Relating Higher-order and First-order Rewriting

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

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

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

Abstract

We define a formal encoding from higher-order rewriting into first-order rewriting modulo an equational theory $\mathcal{E}$. In particular, we obtain a characterization of the class of higher-order rewriting systems which can be encoded by first-order rewriting modulo an empty equational theory (that is, $\mathcal{E} = \emptyset$). This class includes of course the λ-calculus. Our technique does not rely on the use of a particular substitution calculus but on an axiomatic framework of explicit substitutions capturing the notion of substitution in an abstract way. The axiomatic framework specifies the properties to be verified by a substitution calculus used in the translation. Thus, our encoding can be viewed as a parametric translation from higher-order rewriting into first-order rewriting, in which the substitution calculus is the parameter of the translation.

Keywords: Higher-order rewriting, first-order rewriting, explicit substitutions.

1 Introduction

Higher-order substitution is the core operation of languages based on higher-order rewrite systems. This operation, which is used to perform the substitution of variables by terms, cannot be expressed by simple replacement (also known as grafting) of variables as is done in first-order theories since it takes place in the context of languages having variable bindings. Issues such as variable capture and renaming on the one hand, and the complexity of higher-order substitution itself on the other, complicate both the metatheory and the implementation of higher-order rewrite systems. One approach to minimize this burden is given by calculi with *explicit substitutions*. Higher-order substitution is rendered as an object-level operation and thus included in the object theory itself. This allows higher-order systems/formalisms to be expressed in first-order systems/formalisms.

Usually, the process of making the operation of higher-order substitution explicit is accompanied by selecting an appropriate notation for terms, such as for example de Bruijn indices, that tames the aforementioned issues of variable capture and renaming. A well-known example of the combined benefits of de Bruijn indices notation and explicit substitutions is the formulation of different first-order calculi for the λ-calculus [1, 4, 30, 35, 16, 40], which is the paradigmatic example of a higher-order (term) rewriting system. Other examples are the translations of higher-order unification to first-order unification modulo [20], higher-order logic to first-order logic modulo [21], higher-order

Vol. 15 No. 6,
doi:10.1093/logcom/exi050

theorem proving to first-order theorem proving modulo [18], etc.

In a previous paper [11] we introduce a HOR (higher-order rewriting) formalism based on de Bruijn indices (called $SERS_{dB}$) which does away with α-conversion and establishes precise links between the $SERS$ formalism [11] and $SERS_{dB}$. However, substitution remains in both formalisms as a metalevel operation. This becomes a concrete problem in real implementations where substitutions must be denoted by symbols and constructors of the language, and the computational behaviour of substitutions must be specified by reduction rules belonging to the operational rules of the language itself. Thus, in the present article we encode all $SERS_{dB}$ as first-order rewriting systems with the aid of explicit substitutions.

The case of the λ-calculus is interesting but at the same time not fully representative of the problems we are faced with when encoding a higher-order system into a first-order setting. For this particular case it is enough to take care of α-conversion and promote metalevel substitution to the object-level in order to obtain a first-order rewrite system. Indeed, replacing the usual variable names by de Bruijn indices and introducing explicit substitutions suffices to yield a first-order rewrite system. However, this is not always the case for an arbitrary higher-order rewrite system. In other words, eliminating α-conversion and introducing explicit substitutions is not enough to yield an equivalent full first-order system (full in the sense of first-order rewriting modulo an empty equational theory). The reason is that in higher-order rewriting the left-hand side of a rewrite rule is a higher-order pattern. So we must somehow *also* encode higher-order pattern matching when encoding in the first-order framework. The fact that introducing de Bruijn indices plus explicit substitutions suffices for the λ-calculus is saying that for this particular rewrite system higher-order matching is doing nothing more than what first-order matching could do. Consider the η_{dB}-rewrite rule:

$$\lambda(app(X_\alpha, \underline{1})) \rightarrow_{\eta_{dB}} X_\epsilon.$$

One may verify that the term $\lambda(app(\underline{3}, \underline{1}))$ rewrites to $\underline{2}$. In a first-order setting with explicit substitution, we have the alternative formulation:

$$\lambda(app(X[\uparrow], \underline{1})) \rightarrow X.$$

However, in order for the term $X[\uparrow]$ to match the subterm $\underline{3}$ we need $\mathcal{E}$-matching, that is, matching modulo an equational theory $\mathcal{E}$. For an appropriate substitution calculus $\mathcal{E}$ we would need to solve the equation $\underline{3} \stackrel{?}{=}_{\mathcal{E}} X[\uparrow]$. That is to say, syntactic matching no longer suffices to implement the η_{dB} rule in a first-order setting since 'occurs check' is a feature of higher-order pattern matching which first-order matching cannot cope with. This may be seen as the reason why the η_{dB}-rule has received so much attention [45, 24, 14, 29].

Another, perhaps less evident, example is given by the commutation rule C_{dB}:

$$imply(\exists \forall X_{\beta\alpha}, \forall \exists X_{\alpha\beta}) \rightarrow_{C_{dB}} true.$$

The naïve translation to first-order, namely $imply(\exists \forall X, \forall \exists X) \rightarrow true$, is evidently not correct, so that we take its encoding in the de Bruijn higher-order formalism $SERS_{dB}$ and then translate it to first order via the conversion presented in this paper obtaining C_{fo}:

$$imply(\exists \forall X, \forall \exists X[\underline{2} \cdot \underline{1} \cdot \uparrow^2]) \rightarrow true.$$

Now, the rule C_{fo} has exactly the same intended meaning as the original higher-order rule C: in order for a term to be an instance of this rule, the term a instantiated for the leftmost X must be the one instantiated for $X[\underline{2} \cdot \underline{1} \cdot \uparrow^2]$, say a', except that all 1-level and 2-level indices in a shall be

interchanged (via the explicit substitution $[\underline{2} \cdot \underline{1} \cdot \uparrow^2]$) in order to obtain a'. Of course, the following rewrite rule C'_{fo} also does the job:

$$imply(\exists\forall X[\underline{2} \cdot \underline{1} \cdot \uparrow^2], \forall\exists X) \rightarrow true.$$

However, note that both C_{fo} and C'_{fo} induce the same rewrite relation on terms.

The goal of this paper is to provide a conversion algorithm for encoding higher-order rewriting systems into first-order rewriting modulo an equational theory $\mathcal{E}$. A distinctive feature of our algorithm is that we do not attach to the encoding any particular substitution calculus. Instead, we choose to work with an abstract formulation of substitution calculi, called *Basic Substitution Calculi*, as done in [29, 30] to deal with confluence proofs of λ-calculi with explicit substitutions. This macro-based presentation of calculi of explicit substitutions allows us the freedom of choosing from a wide range of calculi of explicit substitution, such as σ [1], $\sigma_{\Uparrow}$ [25], υ [4], f [29], d [29], s [35], λ_ϕ [40], when converting a higher-order rewrite system to first order.

The conversion procedure that we propose in this work takes a $SERS_{dB}$ $\mathcal{R}$ and produces a first-order rewrite system $fo(\mathcal{R})_{\mathcal{W}}$, where $\mathcal{W}$ is some Basic Substitution Calculus (such as for example σ). It is the conversion procedure's responsibility to compute index adjustments in order to correctly encode higher-order pattern matching in the first-order setting. The rewrite rules produced may or may not have occurrences of the explicit substitution operator on the LHSs. In the case that they do, as in the η_{dB} example, we need matching modulo the induced equational theory of the basic substitution calculus $\mathcal{W}$. Otherwise, syntactic matching suffices and thus the $SERS_{dB}$ $\mathcal{R}$, called in this case an *essentially first-order* higher-order rewrite system, can be translated to a full first-order rewrite system, where equational reasoning is not needed at all. This is for example the case of the λ-calculus.

The translation from higher-order rewriting systems to first-order rewriting modulo an equational theory $\mathcal{E}$ is interesting from a theoretical point of view because the expressive power of higher and first-order formalisms may be compared. However, another more practical issue arises, namely that of the possibility of *transferring results* developed in the first-order framework to the higher-order one. This is done for example in [12] for the case of the standardization theorem. Techniques concerning confluence, termination, completion, evaluation strategies, etc. should be looked at. Moreover, this is interesting for two further reasons: On the one hand it is still not fully clear how to transfer techniques such as dependency pairs [2], semantic labelling [52] or completion [5] to the higher-order framework, and on the other hand, termination techniques such as RPO for higher-order systems [28] turn out to be much more complicated than their respective first-order versions [15, 32]. We shall argue that the essentially first-order higher-order systems are better suited for the above mentioned transfer of properties.

This paper begins by recalling in Section 2 the de Bruijn indices based higher-order rewrite formalism $SERS_{dB}$ (Simplified Expression Reduction Systems with Indices). We introduce in Section 3 the first-order rewriting framework with explicit substitutions *ExERS* (Explicit ERS) which constitutes the destination formalism of our conversion procedure. This requires defining Basic Substitution Calculi, adapted to the present setting from [29, 30]. Section 4 introduces the conversion procedure and illustrates its use with some examples. This is followed by a study of the properties of this procedure: independence of pivot selection (a technicality concerning the conversion procedure), the simulation proposition and the projection proposition. The simulation proposition states that $fo(\mathcal{R})_{\mathcal{W}}$, which is the first-order rewrite system obtained by the conversion procedure, is able to simulate $\mathcal{R}$-rewriting. Conversely, the projection proposition states that if a rewrites to b in the system $fo(\mathcal{R})_{\mathcal{W}}$, then $\mathcal{W}(a) \twoheadrightarrow_{\mathcal{R}} \mathcal{W}(b)$, where $\mathcal{W}(a)$ denotes the substitution normal form of a ob-

tained by taking the Basic Substitution Calculus $\mathcal{W}$ as a rewriting system. More precisely, we shall see that one $fo(\mathcal{R})_{\mathcal{W}}$-rewrite step may be encoded as one *parallel* $\mathcal{R}$-rewrite step. We conclude by presenting the definition of the essentially first-order higher-order rewriting systems mentioned previously. This paper is an extended version of the extended abstract published as [10].

Related work

Other approaches to first-order expressions of higher-order formalisms use Nominal Logic. More precisely, the idea is to consider a first-order many-sorted logic with equality containing primitives for renaming via name-swapping for freshness of names and for name-binding. The logic makes use of the Fraenkel–Mostowski (FM) permutation model of set theory. Some of the more representative works along this line can be found in [23, 44, 48].

As regards existing higher-order rewrite formalisms based on de Bruijn notation and/or explicit substitutions to the best of the authors' knowledge there are three: *Explicit* CRS [8], *Explicit Reduction Systems* (XRS) [43], and the *Calculus of Indexed Names and Named Indices* ($CINNI$) [47]. In [8] explicit substitutions à la λx [46, 6] are added to the CRS formalism as a first step towards using higher-order rewriting with explicit substitutions for modelling 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 notation and explicit substitutions. The formalism XRS is presented as a generalization of the first-order $\lambda\sigma_{\Uparrow}$-calculus [25] to higher-order rewriting and *not* as a first-order formulation of higher-order rewriting. 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 in the ERS formalism, such as for example the η or the C-rule, may be written as an XRS. In the case of $CINNI$ a similar situation arises, no relation to established HORS in the literature is presented.

2 The higher-order framework

This section briefly recalls from [9, 13, 11] the de Bruijn indices based higher-order rewrite formalism $SERS_{dB}$ (Simplified Expression Reduction Systems with Indices).

2.1 Metaterms and terms

DEFINITION 2.1 (Labels)
A *label* is a finite sequence of symbols of an alphabet. A *simple* label is a label without repeated symbols. We use $k, l, l_i, \ldots$ to denote arbitrary labels and ϵ for the empty label. If s is a symbol and l is a label then the notation $s \in l$ means that the symbol s *appears in* the label l, and also, we use sl to denote the new label whose head is s and whose tail is l. Other notations are $|l|$ for the *length* of l (number of symbols in l) and $\mathtt{at}(l, n)$ for the nth element of l assuming $n \leq |l|$. Also, if s occurs (at least once) in l then $\mathtt{pos}(s, l)$ denotes the position of the *first occurrence* of s in l. If θ is a function defined on the alphabet of a label $l = s_1 \ldots s_n$, then $\theta(l)$ denotes the label $\theta(s_1) \ldots \theta(s_n)$. We may use a label as a set (e.g. if S is a set then $S \cap l$ denotes the intersection of S with the underlying set determined by l) if no confusion arises.

DEFINITION 2.2 (Signature)
A $SERS_{dB}$ signature Σ consists of the following denumerable and disjoint sets.

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

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

DEFINITION 2.3 (Pre-metaterms)
The set of *pre-metaterms* over Σ, denoted PMT_{dB}, is defined by the following two-sorted grammar:

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

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

The notation $\underline{I}$ in Definition 2.3 is used to denote some *representation* of the metaindex I. Thus, we do not necessarily use natural numbers to represent variables: for example, in $\lambda\sigma$, the representation of 2 is the term $1[\uparrow]$, which is considered a pure term.

We use $A, B, A_i, \ldots$ to denote pre-metaterms and the convention that $\mathtt{S}^0(1) = 1$ and $\mathtt{S}^{j+1}(n) = \mathtt{S}(\mathtt{S}^j(n))$. As usually done for indices, we abbreviate $\mathtt{S}^{j-1}(1)$ as j. The set of *free metavariables* of a pre-metaterm A, written $\mathrm{FMVAR}(A)$, is defined as expected. For example, $\mathrm{FMVAR}(f(\lambda X_\alpha, Y_\epsilon)) = \{X_\alpha, Y_\epsilon\}$. The set of *names* of free metavariables of A is the set $\mathrm{FMVAR}(A)$ where each X_l is replaced simply by X. We also write, by abuse of notation, $\mathrm{FMVAR}(A)$ to denote such a set of names. For example, we might write $\mathrm{FMVAR}(f(\lambda X_\alpha, Y_\epsilon)) = \{X, Y\}$ whenever there is no risk of confusion.

Next we single out a subset of *well-formed* pre-metaterms which we call *metaterms*. These are pre-metaterms in which the labels of metavariables are correct with respect to the context in which they appear and also they ensure that indices like j correspond to bound variables. Indeed, the pre-metaterms $\xi(X_{\alpha\beta})$ and $\xi(\xi(4))$ do not make sense for us, and hence are not considered well-formed.

DEFINITION 2.4 (Metaterms)
A pre-metaterm $A \in \mathsf{PMT}_{dB}$ over Σ is said to be a *metaterm* over Σ iff the predicate $\mathcal{WF}(A)$ holds, where $\mathcal{WF}(A)$ iff $\mathcal{WF}_\epsilon(A)$, and $\mathcal{WF}_l(A)$ is defined by induction on the structure of the pre-metaterm A for any label l as follows:

- $\mathcal{WF}_l(\underline{\mathtt{S}^j(1)})$ iff $j + 1 \leq |l|$
- $\mathcal{WF}_l(\underline{X_k})$ 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. Also, if $\mathcal{WF}_k(A)$, then any metavariable occurring in A must be of the form X_{lk} for some label l (moreover, lk is a simple label).

EXAMPLE 2.5
Pre-metaterms $\xi(X_\alpha, \lambda(Y_{\beta\alpha}, \underline{2}))$ and $g(\lambda(\xi c))$ are metaterms, whereas pre-metaterms $\lambda(\xi(X_{\alpha\alpha}))$ and $\xi(\underline{2})$ are not.

DEFINITION 2.6 (Linear metaterms)
A de Bruijn pre-metaterm (or metaterm) M is *linear* if it contains at most one occurrence of any X-based metavariable. Note that $f(\lambda(\xi X_{\alpha\beta}), \xi(\lambda X_{\beta\alpha}))$ is not linear since there are two occurrences of X-based metavariables, neither is $f(\lambda X_\alpha, \xi X_\alpha)$. However, $app(\lambda X_\alpha, Y_\epsilon)$ is linear.

DEFINITION 2.7 (de Bruijn terms and contexts)
The set of *de Bruijn terms* over Σ, denoted T_{dB}, and the set of de Bruijn contexts over Σ are defined by:

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

As for metaterms, the notation $\underline{n}$ in Definition 2.7 is used to denote some *representation* of the natural number n.

We use $a, b, a_i, b_i, \ldots$ for de Bruijn terms and $E, F, \ldots$ for de Bruijn contexts. The notion of the tree associated to a may be defined as for de Bruijn pre-metaterms. We may refer to the *binder path number* of a context, which is the number of binders between the $\Box$ and the root. Although terms are also pre-metaterms, that is $\mathsf{T}_{dB} \subset \mathsf{PMT}_{dB}$, note that some terms may not be metaterms, i.e. may not be well-formed pre-metaterms. Indeed, the valid term $\xi(\xi(\underline{4}))$ is not a metaterm, however, the index $\underline{4}$ may be seen as a constant in the pre-metaterm $\xi(\xi(\underline{4}))$.

DEFINITION 2.8 (Free de Bruijn indices)
We denote by $FI(a)$ the set of *free indices* of a de Bruijn term a, which is defined as follows:

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

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

DEFINITION 2.9 (de Bruijn substitution and de Bruijn updating function)
The result of substituting a term b for the index $n \geq 1$ in a term a is denoted $a\{\!\{n \leftarrow b\}\!\}$ and defined as:

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

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

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

2.2 $SERS_{dB}$-rewriting

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

DEFINITION 2.10 ($SERS_{dB}$)
A *de Bruijn rewrite rule* over Σ is a pair of de Bruijn metaterms (L, R) over Σ (also written $L \rightarrow R$) such that:

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

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

EXAMPLE 2.11
The λ_{dB}-calculus is defined by considering the signature containing the function symbols $\mathcal{F} = \{app\}$ and binder symbols $\mathcal{B} = \{\lambda\}$, together with the $SERS_{dB}$-rewrite rule:

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

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

DEFINITION 2.12 (de Bruijn valuation)
A *de Bruijn valuation* κ over Σ is a (partial) function from metavariables to terms. It extends uniquely to a function $\overline{\kappa}$ from the set of pre-metaterms A with $\text{FMVAR}(A) \subseteq Dom(\kappa)$, where $Dom(\kappa)$ denotes the domain of κ, to the set of terms as follows:

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

Note that in the above definition the substitution operator $\bullet\{\!\{\bullet \leftarrow \bullet\}\!\}$ refers to the usual substitution defined on terms with de Bruijn indices (Definition 2.9).

In the rewrite rule $\exists\forall X_{\alpha\beta} \rightarrow_{Comm'} \forall\exists X_{\beta\alpha}$, a valuation that assigns the de Bruijn index 1 to the metavariable $X_{\alpha\beta}$ and also to $X_{\beta\alpha}$ does not reflect the binder commutation that the labels of metavariables are expressing. Thus, when defining the rewrite relation on terms induced by a rewrite rule we must restrict our attention to the subset of all valuations that are *coherent* with the contextual information described by the labels of binder indicators in metavariables. Such valuations are dubbed *valid valuations*.

DEFINITION 2.13 (Valid de Bruijn valuation)

- Let $a \in \mathsf{T}_{dB}$, l be a label of binder indicators and $\{x_1, x_2, \ldots\}$ a countable infinite set of variable names. The *value function* $\mathsf{Value}(l, a)$ is defined as $\mathsf{Value}^0(l, a)$ where:

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

- A de Bruijn valuation κ over Σ is said to be *valid* if for every pair of metavariables X_l and $X_{l'}$ in $Dom(\kappa)$, $\mathsf{Value}(l, \kappa X_l) = \mathsf{Value}(l', \kappa X_{l'})$. Likewise, we say that a de Bruijn valuation κ is *valid for a rewrite rule* (L, R) if every metavariable in (L, R) is in $Dom(\kappa)$ and for every pair of metavariables X_l and $X_{l'}$ in (L, R), $\mathsf{Value}(l, \kappa X_l) = \mathsf{Value}(l', \kappa X_{l'})$.

DEFINITION 2.14 (Rewriting de Bruijn terms)
Let $\mathcal{R}$ be a set of de Bruijn rules over Σ and a, b de Bruijn terms over Σ. We say that a *$\mathcal{R}$-rewrites* or *$\mathcal{R}$-reduces to* b, written $a \to_{\mathcal{R}} b$, iff there is a de Bruijn rule $(L, R) \in \mathcal{R}$ and a de Bruijn valuation κ valid for (L, R) such that $a = E[\kappa L]$ and $b = E[\kappa R]$, where E is a de Bruijn context.

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

To finish this section, we give the notion of *orthogonal* systems in the framework of Simplified Expression Reduction Systems with Indices, which is the natural extension of that of orthogonal first-order rewrite systems.

DEFINITION 2.15
Let $(\Sigma, \mathcal{R})$ be a $SERS_{dB}$.

1. $\mathcal{R}$ is *non-overlapping* if for every rewrite rule $(L, R) \in \mathcal{R}$ the following holds:
   - If a reducible term $\kappa(L)$ contains an instance of L' for some *other* $(L', R') \in \mathcal{R}$, then this instance must be already contained in one of the $\kappa(X_l)$, where X_l is a metavariable of L.
   - Likewise if $\kappa(L)$ *properly* contains another instance of L.
2. $\mathcal{R}$ is *left-linear* if all L_i are linear (Definition 2.6).
3. $\mathcal{R}$ is *orthogonal* if it is non-overlapping and left-linear.

REMARK 2.16
This presentation of the $SERS_{dB}$ formalism differs slightly from that of [9, 13, 11]. In *op. cit.*, in addition to term-metavariables such as X_l, index-metavariables are included. Just like term-metavariables are instantiated with terms, index-metavariables are instantiated by de Bruijn indices. Moreover, these indices must be free in the rewrite rule where they appear. The reason for excluding them in this paper is that they do not enjoy good properties. For example, consider the $SERS_{dB}$ $\mathcal{S} = \{app(\lambda X_\alpha, Y_\epsilon) \to_{\beta_{dB}} X_\alpha[Y_\epsilon], f(\widehat{\alpha}) \to_f c\}$ where app, f, c are function symbols, λ is a binder symbol and $\widehat{\alpha}$ is an index-metavariable. Although $\mathcal{S}$ is orthogonal, it is not confluent. Indeed, $app(\lambda(f(\underline{1})), b)$ reduces to $f(b)$ by the β_{dB}-rule and to c by the f-rule followed by an application of the β_{dB}-rule.

3 The first-order framework

In this section we introduce the first-order formalism called Explicit Expression Reduction Systems (*ExERS*) used to translate higher-order rewriting systems based on de Bruijn indices into first-order ones. An *ExERS* is a first-order rewrite system consisting of:

- A set of *proper rewrite rules* governing the behaviour of the function and binder symbols in the signature.
- A set of substitution rewrite rules, called the *substitution calculus* governing the behaviour of the substitution symbols in the signature, and used for propagating and performing/eliminating substitutions.

3.1 Basic substitution calculi

Any set of rewrite rules does not qualify as a substitution calculus so we must indicate under which conditions this is the case. This is achieved by introducing a general macro-based presentation of what constitutes a substitution calculus. Any instance of this calculus, obtained by associating some fixed *substitution operators* to these macros, shall be considered a substitution calculus. This idea has been introduced by D. Kesner [29, 30] with the aim of providing a unique proof of confluence encompassing a series of de Bruijn indices based calculi of explicit substitutions. Here we shall benefit from it by reducing higher-order rewriting to a first-order setting where the substitution calculus may be *any* calculus of explicit substitutions fitting the macro-based presentation. Thus we are not forced to settle with some particular calculus of explicit substitutions.

DEFINITION 3.1 (Substitution declaration and signature)
A *substitution declaration* is a (possibly empty) word over the alphabet $\{\mathsf{T}, \mathsf{S}\}$. The symbol T is used to denote the sort **Term** and S to denote the sort **Substitution**. A *substitution signature* is a set Γ_s of *substitution symbols* equipped with an arity n and a substitution declaration of length n. We write $\sigma : w$ where $w \in \{\mathsf{T}, \mathsf{S}\}^n$ if the substitution symbol σ has arity n and substitution declaration w. We use ϵ to denote the empty word.

The substitution declaration declares the sorts of the arguments of substitution symbols.

