Homogeneity without Loss of Generality*

Paweł Parys
University of Warsaw
Warsaw, Poland
parys@mimuw.edu.pl

Abstract

We consider higher-order recursion schemes as generators of infinite trees. A sort (simple type) is called homogeneous when all arguments of higher order are taken before any arguments of lower order. We prove that every scheme can be converted into an equivalent one (i.e, generating the same tree) that is homogeneous, that is, uses only homogeneous sorts. Then, we prove the same for safe schemes: every safe scheme can be converted into an equivalent safe homogeneous scheme. Furthermore, we compare two definition of safe schemes: the original definition of Damm, and the modern one. Finally, we prove a lemma which illustrates usefulness of the homogeneity assumption. The results are known, but we prove them in a novel way: by directly manipulating considered schemes.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory → Formalisms → Rewrite systems

Keywords and phrases higher-order recursion schemes, λ-calculus, homogeneous types, safe schemes

Digital Object Identifier 10.4230/LIPIcs.FSCD.2018.28

1 Introduction

Higher-order recursion schemes (*schemes* in short) are used to faithfully represent the control flow of programs in languages with higher-order functions. This formalism is equivalent via direct translations to simply-typed λY-calculus [19] and to higher-order OI grammars [9, 15]. Collapsible pushdown systems [10] and ordered tree-pushdown systems [7] are other equivalent formalisms. Schemes cover some other models such as indexed grammars [1] and ordered multi-pushdown automata [4]. We consider schemes as generators of infinite trees, so we say that two schemes are equivalent if they generate the same tree. Likewise, we say that two classes of schemes are equi-expressive, if for every scheme in one of the classes there exists an equivalent scheme in the other class.

A sort (simple type) is called homogeneous when all arguments of higher order are taken before any arguments of lower order; a scheme is homogeneous when it uses only homogeneous sorts. Homogeneous schemes should not be confused with safe schemes. The safety assumption was first introduced implicitly by Damm [9]. His restriction was that when an argument of some order is applied to a function, then all arguments of greater or the same order have to be applied as well. A modern definition of safety (introduced by Knapik, Niwiński, Urzyczyn [14]) is slightly different: it says that a subterm of some order cannot use parameters of a strictly smaller order. We remark that some authors, while defining safe schemes, require that they are also homogeneous [9, 13, 14], while other authors do not impose this requirement [3, 6]. In this paper we treat homogeneity separately from safety.

* Work supported by the National Science Centre, Poland (grant no. 2016/22/E/ST6/00041).

3rd International Conference on Formal Structures for Computation and Deduction (FSCD 2018).
Editor: Hélène Kirchner; Article No. 28; pp. 28:1–28:15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The goal of this paper is to compare the aforementioned notions, and to give simple translations between equi-expressive classes of schemes. The main equi-expressivity result says that every scheme can be converted into a homogeneous scheme that is equivalent, and remains of the same order. This was shown by Broadbent in his PhD thesis [5, Section 3.4], and was never published. Furthermore, it is easy to see that the Damm's definition of safety is more restrictive than the modern one. On the other hand, it was observed by Carayol and Serre [6] that every scheme that is safe according to the modern definition can be turned into an equivalent scheme that is safe according to Damm's definition. Likewise, it was shown by Blum [2] (his paper dates back to 2009, when it was shared on his personal website, but was published on arXiv only in 2017), and independently by Carayol and Serre [6], that every safe scheme (without the homogeneity assumption) can be converted into an equivalent safe scheme that is homogeneous, and remains of the same order.

All the proofs for safe schemes follow the same idea: they inspect the equivalence between safe schemes and higher-order pushdown automata. It is observed that while translating from safe schemes to higher-order pushdown automata, schemes can comply with a less restrictive definition; simultaneously, when translating from automata to schemes, it is easy to fulfill additional requirements on the scheme.

The proof of Broadbent, dealing with schemes that need not to be safe, is even more complicated. Arbitrary schemes are equivalent to collapsible pushdown automata, a generalization of higher-order pushdown automata. We can see, though, that the only known translation from collapsible pushdown automata to recursion schemes [10] results in schemes that are not homogeneous. The actual proof consists of three steps. First, it is observed that already while translating a scheme to a collapsible pushdown automaton, the resulting automaton is of a special shape. Then, such an automaton is further modified (without changing the generated tree), so that it gains some additional properties. Finally, it is observed that for the particular automata obtained this way, the translation from automata to schemes can be altered so that the resulting schemes are homogeneous.

We reprove the above results: we give a simple transformation changing any scheme to an equivalent homogeneous scheme, and another simple transformation changing any safe scheme to a scheme that is safe according to the more restrictive definition of Damm, and moreover homogeneous.

Both our proofs (the one for general schemes, and the one for safe schemes) do not use any detour through automata; we directly show how to syntactically modify a scheme so that it becomes homogeneous. Roughly, in the case of general schemes we artificially increase orders of some arguments, while in the case of safe schemes we split complex rules into multiple simpler rules, and we reorder arguments. Our direct approach has the advantage that it is more transparent and it sheds some light on the nature of the homogeneity assumption (conversely to the previous proofs: while translating a scheme to an automaton and then back to a scheme, we obtain a scheme of a completely different shape than the original one).

In order to give a full picture we have to recall here the result that there is a scheme that is not equivalent to any safe scheme [16]. We thus have two groups of equi-expressive classes: "unsafe" schemes, either homogeneous or not, and safe schemes, either according to the Damm's definition or to the modern definition, and either homogeneous or not.

In addition to the above results, in the final section we prove a simple lemma, which illustrates usefulness of the homogeneity assumption.

2 Preliminaries

Infinitary λ-calculus. The set of *sorts* (aka. simple types) is defined by induction: o is a sort, and if α and β are sorts, then $\alpha \to \beta$ is a sort. We omit brackets on the right of an arrow, so, for example, $o \to (o \to o)$ is abbreviated to $o \to o \to o$. Notice that every sort can be written in the form $\alpha_1 \to \cdots \to \alpha_s \to o$.

The *order* of a sort γ, denoted $ord(\gamma)$, is defined by induction on the structure of γ: $ord(o) = 0$, and $ord(\alpha \to \beta) = \max(ord(\alpha) + 1, ord(\beta))$. We observe that $ord(\alpha_1 \to \cdots \to \alpha_s \to o) = 1 + \max(ord(\alpha_1), \dots, ord(\alpha_s))$ whenever $s \geq 1$.

A sort $\alpha_1 \to \cdots \to \alpha_s \to o$ is *homogeneous* if $ord(\alpha_1) \geq \cdots \geq ord(\alpha_s)$ and all $\alpha_1, \dots, \alpha_s$ are homogeneous. An equivalent definition says that the sort o is homogeneous, and a sort $\alpha \to \beta$ is homogeneous if $ord(\alpha) = ord(\alpha \to \beta) - 1$ and α, β are homogeneous.

While defining λ-terms, we assume existence of the following sets:

- Σ—a set of symbols (alphabet), and
- $\mathcal{V}$—a set of variables with assigned sorts; we write $x^\alpha, y^\alpha, z^\alpha, \dots$ for variables of sort α.

We consider infinitary, sorted λ-calculus. *Infinitary λ-terms* (or just *λ-terms*) are defined by coinduction (for an introduction to coinductive definitions and proofs see, e.g., Czajka [8]), according to the following rules:

- node constructor—if $K_1^o, \dots, K_r^o$ are λ-terms, then $(a\langle K_1^o, \dots, K_r^o\rangle)^o$ is a λ-term, for every $a \in \Sigma$,
- variable—every variable $x^\alpha \in \mathcal{V}$ is a λ-term,
- application—if $K^{\alpha\to\beta}$ and L^α are λ-terms, then $(K^{\alpha\to\beta}\, L^\alpha)^\beta$ is a λ-term, and
- λ-binder—if K^β is a λ-term and $x^\alpha \in \mathcal{V}$ is a variable, then $(\lambda x^\alpha.K^\beta)^{\alpha\to\beta}$ is a λ-term.

We naturally identify λ-terms differing only in names of bound variables. We often omit sort annotations of λ-terms, but we keep in mind that every λ-term (and every variable) has a particular sort. The set of *free variables* of a λ-term M, denoted $FV(M)$, is defined as usual. A λ-term M is *closed* if $FV(M) = \emptyset$. We assume that in $\mathcal{V}$ there are always some fresh variables of every sort, not appearing in λ-terms under consideration.

The *order* of a λ-term M, written $ord(M)$, is just the order of its sort. The *complexity* of a λ-term M is the smallest number $m \in \mathbb{N} \cup \{\infty\}$ such that all subterms of M are of order at most m.

Reductions. By $M[N/x]$ (where we require that N is of the same sort as x) we denote the λ-term obtained by substituting N for x. This is by definition a capture-avoiding substitution, which means that free variables of N are not captured by λ-binders in M; this is achieved by appropriately renaming bound variables in M.

A *compatible closure* $\rightsquigarrow$ of a relation $\twoheadrightarrow$ is defined by induction according to the following rules:

- if $M \twoheadrightarrow N$, then $M \rightsquigarrow N$,
- if $K_j \rightsquigarrow K_j'$ for some $j \in \{1, \dots, r\}$ and $K_i = K_i'$ for all $i \in \{1, \dots, r\} \setminus \{j\}$, then $a\langle K_1, \dots, K_r\rangle \rightsquigarrow a\langle K_1', \dots, K_r'\rangle$,
- if $K \rightsquigarrow K'$, then $K\, L \rightsquigarrow K'\, L$,
- if $L \rightsquigarrow L'$, then $K\, L \rightsquigarrow K\, L'$, and
- if $K \rightsquigarrow K'$, then $\lambda x.K \rightsquigarrow \lambda x.K'$.

