Let $M \in \mathcal{PMT}$ 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. *$\mathcal{WF}_{k'\alpha' l}(M)$ holds.*

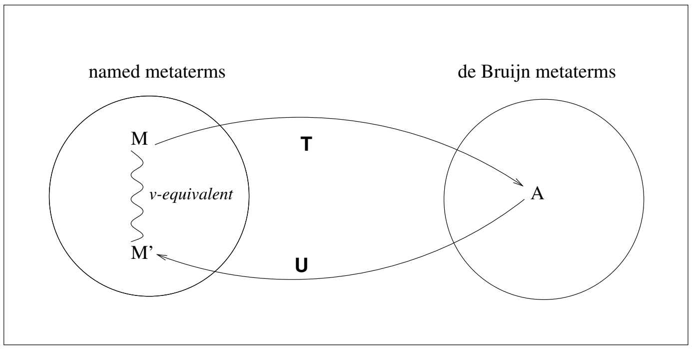


Fig. 4. v-equivalence

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 $k \equiv \beta_1 \ldots \beta_n$ and $k' \equiv \beta'_1 \ldots \beta'_n$. By Hypothesis 3 we have the following cases to consider:

– $M \equiv \alpha'' \in k'$ and hence $\alpha'' \equiv \beta'_j$ for some $1 \leq j \leq n$. Then we have

$$\begin{aligned} U_{k\alpha l}(T_{k'\alpha' l}(M)) \ll\alpha\leftarrow\beta\gg &\equiv \beta_j \ll\alpha\leftarrow\beta\gg \\ &\equiv \beta_j \qquad\qquad \text{(hyp.1)} \\ &\equiv U_{k\beta l}(T_{k'\alpha' l}(M)) \end{aligned}$$

– $M \equiv \alpha'' \in l$ and $\alpha'' \notin k'$. Then we have

$$\begin{aligned} U_{k\alpha l}(T_{k'\alpha' l}(M)) \ll\alpha\leftarrow\beta\gg &\equiv \alpha'' \ll\alpha\leftarrow\beta\gg \\ &\equiv \alpha'' \qquad\qquad \text{(hyp.1)} \\ &\equiv U_{k\beta l}(T_{k'\alpha' l}(M)) \end{aligned}$$

– $M \equiv \alpha'$ and $\alpha' \notin k'$. Then we have

$$\begin{aligned} U_{k\alpha l}(T_{k'\alpha' l}(M)) \ll\alpha\leftarrow\beta\gg &\equiv \alpha \ll\alpha\leftarrow\beta\gg \\ &\equiv \beta \\ &\equiv U_{k\beta l}(T_{k'\alpha' l}(M)) \end{aligned}$$

– $M \equiv \widehat{\alpha}$. Then we have

$$\begin{aligned} U_{k\alpha l}(T_{k'\alpha' l}(M)) \ll\alpha\leftarrow\beta\gg &\equiv \widehat{\alpha} \ll\alpha\leftarrow\beta\gg \\ &\equiv \widehat{\alpha} \\ &\equiv U_{k\beta l}(T_{k'\alpha' l}(M)) \end{aligned}$$

– $M \equiv f(M_1, \ldots, M_n)$. Then we use the induction hypothesis.
– $M \equiv \xi\alpha''.(M_1, \ldots, M_n)$. We reason as follows:

$$(U_{k\alpha l}(T_{k'\alpha' l}(M)))) \ll\alpha\leftarrow\beta\gg \equiv (\xi\beta'.(U_{\beta' k\alpha l}(T_{\alpha'' k'\alpha' l}(M_1)), \ldots, U_{\beta' k\alpha l}(T_{\alpha'' k'\alpha' l}(M_n)))) \ll\alpha\leftarrow\beta\gg$$

for $\beta' \notin k\alpha l$ such that $\mathcal{WF}_{\beta' k\alpha l}(T_{\alpha'' k' \alpha' l}(M_i))$ holds for $1 \leq i \leq n$. Since $\beta \not\equiv \beta'$ by hypothesis 2, we may continue