DEFINITION 3.2 (*ExERS* term algebra)
An *ExERS* signature is a set $\Gamma = \Gamma_f \cup \Gamma_b \cup \Gamma_s$ where $\Gamma_f = \{f_1, f_2, \ldots, \}$ is a set of function symbols, $\Gamma_b = \{\lambda_1, \lambda_2, \ldots\}$ is a set of binder symbols, Γ_s a substitution signature such that Γ_f, Γ_b and Γ_s are pairwise disjoint. Both binder and function symbols come equipped with an arity (non-zero for binder symbols). Given a set of (term) variables $\mathcal{V} = \{X_1, X_2, \ldots\}$, the term algebra of an *ExERS* of signature Γ generated by $\mathcal{V}$, denoted $\mathcal{T}$, is

$$\begin{array}{lrcl}
\texttt{indices} & n & ::= & 1 \mid \mathsf{S}(n) \\
\texttt{terms (T)} & a & ::= & X \mid \underline{n} \mid a[s] \mid f(a_1, \ldots, a_n) \mid \xi(a_1, \ldots, a_n) \\
\texttt{substitutions (S)} & s & ::= & \sigma(d_1, \ldots, d_n)
\end{array}$$

where X ranges over $\mathcal{V}$, f over Γ_f, ξ over Γ_b, and σ over Γ_s. The notation $\underline{n}$ is used to denote some *representation* of the natural number n. The arguments of σ are assumed to respect the sorts prescribed in its substitution declaration (i.e. d_i is a term or substitution in compliance with its substitution declaration), and function and binder symbols are assumed to respect their arities too.

Letters $a, b, c, \ldots$ and $s, s_i, \ldots$ are used for terms and substitutions, respectively. Letters $o, o', \ldots$ are used for all objects of the term algebra without making distinction of sorts. The $\bullet[\bullet]$ operator is called the *substitution operator*. Binder symbols and substitution operators are considered as having binding power. We shall use $a[s]^n$ to abbreviate $a[s] \ldots [s]$ (n-times). Terms without occurrences of the substitution operator (resp. objects in $\mathcal{V}$) are called *pure* (resp. *ground*) terms. Similarly for contexts. A *context* is a ground term with one (and only one) occurrence of a distinguished

term variable called a 'hole' (and denoted $\Box$). Letters $E, E_i, \ldots$ are used for contexts. The notion of *binder path number* is defined for pure contexts exactly as in the case of de Bruijn contexts (Definition 2.7). Note that contexts have no variables (except $\Box$).

The formalism of *ExERS* that we are going to use in order to encode higher-order rewriting consists of two sets of rewrite rules, a set of proper rewrite rules, and a set of substitution rules. Let us define these two concepts formally.

DEFINITION 3.3 (Substitution macros)
Let Γ_s be a substitution signature. The following symbols not included in Γ_s are called *substitution macros*: $cons : (\mathsf{TS})$, $lift : (\mathsf{S})$, $id : (\epsilon)$ and $shift^j : (\epsilon)$ for $j \geq 1$. We shall abbreviate $shift^1$ by $shift$. Also, if $j \geq 0$ then $lift^j(s)$ stands for s if $j = 0$ and for $lift(lift^{j-1}(s))$ otherwise. Furthermore, if $j \geq 1$ then $cons(a_1, \ldots, a_j, s)$ stands for $cons(a_1, \ldots cons(a_j, s))$.

DEFINITION 3.4 (Term rewrite and equational systems)
Let Γ be an *ExERS* signature. An *equation* is a pair of terms $L \doteq R$ over Γ such that L and R have the same sort and a *term rewrite rule* is a pair of terms (L, R) over Γ, such that:

1. L and R have the same sort,
2. the head symbol of L is a function or a binder symbol, and
3. the set of variables of L includes those of R.

An *equational* (resp. *term rewrite*) *system* is a set of equations (resp. term rewrite rules).

As usual, we shall need some mechanism for instantiating rewrite rules.

DEFINITION 3.5 (First-order Valuation)
Let ρ be a (partial) function mapping variables in $\mathcal{V}$ to terms. We define a *first-order valuation* $\overline{\rho}$ as the unique extension of ρ over the set $\mathcal{T}$ such that:

$$\begin{array}{llll}
\overline{\rho}(\underline{n}) & \stackrel{\text{def}}{=} & \underline{n} & \overline{\rho}(f(a_1, \ldots, a_n)) \stackrel{\text{def}}{=} f(\overline{\rho}(a_1), \ldots, \overline{\rho}(a_n)) \\
\overline{\rho}(X) & \stackrel{\text{def}}{=} & \rho(X) & \overline{\rho}(\xi(a_1, \ldots, a_n)) \stackrel{\text{def}}{=} \xi(\overline{\rho}(a_1), \ldots, \overline{\rho}(a_n)) \\
\overline{\rho}(a[s]) & \stackrel{\text{def}}{=} & \overline{\rho}(a)[\overline{\rho}(s)] & \overline{\rho}(\sigma(d_1, \ldots, d_n)) \stackrel{\text{def}}{=} \sigma(\overline{\rho}(d_1), \ldots, \overline{\rho}(d_n))
\end{array}$$

We shall often abbreviate $\overline{\rho}$ as ρ. First-order valuations are required in order to define the rewrite relation induced by a rewrite system.

DEFINITION 3.6 (Rewriting and Equality)
Let o and o' be two ground terms of sort T or S. Given a rewrite system $\mathcal{R}$, we say that o *rewrites to* o' *in one step*, denoted $o \rightarrow_{\mathcal{R}} o'$, iff $o = E[\rho L]$ and $o' = E[\rho R]$ for some first-order valuation ρ, some context E and some rewrite rule (L, R) in $\mathcal{R}$. We shall use $\twoheadrightarrow_{\mathcal{R}}$ to denote the reflexive, transitive closure of the one-step rewrite relation.

Given an equational system $\mathcal{E}$, we consider the relation $=_{\mathcal{E}}$ defined as the least reflexive, symmetric and transitive relation closed under contexts and substitutions and containing all the axioms in $\mathcal{E}$. If o and o' are related by $=_{\mathcal{E}}$, then we write $o =_{\mathcal{E}} o'$ and we say that o *equals* o' *modulo* $\mathcal{E}$.

DEFINITION 3.7 (Substitution calculus)
A *substitution calculus over an ExERS signature* Γ consists of a set $\mathcal{W}$ of first-order term rewrite rules, and an interpretation of each substitution macro as some combination of substitution symbols from Γ_s of corresponding signature. Definition 3.8 requires certain properties for these interpretations to be considered meaningful.

We now give some examples of substitution calculi. We use the notation n^* for $\mathtt{S}(\ldots\mathtt{S}(0)\ldots)$ (the symbol $\mathtt{S}$ appearing n times), and we define $\diamond^n(t)$ by induction as follows:

$$\diamond^0(t) = \mathtt{id}, \diamond^1(t) = t, \diamond^{n+1}(t) = t \circ \diamond^n(t)$$

Calculus	Reference	Variable $\underline{n+1}$	Substitution Signature
σ	[1]	$1[\diamond^n(\uparrow)]$	$\mathtt{id} : (\epsilon), \cdot : (\mathsf{TS}), \uparrow\!: (\epsilon), \circ : (\mathsf{SS})$
$\sigma_{\Uparrow}$	[25]	$n+1$	$\mathtt{id} : (\epsilon), \cdot : (\mathsf{TS}), \Uparrow\!: (\mathsf{S}), \uparrow\!: (\epsilon), \circ : (\mathsf{SS})$
ϕ	[40]	$1[\uparrow^{n^*}]$	$\uparrow^{n^*}\!: (\epsilon), \cdot : (\mathsf{TS}), \circ : (\mathsf{SS})$

Calculus	$shift^1$	$lift^1(s)$	$cons(b,s)$
σ	$\uparrow$	$1 \cdot (s \circ \uparrow)$	$b \cdot s$
$\sigma_{\Uparrow}$	$\uparrow$	$\Uparrow(s)$	$b \cdot s$
ϕ	$\uparrow^{\mathtt{S}(0)}$	$1 \cdot (s \circ \uparrow^{\mathtt{S}(0)})$	$b \cdot s$

The next step is to add further requirements on substitution calculi in order for them to deserve that name. These conditions are assembled in the definition of a *Basic Substitution Calculus*.

DEFINITION 3.8 (Basic substitution calculus)
A substitution calculus $\mathcal{W}$ over Γ is said to be *basic* if the following conditions are satisfied:

1. $\mathcal{W}$ is complete (strongly normalizing and confluent) over the ground terms in $\mathcal{T}$. We use $\mathcal{W}(a)$ to indicate the unique $\mathcal{W}$-normal form of a.
2. $\mathcal{W}$-normal forms of ground terms are pure terms.
3. For each $f \in \Gamma_f$ and $\xi \in \Gamma_b$:

$$\begin{array}{lcl} \mathcal{W}(f(a_1,\ldots,a_n)) & = & f(\mathcal{W}(a_1),\ldots,\mathcal{W}(a_n)) \\ \mathcal{W}(\xi(a_1,\ldots,a_n)) & = & \xi(\mathcal{W}(a_1),\ldots,\mathcal{W}(a_n)) \\ \mathcal{W}(f(a_1,\ldots,a_n)[s]) & = & f(\mathcal{W}(a_1[s]),\ldots,\mathcal{W}(a_n[s])) \\ \mathcal{W}(\xi(a_1,\ldots,a_n)[s]) & = & \xi(\mathcal{W}(a_1[lift(s)]),\ldots,\mathcal{W}(a_n[lift(s)])). \end{array}$$

4. For every substitution s, $\underline{1}[lift(s)] =_{\mathcal{W}} \underline{1}$.
5. For every substitution s and every $m \geq 0$, $\underline{m+1}[lift(s)] =_{\mathcal{W}} \underline{m}[s][shift]$.
6. For every term a and substitution s we have $\underline{1}[cons(a,s)] =_{\mathcal{W}} a$.
7. For every term a, substitution s, $m \geq 0$ we have $\underline{m+1}[cons(a,s)] =_{\mathcal{W}} \underline{m}[s]$.
8. For every $m, j \geq 1$ we have $\underline{m}[shift^j] =_{\mathcal{W}} \underline{m+j}$.
9. For every ground term a we have $a[id] =_{\mathcal{W}} a$.

The first three conditions may be seen as primitive conditions that $\mathcal{W}$ should satisfy in order to be called a substitution calculus. The remaining conditions describe the expected behaviour of the substitution macros.

Examples of basic substitution calculi are σ, $\sigma_{\Uparrow}$ and ϕ, where the set of function and binder symbols are $\{app\}$ and $\{\lambda\}$, respectively. Reduction rules of these calculi appear in Figs 1, 2 and 3. We invite the reader to verify that conditions 3 to 9 in Definition 3 are trivial in these three cases.

The reader may have noted that the macro-based presentation of substitution calculi makes use of parallel substitutions (since $cons(\bullet,\bullet)$ has substitution declaration TS). Nevertheless, the results presented in this work may be achieved also via a macro-based presentation using a simpler set

$$
\begin{array}{lllll}
(App) & (a\ b)[s] & \longrightarrow & (a[s]\ b[s]) \\
(Lambda) & (\lambda a)[s] & \longrightarrow & \lambda(a[1 \cdot (s \circ \uparrow)]) \\
(Clos) & (a[s])[t] & \longrightarrow & a[s \circ t] \\
(VarId) & 1[id] & \longrightarrow & 1 \\
(VarCons) & 1[a \cdot s] & \longrightarrow & a \\
(IdL) & id \circ s & \longrightarrow & s \\
(ShiftId) & \uparrow \circ id & \longrightarrow & \uparrow \\
(ShiftCons) & \uparrow \circ (a \cdot s) & \longrightarrow & s \\
(Ass) & (s_1 \circ s_2) \circ s_3 & \longrightarrow & s_1 \circ (s_2 \circ s_3) \\
(Map) & (a \cdot s) \circ t & \longrightarrow & a[t] \cdot (s \circ t)
\end{array}
$$

FIGURE 1. Reduction rules for σ

$$
\begin{array}{lllll}
(App) & (a\ b)[s] & \longrightarrow & (a[s]\ b[s]) \\
(Lambda) & (\lambda a)[s] & \longrightarrow & \lambda(a[\Uparrow (s)]) \\
(Clos) & (a[s])[t] & \longrightarrow & a[s \circ t] \\
(VarShift1) & n[\uparrow] & \longrightarrow & n+1 \\
(VarShift2) & n[\uparrow \circ s] & \longrightarrow & n+1[s] \\
(FVar) & 1[a \cdot s] & \longrightarrow & a \\
(FVarLift1) & 1[\Uparrow (s)] & \longrightarrow & 1 \\
(FVarLift2) & 1[\Uparrow (s) \circ t] & \longrightarrow & 1[t] \\
(RVar) & n+1[a \cdot s] & \longrightarrow & n[s] \\
(RVarLift1) & n+1[\Uparrow (s)] & \longrightarrow & n[s \circ \uparrow] \\
(RVarLift2) & n+1[\Uparrow (s) \circ t] & \longrightarrow & n[s \circ (\uparrow \circ t)] \\
(Ass) & (s_1 \circ s_2) \circ s_3 & \longrightarrow & s_1 \circ (s_2 \circ s_3) \\
(Map) & (a \cdot s) \circ t & \longrightarrow & a[t] \cdot (s \circ t) \\
(Shift) & \uparrow \circ (a \cdot s) & \longrightarrow & s \\
(ShiftLift1) & \uparrow \circ \Uparrow (s) & \longrightarrow & s \circ \uparrow \\
(ShiftLift2) & \uparrow \circ (\Uparrow (s) \circ t) & \longrightarrow & s \circ (\uparrow \circ t) \\
(Lift1) & \Uparrow (s) \circ \Uparrow (t) & \longrightarrow & \Uparrow (s \circ t) \\
(Lift2) & \Uparrow (s) \circ (\Uparrow (t) \circ u) & \longrightarrow & \Uparrow (s \circ t) \circ u \\
(LiftEnv) & \Uparrow (s) \circ (a \cdot t) & \longrightarrow & a \cdot (s \circ t) \\
(IdL) & id \circ s & \longrightarrow & s \\
(IdR) & s \circ id & \longrightarrow & s \\
(LiftId) & \Uparrow (id) & \longrightarrow & id \\
(Id) & a[id] & \longrightarrow & a
\end{array}
$$

FIGURE 2. Reduction rules for $\sigma_{\Uparrow}$

of substitutions such as for example the one used in [30], where $scons(\bullet)$ (the 's' in $scons$ is for 'simple') has substitution declaration T and the macro $shift^j$ is only defined for $j = 1$. In particular, remark that the expression $scons(a)$ could be denoted as $cons(a, id)$. Conversely, an expression of

$$
\begin{array}{lllll}
(App) & (a\ b)[s] & \longrightarrow & (a[s]\ b[s]) \\
(Lambda) & (\lambda a)[s] & \longrightarrow & \lambda(a[1 \cdot s \circ \uparrow^{\mathsf{S}(0)}]) \\
(Clos) & (a[s])[t] & \longrightarrow & a[s \circ t] \\
(VarCons) & 1[a \cdot s] & \longrightarrow & a \\
(Id) & a[\uparrow^0] & \longrightarrow & a \\
(Map) & (a \cdot s) \circ t & \longrightarrow & a[t] \cdot (s \circ t) \\
(IdL) & \uparrow^0 \circ s & \longrightarrow & s \\
(ShiftCons) & \uparrow^{\mathsf{S}(n)} \circ (a \cdot s) & \longrightarrow & \uparrow^n \circ s \\
(ShiftShift) & \uparrow^{\mathsf{S}(n)} \circ \uparrow^m & \longrightarrow & \uparrow^n \circ \uparrow^{\mathsf{S}(m)} \\
(Shift1) & 1 \cdot \uparrow^{\mathsf{S}(0)} & \longrightarrow & \uparrow^0 \\
(Shift2) & 1[\uparrow^n] \cdot \uparrow^{\mathsf{S}(n)} & \longrightarrow & \uparrow^n
\end{array}
$$

FIGURE 3. Reduction rules for ϕ

the form $a[cons(b_1, \ldots, b_n, shift^j)]$ could be denoted by the expression

$$a[lift^n(shift)]^j[scons(b_1[shift]^{n-1})] \ldots [scons(b_n)].$$

DEFINITION 3.9 (*ExERS* and *FExERS*)
Let Γ be an *ExERS* signature, $\mathcal{W}$ a basic substitution calculus over Γ and $\mathcal{R}$ a set of term rewrite rules. If each rule of $\mathcal{R}$ has sort T then $\mathcal{R}_\mathcal{W} \stackrel{\text{def}}{=} (\Gamma, \mathcal{R}, \mathcal{W})$ is called an Explicit Expression Reduction System (*ExERS*). If, in addition, the LHS of each rule in $\mathcal{R}$ contains no occurrences of the substitution operator $\bullet[\bullet]$, then $\mathcal{R}_\mathcal{W}$ is called a Fully Explicit Expression Reduction System (*FExERS*).

Since rewriting in $SERS_{dB}$ only takes place on terms, and first-order term rewriting systems will be used to simulate higher-order rewriting, all the rules of a term rewrite system $\mathcal{R}$ are assumed to have sort T. However, rewrite rules of $\mathcal{W}$ may have any sort (i.e. T or S).

EXAMPLE 3.10
Consider the signature Γ where $\Gamma_f = \{app\}$ and $\Gamma_b = \{\lambda\}$ and Γ_s is any substitution signature. Let $\mathcal{W}$ be a basic substitution calculus over Γ and let $\mathcal{R}$ be the set containing the term rewrite rule $\{app(\lambda X, Y) \rightarrow_{\beta_{dB}} X[cons(Y, id)]\}$. Then we have that $\mathcal{R}_\mathcal{W}$ is an *FExERS*, and for $\mathcal{R}'$: $\mathcal{R} \cup \{\lambda(app(X[shift], \underline{1})) \rightarrow_{\eta_{dB}} X\}$, $\mathcal{R}'_\mathcal{W}$ is an *ExERS*.

Rewriting in an *ExERS* $\mathcal{R}_\mathcal{W}$ is first-order rewriting in $\mathcal{R}$ modulo $\mathcal{W}$-equality. In contrast, rewriting in a *FExERS* $\mathcal{R}_\mathcal{W}$ is just first-order rewriting in $\mathcal{R} \cup \mathcal{W}$.

DEFINITION 3.11 (*ExERS* and *FExERS*-rewriting)
Let $\mathcal{R}_\mathcal{W}$ be an *ExERS*, $\mathcal{R}'_\mathcal{W}$ a *FExERS* and o, o' ground terms of sort S or T. We say that o $\mathcal{R}_\mathcal{W}$-*reduces* or *rewrites to* o', written $o \rightarrow_{\mathcal{R}_\mathcal{W}} o'$, iff $o \rightarrow_{\mathcal{R}/\mathcal{W}} o'$ (i.e. there exist ground terms o_1 and o'_1 of the same sort as o, o' such that $o =_\mathcal{W} o_1 \rightarrow_\mathcal{R} o'_1 =_\mathcal{W} o'$); and o $\mathcal{R}'_\mathcal{W}$-*reduces* or *rewrites to* o' iff $o \rightarrow_{\mathcal{R}' \cup \mathcal{W}} o'$.

We apologize for the abuse of notation: $o \rightarrow_{\mathcal{R}/\mathcal{W}} o'$ could intuitively suggest that it is equivalence classes of terms that are rewritten, however, as defined above, this is not the case. Instead, it is terms that are rewritten.

EXAMPLE 3.12
Fix $\mathcal{W}$ to be the σ-calculus and consider the *FExERS* $\mathcal{R}_\sigma$ of Example 3.10. Then we have $\underline{1}[app(\lambda\underline{1}, c)\cdot id]{\to}_{\mathcal{R}_\sigma}\ \underline{1}[\underline{1}[c\cdot id]\cdot id]$. Also, $\lambda(app(\underline{3}, \underline{1})){\to}_{\mathcal{R}'_\sigma}\ \underline{2}$, where $\mathcal{R}'_\sigma$ is that of Example 3.10. This follows from observing that $\lambda(app(\underline{3}, \underline{1}))\ =_\sigma\ \lambda(app(\underline{2}[\uparrow], \underline{1})){\to}_{\eta_{dB}}\ \underline{2}$.

3.2 Properties of basic substitution calculi

This subsection takes a look at properties enjoyed by basic substitution calculi and introduces a condition called the *Scheme* [30]. Basic substitution calculi satisfying the scheme ease inductive reasoning when proving properties over them without compromising the genericity achieved by the macro-based presentation.

DEFINITION 3.13 (The Scheme)
We say that a basic substitution calculus $\mathcal{W}$ obeys the *Scheme* iff for every index m and every substitution symbol $\sigma \in \Gamma_s$ of arity q one of the following two conditions hold:

1. There exists a de Bruijn index n, positive numbers $i_1, \ldots, i_r$ $(r \geq 0)$ and substitutions $u_1, \ldots, u_k$ $(k \geq 0)$ such that
   - $1 \leq i_1, \ldots, i_r \leq q$ and all the i_js are distinct;
   - for all $o_1, \ldots, o_q$ we have: $\underline{m}[\sigma(o_1, \ldots, o_q)] =_{\mathcal{W}} \underline{n}[o_{i_1}] \ldots [o_{i_r}][u_1] \ldots [u_k]$.
2. There exists an index i $(1 \leq i \leq q)$ such that for all $o_1, \ldots, o_q$ we have: $\underline{m}[\sigma(o_1, \ldots, o_q)] =_{\mathcal{W}} o_i$.

We assume these equations to be well-typed: whenever the first case holds, then $o_{i_1}, \ldots, o_{i_r}$ are substitutions, whenever the second case holds, o_i is of sort T.

EXAMPLE 3.14
Examples of calculi satisfying the scheme are σ, $\sigma_{\Uparrow}$, υ, f and d [29, 30].

We now take a quick look at some properties of arbitrary basic substitution calculi. On a first reading the reader may wish to skim over this section and proceed to the main section of this paper, namely Section 4.

LEMMA 3.15 (Behaviour of substitutions in basic substitution calculi)
Let $\mathcal{W}$ be a basic substitution calculus and $m \geq 1$.

1. For all $n \geq 0$ and substitution s in S: $\underline{m}[\mathit{lift}^n(s)] =_{\mathcal{W}} \begin{cases} \underline{m-n}[s][\mathit{shift}]^n & \text{if } m > n \\ \underline{m} & \text{if } m \leq n. \end{cases}$
2. For all $n \geq m \geq 1$ and all terms $a_1, \ldots, a_n$: $\underline{m}[cons(a_1, \ldots, a_n, s)] =_{\mathcal{W}} a_m$.
3. For all pure terms a, b and $m \geq 1$: $a\{\!\{m \leftarrow b\}\!\} =_{\mathcal{W}} a[\mathit{lift}^{m-1}(cons(b, id))]$.

The first and third items of Lemma 3.15 are proved in [30], the second item follows from the definition of a basic substitution calculus. For the proof of the following lemma the reader is referred to [30].

LEMMA 3.16
Let $\mathcal{W}$ be a basic substitution calculus, a a pure term and s a term of sort S. For every $m \geq n \geq 0$ we have $a[\mathit{shift}]^n[\mathit{lift}^m(s)] =_{\mathcal{W}} a[\mathit{lift}^{m-n}(s)][\mathit{shift}]^n$.

LEMMA 3.17
Let $\mathcal{W}$ be a basic substitution calculus, a a pure term, and b a term of sort T. For every $n \geq 0$, $a[\mathit{lift}^n(\mathit{shift})][\mathit{lift}^n(cons(b, id))] =_{\mathcal{W}} a$.

PROOF. The proof of this fact uses the following result:

If $\mathcal{W}$ is a basic substitution calculus, c is a term of sort T and s a term of sort S. Then for every $m \geq 1$ and $n \geq 0$

$$\underline{m}[\mathit{lift}^n(\mathit{cons}(c,s))] =_{\mathcal{W}} \begin{cases} \underline{m-n-1}[s][\mathit{shift}]^n & \text{if } m > n+1 \\ \underline{m} & \text{if } m < n+1 \\ c[\mathit{shift}]^n & \text{if } m = n+1. \end{cases}$$

∎

LEMMA 3.18 (Substitution commutation)
Let $\mathcal{W}$ be a basic substitution calculus, a a pure term, b any term, s a term of sort S. Then for every $m \geq n \geq 0$ we have:

$a[\mathit{lift}^n(\mathit{cons}(b,\mathit{id}))][\mathit{lift}^m(s)] =_{\mathcal{W}} a[\mathit{lift}^{m+1}(s)][\mathit{lift}^n(\mathit{cons}(b[\mathit{lift}^{m-n}(s)],\mathit{id}))]$.