The relation $\to_\beta$ of *β-reduction* is defined as the compatible closure of the relation $\{((\lambda x.K)\, L, K[L/x])\}$. The relation $\to_\eta$ of *η-conversion* is defined as the compatible closure of the relation $\{(\lambda x.K\, x, K) \mid x \not\in FV(K)\}$. We let $(\to_{\beta\eta}) = (\to_\beta) \cup (\to_\eta)$. As a restriction of β-reduction, we define the relation $\xrightarrow{h}_\beta$ of *head β-reduction*: it contains all pairs of the form

$$((\lambda x.K)\,L\,P_1\,\dots\,P_n, K[L/x]\,P_1\,\dots\,P_n)\,.$$

For relations $\rightsquigarrow$ and $\twoheadrightarrow$, by $(\rightsquigarrow)\circ(\twoheadrightarrow)$ we denote their composition, by $\rightsquigarrow^k$ (where $k\in\mathbb{N}$) the composition of $\rightsquigarrow$ with itself k times, and by $\rightsquigarrow^*$ the reflexive transitive closure of $\rightsquigarrow$. Moreover, $\rightsquigarrow^\infty$ is the infinitary closure of $\rightsquigarrow$, defined by coinduction, according to the following rules:

- if $M\rightsquigarrow^* a\langle K_1,\dots,K_r\rangle$ and $K_i\rightsquigarrow^\infty K_i'$ for all $i\in\{1,\dots,r\}$, then $M\rightsquigarrow^\infty a\langle K_1',\dots,K_r'\rangle$,
- if $M\rightsquigarrow^* x$ then $M\rightsquigarrow^\infty x$,
- if $M\rightsquigarrow^* K\,L$, and $K\rightsquigarrow^\infty K'$, and $L\rightsquigarrow^\infty L'$, then $M\rightsquigarrow^\infty K'\,L'$, and
- if $M\rightsquigarrow^* \lambda x.K$ and $K\rightsquigarrow^\infty K'$, then $M\rightsquigarrow^\infty \lambda x.K'$.

Trees; Böhm Trees. A *tree* is defined as a λ-term that is built using only node constructors, that is, not using variables, applications, nor λ-binders.

We consider Böhm trees only for closed λ-terms of sort o. For such a λ-term M, its *Böhm tree* $BT(M)$ is defined by coinduction, as follows:

- if $M\xrightarrow{h}{}^*_\beta a\langle K_1,\dots,K_r\rangle$ for some $a\in\Sigma$ and some λ-terms $K_1,\dots,K_r$, then $BT(M)= a\langle BT(K_1),\dots,BT(K_r)\rangle$;
- otherwise $BT(M)=\bot\langle\rangle$ (where $\bot\in\Sigma$ is a distinguished symbol).

With such a definition it is easy to see that for every M there is exactly one Böhm tree. It is a consequence of Kennaway, Klop, Sleep, de Vries [11] and Kennaway, van Oostrom, de Vries [12] that the Böhm tree does not change during $\beta\eta$-reductions.

▶ **Fact 1.** *If M and N are closed λ-terms of sort o and $M\rightarrow^\infty_{\beta\eta} N$, then $BT(M)= BT(N)$.* ◀

Higher-Order Recursion Schemes. A *higher-order recursion scheme* (or just a *scheme*) is a triple $\mathcal{G}=(\mathcal{N},\mathcal{R},X_0^o)$, where $\mathcal{N}\subseteq\mathcal{V}$ is a finite set of nonterminals, $X_0^o\in\mathcal{N}$ is a starting nonterminal, being of sort o, and $\mathcal{R}$ is a function that maps every nonterminal $X\in\mathcal{N}$ to a finite λ-term of the form $\lambda x_1.\cdots.\lambda x_s.M$, where

- the sorts of X and $\lambda x_1.\cdots.\lambda x_s.M$ are the same,
- $FV(M)\subseteq\mathcal{N}\cup\{x_1,\dots,x_s\}$,
- M is of sort o, and
- M is a finite *applicative term*, that is, it does not contain any λ-binders.

We assume that elements of $\mathcal{N}$ are not used as bound variables, and that $\mathcal{R}(X)$ is not a nonterminal.[1] When $\mathcal{R}(X)=\lambda x_1.\cdots.\lambda x_s.M$, we say that $X\,x_1\,\dots\,x_s\to M$ is a *rule* of $\mathcal{G}$, and M is its *right side*. The *order* of the scheme is defined as the maximum of orders of nonterminals in $\mathcal{N}$.

The infinitary λ-term *generated by* a scheme $\mathcal{G}=(\mathcal{N},\mathcal{R},X_0)$ from a λ-term M, denoted $\Lambda_{\mathcal{G}}(M)$, is defined as the limit of the following process starting from M: take any nonterminal X appearing in the current term, and replace it by $\mathcal{R}(X)$. We define $\Lambda(\mathcal{G})=\Lambda_{\mathcal{G}}(X_0)$; observe that this is a closed λ-term of sort o and of complexity not greater than the order of the scheme. The *tree generated by* $\mathcal{G}$ is defined as $BT(\Lambda(\mathcal{G}))$.

We say that a scheme $\mathcal{G}=(\mathcal{N},\mathcal{R},X_0)$ is *homogeneous* if sorts of all nonterminals in $\mathcal{N}$ are homogeneous. Notice that then also the sort of every subterm of $\mathcal{R}(X)$ is homogeneous, for every nonterminal $X\in\mathcal{N}$.

---

[1] Without the last condition, it would be necessary to give a more complicated definition of $\Lambda(\mathcal{G})$. On the other hand, it is easy to ensure this condition, without changing the tree generated by the scheme.

▶ **Example 1.** Consider a scheme $\mathcal{G}_1$ with nonterminals Y_0^o, $Y_1^{o\to((o\to o)\to o)\to o}$, $Y_2^{o\to o}$, and $Y_3^{o\to(o\to o)\to o}$, where Y_0 is starting; and rules

$$Y_0 \to Y_1\,(b\langle c\langle\rangle\rangle)\,(Y_3\,(c\langle\rangle))\,, \qquad Y_2\,x^o \to x\,,$$
$$Y_1\,x^o\,z^{(o\to o)\to o} \to a\langle z\,Y_2, Y_1\,(b\langle x\rangle)\,(Y_3\,x)\rangle\,, \qquad Y_3\,x^o\,y^{o\to o} \to y\,x\,.$$

Then

$$\Lambda(\mathcal{G}_1) = M_1\,(b\langle c\langle\rangle\rangle)\,((\lambda x^o.\lambda y^{o\to o}.y\,x)\,(c\langle\rangle))\,,$$

where M_1 is the unique λ-term such that

$$M_1 = \lambda x^o.\lambda z^{(o\to o)\to o}.a\langle z\,(\lambda x^o.x), M_1\,(b\langle x\rangle)\,((\lambda x^o.\lambda y^{o\to o}.y\,x)\,x)\rangle\,.$$

We can see that $BT(\Lambda(\mathcal{G}_1)) = a\langle T_0, a\langle T_1, a\langle T_2, \dots\rangle\rangle\rangle$, where $T_0 = c\langle\rangle$, and $T_{i+1} = b\langle T_i\rangle$ for $i \in \mathbb{N}$.

Notice that the sorts of Y_1 and of Y_3 are not homogeneous: the first parameter is of order 0, and the second of order 2 or 1. ◀

3 Ensuring Homogeneity

We now prove our main theorem:

▶ **Theorem 2.** *For every scheme $\mathcal{G} = (\mathcal{N}, \mathcal{R}, X_0)$ one can construct in logarithmic space a homogeneous scheme $\mathcal{H}$ that is of the same order as $\mathcal{G}$ and such that $BT(\Lambda(\mathcal{H})) = BT(\Lambda(\mathcal{G}))$.*

Let us first present the general idea of the proof. Consider thus a nonterminal X with $\mathcal{R}(X) = \lambda x.\lambda y.K$, where $ord(x) < ord(y)$ (like Y_1 or Y_3 in Example 1). The sort of X is not homogeneous, as it does not satisfy $ord(x) \geq ord(y)$. How can we make it homogeneous?

One idea, which does not work, is to swap the order of x and y. The sort of $\lambda y.\lambda x.K$ is indeed homogeneous. Such a swap is problematic, though: possibly there are places where only one argument is given to X, corresponding to the parameter x (e.g., in Example 1 we always give only one argument to Y_3). When the parameters are swapped, we cannot pass a value of x to X, without passing a value of y.

There is another simple idea, which actually works. Namely, we should raise the order of x to $ord(y)$. How can we do that? Simply instead of passing to X an argument M of a low order $ord(x)$, we pass a function $\lambda z.M$ (of order $ord(y)$, higher than $ord(x)$), which ignores its argument z and returns M. On the other side, we change every use of x in K to $x\,N$, where N is an arbitrary λ-term of the same sort as z.

