The Computability Path Order for Beta-Eta-Normal Higher-Order Rewriting

Johannes Niederhauser(✉)[0000−0002−8662−6834] and Aart Middeldorp[0000−0001−7366−8464]

Department of Computer Science, University of Innsbruck, Innsbruck, Austria
{johannes.niederhauser,aart.middeldorp}@uibk.ac.at

Abstract. We lift the computability path order and its extensions from plain higher-order rewriting to higher-order rewriting on $\beta\eta$-normal forms where matching modulo $\beta\eta$ is employed. The resulting order NCPO is shown to be useful on practical examples. In particular, it can handle systems where its cousin NHORPO fails even when it is used together with the powerful transformation technique of neutralization. We also argue that automating NCPO efficiently is straightforward using SAT/SMT solvers whereas this cannot be said about the transformation technique of neutralization. Our prototype implementation supports automatic termination proof search for NCPO and is also the first one to automate NHORPO with neutralization.

Keywords: Higher-Order Rewriting · Termination · Lambda Calculus

1 Introduction

Higher-order rewriting is known for its abundance of different formalisms where it is often unclear whether results for a particular notion of higher-order rewriting can be transferred to other kinds of higher-order rewriting [12]. In this paper, we are only concerned with one particular formalism, namely a slightly modified version of the higher-order rewrite systems (HRSs) à la Nipkow [16]. However, as the goal of this paper is to lift a particular termination method from one formalism to another, a short discussion about differences and similarities of the used formalisms is necessary. First and foremost, when we talk about higher-order rewriting in this paper, we mean the particular flavor of higher-order rewriting which considers terms of the simply-typed lambda calculus as objects to be rewritten. Within this class of higher-order rewrite formalisms, we further distinguish between *plain* and *normal* higher-order rewriting: Plain higher-order rewriting uses plain syntactic matching and views lambda calculus' β-reduction as a proper rewrite rule which has to be oriented for termination. On the other hand, normal higher-order rewriting uses matching modulo $\beta\eta$, which means that terms which are $\beta\eta$-equivalent should also be equivalent with respect to the reduction order that is used for establishing termination. Nipkow's HRSs are a premier example of normal higher-order rewriting. HRSs are an interesting formalism for automated reasoning as they can directly represent the terms of

higher-order logic and as such are part of the meta-theory of interactive theorem provers like Isabelle [21]. Furthermore, they come with a critical pair lemma [16] which is vital for automated equational reasoning with techniques such as completion [7]. In particular, automated confluence analysis of higher-order rewriting typically uses HRSs, as witnessed in the Confluence Competition (2015–2020).[1]

While many termination methods are available for plain higher-order rewriting [2, 10, 12], the situation for normal higher-order rewriting is different. For a long time, powerful termination methods for HRSs were only based on van de Pol's results [22] which employ the monotone algebra approach [25], i.e., semantic termination arguments which are difficult to automate except for more specialized cases like polynomial interpretations of fixed shape [5]. An early version of the *higher-order recursive path order* (HORPO) which is designed for plain higher-order rewriting was adapted for the important subclass of pattern HRSs in [23], an implementation of this approach is available in the tool CSIˆho [19]. Furthermore, the formalism of *algebraic functional systems with meta-variables* (AFSMs) captures both plain and normal rewriting and therefore makes a larger number of termination methods available for HRSs [12]. An implementation of these methods is available in the tool WANDA [13]. However, WANDA has not been optimized for HRSs and the theory developed around it only establishes termination for systems where all left-hand sides are patterns [17]. A powerful recursive path order specifically designed for HRSs and without the restriction to patterns as left-hand sides of rules was missing until Jouannaud and Rubio lifted their definite version of HORPO [10] from plain to normal higher-order rewriting, resulting in NHORPO [11]. However, the important extension of HORPO by the computability closure [10] is not considered. Instead, a novel transformation technique named *neutralization* is introduced. Its goal is to simplify a given system with respect to the applicability of NHORPO without affecting the system's termination behavior.