$$\begin{array}{ll} & (\xi\beta'.(U_{\beta' k\alpha l}(T_{\alpha'' k'\alpha' l}(M_1)), \ldots, U_{\beta' k\alpha l}(T_{\alpha'' k'\alpha' l}(M_n)))) \ll\alpha\leftarrow\beta\gg \\ \equiv & \xi\beta'.((U_{\beta' k\alpha l}(T_{\alpha'' k'\alpha' l}(M_1))) \ll\alpha\leftarrow\beta\gg, \ldots, (U_{\beta' k\alpha l}(T_{\alpha'' k'\alpha' l}(M_n))) \ll\alpha\leftarrow\beta\gg) \text{ by i.h.} \\ =_v & \xi\beta'.(U_{\beta' k\beta l}(T_{\alpha'' k'\alpha' l}(M_1)), \ldots, U_{\beta' k\beta l}(T_{\alpha'' k'\alpha' l}(M_n))) \\ \stackrel{\text{def}}{=} & U_{k\beta l}(T_{k'\alpha' l}(\xi\alpha''.(M_1, \ldots, M_n))) \end{array}$$

– $M \equiv M_1[\alpha' \leftarrow M_2]$. Similar to the previous case.

Lemma 24. *Let $M \in \mathcal{PMT}$ and l a simple label. If $\mathcal{WF}_l(M)$ then $U_l(T_l(M)) =_v M$.*

Proof. By induction on M.

– $M \equiv \alpha$. Then since $\mathcal{WF}_l(M)$ we have $\alpha \in l$ and thus $U_l(T_l(\alpha)) \equiv U_l(\mathtt{pos}(\alpha, l)) \equiv \alpha$.
– $M \equiv \widehat{\alpha}$. Then $U_l(T_l(\widehat{\alpha})) \equiv U_l(\mathtt{S}^{|l|}(\widehat{\alpha})) \equiv \widehat{\alpha}$.
– $M \equiv X$. Then $U_l(T_l(X)) \equiv U_l(X_l) \equiv X$.
– $M \equiv f(M_1, \ldots, M_n)$. We use the induction hypothesis.
– $M \equiv \xi\alpha.(M_1, \ldots, M_n)$. We reason as follows:

$$\begin{array}{rl} U_l(T_l(\xi\alpha.(M_1, \ldots, M_n))) & \equiv U_l(\xi(T_{\alpha l}(M_1), \ldots, T_{\alpha l}(M_n))) \\ & \equiv \xi\beta.(U_{\beta l}(T_{\alpha l}(M_1)), \ldots, U_{\beta l}(T_{\alpha l}(M_n))) \end{array}$$

where $\beta \notin l$. We have two further cases to consider:

1. There are no occurrences of t-metavariables in M. Now if $\beta \equiv \alpha$ we conclude by using the induction hypothesis so let us assume then that $\beta \not\equiv \alpha$.

$$\begin{array}{lll} & \xi\beta.(U_{\beta l}(T_{\alpha l}(M_1)), \ldots, U_{\beta l}(T_{\alpha l}(M_n))) & \\ =_v & \xi\beta'.(U_{\beta l}(T_{\alpha l}(M_1)) \ll\beta\leftarrow\beta'\gg, \ldots, U_{\beta l}(T_{\alpha l}(M_n)) \ll\beta\leftarrow\beta'\gg) & (\beta' \text{ not in } U_{\beta l}(T_{\alpha l}(M_i))) \\ \equiv & \xi\beta'.(U_{\beta' l}(T_{\alpha l}(M_1)), \ldots, U_{\beta' l}(T_{\alpha l}(M_n))) & (\text{L.23}) \\ \equiv & \xi\beta'.(U_{\alpha l}(T_{\alpha l}(M_1)) \ll\alpha\leftarrow\beta'\gg, \ldots, U_{\alpha l}(T_{\alpha l}(M_n)) \ll\alpha\leftarrow\beta'\gg) & (\text{L.23}) \\ =_v & \xi\alpha.(U_{\alpha l}(T_{\alpha l}(M_1)), \ldots, U_{\alpha l}(T_{\alpha l}(M_n))) & \\ =_v & \xi\alpha.(M_1, \ldots, M_n) & (\text{i.h.}) \end{array}$$

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 $\beta \equiv \alpha$. 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 shall arrive at $U_{l''\beta l}(X_{l'\alpha l})$, which is defined only for $l''\beta l \equiv l'\alpha l$. Therefore, $\beta \equiv \alpha$ and we use the induction hypothesis.

– $M \equiv M_1[\alpha \leftarrow M_2]$. We proceed as above.