Notice that after such a modification of the sort of x, the order of $\lambda x.\lambda y.K$ remains as before the modification. This is very important: thanks to this property (orders of subterms do not change), we can perform the modification independently in every place. Moreover, as a side effect, also the order of the whole scheme remains unchanged.

There is one more difficulty to overcome, while proving the theorem. Namely, in λ-calculus it would be possible to simply write $\lambda z.M$ instead of M, whenever we wanted to convert M into a function returning M. This is not so trivial for schemes, as we cannot use λ-binders—we should use nonterminals instead. Say that we want to change the order of M from 0 (sort o) to 1 (sort $o \to o$). To this end, we introduce a nonterminal S with $\mathcal{R}(S) = \lambda x^o.\lambda z^o.x$, and we write $S\,M$ instead of $\lambda z.M$ (that is, instead of M in the original scheme).

Notice, though, that the sort of the new nonterminal S has to be homogeneous as well. This means that using such a nonterminal S we can raise the order only by one, as we cannot have $\mathcal{R}(S) = \lambda x^o.\lambda z.x$ with $ord(z) > 0$. If we want to raise the order of M from 0 to 2 (or more), beside of S we need another nonterminal S' which raises the order from 1 to 2 (again only by one), etc. As we start now from sort $o \to o$, we should take $\mathcal{R}(S') = \lambda x_1^{o\to o}.\lambda z_1^{o\to o}.\lambda z^o.x_1\, z$. We then write $S'\,(S\,M)$ instead of M, which, after expanding S and S', equals

$$(\lambda x_1^{o\to o}.\lambda z_1^{o\to o}.\lambda z^o.x_1\, z)\,((\lambda x^o.\lambda z^o.x)\,M)\,.$$

This β-reduces (in three steps) to $\lambda z_1^{o\to o}.\lambda z^o.M$, thus it is a function of order 2 ignoring its arguments and returning M.

We now come to details. First, we define sorts γ_k; these will be sorts of the spare parameters (i.e., of z in the above explanation). The definition is by induction:

$$\gamma_0 = o, \qquad\qquad \gamma_k = \gamma_{k-1} \to o \qquad \text{for } k \geq 1.$$

For example, $\gamma_1 = o \to o$ and $\gamma_2 = (o \to o) \to o$. We see that $ord(\gamma_k) = k$ for all $k \in \mathbb{N}$.

Next, we have an operation $\mathbf{R}_k$, which says how to raise the order of a sort α to k. For every sort α and every $k \geq ord(\alpha)$ we define:

$$\mathbf{R}_k(\alpha) = \gamma_{k-1} \to \gamma_{k-2} \to \cdots \to \gamma_{ord(\alpha)} \to \alpha\,.$$

In particular $\mathbf{R}_{ord(\alpha)}(\alpha) = \alpha$. We see that $ord(\mathbf{R}_k(\alpha)) = k$. Basing on $\mathbf{R}_k$, we define, by induction, a transformation $\mathbf{H}$ changing an arbitrary sort into a homogeneous one:

$$\mathbf{H}(o) = o, \qquad\qquad \mathbf{H}(\alpha \to \beta) = \mathbf{R}_{ord(\alpha\to\beta)-1}(\mathbf{H}(\alpha)) \to \mathbf{H}(\beta).$$

Notice that $ord(\mathbf{H}(\alpha)) = ord(\alpha)$ for every sort α, and that $\mathbf{H}(\alpha)$ is indeed homogeneous.

Next, we come to transforming λ-terms. For every sort α appearing in the original scheme $\mathcal{G}$ (as a sort of a subterm of $\mathcal{R}(X)$ for some nonterminal $X \in \mathcal{N}$), and for every k such that $ord(\alpha) < k \leq ord(\mathcal{G})$, we add a nonterminal $S_{\alpha,k}$. Its sort is $\mathbf{R}_{k-1}(\mathbf{H}(\alpha)) \to \mathbf{R}_k(\mathbf{H}(\alpha))$. Recall that $\mathbf{R}_k(\mathbf{H}(\alpha)) = \gamma_{k-1} \to \mathbf{R}_{k-1}(\mathbf{H}(\alpha))$; let us also write $\mathbf{R}_{k-1}(\mathbf{H}(\alpha)) = \beta_1 \to \cdots \to \beta_s \to o$. Then the rule for $S_{\alpha,k}$ is

$$\mathcal{R}'(S_{\alpha,k}) = \lambda x.\lambda z.\lambda y_1.\cdots.\lambda y_s.x\, y_1\, \ldots\, y_s\,.$$

Here the sort of x is $\mathbf{R}_{k-1}(\mathbf{H}(\alpha))$, the sort of z is γ_{k-1}, and the sorts of $y_1, \ldots, y_s$ are $\beta_1, \ldots, \beta_s$, respectively.

Let again α be a sort appearing in $\mathcal{G}$, and let k be such that $ord(\alpha) \leq k \leq ord(\mathcal{G})$. The sort of a λ-term may be changed from $\mathbf{H}(\alpha)$ to $\mathbf{R}_k(\mathbf{H}(\alpha))$ by applying the following transformation, also called $\mathbf{R}_k$:

$$\mathbf{R}_k(M) = S_{\alpha,k}\,(S_{\alpha,k-1}\,\ldots\,(S_{\alpha,ord(\alpha)+1}\,M)\ldots)\,.$$

Here, by appending a nonterminal $S_{\alpha,i}$ we change the sort from $\mathbf{R}_{i-1}(\mathbf{H}(\alpha))$ to $\mathbf{R}_i(\mathbf{H}(\alpha))$; recall that $\mathbf{R}_{ord(\alpha)}(\mathbf{H}(\alpha)) = \mathbf{H}(\alpha)$.

We also need an opposite operation, which converts a function back to its value, by applying some arguments of sorts γ_k. First we define some nonterminals of such sorts: we fix a symbol $e \in \Sigma$, and for every $k < ord(\mathcal{G})$ we add a nonterminal U_k of sort γ_k, and we take:

$$\mathcal{R}'(U_0) = e\langle\rangle, \qquad\qquad \mathcal{R}'(U_k) = \lambda z^{\gamma_{k-1}}.e\langle\rangle \qquad \text{for } k \geq 1.$$

Clearly U_k has sort γ_k, for every $k \in \mathbb{N}$.

When N is of sort $\mathbf{R}_k(\mathbf{H}(\alpha))$, and $\mathit{ord}(\alpha) = n$ (the relation between k and n is $k \geq n$), we define

$$\mathbf{L}_n(N) = N\, U_{k-1}\, U_{k-2}\, \ldots\, U_n\,.$$

This λ-term is indeed of sort $\mathbf{H}(\alpha)$.

Using the above operations, we define a transformation changing the original scheme into a homogeneous one. Let us first describe this transformation informally. It works as follows. We first change the sort of every λ-term (i.e., every nonterminal, every variable, and every subterm of the right side of every rule) from α to $\mathbf{H}(\alpha)$. This causes a problem on applications, since to a function of sort $\mathbf{H}(\alpha \to \beta) = \mathbf{R}_{\mathit{ord}(\alpha\to\beta)-1}(\mathbf{H}(\alpha)) \to \mathbf{H}(\beta)$ we apply an argument of sort $\mathbf{H}(\alpha)$. We thus repair the argument by applying $\mathbf{R}_{\mathit{ord}(\alpha\to\beta)-1}(\cdot)$ to it. This also causes a problem on λ-binders and on variables: the new sort of a λ-binder $\lambda x^\alpha.K^\beta$ should be $\mathbf{H}(\alpha \to \beta) = \mathbf{R}_{\mathit{ord}(\alpha\to\beta)-1}(\mathbf{H}(\alpha)) \to \mathbf{H}(\beta)$, so the sort of the variable should be $\mathbf{R}_{\mathit{ord}(\alpha\to\beta)-1}(\mathbf{H}(\alpha))$; however, while using this variable, we expect that it will have sort $\mathbf{H}(\alpha)$. We thus apply $\mathbf{L}_{\mathit{ord}(\alpha)}(\cdot)$ to every place where the variable is used. There is no problem with nonterminals: every nonterminal simply changes its sort from α to $\mathbf{H}(\alpha)$.

