From Higher-Order to First-Order Rewriting

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

[1] Departamento de Computación - Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Argentina.
Email:{ebonelli@dc.uba.ar,rios@dc.uba.ar}
[2] LRI (CNRS URA 410) - Bât 490, Université de Paris-Sud, 91405 Orsay Cedex, France. Email:kesner@lri.fr

Abstract. We show how higher-order rewriting may be encoded into first-order rewriting modulo an equational theory $\mathcal{E}$. We obtain a characterization of the class of higher-order rewriting systems which can be encoded by first-order rewriting modulo an empty theory (that is, $\mathcal{E} = \emptyset$). This class includes of course the λ-calculus. Our technique does not rely on a particular substitution calculus but on a set of abstract properties to be verified by the substitution calculus used in the translation.

1 Introduction

Higher-order substitution is a complex operation that consists in the replacement of variables by terms in the context of languages having variable bindings. These bound variables can be annotated by de Bruijn indices so that the renaming operation (α-conversion) which is necessary to carry out higher-order substitution can be avoided. However, substitution is still a complicated notion, which cannot be expressed by simple replacement (a.k.a. grafting) of variables as is done in first-order theories. To solve this problem, many researchers became interested in the formalization of higher-order substitution by *explicit substitutions*, so that higher-order systems/formalisms could be expressible in first-order systems/formalisms: the notion of variable binding is dropped because substitution becomes replacement. A well-known example of the combination of de Bruijn indices and explicit substitutions is the formulation of different first-order calculi for the λ-calculus [1, 4, 17, 15, 6, 22], 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 [11], higher-order logic to first-order logic modulo [12], higher-order theorem proving to first-order theorem proving modulo [9], etc.

All these translations have a theoretical interest because the expressive power of higher and first-order formalisms is put in evidence, but another practical issue arises, that of the possibility of transferring results developed in the first-order framework to the higher-order one.

The goal of this paper is to give a translation of higher-order rewrite systems (*HORS*) to a first-order formalism. As a consequence, properties and techniques

developed for first-order rewriting could be exported to the higher-order formalism. For example, techniques concerning confluence, termination, completion, evaluation strategies, etc. This is very interesting since, on one hand it is still not clear how to transfer techniques such as dependency pairs [2], semantic labelling [26] or completion [3] to the higher-order framework, on the other hand, termination techniques such as RPO for higher-order systems [14] turn out to be much more complicated than their respective first-order versions [7, 16].

The main difficulty encountered in our translation can be intuitively explained by the fact that in higher-order rewriting a metavariable occurring on the right-hand side in a higher-order rewrite rule may occur in a different *binding context* on the left-hand side: for example, in the usual presentation of the extensional rule for functional types in λ-calculus (η):$\lambda\alpha.app(X,\alpha)\longrightarrow X$, the occurrence of X on the right-hand side of the rule, which does not appear in any binding context, is related to the occurrence of X on the left-hand side, which appears inside a binding context. The immediate consequence of this fact is that the first-order translation of the rule (η) cannot be defined as the naive translation taking both sides of the rule independently. This would give the first-order rule $\lambda(app(X,\underline{1}))\longrightarrow X$, which does not reflect the intended semantics and hence the translation would be incorrect.

As mentioned before, the need for (α)-conversion immediately disappears when de Bruijn notation is considered. Following the example recently introduced, one can express the (η)-rule in a higher-order de Bruijn setting, such as for example the $SERS_{DB}$ (Simplified Expression Reduction Systems) formalism [5], by the rule (η_{dB}) :$\lambda(app(X_\alpha,\underline{1}))\longrightarrow X_\epsilon$. The notation used to translate the metavariable X into the de Bruijn formalism enforces the fact that the occurrence of X on the right-hand side of the rule (η) does not appear in a binding context, so it is translated as X_ϵ where ϵ represents an empty binding path, while the occurrence of X on the left-hand side of the rule appears inside a binding context, so it is translated as X_α, where α represents a binding path of length 1 surrounding the metavariable X.

Now, the term $\lambda(app(\underline{3},\underline{1}))$ reduces to $\underline{2}$ via the (η_{dB}) rule. In an explicit substitution setting, we have the alternative formulation:

$$(\eta_{fo}) : \lambda(app(X[\uparrow],\underline{1}))\longrightarrow X$$

However, in order for the metaterm $X[\uparrow]$ to match the subterm $\underline{3}$ first-order matching no longer suffices: 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 $\underline{3} \stackrel{?}{=}_{\mathcal{E}} X[\uparrow]$. Equivalently, we could make use of the theory of conditional rewriting:

$$\lambda(app(Y,\underline{1}))\longrightarrow X \text{ where } Y =_{\mathcal{E}} X[\uparrow]$$

Another less evident example is given by a commutation rule as for example

$$(C) : Implies(\exists\alpha.\forall\beta.X, \forall\beta.\exists\alpha.X)\longrightarrow true$$

which expresses that the formula appearing as the first argument of the $Implies$ function symbol implies the one in the second argument. The naive translation to

first-order $Implies(\exists(\forall(X)), \forall(\exists(X))) \longrightarrow true$ is evidently not correct, so that we take its translation in the de Bruijn higher-order formalism $SERS_{DB}$ and then translate it to first-order via the conversion presented in this work obtaining:

$$(C_{fo}) : Implies(\exists(\forall(X)), \forall(\exists(X[\underline{2} \cdot \underline{1} \cdot id]))) \longrightarrow true$$

Now, the rule (C_{fo}) has exactly the same semantics as the original higher-order rule (C). This difficulty does not seem to appear in other problems dealing with translations from higher-order to first-order recently mentioned.

In this work we shall see how higher-order rewriting may be systematically reduced to first-order rewriting modulo an equational theory $\mathcal{E}$. To do this, we choose to work with Expression Reduction Systems [18], and in particular, with de Bruijn based $SERS_{DB}$ as defined in [5] which facilitates the translation of higher-order systems to first-order ones. However, we claim that the same translation could be applied to other higher-order rewriting formalisms existing in the literature. We obtain a characterization of the class of $SERS_{DB}$ (including λ-calculus) for which a translation to a full ($\mathcal{E} = \emptyset$) first-order rewrite system exists. Thus the result mentioned above on the λ-calculus becomes a particular case of our work.

To the best of our knowledge there is just one formalism, called XRS [24], which studies higher-order rewrite formalisms based on de Bruijn index notation and explicit substitutions. The formalism XRS, which is a first-order formalism, is presented as a generalization of the first-order $\sigma_{\Uparrow}$-calculus [13] to higher-order rewriting and not as a first-order formulation of higher-order rewriting. As a consequence, many well-known higher-order rewriting systems cannot be expressed in such a formalism [5]. Not only do we provide a first-order presentation of higher-order rewriting, but we do not attach to the translation any particular substitution calculus. Instead, we have chosen to work with an abstract formulation of substitution calculi, as done for example in [17] to deal with confluence proofs of λ-calculi with explicit substitutions. As a consequence, the method we propose can be put to work in the presence of different calculi of explicit substitution such as σ [1], $\sigma_{\Uparrow}$ [13], υ [4], f [17], d [17], s [15], χ [20].

The paper is organized as follows. Section 2 recalls the formalism of higher-order rewriting with de Bruijn indices defined in [5], and Section 3 defines a first-order syntax which will be used as target calculus in the conversion procedure given in Section 4. Properties of the conversion procedure are studied in Section 5: the conversion is a translation from higher-order rewriting to first-order rewriting modulo, the translation is conservative and finally we give the syntactical criterion to be used in order to decide if a given higher-order system can be translated into a full first-order one (a first-order system modulo an empty theory). We conclude in Section 6.

2 The Higher-Order Framework

We briefly recall here the de Bruijn indices based higher-order rewrite formalism called Simplified Expression Reduction Systems with de Bruijn Indices

($SERS_{DB}$) which was introduced in [5]. In full precision we shall work with $SERS_{DB}$ without i(ndex)-metavariables (c.f. full version for details).

Definition 1 (Labels). *A* label *is a finite sequence of symbols of an alphabet. We shall use $k, l, l_i, \ldots$ to denote arbitrary labels and ϵ for the empty label. If α is a symbol and l is a label then $\alpha \in l$ means that the symbol α* appears in *the label l. Other notations are $|l|$ for the* length *of l and $\mathtt{at}(l, n)$ for the n-th element of l assuming $n \leq |l|$. Also, if α occurs (at least once) in l then $\mathtt{pos}(\alpha, l)$ denotes the* position of the first occurrence *of α in l. A* simple label *is a label without repeated symbols.*

Definition 2 (de Bruijn signature). *Consider the denumerable and disjoint infinite sets:*

- $\{\alpha_1, \alpha_2, \alpha_3, \ldots\}$ *a set of symbols called* binder indicators, *denoted $\alpha, \beta, \ldots$,*
- $\{X_l^1, X_l^2, X_l^3, \ldots\}$ *a set of* t-metavariables *(t for term), where l ranges over the set of labels built over binder indicators, denoted $X_l, Y_l, Z_l, \ldots$,*
- $\{f_1, f_2, f_3, \ldots\}$ *a set of* function symbols *equipped with a fixed (possibly zero) arity, denoted $f, g, h, \ldots$,*
- $\{\lambda_1, \lambda_2, \lambda_3, \ldots\}$ *a set of* binder symbols *equipped with a fixed (non-zero) arity, denoted $\lambda, \mu, \nu, \xi, \ldots$.*

Definition 3 (de Bruijn pre-metaterms). *The set of* de Bruijn pre-metaterms, *denoted $\mathcal{PMT}_{db}$, is defined by the following two-sorted grammar:*

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

We shall use $A, B, A_i, \ldots$ to denote de Bruijn pre-metaterms. The symbol $.[\![.]\!]$ is called *de Bruijn metasubstitution operator*. We assume 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 shall abbreviate $\mathtt{S}^{j-1}(1)$ as $\underline{j}$.

Definition 4 (Free de Bruijn metavariables). *Let A be a de Bruijn pre-metaterm. The set of* metavariables *of A, $MVar(A)$, is defined as:*

$$\begin{array}{lll} MVar(\underline{n}) =_{def} \emptyset & & MVar(f(A_1, \ldots, A_n)) =_{def} \bigcup_{i=1}^{n} MVar(A_i) \\ MVar(X_l) =_{def} \{X_l\} & & MVar(\xi(A_1, \ldots, A_n)) =_{def} \bigcup_{i=1}^{n} MVar(A_i) \\ & & MVar(A_1[\![A_2]\!]) =_{def} MVar(A_1) \cup MVar(A_2) \end{array}$$

An *X-based* t-metavariable is a t-metavariable of the form X_l for some label l, we say in that case that X is the *name of* X_l. We shall use $NMVar(A)$ to denote the set of names of metavariables in A. In order to say that a t-metavariable X_l occurs in a pre-metaterm A we write $X_l \in A$.

We shall need the notion of *metaterm* (well-formed pre-metaterm). The first motivation is to guarantee that labels of t-metavariables are correct w.r.t the context in which they appear, the second one is to ensure that indices like $\mathtt{S}^i(1)$ correspond to bound variables. Indeed, pre-metaterms like $\xi(X_{\alpha\beta})$ and $\xi(\xi(\underline{4}))$ shall not make sense for us, and hence shall not be considered well-formed. Well-formed pre-metaterms shall be used to describe rewrite rules.

Definition 5 (de Bruijn metaterms). *A pre-metaterm $A \in \mathcal{PMT}_{db}$ is said to be a* metaterm *iff the predicate $\mathcal{WF}(A)$ holds where $\mathcal{WF}(A) =_{def} \mathcal{WF}_\epsilon(A)$ and $\mathcal{WF}_l(A)$ is defined as follows:*

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

Example 1. Pre-metaterms $\xi(X_\alpha, \lambda(Y_{\beta\alpha}, \underline{2}))$ and $g(\lambda(\xi(h)))$ are metaterms, while $f(\underline{1}, \xi(X_\beta))$, $\lambda(\xi(X_{\alpha\alpha}))$ and $\xi(X_\alpha, X_\beta)$ (with $\alpha \neq \beta$) are not.

Definition 6 (de Bruijn terms and de Bruijn contexts). *The set of* de Bruijn terms*, denoted $\mathcal{T}_{db}$, and the set of de Bruijn contexts are defined by:*

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

We use $a, b, a_i, b_i, \ldots$ for de Bruijn terms and $E, F, \ldots$ for de Bruijn contexts. The *binder path number* of a context is the number of binders between the $\Box$ and the root. For example the binder path of $E = f(\underline{3}, \xi(\underline{1}, \lambda(\underline{2}, \Box, \underline{3})), \underline{2})$ is 2.

Remark that de Bruijn terms are also de Bruijn pre-metaterms, that is, $\mathcal{T}_{db} \subset \mathcal{PMT}_{db}$, although note that some de Bruijn terms may not be de Bruijn metaterms, i.e. may not be well-formed de Bruijn pre-metaterms, e.g. $\xi(\xi(\underline{4}))$.

Definition 7 (Free de Bruijn variables). *We denote by $FV(a)$ the set of* free variables *of a de Bruijn term a, which is defined as follows:*

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

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

Definition 8 (de Bruijn substitution). *The result of substituting a term b for the index $n \geq 1$ in a term a is denoted $a\{\!\{n \leftarrow b\}\!\}$ and defined:*

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

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

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

We now recall the definition of rewrite rules, valuations, their validity, and reduction in $SERS_{DB}$.