The aim of this paper is to develop a syntax-directed reduction order for normal higher-order rewriting which is sufficiently powerful compared to NHORPO with neutralization but easier to automate using SAT/SMT solvers. Our starting point is the *computability path order* (CPO) [2] which is defined for plain higher-order rewriting. CPO is an extension of HORPO which incorporates computability closure [3] in a sophisticated way. We follow the general approach from [11] to lift the extension of core CPO with accessible subterms and small symbols to HRSs, resulting in the *$\beta\eta$-normal computability path order* (NCPO). In particular, we will show that there are HRSs which NCPO can prove terminating but where NHORPO (with or without neutralization) as well as the HORPO implementations in WANDA [13] and CSIˆho [19] fail.

In designing a reduction order for normal higher-order rewriting, one has to deal with the well-known problems that well-founded orders which are compatible with $\beta\eta$ cannot be *monotone* (closed under contexts) and *stable* (closed under substitutions) in general as the following examples show:

[1] https://project-coco.uibk.ac.at/

Example 1. Consider $\mathsf{a} > \mathsf{b}$. If $>$ is closed under contexts, we should also have $(\lambda y.\mathsf{c})\mathsf{a} > (\lambda y.\mathsf{c})\mathsf{b}$. If we also want $>$ to be compatible with $\beta\eta$, we obtain $\mathsf{c} > \mathsf{c}$ which means that $>$ cannot be well-founded. Now consider $Fx > x$. If $>$ is closed under substitutions, we should also have $(\lambda x.x)\mathsf{a} > \mathsf{a}$. If we also want $>$ to be compatible with $\beta\eta$, we obtain $\mathsf{a} > \mathsf{a}$ which again means that $>$ cannot be well-founded.

In [22], van de Pol models substitutions via contexts using β-reduction, thereby eliminating the second problem. For the first problem, two different orders $>_1$ and $>_2$ are introduced and it is shown that $s >_1 t$ implies $C[s] >_2 C[t]$ for all terms s, t and contexts C. Since the relations between $>_1$ and $>_2$ are intrinsically semantic, it is not clear how this approach can be transferred to path orders. In [11], both problems were solved by restricting the order $>$ to $\beta\eta$-normal forms and layering it with its plain version ($\sqsupset$) in a way such that $s > t$ implies $s\sigma{\downarrow} \sqsupset^+ t\sigma{\downarrow}$ for all substitutions σ where $s{\downarrow}$ denotes the $\beta\eta$-normal form of a term s. Then, monotonicity and well-foundedness of the plain version can be used in order to show that $(>, \sqsupset)$ is a reduction order. We will utilize this approach in order to define a reduction order $(>, \sqsupset)$ where $>$ is NCPO and $\sqsupset$ is CPO.

The remainder of this paper is structured as follows. Section 2 introduces the basic concepts including the type of higher-order rewriting which we are concerned with in this paper as well as an appropriate notion of $\beta\eta$-normal higher-order reduction orders. Section 3 describes the necessary ingredients of our order regarding types, the important concept of nonversatile terms as well as accessible subterms. Based on this, NCPO is introduced in Section 4 and proven to be a correct termination method for HRSs in Section 5. Finally, our prototype implementation of NCPO is described and compared to NHORPO with neutralization in Section 6 before we conclude in Section 7. The full version of this paper which includes all proof details is available on arXiv [20].

2 Preliminaries

In this paper, we consider higher-order rewriting on simply-typed lambda terms [1,4]. Given a set $\mathcal{B}$ of base types, the set of simple types $\mathcal{T}$ is defined inductively: If $a \in \mathcal{B}$ then $a \in \mathcal{T}$, and for $U, V \in \mathcal{T}$ also $U \to V \in \mathcal{T}$. We follow the usual convention that the function space constructor $\to$ is right-associative, i.e., $a \to b \to c$ denotes $a \to (b \to c)$. Throughout this text, lowercase letters $a, b, c, \ldots$ denote base types while upper case letters $T, U, V, \ldots$ denote arbitrary types. For each type $U \in \mathcal{T}$ we consider an infinite set of variables $\mathcal{V}_U$ as well as a set of function symbols $\mathcal{F}_U$ where $\mathcal{V}_U \cap \mathcal{F}_U = \varnothing$ and $\mathcal{V}_U \cap \mathcal{V}_V = \mathcal{F}_U \cap \mathcal{F}_V = \varnothing$ for $V \neq U$. We denote the set of all variables by $\mathcal{V} = \bigcup\{\mathcal{V}_U \mid U \in \mathcal{T}\}$. The set of all function symbols $\mathcal{F} = \bigcup\{\mathcal{F}_U \mid U \in \mathcal{T}\}$ is referred to as the *signature*. We associate with each function symbol f an arity $\mathsf{ar}(f) \in \mathbb{N}$. The set of well-typed lambda terms of a type U (Λ_U) is defined as follows:

- $\mathcal{V}_U \subseteq \Lambda_U$,

- if $f \in \mathcal{F}_{T_1 \to \cdots \to T_{\mathsf{ar}(f)} \to U}$ and $t_i \in \Lambda_{T_i}$ for $1 \leqslant i \leqslant \mathsf{ar}(f)$ then $f(t_1, \ldots, t_n) \in \Lambda_U$,
- if $x \in \mathcal{V}_U$ and $s \in \Lambda_V$ then $\lambda x.s \in \Lambda_{U \to V}$,
- if $s \in \Lambda_{U \to V}$ and $t \in \Lambda_U$ then $st \in \Lambda_V$.

The set $\Lambda = \bigcup\{\Lambda_U \mid U \in \mathcal{T}\}$ contains all well-typed lambda terms. We define the function $\tau\colon \Lambda \to \mathcal{T}$ as $\tau(s) = U$ if $s \in \Lambda_U$. Throughout this paper we only consider well-typed lambda terms. We also follow the convention that application is left-associative, i.e. stu denotes $(st)u$. Note that our definition of lambda terms with function symbols of fixed arity neither poses a limitation nor adds expressive power as we can always set $\mathsf{ar}(f) = 0$ for all $f \in \mathcal{F}$ (the "return type" of a function symbol does not have to be a base type) and denote $f(t_1, \ldots, t_n)$ by the corresponding application $ft_1 \cdots t_n$. The addition of function symbols with fixed arity is solely needed for an adequate definition of a recursive path order on lambda terms. We sometimes use the shorthand $f(\bar{t})$ to denote the application of f to the list of arguments $\bar{t}$.

We write $\mathsf{FV}(s)$ for the set of free variables of a term s. The term $s[x/t]$ denotes the term where all free occurrences of x have been replaced by t without capturing the free variables of t (*capture-avoiding substitution*). Due to the fact that infinitely many variables are available for each type, this can always be done by renaming the bound variables accordingly (α*-renaming*). In the remainder, we abstract away from this technicality by viewing lambda terms as equivalence classes modulo α-renaming.

Two fundamental concepts in lambda calculus are the notions of β- and η-reduction which are defined as the rule schemas $(\lambda x.s)t \to_\beta s[x/t]$ and $\lambda x.ux \to_\eta u$ if $x \notin \mathsf{FV}(u)$. Note that both β- and η-reduction preserve types. Every term s has a unique $\beta\eta$-normal form which we denote by $s{\downarrow}$. Rewriting to $\beta\eta$-normal form and the set of $\beta\eta$-normal forms are denoted by $\to_{\beta\eta}^!$ and $\mathsf{NF}(\beta\eta) \subseteq \Lambda$, respectively.

A substitution σ is a mapping from variables to terms of the same type where $\mathcal{D}\mathsf{om}(\sigma) = \{x \mid \sigma(x) \neq x\}$ is finite. We often write σ as a set of variable bindings. Given a substitution $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$, we define $s\sigma$ as the simultaneous capture-avoiding substitution $s[x_1/t_1, \ldots, x_n/t_n]$. The free variables of a substitution are defined as follows: $\mathsf{FV}(\sigma) = \bigcup\{\mathsf{FV}(\sigma(x)) \mid x \in \mathcal{D}\mathsf{om}(\sigma)\}$. A substitution is said to be *away from* a finite set of variables X if $(\mathcal{D}\mathsf{om}(\sigma) \cup \mathsf{FV}(\sigma)) \cap X = \varnothing$. We follow the convention that the postfix operations of substitution application as well as $\downarrow$ bind stronger than lambda abstractions and applications, i.e., $u\sigma{\downarrow}v\sigma{\downarrow} = (u\sigma{\downarrow})(v\sigma{\downarrow})$ and $\lambda x.t\sigma{\downarrow} = \lambda x.(t\sigma{\downarrow})$. If $\bar{t} = (t_1, \ldots, t_n)$, we allow ourselves to write $\bar{t}\sigma$ $(\bar{t}\sigma{\downarrow})$ as a shorthand for $(t_1\sigma, \ldots, t_n\sigma)$ $((t_1\sigma{\downarrow}, \ldots, t_n\sigma{\downarrow}))$.