We now define the transformation formally. A *raise environment* is a function Ω mapping some variable names to sorts, where we require that $\Omega(x^\alpha)$ equals $\mathbf{R}_k(\mathbf{H}(\alpha))$ for some $k \geq \mathit{ord}(\alpha)$. Intuitively, $\Omega(x^\alpha)$ is a new sort that the variable gets after the transformation. For a raise environment Ω (such that $FV(M) \subseteq \mathrm{dom}(\Omega)$) we define $\mathbf{H}_\Omega(M)$ by coinduction on the structure of a λ-term M:

$$\begin{aligned}
&\mathbf{H}_\Omega(a\langle K_1,\ldots,K_r\rangle) = a\langle \mathbf{H}_\Omega(K_1),\ldots,\mathbf{H}_\Omega(K_r)\rangle\,. \\
&\mathbf{H}_\Omega(x^\alpha) = \mathbf{L}_{\mathit{ord}(\alpha)}(x^{\Omega(x^\alpha)}) && \text{if } x^\alpha \in \mathcal{V}\setminus\mathcal{N}, \\
&\mathbf{H}_\Omega(X^\alpha) = X^{\mathbf{H}(\alpha)} && \text{if } X^\alpha \in \mathcal{N}, \\
&\mathbf{H}_\Omega(K\,L) = \mathbf{H}_\Omega(K)\,\mathbf{R}_{\mathit{ord}(K)-1}(\mathbf{H}_\Omega(L))\,, \\
&\mathbf{H}_\Omega(\lambda x^\alpha.K) = \lambda x^{\alpha'}.\mathbf{H}_{\Omega[x^\alpha\mapsto\alpha']}(K)\,, && \text{where } \alpha' = \mathbf{R}_{\mathit{ord}(\lambda x^\alpha.K)-1}(\mathbf{H}(\alpha))\,.
\end{aligned}$$

Here by $\Omega[x^\alpha \mapsto \alpha']$ we mean the function that maps x^α to α', and every other variable $y \in \mathrm{dom}(\Omega)$ to $\Omega(y)$. Notice that for M of sort α, the resulting λ-term $\mathbf{H}_\Omega(M)$ is of sort $\mathbf{H}(\alpha)$; in particular, in the case of an application with K of sort $\beta \to \gamma$, the sort of the function $\mathbf{H}_\Omega(K)$ being $\mathbf{H}(\beta\to\gamma) = \mathbf{R}_{\mathit{ord}(\beta\to\gamma)-1}(\mathbf{H}(\beta)) \to \mathbf{H}(\gamma)$ matches well with the sort of the argument, being $\mathbf{R}_{\mathit{ord}(\beta\to\gamma)-1}(\mathbf{H}(\beta))$.

The newly created scheme $\mathcal{H} = (\mathcal{N}', \mathcal{R}', X_0)$ is as follows. For every nonterminal $X^\alpha \in \mathcal{N}$, to $\mathcal{N}'$ we take a nonterminal $X^{\mathbf{H}(\alpha)}$, and we define $\mathcal{R}'(X^{\mathbf{H}(\alpha)}) = \mathbf{H}_\emptyset(\mathcal{R}(X^\alpha))$ (where $\emptyset$ is the raise environment with empty domain). Additionally in $\mathcal{N}'$ we have nonterminals $S_{\alpha,k}$ and U_k, with appropriate rules, as defined above.

▶ **Example 1** (continued). While applying our transformation to the scheme $\mathcal{G}_1$ from Example 1, we obtain a homogeneous scheme with the following rules (where we write S_i instead of $S_{o,i}$):

$$\begin{aligned}
&Y_0 \to Y_1\,(S_2\,(S_1\,(b\langle c\langle\rangle\rangle)))\,(Y_3\,(S_1\,(c\langle\rangle)))\,, \\
&Y_1\,x^{(o\to o)\to o\to o}\,z^{(o\to o)\to o} \to a\langle z\,Y_2, Y_1\,(S_2\,(S_1\,(b\langle x\,U_1\,U_0\rangle)))\,(Y_3\,(S_1\,(x\,U_1\,U_0)))\rangle\,, \\
&Y_2\,x^o \to x\,, \qquad S_1\,x^o\,z^o \to x\,, \qquad U_0 \to e\langle\rangle\,, \\
&Y_3\,x^{o\to o}\,y^{o\to o} \to y\,(x\,U_0)\,, \qquad S_2\,x^{o\to o}\,z^{o\to o}\,y_1^o \to x\,y_1\,, \qquad U_1\,z^o \to e\langle\rangle\,.
\end{aligned}$$ ◀

It is easy to see that $\mathcal{H}$ can be computed in logarithmic space (in particular its size is polynomial in the size of $\mathcal{G}$). We also notice that the order of the scheme remains unchanged; this is the case because $ord(\mathbf{H}(\alpha)) = ord(\alpha)$ for every sort α.

It remains to prove that $BT(\Lambda(\mathcal{H})) = BT(\Lambda(\mathcal{G}))$ for every closed λ-term M^o. To this end, we need to define a variant of our transformation that works with λ-terms, not with schemes. We thus define $\mathbf{R}^\Lambda_k(M)$ is the same way as $\mathbf{R}_k(M)$, but in the definition we replace $S_{\alpha,i}$ with $\mathcal{R}'(S_{\alpha,i})$ (recall that $\mathcal{R}'$ describes rules of the new scheme). Similarly $\mathbf{L}^\Lambda_n(N)$ is defined as $\mathbf{L}_n(N)$, but in the definition we replace U_i with $\mathcal{R}'(U_i)$. Finally, $\mathbf{H}^\Lambda_\Omega(M)$ is defined as $\mathbf{H}_\Omega(M)$, but it uses functions $\mathbf{R}^\Lambda_i$ and $\mathbf{L}^\Lambda_i$ instead of $\mathbf{R}_i$ and $\mathbf{L}_i$. In other words, this variant of the transformation inserts definitions of the nonterminals $S_{\alpha,i}$ and U_i instead of the nonterminals themselves.

We immediately see that $\Lambda(\mathcal{H}) = \mathbf{H}^\Lambda_\emptyset(\Lambda(\mathcal{G}))$. In the remaining part of the section we will prove that $BT(\mathbf{H}^\Lambda_\emptyset(M)) = BT(M)$ for every closed λ-term M^o; this implies that $BT(\Lambda(\mathcal{H})) = BT(\Lambda(\mathcal{G}))$ when instantiated with $M = \Lambda(\mathcal{G})$. The proof is split to several lemmata.

▶ **Lemma 3.** *Let P be a λ-term of sort $\mathbf{R}_{k-1}(\mathbf{H}(\alpha))$, where $k > ord(\alpha)$. In such a situation $\mathcal{R}'(S_{\alpha,k})\, P\, \mathcal{R}'(U_{k-1}) \to^*_{\beta\eta} P$.*

Proof. Let $\mathbf{R}_{k-1}(\mathbf{H}(\alpha)) = \beta_1 \to \cdots \to \beta_s \to o$. Recalling the definition of $\mathcal{R}'(S_{\alpha,k})$ we observe that

$$\begin{aligned}\mathcal{R}'(S_{\alpha,k})\, P\, \mathcal{R}'(U_{k-1}) &= (\lambda x.\lambda z.\lambda y_1.\cdots.\lambda y_s.x\, y_1\, \dots\, y_s)\, P\, \mathcal{R}'(U_{k-1})\\ &\to^2_\beta \lambda y_1.\cdots.\lambda y_s.P\, y_1\, \dots\, y_s \to^s_\eta P\,.\end{aligned}$$ ◀

▶ **Lemma 4.** *Let M be a λ-term of sort $\mathbf{H}(\alpha)$, and let $k \geq ord(\alpha) = n$. In such a situation $\mathbf{L}^\Lambda_n(\mathbf{R}^\Lambda_k(M))) \to^*_{\beta\eta} M$.*

Proof. The thesis follows directly from Lemma 3 once we recall that

$$\begin{aligned}\mathbf{L}^\Lambda_n(\mathbf{R}^\Lambda_k(M)) = \mathcal{R}'(S_{\alpha,k})\, (\mathcal{R}'(S_{\alpha,k-1})\, \dots (\mathcal{R}'(S_{\alpha,n+1})\, M) \dots)&\\ \mathcal{R}'(U_{k-1})\, \mathcal{R}'(U_{k-2})\, \dots\, \mathcal{R}'(U_n)\,.&\end{aligned}$$ ◀

▶ **Lemma 5.** *Let M and N^α be λ-terms, x^α a variable, and Ω a raise environment such that $FV(M) \setminus \{x^\alpha\} \cup FV(N) \subseteq \mathrm{dom}(\Omega)$. Let also $\alpha' = \mathbf{R}_k(\mathbf{H}(\alpha))$ for some $k \geq ord(\alpha)$. In such a situation $\mathbf{H}^\Lambda_{\Omega[x^\alpha \mapsto \alpha']}(M)[\mathbf{R}^\Lambda_k(\mathbf{H}^\Lambda_\Omega(N))/x^{\alpha'}] \to^\infty_{\beta\eta} \mathbf{H}^\Lambda_\Omega(M[N/x^\alpha])$.*

Proof. The proof is by coinduction on the structure of M. Only the case of $M = x^\alpha$ is interesting. In this case $\mathbf{H}^\Lambda_{\Omega[x^\alpha \mapsto \alpha']}(M) = \mathbf{L}^\Lambda_{ord(\alpha)}(x^{\alpha'})$, so $\mathbf{H}^\Lambda_{\Omega[x^\alpha \mapsto \alpha']}(M)[\mathbf{R}^\Lambda_k(\mathbf{H}^\Lambda_\Omega(N))/x^{\alpha'}] = \mathbf{L}^\Lambda_{ord(\alpha)}(\mathbf{R}^\Lambda_k(\mathbf{H}^\Lambda_\Omega(N)))$, and by Lemma 4 we have that $\mathbf{L}^\Lambda_{ord(\alpha)}(\mathbf{R}^\Lambda_k(\mathbf{H}^\Lambda_\Omega(N))) \to^*_{\beta\eta} \mathbf{H}^\Lambda_\Omega(N)$, which is what we need since $M[N/x^\alpha] = N$.