Definition 9 (de Bruijn rewrite rule). *A* de Bruijn rewrite rule *is a pair of de Bruijn metaterms (L, R) (also written $L \longrightarrow R$) such that*

- *the first symbol in L is a function symbol or a binder symbol*
- *$NMVar(R) \subseteq NMVar(L)$*
- *the metasubstitution operator $.[\![.]\!]$ does not occur in L*

Definition 10 (de Bruijn valuation). *A* de Bruijn valuation $\overline{\kappa}$ *is a (partial) function from t-metavariables to de Bruijn terms. A valuation $\overline{\kappa}$ determines in a unique way a function κ (also called valuation) on pre-metaterms as follows:*

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

We write $Dom(\kappa)$ for the set $\{X_l \mid \kappa X_l \text{ is defined}\}$, called the *domain* of κ. Note that in the above definition the substitution operator $.\{\!\{. \leftarrow .\}\!\}$ refers to the usual substitution defined on terms with de Bruijn indices (cf. Definition 8).

We now introduce the notion of *value function* which is used to give semantics to metavariables with labels in the $SERS_{DB}$ formalism. The goal pursued by the labels of metavariables is that of incorporating "context" information as a defining part of a metavariable. A typical example is given by a rule like $\mathcal{C} : \xi(\xi(X_{\beta\alpha})) \longrightarrow \xi(\xi(X_{\alpha\beta}))$ where the X-occurrence on the *RHS* of the rule denotes a permutation of the binding context of the X-occurrence on the *LHS*.

As a consequence, we must verify that the terms substituted for every occurrence of a fixed metavariable coincide "modulo" their corresponding context. Dealing with such notion of "coherence" of substitutions in a de Bruijn formalism is also present in other formalisms but in a more restricted form. Thus for example a pre-cooking function[1] is used in [9] to avoid variable capture in the higher-order unification procedure. In *XRS* [24] the notions of binding arity and

[1] The pre-cooking function translates a de Bruijn λ-term with metavariables into a $\lambda\sigma$-term by suffixing each metavariable X with as many explicit shift operators as the binder path number of the context obtained by replacing X by $\Box$. This avoids variable capture when the higher-order unification procedure finds solutions for the t-metavariables.

pseudo-binding arity are introduced to take into account the binder path number of rewrite rules. Then it is required (roughly) that the binding arity of a t-metavariable on the LHS of a rewrite rule (rewrite rules are required to be left-linear) equals the pseudo-binding arity of the same t-metavariable occurring on the RHS of the rule. Our notion of "coherence" is implemented with *valid valuations* (cf. Definition 12) and it turns out to be more general than the solutions proposed in [9] and [24].

Definition 11 (Value function). *Let a be a de Bruijn term and l be a label of binder indicators. We define the value function $Value(l,a)$ as $Value^0(l,a)$ where*

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

It is worth noting that $Value^i(l,\underline{n})$ may give three different kinds of results. This is just a technical resource. Indeed, $Value(\alpha\beta, \xi(f(\underline{3},\underline{1}))) = \xi(f(\beta,\underline{1})) = Value(\beta\alpha, \xi(f(\underline{2},\underline{1})))$ and $Value(\epsilon, f(\xi(\underline{1}), \lambda(\underline{2}))) = f(\xi(\underline{1}), \lambda(x_1)) \neq f(\xi(\underline{1}), \lambda(\alpha)) = Value(\alpha, f(\xi(\underline{1}), \lambda(\underline{2})))$. Thus the function $Value(l,a)$ interprets the de Bruijn term a in an l-context: bound indices are left untouched, free indices referring to the l-context are replaced by the corresponding binder indicator and the remaining free indices are replaced by adequate variable names.

Definition 12 (Valid de Bruijn valuation). *A de Bruijn valuation κ is said to be* valid *if for every pair of t-metavariables X_l and $X_{l'}$ in $Dom(\kappa)$ we have $Value(l, \kappa X_l) = Value(l', \kappa X_{l'})$. Likewise, we say that a de Bruijn valuation κ is* valid for a rewrite rule (L,R) *if for every pair of t-metavariables X_l and $X_{l'}$ in (L,R) we have $Value(l, \kappa X_l) = Value(l', \kappa X_{l'})$.*

Example 2. Let us consider the de Bruijn rule $\mathcal{C} : \xi(\xi(X_{\beta\alpha})) \longrightarrow \xi(\xi(X_{\alpha\beta}))$. We have that $\kappa = \{X_{\beta\alpha}/\underline{2}, X_{\alpha\beta}/\underline{1}\}$ is valid since $Value(\beta\alpha, \underline{2}) = \alpha = Value(\alpha\beta, \underline{1})$.

As already mentioned the η-contraction rule $\lambda x.app(X,x) \longrightarrow X$ can be expressed in the $SERS_{DB}$ formalism as the rule (η_{dB}) $\lambda(app(X_\alpha, \underline{1})) \longrightarrow X_\epsilon$. Our formalism, like other *HORS* in the literature, allows us to use rules like η_{dB} because valid valuations will test for coherence of values.

Definition 13 ($SERS_{DB}$-rewriting). *Let $\mathcal{R}$ be a set of de Bruijn rewrite rules and a, b be de Bruijn terms. We say that a* $\mathcal{R}$-reduces or rewrites to *b, written $a \longrightarrow_{\mathcal{R}} b$, iff there is a de Bruijn rule $(L,R) \in \mathcal{R}$, a de Bruijn valuation κ valid for (L,R), and a de Bruijn context E such that $a = E[\kappa L]$ and $b = E[\kappa R]$.*

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 $\kappa = \{X_\alpha/\lambda(app(\underline{1},\underline{3})), X_\epsilon/\lambda(app(\underline{1},\underline{2}))\}$.

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.

Definition 14. *A* substitution declaration *is a (possibly empty) word over the alphabet* $\{\mathtt{T}, \mathtt{S}\}$. *The symbol* `T` *is used to denote terms and* `S` *to denote substitutions. A* substitution signature *is a set* Γ_s *of* substitution symbols *equipped with an arity* n *and a substitution declaration of length* n. *We use* $\sigma : (w)$ *where* $w \in \{\mathtt{T}, \mathtt{S}\}^n$ *if the substitution symbol* σ *has arity* n *and substitution declaration* w. *We use* ϵ *to denote the empty word.*

Definition 15 (*ExERS* term algebra). *An ExERS signature is a set* $\Gamma = \Gamma_f \cup \Gamma_b \cup \Gamma_s$ *where* $\Gamma_f = \{f_1, \ldots, f_n\}$ *is a set of function symbols,* $\Gamma_b = \{\lambda_1, \ldots, \lambda_n\}$ *is a set of binder symbols,* Γ_s *a substitution signature and* Γ_f, Γ_b *and* Γ_s *are pairwise disjoint. Both binder and function symbols come equipped with an arity. Given a set of (term) variables* $\mathcal{V} = \{X_1, X_2, \ldots\}$, *the term algebra of an ExERS of signature* Γ *generated by* $\mathcal{V}$, *is denoted by* $\mathcal{T}$ *and contains all the objects (denoted by letters* o *and* p*) generated by the following grammar:*

`indices`	$n ::= 1 \mid \mathtt{S}(n)$
`terms (T)`	$a ::= X \mid n \mid a[s] \mid f(a_1, \ldots, a_n) \mid \xi(a_1, \ldots, a_n)$
`substitutions (S)`	$s ::= \sigma(o_1, \ldots, o_n)$

where X *ranges over* $\mathcal{V}$, f *over* Γ_f, ξ *over* Γ_b, *and* σ *over* Γ_s. *The arguments of* σ *are assumed to respect the sorts prescribed in its substitution declaration and function and binder symbols are assumed to respect their arities.*

Letters $a, b, c, \ldots$ and $s, s_i, \ldots$ are used for terms and substitutions, respectively. The .[.] 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. 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.

The formalism of *ExERS* that we are going to use in order to encode higher-order rewriting consists of two sets of rewrite rules:

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

Let us define these two concepts formally.

Definition 16 (Substitution macros). *Let Γ_s be a substitution signature. The following symbols not included in Γ_s are called* substitution macros*: cons* : (TS)*, lift* : (S)*, id* : (ϵ) *and shift^j* : (ϵ) *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, cons$(a_1, \ldots, a_i, s)$ stands for cons$(a_1, \ldots$ cons$(a_i, s))$.*

Definition 17 (Term rewrite and equational systems). *Let Γ be an Ex-ERS 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 the LHS of the rule is a function or a binder symbol, and (3) the set of variables of the LHS includes those of the RHS. An equational (resp. term rewrite) system is a set of equations (resp. term rewrite rules).*

Definition 18 (Substitution calculus). *A* substitution calculus over an *Ex-ERS* signature *Γ consists of a set $\mathcal{W}$ of rewrite rules, and an interpretation of each substitution macro as some combination of substitution symbols from Γ_s of corresponding signature. Definition 19 shall require certain properties for these interpretations to be considered meaningful.*

An example of a substitution calculus is σ [1] with $cons(t, s) =_{def} t \cdot s$, $lift(s) =_{def} \underline{1} \cdot (s \circ \uparrow)$, $id =_{def} id$ and $shift^j =_{def} \uparrow \circ \ldots (\uparrow \circ \uparrow)$, where $\uparrow$ appears j times.

Definition 19 (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 $\xi \in \Gamma_b$: $\mathcal{W}(f(a_1, \ldots, a_n)) = f(\mathcal{W}(a_1), \ldots, \mathcal{W}(a_n))$ and $\mathcal{W}(\xi(a_1, \ldots, a_n)) = \xi(\mathcal{W}(a_1), \ldots, \mathcal{W}(a_n))$.*
4. *Rules for propagating substitutions over function symbols and binders are contained in $\mathcal{W}$, for each $f \in \Gamma_f$ and $\xi \in \Gamma_b$:*

$$\begin{array}{ll} (func_f) & f(X^1, \ldots, X^n)[s] \longrightarrow f(X^1[s], \ldots, X^n[s]) \\ (bind_\xi) & \xi(X^1, \ldots, X^n)[s] \longrightarrow \xi(X^1[lift(s)], \ldots, X^n[lift(s)]) \end{array}$$

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

Example 3. The σ [1], $\sigma_{\Uparrow}$ [13] and ϕ [22] calculi are basic substitution calculi where the set of function and binder symbols are $\{app\}$ and $\{\lambda\}$, respectively.

The reader may have noted that the macro-based presentation of substitution calculi makes use of parallel substitutions (since $cons(.,.)$ has substitution declaration TS). Nevertheless, the results presented in this work may be achieved via a macro-based presentation using a simpler set of substitutions (such as for example the one used in [17]), where $scons(.)$ has substitution declaration T and the macro $shift^i$ is only defined for $i = 1$. Indeed, the expression $b[cons(a_1, \ldots, a_n, shift^j)]$ could be denoted by the expression

$$b[lift^n(shift)]^j[scons(a_1[shift]^{n-1})] \ldots [scons(a_n)]$$

Definition 20 (*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 $\mathcal{R}_\mathcal{W} =_{def} (\Gamma, \mathcal{R}, \mathcal{W})$ 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 .[.] then $\mathcal{R}_\mathcal{W}$ is called a Fully Explicit Expression Reduction System (FExERS).*

Since reduction in $SERS_{DB}$ only takes place on terms, and first-order term rewrite systems will be used to simulate higher-order reduction, 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.

Example 4. Consider the signature Γ formed by $\Gamma_f = \{app\}$, $\Gamma_b = \{\lambda\}$ and Γ_s any substitution signature. Let $\mathcal{W}$ be a basic substitution calculus over Γ. Then for $\mathcal{R} : app(\lambda(X), Y) \longrightarrow_{\beta\, db} X[cons(Y, id)]$ we have that $\mathcal{R}_\mathcal{W}$ is an *FExERS*, and for $\mathcal{R}' : \mathcal{R} \cup \{\lambda(app(X[shift], \underline{1})) \longrightarrow_{\eta db} X\}$, $\mathcal{R}'_\mathcal{W}$ is an *ExERS*.

Reduction in an *ExERS* $\mathcal{R}_\mathcal{W}$ is first-order reduction in $\mathcal{R}$ modulo $\mathcal{W}$-equality. In contrast, reduction in a *FExERS* $\mathcal{R}_\mathcal{W}$ is just first-order reduction in $\mathcal{R} \cup \mathcal{W}$. Before defining these notions more precisely we recall the definition of assignment.

Definition 21 (Assignment (a.k.a. grafting)). *Let $\overline{\rho}$ be a (partial) function mapping variables in $\mathcal{V}$ to ground terms. We define an* assignment *ρ as the unique homeomorphic extension of $\overline{\rho}$ over the set $\mathcal{T}$.*

Definition 22 (Reduction and Equality). *Let o and p be two ground terms of sort* T *or* S*. Given a rewrite system $\mathcal{R}$, we say that o rewrites to p in one step, denoted $o \longrightarrow_\mathcal{R} p$, iff $o = E[\rho L]$ and $p = E[\rho R]$ for some assignment ρ, some context E and some rewrite rule (L, R) in $\mathcal{R}$. We shall use $\stackrel{*}{\longrightarrow}_\mathcal{R}$ to denote the reflexive transitive closure of the one-step rewrite relation.*

Given an equational system $\mathcal{E}$, we say that o equals p modulo $\mathcal{E}$ in one step, denoted $o =^1_\mathcal{E} p$, iff $o = E[\rho L]$ and $p = E[\rho R]$ for some assignment ρ, some context E and some equation $L \doteq R$ in $\mathcal{E}$. We use $=_\mathcal{E}$ to denote the reflexive symmetric transitive closure of $=^1_\mathcal{E}$, and say that o equals p modulo $\mathcal{E}$ if $o =_\mathcal{E} p$.