PROOF. By induction on the structure of a. See [30]. ∎

4 The Conversion Procedure

We now present the *Conversion Procedure*, an algorithm to translate any higher-order rewrite system in the formalism $SERS_{dB}$ to a first-order *ExERS*. The Conversion Procedure is somewhat involved since several conditions, mainly related to the labels of metavariables, must be met in order for a valuation to be admitted as *valid* (Definition 2.13). Consider for instance the η_{dB}-rewrite rule $\lambda(\mathit{app}(X_\alpha, \underline{1}))\rightarrow X_\epsilon$. The condition on valuations in $SERS_{dB}$ in order to participate in the induced rewrite relation on terms is that they be valid, as we have seen in Section 2. Validity ensures, in this case, that the metavariable X_α is not instantiated to the index $\underline{1}$. The Conversion Procedure has to guarantee that this holds in a first-order setting. The idea is to replace all occurrences of metavariables X_l by a first-order variable X followed by an appropriate *index-adjusting explicit substitution* which computes valid valuations. Thus, the output would be: $\lambda(\mathit{app}(X[\mathit{shift}], \underline{1}))\rightarrow X$. However this is just a simple case, and in the general situation, incorporating shift macros will not suffice. A witness to this fact is the commutation of binders rule in the introduction to this paper.

We first give the conversion rules of the translation, then we prove its properties in Section 5.

In order to define the conversion procedure we need two key notions that are essential to correctly manipulate all the metavariables appearing in a de Bruijn rewriting rule. The first notion, called binding allowance, gives the common binder indicators appearing in all the labels of the metavariables of a rule. If this binding allowance is empty, then the conversion is trivial, otherwise, we have to take into account the position in which these binder indicators occur to correctly define the conversion. This is done via the second notion called the shifting index.

DEFINITION 4.1 (Binding allowance)
Let A be a metaterm and $\{X_{l_1}, \ldots, X_{l_n}\}$ the set of all the metavariables with name X occurring in A. Then, the *binding allowance of X in A*, noted $\mathtt{Ba}_A(X)$, is the set $\bigcap_{i=1}^n l_i$. Likewise, we define the *binding allowance of X in a rule* (L,R), written $\mathtt{Ba}_{(L,R)}(X)$, as the set $\bigcap_{i=1}^n l_i$ where $\{X_{l_1}, \ldots, X_{l_n}\}$ is the set of all metavariables with the name X in L or R.

EXAMPLE 4.2
Let $A = f(\xi X_\alpha, g(\xi\lambda X_{\beta\alpha}, \xi\lambda X_{\alpha\gamma}))$, then $\mathtt{Ba}_A(X) = \{\alpha\}$.

DEFINITION 4.3 (Shifting index)
Let A be a metaterm, X_l a metavariable occurring in A, and i a position in l. The *shifting index determined by X_l at position i*, denoted $\mathtt{Sh}(X_l, i)$, is defined as

$$\mathtt{Sh}(X_l, i) \stackrel{\text{def}}{=} |\{j \mid \mathtt{at}(l,j) \notin \mathtt{Ba}_A(X), j \in 1..i-1\}|$$

Thus $\mathtt{Sh}(X_l, i)$ is just the total number of binder indicators in l at positions $1..i-1$ that do not belong to $\mathtt{Ba}_A(X)$. Remark that $\mathtt{Sh}(X_l, 1)$ is always 0.

EXAMPLE 4.4
If A is the metaterm $f(\xi X_\alpha, g(\xi\lambda X_{\beta\alpha}, \xi\lambda X_{\alpha\gamma}))$, then $\mathtt{Sh}(X_\alpha, 1) = \mathtt{Sh}(X_{\alpha\gamma}, 2) = 0$ and $\mathtt{Sh}(X_{\beta\alpha}, 2) = 1$.

Consider the rewrite rule $\lambda(\lambda(X_{\alpha\beta}))\rightarrow \lambda(\lambda(X_{\beta\alpha}))$ and a valid valuation for this rule κ. If κ maps the metavariable $X_{\alpha\beta}$ to a term a, then by the condition of validity it must be the case that it maps $X_{\beta\alpha}$ to the term b resulting from a where all 1-level and 2-level indices have been interchanged. For example, if $a = \underline{1}$ then $b = \underline{2}$ and if $a = \lambda\underline{2}$ then $b = \lambda\underline{3}$. Therefore, the conversion of the aforementioned rule would be

$$\lambda(\lambda X)\rightarrow \lambda(\lambda(X[\underline{2}\cdot\underline{1}\cdot(\uparrow\circ\uparrow)])). \tag{4.1}$$

In this discussion our focus was set on the metavariable $X_{\alpha\beta}$ in the sense that κ was assumed valid if the term to which $X_{\beta\alpha}$ is mapped was a suitable transformation of the one to which $X_{\alpha\beta}$ is mapped by κ. However, we may also state that κ is valid if the term it maps to $X_{\alpha\beta}$ is a suitable transformation of the one to which $X_{\beta\alpha}$ is mapped by κ. In this case, the conversion of the rewrite rule would be

$$\lambda(\lambda(X[\underline{2}\cdot\underline{1}\cdot(\uparrow\circ\uparrow)]))\rightarrow \lambda(\lambda X). \tag{4.2}$$

As a consequence, for each metavariable name in a rewrite rule, the metavariable that is set into focus determines the form that the conversion of this rule shall take (see also Example 4.13). The metavariable that is set into focus is called the *pivot* metavariable.

DEFINITION 4.5 (Pivot)
Let (L, R) be a $SERS_{dB}$-rewrite rule and $\{X_{l_1}, \ldots, X_{l_n}\}$ the set of all X-based metavariables in (L, R). If $\mathtt{Ba}_{(L,R)}(X) \neq \emptyset$, then X_{l_j} for some $j \in 1..n$ is called an *(X-based) pivot* if

1. $|l_j| \leq |l_i|$ for all $i \in 1..n$, and
2. $X_{l_j} \in L$, or $X_{l_j} \in R$ and $|l_j| < |l_i|$ for all $X_{l_i} \in \mathrm{FMVAR}(L)$.

A *pivot set for a rewrite rule* (L, R) is a set of pivot metavariables, one for each name X in L such that $\mathtt{Ba}_{(L,R)}(X) \neq \emptyset$. This notion extends to a set of rewrite rules as expected.

A pivot set for (L, R) fixes a metavariable for each metavariable name having a non-empty binding allowance. Note that Definition 4.5 admits the existence of more than one X-based pivot metavariable. We shall prove (Proposition 4.14), however, that the induced rewrite relation is unique, thus it is not biased by any particular choice of pivots. Nevertheless, the fact remains that the converted rewrite rule in each case differs substantially. For example, the rule (4.1) is a first-order rule in which syntactic matching suffices in order to apply it. However, the rule (4.2) requires matching modulo the equational theory of the substitution calculus. In order to favour the former over the latter in our definition of pivot we select a metavariable with shortest label on the LHS whenever possible. As a consequence, rule (4.2) is no longer obtainable since $X_{\beta\alpha}$ is not considered a valid X-based pivot according to Definition 4.5.

EXAMPLE 4.6
Both metavariables $X_{\alpha\beta}$ and $X_{\beta\alpha}$ can be chosen as X-based pivot in the rewrite rule

$$Implies(\exists\forall X_{\alpha\beta}, \forall\exists X_{\beta\alpha})\rightarrow true$$

In the rewrite rule $f(Y_\epsilon, g(\lambda\xi X_{\alpha\beta}, \lambda\xi X_{\beta\alpha}) \to \xi(X_\alpha, Y_\alpha)$ the metavariable X_α is the only possible X-based pivot. Also, since the binding allowance of Y in this rewrite rule is the empty set, no Y-based metavariable is declared as pivot.

Let us recall some notation from Definition 2.1. If $l = \alpha_1 \dots \alpha_n$ is a label of binder indicators then $\mathtt{at}(l, i) = \alpha_i$ for $i \in 1..n$. Also, $\mathtt{pos}(\alpha, l) = i$ where i is the smallest number in $1..n$ such that $\alpha = \alpha_i$, and is undefined otherwise.

DEFINITION 4.7 (Conversion of metavariables)
Consider a $SERS_{dB}$-rewrite rule (L, R) and a pivot set for (L, R). We consider the following cases for every metavariable name X occurring in L:

1. $\mathtt{Ba}_{(L,R)}(X) = \emptyset$. Then convert each metavariable X_l in (L, R) to the term $X[shift^{|l|}]$, and those metavariables X_l with $l = \epsilon$ simply to X.
   This shall allow, for example, the rewrite rule $f(\lambda(app(X_\alpha, \underline{1}), X_\epsilon)) \to X_\epsilon$ to be converted to the first-order rewrite rule $f(\lambda(app(X[shift], \underline{1}), X)) \to X$.
2. $\mathtt{Ba}_{(L,R)}(X) = \{\beta_1, \dots, \beta_m\}$ with $m > 0$. Let X_l be the pivot metavariable for X given by the hypothesis. We convert all occurrences of a metavariable X_k in (L, R) to the term $X[cons(b_1, \dots, b_{|l|}, shift^j)]$ where $j = |k| + |l \setminus \mathtt{Ba}_{(L,R)}(X)|$. The b_is depend on whether X_k is a pivot metavariable or not, as described below.
   As an optimization, in the particular case that the resulting term $X[cons(b_1, \dots, b_{|l|}, shift^j)]$ is of the form $X[cons(1, \dots, |l|, shift^{|l|})]$, we simply convert X_k to X.
   The substitution $cons(b_1, \dots, b_{|l|}, shift^j)$, is coined the *index-adjusting substitution corresponding to* X_k, and each b_i is defined as follows:
   (a) if X_k is the pivot (hence $l = k$), then

$$b_i = \begin{cases} \underline{i} & \text{if } \mathtt{at}(l, i) \in \mathtt{Ba}_{(L,R)}(X) \\ \underline{|l| + 1 + \mathtt{Sh}(X_l, i)} & \text{if } \mathtt{at}(l, i) \notin \mathtt{Ba}_{(L,R)}(X). \end{cases}$$

   (b) if X_k is not the pivot then

$$b_i = \begin{cases} \underline{\mathtt{pos}(\beta_h, k)} & \text{if } i = \mathtt{pos}(\beta_h, l) \text{ for some } \beta_h \in \mathtt{Ba}_{(L,R)}(X) \\ \underline{|k| + 1 + \mathtt{Sh}(X_l, i)} & \text{otherwise.} \end{cases}$$

   Recall that $\mathtt{at}(l, i)$ returns the symbol in label l at position i with $1 \le i \le |l|$, and $\mathtt{pos}(\alpha, l)$ returns the position of α in the label l assuming it is in l.

Note that for an index-adjusting substitution $cons(b_1, \dots, b_{|l|}, shift^j)$ each b_i is a distinct de Bruijn index and less than or equal to j. Substitutions of this form, in the particular case where we fix the basic substitution calculus to σ, have been called pattern substitutions in [19], where unification of higher-order patterns via explicit substitutions is studied.

Now that we know how to convert metavariables we can address the conversion of rewrite rules. Before proceeding we recall that the name of a metavariable X_l is X and that by abuse of notation we write $\text{FMVAR}(A)$ to denote the set of all the names of the free metavariables of M.

DEFINITION 4.8 (Conversion of rewrite rules)
Let (L, R) be a $SERS_{dB}$-rewrite rule and let P be a pivot set for (L, R). The *conversion of the rewrite rule* (L, R) *via* P, denoted $\mathcal{C}_P(L, R)$, is defined as $\mathcal{C}_P(L, R) \stackrel{\text{def}}{=} (\mathcal{C}_P^{(L,R)}(L), \mathcal{C}_P^{(L,R)}(R))$

where $\mathcal{C}_P^{(L,R)}(A)$ is defined by induction on A, where $\text{FMVAR}(A) \subseteq \text{FMVAR}(L)$, as:

$$
\begin{array}{lcl}
\mathcal{C}_P^{(L,R)}(\underline{n}) & \stackrel{\text{def}}{=} & \underline{n} \\
\mathcal{C}_P^{(L,R)}(X_l) & \stackrel{\text{def}}{=} & \begin{cases} X[shift^{|l|}] & \text{if } \mathtt{Ba}_{(L,R)}(X) = \emptyset \text{ and } l \neq \epsilon \\ X[cons(b_1, \ldots, b_{|l|}, shift^j)] & \text{if } \mathtt{Ba}_{(L,R)}(X) \neq \emptyset \text{ and } cons(b_1, \ldots, b_{|l|}, shift^j) \neq cons(1, \ldots, |l|, shift^{|l|}) \\ X & \text{otherwise} \end{cases} \\
\mathcal{C}_P^{(L,R)}(f(A_1, \ldots, A_n)) & \stackrel{\text{def}}{=} & f(\mathcal{C}_P^{(L,R)}(A_1), \ldots, \mathcal{C}_P^{(L,R)}(A_n)) \\
\mathcal{C}_P^{(L,R)}(\xi(A_1, \ldots, A_n)) & \stackrel{\text{def}}{=} & \xi(\mathcal{C}_P^{(L,R)}(A_1), \ldots, \mathcal{C}_P^{(L,R)}(A_n)) \\
\mathcal{C}_P^{(L,R)}(A_1[A_2]) & \stackrel{\text{def}}{=} & \mathcal{C}_P^{(L,R)}(A_1)[cons(\mathcal{C}_P^{(L,R)}(A_2), id)].
\end{array}
$$

The term $X[cons(b_1, \ldots, b_{|l|}, shift^j)]$ on the RHS of the second clause is the index-adjusting substitution computed in Definition 4.7.

It should be noted how the de Bruijn metasubstitution operator $\bullet[\bullet]$ is converted to the term substitution operator $\bullet[\bullet]$.

EXAMPLE 4.9
Below we present some examples of conversion of rules. We have fixed $\mathcal{W}$ to be the $\sigma_{\Uparrow}$-calculus.

$SERS_{dB}$**-rewrite rule**	**Converted rewrite rule**
$\lambda(app(X_\alpha, 1)) \rightarrow X_\epsilon$	$\lambda(app(X[\uparrow], 1)) \rightarrow X$
$\lambda(\lambda(X_{\alpha\beta})) \rightarrow \lambda(\lambda(X_{\beta\alpha}))$	$\lambda(\lambda X) \rightarrow \lambda(\lambda(X[2 \cdot 1 \cdot (\uparrow \circ \uparrow)]))$
$f(\lambda(\lambda(X_{\alpha\beta})), \lambda(\lambda(X_{\beta\alpha}))) \rightarrow \lambda(X_\gamma)$	$f(\lambda(\lambda(X[\uparrow \circ \uparrow])), \lambda(\lambda(X[\uparrow \circ \uparrow]))) \rightarrow \lambda(X[\uparrow])$
$app(\lambda X_\alpha, Y_\epsilon) \rightarrow_{\beta_{dB}} X_\alpha[Y_\epsilon]$	$app(\lambda X, Y) \rightarrow X[Y \cdot id]$

Regarding pivot selection:

1. the first rule requires no X-based pivot since the binding allowance of X is empty,
2. in the second rule $X_{\alpha\beta}$ is selected as X-based pivot,
3. the third rule requires no X-based pivot since the binding allowance of X is empty,
4. the fourth rule requires no Y-based pivot, however, the occurrence of X_α on the LHS is selected as X-based pivot.

Note that if the $SERS_{dB}$-rewrite rule (L, R) which is input to the Conversion Procedure is such that for every name X in (L, R) there is a label l with *all* metavariables in (L, R) of the form X_l, then all X_l are replaced simply by X. This is the case of β_{dB} of Example 4.9.

EXAMPLE 4.10 (Foldl)
Let us represent the usual $foldl$-recursion scheme over lists. Consider the *ExERS* signature containing $\Gamma_f = \{nil, const^1, foldl\}$ and $\Gamma_b = \{\xi\}$. Then the $foldl$-rewrite system:

$$\begin{array}{lcl} foldl(\xi(\xi(X_{\alpha\beta})), Y_\epsilon, nil) & \to & Y_\epsilon \\ foldl(\xi(\xi(X_{\alpha\beta})), Y_\epsilon, const(Z_\epsilon, W_\epsilon)) & \to & foldl(\xi(\xi(X_{\alpha\beta})), X_{\alpha\beta}[Y_\beta][Z_\epsilon], W_\epsilon) \end{array}$$

is converted to

$$\begin{array}{lcl} foldl(\xi(\xi(X)), Y, nil) & \to & Y \\ foldl(\xi(\xi(X)), Y, const(Z, W)) & \to & foldl(\xi(\xi(X)), X[cons(Y[\uparrow], id)][cons(Z, id)], W). \end{array}$$

EXAMPLE 4.11 (Natural numbers recursor)
Consider the *ExERS* signature containing the function symbols $\Gamma_f = \{zero, suc, rec\}$ and binder symbols $\Gamma_b = \{\xi\}$. Then the rec-rewrite system:

$$\begin{array}{lcl} rec(\xi(\xi(X_{\alpha\beta})), Y_\epsilon, zero) & \to & Y_\epsilon \\ rec(\xi(\xi(X_{\alpha\beta})), Y_\epsilon, suc(Z_\epsilon)) & \to & X_{\alpha\beta}[Z_\beta][rec(\xi(\xi(X_{\alpha\beta})), Y_\epsilon, Z_\epsilon)] \end{array}$$

is converted to

$$\begin{array}{lcl} rec(\xi(\xi(X)), Y, zero) & \to & Y \\ rec(\xi(\xi(X)), Y, suc(Z)) & \to & X[cons(Z[\uparrow], id)][cons(rec(\xi(\xi(X)), Y, Z), id)]. \end{array}$$

Also, observe that if we replace our $cons(\bullet, \bullet)$ macro by a $scons(\bullet)$ of substitution declaration T as defined in [29, 30] then the last clause of Definition 4.8 converts a metaterm of the form $A[B]$ into $A[scons(B)]$, yielding first-order systems based on substitution calculi, such as υ, which do not implement parallel substitution.

The system resulting from the Conversion Procedure is coded as an *ExERS*, a framework for defining first-order rewriting systems where $\mathcal{W}$-matching is used. Moreover, if it is possible, an *ExERS* may further be coded as a *FExERS* (Definition 3.9) where reduction is defined on first-order terms and matching is just syntactic first-order matching, obtaining a *full first-order system*.

DEFINITION 4.12 (Conversion Procedure)
Let Γ be an *ExERS* signature, let $\mathcal{R}$ be a $SERS_{dB}$, and let $\mathcal{W}$ be a substitution calculus over Γ. The *Conversion Procedure* consists in selecting a pivot set for each rewrite rule in $\mathcal{R}$ and converting all its rewrite rules as dictated by Definition 4.8. The resulting set of rewrite rules is written $fo(\mathcal{R})$. The *ExERS* $fo(\mathcal{R})_\mathcal{W}$ is called a *first order-version* of $\mathcal{R}$.

In what follows we shall assume given some fixed basic substitution calculus $\mathcal{W}$. Thus, given a $SERS_{dB}$ $\mathcal{R}$ we shall speak of *the* first-order version of $\mathcal{R}$.

Of course, we must also consider pivot selection and how it affects the conversion procedure. Assume given some rewrite rule (L, R) and different pivot sets P and Q for this rule. It is clear that $\mathcal{C}_P(L, R)$ and $\mathcal{C}_Q(L, R)$ are not identical.

EXAMPLE 4.13
Consider the following binder-commutation rule

$$imply(\exists\forall X_{\beta\alpha}, \forall\exists X_{\alpha\beta}) \to_C true.$$

[1] Although $cons$ is the usual abbreviation for the list constructor, we shall use $const$ so as not to cause confusion with the $cons$-macro.

If we select $X_{\beta\alpha}$ as the X-based pivot we obtain the following conversion of C:

$$imply(\exists\forall X, \forall\exists X[\underline{2} \cdot \underline{1} \cdot \uparrow^2]) \rightarrow_{C_{fo}} true.$$

However, $X_{\alpha\beta}$ may also be selected as an X-based pivot metavariable. In this case, the resulting converted rewrite rule will be different: $imply(\exists\forall X[\underline{2} \cdot \underline{1} \cdot \uparrow^2], \forall\exists X) \rightarrow_{C'_{fo}} true$

Nevertheless, the rewrite relation generated by both of these converted rewrite rules is identical.

PROPOSITION 4.14 (Pivot Selection)
Let (L, R) be a $SERS_{dB}$-rewrite rule and let P and Q be different pivot sets for this rule. Then the rewrite relation generated by both $\mathcal{C}_P(L, R)$ and $\mathcal{C}_Q(L, R)$ are identical.

Proposition 4.14 is important, for it makes clear that the Conversion Procedure is not biased by the selection of pivot sets (as regards the induced rewrite relation). Thus, in full precision, only now may we speak of *the* first-order version of a $SERS_{dB}$ $\mathcal{R}$. The proof of this proposition is rather technical and is relegated to the appendix.

5 Properties of the Conversion Procedure

This section studies the connection between higher-order rewriting and first-order rewriting modulo. Section 5.1 shows that the *Simulation Proposition* holds: any higher-order rewrite step may be simulated or implemented by first-order rewriting. Section 5.2 considers the *Projection Proposition*, namely, that rewrite steps in the first-order version of a higher-order system $\mathcal{R}$ can be projected in $\mathcal{R}$. Finally, we give in Section 5.3 a syntactical characterization of higher-order rewriting systems that can be translated into first-order rewriting systems modulo an empty theory. We shall see that, for example, the λ-calculus is covered by this characterization.

5.1 The Simulation Proposition

In order to simulate higher-order rewriting in a first-order framework we have to deal with the conversion of valid valuations into first-order valuations. Recall that valuations are the devices through which $SERS_{dB}$-rewrite rules are instantiated in order to obtain the induced rewrite relation. Likewise, first-order valuations are used for instantiating first-order rewrite rules, i.e. *ExERS*-rewrite rules. For converting valuations to first-order valuations two families of index-adjustment operations are required, decrementors and adjusters.

Consider a metavariable X_l in a $SERS_{dB}$-rewrite rule (L, R), and suppose we are given a valid de Bruijn valuation κ. Let $X[cons(b_1, \ldots, b_{|k|}, shift^j)]$ be the conversion of the metavariable X_l (Definition 4.7) where k is the label of the X-based pivot metavariable. We shall seek to define a first-order valuation ρ such that the value that ρ assigns to X satisfies the following equation:

$$\rho(X)[cons(b_1, \ldots, b_{|k|}, shift^j)] =_{\mathcal{W}} \kappa(X_l). \tag{5.1}$$

The term assigned to $\rho(X)$ is obtained by computing, $cons(b_1, \ldots, b_{|k|}, shift^j)^{-1}$, an 'inverse' substitution of $cons(b_1, \ldots, b_{|k|}, shift^j)$ from $\kappa(X_l)$ so that:

$$\rho(X) = \kappa(X_l)[cons(b_1, \ldots, b_{|k|}, shift^j)^{-1}].$$

In the case that $|k| = 0$ the inverse substitution of $shift^j$ is computed by the so called *decrementors* (Definition 5.1). Otherwise, it is computed by the *adjusters* (Definition 5.3). Decrementors

and adjusters are then used for defining the conversion of a valid valuation κ (Definition 5.6). Finally, Lemma 5.7 proves that the aforementioned conversion of κ behaves as expected, namely that equation (5.1) is verified.

DEFINITION 5.1 (Decrementors)
For every $i, j \geq 0$ and de Bruijn ground term a we define $\mathcal{D}_i^j(a)$ as follows:

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

LEMMA 5.2 (Decrementors)
Let (L, R) be a $SERS_{dB}$-rewrite rule, X_l, X_k metavariables in (L, R) and κ a valuation valid for (L, R). For all $i \geq 0$, if

1. $\kappa X_l = D[a]$ for some pure context D having binder path number i,
2. $\mathsf{Value}^i(l, a) = \mathsf{Value}^i(k, b)$, and
3. the binding allowance of X in (L, R) is the empty set (i.e. $\mathtt{Ba}_{(L,R)}(X) = \emptyset$),

then $\mathcal{D}_i^{|l|}(a)[\mathit{lift}^i(\mathit{shift}^{|k|})] =_{\mathcal{W}} b$.