Corollary 3. *Let $M \in \mathcal{PMT}$ such that $\mathcal{WF}(M)$. Then $U(T(M)) =_v M$.*

Lemma 25. *Let $t \in \mathcal{T}$ such that $FV(t) \subseteq S \cup l$ and l be any label. Then $U_l^S(T_l(t)) \equiv_\alpha t$.*

Proof. By induction on the structure of t.

– $t \equiv x$. Then there are two cases to consider:
  • $x \in l$. Then $U_l^S(T_l(x)) \equiv U_l^S(\mathtt{pos}(x, l)) \equiv x$.

- $x \notin l$. Then $U_l^S(T_l(x)) \equiv U_l^S(\mathcal{O}(x) + |l|)$. By hypothesis $x \in S \cup l$ so that $x \in S$ and then $U_l^S(\mathcal{O}(x) + |l|) \equiv x_{\mathcal{O}(x)} \equiv x$

– $t \equiv f(t_1, \ldots, t_n)$. Then $U_l^S(T_l(t)) \equiv f(U_l^S(T_l(t_1)), \ldots, U_l^S(T_l(t_n))) \equiv_\alpha f(t_1, \ldots, t_n)$. The last step holds by induction hypothesis.

– $t \equiv \xi x.(t_1, \ldots, t_n)$. Then

$$
\begin{array}{lll}
 & U_l^S(T_l(\xi x.(t_1, \ldots, t_n))) & \\
\equiv & U_l^S(\xi(T_{xl}(t_1), \ldots, T_{xl}(t_n))) & \\
\equiv & \xi z.(U_{zl}^S(T_{xl}(t_1)), \ldots, U_{zl}^S(T_{xl}(t_n))) & (z \notin S \cup l) \\
\equiv_\alpha & \xi z'.(U_{zl}^S(T_{xl}(t_1))\{z \leftarrow z'\}, \ldots, U_{zl}^S(T_{xl}(t_n))\{z \leftarrow z'\}) & (z' \text{ not in } U_{z'l}^S(T_{xl}(t_i))) \\
\equiv & \xi z'.(U_{z'l}^S(T_{xl}(t_1)), \ldots, U_{z'l}^S(T_{xl}(t_n))) & (\text{L.13}) \\
\equiv & \xi z'.(U_{xl}^S(T_{xl}(t_1))\{x \leftarrow z'\}, \ldots, U_{xl}^S(T_{xl}(t_n))\{x \leftarrow z'\}) & (\text{L.13}) \\
\equiv_\alpha & \xi x.(U_{xl}^S(T_{xl}(t_1)), \ldots, U_{xl}^S(T_{xl}(t_n))) & (x \notin l) \\
\equiv_\alpha & \xi x.(t_1, \ldots, t_n) & (\text{i.h.})
\end{array}
$$

Corollary 4. *Let $t \in \mathcal{T}$. Then $U(T(t)) \equiv_\alpha t$.*

Lemma 26. *Let $A \in \mathcal{PMT}_{db}$ and l be any simple label. If $\mathcal{WF}_l(A)$ then $T_l(U_l(A)) \equiv A$.*

Proof. By induction on A.

– $A \equiv \mathtt{S}^j(1)$. Then $T_l(U_l(A)) \equiv T_l(\mathtt{at}(l, j+1)) \equiv \mathtt{pos}(\mathtt{at}(l, j+1), l)$. Since l is simple then $\mathtt{pos}(\mathtt{at}(l, j+1), l) \equiv \mathtt{S}^j(1)$ and we are done.
– $A \equiv \mathtt{S}^j(\widehat{\alpha})$. Then $j = |l|$ and $T_l(U_l(A)) \equiv T_l(\widehat{\alpha}) \equiv \mathtt{S}^j(\widehat{\alpha})$.
– $A \equiv X_k$. Then $l = k$ and $T_l(U_l(A)) \equiv T_l(X) \equiv X_k$.
– $A \equiv f(A_1, \ldots, A_n)$. Then

$$
\begin{array}{ll}
 & T_l(U_l(A)) \\
\equiv & T_l(f(U_l(A_1), \ldots, U_l(A_n))) \\
\equiv_{i.h.} & f(T_l(U_l(A_1)), \ldots, T_l(U_l(A_1))) \\
\equiv & f(A_1, \ldots, A_1)
\end{array}
$$