Definition 23 (*ExERS* and *FExERS*-rewriting). *Let $\mathcal{R}_\mathcal{W}$ be an ExERS, $\mathcal{R}'_\mathcal{W}$ a FExERS and o, p ground terms of sort* S *or* T*. We say that o* $\mathcal{R}_\mathcal{W}$-reduces or rewrites to *p, written $o \longrightarrow_{\mathcal{R}_\mathcal{W}} p$, iff $o \longrightarrow_{\mathcal{R}/\mathcal{W}} p$ (i.e. $o =_\mathcal{W} o' \longrightarrow_\mathcal{R} p' =_\mathcal{W} p$); and o* $\mathcal{R}'_\mathcal{W}$-reduces to *p, iff $o \longrightarrow_{\mathcal{R}' \cup \mathcal{W}} p$.*

3.1 Properties of Basic Substitution Calculi

This subsection takes a look at properties enjoyed by basic substitution calculi and introduces a condition called the *Scheme* [17]. Basic substitution calculi satisfying the scheme ease inductive reasoning when proving properties over them without compromising the genericity achieved by the macro-based presentation.

Lemma 1 (Behavior 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 $\mathtt{S}$:* $\underline{m}[lift^n(s)] =_{\mathcal{W}} \begin{cases} \underline{m-n}[s][shift]^n & \text{if } m > n \\ \underline{m} & \text{if } m \leq n \end{cases}$
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 $m \geq 1$:* $a\{\!\{m \leftarrow b\}\!\} =_{\mathcal{W}} a[lift^{m-1}(cons(b, id))]$.

The first and third items of Lemma 1 are proved in [17], the second item follows from the definition of a basic substitution calculus. For the proof of the following lemma the reader is referred to [17].

Lemma 2. *Let $\mathcal{W}$ be a basic substitution calculus, a a pure term and s a term of sort $\mathtt{S}$. Then $\mathcal{W}(a[s][shift]) = \mathcal{W}(a[shift][lift(s)])$.*

Corollary 1. *Let $\mathcal{W}$ be a basic substitution calculus, a a pure term and s a term of sort $\mathtt{S}$. Then for every $p \geq n \geq 0$ we have $a[shift]^n[lift^p(s)] =_{\mathcal{W}} a[lift^{p-n}(s)][shift]^n$.*

Lemma 3. *Let $\mathcal{W}$ be a basic substitution calculus, a a pure term, b a term of sort $\mathtt{T}$ and s a term of sort $\mathtt{S}$. Then for every $n \geq 0$, $a[lift^n(shift)][lift^n(cons(b, id))] =_{\mathcal{W}} a$*

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

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

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

Lemma 4 (Substitution commutation). *Let $\mathcal{W}$ be a basic substitution calculus, a a pure term, b any term, s a term of sort $\mathtt{S}$. Then for every $p \geq n \geq 0$ we have:*

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

Proof. By induction on the structure of a.

- $a = \underline{m}$. Then we consider three further cases:

- $m > n+1$.

$$\begin{array}{ll} & \mathcal{W}(a[lift^n(cons(b,id))][lift^p(s)]) \\ =_{L.\ 1(1)} & \mathcal{W}(\underline{m-n}[cons(b,id)][shift]^n[lift^p(s)]) \\ = & \mathcal{W}(\underline{m-n-1}[shift]^n[lift^p(s)]) \\ =_{L\ 1} & \mathcal{W}(\underline{m-n-1}[lift^{p-n}(s)][shift]^n) \end{array}$$

and

$$\begin{array}{ll} & \mathcal{W}(a[lift^{p+1}(s)][lift^n(cons(b[lift^{p-n}(s)],id))]) \\ = & \mathcal{W}(\underline{m-n-1}[lift^{p-n}(s)][shift][shift]^n[lift^n(cons(b[lift^{p-n}(s)],id))]) \\ = & \mathcal{W}(\underline{m-n-1}[lift^{p-n}(s)][shift][cons(b[lift^{p-n}(s)],id)][shift]^n) \\ =_{L\ 3} & \mathcal{W}(\underline{m-n-1}[lift^{p-n}(s)][id][shift]^n) \\ =_{Def\ 19(10)} & \mathcal{W}(\underline{m-n-1}[lift^{p-n}(s)][shift]^n) \end{array}$$

- $m = n+1$.

$$\begin{array}{ll} & \mathcal{W}(\underline{m}[lift^n(cons(b,id))][lift^p(s)]) \\ =_{L.\ 1(1)} & \mathcal{W}(\underline{1}[cons(b,id)][shift]^n[lift^p(s)]) \\ = & \mathcal{W}(b[shift]^n[lift^p(s)]) \\ = & \mathcal{W}(b[lift^{p-n}(s)][shift]^n) \end{array}$$

and

$$\begin{array}{l} \mathcal{W}(\underline{m}[lift^{p+1}(s)][lift^n(cons(b[lift^{p-n}(s)],id))]) \\ = \mathcal{W}(\underline{1}[lift^{p-n+1}(s)][shift]^n[lift^n(cons(b[lift^{p-n}(s)],id))]) \\ = \mathcal{W}(\underline{1}[shift]^n[lift^n(cons(b[lift^{p-n}(s)],id))]) \\ = \mathcal{W}(\underline{n+1}[lift^n(cons(b[lift^{p-n}(s)],id))]) \\ = \mathcal{W}(\underline{1}[cons(b[lift^{p-n}(s)],id)][shift]^n) \\ = \mathcal{W}(b[lift^{p-n}(s)][shift]^n) \end{array}$$

- $m < n+1$. Then we have:

$$\begin{array}{ll} & \mathcal{W}(\underline{m}[lift^n(cons(b,id))][lift^p(s)]) \\ =_{L.\ 1(1)} & \mathcal{W}(\underline{m}[lift^p(s)]) \\ =_{L.\ 1(1)} & \underline{m} \\ =_{L.\ 1(1)} & \mathcal{W}(\underline{m}[lift^n(cons(b[lift^{p-n}(s)],id))]) \\ =_{L.\ 1(1)} & \mathcal{W}(\underline{m}[lift^{p+1}(s)][lift^n(cons(b[lift^{p-n}(s)],id))]) \end{array}$$

– $a = f(a_1,\ldots,a_n)$. Use the induction hypothesis.
– $a = \xi(a_1,\ldots,a_n)$. Use the induction hypothesis.

Definition 24 (The Scheme). *We say that a basic substitution calculus $\mathcal{W}$ obeys the* scheme *iff for every index $\underline{m}$ 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 $\underline{n}$, 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_j's 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* $\mathtt{T}$.

Example 5. Example of calculi satisfying the scheme are σ, $\sigma_{\Uparrow}$, υ, f and d [17].

4 From Higher-Order to First-Order rewriting

In this section we give an algorithm, referred to as the Conversion Procedure, 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 t-metavariables, must be met in order for a substitution to be admitted as *valid*. The idea is to replace all occurrences of t-metavariables X_l by a first-order variable X followed by an appropriate *index-adjusting explicit substitution* which computes valid valuations.

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

4.1 The Conversion Procedure

Definition 25 (Binding allowance). *Let* M *be a metaterm and* $\{X_{l_1}, \ldots, X_{l_n}\}$ *be the set of all the t-metavariables with name* X *occurring in* M*. Then, the* binding allowance of X in M*, noted* $\mathtt{Ba}_M(X)$*, is the set* $\bigcap_{i=1}^{n} l_i$*. Likewise, we define the* binding allowance of X in a rule (L, R)*, written* $\mathtt{Ba}_{(L,R)}(X)$.

Example 6. Let $M = f(\xi(X_\alpha), \xi(\lambda(X_{\beta\alpha})), \nu(\xi(X_{\alpha\gamma})))$, then $\mathtt{Ba}_M(X) = \{\alpha\}$.

Definition 26 (Shifting index). *Let* M *be a metaterm,* X_l *a t-metavariable occurring in* M*, and* i *a position in* l*. The* shifting index determined by X_l in M at position i*, denoted* $\mathtt{Sh}(M, X_l, i)$*, is defined as*

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

$\mathtt{Sh}(M, X_l, i)$ *is just the total number of binder indicators in* l *at positions* $1..i-1$ *that do not belong to* $\mathtt{Ba}_M(X)$ *(thus* $\mathtt{Sh}(M, X_l, 1)$ *is always* 0*). Likewise, we define the* shifting index determined by X_l in a rule (L, R) at position i*, written* $\mathtt{Sh}((L, R), X_l, i)$.

Example 7. Let $M = f(\xi(X_\alpha), \xi(\lambda(X_{\beta\alpha})), \nu(\xi(X_{\alpha\gamma})))$. Then $\mathtt{Sh}(M, X_{\beta\alpha}, 2) = 1$ and $\mathtt{Sh}(M, X_\alpha, 1) = \mathtt{Sh}(M, X_{\alpha\gamma}, 2) = 0$.

Definition 27 (Pivot). *Let* (L, R) *be a* $SERS_{DB}$*-rewrite rule and let us suppose that* $\{X_{l_1}, \ldots, X_{l_n}\}$ *is the set of all* X*-based t-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. $X_{l_j} \in L$ *whenever possible.*

A pivot set for a rewrite rule (L, R) *is a set of pivot t-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.*

Note that Definition 27 admits the existence of more than one X-based pivot t-metavariable. A pivot set for (L, R) fixes a t-metavariable for each t-metavariable name having a non-empty binding allowance.

Example 8. Both t-metavariables $X_{\alpha\beta}$ and $X_{\beta\alpha}$ can be chosen as X-based pivot in the rewrite rule $\mathcal{R} : \mathit{Implies}(\exists(\forall(X_{\alpha\beta})), \forall(\exists(X_{\beta\alpha}))) \longrightarrow \mathit{true}$. In the rewrite rule $\mathcal{R}' : f(Y_\epsilon, \lambda(\xi(X_{\alpha\beta})), \nu(\lambda(X_{\beta\alpha}))) \longrightarrow \mu(X_\alpha, Y_\alpha)$ the t-metavariable X_α is the only possible X-based pivot.

Definition 28 (Conversion of t-metavariables). *Consider a* $SERS_{DB}$*-rewrite rule* (L, R) *and a pivot set for* (L, R). *We consider the following cases for every t-metavariable name* X *occurring in* L*:*

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

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

   (b) if X_k *is not the pivot then*

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

Note that for an index-adjusting substitution $X[\mathit{cons}(b_1, \ldots, b_{|l|}, \mathit{shift}^j)]$ each b_i is a distinct de Bruijn index and less than or equal to j. Substitutions of this form have been called pattern substitutions in [10], where unification of higher-order patterns via explicit substitutions is studied.

Definition 29 (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 $\mathcal{C}_P(L,R) =_{def} (\mathcal{C}_P^{(L,R)}(L), \mathcal{C}_P^{(L,R)}(R))$ where $\mathcal{C}_P^{(L,R)}(M)$ is defined by induction on M, where $NMVar(M) \subseteq NMVar(L)$, as:*

$$
\begin{array}{lll}
\mathcal{C}_P^{(L,R)}(\underline{n}) & =_{def} \underline{n} & \\
\mathcal{C}_P^{(L,R)}(X_l) & =_{def} \begin{cases} X[shift^{|l|}] & \text{if } \mathtt{Ba}_{(L,R)}(X) = \emptyset \\ X[cons(b_1,\ldots,b_{|l|},shift^j)] & \text{if } \mathtt{Ba}_{(L,R)}(X) \neq \emptyset \text{ and} \\ & cons(b_1,\ldots,b_{|l|},shift^j) \neq \\ & cons(\underline{1},\ldots,\underline{|l|},shift^{|l|}) \\ X & \text{otherwise} \end{cases} \\
\mathcal{C}_P^{(L,R)}(f(M_1,\ldots,M_n)) & =_{def} f(\mathcal{C}_P^{(L,R)}(M_1),\ldots,\mathcal{C}_P^{(L,R)}(M_n)) & \\
\mathcal{C}_P^{(L,R)}(\xi(M_1,\ldots,M_n)) & =_{def} \xi(\mathcal{C}_P^{(L,R)}(M_1),\ldots,\mathcal{C}_P^{(L,R)}(M_n)) & \\
\mathcal{C}_P^{(L,R)}(M_1[\![M_2]\!]) & =_{def} \mathcal{C}_P^{(L,R)}(M_1)[cons(\mathcal{C}_P^{(L,R)}(M_2), id)] &
\end{array}
$$

The term $X[cons(b_1,\ldots,b_{|l|},shift^j)]$ in the third clause is the index-adjusting substitution computed in Definition 28.

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

Example 9. Below we present some examples of conversion of rules. We have fixed $\mathcal{W}$ to be the σ-calculus.