PROOF. By induction on a.

- $a = \underline{n}$. We have three further cases to consider:
  1. $n \leq i$. Then $\mathcal{D}_i^{|l|}(\underline{n})[\mathit{lift}^i(\mathit{shift}^{|k|})] = \underline{n}[\mathit{lift}^i(\mathit{shift}^{|k|})] =_{\mathcal{W}}^{Lemma\ 3.15(1)} \underline{n}$. Now by Hypothesis 2 we have $\mathsf{Value}^i(l, \underline{n}) = \underline{n} = \mathsf{Value}^i(k, b)$ and therefore $b = \underline{n}$ and we are done.
  2. $i < n \leq i + |l|$. Since by Hypothesis 2 we have $\mathsf{Value}^i(l, \underline{n}) = \mathtt{at}(l, n-i) = \mathsf{Value}^i(k, b)$, it must be the case that $b = \underline{m}$ with $i < m \leq i + |k|$ and $\mathtt{at}(l, n-i) = \mathtt{at}(k, m-i)$. However by Hypothesis 3 there must be some $X_{l'}$ in (L, R) such that $\mathtt{at}(l, n-i) \notin l'$. Therefore $\mathsf{Value}(l', \kappa X_{l'}) \neq \mathsf{Value}(l, \kappa X_l)$ follows by Definition 2.13 (since $\mathtt{at}(l, n-i)$ occurs in $\mathsf{Value}(l, \kappa X_l)$ but $\mathtt{at}(l, n-i)$ does not occur in $\mathsf{Value}(l', \kappa X_{l'})$), contradicting the assumption that κ is valid.
  3. $n > i + |l|$. We reason as follows:

$$\begin{array}{lll} \mathcal{D}_i^{|l|}(\underline{n})[\mathit{lift}^i(\mathit{shift}^{|k|})] & = & (\underline{n - |l|})[\mathit{lift}^i(\mathit{shift}^{|k|})] \\ & =_{\mathcal{W}}^{Lemma.\ 3.15(1)} & (\underline{n - |l| - i})[\mathit{shift}^{|k|}][\mathit{shift}]^i \\ & =_{\mathcal{W}}^{Definition\ 3.8(8)} & (\underline{n - |l| - i + |k|})[\mathit{shift}]^i \\ & =_{\mathcal{W}} & \underline{n - |l| + |k|} \end{array}$$

     The last equality follows from i applications of Definition 3.8(8).
     By Hypothesis 2 we have $\mathsf{Value}^i(l, \underline{n}) = x_{n-i-|l|} = \mathsf{Value}^i(k, b)$ and therefore $b = \underline{m}$ with $m > i + |k|$ and $n - i - |l| = m - i - |k|$. From this it follows that $n - |l| = m - |k|$ and we are done.
- $a = f(a_1, \ldots, a_n)$. Then

$$\begin{array}{l} \mathcal{D}_i^{|l|}(a)[\mathit{lift}^i(\mathit{shift}^{|k|})] =_{\mathcal{W}} \\ f(\mathcal{D}_i^{|l|}(a_1)[\mathit{lift}^i(\mathit{shift}^{|k|})], \ldots, \mathcal{D}_i^{|l|}(a_n)[\mathit{lift}^i(\mathit{shift}^{|k|})]) \end{array}$$

by condition 3 of Definition 3.8.
By Hypothesis 2 we have that $b = f(b_1, \ldots, b_n)$ with $\mathsf{Value}^i(l, a_j) = \mathsf{Value}^i(k, b_j)$ for all $1 \leq j \leq n$. Then the induction hypothesis yields $\mathcal{D}_i^{|l|}(a_j)[\mathit{lift}^i(\mathit{shift}^{|k|})] =_{\mathcal{W}} b_j$ for $j \in 1..n$ and we may conclude the case.

- $a = \xi(a_1, \ldots, a_n)$. By condition 3 of Definition 3.8

$$\mathcal{D}_i^{|l|}(a)[\mathit{lift}^i(\mathit{shift}^{|k|})] =_{\mathcal{W}} \xi(\mathcal{D}_{i+1}^{|l|}(a_1)[\mathit{lift}^{i+1}(\mathit{shift}^{|k|})], \ldots, \mathcal{D}_{i+1}^{|l|}(a_n)[\mathit{lift}^{i+1}(\mathit{shift}^{|k|})]).$$

Finally, by Hypothesis 2, $b = \xi(b_1, \ldots, b_n)$ with $\mathsf{Value}^{i+1}(l, a_j) = \mathsf{Value}^{i+1}(k, b_j)$ for all $1 \leq j \leq n$. The induction hypothesis concludes the case.

∎

DEFINITION 5.3 (Adjusters)
Let X_l be a pivot metavariable in a $SERS_{dB}$-rewrite rule (L, R), $i \geq 1$, a a de Bruijn ground term and let $\mathit{cons}(b_1, \ldots, b_{|l|}, \mathit{shift}^{|l|+|l \setminus \mathtt{Ba}_{(L,R)}(X)|})$ be the index-adjusting substitution corresponding to X_l. Then $\mathcal{A}_i^l(a)$ is defined as follows:

$$\mathcal{A}_i^l(\underline{n}) \stackrel{\text{def}}{=} \begin{cases} \underline{n} & \text{if } n \leq i \\ \underline{n} & \text{if } \mathtt{at}(l, n-i) \in \mathtt{Ba}_{(L,R)}(X) \text{ and } 0 < n - i \leq |l| \\ \mathtt{undefined} & \text{if } \mathtt{at}(l, n-i) \notin \mathtt{Ba}_{(L,R)}(X) \text{ and } 0 < n - i \leq |l| \\ \underline{\mathtt{pos}(n - i, b_1 \ldots b_{|l|}) + i} & \text{if } |l| < n - i \leq |l| + |l \setminus \mathtt{Ba}_{(L,R)}(X)| \\ \underline{n - |l \setminus \mathtt{Ba}_{(L,R)}(X)|} & \text{if } n - i > |l| + |l \setminus \mathtt{Ba}_{(L,R)}(X)| \end{cases}$$

$\mathcal{A}_i^l(f(a_1 \ldots a_n)) \stackrel{\text{def}}{=} f(\mathcal{A}_i^l(a_1) \ldots \mathcal{A}_i^l(a_n))$
$\mathcal{A}_i^l(\xi(a_1 \ldots a_n)) \stackrel{\text{def}}{=} \xi(\mathcal{A}_{i+1}^l(a_1) \ldots \mathcal{A}_{i+1}^l(a_n))$.

LEMMA 5.4 (Well-definedness of Adjusters)
Consider a $SERS_{dB}$-rewrite rule (L, R) and some pivot set P for (L, R). Let $X_l \in (L, R)$ be the X-based pivot metavariable for some $X \in \mathrm{FMVAR}(L)$, and let κ be a valuation valid for (L, R). For all $i \geq 0$, if

1. $\kappa X_l = E[a]$ for some pure context E with i the binding path number of E, and
2. the binding allowance of X in (L, R) is not empty (i.e. $\mathtt{Ba}_{(L,R)}(X) \neq \emptyset$),

then $\mathcal{A}_i^l(a)$ is defined.

PROOF. By induction on a. We shall only consider the base case, the others follow by using the induction hypothesis. Suppose $a = \underline{n}$. We have four further cases to consider:

1. $n \leq i$. Then there is no problem.
2. $i < n \leq i + |l|$. The only case of conflict is if $\mathtt{at}(l, n - i) \notin \mathtt{Ba}_{(L,R)}(X)$. Then there must exist $X_{l'}$ in L such that $\mathtt{at}(l, n - i) \notin l'$. Consequently $\mathsf{Value}(l, \kappa X_l) \neq \mathsf{Value}(l', \kappa X_{l'})$ since $\mathtt{at}(l, n - i)$ occurs in $\mathsf{Value}(l, \kappa X_l)$ but $\mathtt{at}(l, n - i)$ does not occur in $\mathsf{Value}(l', \kappa X_{l'})$. This contradicts the assumption that κ is valid for (L, R).
3. $|l| < n - i \leq |l| + |l \setminus \mathtt{Ba}_{(L,R)}(X)|$. Then we must verify that $\mathtt{pos}(n - i, b_1 \ldots b_{|l|})$ is defined. Now let $r = |l \setminus \mathtt{Ba}_{(L,R)}(X)|$ then by Definition 4.7 there are subindices $j_1 < \ldots < j_r$ such that $b_{j_1} = \underline{|l| + 1 + \mathtt{Sh}(X_l, j_1)}, \ldots, b_{j_r} = \underline{|l| + 1 + \mathtt{Sh}(X_l, j_r)}$. By noting that $1 + \mathtt{Sh}(X_l, j_q) = q$ for $q \in 1 \ldots r$ we are done.

4. $n - i > |l| + |l \setminus \mathtt{Ba}_{(L,R)}(X)|$. This case presents no problems.

∎

LEMMA 5.5 (Adjusters)
Consider a $SERS_{dB}$-rewrite rule (L, R) and some pivot set P for (L, R). Let $X_l \in (L, R)$ be the X-based pivot metavariable for some $X \in \text{FMVAR}(L)$, let $X_k \in (L, R)$, and let κ be a valuation valid for (L, R). For all $i \geq 0$, if

1. $\kappa X_l = D[a]$ for some pure context D having binder path number equal to i,
2. $\mathsf{Value}^i(l, a) = \mathsf{Value}^i(k, b)$, and
3. the binding allowance of X in (L, R) is not empty (i.e. $\mathtt{Ba}_{(L,R)}(X) \neq \emptyset$),

then $\mathcal{A}^l_i(a)[\mathit{lift}^i(s)] =_{\mathcal{W}} b$ where $s = \mathit{cons}(c_1, \ldots, c_{|l|}, \mathit{shift}^{|k|+|l\setminus \mathtt{Ba}_{(L,R)}(X)|})$ is the index-adjusting substitution corresponding to X_k.

PROOF. Let $j = |k| + |l \setminus \mathtt{Ba}_{(L,R)}(X)|$. We proceed by induction on a.

- $a = \underline{n}$. We have four further cases to consider:
  1. $n \leq i$. Then
  $$\mathcal{A}^l_i(\underline{n})[\mathit{lift}^i(\mathit{cons}(c_1, \ldots, c_{|l|}, \mathit{shift}^j))] =$$
  $$\underline{n}[\mathit{lift}^i(\mathit{cons}(c_1, \ldots, c_{|l|}, \mathit{shift}^j))] =^{\mathit{Lemma}\ 3.15(1)}_{\mathcal{W}} \underline{n}.$$
  By Hypothesis 2, $\mathsf{Value}^i(l, \underline{n}) = \underline{n} = \mathsf{Value}^i(k, b)$ and therefore $b = \underline{n}$ and we are done.
  2. $i < n \leq i + |l|$. Here we consider the two cases:
  – $\mathtt{at}(l, n - i) \in \mathtt{Ba}_{(L,R)}(X)$. We reason as follows
  $$\begin{array}{ll} & \mathcal{A}^l_i(\underline{n})[\mathit{lift}^i(\mathit{cons}(c_1, \ldots, c_{|l|}, \mathit{shift}^j))] \\ = & \underline{n}[\mathit{lift}^i(\mathit{cons}(c_1, \ldots, c_{|l|}, \mathit{shift}^j))] \\ =^{L.\ 3.15(1)}_{\mathcal{W}} & (\underline{n - i})[\mathit{cons}(c_1, \ldots, c_{|l|}, \mathit{shift}^j)][\mathit{shift}]^i \\ =_{\mathcal{W}} & c_{n-i}[\mathit{shift}]^i \\ =_{\mathcal{W}} & \underline{c + i}\ (\text{for } c_{n-i} = \underline{c}). \end{array}$$
  So we are left to verify that $\underline{c + i} = b$.
  By Hypothesis 2, $\mathsf{Value}^i(l, \underline{n}) = \mathtt{at}(l, n - i) = \mathsf{Value}^i(k, b)$ and therefore $b = \underline{m}$ with $i < m \leq |k| + i$ and $\mathtt{at}(l, n - i) = \mathtt{at}(k, m - i)$.
  We consider where $c_{n-i} = \underline{c}$ might 'come from'.
  (a) $n - i = \mathtt{pos}(\beta_h, l)$ with $\beta_h \in \mathtt{Ba}_{(L,R)}(X)$ and $c_{n-i} = \mathtt{pos}(\beta_h, k)$. But then by Hypothesis 2 and the fact that k is a simple label we must have $c_{n-i} = \underline{m - i}$, which concludes the case.
  (b) There is no $\beta_h \in \mathtt{Ba}_{(L,R)}(X)$ with $n - i = \mathtt{pos}(\beta_h, l)$. This contradicts our assumption that $\mathtt{at}(l, n - i) \in \mathtt{Ba}_{(L,R)}(X)$.
  Note that in the particular case that $X_k = X_l$, then $c_{n-i} = \underline{n - i}$ and we have $n - i + i = n$.
  – $\mathtt{at}(l, n - i) \notin \mathtt{Ba}_{(L,R)}(X)$. By well-definedness of adjusters (Lemma 5.4) this case is not possible.
  3. $|l| < n - i \leq |l| + |l \setminus \mathtt{Ba}_{(L,R)}(X)|$. Then
  $$\begin{array}{ll} & \mathcal{A}^l_i(\underline{n})[\mathit{lift}^i(\mathit{cons}(c_1, \ldots, c_{|l|}, \mathit{shift}^j))] \\ = & (\underline{\mathtt{pos}(n - i, d_1 \ldots d_{|l|}) + i})[\mathit{lift}^i(\mathit{cons}(c_1, \ldots, c_{|l|}, \mathit{shift}^j))] \\ =_{\mathcal{W}} & \underline{\mathtt{pos}(n - i, d_1 \ldots d_{|l|})}[\mathit{cons}(c_1, \ldots, c_{|l|}, \mathit{shift}^j)][\mathit{shift}]^i \\ =_{\mathcal{W}} & c_r[\mathit{shift}]^i \\ =_{\mathcal{W}} & \underline{c + i}\ (\text{for } c_r = \underline{c}) \end{array}$$

where $r = \mathtt{pos}(n-i, d_1 \dots d_{|l|})$. Note that Lemma 5.4 is used here.
So we are left to verify that $\underline{c+i} = b$.
We must consider where c_r might 'come from':

(a) $r = \mathtt{pos}(\beta_h, l)$ with $\beta_h \in \mathtt{Ba}_{(L,R)}(X)$ and $c_r = \mathtt{pos}(\beta_h, k)$. Then clearly, $\mathtt{at}(l,r) \in \mathtt{Ba}_{(L,R)}(X)$. However, since $r = \mathtt{pos}(n-i, d_1 \dots d_{|l|})$ this means that $d_r = \underline{n-i}$. Also, recall that we are currently considering the case $d_r = \underline{n-i}$ where $n - i > |l|$. But then by Definition 4.7 $\mathtt{at}(l,r) \notin \mathtt{Ba}_{(L,R)}(X)$ contradicting our knowledge of the opposite fact.

(b) $c_r = \underline{|k| + 1 + \mathtt{Sh}(X_l, r)}$.
Now note that it is not possible for $d_r = \underline{r}$ (and hence $\mathtt{at}(l,r) \in \mathtt{Ba}_{(L,R)}(X)$) since then we may reason as in item 3a. So $d_r = \underline{n-i} = \underline{|l| + 1 + \mathtt{Sh}(X_l, r)}$ (*). Recall that we are left to verify that $\underline{|k| + 1 + \mathtt{Sh}(X_l, r) + i} = b$.
Now by Hypothesis 2 we have $\mathsf{Value}^i(l, \underline{n}) = x_{n-i-|l|} = \mathsf{Value}^i(k, b)$ and therefore $b = \underline{m}$ with $m > i + |k|$ and $n - i - |l| = m - i - |k|$. From this it follows that $n - |l| = m - |k|$. So now we must see that $|k| + 1 + \mathtt{Sh}(X_l, r) + i = n - |l| + |k|$, or simply $1 + \mathtt{Sh}(X_l, r) + i = n - |l|$. This follows from (*).

Note that in the particular case where $X_k = X_l$, then $c_r = n - i$ and we have $n - i + i = n$.

4. $n - i > |l| + |l \setminus \mathtt{Ba}_{(L,R)}(X)|$. We reason as follows

$$\begin{array}{ll} & \mathcal{A}_i^l(\underline{n})[\mathit{lift}^i(\mathit{cons}(c_1, \dots, c_{|l|}, \mathit{shift}^j))] \\ = & \underline{(n - |l \setminus \mathtt{Ba}_{(L,R)}(X)|)}[\mathit{lift}^i(\mathit{cons}(c_1, \dots, c_{|l|}, \mathit{shift}^j))] \\ =_{\mathcal{W}} & \underline{(n - |l \setminus \mathtt{Ba}_{(L,R)}(X)| - i)}[\mathit{cons}(c_1, \dots, c_{|l|}, \mathit{shift}^j)][\mathit{shift}]^i \\ =_{\mathcal{W}} & \underline{n - |l \setminus \mathtt{Ba}_{(L,R)}(X)| - i - |l| + |k| + |l \setminus \mathtt{Ba}_{(L,R)}(X)| + i} \\ = & \underline{n - |l| + |k|}. \end{array}$$

Note that in the particular case that $X_k = X_l$, we have $k = l$ and the result holds directly. Otherwise, by Hypothesis 2, $\mathsf{Value}^i(l, \underline{n}) = x_{n-i-|l|} = \mathsf{Value}^i(k, b)$ and therefore $b = \underline{m}$ with $m > i + |k|$ and $n - i - |l| = m - i - |k|$. From this it follows that $n - |l| = m - |k|$ and we may conclude the case.

- $a = f(a_1, \dots, a_n)$ (the case $a = \xi(a_1, \dots, a_n)$ is similar to this one and hence is ommitted). In this case, if $t = \mathit{cons}(c_1, \dots, c_{|l|}, \mathit{shift}^j)$, then

$$\mathcal{A}_i^l(a)[\mathit{lift}^i(t)] =_{\mathcal{W}} f(\mathcal{A}_i^l(a_1)[\mathit{lift}^i(t)], \dots, \mathcal{A}_i^l(a_n)[\mathit{lift}^i(t)]).$$

By Hypothesis 2, $b = f(b_1, \dots, b_n)$ with $\mathsf{Value}^i(l, a_j) = \mathsf{Value}^i(k, b_j)$ for all $1 \le j \le n$. The induction hypothesis concludes the case.

∎

We know how to convert $SERS_{dB}$-rewrite rules. In order to prove our simulation result we must convert $SERS_{dB}$-valuations. As already stated, this makes use of decrementors and adjusters.

DEFINITION 5.6 (Valuation conversion)
Let (L, R) be a $SERS_{dB}$-rewrite rule, κ a valid valuation for (L, R) and P a pivot set for (L, R). The *conversion of κ via P* is defined as the first-order valuation ρ where for each $X \in \mathrm{FMVAR}(L)$:

- Case $\mathtt{Ba}_{(L,R)}(X) = \emptyset$. Then $\rho(X) \stackrel{\text{def}}{=} \mathcal{D}_0^{|l|}(\kappa X_l)$ where X_l is any metavariable from L. Validity of κ implies that ρ does not depend on the particular metavariable X_l chosen. A formal proof of this fact may be found in the appendix (Lemma A.3, taking $D = \Box$).

- Case $\mathtt{Ba}_{(L,R)}(X) = \{\beta_1, \ldots, \beta_n\}$ with $n > 0$. Then we define $\rho(X) \stackrel{\text{def}}{=} \mathcal{A}_0^l(\kappa X_l)$ where X_l is the X-based pivot metavariable as dictated by P.

We need one final result before considering the simulation proposition, namely the one that states that the valuation conversion as defined above indeed verifies equation (5.1). Its proof relies on the Decrementors Lemma and the Adjustors Lemma and may be found in the appendix.

LEMMA 5.7
Let (L, R) be a $SERS_{dB}$-rewrite rule, κ a valid valuation for (L, R) and $\overline{\rho}$ the conversion of κ via P for some pivot set P for (L, R). If $L = C[A]$ for some metacontext C and metaterm A, then $\overline{\rho}(\mathcal{C}_P^{(L,R)}(A)) =_{\mathcal{W}} \kappa A$. Likewise, if $R = C[A]$ then $\overline{\rho}(\mathcal{C}_P^{(L,R)}(A)) =_{\mathcal{W}} \kappa(A)$.

PROPOSITION 5.8 (Simulation Proposition)
Let $\mathcal{R}$ be a $SERS_{dB}$ and let $fo(\mathcal{R})_{\mathcal{W}}$ be its first-order version. Suppose $a \to_{\mathcal{R}} b$ then

1. if $fo(\mathcal{R})_{\mathcal{W}}$ is an *ExERS* then $a \to_{fo(\mathcal{R})/\mathcal{W}} b$;
2. if $fo(\mathcal{R})_{\mathcal{W}}$ is a *FExERS* then $a \to_{fo(\mathcal{R})} \circ \twoheadrightarrow_{\mathcal{W}} b$ where $\circ$ denotes relation composition.

PROOF. For the first item, suppose $a \to_{\mathcal{R}} b$. Then there must be a $SERS_{dB}$-rewrite rule $(L, R) \in \mathcal{R}$, a valuation κ valid for (L, R) and a pure context E such that $a = E[\kappa L]$ and $b = E[\kappa R]$. Let $(L', R') = \mathcal{C}_P(L, R)$ be the converted version of rule (L, R) via some pivot set P for (L, R). Let $\overline{\rho}$ be the conversion of κ via P (Definition 5.6). By Lemma 5.7 we have:

1. $\overline{\rho}(L') =_{\mathcal{W}} \kappa L$ and
2. $\overline{\rho}(R') =_{\mathcal{W}} \kappa R$.

Thus from $\overline{\rho}(L') =_{\mathcal{W}} \kappa L$ and $\overline{\rho}(R') =_{\mathcal{W}} \kappa R$ we have $E[\overline{\rho}(L')] =_{\mathcal{W}} E[\kappa L]$ and $E[\overline{\rho}(R')] =_{\mathcal{W}} E[\kappa R]$, respectively. Finally, we have on the the one hand $a = E[\kappa L] =_{\mathcal{W}} E[\overline{\rho}(L')]$, so $a =_{\mathcal{W}} E[\overline{\rho}(L')]$, and on the other, $b = E[\kappa R] =_{\mathcal{W}} E[\overline{\rho}(R')]$, so $b =_{\mathcal{W}} E[\overline{\rho}(R')]$.

As for the second item note that if $fo(\mathcal{R})_{\mathcal{W}}$ is a *FExERS* then L' is a pure term. Also, by definition, κ is a pure first-order valuation. Thus $\overline{\rho}(L') = \kappa L$. And $\overline{\rho}(R') \twoheadrightarrow_{\mathcal{W}} \kappa R$ since κR is a pure term. Therefore we have $a = E[\kappa L] = E[\overline{\rho}(L')] \to_{(L',R')} E[\overline{\rho}(R')] \twoheadrightarrow_{\mathcal{W}} E[\kappa R]$.

∎

5.2 The projection proposition

We now wish to prove that derivations in an *ExERS* or *FExERS* $fo(\mathcal{R})_{\mathcal{W}}$ may be projected into derivations in $\mathcal{R}$. This ensures in some sense that we did not add meaningless computations in the translated first-order system. As a consequence we prove that $fo(\mathcal{R})_{\mathcal{W}}$ is conservative over $\mathcal{R}$ (Definition 5.17).

We shall first begin by showing that if $a \rightrightarrows_{(L,R)} b$, then for any term s of sort S we have $\mathcal{W}(a[s]) \rightrightarrows_{(L,R)} \mathcal{W}(b[s])$. Intuitively, $a \rightrightarrows_{(L,R)} b$ means that a rewrites to b by applying a number of *parallel* (L, R)-rewrite steps (Definition 5.11).

LEMMA 5.9
Let A be a pre-metaterm and suppose $\mathcal{WF}_k(A)$. Consider a valuation κ such that $\text{MVAR}(A) \subseteq Dom(\kappa)$. Then $\mathcal{W}((\kappa A)[\mathit{lift}^{|k|}(s)]) = \iota_k A$ where ι_k is a valuation defined as:

$$\iota_k(X_{lk}) \stackrel{\text{def}}{=} \mathcal{W}((\kappa X_{lk})[\mathit{lift}^{|lk|}(s)])$$

for all l such that X_{lk} occurs in A.