Contexts C are lambda terms which contain exactly one occurrence of the special symbol $\Box$ which can assume any type. We write $C[s]$ for the lambda term C where $\Box$ is replaced by s without employing capture-avoiding substitution. A binary relation $R \subseteq \Lambda \times \Lambda$ is *monotone* if $s\ R\ t$ implies $C[s]\ R\ C[t]$ for every context C. Furthermore, we say that a term t is a *subterm* of s ($s \trianglerighteq t$) if there exists a context C such that $s = C[t]$ and define the proper subterm relation $s \triangleright t$

as $s \triangleright t$ and $s \neq t$. Since we view lambda terms as equivalence classes modulo α-renaming, the subterm relation is also defined modulo α-renaming. Hence, we have e.g. $\lambda x.t \triangleright t[x/z]$ for a fresh variable z. Now, we are able to define the notion of higher-order rewriting we are concerned with in this paper.

Definition 1. *A* rewrite rule *is a tuple* $\ell \to r$ *with* $\ell, r \in \mathsf{NF}(\beta\eta)$ *where* ℓ *is not of the form* $xs_1 \cdots s_n$, $\tau(\ell) = \tau(r)$ *and* $\mathsf{FV}(r) \subseteq \mathsf{FV}(\ell)$. *A* higher-order rewrite system (HRS) *is a set of rewrite rules. Given an* HRS $\mathcal{R}$, *there is a* rewrite step $s \to_{\mathcal{R}} t$ *if there exist a rule* $\ell \to r \in \mathcal{R}$, *a substitution* σ *and a context* C *such that* $s = C[\ell\sigma\downarrow] \in \mathsf{NF}(\beta\eta)$ *and* $t = C[r\sigma\downarrow]\downarrow$.

Note that $\to_{\mathcal{R}} \subseteq \mathsf{NF}(\beta\eta) \times \mathsf{NF}(\beta\eta)$ as $C[\ell\sigma\downarrow], C[r\sigma\downarrow]\downarrow \in \mathsf{NF}(\beta\eta)$ by definition. Hence, both rules and rewrite steps only consider terms in their unique $\beta\eta$-normal form whereas matching is performed modulo $\beta\eta$. In the original definition of HRSs [16], the long $\beta\eta$-normal form based on η-expansion instead of η-reduction is used. Using $\beta\eta$-normal forms instead of long $\beta\eta$-normal forms leads to a different rewrite relation and might change the termination behavior of a given rewrite system (cf. Example 2). The reason of our deviation from the standard definition of Nipkow's HRSs is that higher-order recursive path orders usually have a hard time dealing with lambdas, and we wanted keep Theorem 1 simple by using the same canonical form for rewriting as well as higher-order reduction orders. Furthermore, higher-order rewriting on $\beta\eta$-normal forms is interesting in its own right [8, 9]. Note that NHORPO [11] can accommodate any orientation of η by adding an appropriate wrapper around the core order which is only defined on $\beta\eta$-normal forms. We expect that a similar strategy works for NCPO, thus making it applicable to the original formulation of HRSs by Nipkow. Another difference from the usual definition of HRSs is that we allow rules of non-base type. However, this is not new [9]. In all other aspects, our definition is equivalent to the original one given in [16].