$SERS_{DB}$ rule	*Pivot selected*	*Transformed rule*
$\lambda(app(X_\alpha,\underline{1})) \longrightarrow X_\epsilon$	-	$\lambda(app(X[\uparrow],\underline{1})) \longrightarrow X$
$\lambda(\lambda(X_{\alpha\beta})) \longrightarrow \lambda(\lambda(X_{\beta\alpha}))$	$X_{\alpha\beta}$	$\lambda(\lambda(X)) \longrightarrow \lambda(\lambda(X[\underline{2}\cdot\underline{1}\cdot(\uparrow\circ\uparrow)]))$
$f(\lambda(\lambda(X_{\alpha\beta})),\lambda(\lambda(X_{\beta\alpha}))) \longrightarrow \lambda(X_\gamma)$	-	$f(\lambda(\lambda(X[\uparrow\circ\uparrow])),\lambda(\lambda(X[\uparrow\circ\uparrow]))) \longrightarrow \lambda(X[\uparrow])$
$app(\lambda(X_\alpha),Z_\epsilon) \longrightarrow_{\beta_{db}} X_\alpha[\![Z_\epsilon]\!]$	X_α, Z_ϵ on LHS	$app(\lambda(X),Z) \longrightarrow X[cons(Z,id)]$

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 the rule β_{db} of Example 9.

Also, observe that if we replace our $cons(.,.)$ macro by a $scons(.)$ of substitution declaration `T` as defined in [17] then the "Replace metasubstitution operator" step in Definition 30 converts a metaterm of the form $M[\![N]\!]$ into $M[scons(N)]$, yielding first-order systems based on substitution calculi, such as $\lambda\upsilon$, which do not implement parallel substitution.

The resulting system of the Conversion Procedure is coded as an *ExERS*, a framework for defining first-order rewrite systems where $\mathcal{E}$-matching is used, $\mathcal{E}$ being an equational theory governing the behaviour of the index adjustment substitutions. Moreover, if it is possible, an *ExERS* may further be coded as a

FExERS where reduction is defined on first-order terms and matching is just syntactic first-order matching, obtaining a *full first-order system.*

Definition 30 (First-order version of $\mathcal{R}$). *Let Γ be an ExERS signature and let $\mathcal{R}$ be a $SERS_{DB}$. Consider the system $fo(\mathcal{R})$ obtained by applying the Conversion Procedure to $\mathcal{R}$ and let $\mathcal{W}$ be a substitution calculus over Γ. Then 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 $\mathcal{R}$. This requires considering pivot selection, an issue we take up in Section 4.2.

4.2 On pivot selection

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 $\mathcal{C}_Q(L, R)$ shall not be identical. Nevertheless, in this subsection we shall show that the reduction 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 t-metavariables in (L, R) with $\mathtt{Ba}_{(L,R)}(X) \neq \emptyset$. Let X_{l_1} and X_{l_2} be two possible X-based pivot 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 $X_k \in L$). We write M to denote either L or R. Also, we have $|l_1| = |l_2|$, a fact that shall be made use of freely below.

Let us consider two different translations (a) and (b) as dictated by Definition 28 taking any t-metavariable X_{l_i} $(i = 1 \ldots n)$ and yielding a first-order term:

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

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.

Define the following indices c_i $(i = 1..|l_1|)$:

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

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 $a^2_j = |l_2| + 1 + \mathtt{Sh}((L, R), X_{l_1}, j)$ and since $|l_1| = |l_2|$, l_1 and l_2 are simple, 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 is a $j' \in 1..|l_1|$ such that $b^1_{j'} = |l_1| + 1 + \mathtt{Sh}((L, R), X_{l_2}, j')$ with $\mathtt{Sh}((L, R), X_{l_2}, j') = \mathtt{Sh}((L, R), X_{l_1}, j)$, and hence $a^2_j = b^1_{j'}$.

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

Lemma 5. *Let n be the number of X-based t-metavariables in (L,R) and let X_{l_1} and X_{l_2} be two distinct pivots for (L,R) (we write M to denote L or R). Let $h \geq 0$ and $1 \leq i \leq n$. Take any assignment ρ and indices a^i_j $(1 \leq j \leq |l_1|)$ and b^i_j $(1 \leq j \leq |l_2|)$ as indicated before in the translations (a) and (b). Then*

$$\begin{array}{rl} & (\rho X)[\mathit{lift}^h(\mathit{cons}(a^i_1,\ldots,a^i_{|l_1|},\mathit{shift}^{|l_i|+|l_1\backslash \mathtt{Ba}_{(L,R)}(X)|}))] \\ =_{\mathcal{W}} & (\rho X)[\mathit{lift}^h(s)][\mathit{lift}^h(\mathit{cons}(b^i_1,\ldots,b^i_{|l_2|},\mathit{shift}^{|l_i|+|l_2\backslash \mathtt{Ba}_{(L,R)}(X)|}))] \end{array}$$

where $s = \mathit{cons}(c_1,\ldots,c_{|l_1|},\mathit{shift}^{|l_1|})$.

Proof. Note that we may assume that ρX is a pure term without loss of generality. We proceed by induction on ρX.

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

    $$\begin{array}{l} \mathit{LHS} =_{\mathcal{W}} \underline{j-h}[\mathit{cons}(a^i_1,\ldots,a^i_{|l_1|},\mathit{shift}^{|l_i|+|l_1\backslash \mathtt{Ba}_{(L,R)}(X)|})][\mathit{shift}]^h =_{\mathcal{W}} a^i_{j-h}+h \\ \mathit{RHS} =_{\mathcal{W}} c_{j-h}[\mathit{cons}(b^i_1,\ldots,b^i_{|l_2|},\mathit{shift}^{|l_i|+|l_2\backslash \mathtt{Ba}_{(L,R)}(X)|})][\mathit{shift}]^h \end{array}$$

    We now 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

         $$\begin{array}{rl} & \mathit{RHS} \\ = & a^2_{j-h}[\mathit{cons}(b^1_1,\ldots,b^1_{|l_2|},\mathit{shift}^{|l_1|+|l_2\backslash \mathtt{Ba}_{(L,R)}(X)|})][\mathit{shift}]^h \\ = & \mathtt{pos}(\beta,l_2)[\mathit{cons}(b^1_1,\ldots,b^1_{|l_2|},\mathit{shift}^{|l_1|+|l_2\backslash \mathtt{Ba}_{(L,R)}(X)|})][\mathit{shift}]^h \\ =_{\mathcal{W}} & b^1_{\mathtt{pos}(\beta,l_2)}+h \\ = & \mathtt{pos}(\beta,l_1)+h \\ = & \underline{j-h+h} \\ =_{\mathcal{W}} & \mathit{LHS} \end{array}$$

         Recall that all labels are simple (no duplicated elements).
       - $\mathtt{at}(l_1, j-h) \notin \mathtt{Ba}_{(L,R)}(X)$.

         $$\begin{array}{rl} & \mathit{RHS} \\ = & \mathtt{pos}(a^2_{j-h}, b^1_1..b^1_{|l_1|})[\mathit{cons}(b^1_1,\ldots,b^1_{|l_2|},\mathit{shift}^{|l_1|+|l_2\backslash \mathtt{Ba}_{(L,R)}(X)|})][\mathit{shift}]^h \\ = & \mathtt{pos}(|l_2|+1+\mathtt{Sh}((L,R),X_{l_1},j-h), b^1_1..b^1_{|l_1|})[\mathit{cons}(b^1_1,\ldots,b^1_{|l_2|},\mathit{shift}^{|l_1|+|l_2\backslash \mathtt{Ba}_{(L,R)}(X)|})][\mathit{shift} \\ =_{\mathcal{W}} & \underline{|l_2|+1+\mathtt{Sh}((L,R),X_{l_1},j-h)+h} \\ = & \underline{|l_1|+1+\mathtt{Sh}((L,R),X_{l_1},j-h)+h} \\ =_{\mathcal{W}} & \mathit{LHS} \end{array}$$

    2. $i \geq 2$. Suppose

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

$$\begin{array}{ll} & RHS \\ = & a^2_{j-h}[cons(b^i_1, \ldots, b^i_{|l_2|}, shift^{|l_i|+|l_2 \setminus \mathtt{Ba}_{(L,R)}(X)|})][shift]^h \\ = & \mathtt{pos}(\beta, l_2)[cons(b^i_1, \ldots, b^i_{|l_2|}, shift^{|l_i|+|l_2 \setminus \mathtt{Ba}_{(L,R)}(X)|})][shift]^h \\ =_{\mathcal{W}} & b^i_{\mathtt{pos}(\beta,l_2)} + h \\ = & \mathtt{pos}(\beta, l_i) + h \\ =_{\mathcal{W}} & LHS \end{array}$$

If $i = 2$ then the last step follows from the case (a) of Definition 28 (since X_{l_2} is a pivot t-metavariable occurrence for conversion (b)), otherwise it follows from case (b) of the same definition.

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

$$\begin{array}{l} RHS \\ = \mathtt{pos}(a^2_{j-h}, b^1_1..b^1_{|l_1|})[cons(b^i_1, \ldots, b^i_{|l_2|}, shift^{|l_i|+|l_2 \setminus \mathtt{Ba}_{(L,R)}(X)|})][shift]^h \\ = \mathtt{pos}(|l_2| + 1 + \mathtt{Sh}((L,R), X_{l_1}, j-h), b^1_1..b^1_{|l_1|})[cons(b^i_1, \ldots, b^i_{|l_2|}, shift^{|l_i|+|l_2 \setminus \mathtt{Ba}_{(L,R)}(X)|})][shift]^h \end{array}$$

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

$$\begin{array}{ll} & \mathtt{pos}(|l_2| + 1 + \mathtt{Sh}((L,R), X_{l_1}, j-h), b^1_1..b^1_{|l_1|})[cons(b^i_1, \ldots, b^i_{|l_2|}, shift^{|l_i|+|l_2 \setminus \mathtt{Ba}_{(L,R)}(X)|})][shift \\ = & j'[cons(b^i_1, \ldots, b^i_{|l_2|}, shift^{|l_i|+|l_2 \setminus \mathtt{Ba}_{(L,R)}(X)|})][shift]^h \\ =_{\mathcal{W}} & b^i_{j'} + h \\ = & \underline{|l_i| + 1 + \mathtt{Sh}((L,R), X_{l_2}, j') + h} \end{array}$$

The last equality follows from the fact that $b^1_{j'} > |l_1|$ and hence $\mathtt{at}(l_2, j') \notin \mathtt{Ba}_{(L,R)}(X)$.

- $j > |l_1| + h$. Then

$$\begin{array}{ll} & LHS \\ =_{\mathcal{W}} & \underline{j-h}[cons(a^i_1, \ldots, a^i_{|l_1|}, shift^{|l_i|+|l_1 \setminus \mathtt{Ba}_{(L,R)}(X)|})][shift]^h \\ =_{\mathcal{W}} & \underline{j - h - |l_1| + |l_i| + |l_1 \setminus \mathtt{Ba}_{(L,R)}(X)| + h} \\ = & \underline{j - |l_1| + |l_i| + |l_1 \setminus \mathtt{Ba}_{(L,R)}(X)|} \\ = & \underline{j - |l_2| + |l_i| + |l_2 \setminus \mathtt{Ba}_{(L,R)}(X)|} \\ =_{\mathcal{W}} & \underline{j}[lift^h(cons(b^i_1, \ldots, b^i_{|l_2|}, shift^{|l_i|+|l_2 \setminus \mathtt{Ba}_{(L,R)}(X)|}))] \\ =_{\mathcal{W}} & \underline{j-h}[cons(c_1, \ldots, c_{|l_1|}, shift^{|l_1|})][shift]^h[lift^h(cons(b^i_1, \ldots, b^i_{|l_2|}, shift^{|l_i|+|l_2 \setminus \mathtt{Ba}_{(L,R)}(X)|}))] \\ =_{\mathcal{W}} & RHS \end{array}$$

– $\rho X = f(o_1, \ldots, o_n)$. We use the induction hypothesis.

– $\rho X = \xi(o_1, \ldots, o_n)$. Then by the induction hypothesis we have

$$\begin{aligned} & o_j[\mathit{lift}^{h+1}(\mathit{cons}(a^i_1, \ldots, a^i_{|l_1|}, \mathit{shift}^{|l_i|+|l_1 \backslash \mathtt{Ba}_{(L,R)}(X)|}))] \\ =_{\mathcal{W}} & o_j[\mathit{lift}^{h+1}(\mathit{cons}(c_1, \ldots, c_{|l_1|}, \mathit{shift}^{|l_1|}))][\mathit{lift}^{h+1}(\mathit{cons}(b^i_1, \ldots, b^i_{|l_2|}, \mathit{shift}^{|l_i|+|l_2 \backslash \mathtt{Ba}_{(L,R)}(X)|}))] \end{aligned}$$

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

Proposition 1. *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.*

Proof. Let $(L_1, R_1) =_{def} \mathcal{C}_P(L, R)$ and $(L_2, R_2) =_{def} \mathcal{C}_Q(L, R)$. Suppose $a \longrightarrow_{(L_1, R_1)} b$. Then there exists a context E and an assignment ρ such that $a =_{\mathcal{W}} E[\rho(L_1)]$ and $b =_{\mathcal{W}} E[\rho(R_1)]$.

For all $X \in \mathit{NMVar}(L)$ define the assignment η as follows: $\overline{\eta}X =_{def} \rho(X)[s]$ where $s = \mathit{cons}(c_1, \ldots, c_{|l_1|}, \mathit{shift}^{|l_1|})$. Consider now an occurrence of a t-metavariable $X_{l_i} \in (L, R)$ where $\{X_{l_1}, \ldots, X_{l_n}\}$ are all the X-based t-metavariables in (L, R).