PROOF. By induction on A.

- $A = \underline{n}$. Note that since $\mathcal{WF}_k(\underline{n})$ we have $n \leq |k|$. Then $LHS = \mathcal{W}((\kappa\underline{n})[lift^{|k|}(s)]) = \mathcal{W}(\underline{n}[lift^{|k|}(s)]) = \underline{n} = \iota_k\underline{n} = RHS$.
- $A = X_{k'}$. Then since $\mathcal{WF}_k(X_{k'})$ we have $k = k'$ and $LHS = \mathcal{W}((\kappa X_k)[lift^{|k|}(s)]) = \iota_k A$.
- $A = f(A_1, \ldots, A_n)$. Then

$$\begin{array}{lll} LHS & =^{Definition\ 3.8(3)} & f(\mathcal{W}((\kappa A_1)[lift^{|k|}(s)]), \ldots, \mathcal{W}((\kappa A_n)[lift^{|k|}(s)])) \\ & =^{i.h.}_{\mathcal{W}} & f(\iota_k^1 A_1, \ldots, \iota_k^n A_n) \\ & = & f(\iota_k A_1, \ldots, \iota_k A_n) \\ & = & RHS \end{array}$$

  where $\iota_k = \bigcup_{i=1}^n \iota_k^i$. Note that if $X_p \in Dom(\iota_k^j) \cap Dom(\iota_k^{j'})$ for $j, j' \in 1..n$ with $j \neq j'$ then $\iota_k^j(X_p) = \iota_k^{j'}(X_p)$.
- $A = \xi(A_1, \ldots, A_n)$. By hypothesis there is an α such that $\mathcal{WF}_{\alpha k}(A_i)$ for all $i \in 1..n$. Then

$$\begin{array}{lll} LHS & =^{Definition\ 3.8(3)} & \xi(\mathcal{W}((\kappa A_1)[lift^{|k|+1}(s)]), \ldots, \mathcal{W}((\kappa A_n)[lift^{|k|+1}(s)])) \\ & =^{i.h.}_{\mathcal{W}} & \xi(\iota_{\alpha k}^1 A_1, \ldots, \iota_{\alpha k}^n A_n) \\ & = & \xi(\iota_{\alpha k} A_1, \ldots, \iota_{\alpha k} A_n) \end{array}$$

  where $\iota_{\alpha k} = \bigcup_{i=1}^n \iota_{\alpha k}^i$. Note that if $X_p \in Dom(\iota_{\alpha k}^j) \cap Dom(\iota_{\alpha k}^{j'})$ for $j, j' \in 1..n$ with $j \neq j'$ then $\iota_{\alpha k}^j(X_p) = \iota_{\alpha k}^{j'}(X_p)$.
  By the well-formedness predicate we know that since any metavariable in A_i has the form $X_{p\alpha k}$ for some label p we have $\iota_k(A_i) = \iota_{\alpha k} A_i$ for all $i \in 1..n$. More precisely, in the definition of $\iota_{\alpha k}$ let p be a label such that $X_{p\alpha k}$ is a metavariable in A_i for some $i \in 1..n$, then in the definition of ι_k we take $p' = p\alpha$ and obtain $\iota_k(X_{p'k}) = \iota_{\alpha k}(X_{p\alpha k})$. Hence we may continue as follows:

$$\xi(\iota_{\alpha k} A_1, \ldots, \iota_{\alpha k} A_n) = \xi(\iota_k A_1, \ldots, \iota_k A_n) = \iota_k A$$

- $A = A_1[A_2]$. By hypothesis there is an α such that $\mathcal{WF}_{\alpha k}(A_1)$, and $\mathcal{WF}_k(A_2)$. Then

$$\begin{array}{lll} LHS & = & \mathcal{W}((\kappa(A_1[A_2]))[lift^{|k|}(s)]) \\ & = & \mathcal{W}((\kappa A_1\{\!\{1 \leftarrow \kappa A_2\}\!\})[lift^{|k|}(s)]) \\ & =_{Lemma\ 3.15(3)} & \mathcal{W}((\kappa A_1[cons(\kappa A_2, id)])[lift^{|k|}(s)]) \\ & =_{Lemma\ 3.18} & \mathcal{W}((\kappa A_1)[lift^{|k|+1}(s)][cons((\kappa A_2)[lift^{|k|}(s)], id)]) \\ & = & \mathcal{W}(\mathcal{W}((\kappa A_1)[lift^{|k|+1}(s)])[cons(\mathcal{W}((\kappa A_2)[lift^{|k|}(s)]), id)]) \\ & =_{Lemma\ 3.15(3)} & \mathcal{W}((\kappa A_1)[lift^{|k|+1}(s)])\{\!\{1 \leftarrow \mathcal{W}((\kappa A_2)[lift^{|k|}(s)])\}\!\} \\ & =_{i.h.} & \iota_{\alpha k}(A_1)\{\!\{1 \leftarrow \iota_k(A_2)\}\!\} \\ & = & \iota_k(A_1)\{\!\{1 \leftarrow \iota_k(A_2)\}\!\} \\ & = & \iota_k(A_1[A_2]). \end{array}$$

  The before last equality may be justified as in the previous case.

∎

We now verify that the valuation ι from Lemma 5.9 ($k = \epsilon$) is a valid valuation assuming κ is, and hence can be used in rewriting terms. More precisely,

LEMMA 5.10
Let κ be a valid valuation for a $SERS_{dB}$-rewrite rule (L, R) and let s be any substitution. Then ι is also valid for (L, R), where $\iota(X_l) \stackrel{\text{def}}{=} \mathcal{W}((\kappa X_l)[lift^{|l|}(s)])$ for all X_l in (L, R).

PROOF. This follows from the following more general result by considering the case $i = 0$. Let a, b be pure terms. Then for all $i \geq 0$, $\mathsf{Value}^i(k_1, a) = \mathsf{Value}^i(k_2, b)$ implies

$$\mathsf{Value}^i(k_1, \mathcal{W}(a[lift^{|k_1|+i}(s)])) = \mathsf{Value}^i(k_2, \mathcal{W}(b[lift^{|k_2|+i}(s)])).$$

The latter is proved by induction on a. We shall consider the case where a is an index for the other cases follow by using the induction hypothesis. Let $a = \underline{n}$, we consider three further subcases:

- $n \leq i$. Then $b = \underline{n}$ and
$$\mathsf{Value}^i(k_1, \mathcal{W}(a[lift^{|k_1|+i}(s)])) = \underline{n} = \mathsf{Value}^i(k_2, \mathcal{W}(b[lift^{|k_2|+i}(s)])).$$
The latter holds by Lemma 3.15 (1), and therefore, the result holds.
- $i < n \leq |k_1| + i$. Then $b = \underline{m}$ with $i < m \leq |k_2| + i$ and $\mathtt{at}(k_1, n - i) = \mathtt{at}(k_2, m - i)$. We have $\mathcal{W}(a[lift^{|k_1|+i}(s)]) = \underline{n}$ and $\mathcal{W}(b[lift^{|k_2|+i}(s)]) = \underline{m}$, by Lemma 3.15(1). Thus, we have
$$\begin{array}{rcl} & & \mathsf{Value}^i(k_1, \mathcal{W}(a[lift^{|k_1|+i}(s)])) \\ = & & \mathtt{at}(k_1, n - i) \\ = & & \mathtt{at}(k_2, m - i) \\ = & & \mathsf{Value}^i(k_2, \mathcal{W}(b[lift^{|k_2|+i}(s)])). \end{array}$$
- $n > |k_1| + i$. Then $b = \underline{m}$ with $m > |k_2| + i$ and $x_{n-|k_1|-i} = x_{m-|k_2|-i}$. Then we reason as follows:
$$\begin{array}{rcl} \mathcal{W}(a[lift^{|k_1|+i}(s)]) & = & \mathcal{W}(\underline{n - |k_1| - i}[s][shift]^{|k_1|+i}) \\ & = & \mathcal{W}(\mathcal{W}(\underline{n - |k_1| - i}[s])[shift]^{|k_1|+i}). \end{array}$$
And likewise,
$$\begin{array}{rcl} \mathcal{W}(b[lift^{|k_2|+i}(s)]) & = & \mathcal{W}(\underline{m - |k_2| - i}[s][shift]^{|k_2|+i}) \\ & = & \mathcal{W}(\mathcal{W}(\underline{m - |k_2| - i}[s])[shift]^{|k_2|+i}). \end{array}$$
Now $\mathcal{W}(\underline{n - |k_1| - i}[s]) = \mathcal{W}(\underline{m - |k_2| - i}[s])$, since $n - |k_1| - i = m - |k_2| - i$.

Observation: $\mathsf{Value}^i(k_1, \mathcal{W}(a[shift]^{|k_1|+i})) = \mathsf{Value}^i(k_2, \mathcal{W}(a[shift]^{|k_2|+i}))$ holds for any pure term a. This may be verified by induction on a and using condition 8 of the definition of a Basic Substitution Calculus (Definition 3.8).

By the observation we may conclude the case.

∎

DEFINITION 5.11 (Parallel $SERS_{dB}$-rewriting)
Let $\mathcal{R}$ be a $SERS_{dB}$ and let a and b be de Bruijn terms. We say that a *$\mathcal{R}$-rewrites in parallel* to b iff $a \rightrightarrows_{\mathcal{R}} b$, where the latter relation is defined as:

$$\frac{}{a \rightrightarrows_{\mathcal{R}} a}\,(\mathtt{refl}) \qquad \frac{\kappa \text{ valid for } (L, R) \in \mathcal{R}}{\kappa L \rightrightarrows_{\mathcal{R}} \kappa R}\,(\mathtt{red})$$

$$\frac{a_i \rightrightarrows_{\mathcal{R}} b_i \quad \text{for all } 1 \leq i \leq n}{f(a_1, \ldots, a_n) \rightrightarrows_{\mathcal{R}} f(b_1, \ldots, b_n)}\,(\mathtt{clos\text{-}f}) \qquad \frac{a_i \rightrightarrows_{\mathcal{R}} b_i \quad \text{for all } 1 \leq i \leq n}{\xi(a_1, \ldots, a_n) \rightrightarrows_{\mathcal{R}} \xi(b_1, \ldots, b_n)}\,(\mathtt{clos\text{-}b})$$

Note that $\rightarrow_{\mathcal{R}} \subseteq \rightrightarrows_{\mathcal{R}} \subseteq \twoheadrightarrow_{\mathcal{R}}$, and that $\rightrightarrows_{\mathcal{R}}$ is reflexive. In the case of $\mathcal{R} = \{(L, R)\}$ we shall abbreviate $a \rightrightarrows_{\mathcal{R}} b$ as $a \rightrightarrows_{(L,R)} b$.

LEMMA 5.12
Let a, b be pure terms and let (L, R) be a $SERS_{dB}$-rewrite rule. If $a \rightrightarrows_{(L,R)} b$, then for any term s of sort S we have $\mathcal{W}(a[s]) \rightrightarrows_{(L,R)} \mathcal{W}(b[s])$.

PROOF. By induction on the derivation of $a \rightrightarrows_{(L,R)} b$.

- `refl`. Then the result holds trivially.
- `red`. Then a can be written as $\kappa L = L\{X^{i_1}_{l_1}/\kappa X^{i_1}_{l_1}, \ldots, X^{i_n}_{l_n}/\kappa X^{i_n}_{l_n}\}$, where $X^{i_1}_{l_1}, \ldots, X^{i_n}_{l_n}$ are all the metavariables in L, and κ is a valid valuation for (L, R). We have

  $$\mathcal{W}(a[s]) = L\{X^{i_1}_{l_1}/\mathcal{W}((\kappa X^{i_1}_{l_1})[\mathit{lift}^{|l_1|}(s)]), \ldots, X^{i_n}_{l_n}/\mathcal{W}((\kappa X^{i_n}_{l_n})[\mathit{lift}^{|l_n|}(s)])\}.$$

  So define $\iota X^{i_j}_{l_j} \stackrel{\text{def}}{=} \mathcal{W}((\kappa X^{i_j}_{l_j})[\mathit{lift}^{|l_j|}(s)])$. Since ι is valid for (L, R) by Lemma 5.10, an application of `red` allows us to conclude: $\mathcal{W}(a[s]) \rightrightarrows_{(L,R)} \iota(R) =_{Lemma\ 5.9(k=\epsilon)} \mathcal{W}((\kappa R)[s]) = \mathcal{W}(b[s])$.
- `clos-f`. Then by the induction hypothesis we have $\mathcal{W}(a_i[s]) \rightrightarrows_{(L,R)} \mathcal{W}(b_i[s])$ for all $1 \leq i \leq n$. We conclude using `clos-f` $\mathcal{W}(f(a_1, \ldots, a_n)[s]) = f(\mathcal{W}(a_1[s]), \ldots, \mathcal{W}(a_n[s])) \rightrightarrows_{(L,R)} f(\mathcal{W}(b_1[s]), \ldots, \mathcal{W}(b_n[s])) = \mathcal{W}(f(b_1, \ldots, b_n)[s])$.
- `clos-b`. As in the case `clos-f`.

∎

Note that in particular Lemma 5.12 holds when $a \rightarrow_{(L,R)} b$ since the one-step rewrite relation is included in the parallel rewrite relation: if $a \rightarrow_{(L,R)} b$, then for any term s of sort S we have $\mathcal{W}(a[s]) \rightrightarrows_{(L,R)} \mathcal{W}(b[s])$.

LEMMA 5.13 (Projecting first-order valuations)
Let (L, R) be a $SERS_{dB}$-rewrite rule, $(L', R') = \mathcal{C}_P(L, R)$ for some pivot set P for (L, R), let ρ be a first-order valuation for (L', R').

Define the valuation κ as:

$$\kappa X_k \stackrel{\text{def}}{=} \mathcal{W}(\overline{\rho}(\mathcal{C}_P^{(L,R)}(X_k))).$$

For any metaterm A, $\mathcal{W}(\overline{\rho}(\mathcal{C}_P^{(L,R)}(A))) = \kappa A$.

PROOF. By induction on A.

- $A = \underline{n}$. Then $LHS = \mathcal{W}(\overline{\rho}(\underline{n})) = \underline{n} = \kappa\underline{n} = RHS$.
- $A = X_k$. Then $LHS = \mathcal{W}(\overline{\rho}(\mathcal{C}_P^{(L,R)}(X_k))) =_{hypothesis} \kappa X_k$.
- $A = f(A_1, \ldots, A_n)$. Then

  $$\begin{array}{l} LHS =^{Definition\ 3.8(3)} \\ f(\mathcal{W}(\overline{\rho}(\mathcal{C}_P^{(L,R)}(A_1))), \ldots, \mathcal{W}(\overline{\rho}(\mathcal{C}_P^{(L,R)}(A_n)))) =^{i.h.}_{\mathcal{W}} \\ f(\kappa A_1, \ldots, \kappa A_n) = RHS. \end{array}$$

- $A = \xi(A_1, \ldots, A_n)$. As the previous case.

- $A = A_1[A_2]$. Then

$$\begin{array}{lll} LHS & = & \mathcal{W}(\overline{\rho}(\mathcal{C}_P^{(L,R)}(A_1[A_2]))) \\ & = & \mathcal{W}(\overline{\rho}(\mathcal{C}_P^{(L,R)}(A_1))[cons(\overline{\rho}(\mathcal{C}_P^{(L,R)}(A_2)), id)]) \\ & =^{Definition\ 3.8(1)} & \mathcal{W}(\mathcal{W}(\overline{\rho}(\mathcal{C}_P^{(L,R)}(A_1)))[cons(\mathcal{W}(\overline{\rho}(\mathcal{C}_P^{(L,R)}(A_2))), id)]) \\ & =^{i.h.}_{\mathcal{W}} & \mathcal{W}(\kappa A_1[cons(\kappa A_2, id)]) \\ & =_{Lemma\ 3.15(3)} & \kappa A_1 \{\!\{1 \leftarrow \kappa A_2\}\!\} \\ & = & \kappa(A_1[A_2]) \\ & = & RHS. \end{array}$$

∎

In order to use the valuation of Lemma 5.13 we need to prove that it is valid.

LEMMA 5.14 (From first-order valuations to valid valuations)
Consider rewrite rule (L, R) in the $SERS_{dB}$ formalism, metavariables X_{k_1}, X_{k_2} occurring in (L, R) and a designated pivot metavariable X_l. Let ρ be a first-order valuation. Then

$$\mathsf{Value}(k_1, \mathcal{W}(\overline{\rho}(X[s_1]))) = \mathsf{Value}(k_2, \mathcal{W}(\overline{\rho}(X[s_2])))$$

where

- $s_1 = cons(b_1, \ldots, b_{|l|}, shift^{|k_1|+|l \setminus \mathtt{Ba}_{(L,R)}(X)|})$ and
- $s_2 = cons(c_1, \ldots, c_{|l|}, shift^{|k_2|+|l \setminus \mathtt{Ba}_{(L,R)}(X)|})$

are the index-adjusting substitutions (using pivot X_l) of X_{k_1} and X_{k_2}, respectively.

PROOF. In order to prove this property we show a more general one stating that for all pure term a and for all $i \geq 0$ we have:

$$\mathsf{Value}^i(k_1, \mathcal{W}(a[lift^i(s_1)])) = \mathsf{Value}^i(k_2, \mathcal{W}(a[lift^i(s_2)]))$$

with s_1 and s_2 the index-adjusting substitutions as before.

We shall assume that $X_{k_1} \neq X_l$ and $X_{k_2} \neq X_l$. The case where $X_{k_1} = X_l$ or $X_{k_2} = X_l$ is analogous. We proceed by induction on a.

- $a = \underline{n}$. We have three subcases to consider.
  - $n \leq i$. Then by Lemma 3.15 (1) $\mathsf{Value}^i(k_1, \underline{n}) = \underline{n} = \mathsf{Value}^i(k_2, \underline{n})$.
  - $i < n \leq |l| + i$. Now we consider two further cases:
    - $n - i = \mathtt{pos}(\beta_h, l)$ for some $\beta_h \in \mathtt{Ba}_{(L,R)}(X)$.
      Then $b_{n-i} = \underline{\mathtt{pos}(\beta_h, k_1)}$ and $c_{n-i} = \underline{\mathtt{pos}(\beta_h, k_2)}$ by Definition 4.7.
      Therefore $\mathsf{Value}^i(k_1, b_{n-i} + i) = \beta_h = \mathsf{Value}^i(k_2, c_{n-i} + i)$.
    - There is no $\beta_h \in \mathtt{Ba}_{(L,R)}(X)$ such that $n - i = \mathtt{pos}(\beta_h, l)$.
      Then $b_{n-i} = \underline{|k_1| + 1 + \mathtt{Sh}(X_l, n - i)}$ and $c_{n-i} = \underline{|k_2| + 1 + \mathtt{Sh}(X_l, n - i)}$.
      Hence, $\mathsf{Value}^i(k_1, b_{n-i} + i) = x_{1+\mathtt{Sh}(X_l, n-i)} = \mathsf{Value}^i(k_2, c_{n-i} + i)$.
  - $n > |l| + i$. Then $\mathcal{W}(a[lift^i(s_1)]) = \underline{n - |l| + |k_1| + |l \setminus \mathtt{Ba}_{(L,R)}(X)|}$ and we also have $\mathcal{W}(a[lift^i(s_2)]) = \underline{n - |l| + |k_2| + |l \setminus \mathtt{Ba}_{(L,R)}(X)|}$. As a consequence

$$\begin{array}{lll} & & \mathsf{Value}^i(k_1, \underline{n - |l| + |k_1| + |l \setminus \mathtt{Ba}_{(L,R)}(X)|}) \\ & = & x_{n-i-|l|+|l \setminus \mathtt{Ba}_{(L,R)}(X)|} \\ & = & \mathsf{Value}^i(k_2, \underline{n - |l| + |k_2| + |l \setminus \mathtt{Ba}_{(L,R)}(X)|}). \end{array}$$

- $a = f(a_1, \ldots, a_n)$. Then

$$\begin{array}{ll} & \mathsf{Value}^i(k_1, \mathcal{W}(a[\mathit{lift}^i(s_1)])) \\ = & \mathsf{Value}^i(k_1, f(\mathcal{W}(a_1[\mathit{lift}^i(s_1)]), \ldots, \mathcal{W}(a_n[\mathit{lift}^i(s_1)]))) \\ = & f(\mathsf{Value}^i(k_1, \mathcal{W}(a_1[\mathit{lift}^i(s_1)])), \ldots, \mathsf{Value}^i(k_1, \mathcal{W}(a_n[\mathit{lift}^i(s_1)]))) \\ =_{ih} & f(\mathsf{Value}^i(k_2, \mathcal{W}(a_1[\mathit{lift}^i(s_2)])), \ldots, \mathsf{Value}^i(k_2, \mathcal{W}(a_n[\mathit{lift}^i(s_2)]))) \\ = & \mathsf{Value}^i(k_2, \mathcal{W}(a[\mathit{lift}^i(s_2)])). \end{array}$$

- $a = \xi(a_1, \ldots, a_n)$. Similar to the previous case.

Now in order to obtain the general result of the lemma we remark that $\mathcal{W}$ is a basic substitution calculus, so that it has unique normal forms and in particular $\mathcal{W}(\overline{\rho}(X)[s_1]) = \mathcal{W}(\mathcal{W}(\overline{\rho}(X))[s_1])$. Also, by condition 2 of Definition 3.8 we have that $\mathcal{W}(\overline{\rho}(X))$ is a pure term and therefore we can take $a = \mathcal{W}(\overline{\rho}(X))$ and $i = 0$.

∎

LEMMA 5.15 (Projection of *ExERS*-rewriting)
Let $\mathcal{W}$ be a basic substitution calculus satisfying the scheme. Let o be a term of $\mathcal{W}$ of sort T or S. Let $(L', R') = \mathcal{C}_P(L, R)$. If $o \rightarrow_{(L',R')} o'$, then

1. if o is of sort T then $\mathcal{W}(o) \rightrightarrows_{(L,R)} \mathcal{W}(o')$;
2. for every pure term d of sort T such that o is a term of sort S, and every $n \geq 0$, $\mathcal{W}(d[\mathit{lift}^n(o)]) \rightrightarrows_{(L,R)} \mathcal{W}(d[\mathit{lift}^n(o')])$.

PROOF. We show simultaneously the two items by induction on the lexicographic ordering on pairs (o, d), where the orders on the components are given by the lengths of their respective terms[2].

- o is a de Bruijn index or a substitution constant. Then both items hold vacuously since by definition the LHS of a $SERS_{dB}$-rewrite rule must have a function or binder symbol as head symbol. Thus o is a normal form.
- $o = f(a_1, \ldots, a_n)$ or $o = \xi(a_1, \ldots, a_n)$. There is nothing to prove for the second item. For the first item we consider two cases.
  – Suppose the reduction is at the root. Then $o = \overline{\rho}L'$. Define κ for all $X_k \in L$ as:

$$\kappa X_k \stackrel{\text{def}}{=} \mathcal{W}(\overline{\rho}(\mathcal{C}_P^{(L,R)}(X_k))).$$

    Note that κ is a valid valuation by Lemma 5.14, and also, $\mathcal{W}(\overline{\rho}L') = \kappa L$ by Lemma 5.13. So $\kappa L \rightrightarrows_{(L,R)} \kappa R =_{Lemma\ 5.13} \mathcal{W}(\overline{\rho}R') = \mathcal{W}(o')$.
  – Suppose the reduction is internal. Then we use the induction hypothesis.
- $o = a[s]$. There is nothing to prove for the second item. Since reduction at the root of the term is not possible, we consider the following two cases for the first property:
  – $o' = a'[s]$ with $a \rightarrow_{(L',R')} a'$. By the i.h. $\mathcal{W}(a) \rightrightarrows_{(L,R)} \mathcal{W}(a')$. Then

$$\mathcal{W}(a[s]) = \mathcal{W}(\mathcal{W}(a)[s]) \rightrightarrows_{(L,R)} \mathcal{W}(\mathcal{W}(a')[s])$$

    by applying Lemma 5.12.