We remark that in the case of $M = \lambda y^\beta.K$, we use the assumption of coinduction for the extended raise environment $\Omega[y^\beta \mapsto \beta']$, and we observe that $\mathbf{H}^\Lambda_\Omega(N) = \mathbf{H}^\Lambda_{\Omega[y^\beta \mapsto \beta']}(N)$ when (without loss of generality) we assume that y^β is not free in N. ◀

▶ **Lemma 6.** *If $M \xrightarrow{h}_\beta N$, and Ω is a raise environment such that $FV(M) \subseteq \mathrm{dom}(\Omega)$, then $(\mathbf{H}^\Lambda_\Omega(M), \mathbf{H}^\Lambda_\Omega(N)) \in (\xrightarrow{h}_\beta) \circ (\to^\infty_{\beta\eta})$.*

Proof. The proof is by induction on the depth of the head redex in M. The induction step is trivial. Consider thus the base case, when $M = (\lambda x^\alpha.K)\, L$, and $N = K[L/x^\alpha]$. Let $k = ord(\lambda x.K) - 1$, and $\alpha' = \mathbf{R}_k(\mathbf{H}(\alpha))$; clearly $k \geq ord(\alpha)$. By definition we have that

$$\mathbf{H}^\Lambda_\Omega(M) = (\lambda x^{\alpha'}.\mathbf{H}^\Lambda_{\Omega[x^\alpha \mapsto \alpha']}(K))\, \mathbf{R}^\Lambda_k(\mathbf{H}^\Lambda_\Omega(L))\,.$$

Taking $P = \mathbf{H}^{\Lambda}_{\Omega[x^\alpha \mapsto \alpha']}(K)[\mathbf{R}^{\Lambda}_k(\mathbf{H}^{\Lambda}_{\Omega}(L))/x^{\alpha'}]$ we see that $\mathbf{H}^{\Lambda}_{\Omega}(M) \xrightarrow{h}_\beta P$, and from Lemma 5 we obtain that $P \to^{\infty}_{\beta\eta} \mathbf{H}^{\Lambda}_{\Omega}(N)$. ◀

Using Lemma 6 it is easy to prove by coinduction that for every closed λ-term M of sort o it holds that $BT(\mathbf{H}^{\Lambda}_{\emptyset}(M)) = BT(M)$. Let us write this in details. The proof is by coinduction on the structure of these Böhm trees. According to the definition of a Böhm tree, we have two cases. The first of them is that $M \xrightarrow{h}^*_\beta N$ for some N that starts with a node constructor. In this case, by Lemma 6 (applied to every reduction in the sequence of reductions witnessing $M \xrightarrow{h}^*_\beta N$) we have that $(\mathbf{H}^{\Lambda}_{\emptyset}(M), \mathbf{H}^{\Lambda}_{\emptyset}(N)) \in ((\xrightarrow{h}_\beta) \circ (\to^{\infty}_{\beta\eta}))^*$. Clearly $(\xrightarrow{h}_\beta) \subseteq (\to^{\infty}_{\beta\eta})$, thus using Fact 1 (multiple times) we obtain that $BT(\mathbf{H}^{\Lambda}_{\emptyset}(M)) = BT(\mathbf{H}^{\Lambda}_{\emptyset}(N))$. Let us write $N = a\langle K_1, \dots, K_r\rangle$; then $\mathbf{H}^{\Lambda}_{\emptyset}(N) = a\langle \mathbf{H}^{\Lambda}_{\emptyset}(K_1), \dots, \mathbf{H}^{\Lambda}_{\emptyset}(K_r)\rangle$. Since $BT(\mathbf{H}^{\Lambda}_{\emptyset}(K_i)) = BT(K_i)$ by the assumption of coinduction, we can conclude that

$$\begin{aligned} BT(\mathbf{H}^{\Lambda}_{\emptyset}(M)) &= BT(\mathbf{H}^{\Lambda}_{\emptyset}(N)) = a\langle BT(\mathbf{H}^{\Lambda}_{\emptyset}(K_1)), \dots, BT(\mathbf{H}^{\Lambda}_{\emptyset}(K_r))\rangle \\ &= a\langle BT(K_1), \dots, BT(K_r)\rangle = BT(M)\,. \end{aligned}$$

The opposite case is that no sequence of head β-reductions from M leads to a λ-term starting with a node constructor. It is then possible that $M \xrightarrow{h}^*_\beta N$ for some N such that no head β-reduction can be performed from N (but N does not start with a node constructor). Since M, and thus also N, are closed and of sort o, this implies that N is of the form $\dots K_3\, K_2\, K_1$ (infinite application). From the definition of $\mathbf{H}^{\Lambda}_{\emptyset}$ it follows that $\mathbf{H}^{\Lambda}_{\emptyset}(N)$ is also such an infinite application, and thus no head β-reduction can be performed from N. Moreover, as in the previous case, we can see that $BT(\mathbf{H}^{\Lambda}_{\emptyset}(M)) = BT(\mathbf{H}^{\Lambda}_{\emptyset}(N))$. We thus have

$$BT(\mathbf{H}^{\Lambda}_{\emptyset}(M)) = BT(\mathbf{H}^{\Lambda}_{\emptyset}(N)) = \bot\langle\rangle = BT(M)\,.$$

Another possibility is that an infinite sequence of head β-reductions can be performed from M. In other words, for every $n \in \mathbb{N}$ there is a λ-term N such that $M \xrightarrow{h}^n_\beta N$. Fix some such n and N. Lemma 6 implies that $(\mathbf{H}^{\Lambda}_{\emptyset}(M), \mathbf{H}^{\Lambda}_{\emptyset}(N)) \in ((\xrightarrow{h}_\beta) \circ (\to^{\infty}_{\beta\eta}))^n$. Using Fact 7 (below) we can move all head β-reductions to the front, and obtain that $(\mathbf{H}^{\Lambda}_{\emptyset}(M), \mathbf{H}^{\Lambda}_{\emptyset}(N)) \in (\xrightarrow{h}_\beta)^n \circ (\to^{\infty}_{\beta\eta})^n$ (we suppress the proof of Fact 7, as the fact is standard, and the proof is not difficult). This can be done for every n, which means that arbitrarily long sequences of head β-reductions start in $\mathbf{H}^{\Lambda}_{\emptyset}(M)$. Recalling that for every P there is at most one Q such that $P \xrightarrow{h}_\beta Q$, and that no head β-reduction can be performed from a λ-term starting with a node constructor, we conclude that $BT(\mathbf{H}^{\Lambda}_{\emptyset}(M)) = \bot\langle\rangle = BT(M)$.

▶ **Fact 7.** *For all λ-terms M, N of sort o, if $(M,N) \in (\to^{\infty}_{\beta\eta}) \circ (\xrightarrow{h}_\beta)$, then $(M,N) \in (\xrightarrow{h}_\beta) \circ (\to^{\infty}_{\beta\eta})$.* ◀

4 Safe Schemes

In this section we consider safe schemes. Let us recall that we have two definitions of safety. Following Carayol and Serre [6] we use the name "safe schemes" for schemes that are safe according to the modern definition, and "Damm-safe schemes" for schemes that are safe according to the definition of Damm. We now give these definitions.

We define by coinduction when an applicative term is *safe*, with respect to a set of nonterminals $\mathcal{N}$:

- $M = a\langle K_1, \dots, K_r\rangle$ is safe if $K_1, \dots, K_r$ are safe,
- $M = x \in \mathcal{V}$ (in particular $M = X \in \mathcal{N}$) is always safe, and
- $M = K\, L_1\, \dots\, L_s$ (with $s \geq 1$) is safe if $K, L_1, \dots, L_s$ are safe, and additionally $ord(x) \geq ord(M)$ for all $x \in FV(M) \setminus \mathcal{N}$.

Damm-safe applicative terms are also defined by coinduction:

- $M = a\langle K_1, \dots, K_r\rangle$ is Damm-safe if $K_1, \dots, K_r$ are Damm-safe,
- $M = x \in \mathcal{V}$ (in particular $M = X \in \mathcal{N}$) is always Damm-safe, and
- $M = K\, L_1\, \dots\, L_s$ (with $s \geq 1$) is Damm-safe if $K, L_1, \dots, L_s$ are Damm-safe, and additionally $ord(L_i) \geq ord(M)$ for all $i \in \{1, \dots, s\}$.

A scheme $\mathcal{G} = (\mathcal{N}, \mathcal{R}, X_0)$ is safe (Damm-safe) if the right side of every of its rules (i.e., the term M when $\mathcal{R}(X) = \lambda x_1.\cdots.\lambda x_s.M$) is safe (Damm-safe, respectively).

Notice that not every subterm of a (Damm-)safe term need to be (Damm-)safe. But, for example, subterms appearing as arguments are (Damm-)safe, etc. We remark that the definition of safe applicative terms can be extended to λ-terms which are not applicative [3], but we refrain from doing this.

▶ **Example 2.** Consider a scheme $\mathcal{G}_2$ with the following rules:

$$S \to W\,(X\,(b\langle\rangle))\,, \qquad W\,g^{(o\to o)\to o} \to Y\,(X\,(Y\,g))\,, \qquad Y\,g^{(o\to o)\to o} \to g\,A\,,$$
$$X\,x^o\,f^{o\to o} \to f\,x\,, \qquad A\,x^o \to a\langle x\rangle\,.$$

This scheme is safe, but not Damm-safe; in particular the subterm $X\,(Y\,g)$ is not Damm-safe since $ord(Y\,g) = 0 < 2 = ord(X\,(Y\,g))$. Moreover, the sort of X is not homogeneous. Notice that $BT(\Lambda(\mathcal{G}_2)) = a\langle a\langle b\langle\rangle\rangle\rangle$. ◀

It is easy to prove by coinduction that every Damm-safe applicative term is also safe; in consequence every Damm-safe scheme is also safe. We now give two transformations: first we show how to convert a safe scheme into an equivalent scheme that is Damm-safe; then we show how to convert a Damm-safe scheme into an equivalent scheme that is Damm-safe and homogeneous.