– If $\mathtt{Ba}_{(L,R)}(X) = \emptyset$ then both conversions shall convert X_{l_i} to the term $X[\mathit{shift}^{|l_i|}]$. 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:
$X[\mathit{cons}(a^i_1, \ldots, a^i_{|l_1|}, \mathit{shift}^{|l_i|+|l_1 \backslash \mathtt{Ba}_{(L,R)}(X)|})]$ on one hand, and $X[\mathit{cons}(b^i_1, \ldots, b^i_{|l_2|}, \mathit{shift}^{|l_i|+|l_2 \backslash \mathtt{Ba}_{(L,R)}(X)|})]$, on the other. Here we may apply Lemma 5 and obtain:
$(\rho X)[\mathit{cons}(a^i_1, \ldots, a^i_{|l_1|}, \mathit{shift}^{|l_i|+|l_1 \backslash \mathtt{Ba}_{(L,R)}(X)|})] =_{\mathcal{W}} (\eta X)[\mathit{cons}(b^i_1, \ldots, b^i_{|l_2|}, \mathit{shift}^{|l_i|+|l_2 \backslash \mathtt{Ba}_{(L,R)}(X)|})]$.
Note that if conversion (a) deployed the identity optimization then $\mathit{cons}(a^i_1, \ldots, a^i_{|l_1|}, \mathit{shift}^{|l_i|+|l_1 \backslash \mathtt{Ba}_{(L,R)}(X)|}) = \mathit{cons}(\underline{1}, \ldots, \underline{|l_1|}, \mathit{shift}^{|l_1|})$ and X_{l_i} is converted to X and we may use the fact that
$\rho X =_{\mathcal{W}} (\rho X)[\mathit{cons}(\underline{1}, \ldots, \underline{|l_i|}, \mathit{shift}^{|l_i|})]$ and Lemma 5 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. $a \longrightarrow_{(L_2, R_2)} b$.

5 Properties of the conversion

This section studies the connection between higher-order rewriting and first-order rewriting modulo. We first show in Section 5.1 how to simulate a higher-order rewrite step by first-order rewriting. Section 5.2 shows 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 rewrite systems that can be translated into first-order rewrite systems modulo an empty theory. We shall see that, for example, the λ-calculus is covered by this characterization.

5.1 Simulating Higher-Order rewriting via First-Order rewriting

In order to simulate higher-order rewriting in a first-order framework we have to deal with the conversion of valid valuations into assignments. This requires two families of index-adjustment operations, decrementors and adjusters, which pave the way for the desired result.

Definition 31 (Decrementors). *For every $i, j \geq 0$ and de Bruijn ground term a we define $\mathcal{D}_i^j(a)$ as follows:*

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

Lemma 6. *Consider a $SERS_{DB}$ rule (L, R), t-metavariables $X_l, X_k \in (L, R)$, a valuation κ valid for (L, R) and an $i \geq 0$. If*

1. *$\kappa X_l = D[a]$ for some pure context D having binder path number i,*
2. *$Value^i(l, a) = Value^i(k, b)$, 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 $Value^i(l, a) = \underline{n} = Value^i(k, b)$, we have $b = \underline{n}$ and the result holds.
    Otherwise, if $i < n \leq i + |l|$ then since by Hypothesis 2 we have $Value^i(l, \underline{n}) = \mathtt{at}(l, n-i) = Value^i(k, b)$ 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 $Value(l', \kappa X_{l'}) \neq Value(l, \kappa X_l)$ by Definition 11 (since $\mathtt{at}(l, n-i)$ occurs in $Value(l, \kappa X_l)$ but $\mathtt{at}(l, n-i)$ does not occur in $Value(l', \kappa X_{l'})$), contradicting the assumption that κ is valid.
  - $n > i + |l|$. Then $\mathcal{D}_i^{|l|}(a) = \underline{n - |l|}$. Also, since $Value^i(l, a) = x_{n-i-|l|} = Value^i(k, b)$, 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 $Value^i(l, a_j) = Value^i(k, b_j)$ for $j \in 1..n$. 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 31.
- $a = \xi(a_1, \ldots, a_n)$. 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 $Value^{i+1}(l, a_j) = Value^{i+1}(k, b_j)$ for $j \in 1..n$. Then the induction hypothesis concludes the case.

Lemma 7. *Consider a $SERS_{DB}$ rule (L,R), t-metavariables $X_l, X_k \in (L,R)$, a valuation κ valid for (L,R) and an $i \geq 0$. If*

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

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

Proof. By induction on a.

- $a = \underline{n}$. Then we have three further cases to consider:
  1. $n \leq i$. Then $\mathcal{D}_i^{|l|}(\underline{n})[lift^i(shift^{|k|})] = \underline{n}[lift^i(shift^{|k|})] =_{\mathcal{W}}^{L.1} \underline{n}$. Now by Hypothesis 2 we have $Value^i(l,\underline{n}) = \underline{n} = Value^i(k,b)$ and therefore $b = \underline{n}$ and we are done.
  2. $i < n \leq i + |l|$. Then since by Hypothesis 2 we have $Value^i(l,\underline{n}) = \mathtt{at}(l, n-i) = Value^i(k,b)$ 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 $Value(l', \kappa X_l') \neq Value(l, \kappa X_l)$ by Definition (def. chk) (since $\mathtt{at}(l,n-i)$ occurs in $Value(l,\kappa X_l)$ but $\mathtt{at}(l,n-i)$ does not occur in $Value(l',\kappa X_{l'})$), contradicting the assumption that κ is valid.
  3. $n > i + |l|$. Then

$$\begin{array}{ll} & \mathcal{D}_i^{|l|}(\underline{n})[lift^i(shift^{|k|})] \\ = & (\underline{n - |l|})[lift^i(shift^{|k|})] \\ =_{\mathcal{W}}^{L.1} & (\underline{n - |l| - i})[shift^{|k|}][shift]^i \\ =_{\mathcal{W}}^{Def.19(9)} & (\underline{n - |l| - i + |k|})[shift]^i \\ =_{\mathcal{W}} & \underline{n - |l| + |k|} \end{array}$$

     The last equality follows from i applications of Definition 19(9).
     Now by Hypothesis 2 we have $Value^i(l,\underline{n}) = x_{n-i-|l|} = Value^i(k,b)$ 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 $\mathcal{D}_i^{|l|}(a)[lift^i(shift^{|k|})] =_{\mathcal{W}} f(\mathcal{D}_i^{|l|}(a_1)[lift^i(shift^{|k|})], \ldots, \mathcal{D}_i^{|l|}(a_n)[lift^i(shift^{|k|})])$ by condition 2 of Definition 18.
  Now by Hypothesis 2 we have that $b = f(b_1, \ldots, b_n)$ with $Value^i(l,a_j) = Value^i(k,b_j)$ for $j \in 1..n$. Then the induction hypothesis yields $\mathcal{D}_i^{|l|}(a_j)[lift^i(shift^{|k|})] =_{\mathcal{W}} b_j$ for $j \in 1..n$ and we may conclude the case.
- $a = \xi(a_1, \ldots, a_n)$. Then $\mathcal{D}_i^{|l|}(a)[lift^i(shift^{|k|})] =_{\mathcal{W}} \xi(\mathcal{D}_{i+1}^{|l|}(a_1)[lift^{i+1}(shift^{|k|})], \ldots, \mathcal{D}_{i+1}^{|l|}(a_n)[lift^{i+1}(shift^{|k|})]$ by condition 2 of Definition 18.
  Now by Hypothesis 2 we have that $b = \xi(b_1, \ldots, b_n)$ with $Value^{i+1}(l,a_j) = Value^{i+1}(k,b_j)$ for $j \in 1..n$. Then the induction hypothesis concludes the case.

Definition 32 (Adjusters). *Let X_l be a pivot t-metavariable in (L,R), $i \geq 1$, a a de Bruijn ground term and let $cons(c_1,\ldots,c_{|l|}, shift^{|l|+|l\setminus \mathtt{Ba}_{(L,R)}(X)|})$ be the index-adjusting substitution corresponding to X_l. Then $\mathcal{A}^l_i(a)$ is defined as follows:*

$$\mathcal{A}^l_i(\underline{n}) =_{def} \begin{cases} \underline{n} & \text{if } n \leq i \\ \underline{n} & \text{if } \mathtt{at}(l, n-i) \in \mathtt{Ba}_{(L,R)}(X) \text{ and } 0 < n-i \leq |l| \\ \mathtt{undefined} & \text{if } \mathtt{at}(l, n-i) \notin \mathtt{Ba}_{(L,R)}(X) \text{ and } 0 < n-i \leq |l| \\ \underline{\mathtt{pos}(n-i, c_1 \ldots c_{|l|}) + i} & \text{if } |l| < n-i \leq |l| + |l \setminus \mathtt{Ba}_{(L,R)}(X)| \\ \underline{n - |l \setminus \mathtt{Ba}_{(L,R)}(X)|} & \text{if } n-i > |l| + |l \setminus \mathtt{Ba}_{(L,R)}(X)| \end{cases}$$

$$\mathcal{A}^l_i(f(a_1 \ldots a_n)) =_{def} f(\mathcal{A}^l_i(a_1) \ldots \mathcal{A}^l_i(a_n))$$
$$\mathcal{A}^l_i(\xi(a_1 \ldots a_n)) =_{def} \xi(\mathcal{A}^l_{i+1}(a_1) \ldots \mathcal{A}^l_{i+1}(a_n))$$

Lemma 8 (Well-definedness of Adjusters).

Consider a $SERS_{DB}$ rule (L,R) and some pivot set P for (L,R). Let $X_l \in (L,R)$ be the X-based pivot t-metavariable for some $X \in NMVar(L)$, 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 $\mathcal{A}^l_i(a)$ is defined.

Proof. By induction on a.

- $a = \underline{n}$. Then 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 be a $X_{l'}$ in L such that $\mathtt{at}(l, n-i) \notin l'$. Consequently $Value(l, \kappa X_l) \neq Value(l', \kappa X_{l'})$ since $\mathtt{at}(l, n-i)$ occurs in $Value(l, \kappa X_l)$ but $\mathtt{at}(l, n-i)$ does not occur in $Value(l', \kappa X_{l'})$. 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 $\mathtt{pos}(n - i, c_1 \ldots c_{|l|})$ is defined. Now let $r = |l \setminus \mathtt{Ba}_{(L,R)}(X)|$ then by Definition 28 there are subindices $j_1 < \ldots < j_r$ such that $c_{j_1} = |l|+1+\mathtt{Sh}((L,R), X_l, j_1), \ldots, c_{j_r} = |l| + 1 + \mathtt{Sh}((L,R), X_l, j_r)$. By noting that $1 + \mathtt{Sh}((L,R), X_l, j_r) = r$ we are done.
  4. $n - i > |l| + |l \setminus \mathtt{Ba}_{(L,R)}(X)|$. This case presents no problems.
- $a = f(a_1, \ldots, a_n)$. Then $\mathcal{A}^l_i(a) = f(\mathcal{A}^l_i(a_1), \ldots, \mathcal{A}^l_i(a_n))$ and we may conclude using the induction hypothesis.
- $a = \xi(a_1, \ldots, a_n)$. Then $\mathcal{A}^l_i(a) = \xi(\mathcal{A}^l_{i+1}(a_1), \ldots, \mathcal{A}^l_{i+1}(a_n))$ and we may conclude using the induction hypothesis.

Lemma 9. *Consider a $SERS_{DB}$ rule (L,R) and some pivot set P for (L,R). Let $X_l \in (L,R)$ be the X-based pivot t-metavariable for some $X \in NMVar(L)$, 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. $Value^i(l,a) = Value^i(k,b)$, *and*
3. *The binding allowance of* X *in* (L,R) *is not empty (i.e.* $\mathtt{Ba}_{(L,R)}(X) \neq \emptyset$).

then $\mathcal{A}_i^l(a)[lift^i(s)] =_{\mathcal{W}} b$ *where* $s = cons(p_1, \ldots, p_{|l|}, shift^{|k|+|l \setminus \mathtt{Ba}_{(L,R)}(X)|})$ *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.