– $A \equiv \xi(A_1, \ldots, A_n)$. Then

$$
\begin{array}{ll}
 & T_l(U_l(A)) \\
\equiv & T_l(\xi\alpha.(U_{\alpha l}(A_1), \ldots, U_{\alpha l}(A_n))) \\
\equiv & \xi(T_{\alpha l}(U_{\alpha l}(A_1)), \ldots, T_{\alpha l}(U_{\alpha l}(A_n))) \\
\equiv_{i.h.} & \xi(A_1, \ldots, A_n)
\end{array}
$$

We remark that $\alpha \notin l$ by Definition 33 so the induction hypothesis can be applied.
– $A \equiv A_1[\![A_2]\!]$. Then

$$
\begin{array}{ll}
T_l(U_l(A)) & \\
\equiv & T_l(U_{\alpha l}(A_1)[\alpha \leftarrow U_l(A_2)]) \\
\equiv & T_{\alpha l}(U_{\alpha l}(A_1))[\![T_l(U_l(A_2))]\!] \\
\equiv_{i.h.} & A_1[\![A_2]\!]
\end{array}
$$

We remark that $\alpha \notin l$ by Definition 33 so the induction hypothesis can be applied.

Corollary 5. *Let $A \in \mathcal{PMT}_{db}$. If $\mathcal{WF}(A)$ then $T(U(A)) \equiv A$.*

Lemma 27. *Let a be a de Bruijn term and l be any* simple *label such that* $\mathtt{Names}(FV(a) \setminus\!\!\setminus |l|) \subseteq S$ *and $l \cap S = \emptyset$. Then $T_l(U_l^S(a)) \equiv a$.*

Proof. By induction on the structure of a.

– $a \equiv n$. Then there are two cases to consider:
  - $n \leq |l|$. Then $T_l(U_l^S(a)) \equiv T_l(\mathtt{at}(l,n)) \equiv \mathtt{pos}(\mathtt{at}(l,n),l)$. Since l is simple, then $\mathtt{pos}(\mathtt{at}(l,n),l) \equiv n$.
  - $n > |l|$ and $x_{n-|l|} \in S$. $T_l(U_l^S(a)) \equiv T_l(x_{n-|l|})$. Since l does not contain variables in S, then $T_l(x_{n-|l|}) \equiv n - |l| + |l| \equiv n$.

– $a \equiv f(a_1, \ldots, a_n)$. Then $T_l(U_l^S(a)) \equiv f(T_l(U_l^S(a_1)), \ldots, T_l(U_l^S(a_n))) \equiv_{i.h} f(a_1, \ldots, a_n)$.

– $a \equiv \xi(a_1, \ldots, a_n)$. Then $T_l(U_l^S(a)) \equiv \xi(T_{xl}(U_{xl}^S(a_1)), \ldots, T_{xl}(U_{xl}^S(a_1))) \equiv_{i.h} \xi(a_1, \ldots, a_n)$. We remark that $x \notin l \cup S$ by Definition 32 so that xl is simple and it does not contain variables of S so that we can apply the induction hypothesis.

Corollary 6. *Let $a \in \mathcal{T}_{db}$. Then $T(U(a)) \equiv a$.*

The notion of v-equivalence has been introduced in order to allow the renaming of some bound metavariables of a rewrite rule, which are built with metaterms, but keeping the same semantics. This is stated in the following lemma.

Lemma 28. *Let (G,D) and (G',D') be SERS rewrite rules such that $G =_v G'$ and $D =_v D'$. Then $\longrightarrow_{(G,D)} \; = \; \longrightarrow_{(G',D')}$.*

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

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. We define the valuation $\theta' = (\theta'_t, \theta'_v)$ as follows:

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

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. $s \equiv_\alpha C[\theta G] \equiv_\alpha C[\theta' G']$ and $t \equiv_\alpha C[\theta D] \equiv_\alpha C[\theta' D']$.

Hence $s \longrightarrow_{(G',D')} t$.

Corollary 7. *Let (G,D) be an SERS rewrite rule. Then the reduction relations generated by (G,D) and $\longrightarrow_{U(T(G,D))}$ are the same.*

Proof. Use Corollary 3, Lemma 28 and the fact the translations preserve well-formedness.