[2]The length of an index is 1.

– $o' = a[s']$ with $s \to_{(L',R')} s'$. Since $\mathcal{W}(a)$ is a pure term we have that

$$\mathcal{W}(o) = \mathcal{W}(\mathcal{W}(a)[s]) \rightrightarrows_{(L,R)} \mathcal{W}(\mathcal{W}(a)[s']) = \mathcal{W}(o')$$

by the induction hypothesis of item 2 since $(s, d) < (a[s], d)$.

- o is a substitution $\sigma(s_1, \ldots, s_j, \ldots, s_q)$ $(q > 0)$, and $o' = \sigma(s_1, \ldots, s'_j, \ldots, s_q)$, where $s_j \to_{(L',R')} s'_j$. There is nothing to prove for the first property since o is not a term. For the second property we proceed by induction on d.
  – $d = f(d_1, \ldots, d_n)$ or $d = \xi(d_1, \ldots, d_n)$ then the property holds by the induction hypothesis since $(o, d_i) < (o, d)$ for all $1 \leq i \leq n$, and applying `clos-f` or `clos-b`.
  – $d = \underline{m}$. We must verify that for all $n \geq 0$: $\mathcal{W}(\underline{m}[\mathit{lift}^n(o)]) \rightrightarrows_{(L,R)} \mathcal{W}(\underline{m}[\mathit{lift}^n(o')])$.
  We proceed by induction on n.
    1. If $n = 0$, then we proceed by cases as dictated by the definition of the scheme (Definition 3.13).
    (a) Suppose there exists a de Bruijn index $\underline{r}$, indices $i_1, \ldots, i_p$ $(p \geq 0)$ and also substitutions $u_1, \ldots, u_k$ $(k \geq 0)$ such that $1 \leq i_1, \ldots, i_p \leq q$, the i_j's are all distinct and for $s_1 \ldots s_q$, $\underline{m}[\sigma(s_1, \ldots s_q)] =_{\mathcal{W}} \underline{r}[s_{i_1}] \ldots [s_{i_p}][u_1] \ldots [u_k]$.
      i. If $j \notin \{i_1, \ldots, i_p\}$, then $\mathcal{W}(\underline{m}[o']) = \mathcal{W}(\underline{r}[s_{i_1}] \ldots [s_{i_p}][u_1] \ldots [u_k])$ and the property is trivial since $\mathcal{W}(\underline{m}[o]) = \mathcal{W}(\underline{m}[o'])$.
      ii. If $j \in \{i_1, \ldots, i_p\}$, let us say $j = i_h$, then the term $\mathcal{W}(\underline{m}[o'])$ is equal to the term $\mathcal{W}(\underline{r}[s_{i_1}] \ldots [s'_{i_h}] \ldots [s_{i_p}][u_1] \ldots [u_k])$ and $\mathcal{W}(\underline{r}[s_{i_1}] \ldots [s_{i_{h-1}}]) = e$ is a pure term by Definition 3.8(2). Since $(s_{i_h}, e) < (\sigma(s_1, \ldots, s_j, \ldots, s_q), \underline{m})$, we can apply the induction hypothesis (2) to obtain:

$$\mathcal{W}(e[s_{i_h}]) \rightrightarrows_{(L,R)} \mathcal{W}(e[s'_{i_h}]).$$

Now, the term $\mathcal{W}(e[s_{i_h}])$ is pure, so that we can repeatedly apply Lemma 5.12 to obtain:

$$\begin{array}{lll} \mathcal{W}(\underline{m}[o]) & = & \mathcal{W}(\mathcal{W}(e[s_{i_h}])[s_{i_{h+1}}] \ldots [s_{i_p}][u_1] \ldots [u_k]) \\ & \rightrightarrows_{(L,R)} & \mathcal{W}(\mathcal{W}(e[s'_{i_h}])[s_{i_{h+1}}] \ldots [s_{i_p}][u_1] \ldots [u_k]) \\ & = & \mathcal{W}(\underline{m}[o']). \end{array}$$

    (b) Suppose there exists an index $\underline{i}$ with $1 \leq i \leq q$ such that for $s_1 \ldots s_q$ we have that $\underline{m}[\sigma(s_1, \ldots s_q)] =_{\mathcal{W}} s_i$.
      i. If $i \neq j$, then the term $\mathcal{W}(\underline{m}[o'])$ is also equal to $\mathcal{W}(s_i)$ and the property is trivial since $\mathcal{W}(\underline{m}[o]) = \mathcal{W}(\underline{m}[o'])$.
      ii. If $i = j$, then $\mathcal{W}(\underline{m}[o']) = \mathcal{W}(s'_j)$ (where s_j is a term because the equations are well-typed) and $(s_j, \underline{m}) < (\sigma(s_1, \ldots, s_j, \ldots, s_q), \underline{m})$, so the property holds by the induction hypothesis (1) since $\mathcal{W}(\underline{m}[o]) = \mathcal{W}(s_j) \rightrightarrows_{(L,R)} \mathcal{W}(s'_j) = \mathcal{W}(\underline{m}[o'])$.
    2. If $n > 0$, then we consider two cases:
    (a) if $m \leq n$, then by Lemma 3.15(1) we obtain:

$$\mathcal{W}(\underline{m}[\mathit{lift}^n(o)]) = \underline{m} = \mathcal{W}(\underline{m}[\mathit{lift}^n(o')])$$

    (b) if $m > n$, then by Lemma 3.15(1) we obtain:

$$\mathcal{W}(\underline{m}[\mathit{lift}^n(o)]) = \mathcal{W}(\underline{m-1}[\mathit{lift}^{n-1}(o)][\mathit{shift}])$$

and

$$\mathcal{W}(\underline{m}[\mathit{lift}^n(o')]) = \mathcal{W}(\underline{m-1}[\mathit{lift}^{n-1}(o')][\mathit{shift}])$$

Since indices are equivalent w.r.t our ordering, $(\sigma(s_1,\ldots,s_j,\ldots,s_q),\underline{m}) = (\sigma(s_1,\ldots,s_j,\ldots,s_q),\underline{m-1})$, and then the induction hypothesis on n can be applied to obtain

$$\mathcal{W}(\underline{m-1}[\mathit{lift}^{n-1}(o)]) \rightrightarrows_{(L,R)} \mathcal{W}(\underline{m-1}[\mathit{lift}^{n-1}(o')]).$$

Since every $\mathcal{W}$-normal form is a pure term by Definition 3.8(2), we may finally apply Lemma 5.12, so that

$$\begin{array}{lll} \mathcal{W}(\underline{m}[\mathit{lift}^n(o)]) & = & \mathcal{W}(\mathcal{W}(\underline{m-1}[\mathit{lift}^{n-1}(o)])[\mathit{shift}]) \\ & \rightrightarrows_{(L,R)} & \mathcal{W}(\mathcal{W}(\underline{m-1}[\mathit{lift}^{n-1}(o')])[\mathit{shift}]) \\ & = & \mathcal{W}(\underline{m}[\mathit{lift}^n(o')]). \end{array}$$

∎

PROPOSITION 5.16 (Projection Proposition)
Let $\mathcal{R}$ be a $SERS_{dB}$ and let $fo(\mathcal{R})_{\mathcal{W}}$ be its first-order version where $\mathcal{W}$ is a basic substitution calculus satisfying the scheme. If $a \rightarrow_{fo(\mathcal{R})_{\mathcal{W}}} b$, then $\mathcal{W}(a) \rightrightarrows_{\mathcal{R}} \mathcal{W}(b)$.

PROOF. We consider two cases, one for *ExERS*-rewriting and one for *FExERS*-rewriting.

ExERS-**rewriting** Suppose that $a \rightarrow_{fo(\mathcal{R})/\mathcal{W}} b$ using rewrite rule $(L',R') = \mathcal{C}_P(L,R)$ where P is a pivot set for $(L,R) \in \mathcal{R}$, a context E and first-order valuation ρ. Thus $a =_{\mathcal{W}} E[\overline{\rho}(L')]$ and $b =_{\mathcal{W}} E[\overline{\rho}(R')]$.
Since $E[\overline{\rho}(L')] \rightarrow_{(L',R')} E[\overline{\rho}(R')]$, we conclude that $\mathcal{W}(E[\overline{\rho}(L')]) \rightrightarrows_{(L,R)} \mathcal{W}(E[\overline{\rho}(R')])$ by Lemma 5.15. Also since $a =_{\mathcal{W}} E[\overline{\rho}(L')]$ we know that $\mathcal{W}(a) = \mathcal{W}(E[\overline{\rho}(L')])$, likewise we know that $\mathcal{W}(b) = \mathcal{W}(E[\overline{\rho}(R')])$. Finally, $\mathcal{W}(a) = \mathcal{W}(E[\overline{\rho}(L')]) \rightrightarrows_{(L,R)} \mathcal{W}(E[\overline{\rho}(R')]) = \mathcal{W}(b)$ as desired.

FExERS-**rewriting** Suppose $fo(\mathcal{R})$ is a *FExERS* and that $a \rightarrow_{fo(\mathcal{R})\cup\mathcal{W}} b$. Then if $a \rightarrow_{\mathcal{W}} b$ the result holds trivially. Thus let us assume that $a \rightarrow_{fo(\mathcal{R})} b$ using rewrite rule $(L',R') = \mathcal{C}_P(L,R)$ where P is a pivot set for $(L,R) \in \mathcal{R}$, a context E and first-order valuation ρ. Then $a = E[\overline{\rho}(L')]$ and $b = E[\overline{\rho}(R')]$. Now since $E[\overline{\rho}(L')] \rightarrow_{(L',R')} E[\overline{\rho}(R')]$ then by Lemma 5.15 we may conclude that $\mathcal{W}(a) = \mathcal{W}(E[\overline{\rho}(L')]) \rightrightarrows_{(L,R)} \mathcal{W}(E[\overline{\rho}(R')]) = \mathcal{W}(b)$ as desired.

∎

Since $\rightrightarrows_{\mathcal{R}} \subseteq \twoheadrightarrow_{\mathcal{R}}$, we may replace $\mathcal{W}(a) \rightrightarrows_{\mathcal{R}} \mathcal{W}(b)$ by $\mathcal{W}(a) \twoheadrightarrow_{\mathcal{R}} \mathcal{W}(b)$ in the statement of the Projection Proposition.

DEFINITION 5.17
Let R and S be binary relations defined over sets A and B with $A \subseteq B$, respectively. We say S is *conservative over* R if aSb implies aRb for all $a \in A$.

Noting that $\mathcal{W}(a) = a$ for pure terms a (Definition 3.8(2)) we may conclude.

COROLLARY 5.18 (Conservativity)
Let $\mathcal{R}$ be a $SERS_{dB}$. Then $fo(\mathcal{R})_{\mathcal{W}}$-rewriting is conservative over $\mathcal{R}$-rewriting, that is to say, if $a \xrightarrow{*}_{fo(\mathcal{R})_{\mathcal{W}}} b$ for a, b pure terms, then $a \xrightarrow{*}_{\mathcal{R}} b$.

5.3 Essentially first-order HORS

This last subsection provides a very simple syntactical criterion that can be used to decide if a given higher-order rewrite system can be translated into a full first-order rewrite system (modulo an empty equational theory). In particular, we can check that many higher-order calculi in the literature, such as the lambda calculus, verify this property.

DEFINITION 5.19 (Essentially first-order HORS)
A $SERS_{dB}$ $\mathcal{R}$ is called *essentially first-order* if the first-order version of $\mathcal{R}$, namely $fo(\mathcal{R})_{\mathcal{W}}$, is a *FExERS* for $\mathcal{W}$ a basic substitution calculus.

Recall from Section 3 (Definition 3.9) that an *ExERS* (Explicit Expression Reduction System) $\mathcal{R}$ is a *FExERS* (*Fully* Explicit Expression Reduction System), if the LHS of each rule in $\mathcal{R}$ contains no occurrences of the substitution operator $\bullet[\bullet]$.

DEFINITION 5.20 (fo-condition)
A $SERS_{dB}$ $\mathcal{R}$ satisfies the *fo-condition* if every rewrite rule (L, R) in $\mathcal{R}$ satisfies: for every name X in L let $X_{l_1}, \ldots, X_{l_n}$ be all the X-based metavariables in L, then

1. $l_1 = l_2 \ldots = l_n$ and (the underlying set of) l_1 is $\mathtt{Ba}_{(L,R)}(X)$, and
2. for all $X_k \in R$ we have $|k| \geq |l_1|$.

In the above definition note that $l_1 = l_2 \ldots = l_n$ means that labels $l_1, \ldots, l_n$ must be *identical* (for example $\alpha\beta \neq \beta\alpha$). Also, by Definition 2.10, l_1 is simple, in other words, it does not have repeated elements.

EXAMPLE 5.21
Consider the λ_{dB}-calculus consisting of the sole rule: $app(\lambda X_\alpha, Y_\epsilon) \rightarrow_{\beta_{dB}} X_\alpha[Y_\epsilon]$. The β_{dB}-calculus satisfies the fo-condition. However, the η_{db} rule $\lambda(app(X_\alpha, \underline{1})) \rightarrow X_\epsilon$ does not satisfy the fo-condition: the label of X_α in $\lambda(app(X_\alpha, \underline{1}))$ does not coincide with the binding allowance of X in η_{db} ($\mathtt{Ba}_{(\lambda(app(X_\alpha,\underline{1})),X_\epsilon)}(X) = \emptyset$).

Proposition 5.22 puts forward the importance of the fo-condition. Its proof relies on a close inspection of the Conversion Procedure.

PROPOSITION 5.22
Let $\mathcal{R}$ be a $SERS_{dB}$. Then $\mathcal{R}$ satisfies the fo-condition iff $\mathcal{R}$ is essentially first-order.

Further examples of essentially first-order $SERS_{dB}$ are the $foldl$-rewrite system of Example 4.10 and the natural numbers recursor rewrite system rec of Example 4.11.

Note that many results on higher-order systems (e.g. perpetuality [34], standardization [37]) require *left-linearity* (a metavariable may occur at most once on the LHS of a rewrite rule), and *fully-extendedness or locality* (if a metavariable $X(t_1, \ldots, t_n)$ occurs on the LHS of a rewrite rule then $t_1, \ldots, t_n$ is the list of variables bound above it). The reader may find it interesting to observe that these conditions together seem to imply the fo-condition. A proof of this fact would require either developing the results of this work in the above mentioned HORS or via some suitable translation to the $SERS_{dB}$ formalism, and is left to future work.

Of course, all first-order rewriting systems are essentially first-order $SERS_{dB}$: indeed all metavariables in first-order rewriting systems carry ϵ as label. Hence the latter systems need not be left-linear. Also, an orthogonal $SERS_{dB}$ (Definition 2.15) need not be essentially first-order, the prime example of this fact being the rewrite system consisting of the sole rule η_{dB}. This is summarized in Figure 4.

It seems fair to say that a $SERS_{dB}$ system is essentially first-order if higher-order pattern matching may be reduced to syntactic first-order matching. We claim that essentially first-order $SERS_{dB}$ systems are appropriate for transferring results from first-order systems. A first step towards this claim can be found in [12] where the Standardization Theorem is transferred from (left-linear) first-order rewriting systems to essentially first-order higher-order rewriting systems.

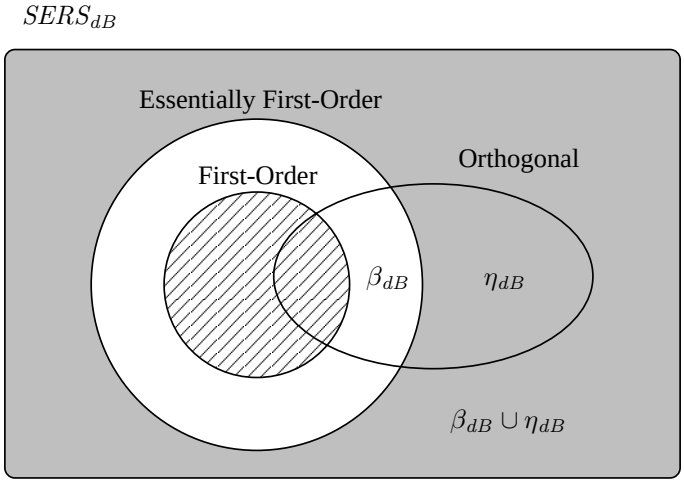

FIGURE 4. Essentially first-order systems

6 Other Higher-order rewriting formalisms

Besides $SERS$ and ERS, many other HOR formalisms have been extensively studied. Two of these are J-W. Klop's CRS [33] and T. Nipkow's HRS [42]. Figure 5 illustrates how the β-rule is represented in each of these formalisms. In this section we relate higher-order rewriting in the HRS formalism and first-order rewriting. Since CRS are a particular case of HRS [50] we shall also relate CRS and first-order rewriting. After a brief overview of the HRS formalism we propose a rather simple two-step conversion procedure, the HRS-conversion procedure, which converts any HRS to a corresponding first-order rewrite system in which rewriting takes place modulo a first-order equational theory. We then extend this procedure with three additional steps in order to specialize it to the case of *pattern* HRS (see below). Some comments on differences between the HRS-conversion procedure and the one developed in Section 4 for $SERS$ are interwoven.

$app(lam([z]X(z)), Y) \rightarrow X(Y)$	(CRS)
$app(lam(\lambda z.x\, z), y) \rightarrow x\, y$	(HRS)
$app(\lambda\alpha.X, Y) \rightarrow X[\alpha \leftarrow Y]$	$(SERS)$

FIGURE 5. The β-rule in various HOR formalisms

As mentioned in the introduction to this paper the reason we have studied the $SERS$ formalism is that we find its notation appealing in that it allows a representation of rewrite rules which is close to the usual informal presentation. This is particularly evident in the representation of the β-rewrite rule. Although no formal relation between $SERS$ and CRS, that we know of, has been established in the literature, we believe that choosing between $SERS$ or CRS is largely a matter of taste. However, HRS differs from both of these formalisms since HRS is a *typed* formalism allowing the representation of rewrite systems of arbitrary types. Indeed, CRS (and $SERS$) are second-order rewrite systems [50] whereas an HRS may be of any order.

An HRS consists of an alphabet $\mathcal{F}$ of symbols (each equipped with a type) and a set of HRS-rewrite rules $\mathcal{R}$. Term formation is specified using the simply typed lambda calculus. *Types* ($\tau, \upsilon, \ldots$)

are built from a non-empty set of base types and the binary type constructor $\rightarrow$. We use $\overline{\tau_i} \rightarrow \tau$ as a shorthand for the type $\tau_1 \rightarrow \tau_2 \rightarrow ... \rightarrow \tau_n \rightarrow \tau$ where τ is a base type and $\rightarrow$ associates to the right. For every type τ we assume a denumerable infinite number of *variables* $x, y, z, \ldots$ of that type. The set of *preterms* of type τ over the signature $\mathcal{F}$ is formed from the following rules:

1. $x : \tau$ if x is a variable of type τ.
2. $f : \tau$ for every *function symbol* $f \in \mathcal{F}$ of type τ.
3. if $x : \tau_1$ and $M : \tau_2$ then $\lambda x.M : \tau_1 \rightarrow \tau_2$.
4. if $M : \tau_1 \rightarrow \tau_2$ and $N : \tau_1$ then $M\,N : \tau_2$.

If $M : \tau$ can be derived from these rules, then we say that M is a preterm of type τ. A preterm of the form $f\,M_1 \ldots M_n$ is sometimes written $f(M_1, \ldots, M_n)$. Also, α-equivalent preterms are identified. A context is a preterm with one occurrence of a hole $\Box$. The *long η-normal form* of a preterm is obtained by repeatedly replacing $C[M]$ by $C[\lambda x.M\,x]$ where M is of type $\tau_1 \rightarrow \tau_2$, x of type τ_1 does no occur free in M, M is not an abstraction and the occurrence of the hole $\Box$ in C is not functional (i.e. it does not occur as the left part of an application). This process is called *restricted η-expansion* due to the latter conditions which guarantee that no β-redexes are created. The *long $\beta\eta$-normal form* of a preterm may be computed by taking the long η-normal form of its β-normal form. A *term* is a preterm in long $\beta\eta$-normal form.

A *rewrite rule* is a pair of terms (L, R) such that:

1. L and R are of the same base type,
2. L is of the form $f(M_1, \ldots, M_n)$, and
3. all free variables[3] in R also occur in L.

A term M satisfies the *pattern condition* if every free variable x in M occurs in the form $x\,M_1 \ldots M_n$ with each M_i η-equivalent (i.e. equal modulo η) to distinct bound variables, for $i \in 1..n$. For example, $\lambda x.y\,x$ satisfies the pattern condition, however all of $\lambda x.y\,x\,x$, $\lambda x.y\,z$, and $\lambda x.y\;cons(zero, x)$ do not. If (L, R) is a rewrite rule and L satisfies the pattern condition, then (L, R) is a *pattern rewrite rule*. A set of pattern rewrite rules is called a *Pattern HRS*, abbreviated PRS. The pattern condition guarantees that the induced rewrite relation is decidable because unification of patterns is decidable [39].

EXAMPLE 6.1
An example of an HRS-rewrite rule (taken from [36]) is:

$$f(\lambda x.y\;cons(zero, x)) \rightarrow f(\lambda x.y\,x).$$

It is an example of a rewrite rule whose LHS is not a pattern. The net effect of its execution is that of replacing all $cons$-headers whose tail is the bound variable x under f with the bound variable x. Note that if there are occurrences of the bound variable x under f which do not occur under the form $cons(zero, x)$, then the rule is not applicable[4].

An example of a pattern rewrite rule is the β-rule of Figure 5. We assume there is only one base type 0 (the type of the 'terms'). The alphabet consists of the two function symbols $app : 0 \rightarrow 0 \rightarrow 0$ and $lam : (0 \rightarrow 0) \rightarrow 0$. The free variable x is of type $0 \rightarrow 0$ and y of type 0. See Figure 6(b) for another example of a pattern rewrite rule.

[3]The notion of free variable is defined as usual [3].
[4]For instance this rule is not applicable to the term $f(\lambda x.g(x, cons(zero, x)))$.

Given that β together with restricted η-expansion is confluent and terminating on the set of preterms [26, 27] every preterm has a unique long $\beta\eta$-normal form. Thus we may restrict the induced rewrite relation to terms without loss of generality.

An *assignment* is a finite mapping from variables to terms of the same type. An assignment κ may be extended to a mapping from terms to terms, with the aid of the usual variable convention of the λ-calculus, as follows:

$$\begin{array}{lcll} f^\kappa & \stackrel{\text{def}}{=} & f & f \in \mathcal{F} \\ x^\kappa & \stackrel{\text{def}}{=} & \kappa(x) & \\ (\lambda x.M)^\kappa & \stackrel{\text{def}}{=} & \lambda x.M^\kappa & \\ (M\,N)^\kappa & \stackrel{\text{def}}{=} & M^\kappa\,N^\kappa & \end{array}$$

Note that this extension of an assignment requires variable renaming in order to avoid unwanted capture of free variables. Thus it is not simple replacement. Also observe that κ does not reduce β-redexes.

Let $M{\downarrow_\beta}$ denote the β-normal form of the preterm M. The rewrite relation induced by an HRS-rewrite rule (L, R), is defined as:

$$C[L^\kappa{\downarrow_\beta}]{\rightarrow_{(L,R)}}\, C[R^\kappa{\downarrow_\beta}]$$

where C is any context.

6.1 Converting HRS

HRS-rewriting and first-order rewriting is related by the *HRS-conversion procedure* and is described in this subsection. Since HRS is a typed rewriting formalism we would require a corresponding typed substitution calculus in order to implement an HRS-rewrite step in a first-order setting. Moreover, since substitution at the metalevel in an HRS is implemented by the full typed lambda calculus the notion of basic substitution calculi (Definition 3.8) would have to be extended to include a $Beta$-rewrite rule $(\lambda M)N{\rightarrow}\ M[cons(N, id)]$. Such substitution calculi have been studied in [29, 30, 49, 51]. Although such extensions of substitution calculi present no major problems, for expository reasons we shall restrict ourselves to the $\lambda\sigma$-calculus implementation of the simply typed lambda calculus.