- $a = \underline{n}$. Then we have four further cases to consider:
  1. $n \leq i$. Then $\mathcal{A}_i^l(\underline{n})[lift^i(cons(p_1, \ldots, p_{|l|}, shift^j))] = \underline{n}[lift^i(cons(p_1, \ldots, p_{|l|}, shift^j))] =_{\mathcal{W}}^{L,1} \underline{n}$.
     Now by Hypothesis 2 we have $Value^i(l, \underline{n}) = \underline{n} = Value^i(k,b)$ and therefore $b = \underline{n}$ and we are done.
  2. $i < n \leq i + |l|$. Here we consider the two cases:
     - $\mathtt{at}(l, n-i) \in \mathtt{Ba}_{(L,R)}(X)$. Then

$$\begin{array}{ll} & \mathcal{A}_i^l(\underline{n})[lift^i(cons(p_1, \ldots, p_{|l|}, shift^j))] \\ = & \underline{n}[lift^i(cons(p_1, \ldots, p_{|l|}, shift^j))] \\ =_{\mathcal{W}}^{L,1} & (\underline{n-i})[cons(p_1, \ldots, p_{|l|}, shift^j)][shift]^i \\ =_{\mathcal{W}} & p_{n-i}[shift]^i \\ =_{\mathcal{W}} & p_{n-i} + i \end{array}$$

       So we are left to verify that $p_{n-i} + i = b$.
       Now by Hypothesis 2 we have $Value^i(l, \underline{n}) = \mathtt{at}(l, n-i) = Value^i(k,b)$ 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 p_{n-i} might "come from".
       (a) $n - i = \mathtt{pos}(\beta_h, l)$ with $\beta_h \in \mathtt{Ba}_{(L,R)}(X)$ and $p_{n-i} = \mathtt{pos}(\beta_h, k)$. But then by Hypothesis 2 and the fact that k is a simple label we must have $p_{n-i} = \underline{m-i}$, 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 $p_{n-i} = \underline{n-i}$ and we have $n - i + i = n$.
     - $\mathtt{at}(l, n-i) \notin \mathtt{Ba}_{(L,R)}(X)$. By Well-definedness of Adjusters (Lemma 8) this case is not possible.
  3. $|l| < n - i \leq |l| + |l \setminus \mathtt{Ba}_{(L,R)}(X)|$. Then

$$\begin{array}{ll} & \mathcal{A}_i^l(\underline{n})[lift^i(cons(p_1, \ldots, p_{|l|}, shift^j))] \\ = & \underline{(\mathtt{pos}(n-i, c_1 \ldots c_{|l|}) + i)}[lift^i(cons(p_1, \ldots, p_{|l|}, shift^j))] \\ =_{\mathcal{W}} & \underline{\mathtt{pos}(n-i, c_1 \ldots c_{|l|})}[cons(p_1, \ldots, p_{|l|}, shift^j)][shift]^i \\ =_{\mathcal{W}} & p_r[shift]^i \\ =_{\mathcal{W}} & p_r + i \end{array}$$

       where $r = \mathtt{pos}(n-i, c_1 \ldots c_{|l|})$. Note that Lemma 8 is used here.
       So we are left to verify that $p_r + i = b$.
       We must consider where p_r might "come from":

(a) $r = \mathtt{pos}(\beta_h, l)$ with $\beta_h \in \mathtt{Ba}_{(L,R)}(X)$ and $p_r = \mathtt{pos}(\beta_h, k)$. Then clearly, $\mathtt{at}(l, r) \in \mathtt{Ba}_{(L,R)}(X)$. However, since $r = \mathtt{pos}(n-i, c_1 \ldots c_{|l|})$ this means that $c_r = \underline{n-i}$. Also, recall that we are currently considering the case $c_r = n - i > |l|$. But then by Definition 28 $\mathtt{at}(l, r) \notin \mathtt{Ba}_{(L,R)}(X)$ contradicting our knowledge of the opposite fact.

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

Note that in the particular case where $X_k = X_l$ then $p_r = \underline{n-i}$ and we have $n - i + i = n$.

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

$$\begin{array}{ll} & \mathcal{A}^l_i(\underline{n})[lift^i(cons(p_1, \ldots, p_{|l|}, shift^j))] \\ = & \underline{(n - |l \setminus \mathtt{Ba}_{(L,R)}(X)|)}[lift^i(cons(p_1, \ldots, p_{|l|}, shift^j))] \\ =_{\mathcal{W}} & \underline{(n - |l \setminus \mathtt{Ba}_{(L,R)}(X)| - i)}[cons(p_1, \ldots, p_{|l|}, shift^j)][shift]^i \\ =_{\mathcal{W}} & \underline{n - |l \setminus \mathtt{Ba}_{(L,R)}(X)| - i - |l| + |k| + |l \setminus \mathtt{Ba}_{(L,R)}(X)| + i} \\ = & \underline{n - |l| + |k|} \end{array}$$

Note that in the particular case that $X_k = X_l$ we have $k = l$ and the result holds directly. Otherwise, by Hypothesis 2 we have $Value^i(l, \underline{n}) = x_{n-i-|l|} = Value^i(k, b)$ 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.

– $a = f(a_1, \ldots, a_n)$. Then

$$\begin{array}{l} \mathcal{A}^l_i(a)[lift^i(cons(p_1, \ldots, p_{|l|}, shift^j))] \\ =_{\mathcal{W}} f(\mathcal{A}^l_i(a_1)[lift^i(cons(p_1, \ldots, p_{|l|}, shift^j))], \ldots, \mathcal{A}^l_i(a_n)[lift^i(cons(p_1, \ldots, p_{|l|}, shift^j))]) \end{array}$$

Now by Hypothesis 2 we have that $b = f(b_1, \ldots, b_n)$ with $Value^i(l, a_j) = Value^i(k, b_j)$ for $j \in 1..n$. Then the induction hypothesis concludes the case.

– $a = \xi(a_1, \ldots, a_n)$. Then

$$\begin{array}{l} \mathcal{A}^l_i(a)[lift^i(cons(p_1, \ldots, p_{|l|}, shift^j))] \\ =_{\mathcal{W}} \xi(\mathcal{A}^l_{i+1}(a_1)[lift^{i+1}(cons(p_1, \ldots, p_{|l|}, shift^j))], \ldots, \mathcal{A}^l_{i+1}(a_n)[lift^{i+1}(cons(p_1, \ldots, p_{|l|}, shift^j))]) \end{array}$$

Now by Hypothesis 2 we have that $b = \xi(b_1, \ldots, b_n)$ with $Value^{i+1}(l, a_j) = Value^{i+1}(k, b_j)$ for $j \in 1..n$. Then 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. This makes use of decrementors and adjusters.

Definition 33 (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 assignment ρ determined by $\overline{\rho}$ where for each $X \in NMVar(L)$ we define $\overline{\rho}$ as:*

- *Case* $\mathtt{Ba}_{(L,R)}(X) = \emptyset$. *Then* $\overline{\rho}(X) =_{def} \mathcal{D}_0^{|l|}(\kappa X_l)$ *where X_l is any t-metavariable from L. Note that Lemma 6 guarantees that this is a correct definition (take $D = E = \Box$).*
- *Case* $\mathtt{Ba}_{(L,R)}(X) = \{\beta_1, \ldots, \beta_n\}$ *with $n > 0$. Then we define* $\overline{\rho}(X) =_{def} \mathcal{A}_0^l(\kappa X_l)$ *where X_l is the X-based pivot t-metavariable as dictated by P.*

Lemma 10. *Let (L,R) be a $SERS_{DB}$-rewrite rule, κ a valid valuation for (L,R) and ρ the conversion of κ via P for some pivot set P for (L,R). If $L = C[M]$ then $\rho(\mathcal{C}_P^{(L,R)}(M)) =_{\mathcal{W}} \kappa M$. Likewise, if $R = C[M]$ then $\rho(\mathcal{C}_P^{(L,R)}(M)) =_{\mathcal{W}} \kappa M$.*

Proof. Both items are proved by induction on M.

- $M = \underline{n}$. Then $LHS = \rho(\underline{n}) = \underline{n} = \kappa\underline{n} = RHS$.
- $M = X_k$. Note that since X_k is a subterm of a metaterm (i.e. a well-formed pre-metaterm [5]) k is a simple label. According to Definition 29 we have three subcases to consider:
  1. $\mathtt{Ba}_{(L,R)}(X) = \emptyset$. Then

$$\begin{array}{lll} & & LHS \\ = & & \rho(X[shift^{|k|}]) \\ = & & \rho(X)[shift^{|k|}] \\ =_{Def\ 33} & & \mathcal{D}_0^{|l|}(\kappa X_l)[shift^{|k|}] \\ =_{\mathcal{W}}^{L.\ 7} & & \kappa X_k \end{array}$$

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

$$\begin{array}{lll} & & LHS \\ = & & \rho(X[cons(b_1, \ldots, b_{|l|}, shift^j)]) \\ = & & \rho(X)[cons(b_1, \ldots, b_{|l|}, shift^j)] \\ = & & \mathcal{A}_0^l(\kappa X_l)[cons(b_1, \ldots, b_{|l|}, shift^j)] \\ =_{\mathcal{W}}^{L.\ 9} & & \kappa X_k \end{array}$$

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

$$
\begin{array}{ll}
 & LHS \\
= & \rho(X) \\
= & \mathcal{A}_0^l(\kappa X_l) \\
=_{\mathcal{W}} & \mathcal{A}_0^l(\kappa X_l)[cons(\underline{1},\ldots,\underline{|l|}, shift^{|l|})] \\
=_{\mathcal{W}}^{L.\ 9} & \kappa X_k
\end{array}
$$

Note that the third equality holds by the fact that $cons(\underline{1},\ldots,\underline{|l|}, shift^{|l|})$ behaves as the identity substitution.

- $M = f(M_1,\ldots,M_n)$. Then

$$
\begin{array}{ll}
 & LHS \\
= & f(\rho(\mathcal{C}_P^{(L,R)}(M_1)),\ldots,\rho(\mathcal{C}_P^{(L,R)}(M_n))) \\
=_{\mathcal{W}}^{i.h.} & f(\kappa M_1,\ldots,\kappa M_n) \\
= & RHS
\end{array}
$$

- $M = \xi(M_1,\ldots,M_n)$. Then

$$
\begin{array}{ll}
 & LHS \\
= & \xi(\rho(\mathcal{C}_P^{(L,R)}(M_1)),\ldots,\rho(\mathcal{C}_P^{(L,R)}(M_n))) \\
=_{\mathcal{W}}^{i.h.} & \xi(\kappa M_1,\ldots,\kappa M_n) \\
= & RHS
\end{array}
$$

- $M = M_1[\![M_2]\!]$. This case is considered for the second item only since the de Bruijn metasubstitution operator may not occur on the *LHS* of a $SERS_{DB}$-rewrite rule.

$$
\begin{array}{ll}
 & LHS \\
= & \rho(\mathcal{C}_P^{(L,R)}(M_1[\![M_2]\!])) \\
= & \rho(\mathcal{C}_P^{(L,R)}(M_1))[cons(\rho(\mathcal{C}_P^{(L,R)}(M_2)), id)] \\
=_{\mathcal{W}}^{i.h.} & \kappa M_1[cons(\kappa M_2, id)] \\
=^{L.\ 3} & \kappa M_1\{1 \leftarrow \kappa M_2\} \\
= & \kappa(M_1[\![M_2]\!]) \\
= & RHS
\end{array}
$$

Proposition 2 ($fo(\mathcal{R})_{\mathcal{W}}$ simulates $\mathcal{R}$). *Let $\mathcal{R}$ be an $SERS_{DB}$ and let $fo(\mathcal{R})_{\mathcal{W}}$ be its first-order version. Suppose $a \longrightarrow_{\mathcal{R}} b$ then*

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

Proof. For the first item, suppose $a \longrightarrow_{\mathcal{R}} b$. 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 ρ be the conversion of κ via P (Definition 33). We must now verify that

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

This holds by applying Lemma 10. Thus from $\rho(L') =_{\mathcal{W}} \kappa L$ and $\rho(R') =_{\mathcal{W}} \kappa R$ we have $E[\rho(L')] =_{\mathcal{W}} E[\kappa L]$ and $E[\rho(R')] =_{\mathcal{W}} E[\kappa R]$, respectively. Finally, we have on one hand $a = E[\kappa L] =_{\mathcal{W}} E[\rho(L')]$, so $a =_{\mathcal{W}} E[\rho(L')]$, and on the other, $b = E[\kappa R] =_{\mathcal{W}} E[\rho(R')]$, so $b =_{\mathcal{W}} E[\rho(R')]$.

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 assignment. Thus $\rho(L') = \kappa L$. And $\rho(R') \xrightarrow{*}_{\mathcal{W}} \kappa R$ since κR is a pure term. Therefore we have $a = E[\kappa L] = E[\rho(L')] \longrightarrow_{(L',R')} E[\rho(R')] \xrightarrow{*}_{\mathcal{W}} E[\kappa R]$.

5.2 Conservativity

We now wish to prove that reductions in an *ExERS* or *FExERS* $fo(\mathcal{R})_{\mathcal{W}}$ may be projected into reductions 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 35).

Definition 34 (Skeleton of a metaterm). *The skeleton of a metaterm M is the context obtained by replacing every t-metavariable X_l by some hole $\Box_{X_l}$. Note that if there is more than one occurrence of a t-metavariable then there shall be more than one occurrence of its associated hole. We shall use* $\texttt{skel}(M)$ *to denote the skeleton of M.*

Remark 1. Let M be a pre-metaterm and suppose $\mathcal{WF}_k(M)$. Then any t-metavariable occurring in M must be of the form X_{lk} for some label l. Moreover lk is a simple label.

Lemma 11. *Let M be a pre-metaterm and suppose $\mathcal{WF}_k(M)$. Consider a valuation κ such that $MVar(M) \subseteq Dom(\kappa)$. Then $\mathcal{W}((\kappa M)[lift^{|k|}(s)]) = \iota_k M$ where ι_k is a valuation defined as: $\iota_k(X_{lk}) =_{def} \mathcal{W}((\kappa X_{lk})[lift^{|lk|}(s)])$ for all l such that X_{lk} occurs in M.*

Proof. By induction on M.