We are now able to show the main results of this section:

Theorem 1. *If $\mathcal{R}$ is a confluent SERS then $T(\mathcal{R})$ is a confluent $SERS_{DB}$.*

Proof. Suppose $a \longrightarrow_{T(\mathcal{R})} b$ and $a \longrightarrow_{T(\mathcal{R})} c$ for some de Bruijn terms a, b, c. Applying the translation mapping $U(.)$ and using Proposition 2 we may obtain the diagram (b) of Figure 5. The reduction denoted by the dotted lines are obtained by Corollary 7 and the confluence of $\mathcal{R}$.

Now applying the translation mapping $T(.)$ and using Proposition 1 we obtain the diagram (c) of Figure 5. Finally, Corollary 6 and Corollary 5 yield the desired diagram illustrated as diagram (a) in Figure 5.

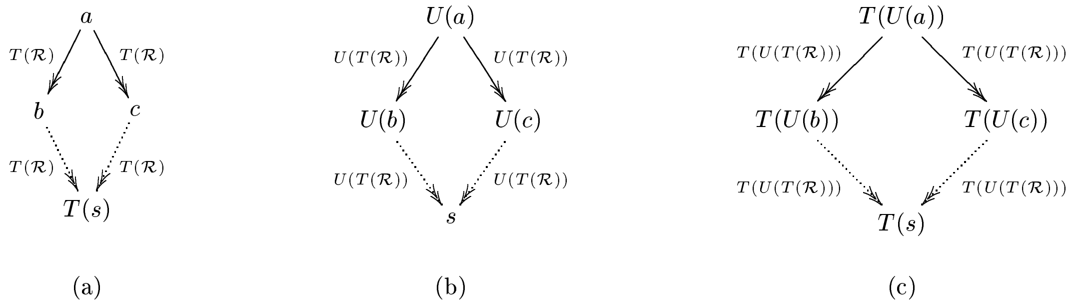


Fig. 5. Diagrams for preservation of confluence

Theorem 2. *If $\mathcal{R}$ is a confluent $SERS_{DB}$ then $U(\mathcal{R})$ is a confluent SERS.*

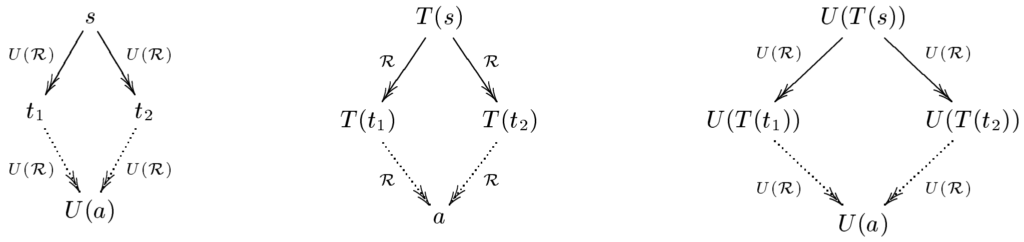
Proof. Suppose $s \longrightarrow_{U(\mathcal{R})} t_1$ and $s \longrightarrow_{U(\mathcal{R})} t_2$ for some terms s, t_1, t_2. Applying the translation mapping $T(.)$ and using Proposition 1 we may obtain the diagram (b) of Figure 6. The reduction denoted by the dotted lines are obtained by the confluence of $\mathcal{R}$. Note also that Corollary 5 has been used.

Now applying the translation mapping $U(.)$ and using Proposition 2 we obtain the diagram (c) of Figure 6. Finally, Corollary 4 and the definition of reduction Definition 15 yield the desired diagram illustrated as diagram (a) in Figure 6.

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 semantically 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 rewrite systems. This work fills the gap between classical presentations of higher-order rewriting with names existing in the literature and first-order presentations of higher-order rewriting such as [17]. Moreover, it explicitly suggests that *XRS* are not sufficient to express *an arbitrary* higher-order rewrite system.

Further ongoing work uses the formalism presented here to propose a tool for implementing higher-order rewrite systems via first-order ones. This tool would incorporate not only de Bruijn notation but also explicit substitutions in a very general form.