▶ **Theorem 8.** *For every safe scheme $\mathcal{G} = (\mathcal{N}, \mathcal{R}, X_0)$ one can construct in logarithmic space a Damm-safe scheme $\mathcal{H} = (\mathcal{N}', \mathcal{R}', Y_0)$ that is of the same order as $\mathcal{G}$ and such that $BT(\Lambda(\mathcal{H})) = BT(\Lambda(\mathcal{G}))$.*

Let us fix some (arbitrary) order $\prec$ on variables. When $FV(M) \setminus \{\mathcal{N}\} = \{x_1, \dots, x_k\}$, where $x_1 \prec \cdots \prec x_k$, then we write $OV(M)$ for the sequence $(x_1, \dots, x_k)$.

The transformation of Theorem 8 amounts to splitting every rule of $\mathcal{G}$ into multiple simpler rules. More precisely, for every safe subterm M of the right side of every rule of $\mathcal{G}$, and for every nonterminal $M = X \in \mathcal{N}$, we create a new nonterminal denoted $\lfloor M\rfloor$. If $OV(M) = (x_1^{\alpha_1}, \dots, x_k^{\alpha_k})$, and if the sort of M is β, then the sort of $\lfloor M\rfloor$ is $\alpha_1 \to \cdots \to \alpha_k \to \beta$. To the new set of nonterminals $\mathcal{N}'$, we take all such nonterminals $\lfloor M\rfloor$. As the starting nonterminal we take $Y_0 = \lfloor X_0\rfloor$.

We now define $\mathcal{R}'(\lfloor M\rfloor)$ for every nonterminal $\lfloor M\rfloor \in \mathcal{N}'$. Consider first the case when $M = X$ is a nonterminal from $\mathcal{N}$. Suppose that $\mathcal{R}(X) = \lambda x_1.\cdots.\lambda x_s.K$, and $OV(K) = (y_1, \dots, y_k)$. In such a situation we put $\mathcal{R}'(\lfloor M\rfloor) = \lambda x_1.\cdots.\lambda x_s.\lfloor K\rfloor\, y_1\, \dots\, y_r$ (on the list $y_1, \dots, y_r$ we have those of $x_1, \dots, x_s$ which are free in K, reordered according to $\prec$).

Suppose now that M is not a nonterminal from $\mathcal{N}$. Let $OV(M) = (x_1, \dots, x_k)$. Let also $y_1, \dots, y_s$ be variables of sorts $\alpha_1, \dots, \alpha_s$, where $\alpha_1 \to \cdots \to \alpha_s \to o$ is the sort of M. We have three possibilities, depending on the shape of M.

- If $M = a\langle K_1, \dots, K_r\rangle$, and $OV(K_i) = (z_{i,1}, \dots, z_{i,m_i})$ for all $i \in \{1, \dots, r\}$, then

$$\mathcal{R}'(\lfloor M\rfloor) = \lambda x_1.\cdots.\lambda x_k.a\langle \lfloor K_1\rfloor\, z_{1,1}\, \dots\, z_{1,m_1}, \dots, \lfloor K_r\rfloor\, z_{r,1}\, \dots\, z_{r,m_r}\rangle\,.$$

- If $M = x$, then $\mathcal{R}'(\lfloor M\rfloor) = \lambda x.\lambda y_1.\cdots.\lambda y_s.x\, y_1\, \dots\, y_s$.
- If $M = K_0\, K_1\, \dots\, K_r$, where $r \geq 1$, and K_0 is not an application, and $OV(K_i) = (z_{i,1}\, \dots\, z_{i,m_i})$ for all $i \in \{0, \dots, r\}$, then

$$\mathcal{R}'(\lfloor M\rfloor) = \lambda x_1.\cdots.\lambda x_k.\lambda y_1.\cdots.\lambda y_s.\lfloor K_0\rfloor\, z_{0,1}\, \dots\, z_{0,m_0}$$
$$(\lfloor K_1\rfloor\, z_{1,1}\, \dots\, z_{1,m_1}) \dots (\lfloor K_r\rfloor\, z_{r,1}\, \dots\, z_{r,m_r})\, y_1\, \dots\, y_s\,.$$

Notice that in the first and the third case, the subterms K_i are safe, so $\lfloor K_i\rfloor$ is indeed a nonterminal in $\mathcal{N}'$. It is also easy to prove that the right side of every rule is Damm-safe. Indeed, for subterms of sort o (i.e., of order 0) there is nothing to check. The only subterms which are of higher order (and which are not a part of a larger application) are $\lfloor K_i\rfloor\, z_{i,1}\, \dots\, z_{i,m_i}$ in the last case of the definition. By safety of K_i we have that $ord(z_{i,j}) \geq ord(K_i)$, since $z_{i,j}$ is free in K_i, and exactly this is needed to claim that $\lfloor K_i\rfloor\, z_{i,1}\, \dots\, z_{i,m_i}$ is Damm-safe.

Let $Exp(K)$ be the λ-term obtained by repeatedly replacing in K all nonterminals $\lfloor L\rfloor$ such that $L \notin \mathcal{N}$ by $\mathcal{R}'(L)$ (this is similar to $\Lambda_{\mathcal{H}}(K)$, but we do not expand nonterminals of the form $\lfloor X\rfloor$, where $X \in \mathcal{N}$). It is easy to prove by induction on the structure of a finite applicative term M, that if $OV(M) = (x_1, \dots, x_k)$, then $Exp(\mathcal{R}'(\lfloor M\rfloor))\, x_1\, \dots\, x_k \to^*_{\beta\eta} M$ (if we identify nonterminals $X \in \mathcal{N}$ with $\lfloor X\rfloor$). In consequence $\Lambda(\mathcal{H}) \to^\infty_{\beta\eta} \Lambda(\mathcal{G})$, which implies that $BT(\Lambda(\mathcal{H})) = BT(\Lambda(\mathcal{G}))$, by Fact 1.

▶ **Example 2** (continued). While applying our transformation to the safe scheme $\mathcal{G}_2$ from Example 2, we obtain a Damm-safe scheme $\mathcal{H}_2$ with the following rules (where variables x, f, g are of sorts o, $o \to o$, and $(o \to o) \to o$, respectively; we assume that $f \prec g \prec x$):

$$\lfloor S\rfloor \to \lfloor W\,(X\,(b\langle\rangle))\rfloor\,, \qquad \lfloor X\rfloor\, x\, f \to \lfloor f\, x\rfloor\, f\, x\,, \qquad \lfloor g\rfloor\, g\, f \to g\, f\,,$$
$$\lfloor W\rfloor\, g \to \lfloor Y\,(X\,(Y\, g))\rfloor\, g\,, \qquad \lfloor A\rfloor\, x \to \lfloor a\langle x\rangle\rfloor\, x\,, \qquad \lfloor x\rfloor\, x \to x\,,$$
$$\lfloor Y\rfloor\, g \to \lfloor g\, A\rfloor\, g\,, \qquad \lfloor f\rfloor\, f\, x \to f\, x\,, \qquad \lfloor b\langle\rangle\rfloor \to b\langle\rangle\,,$$
$$\lfloor W\,(X\,(b\langle\rangle))\rfloor \to \lfloor W\rfloor\, \lfloor X\,(b\langle\rangle)\rfloor\,, \qquad \lfloor Y\, g\rfloor\, g \to \lfloor Y\rfloor\, (\lfloor g\rfloor\, g)\,,$$
$$\lfloor X\,(b\langle\rangle)\rfloor\, f \to \lfloor X\rfloor\, \lfloor b\langle\rangle\rfloor\, f\,, \qquad \lfloor g\, A\rfloor\, g \to \lfloor g\rfloor\, g\, \lfloor A\rfloor\,,$$
$$\lfloor Y\,(X\,(Y\, g))\rfloor\, g \to \lfloor Y\rfloor\, (\lfloor X\,(Y\, g)\rfloor\, g)\,, \qquad \lfloor f\, x\rfloor\, f\, x \to \lfloor f\rfloor\, f\, (\lfloor x\rfloor\, x)\,,$$
$$\lfloor X\,(Y\, g)\rfloor\, g\, f \to \lfloor X\rfloor\, (\lfloor Y\, g\rfloor\, g)\, f\,, \qquad \lfloor a\langle x\rangle\rfloor\, x \to a\langle \lfloor x\rfloor\, x\rangle\,.$$ ◀

We now come to the second transformation.

▶ **Theorem 9.** *For every Damm-safe scheme $\mathcal{G} = (\mathcal{N}, \mathcal{R}, X_0)$ one can construct in logarithmic space a homogeneous Damm-safe scheme $\mathcal{H} = (\mathcal{N}', \mathcal{R}', X_0)$ that is of the same order as $\mathcal{G}$ and such that $BT(\Lambda(\mathcal{H})) = BT(\Lambda(\mathcal{G}))$.*

We remark that the transformation from the previous section (which converts a scheme to a homogeneous scheme), when applied to a Damm-safe scheme results in a scheme that is homogeneous, but no longer (Damm-)safe. Indeed, we have there (on argument positions) subterms of the form $S_{\alpha,k+1}\, M$, where $k = ord(M)$. Recalling that the order of $S_{\alpha,k+1}\, M$ is $k+1$, we notice that such a subterm is not Damm-safe (and if, e.g., M is a variable, it is also not safe).