- $M = \underline{n}$. Note that since $\mathcal{WF}_k(\underline{n})$ we have $n \leq |k|$. Then $LHS = \mathcal{W}((\kappa\underline{n})[lift^{|k|}(s)]) = \mathcal{W}(\underline{n}[lift^{|k|}(s)]) = \underline{n} = \iota_k\underline{n} = RHS$.
- $M = X_p$. Then since $\mathcal{WF}_k(X_p)$ we have $k = p$ and

$$\begin{aligned} & LHS \\ &= \mathcal{W}((\kappa X_k)[lift^{|k|}(s)]) \\ &= \iota_k M \end{aligned}$$

- $M = f(M_1, \ldots, M_n)$. Then

$$
\begin{array}{ll}
 & LHS \\
=^{Def.\ 19(3,4)} & f(\mathcal{W}((\kappa M_1)[lift^{|k|}(s)]), \ldots, \mathcal{W}((\kappa M_n)[lift^{|k|}(s)])) \\
=^{i.h.}_{\mathcal{W}} & f(\iota_k^1 M_1, \ldots, \iota_k^n M_n) \\
= & f(\iota_k M_1, \ldots, \iota_k M_n) \\
= & RHS
\end{array}
$$

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 $\iota_k^j(X_p) = \iota_k^{j'}(X_p)$.

– $M = \xi(M_1, \ldots, M_n)$. By hypothesis there is an α such that $\mathcal{WF}_{\alpha k}(M_i)$ for all $i \in 1..n$. Then

$$
\begin{array}{ll}
 & LHS \\
=^{Def.\ 19(3,4)} & \xi(\mathcal{W}((\kappa M_1)[lift^{|k|+1}(s)]), \ldots, \mathcal{W}((\kappa M_n)[lift^{|k|+1}(s)])) \\
=^{i.h.}_{\mathcal{W}} & \xi(\iota_{\alpha k}^1 M_1, \ldots, \iota_{\alpha k}^n M_n) \\
= & \xi(\iota_{\alpha k} M_1, \ldots, \iota_{\alpha k} M_n)
\end{array}
$$

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 t-metavariable in M_i has the form $X_{p\alpha k}$ for some label p we have $\iota_k(M_i) = \iota_{\alpha k} M_i$ 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 t-metavariable in M_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:

$$
\begin{array}{ll}
 & \xi(\iota_{\alpha k} M_1, \ldots, \iota_{\alpha k} M_n) \\
= & \xi(\iota_k M_1, \ldots, \iota_k M_n) \\
= & \iota_k M
\end{array}
$$

– $M = M_1[\![M_2]\!]$. By hypothesis there is an α such that $\mathcal{WF}_{\alpha k}(M_1)$, and $\mathcal{WF}_k(M_2)$. Then

$$
\begin{array}{ll}
 & LHS \\
= & \mathcal{W}((\kappa(M_1[\![M_2]\!]))[lift^{|k|}(s)]) \\
= & \mathcal{W}((\kappa M_1\{1 \leftarrow \kappa M_2\})[lift^{|k|}(s)]) \\
=_{L.3} & \mathcal{W}((\kappa M_1[cons(\kappa M_2, id)])[lift^{|k|}(s)]) \\
=_{L.4} & \mathcal{W}((\kappa M_1)[lift^{|k|+1}(s)][cons((\kappa M_2)[lift^{|k|}(s)], id)]) \\
= & \mathcal{W}(\mathcal{W}((\kappa M_1)[lift^{|k|+1}(s)])[cons(\mathcal{W}((\kappa M_2)[lift^{|k|}(s)]), id)]) \\
= & \mathcal{W}((\kappa M_1)[lift^{|k|+1}(s)])\{1 \leftarrow \mathcal{W}((\kappa M_2)[lift^{|k|}(s)])\} \\
=_{i.h.} & \iota_{\alpha k}(M_1)\{1 \leftarrow \iota_k(M_2)\} \\
= & \iota_k(M_1)\{1 \leftarrow \iota_k(M_2)\} \\
= & \iota_k(M_1[\![M_2]\!])
\end{array}
$$

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

Lemma 12. *Let a, b be pure terms and let (L, R) be a $SERS_{DB}$-rewrite rule. If $a \longrightarrow_{(L,R)} b$ then for any term s of sort S we have $\mathcal{W}(a[s]) \longrightarrow_{(L,R)} \mathcal{W}(b[s])$.*

Proof. Suppose $a \longrightarrow_{(L,R)} b$. Then there is a pure context E and a valuation κ valid for (L,R) such that $a = E[\kappa L]$ and $b = E[\kappa R]$. We consider two cases:

- The reduction is at the root. Let $G = \mathtt{skel}(L)$. Then $a = G[\kappa X_{l_1}^{i_1}]_{X_{l_1}^{i_1}} .. [\kappa X_{l_n}^{i_n}]_{X_{l_n}^{i_n}}$ where $X_{l_1}^{i_1}, \ldots, X_{l_n}^{i_n}$ are all the t-metavariables in L.
  Then $\mathcal{W}(a[s]) = G[\mathcal{W}((\kappa X_{l_1}^{i_1})[\mathit{lift}^{|l_1|}(s)])]_{X_{l_1}^{i_1}} .. [\mathcal{W}((\kappa X_{l_n}^{i_n})[\mathit{lift}^{|l_1|}(s)])]_{X_{l_n}^{i_n}}$.
  So define $\iota X_{l_j}^{i_j} =_{def} \mathcal{W}((\kappa X_{l_j}^{i_j})[\mathit{lift}^{|l_j|}(s)])$. Then $\mathcal{W}(a[s]) \longrightarrow_{(L,R)} \iota(R) =_{L.\ 11(k=\epsilon)} \mathcal{W}((\kappa R)[s]) = \mathcal{W}(b[s])$.
- The reduction is internal.
  - $a = f(a_1, \ldots, a_n)$. Use the i.h.
  - $a = \xi(a_1, \ldots, a_n)$. Use the i.h.

Lemma 13. *Let (L,R) be a $SERS_{DB}$-rewrite rule, let $(L',R') = \mathcal{C}_P(L,R)$ for some pivot set P for (L,R), let ρ be an assignment for (L',R').*

Define the valuation κ as:

$$\kappa X_k =_{def} \mathcal{W}(\rho(\mathcal{C}_P^{(L,R)}(X_k)))$$

If $L = C[M]$ then $\mathcal{W}(\rho(\mathcal{C}_P^{(L,R)}(M))) = \kappa M$. Likewise, if $R = C[M]$ then $\mathcal{W}(\rho(\mathcal{C}_P^{(L,R)}(M))) = \kappa M$.

Proof. Both items are proved by induction on M.

- $M = \underline{n}$. Then $LHS = \mathcal{W}(\rho(\underline{n})) = \underline{n} = \kappa\underline{n} = RHS$.
- $M = X_k$. Then

$$\begin{array}{ll} & LHS \\ = & \mathcal{W}(\rho(\mathcal{C}_P^{(L,R)}(X_k))) \\ =^{hypothesis} & \kappa X_k \end{array}$$

- $M = f(M_1, \ldots, M_n)$. Then

$$\begin{array}{ll} & LHS \\ =^{Def.\ 19(3)} & f(\mathcal{W}(\rho(\mathcal{C}_P^{(L,R)}(M_1))), \ldots, \mathcal{W}(\rho(\mathcal{C}_P^{(L,R)}(M_n)))) \\ =^{i.h.}_{\mathcal{W}} & f(\kappa M_1, \ldots, \kappa M_n) \\ = & RHS \end{array}$$

- $M = \xi(M_1, \ldots, M_n)$. Then

$$\begin{array}{ll} & LHS \\ =^{Def.\ 19(3)} & \xi(\mathcal{W}(\rho(\mathcal{C}_P^{(L,R)}(M_1))), \ldots, \mathcal{W}(\rho(\mathcal{C}_P^{(L,R)}(M_n)))) \\ =^{i.h.}_{\mathcal{W}} & \xi(\kappa M_1, \ldots, \kappa M_n) \\ = & RHS \end{array}$$

– $M = M_1[\![M_2]\!]$. This case is considered for the second item only since the de Bruijn metasubstitution operator may not occur on the *LHS* of a $SERS_{DB}$-rewrite rule.

$$\begin{array}{lll} & LHS & \\ = & \mathcal{W}(\rho(\mathcal{C}_P^{(L,R)}(M_1[\![M_2]\!]))) & \\ = & \mathcal{W}(\rho(\mathcal{C}_P^{(L,R)}(M_1))[cons(\rho(\mathcal{C}_P^{(L,R)}(M_2)), id)]) & \\ =^{Def.\ 19(1)} & \mathcal{W}(\mathcal{W}(\rho(\mathcal{C}_P^{(L,R)}(M_1)))[cons(\mathcal{W}(\rho(\mathcal{C}_P^{(L,R)}(M_2))), id)]) & \\ =^{i.h.}_{\mathcal{W}} & \mathcal{W}(\kappa M_1[cons(\kappa M_2, id)]) & \\ =^{L.\ 3} & \kappa M_1\{1 \leftarrow \kappa M_2\} & \\ = & \kappa(M_1[\![M_2]\!]) & \\ = & RHS & \end{array}$$

Lemma 14 (Reduction interpretation).
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 $o \longrightarrow_{(L',R')} o'$ then*

1. *if o is of sort* `T` *then $\mathcal{W}(o) \xrightarrow{*}_{(L,R)} \mathcal{W}(o')$.*
2. *for every pure term d of sort* `T` *such that $d[o]$ is a term of sort* `T`*, and every $n \geq 0$, we have $\mathcal{W}(d[lift^n(o)]) \xrightarrow{*}_{(L,R)} \mathcal{W}(d[lift^n(o')])$*

Proof. We show simultaneously the two items by induction on the lexicographic ordering (o, d), where o and d denote respectively the ordering induced by their structures.

– o is a de Bruijn index or a substitution constant. Then both items holds 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. As for the first item:
  - the reduction is internal. Then we use the i.h.
  - the reduction is at the root. Then $o = \rho L'$. Define κ as:

$$\kappa X_k =_{def} \mathcal{W}(\rho(\mathcal{C}_P^{(L,R)}(X_k)))$$

    Then $\mathcal{W}(\rho L') = \kappa L$ by Lemma 13. So $\kappa L \longrightarrow_{(L,R)} \kappa R =_{L.\ 13} \mathcal{W}(\rho R') = \mathcal{W}(o')$.
– $o = a[v]$. There is nothing to prove for the second item. We consider two cases for the first property:
  - $o' = a'[v]$ with $a \longrightarrow a'$. By the i.h. $\mathcal{W}(a) \xrightarrow{*}_{(L,R)} \mathcal{W}(a')$. Then $\mathcal{W}(a[v]) = \mathcal{W}(\mathcal{W}(a)[v]) \xrightarrow{*}_{(L,R)} \mathcal{W}(\mathcal{W}(a')[v])$ by applying Lemma 12 many times.
  - $o' = a[v']$ with $v \longrightarrow v'$. Since $\mathcal{W}(a)$ is a pure term we have that $\mathcal{W}(o) = \mathcal{W}(\mathcal{W}(a)[v]) \xrightarrow{*}_{(L,R)} \mathcal{W}(\mathcal{W}(a)[v']) = \mathcal{W}(o')$ by the induction hypothesis of item 2 since $(v, \mathcal{W}(a)) < (a[v], \mathcal{W}(a))$.

– 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 $s_j \longrightarrow_{(L',R')} s'_j$, then 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.

- $d = f(o_1, \ldots, o_n)$ or $d = \xi(o_1, \ldots, o_n)$ then the property holds by the induction hypothesis since $(o_i, o) < (d, o)$ for $i \in 1..n$.
- $d = \underline{m}$. Then we must verify that

$$\forall n \geq 0 \quad \mathcal{W}(\underline{m}[\mathit{lift}^n(o)] \xrightarrow{*}_{(L,R)} \mathcal{W}(\underline{m}[\mathit{lift}^n(o')]$$

We proceed by induction on n.