(a) (b) (c)

Fig. 6. Diagrams for preservation of confluence

References

1. Roel Bloo. *Preservation of Termination for Explicit Substitution.* PhD thesis, Eindhoven University, 1997.
2. Roel Bloo and Kristoffer Rose. Combinatory reduction systems with explicit substitution that preserve strong normalisation. In Harald Ganzinger, editor, *Proc. of the Seventh International Conference on Rewriting Techniques and Applications (RTA)*, number 1103 in Lecture Notes in Computer Science. Springer Verlag, 1996.
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. Eduardo Bonelli, Delia Kesner, and Alejandro Ríos. A de Bruijn notation for higher-order rewriting, 2000. Available as `ftp://ftp.lri.fr/LRI/articles/kesner/dBhor.ps.gz`.
5. Pierre-Louis Curien. *Categorical combinators, sequential algorithms and functional programming.* Progress in Theoretical Computer Science. Birkhaüser, 1993. Second edition.
6. Pierre-Louis Curien, Thérèse Hardin, and Jean-Jacques Lévy. Confluence properties of weak and strong calculi of explicit substitutions. *Journal of the ACM*, 43(2):362–397, march 1996.
7. N. de Bruijn. Lambda-calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the church-rosser theorem. *Indag. Mat.*, 5(35):381–392, 1972.
8. N. de Bruijn. A namefree lambda calculus with facilities for internal definition of expressions and segments. Technical Report 78-WSK-03, Eindhoven University of Technology, 1978.
9. Gilles Dowek. *La part du calcul.* PhD thesis, Univesité de Paris VII, 1999. Thèse d'Habilitation à diriger des recherches.
10. Gilles Dowek, Thérèse Hardin, and Claude Kirchner. Higher-order unification via explicit substitutions. In Dexter Kozen, editor, *Proceedings of Tenth Annual IEEE Symposium on Logic in Computer Science (LICS)*, San Diego, USA, 1995.
11. Roger Hindley and Jonathan P. Seldin. *Introduction to Combinators and λ-calculus.* London Mathematical Society, 1980.
12. H.P.Barendregt. *The Lambda Calculus: Its Syntax and Semantics.* Studies in Logic and the Foundations of Mathematics. North-Holland, Amsterdam, 1984. Revised edition.
13. Fairouz Kamareddine and Alejandro Ríos. Bridging de bruijn indices and variable names in explicit substitutions calculi. *Logic Journal of the Interest Group of Pure and Applied Logic (IGPL)*, 6(6):843–874, 1998.
14. Zurab Khasidashvili. Expression reduction systems. In *Proceedings of I. Vekua Institute of Applied Mathematics*, volume 36, Tbilisi, 1990.
15. Jan Willem Klop. *Combinatory Reduction Systems*, volume 127 of *Mathematical Centre Tracts.* CWI, Amsterdam, 1980. PhD Thesis.
16. Tobias Nipkow. Higher order critical pairs. In *Proceedings of the Sixth Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 342–349, Amsterdam, The Netherlands, 1991.

17. Bruno Pagano. *Des Calculs de Susbtitution Explicite et leur application à la compilation des langages fonctionnels*. PhD thesis, Université Paris VI, 1998.
18. Randy Pollack. Closure under alpha-conversion. In Henk Barendregt and Tobias Nipkow, editors, *Types for Proofs and Programs (TYPES)*, number 806 in Lecture Notes in Computer Science, Nijmegen, The Netherlands, 1993.
19. Jakob Rehof and Morten Heine Sørensen. The λ_Δ calculus. In M. Hagiya and J. Mitchell, editors, *Theoretical Aspects of Computer Software (TACS)*, number 789 in Lecture Notes in Computer Science, pages 516–542. Springer Verlag, 1994.
20. Kristoffer Rose. Explicit cyclic substitutions. In Michael Rusinowitch and Jean-Luc Rémy, editors, *Proc. of the Third International Workshop on Conditional Term Rewriting Systems (CTRS)*, number 656 in LNCS, pages 36–50, 1992.
21. Vincent van Oostrom and Femke van Raamsdonk. Weak orthogonality implies confluence: the higher-order case. In *Proc. of the third International Symposium on Logical Foundations of Computer Science (LFCS)*, number 813 in Lecture Notes in Computer Science, pages 379–392, St Petersburg, 1994. Springer Verlag.
22. Femke van Raamsdonk. *Confluence and Normalization for higher-Order Rewriting*. PhD thesis, Amsterdam University, The Netherlands, 1996.
23. David Wolfram. *The Clausal Theory of Types*, volume 21 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1993.