The HRS-conversion procedure consists of the following two steps:

Step 1. (de Bruijn indices notation)

Let $\bullet_{dB}$ be the function that translates HRS terms into de Bruijn terms: bound variables are replaced by de Bruijn indices and free variables (which in the HRS formalism correspond to the metavariables of the $SERS$ formalism) are left unaltered.

The result of applying **Step 1** to an HRS rewrite rule (L, R) (resp. valuation κ) is a rewrite rule (L_{dB}, R_{dB}) (resp. valuation κ_{dB}) in the HRS_{dB} formalism. The latter formalism is the naïve de Bruijn variant of the HRS formalism and is defined as expected; similar comments apply to notions such as β_{dB}-reduction, η_{dB}-reduction and the corresponding notions of normal form and long normal form. Note that the HRS_{dB} formalism is still a higher-order rewrite formalism, it relies on metalevel substitution and in order to instantiate rewrite rules valuations must do some index adjusting on the metalevel.

The following holds:

$$M \;=\; C[L^\kappa{\downarrow_\beta}] \;\rightarrow_{(L,R)}\; C[R^\kappa{\downarrow_\beta}] \;=\; N \tag{6.1}$$

iff

$$M_{dB} \quad = \quad C_{dB}[L_{dB}^{\kappa_{dB}}\downarrow_{\beta_{dB}}] \quad \rightarrow_{(L_{dB},R_{dB})} \quad C_{dB}[R_{dB}^{\kappa_{dB}}\downarrow_{\beta_{dB}}] \quad = \quad N_{dB}. \tag{6.2}$$

Observe that κ in (6.1) must be careful to rename whenever necessary in order to obtain L^{κ} (and R^{κ}). The de Bruijn valuation κ_{dB} does not constitute a simple replacement operation either since in order to obtain $L_{dB}^{\kappa_{dB}}$ in (6.2) it must do some index adjustment.

See Figure 6 for examples.

Step 2. (pre-cooking [17])

Each free variable x in (L_{dB}, R_{dB}) resulting from applying **Step 1** to (L, R) is replaced by the term $x[\uparrow^k]$ where k is the number of binders above that occurrence of x in (L_{dB}, R_{dB}). The resulting rewrite rule (L_2, R_2)[5] is now a first-order rewrite rule in that the induced rewrite relation is obtained by applying valuations that have become simple replacement; rewriting is now first-order rewriting modulo $\lambda\sigma$. Therefore, item (6.2) in the equivalence mentioned in **Step 1** can now be replaced by:

$$\underbrace{M_{dB} \ =_{\lambda\sigma} \ C_{dB}[L_2^{\kappa_2}]}_{\text{matching phase}} \ \rightarrow_{(L_2,R_2)} \ \underbrace{C_{dB}[R_2^{\kappa_2}] \ =_{\lambda\sigma} \ N_{dB}}_{\text{substitution phase}} \tag{6.3}$$

κ_{dB} and κ_2 share the same underlying assignment. However, κ_{dB} has to do some index adjustment when traversing the structure of a term, whereas κ_2 does no such adjustment, it simply replaces the free variables by their corresponding terms.

Note that since M_{dB} is in β_{dB}-normal form and $\lambda\sigma$ is confluent (on ground terms) we may replace the matching phase in (6.3) by $C_{dB}[L_2^{\kappa_2}] \rightarrowtail_{\lambda\sigma} M_{dB}$ and, similarly, the substitution phase by $C_{dB}[R_2^{\kappa_2}] \rightarrowtail_{\lambda\sigma} N_{dB}$, where the notation $N \rightarrowtail_{\lambda\sigma} N'$ is used to say that N $\lambda\sigma$-rewrites to the $\lambda\sigma$-normal form of N'. This notion is well-defined since $\lambda\sigma$ is weakly normalizing [41, 22] (and confluent as already mentioned).

See Figure 6.

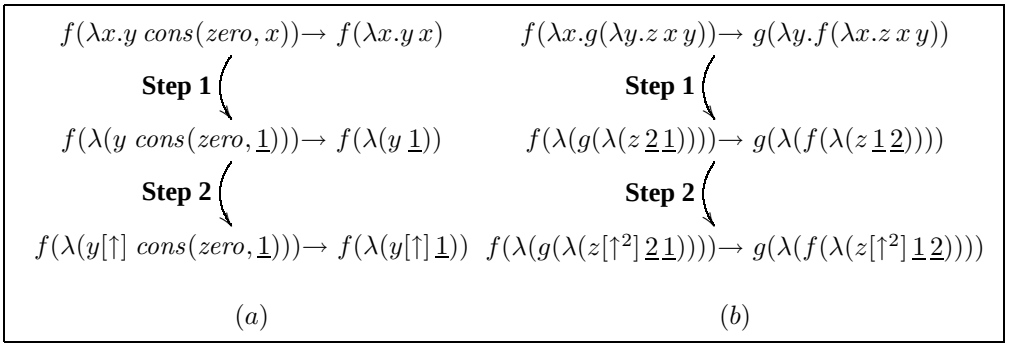

FIGURE 6. Examples of HRS-conversion procedure

[5] We use L_{dB} for the result of applying **Step 1** to L, L_2 for the result of applying steps 1 and 2 to L, . . ., and L_5 for the result of applying steps 1 to 5 to L.

REMARK 6.2
The acute reader may have noted that the Simulation Proposition (Proposition 5.8) for $SERS$ uses rewriting modulo the substitution calculus σ (and not (typed) $\lambda\sigma$ as in (6.3) of **Step 2**) for the substitution phase. This is coherent with two facts: the observation made in [50] that HRS have more 'rewriting power' than CRS, and the fact that CRS and $SERS$ are equivalent formalisms. A similar comment applies to the matching phase in (6.3) of **Step 2**: the HRS formalism allows terms of any order and thus requires higher-order matching whereas $SERS$ is a second-order pattern rewriting formalism which relies on second-order pattern matching.

6.2 Converting PRS

The rewrite system which results from applying steps 1 and 2 is a first-order rewriting system *modulo* a (first-order) equational theory. However, the matching phase (see (6.3) in **Step 2**) could be simplified incrementally in two ways.

- *$Beta$-simplification.* Suppose an application of the form $x[\uparrow^k] a_1 \dots a_n$ occurs in (L_2, R_2) with x a free variable of type $\overline{\tau_i} \to \tau$. Then since κ_{dB} assigns terms in long $\beta_{dB}\eta_{dB}$-normal form to free variables we may assume that $\kappa_2(x) = \lambda \dots \lambda a$ where there are exactly n lambda binders above a. Therefore, we could be tempted to replace the aforementioned application by $x'[cons(a_n, \dots cons(a_1, \uparrow^k))]$ where x' is some free-variable of type τ. This would be done in order to simplify the matching phase in (6.3) of **Step 2** by eliminating the $Beta$-steps: This phase would be replaced by $C_{dB}[L_2^{\kappa_2'}] \rightarrowtail_\sigma M_{dB}$, where $\kappa_2'(x) = a$ (and similarly for the other variables in the domain of κ_2').
  However, this simplification is not possible in general since σ may create $Beta$-redexes. Nevertheless, in the particular case that all the (η_{dB}-normal forms of the) a_is are de Bruijn indices, then only 'trivial' $Beta$-redexes can be created [50, Prop.4.8a] in the reduction from $a[cons(a_n, \dots cons(a_1, \uparrow^k))]$ to its $\lambda\sigma$-normal form. By 'trivial' we mean a $Beta$-redex of the form $a_i\, c$ for some term c in some $\lambda\sigma$-reduct of $a[cons(a_n, \dots cons(a_1, \uparrow^k))]$.
- *Full simplification.* If the matching phase in (6.3) of **Step 2** were replaceable by syntactic identity $M_{dB} = C_{dB}[L_2^{\kappa_2}]$ then we would be in a first-order setting where syntactic matching suffices: the equational theory $\lambda\sigma$ would not be necessary for matching. We shall see that in some cases this is possible (see **Step 4** and **Step 5** below).

As remarked in the $Beta$-simplification entry the full $\lambda\sigma$ is required in the matching phase in order to simulate HRS-rewriting in a first-order setting. However, if we restrict attention to the subclass of *pattern HRS* then $Beta$-simplification is applicable. Thus the PRS-conversion procedure consists of **Step 1** and **Step 2** of the HRS-conversion procedure together with additional steps, namely steps 3, 4 and 5. **Step 3** implements $Beta$-simplification of the matching phase whereas **Step 4** and **Step 5** considers full simplification.

Figure 6(b) presents the PRS-rewrite rule we shall use as an example in this subsection. This rule models binder commutation. Below it, we depict the rules resulting from applying **Step 1** and **Step 2** to it.

Step 3. ($Beta$-simplification)
Let (L_2, R_2) be the result of applying steps 1 and 2 to the PRS-rewrite rule (L, R).
1. Replace all the applications of the form $x[\uparrow^k] a_1 \dots a_n$ in R_2 where x is a free-variable of type $\overline{\tau_i} \to \tau$ by $x'[cons(a_n, \dots cons(a_1, \uparrow^k))]$. The type of x' is τ. This simplification is justified by observing that $(\lambda \dots \lambda a)[\uparrow^k] a_1 \dots a_n =_{\lambda\sigma} a[cons(a_n, \dots cons(a_1, \uparrow^k))]$ where there are exactly n lambda binders above a and $a, a_1, \dots, a_n$ are de Bruijn terms.

2. Replace all the applications of the form $x[\uparrow^k]\, a_1 \dots a_n$ in L_2 (note that $k \geq n$) where x is a free-variable of type $\overline{\tau_i} \rightarrow \tau$ by $x'[cons(a_n \downarrow_{\eta_{dB}}, \dots cons(a_1 \downarrow_{\eta_{dB}}, \uparrow^k))]$. The type of x' is τ. This simplification is justified by observing that $((\lambda \dots \lambda a)[\uparrow^k]\, a_1 \dots a_n)\downarrow_{\lambda\sigma} = (a[cons(a_n\downarrow_{\eta_{dB}}, \dots cons(a_1\downarrow_{\eta_{dB}}, \uparrow^k))])\downarrow_\sigma$ where there are exactly n lambda binders above a, a is in long $\beta_{dB}\eta_{dB}$-normal form and the (η_{dB}-normal forms of the) a_is are de Bruijn indices.

Let (L_3, R_3) be the resulting rule. Then:

$$M \;=\; C[L^\kappa\downarrow_\beta] \;\rightarrow_{(L,R)}\; C[R^\kappa\downarrow_\beta] \;=\; N \tag{6.4}$$

iff

$$M_{dB} \;\;{}_\sigma\!\leftarrowtail\;\; C_{dB}[L_3^{\kappa_3}] \;\rightarrow_{(L_3,R_3)}\; C_{dB}[R_3^{\kappa_3}] \;\rightarrowtail_{\lambda\sigma}\; N_{dB} \tag{6.5}$$

κ_3 results from κ_2 as follows: if $x : \overline{\tau_i} \rightarrow \tau$ is in the domain of κ_2, and thus $\kappa_2(x) = \lambda \dots \lambda a$ where there are n lambda binders above a, then in that case we define $\kappa_3(x') = a$.

Figure 7 applies **Step 3** to the last rule of Figure 6(b).

$$f(\lambda(g(\lambda(z[\uparrow^2]\,\underline{2}\,\underline{1}))) \rightarrow\; g(\lambda(f(\lambda(z[\uparrow^2]\,\underline{1}\,\underline{2}))))$$

Step 3

$$f(\lambda(g(\lambda(z'[cons(\underline{1}, cons(\underline{2}, \uparrow^2))])))) \rightarrow\; g(\lambda(f(\lambda(z'[cons(\underline{2}, cons(\underline{1}, \uparrow^2))])))))$$

Step 4

$$f(\lambda(g(\lambda z'))) \rightarrow\; g(\lambda(f(\lambda(z'[cons(\underline{2}, cons(\underline{1}, \uparrow^2))])))))$$

FIGURE 7. Example of HRS-conversion procedure (cont.)

Step 4. (Naïve full-simplification)

For every term of the form $x'[cons(a_n, \dots cons(a_1, \uparrow^k))]$ in L_3 where the a_is are de Bruijn indices, if it is of the form $x'[cons(\underline{1}, \dots cons(\underline{k}, \uparrow^k))]$ then replace it by the free variable x'. In other words, throw away the substitution $[cons(\underline{1}, \dots cons(\underline{k}, \uparrow^k))]$ since it behaves as the identity substitution[6]. Let (L_4, R_4) be the resulting rewrite rule (note that $R_4 = R_3$). If no explicit substitutions in L_4 remain then we may replace the matching phase in (6.5) of **Step 3** by syntactical identity:

$$M_{dB} \;=\; C_{dB}[L_4^{\kappa_4}] \;\rightarrow_{(L_4,R_4)}\; C_{dB}[R_4^{\kappa_4}] \;\rightarrowtail_{\lambda\sigma}\; N_{dB} \tag{6.6}$$

where $\kappa_4 = \kappa_3$.

See Figure 7 for an example.

Let $\mathcal{R}$ be a PRS and let $\mathcal{R}_4$ be the result of applying the PRS-conversion procedure to $\mathcal{R}$. We could say $\mathcal{R}$ is *essentially first-order* if there are no explicit substitution operators in the LHSs of the rewrite rules in $\mathcal{R}_4$. However, this is not fully convincing. Consider the following two PRS-rewrite rules:

[6] In the sense that for any pure term a and for all $k > 0$ (we assume $\uparrow^0 = id$): $a =_\sigma a[id] =_\sigma a[cons(\underline{1}, \dots cons(\underline{k}, \uparrow^k))]$.

$$f(\lambda x.g(\lambda y.z\,x\,y)) \quad \rightarrow_{c_1} \quad g(\lambda y.f(\lambda x.z\,x\,y))$$

$$f(\lambda x.g(\lambda y.z\,y\,x)) \quad \rightarrow_{c_2} \quad g(\lambda y.f(\lambda x.z\,y\,x)).$$

They both induce the same rewrite relation and differ only in the way the free variable z is applied to the bound variables x and y. However, the first rule satisfies the definition of essentially first-order given above but the second does not. Indeed, steps 1 to 4 applied to the rule c_2 yields:

$$f(\lambda(g(\lambda(z'[cons(\underline{2}, cons(\underline{1}, \uparrow^2))])))) \rightarrow g(\lambda(f(\lambda(z'[cons(\underline{1}, cons(\underline{2}, \uparrow^2))])))$$

as the term $z'[cons(\underline{2}, cons(\underline{1}, \uparrow^2))]$ cannot be replaced by z' using **Step 4**.

We may observe that the free variable z in the LHS of c_1 and c_2 is applied to *all* the bound variables above it. Therefore, from the point of view of the induced rewrite relation, the order in which they are applied is irrelevant. Indeed, if we were to rearrange $z\,y\,x$ on the LHS of c_2 to $z\,x\,y$ (and apply such a transformation also to the RHS of c_2), then we would obtain c_1. And, as already observed, the induced rewrite relation remains unaltered. As a matter of fact, the $SERS$-Conversion Procedure does exactly this: the user fixes an order on the indices representing bound variables by selecting a pivot metavariable.

As a consequence, we shall add a new step, **Step 5**, to the PRS-Conversion Procedure in order to attain a more convincing notion of essentially first-order PRS.

Step 5. (Full-simplification)

Let x' be a variable in L_4 occurring m times and suppose that every occurrence i of x' with $1 \leq i \leq m$ is embraced by a substitution of the form $x'_i[cons(a^i_k, \ldots cons(a^i_1, \uparrow^k))]$ where $k > 0$ is the number of binders above x'_i and the (η_{dB}-normal forms of) $a^i_k, \ldots, a^i_1$s are de Bruijn indices. Note that:

- the simplification in **Step 4** was surely not applied to one of these occurrences for then there would be no substitution embracing it ($k > 0$ would not be possible); therefore none of the $x'_i[cons(a^i_k, \ldots cons(a^i_1, \uparrow^k))]$ is of the form $x'_i[cons(\underline{1}, \ldots cons(\underline{k}, \uparrow^k))]$,
- the number of binders above all the occurrences of x' is the same and equal to k, and
- due to the pattern-condition, for every $1 \leq i \leq m$ the indices a^i_j with $1 \leq j \leq k$ are distinct bound indices (i.e. $(a^i_k, \ldots, a^i_1)$ is a permutation of $(1, \ldots, k)$).

Pick any occurrence $x'_i[cons(a^i_k, \ldots cons(a^i_1, \uparrow^k))]$, the *pivot occurrence of* x'. Let π be the permutation of $(a^i_k, \ldots, a^i_1)$ such that $\pi(a^i_k, \ldots, a^i_1) = (1, \ldots, k)$.

1. replace $x'_i[cons(a^i_k, \ldots cons(a^i_1, \uparrow^k))]$ by x'_i,
2. replace $x'_j[cons(a^j_k, \ldots cons(a^j_1, \uparrow^k))]$ for $j \neq i$ by $x'_i[cons(b^j_k, \ldots cons(b^j_1, \uparrow^k))]$ where $(b^j_k, \ldots, b^j_1) = \pi(a^j_k, \ldots, a^j_1)$, and
3. replace all substitutions of the form $x'_i[cons(c_k, \ldots cons(c_1, \uparrow^n))]$ in R_4 (note that in R_4 we could have $k \neq n$) by $x'_i[cons(d_k, \ldots cons(d_1, \uparrow^n))]$ where $(d_k, \ldots, d_1) = \pi(c_k, \ldots, c_1)$.

As the reader may note the rewrite relations induced by (L_4, R_4) and (L_5, R_5) are identical.

Let us now define a PRS $\mathcal{R}$ to be *essentially first-order* if there are no explicit substitution operators in the LHSs of the rewrite rules in $\mathcal{R}_5$, where $\mathcal{R}_5$ is the system resulting from applying steps 1 to 5 to $\mathcal{R}$. This time, applying steps 1 to 5 to the rule c_2 yields:

$$f(\lambda(g(\lambda z'))) \rightarrow g(\lambda(f(\lambda(z'[cons(\underline{2}, cons(\underline{1}, \uparrow^2))])))).$$

The permutation π in this example is the one that interchanges the first and second elements of $(2, 1)$: $\pi(2, 1) = (1, 2)$. Thus we may correctly deduce that c_2 is an essentially first-order PRS.

7 Conclusions

We have presented an encoding of higher-order term rewriting systems with indices into first-order rewriting systems modulo an equational theory. This equational theory models the substitution process. The encoding has furthermore allowed us to identify in a simple syntactical manner, via the so-called *fo-condition*, a class of HORS that are fully first-order in that they may be encoded as first-order rewriting systems modulo an empty equational theory. This amounts to incorporating, into the first-order notion of reduction, not only the computation of substitutions but also the higher-order (pattern) matching process. It is fair to say that a higher-order rewrite system satisfying this condition requires a simple matching process, in contrast to those that do not satisfy this condition (such as the $\lambda\beta\eta$-calculus). Other syntactical restrictions, such as linearity and locality, imposed on higher-order rewriting systems (see for example [31, 37]) in order to reason about their properties are closely related to the *fo-condition*.

Moreover, this encoding has been achieved by working with a general presentation of substitution calculi rather than dealing with some particular substitution calculus. Any calculus of explicit substitutions satisfying this general presentation based on macros will serve its purpose, namely that of modelling matching and substitution.

Some further research directions are summarized below:

- As already mentioned, the encoding opens up the possibility of transferring results concerning confluence, termination, completion, evaluation strategies, implementation techniques, etc. from the first-order framework to the higher-order framework. A first step in this direction is studied in [38, 12] where the standardization property is lifted form first-order to higher-order rewriting. Thus, the translation proposed in this paper to encode higher-order rewriting could provide a new means for studying properties of higher-order rewriting through corresponding results in the first-order setting.
- Given a $SERS_{dB}$ $\mathcal{R}$ note that the LHSs of rules in $fo(\mathcal{R})$ may contain occurrences of the substitution operator (pattern substitutions). It would be interesting to deal with pattern substitutions and 'regular' term substitutions (those arising from the conversion of the de Bruijn metasubstitution operator $\bullet[\bullet]$) as different substitution operators at the object-level. This would neatly separate the explicit matching computation from that of the usual substitution replacing terms for variables.
- Given a $SERS_{dB}$ $\mathcal{R}$ which enjoys termination, what are the abstract properties to be imposed on its first-order version $fo(\mathcal{R})_{\mathcal{W}}$ in order for it to be terminating too? This point concerns both the basic substitution calculus $\mathcal{W}$ and the substitutions generated by the rules of $\mathcal{R}$. A first step in this direction is done in [7] for the explicit version of CRS with names, where the substitution calculus is $\mathtt{x}$.

References

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

[2] T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, **236**, 133–178, 2000.

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

[4] Z.-El-Abidine Benaissa, D. Briaud, P. Lescanne, and J. Rouyer-Degli. $\lambda\upsilon$, a calculus of explicit substitutions which preserves strong normalisation. *Journal of Functional Programming*, **6**, 699–722, 1996.

[5] L. Bachmair and N. Dershowitz. Critical pair criteria for completion. *Journal of Symbolic Computation*, **6**, 1–18, 1988.

[6] R. Bloo. *Preservation of Termination for Explicit Substitution*. PhD thesis, Eindhoven University of Technology, 1997.

[7] R. Bloo and K. Rose. Preservation of strong normalization in named lambda calculi with explicit substitution and garbage collection. In *Computing Science in the Netherlands*, pp. 62–72. Netherlands Computer Science Research Foundation, 1995.

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

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

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

[11] E. Bonelli, D. Kesner, and A. Ríos. de Bruijn indices for Metaterms. *Journal of Logic and Computation*, **15**, 857–902, 2005.

[12] E. Bonelli. A normalization result for higher-order calculi with explicit substitutions. In *Foundations of Software Science and Computation Structures*, A. Gordon, ed., pp. 153–168. Vol. 2626 of *Lecture Notes in Computer Science*, Springer-Verlag, 2003.

[13] E. Bonelli. *Substitutions explicites et réécriture de termes*. Thèse de doctorat, Université Paris XI, Orsay, November 2001.

[14] D. Briaud. An explicit *eta* rewrite rule. In *Proceedings of the 2nd International Conference of Typed Lambda Calculus and Applications*, M. Dezani-Ciancaglini and G. Plotkin, eds, pp. 94–108. Volume 902 of *Lecture Notes in Computer Science*, Springer-Verlag, 1995.

[15] N. Dershowitz. Orderings for term rewriting systems. *Theoretical Computer Science*, **17**, 279–301, 1982.

[16] R. David and B. Guillaume. The λ_l-calculus. In *Proceedings of the 2nd Workshop on Explicit Substitutions: Theory and Applications to Programs and Proofs*, D. Kesner, ed., pp. 2–13, July 1999.

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

[18] G. Dowek, T. Hardin, and C. Kirchner. Theorem proving modulo. Technical Report RR 3400, INRIA, 1998.

[19] G. Dowek, T. Hardin, and C. Kirchner. Unification via explicit substitutions: The case of higher-order patterns. Technical Report RR3591, INRIA, December 1998.

[20] G. Dowek, T. Hardin, and C. Kirchner. Higher-order unification via explicit substitutions. *Information and Computation*, **157**, 183–235, 2000.

[21] G. Dowek, T. Hardin, and C. Kirchner. Hol-lambda-sigma: an intentional first-order expression of higher-order logic. *Mathematical Structures in Computer Science*, **11**, 1–25, 2001.

[22] J. Goubault-Larrecq. A proof of weak termination of typed lambda sigma-calculi. In *Proceedings of the International Workshop Types for Proofs and Programs*, T. Altenkirch, W. Naraschewski, and B. Reus, eds, pp. 134–151. Volume 1512 of *Lecture Notes in Computer Science*, Springer-Verlag, 1998.

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

[24] T. Hardin. η-reduction for explicit substitutions. In *Algebraic and Logic Programming'92*, number 632 in *Lecture Notes in Computer Science*, 1992.

[25] T. Hardin and J.-J. Lévy. A confluent calculus of substitutions. In *France-Japan Artificial Intelligence and Computer Science Symposium*, Izu (Japan), 1989.

[26] C. B. Jay and N. Ghani. The virtues of eta-expansion. Technical Report ECS-LFCS-92-243, LFCS, University of Edimburgh, 1992.

[27] C. B. Jay and N. Ghani. The Virtues of Eta-expansion. *Journal of Functional Programming*, **5**, 135–154, 1995.