1. if $n = 0$, then we proceed by cases as dictated by Definition 24.
   (a) Suppose there exists a de Bruijn index $\underline{l}$, indices $i_1, \ldots, i_p$ $(p > 0)$ and 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 all $s_1 \ldots s_q$ $\underline{m}[\sigma(s_1, \ldots s_q)] =_{\mathcal{W}} \underline{l}[s_{i_1}] \ldots [s_{i_p}][u_1] \ldots [u_k]$.
      i. if $j \notin \{i_1, \ldots, i_p\}$, then the term $\mathcal{W}(\underline{m}[o'])$ is also equal to $\mathcal{W}(\underline{l}[s_{i_1}] \ldots [s_{i_p}][u_1] \ldots [u_k])$ 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 $\mathcal{W}(\underline{l}[s_{i_1}] \ldots [s'_{i_h}] \ldots [s_{i_p}][u_1] \ldots [u_k])$ and $\mathcal{W}(\underline{l}[s_{i_1}] \ldots [s_{i_{h-1}}]) = e$ is a pure term by Definition 19(2). We have $(s_{i_h}, e) < (\sigma(s_1, \ldots, s_j, \ldots, s_q), \underline{m})$, so that we can apply the induction hypothesis (2) to obtain:

$$\mathcal{W}(e[s_{i_h}]) = \mathcal{W}(e[s'_{i_h}])$$

      Now, the term $\mathcal{W}(e[s_{i_h}])$ is pure, so that we can repeatedly apply Lemma 12 to obtain:

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

   (b) Suppose that there exists an index $\underline{i}$ $(1 \leq i \leq q)$ such that for all $s_1 \ldots s_q$ we have $\underline{m}[\sigma(s_1, \ldots s_q)] =_{\mathcal{W}} s_i$.
      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 $\mathcal{W}(\underline{m}[o]) = \mathcal{W}(s_j) \xrightarrow{*}_{(L,R)} \mathcal{W}(s'_j) = \mathcal{W}(\underline{m}[o'])$.
2. if $n > 0$, then we consider two cases:
   (a) if $m \leq n$, then by Lemma 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 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 variables are equivalent with respect to our ordering (o, d), we have $(\sigma(s_1, \ldots, s_j, \ldots, s_q), \underline{m}) = (\sigma(s_1, \ldots, s_j, \ldots, s_q), \underline{m-1})$, and then the induction hypothesis on n can be applied to obtain

$$W(\underline{m-1}[\mathit{lift}^{n-1}(o)]) \xrightarrow{*}_{(L,R)} W(\underline{m-1}[\mathit{lift}^{n-1}(o')])$$

Since every $\mathcal{W}$-normal form is a pure term by Definition 19(2), we may finally apply Lemma 12, so that

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

Definition 35. *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 $a \in A$.*

Lemma 15. *Consider a $SERS_{DB}$ rewrite rule (L, R), t-metavariables X_{k_1}, X_{k_2} occurring in (L, R) and a designated pivot t-metavariable X_l. Let a be any pure term. Then for all $i \geq 0$ we have $Value^i(k_1, \mathcal{W}(a[\mathit{lift}^i(s_1)])) = Value^i(k_2, \mathcal{W}(a[\mathit{lift}^i(s_2)]))$ where $s_1 = cons(b_1, \ldots, b_{|l|}, \mathit{shift}^{|k_1|+|l \setminus \mathtt{Ba}_{(L,R)}(X)|})$ and $s_2 = cons(c_1, \ldots, c_{|l|}, \mathit{shift}^{|k_2|+|l \setminus \mathtt{Ba}_{(L,R)}(X)|})$ are the index-adjusting substitutions using pivot X_l of X_{k_1} and X_{k_2}, respectively.*

Proof. 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 1 $Value^i(k_1, \underline{n}) = \underline{n} = Value^i(k_2, \underline{n})$.
  - $i < n \leq |l| + i$. 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 28. Therefore $Value^i(k_1, b_{n-i} + i) = \beta_h = Value^i(k_2, c_{n-i} + i)$.
    - There is no $\beta_h \in \mathtt{Ba}_{(L,R)}(X)$ such that $n - i = \mathtt{pos}(\beta_h, l)$. Then $b_{n-i} = \underline{|k_1| + 1 + \mathtt{Sh}((L,R), X_l, n-i)}$ and $c_{n-i} = \underline{|k_2| + 1 + \mathtt{Sh}((L,R), X_l, n-i)}$. Hence, $Value^i(k_1, b_{n-i}+i) = x_{1+\mathtt{Sh}((L,R),X_l,n-i)} = Value^i(k_2, c_{n-i} + i)$.
  - $n > |l| + i$. Then $\mathcal{W}(a[\mathit{lift}^i(s_1)]) = \underline{n - |l| + |k_1| + |l \setminus \mathtt{Ba}_{(L,R)}(X)|}$ and $\mathcal{W}(a[\mathit{lift}^i(s_2)]) = \underline{n - |l| + |k_2| + |l \setminus \mathtt{Ba}_{(L,R)}(X)|}$. Thus $Value^i(k_1, \underline{n - |l| + |k_1| + |l \setminus \mathtt{Ba}_{(L,R)}(X)|}) = x_{n-i-|l|+|l \setminus \mathtt{Ba}_{(L,R)}(X)|} = Value^i(k_2, \underline{n - |l| + |k_2| + |l \setminus \mathtt{Ba}_{(L,R)}(X)|})$.

– $a = f(a_1, \ldots, a_n)$. Then

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

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

Corollary 2. *Consider a $SERS_{DB}$ rewrite rule (L, R), t-metavariables X_{k_1}, X_{k_2} occurring in (L, R) and a designated pivot t-metavariable X_l. Let ρ be an assignment.*

Then $Value(k_1, \mathcal{W}(\overline{\rho}(X)[s_1])) = Value(k_2, \mathcal{W}(\overline{\rho}(X)[s_2]))$ 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. Since $\mathcal{W}$is a basic substitution calculus it has unique normal forms. Also, by condition 2 of Definition 19 we have that $\mathcal{W}(a)$ is a pure term for any term a. Thus

$$\begin{array}{ll} & Value(k_1, \mathcal{W}(\overline{\rho}(X)[s_1])) \\ = & Value(k_1, \mathcal{W}(\mathcal{W}(\overline{\rho}(X))[s_1])) \\ = & Value^0(k_1, \mathcal{W}(\mathcal{W}(\overline{\rho}(X))[s_1])) \\ =^{L\ 15} & Value^0(k_2, \mathcal{W}(\mathcal{W}(\overline{\rho}(X))[s_2])) \\ = & Value(k_2, \mathcal{W}(\mathcal{W}(\overline{\rho}(X))[s_2])) \\ = & Value(k_2, \mathcal{W}(\overline{\rho}(X)[s_2])) \end{array}$$

Proposition 3 (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 $a \longrightarrow_{fo(\mathcal{R})_{\mathcal{W}}} b$ then $\mathcal{W}(a) \xrightarrow{*}_{\mathcal{R}} \mathcal{W}(b)$.

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

ExERS-**rewriting** Suppose that $a \longrightarrow_{fo(\mathcal{R})/\mathcal{W}} b$ using rewrite rule $(L', R') = \mathcal{C}_P(L, R)$ where P is a pivot set for $(L, R) \in \mathcal{R}$, a (not necessarily pure) context E and assignment ρ. Thus $a =_{\mathcal{W}} E[\rho(L')]$ and $b =_{\mathcal{W}} E[\rho(R')]$.
Now since $E[\rho(L')] \longrightarrow_{(L',R')} E[\rho(R')]$ then by Lemma 14 we may conclude that $\mathcal{W}(E[\rho(L')]) \xrightarrow{*}_{(L,R)} \mathcal{W}(E[\rho(R')])$. Also since $a =_{\mathcal{W}} E[\rho(L')])$ we know that $\mathcal{W}(a) = \mathcal{W}(E[\rho(L')]$, likewise we know that $\mathcal{W}(b) = \mathcal{W}(E[\rho(R')]$. Therefore $\mathcal{W}(a) = \mathcal{W}(E[\rho(L')] \xrightarrow{*}_{(L,R)} \mathcal{W}(E[\rho(R')]) = \mathcal{W}(b)$ as desired.

FExERS-**rewriting** Suppose $fo(\mathcal{R})$ is a *FExERS* and that $a \longrightarrow_{fo(\mathcal{R})\cup\mathcal{W}} b$. Then if $a \longrightarrow_{\mathcal{W}} b$ the result holds trivially. Thus let us assume that $a \longrightarrow_{fo(\mathcal{R})} b$ using rewrite rule $(L', R') = \mathcal{C}_P(L, R)$ where P is a pivot set for $(L, R) \in \mathcal{R}$, a (not necessarily pure) context E and assignment ρ. Then $a = E[\rho(L')]$ and

$b = E[\rho(R')]$. Now since $E[\rho(L')] \longrightarrow_{(L',R')} E[\rho(R')]$ then by Lemma 14 we may conclude that $\mathcal{W}(a) = \mathcal{W}(E[\rho(L')]) \xrightarrow{*}_{(L,R)} \mathcal{W}(E[\rho(R')]) = \mathcal{W}(b)$ as desired.

Noting that $\mathcal{W}(a) = a$ for pure terms a (Definition 19(2)) we may conclude.

Corollary 3 (Conservativity). *Let $\mathcal{R}$ be an $SERS_{DB}$. Then $fo(\mathcal{R})_{\mathcal{W}}$-rewriting is conservative over $\mathcal{R}$-rewriting.*

5.3 Essentially first-order *HORS*

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

Definition 36 (Essentially first-order *HORS*). *A $SERS_{DB}$ $\mathcal{R}$ is called* essentially first-order *if $fo(\mathcal{R})_{\mathcal{W}}$ is a FExERS for $\mathcal{W}$ a basic substitution calculus.*

Definition 37 (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 t-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 9, l_1 is simple.

Example 10. Consider the β_{db}-calculus consisting of the sole rule: $app(\lambda(X_\alpha), Z_\epsilon) \longrightarrow_{\beta_{db}} X_\alpha[\![Z_\epsilon]\!]$. The β_{db}-calculus satisfies the fo-condition.

Proposition 4 puts forward the importance of the fo-condition. Its proof relies on a close inspection of the Conversion Procedure.

Proposition 4. *Let $\mathcal{R}$ be a $SERS_{DB}$ satisfying the fo-condition. Then $\mathcal{R}$ is essentially first-order.*

Note that many results on higher-order systems (e.g. perpetuality [19], standardization [21]) 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.

6 Conclusions and future work

This work presents an encoding of higher-order term rewriting systems into first-order rewriting systems modulo an equational theory. This equational theory takes care of 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 rewrite 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 rewrite systems in [19, 21] in order to reason about their properties can be related to the fo-condition in a very simple way. This justifies that the fo-condition, even if obtained very technically in this paper, may be seen as an interpretation of what a well-behaved higher-order rewrite system is.

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 do.

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.
- Given a $SERS_{DB}$ $\mathcal{R}$ note that the *LHS*s 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 $.[\![.]\!]$) 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.
- This work has been developed in a type-free framework. The notion of type is central to Computer Science. This calls for a detailed study of the encoding process dealing with typed higher-order rewrite systems such as *HRS* [23].
- The ideas presented in this paper could be used to relax conditions in [9] where only rewrite systems with atomic propositions on the *LHS*s of rules are considered.

Acknowledgements. We thank Bruno Guillaume for helpful remarks.

References

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

2. T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133–178, 2000.
3. L. Bachmair and N. Dershowitz. Critical pair criteria for completion. *Journal of Symbolic Computation*, 6(1):1–18, 1988.
4. Z. Benaissa, D. Briaud, P. Lescanne, and J. Rouyer-Degli. λv, a calculus of explicit substitutions which preserves strong normalisation. *Journal of Functional Programming*, 6(5):699–722, 1996.
5. E. Bonelli, D. Kesner, and A. Ríos. A de Bruijn notation for higher-order rewriting. In *Proc. of the Eleventh Int. Conference on Rewriting Techniques and Applications*, Norwich, UK, July 2000.
6. R. David and B. Guillaume. A λ-calculus with explicit weakening and explicit substitutions. *Journal of Mathematical Structures in Computer Science*, 2000. To appear.
7. N. Dershowitz. Orderings for term rewriting systems. *Theoretical Computer Science*, 17(3):279–301, 1982.
8. N. Dershowitz and J-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 243–309. North-Holland, 1990.
9. G. Dowek, T. Hardin and C. Kirchner. Theorem proving modulo. Technical Report RR 3400, INRIA, 1998.
10. G. Dowek, T. Hardin, C. Kirchner and F. Pfenning. Unification via Explicit Substitutions: The Case of Higher-Order Patterns. Technical Report RR3591. INRIA. 1998.
11. G. Dowek, T. Hardin and C. Kirchner. Higher-order unification via explicit substitutions. *Information and Computation*, 157:183–235, 2000.
12. 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.
13. T. Hardin and J-J. Lévy. A confluent calculus of substitutions. In *France-Japan Artificial Intelligence and Computer Science Symposium*, 1989.
14. 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.
15. F. Kamareddine and A. Ríos. A λ-calculus à la de Bruijn with explicit substitutions. In *Proc. of the Int. Symposium on Programming Language Implementation and Logic Programming (PLILP)*, LNCS 982. 1995.
16. S. Kamin and J. J. Lévy. Attempts for generalizing the recursive path orderings. University of Illinois, 1980.
17. D. Kesner. Confluence of extensional and non-extensional lambda-calculi with explicit substitutions. *Theoretical Computer Science*, 238(1-2):183–220, 2000.
18. Z. Khasidashvili. Expression Reduction Systems. In *Proc. of I. Vekua Institute of Applied Mathematics*, volume 36, Tbilisi, 1990.
19. Z. Khasidashvili and M. Ogawa and V. van Oostrom. Uniform Normalization beyond Orthogonality. Submitted for publication, 2000.
20. P. Lescanne and J. Rouyer-Degli. Explicit substitutions with de Bruijn levels. In *Proc. of the Sixth Int. Conference on Rewriting Techniques and Applications*, LNCS 914, 1995.
21. P-A. Melliès. Description Abstraite des Systèmes de Réécriture. Thèse de l'Université Paris VII, December 1996.
22. C. Muñoz. A left-linear variant of $\lambda\sigma$. In Michael Hanus, J. Heering, and K. (Karl) Meinke, editors, *Proc. of the 6th Int. Conference on Algebraic and Logic Programming (ALP'97)*, LNCS 1298. 1997.

23. T. Nipkow. Higher-order critical pairs. in *Proc. of the Sixth Annual IEEE Symposium on Logic in Computer Science*, 1991.
24. B. Pagano. *Des Calculs de Susbtitution Explicite et leur application à la compilation des langages fonctionnels.* PhD thesis, Université Paris VI, 1998.
25. R. Pollack. Closure under alpha-conversion. In Henk Barendregt and Tobias Nipkow, editors, *Types for Proofs and Programs (TYPES)*, LNCS 806. 1993.
26. H. Zantema. Termination of Term Rewriting by Semantic Labelling. Fundamenta Informaticae, volume 24, pages 89-105, 1995.