We thus use a different approach: we reorder parameters / arguments. This approach works only because the scheme is Damm-safe. Indeed, Damm-safety ensures that when an argument of some order k is applied, then simultaneously all arguments of orders higher than k are applied, and thus we can move our argument of order k behind these arguments.

Before giving a formal definition of our transformation, let us extend the notion of Damm-safety from applicative terms to λ-terms. To this end, to the definition of a Damm-safe terms, we add an item saying that a λ-term $M = \lambda x_1.\cdots.\lambda x_s.K$ (with $s \geq 1$) is Damm-safe if K is Damm-safe, and additionally $\mathit{ord}(x_i) \geq \mathit{ord}(K)$ for all $i \in \{1, \dots, s\}$.

For sorts $\alpha_1, \dots, \alpha_s$, let $\mathit{sort}(\alpha_1, \dots, \alpha_s)$ be the permutation $(i_1, \dots, i_s)$ of $(1, \dots, s)$ for which either $\mathit{ord}(\alpha_{i_j}) = \mathit{ord}(\alpha_{i_{j+1}})$ and $i_j < i_{j+1}$, or $\mathit{ord}(\alpha_{i_j}) > \mathit{ord}(\alpha_{i_{j+1}})$, for every $j \in \{1, \dots, s\}$. Having the sorting function, we define our transformation on sorts, by induction: when $\alpha = \alpha_1 \to \cdots \to \alpha_s \to o$, and $\mathit{sort}(\alpha_1, \dots, \alpha_s) = (i_1, \dots, i_s)$, we put $\mathbf{S}(\alpha) = \mathbf{S}(\alpha_{i_1}) \to \cdots \to \mathbf{S}(\alpha_{i_s}) \to o$ (in particular $\mathbf{S}(o) = o$). Similarly, for Damm-safe λ-terms we define by coinduction:

- if $M = a\langle K_1, \dots, K_r\rangle$, then $\mathbf{S}(M) = a\langle \mathbf{S}(K_1), \dots, \mathbf{S}(K_r)\rangle$,
- if $M = x^\alpha \in \mathcal{V}$, then $\mathbf{S}(M) = x^{\mathbf{S}(\alpha)}$ (where x is either a "real" variable, or a nonterminal),
- if $M = K\, L_1^{\alpha_1} \dots L_s^{\alpha_s}$ (with $s \geq 1$), and $\mathit{sort}(\alpha_1, \dots, \alpha_s) = (i_1, \dots, i_s)$, and K is Damm-safe, then $\mathbf{S}(M) = \mathbf{S}(K)\, \mathbf{S}(L_{i_1}) \dots \mathbf{S}(L_{i_s})$, and
- finally, if $M = \lambda x_1^{\alpha_1}.\cdots.\lambda x_s^{\alpha_s}.K$ (with $s \geq 1$), and $\mathit{sort}(\alpha_1, \dots, \alpha_s) = (i_1, \dots, i_s)$, and $\mathit{ord}(x_i) \geq \mathit{ord}(K)$ for all $i \in \{1, \dots, s\}$, then $\mathbf{S}(M) = \lambda x_{i_1}^{\mathbf{S}(\alpha_{i_1})}.\cdots.\lambda x_{i_s}^{\mathbf{S}(\alpha_{i_s})}.\mathbf{S}(K)$.

Notice that for a λ-term M of sort α, the sort of $\mathbf{S}(M)$ is $\mathbf{S}(\alpha)$.

It may appear that the definition is ambiguous (but it is not). The problem is that while transforming an application $M = K\, L_1 \dots L_{k+m}$, where both K and $N = K\, L_1 \dots L_k$ are Damm-safe, we may proceed in two ways: we may sort all the arguments $L_1 \dots L_{k+m}$, but we may also separately sort the arguments $L_1 \dots L_k$ and separately the arguments $L_{k+1} \dots L_{k+m}$. We notice, though, that the effect will be the same. Indeed, we have that $\mathit{ord}(L_i) \geq \mathit{ord}(N)$ for $i \leq k$, because N is Damm-safe, and $\mathit{ord}(L_i) < \mathit{ord}(N)$ for $i > k$ because these L_i are given as arguments to N. This means that even while sorting all the arguments $L_1 \dots L_{k+m}$ together, the arguments L_i for $i \leq k$ will appear before the arguments for $i > k$. The same can be said about a sequence of λ-binders $M = \lambda x_1.\cdots.\lambda x_{k+m}.K$ in which $\mathit{ord}(x_i) \geq \mathit{ord}(\lambda x_{k+1}.\cdots.\lambda x_{k+m}.K)$ for all $i \in \{1, \dots, k\}$.

Having a transformation of λ-terms, it is immediate to define a transformation on schemes: we take $\mathcal{N}' = \{X^{\mathbf{S}(\alpha)} \mid X^\alpha \in \mathcal{N}$, and $\mathcal{R}'(X^{\mathbf{S}(\alpha)}) = \mathbf{S}(\mathcal{R}(X^\alpha))$ for all $X^\alpha \in \mathcal{N}$.

On the one hand, it should be clear that $\mathcal{H}$ is homogeneous, Damm-safe, and of the same order as $\mathcal{G}$. On the other hand, it is easy to prove the following lemma.

▶ **Lemma 10.** *If $M = (\lambda x_1.\cdots.\lambda x_s.K)\, L_1 \dots L_s$ is a Damm-safe λ-term, and $M \xrightarrow{h}{}^{s}_{\beta} N$, then N is Damm-safe, and $\mathbf{S}(M) \xrightarrow{h}{}^{s}_{\beta} \mathbf{S}(N)$.* ◀

Using the above lemma it is easy to prove by coinduction that $BT(\mathbf{S}(M)) = BT(M)$ for every Damm-safe λ-term M. Because $\Lambda(\mathcal{H}) = \mathbf{S}(\Lambda(\mathcal{G}))$, and because $\Lambda(\mathcal{G})$ is Damm-safe, it follows that $BT(\Lambda(\mathcal{H})) = BT(\Lambda(\mathcal{G}))$. Notice that in Lemma 10 it is essential that we perform all the s head β-reductions at once, not only a single one (since in $\mathbf{S}(M)$ the s arguments are applied in different order than in M).

▶ **Example 2** (continued). Let us apply the transformation to the Damm-safe scheme $\mathcal{H}_2$ from our example. Since $\lfloor X \rfloor$ is the only nonterminal having a non-homogeneous sort, only the rules involving $\lfloor X \rfloor$ are changed, as follows:

$$\lfloor X \rfloor\, f\, x \to \lfloor f\, x \rfloor\, f\, x\,, \qquad \lfloor X\, (Y\, g) \rfloor\, g\, f \to \lfloor X \rfloor\, f\, (\lfloor Y\, g \rfloor\, g)\,,$$

$$\lfloor X\, (b\langle\rangle) \rfloor\, f \to \lfloor X \rfloor\, f\, \lfloor b\langle\rangle \rfloor\,.$$

Notice that it does not make sense to apply the transformation to the scheme $\mathcal{G}_2$, which is not Damm-safe. Indeed, it would be impossible to swap the order of the parameters of X, since in the subterm $X\,(Y\,g)$ we are applying only one argument to X. ◀

5 Consequences of Homogeneity

Let us say that a λ-term is homogeneous if sorts of all its subterms are homogeneous. By definition this means that arguments of higher order are always applied before arguments of lower order. Due to this fact, in a homogeneous λ-term (unlike in an arbitrary λ-term) we can perform β-reductions starting from redexes concerning variables of the highest order. In this section we formalize and prove this property of homogeneous λ-terms (Lemmata 11 and 12). We remark that this property turned out to be useful e.g. in Parys [17].

We define the order of a β-reduction as the order of the involved variable. More precisely, for a number $k \in \mathbb{N}$, the relation $\to_{\beta(k)}$ of *β-reduction of order k* is defined as the compatible closure of the relation $\{((\lambda x.K)\,L, K[L/x]) \mid \mathit{ord}(x) = k\}$.

We first give our result for finite λ-terms.

▶ **Lemma 11.** *Let M be a finite closed homogeneous λ-term of sort o and complexity at most n. Then there exist λ-terms $N_n, N_{n-1}, \dots, N_0$ such that $M = N_n$, and for every $k \in \{0, \dots, n-1\}$, N_k is of complexity at most k and $N_{k+1} \to^*_{\beta(k)} N_k$, and $N_0 = BT(M)$.*

For infinite λ-terms we need to be slightly more careful: it is not enough to replace the reflexive transitive closure $\to^*_{\beta(k)}$ by the infinitary closure $\to^\infty_{\beta(k)}$. The problem lies in subterms which do not have so-called head normal form: infinite applications $\dots K_3\,K_2\,K_1$, and subterms from which we can perform infinitely many head β-reductions. These are subterms responsible for creating nodes labeled by $\bot$ in the Böhm tree. We cannot deal with these subterms by only applying β-reductions. We need to introduce relations that explicitly replace such "invalid" subterms by $\bot\langle\rangle$.