[28] J.-P. Jouannaud and A. Rubio. The higher-order recursive path ordering. In *Fourteenth Annual IEEE Symposium on Logic in Computer Science*, Trento, Italy, 1999.

[29] D. Kesner. Confluence properties of extensional and non-extensional λ-calculi with explicit substitutions. In *7th International Conference on Rewriting Techniques and Applications*, New Brunswick, NJ, USA. H. Ganzinger, ed., pp. 184–199. Volume 1103 of *Lecture Notes in Computer Science*, Springer-Verlag, 1996.

[30] D. Kesner. Confluence of extensional and non-extensional lambda-calculi with explicit substitutions. *Theoretical Computer Science*, **238**, 183–220, 2000.

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

[32] S. Kamin and J.-J. Lévy. Attempts for generalizing the recursive path orderings. Handwritten paper, University of Illinois, 1980.

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

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

[35] F. Kamareddine and A. Ríos. A λ-calculus à la de Bruijn with explicit substitutions. In *Proceedings of the 7th International Symposium on Proceedings of the International Symposium on Programming Language Implementation and Logic Programming*, D. Swierstra and M. Hermenegildo, eds, pp. 45–62. Volume 982 of *Lecture Notes in Computer Science*, Springer-Verlag, September 1995.

[36] J.-W. Klop, V. van Oostrom, and F. van Raamsdonk. Combinatory reduction systems: introduction and survey. *Theoretical Computer Science*, **121**, 279–308, 1993.

[37] P.-A. Melliès. *Description Abstraite des Systèmes de Réécriture*. PhD thesis, Université Paris VII, 1996.

[38] P.-A. Melliès. Axiomatic rewriting theory II: the lambda-sigma calculus enjoys finite normalisation cones. *Journal of Logic and Computation*, **10**, 461–487, 2000.

[39] D. Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. *Journal of Logic and Computation*, **1**, 497–536, 1991.

[40] C. Muñoz. A left-linear variant of $\lambda\sigma$. In *Proceedings of the 6th International Conference on Algebraic and Logic Programming (ALP'97)*, M. Hanus, J. Heering, and K. Meinke, eds. Volume 1298 in *Lecture Notes in Computer Science*, Springer-Verlag, 1997.

[41] C. Muñoz. Dependent types and explicit substitutions: a meta-theoretical development. *Mathematical Structures in Computer Science*, **11**, 2001.

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

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

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

[45] A. Ríos. *Contribution à l'étude des λ-calculus avec substitutions explicites*. Thèse de doctorat, Université de Paris VII, 1993.

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

[47] M.-O. Stehr. CINNI - a generic calculus of explicit substitutions and its application to λ-, ς- and π-Calculi. In *Electronic Notes in Theoretical Computer Science*, volume 36. K. Futatsugi, ed. Elsevier, 2001.

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

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

[50] V. van Oostrom and F. van Raamsdonk. Comparing combinatory reduction systems and higher-order rewrite systems. In *1st International Workshop on Higher-Order Algebra, Logic, and Term Rewriting*, J. Heering, K. Meinke, B. Möller, and T. Nipkow, eds, pp. 276–304. Volume 816 of *Lecture Notes in Computer Science*, Springer-Verlag, 1994. Selected Papers.

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

[52] H. Zantema. Termination of term rewriting by semantic labelling. *Fundamenta Informaticae*, **24**, 89–105, 1995.

Appendix

A On pivot selection

It is clear that $\mathcal{C}_P(L,R)$ and $\mathcal{C}_Q(L,R)$ shall not be identical. Nevertheless, the rewrite relation generated by both of these converted rewrite rules is identical.

Before proving this proposition, let us consider a rewrite rule (L,R) and let $X_{l_1},\ldots,X_{l_n}$ be all the X-based meta-variables in (L,R) with $\mathtt{Ba}_{(L,R)}(X) \neq \emptyset$. Let X_{l_1} and X_{l_2} be two possible X-based pivots for (L,R). Note that we must

have either $X_{l_1}, X_{l_2} \in L$, or $X_{l_1}, X_{l_2} \in R$ (in which case $|k| > |l_1|$ and $|k| > |l_2|$ for all $X_k \in L$). Also, we have $|l_1| = |l_2|$, a fact that shall be made use of freely below.

Let us consider two different conversions (a) and (b) as dictated by Definition 4.7 taking any metavariable X_{l_i} for $1 \leq i \leq n$ and yielding a first-order term:

$$\begin{array}{lll} \text{(a)} & X_{l_i} & \rightsquigarrow \quad X[cons(a^i_1, \ldots, a^i_{|l_1|}, shift^{|l_i|+|l_1 \setminus \mathtt{Ba}_{(L,R)}(X)|})] \\ & & \text{and} \\ \text{(b)} & X_{l_i} & \rightsquigarrow \quad X[cons(b^i_1, \ldots, b^i_{|l_2|}, shift^{|l_i|+|l_2 \setminus \mathtt{Ba}_{(L,R)}(X)|})]. \end{array}$$

Note that clause 1 of Definition 4.7 does not present itself since the case of interest is when $\mathtt{Ba}_{(L,R)}(X) \neq \emptyset$.

The first translation (a) corresponds to the conversion dictated assuming X_{l_1} as the pivot, while the second translation (b) assumes that X_{l_2} is the pivot.

On an informal account, the substitution $cons(a^i_1, \ldots, a^i_{|l_1|}, shift^{|l_i|+|l_1 \setminus \mathtt{Ba}_{(L,R)}(X)|})$ may be seen as representing a function f_i from indices to indices (hence assuming X is only instantiated with indices). Likewise, $cons(b^i_1, \ldots, b^i_{|l_2|}, shift^{|l_i|+|l_2 \setminus \mathtt{Ba}_{(L,R)}(X)|})$ represents a function g_i. We shall therefore be intersted in finding a function h which may be represented by a pattern substitution such that $f_i = g_i \circ h$. We shall see that the pattern substitution $cons(c_1, \ldots, c_{|l_1|}, shift^{|l_1|})$ defined below satisfies this requirement. Define the following indices c_j for all $1 \leq j \leq |l_1|$:

$$c_j = \begin{cases} a^2_j & \text{if } \mathtt{at}(l_1, j) \in \mathtt{Ba}_{(L,R)}(X) \\ \mathtt{pos}(a^2_j, b^1_1 \ldots b^1_{|l_1|}) & \text{otherwise.} \end{cases} \tag{A.1}$$

REMARK A.1
Note that the second clause of the definition of c_j is defined. Indeed, if $\mathtt{at}(l_1, j) \notin \mathtt{Ba}_{(L,R)}(X)$ then $a^2_j = |l_2| + 1 + \mathtt{Sh}(X_{l_1}, j)$ and since $|l_1| = |l_2|$, l_1 and l_2 are simple labels and $\mathtt{Ba}_{(L,R)}(X) \neq \emptyset$ both l_1 and l_2 have the same number of o-metavariables not included in $\mathtt{Ba}_{(L,R)}(X)$. Thus there exists $j' \in 1..|l_1|$ such that $b^1_{j'} = |l_1| + 1 + \mathtt{Sh}(X_{l_2}, j')$ with $\mathtt{Sh}(X_{l_2}, j') = \mathtt{Sh}(X_{l_1}, j)$, and hence $a^2_j = b^1_{j'}$.

The relation between the two translations (a) and (b) given above can be summarized by the following result:

LEMMA A.2
Let n be the number of X-based metavariables in (L, R) and let X_{l_1} and X_{l_2} be two distinct pivots for (L, R). Let $h \geq 0$ and $1 \leq i \leq n$. Take any first-order valuation ρ and indices a^i_j $(1 \leq j \leq |l_1|)$ and b^i_j $(1 \leq j \leq |l_2|)$ as indicated above in the translations (a) and (b). Then

$$\begin{array}{ll} & (\rho X)[lift^h(s_{a^i})] \\ =_{\mathcal{W}} & (\rho X)[lift^h(s)][lift^h(s_{b^i})] \end{array}$$

where

- $s = cons(c_1, \ldots, c_{|l_1|}, shift^{|l_1|})$ is defined in equation A.1,
- $s_{a^i} = cons(a^i_1, \ldots, a^i_{|l_1|}, shift^{|l_i|+|l_1 \setminus \mathtt{Ba}_{(L,R)}(X)|})$ and
- $s_{b^i} = cons(b^i_1, \ldots, b^i_{|l_2|}, shift^{|l_i|+|l_2 \setminus \mathtt{Ba}_{(L,R)}(X)|})$.

PROOF. We proceed by induction on $\mathcal{W}(\rho X)$.

- $\mathcal{W}(\rho X) = \underline{j}$. We consider three subcases:
  - $j \leq h$. Then Lemma 3.15 (1) allows us to conclude this case.
  - $h < j \leq |l_1| + h$. Then

$$\begin{array}{lll} LHS & =_{\mathcal{W}} & \underline{j-h}[s_{a^i}][shift]^h =_{\mathcal{W}} a^i_{j-h} + h \\ RHS & =_{\mathcal{W}} & c_{j-h}[s_{b^i}][shift]^h. \end{array}$$

    We shall consider two further cases. Recall that X_{l_1} is an X-based pivot for conversion (a) and X_{l_2} is an X-based pivot for conversion (b).

    1. $i = 1$. Suppose
       - $\mathtt{at}(l_1, j-h) = \beta \in \mathtt{Ba}_{(L,R)}(X)$. Then

$$\begin{array}{lll} RHS & = & a^2_{j-h}[s_{b^1}][shift]^h \\ & = & \mathtt{pos}(\beta, l_2)[s_{b^1}][shift]^h \\ & =_{\mathcal{W}} & b^1_{\mathtt{pos}(\beta, l_2)} + h \\ & = & \underline{\mathtt{pos}(\beta, l_1) + h} \\ & = & \underline{j-h+h} \\ & =_{\mathcal{W}} & LHS \end{array}$$

Recall that all labels are simple (no repeated elements).

∗ $\mathtt{at}(l_1, j-h) \notin \mathtt{Ba}_{(L,R)}(X)$.

$$\begin{array}{lll} & RHS & \\ = & \mathtt{pos}(a^2_{j-h}, b^1_1..b^1_{|l_1|})[s_{b^1}][shift]^h & \\ = & \mathtt{pos}(|l_2|+1+\mathtt{Sh}(X_{l_1}, j-h), b^1_1..b^1_{|l_1|})[s_{b^1}][shift]^h & \\ =_{\mathcal{W}} & |l_2|+1+\mathtt{Sh}(X_{l_1}, j-h)+h & \\ = & \underline{|l_1|+1+\mathtt{Sh}(X_{l_1}, j-h)+h} & \\ =_{\mathcal{W}} & LHS & \end{array}$$

2. $i \geq 2$. Suppose

∗ $\mathtt{at}(l_1, j-h) = \beta \in \mathtt{Ba}_{(L,R)}(X)$. Then

$$\begin{array}{lll} RHS & = & a^2_{j-h}[s_{b^i}][shift]^h \\ & = & \mathtt{pos}(\beta, l_2)[s_{b^i}][shift]^h \\ & =_{\mathcal{W}} & b^i_{\mathtt{pos}(\beta,l_2)}+h \\ & = & \mathtt{pos}(\beta, l_i)+h \\ & =_{\mathcal{W}} & LHS \end{array}$$

The last step follows from case 2(b) of Definition 4.7 since X_{l_i} is a not the pivot metavariable occurrence for conversion (a), for any $i \geq 2$.

∗ $\mathtt{at}(l_1, j-h) \notin \mathtt{Ba}_{(L,R)}(X)$. Then $LHS =_{\mathcal{W}} |l_i|+1+\mathtt{Sh}(X_{l_1}, j-h)+h$. Also,

$$\begin{array}{lll} & RHS & \\ = & \mathtt{pos}(a^2_{j-h}, b^1_1..b^1_{|l_1|})[s_{b^i}][shift]^h & \\ = & \mathtt{pos}(|l_2|+1+\mathtt{Sh}(X_{l_1}, j-h), b^1_1..b^1_{|l_1|})[s_{b^i}][shift]^h. & \end{array}$$

Now since $|l_2|+1+\mathtt{Sh}(X_{l_1}, j-h) > |l_1|$ and $|l_1| = |l_2|$ there must be some $1 \leq j' \leq |l_1|$ such that $b^1_{j'} = |l_1|+1+\mathtt{Sh}(X_{l_2}, j')$ (and hence $\mathtt{at}(l_2, j') \notin \mathtt{Ba}_{(L,R)}(X)$) with $\mathtt{Sh}(X_{l_1}, j-h) = \mathtt{Sh}(X_{l_2}, j')$ (see Remark A.1). Thus we may continue as follows:

$$\begin{array}{lll} & \mathtt{pos}(|l_2|+1+\mathtt{Sh}(X_{l_1}, j-h), b^1_1..b^1_{|l_1|})[s_{b^i}][shift]^h & \\ = & j'[s_{b^i}][shift]^h & \\ =_{\mathcal{W}} & b^i_{j'}+h & \\ = & \underline{|l_i|+1+\mathtt{Sh}(X_{l_2}, j')+h}. & \end{array}$$

The last equality follows from the fact that $\mathtt{at}(l_2, j') \notin \mathtt{Ba}_{(L,R)}(X)$.

– $j > |l_1|+h$. Then

$$\begin{array}{lll} & LHS & \\ =_{\mathcal{W}} & \underline{j-h}[s_{a^i}][shift]^h & \\ =_{\mathcal{W}} & \underline{j-h-|l_1|+|l_i|+|l_1 \setminus \mathtt{Ba}_{(L,R)}(X)|+h} & \\ = & \underline{j-|l_1|+|l_i|+|l_1 \setminus \mathtt{Ba}_{(L,R)}(X)|} & \\ = & \underline{j-|l_2|+|l_i|+|l_2 \setminus \mathtt{Ba}_{(L,R)}(X)|} & \\ =_{\mathcal{W}} & \underline{j}[lift^h(s_{b^i})] & \\ =_{\mathcal{W}} & \underline{j-h}[s][shift]^h[lift^h(s_{b^i})] & \\ =_{\mathcal{W}} & RHS & \end{array}$$

- $\mathcal{W}(\rho X) = f(d_1, \ldots, d_n)$. We use the induction hypothesis.
- $\mathcal{W}(\rho X) = \xi(d_1, \ldots, d_n)$. Then by the induction hypothesis we have

$$d_j[lift^{h+1}(s_{a^i})] =_{\mathcal{W}} d_j[lift^{h+1}(s)][lift^{h+1}(s_{b^i})]$$

for all $j \in 1..n$ which allows us to conclude the case.

∎

PROOF. [of Proposition 4.14] Let $(L_1, R_1) \stackrel{\text{def}}{=} \mathcal{C}_P(L, R)$ and $(L_2, R_2) \stackrel{\text{def}}{=} \mathcal{C}_Q(L, R)$. Suppose that $a \to_{(L_1,R_1)} b$. Then there exists a context E and a first-order valuation ρ such that $a =_{\mathcal{W}} E[\rho(L_1)]$ and $b =_{\mathcal{W}} E[\rho(R_1)]$.

For all $X \in \text{FMVAR}(L)$ define the first-order valuation η as: $\overline{\eta}X \stackrel{\text{def}}{=} \rho(X)[s]$ where $s = cons(c_1, \ldots, c_{|l_1|}, shift^{|l_1|})$ and the $c_i s$ are defined in equation A.1. Consider now an occurrence of a metavariable $X_{l_i} \in (L, R)$ where $\{X_{l_1}, \ldots, X_{l_n}\}$ are all the X-based metavariables in (L, R).

- If $\mathtt{Ba}_{(L,R)}(X) = \emptyset$ then both conversions shall convert X_{l_i} to the term $X[shift^{|l_i|}]$. This case needs no further consideration.
- If $\mathtt{Ba}_{(L,R)}(X) \neq \emptyset$ then each conversion shall convert X_{l_i} to (possibly) different terms: $X[s_{a^i}]$ on the one hand, and on the other $X[s_{b^i}]$. Here we may apply Lemma A.2 and obtain:
$$(\rho X)[s_{a^i}] =_{\mathcal{W}} (\eta X)[s_{b^i}].$$
If conversion (a) deployed the identity optimization then
$$s_{a^i} = cons(1, \ldots, |l_1|, shift^{|l_1|})$$
and X_{l_i} is converted to X. We may then make use of the fact that
$$\rho X =_{\mathcal{W}} (\rho X)[cons(1, \ldots, |l_i|, shift^{|l_i|})]$$
and resort to Lemma A.2 as above. A similar observation holds for the (b) conversion.

Therefore we may obtain $\rho(L_1) =_{\mathcal{W}} \eta(L_2)$ and $\rho(R_1) =_{\mathcal{W}} \eta(R_2)$, so that $a =_{\mathcal{W}} E[\rho(L_1)] =_{\mathcal{W}} E[\eta(L_2)]$ and $b =_{\mathcal{W}} E[\rho(R_1)] =_{\mathcal{W}} E[\eta(R_2)]$, i.e. $a \to_{(L_2,R_2)} b$.

∎

Valuation Conversion

LEMMA A.3

Consider a $SERS_{dB}$-rewrite rule (L, R), metavariables $X_l, X_k \in (L, R)$, and a valuation κ valid for (L, R). For all $i \geq 0$, if

1. $\kappa X_l = D[a]$ for some pure context D having binder path number i,
2. $\mathsf{Value}^i(l, a) = \mathsf{Value}^i(k, b)$, and
3. the binding allowance of X in (L, R) is the empty set (i.e. $\mathtt{Ba}_{(L,R)}(X) = \emptyset$),

then $\mathcal{D}_i^{|l|}(a) = \mathcal{D}_i^{|k|}(b)$.

PROOF. By induction on a.

- $a = \underline{n}$. We consider the following cases:
  - $n \leq i + |l|$. Then $\mathcal{D}_i^{|l|}(a) = \underline{n}$. If $n \leq i$ then since $\mathsf{Value}^i(l, a) = \underline{n} = \mathsf{Value}^i(k, b)$, we have $b = \underline{n}$ and the result holds.
    Otherwise, if $i < n \leq i + |l|$ then since by Hypothesis 2 we have $\mathsf{Value}^i(l, \underline{n}) = \mathtt{at}(l, n - i) = \mathsf{Value}^i(k, b)$ we must have $b = \underline{m}$ with $i < m \leq i + |k|$ and $\mathtt{at}(l, n - i) = \mathtt{at}(k, m - i)$. But by Hypothesis 3 there must be some $X_{l'}$ in (L, R) such that $\mathtt{at}(l, n - i) \notin l'$, and hence $\mathsf{Value}(l', \kappa X_{l'}) \neq \mathsf{Value}(l, \kappa X_l)$ by the definition of the value function Definition 2.13 (since $\mathtt{at}(l, n-i)$ occurs in $\mathsf{Value}(l, \kappa X_l)$ but $\mathtt{at}(l, n-i)$ does not occur in $\mathsf{Value}(l', \kappa X_{l'})$), contradicting the assumption that κ is valid.
  - $n > i + |l|$. Then $\mathcal{D}_i^{|l|}(a) = \underline{n - |l|}$. Also, since $\mathsf{Value}^i(l, a) = x_{n-i-|l|} = \mathsf{Value}^i(k, b)$, we have $b = \underline{m}$ with $m > |k| + i$ and $n - |l| - i = m - |k| - i$. Then $\mathcal{D}_i^{|k|}(b) = \underline{m - |k|}$ and the result holds.
- $a = f(a_1, \ldots, a_n)$. Then $\mathcal{D}_i^{|l|}(a) = f(\mathcal{D}_i^{|l|}(a_1), \ldots, \mathcal{D}_i^{|l|}(a_n))$.
  Now by Hypothesis 2 we have that $b = f(b_1, \ldots, b_n)$ with $\mathsf{Value}^i(l, a_j) = \mathsf{Value}^i(k, b_j)$ for all $1 \leq j \leq n$. Then the induction hypothesis yields $\mathcal{D}_i^{|l|}(a_j) = \mathcal{D}_i^{|k|}(b_j)$ for $j \in 1..n$ and we may conclude the case by Definition 5.1.
- $a = \xi(a_1, \ldots, a_n)$. Then $\mathcal{D}_i^{|l|}(a) = \xi(\mathcal{D}_{i+1}^{|l|}(a_1), \ldots, \mathcal{D}_{i+1}^{|l|}(a_n))$.
  Now by Hypothesis 2 we have that $b = \xi(b_1, \ldots, b_n)$ with $\mathsf{Value}^{i+1}(l, a_j) = \mathsf{Value}^{i+1}(k, b_j)$ for all $1 \leq j \leq n$. Then the induction hypothesis concludes the case.

∎

PROOF. [of Lemma 5.7] Both items are proved by induction on A.

- $A = \underline{n}$. Then $LHS = \overline{\rho}(\underline{n}) = \underline{n} = \kappa\underline{n} = RHS$.

- $A = X_k$. Note that since X_k is a subterm of a metaterm (i.e. a well-formed pre-metaterm) k is a simple label. According to Definition 4.8 we have three subcases to consider:
  1. $\mathtt{Ba}_{(L,R)}(X) = \emptyset$. We reason as follows

$$\begin{array}{lll} LHS & = & \overline{\rho}(X[shift^{|k|}]) \\ & = & \overline{\rho}(X)[shift^{|k|}] \\ & =_{Definition\ 5.6} & \mathcal{D}_0^{|l|}(\kappa X_l)[shift^{|k|}] \\ & =^{Lemma\ 5.2}_{\mathcal{W}} & \kappa X_k \end{array}$$

     where X_l is any metavariable from L.
  2. $\mathtt{Ba}_{(L,R)}(X) \neq \emptyset$ and $cons(b_1, \ldots, b_{|l|}, shift^j) \neq cons(1, \ldots, |l|, shift^{|l|})$ where X_l is the X-based pivot metavariable as dictated by P. We reason as follows:

$$\begin{array}{lll} LHS & = & \overline{\rho}(X[cons(b_1, \ldots, b_{|l|}, shift^j)]) \\ & = & \overline{\rho}(X)[cons(b_1, \ldots, b_{|l|}, shift^j)] \\ & = & \mathcal{A}_0^l(\kappa X_l)[cons(b_1, \ldots, b_{|l|}, shift^j)] \\ & =^{Lemma\ 5.5}_{\mathcal{W}} & \kappa X_k. \end{array}$$

  3. $\mathtt{Ba}_{(L,R)}(X) \neq \emptyset$ and $cons(b_1, \ldots, b_{|l|}, shift^j) = cons(1, \ldots, |l|, shift^{|l|})$ where X_l is the X-based pivot metavariable as dictated by P. We reason as follows:

$$\begin{array}{lll} LHS & = & \overline{\rho}(X) \\ & = & \mathcal{A}_0^l(\kappa X_l) \\ & =_{\mathcal{W}} & \mathcal{A}_0^l(\kappa X_l)[cons(1, \ldots, |l|, shift^{|l|})] \\ & =^{Lemma\ 5.5}_{\mathcal{W}} & \kappa X_k. \end{array}$$

     Note that the third equality holds by the fact that $cons(1, \ldots, |l|, shift^{|l|})$ behaves as the identity substitution.
- $A = f(A_1, \ldots, A_n)$ or $A = \xi(A_1, \ldots, A_n)$. Then for the first case we have

$$LHS = f(\overline{\rho}(\mathcal{C}_P^{(L,R)}(A_1)), \ldots, \overline{\rho}(\mathcal{C}_P^{(L,R)}(A_n))) =^{i.h.}_{\mathcal{W}} f(\kappa A_1, \ldots, \kappa A_n) = RHS.$$

  The second case is similar.
- $A = A_1[A_2]$. Then

$$\begin{array}{lll} LHS & = & \overline{\rho}(\mathcal{C}_P^{(L,R)}(A_1[A_2])) \\ & = & \overline{\rho}(\mathcal{C}_P^{(L,R)}(A_1))[cons(\overline{\rho}(\mathcal{C}_P^{(L,R)}(A_2)), id)] \\ & =^{i.h.}_{\mathcal{W}} & (\kappa A_1)[cons(\kappa A_2, id)] \\ & =^{Lemma\ 3.15(3)} & \kappa A_1\{\!\{1 \leftarrow \kappa A_2\}\!\} \\ & = & \kappa(A_1[A_2]) \\ & = & RHS \end{array}$$

∎

Received May 2002