The relation $\xrightarrow{h}_{\beta(k)}$ of *head β-reduction of order k* (where $k \in \mathbb{N}$) is defined as

$$\{((\lambda x.K)\,L\,P_1\,\dots\,P_n, K[L/x]\,P_1\,\dots\,P_n) \mid \mathit{ord}(x) = k\}\,.$$

Consider now the relation containing all pairs of the form $(K, \lambda x_1.\cdots.\lambda x_s.\bot\langle\rangle)$, where K and $\lambda x_1.\cdots.\lambda x_s.\bot\langle\rangle$ are of the same sort, and either for every $n \in \mathbb{N}$ there is L such that $K \xrightarrow{h}{}^n_{\beta(k)} L$, or K is an infinite application. The compatible closure of this relation is denoted $\to_{\bot(k)}$. By $\to_{\beta\bot(k)}$ we denote the union of $\to_{\beta(k)}$ and $\to_{\bot(k)}$. Using this relation we can now reformulate Lemma 11 for infinite λ-terms.

▶ **Lemma 12.** *Let M be a closed homogeneous λ-term of sort o and complexity at most n. Then there exist λ-terms $N_n, N_{n-1}, \dots, N_0$ such that $M = N_n$, and for every $k \in \{0, \dots, n-1\}$, N_k is of complexity at most k and $N_{k+1} \to^\infty_{\beta\bot(k)} N_k$, and $N_0 = BT(M)$.*

Notice that Lemma 11 is an immediate consequence of Lemma 12, because when a λ-term K is finite, then there is no L such that $K \to_{\bot(k)} L$, and $K \to^\infty_{\beta(k)} M$ implies $K \to^*_{\beta(k)} M$ (every sequence of β-reductions from a finite λ-term is finite). Lemma 12, in turn, is a consequence of the following lemma.

▶ **Lemma 13.** *Let M be a λ-term of complexity at most k, order at most $k-1$, and such that all free variables of M have order at most $k-1$. Then there exists a λ-term P of complexity at most $k-1$ such that $M \to^\infty_{\beta\bot(k)} P$.*

Proof. The proof is by coinduction. Suppose first that for every $n \in \mathbb{N}$ there is N such that $M \xrightarrow{h}{}^{n}_{\beta(k)} N$. In this situation $M \to_{\bot(k)} \lambda x_1.\cdots.\lambda x_s.\bot\langle\rangle$ (for an appropriate sequence of variables $x_1, \ldots, x_s$, corresponding to the sort of M). Denoting the latter λ-term P we obtain the thesis, since the complexity of P equals $\mathit{ord}(P) = \mathit{ord}(M) \leq k-1$.

The opposite case is that $M \xrightarrow{h}{}^{*}_{\beta(k)} N$ for some N, but there is no N' such that $N \xrightarrow{h}_{\beta(k)} N'$. When N is a variable, the thesis is trivial for $P = N$, and when $N = a\langle K_1, \ldots, K_r\rangle$, the thesis follows directly from the assumption of coinduction. When $N = \lambda x.K$, the thesis also follows from the assumption of coinduction; we only need to observe that $\mathit{ord}(N) = \mathit{ord}(M) \leq k-1$ implies that $\mathit{ord}(x) \leq k-2 \leq k-1$. Suppose thus that N is an application. When N is an infinite application, we again have $M \to_{\bot(k)} \lambda x_1.\cdots.\lambda x_s.\bot\langle\rangle$, and we are done. When $N = x\,L_1\,\ldots\,L_s$, by assumption the order of x is at most $k-1$, so we can simply use the assumption of coinduction for all L_i. Otherwise N is of the form $(\lambda x.K)\,L_1\,\ldots\,L_s$. Since no head β-reduction of order k starts in N, necessarily $\mathit{ord}(x) \neq k$. Knowing that the complexity of N is at most k, and that the sort of $\lambda x.K$ is homogeneous, this implies that $\mathit{ord}(\lambda x.K) = \mathit{ord}(x) - 1 \leq k-1$. We can thus again use the assumption of coinduction for all the subterms $K, L_1, \ldots, L_s$. ◀

▶ Remark. We notice that Lemmata 11 and 12 would be false if we have allowed λ-terms involving non-homogeneous sorts. For example, from a λ-term of the form $(\lambda x.\lambda y.K)\,L\,M$ with $\mathit{ord}(x) = 0$ and $\mathit{ord}(y) = 1$ we have to perform a β-reduction of order 0 concerning x before a β-reduction of order 1 concerning y. It is, though, possible to reformulate our lemmata without the homogeneity assumption. One only has to define the order of a β-reduction $(\lambda x.K)\,L \to_\beta K[L/x]$ in a less natural way, as $\mathit{ord}(\lambda x.K) - 1$, not as $\mathit{ord}(x)$ (notice that these two numbers coincide for homogeneous sorts).

Acknowledgment. We thank the anonymous reviewers of the previous version of this paper for some useful comments.

References

1. Alfred V. Aho. Indexed grammars - an extension of context-free grammars. *J. ACM*, 15(4):647–671, 1968. doi:10.1145/321479.321488.
2. William Blum. Type homogeneity is not a restriction for safe recursion schemes. *CoRR*, abs/1701.02118, 2017. URL: http://arxiv.org/abs/1701.02118.
3. William Blum and C.-H. Luke Ong. The safe lambda calculus. *Logical Methods in Computer Science*, 5(1), 2009. doi:10.2168/LMCS-5(1:3)2009.
4. Luca Breveglieri, Alessandra Cherubini, Claudio Citrini, and Stefano Crespi-Reghizzi. Multi-push-down languages and grammars. *Int. J. Found. Comput. Sci.*, 7(3):253–292, 1996. doi:10.1142/S0129054196000191.
5. Christopher H. Broadbent. *On collapsible pushdown automata, their graphs and the power of links.* PhD thesis, University of Oxford, 2011. URL: https://ora.ox.ac.uk/objects/uuid:aaa328fe-60be-4abe-8f84-97dbd9e0a3fe.
6. Arnaud Carayol and Olivier Serre. Collapsible pushdown automata and labeled recursion schemes: Equivalence, safety and effective selection. In Proceedings of the 27th Annual [18], pages 165–174. doi:10.1109/LICS.2012.73.
7. Lorenzo Clemente, Paweł Parys, Sylvain Salvati, and Igor Walukiewicz. Ordered tree-pushdown systems. In Prahladh Harsha and G. Ramalingam, editors, *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science,*

FSTTCS 2015, December 16-18, 2015, Bangalore, India, volume 45 of *LIPIcs*, pages 163–177. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPIcs.FSTTCS.2015.163.

8 Łukasz Czajka. Coinductive techniques in infinitary lambda-calculus. *CoRR*, abs/1501.04354, 2015. URL: http://arxiv.org/abs/1501.04354.

9 Werner Damm. The IO- and OI-hierarchies. *Theor. Comput. Sci.*, 20:95–207, 1982. doi:10.1016/0304-3975(82)90009-3.

10 Matthew Hague, Andrzej S. Murawski, C.-H. Luke Ong, and Olivier Serre. Collapsible pushdown automata and recursion schemes. In *Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science, LICS 2008, 24-27 June 2008, Pittsburgh, PA, USA*, pages 452–461. IEEE Computer Society, 2008. doi:10.1109/LICS.2008.34.

11 Richard Kennaway, Jan Willem Klop, M. Ronan Sleep, and Fer-Jan de Vries. Infinitary lambda calculus. *Theor. Comput. Sci.*, 175(1):93–125, 1997. doi:10.1016/S0304-3975(96)00171-5.

12 Richard Kennaway, Vincent van Oostrom, and Fer-Jan de Vries. Meaningless terms in rewriting. *Journal of Functional and Logic Programming*, 1999(1), 1999.

13 Teodor Knapik, Damian Niwinski, and Paweł Urzyczyn. Deciding monadic theories of hyperalgebraic trees. In *TLCA*, pages 253–267, 2001. doi:10.1007/3-540-45413-6_21.

14 Teodor Knapik, Damian Niwiński, and Paweł Urzyczyn. Higher-order pushdown trees are easy. In Mogens Nielsen and Uffe Engberg, editors, *Foundations of Software Science and Computation Structures, 5th International Conference, FOSSACS 2002. Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2002 Grenoble, France, April 8-12, 2002, Proceedings*, volume 2303 of *Lecture Notes in Computer Science*, pages 205–222. Springer, 2002. doi:10.1007/3-540-45931-6_15.

15 Gregory M. Kobele and Sylvain Salvati. The IO and OI hierarchies revisited. *Inf. Comput.*, 243:205–221, 2015. doi:10.1016/j.ic.2014.12.015.

16 Paweł Parys. On the significance of the collapse operation. In Proceedings of the 27th Annual [18], pages 521–530. doi:10.1109/LICS.2012.62.

17 Paweł Parys. The complexity of the diagonal problem for recursion schemes. In Satya V. Lokam and R. Ramanujam, editors, *37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2017, December 11-15, 2017, Kanpur, India*, volume 93 of *LIPIcs*, pages 45:1–45:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.FSTTCS.2017.45.

18 *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25-28, 2012*. IEEE Computer Society, 2012.

19 Sylvain Salvati and Igor Walukiewicz. Simply typed fixpoint calculus and collapsible pushdown automata. *Mathematical Structures in Computer Science*, 26(7):1304–1350, 2016. doi:10.1017/S0960129514000590.