Example 2. Let $\mathcal{B} = \{a\}$ and consider the function symbols $\mathsf{c} \in \mathcal{F}_a$, $\mathsf{f} \in \mathcal{F}_{a \to a}$ and $\mathsf{g} \in \mathcal{F}_{a \to a \to a}$ as well as the variable $x \in \mathcal{V}_a$. The $\beta\eta$-normal term f cannot be rewritten using the HRS $\mathcal{R}$ consisting of the single rule $\mathsf{f}x \to \mathsf{g}x\mathsf{c}$. However, for its η-long normal form $\lambda y.\mathsf{f}y$ we have $\lambda y.\mathsf{f}y \to_{\mathcal{R}} \lambda y.\mathsf{g}y\mathsf{c}$ in the standard definition of HRSs.

Next, we recall some important definitions about relations which are used throughout this paper. Given a binary relation R, R^+ and R^* denote its transitive and transitive-reflexive closure, respectively. The composition of a two binary relations R and S is defined as follows: $a \; R \cdot S \; b$ if there exists an element c such that $a \; R \; c$ and $c \; S \; b$. A *preorder* is a reflexive and transitive relation. Given a relation $>$ we denote its inverse by $<$ and its reflexive closure by $\geqslant$. Note that the reflexive closure of a binary relation R defined as $R \subseteq A \times A$ contains all elements of the set A even when the strict part of R assumes additional properties of the elements in A which belong to the relation. A binary relation R on a set A is *well-founded* if there is no infinite sequence $a_1 \; R \; a_2 \; R \cdots$ where $a_i \in A$ for $i \in \mathbb{N}$. We say that an HRS $\mathcal{R}$ is *terminating* if $\to_{\mathcal{R}}$ is well-founded. We now define the notion of $\beta\eta$-normal higher-order reduction orders.

Definition 2. *A* $\beta\eta$-normal higher-order reduction order *is a pair* $(>, \sqsupset)$ *which satisfies the following conditions:*

- $\sqsupset \subseteq \Lambda \times \Lambda$ *is a well-founded relation,*
- $\sqsupset$ *is monotone,*
- $\to_\beta, \to_\eta \subseteq \sqsupset$,
- $s > t$ *implies* $s\sigma{\downarrow} \sqsupset^+ t\sigma{\downarrow}$ *for all* $s, t \in \mathsf{NF}(\beta\eta)$ *and substitutions* σ.

We often refer to the last condition as $\beta\eta$-normal stability. *An* HRS $\mathcal{R}$ *is* compatible *with a* $\beta\eta$*-normal higher-order reduction order* $(>, \sqsupset)$ *if* $\ell >^+ r$ *for all* $\ell \to r \in \mathcal{R}$.

As in [11], the intuition behind this definition is that $>$ will be used to orient the rules of HRSs while relying on the termination proof of its plain variant $\sqsupset$. Despite calling $(>, \sqsupset)$ an order, we do not demand transitivity of any of its components. In the context of higher-order rewriting, this is standard as $\sqsupset$ contains β-reduction which is not transitive. By taking the identity substitution, we can see that $\beta\eta$-normal stability implies $> \subseteq \sqsupset^+$. The following theorem shows that $\beta\eta$-normal higher-order reduction orders can be used to show termination of our flavor of HRSs.

Theorem 1. *If an* HRS $\mathcal{R}$ *is compatible with a* $\beta\eta$*-normal higher-order reduction order* $(>, \sqsupset)$*, then* $\mathcal{R}$ *is terminating.*

Proof. For a proof by contradiction, consider an infinite rewrite sequence $s_1 \to_\mathcal{R} s_2 \to_\mathcal{R} \cdots$. By definition, $s_1 = C[\ell\sigma{\downarrow}]$ and $s_2 = C[r\sigma{\downarrow}]{\downarrow}$ for some $\ell \to r \in \mathcal{R}$ where σ is a substitution and C is a context such that $C[\ell\sigma{\downarrow}] \in \mathsf{NF}(\beta\eta)$. By assumption, $\ell >^+ r$, so $\ell\sigma{\downarrow} \sqsupset^+ r\sigma{\downarrow}$ follows from $\beta\eta$-normal stability and we obtain $C[\ell\sigma{\downarrow}] \sqsupset^+ C[r\sigma{\downarrow}]$ by monotonicity of $\sqsupset$. Finally, $C[(\ell\sigma){\downarrow}] \sqsupset^+ C[(r\sigma){\downarrow}]{\downarrow}$ since $\sqsupset$ contains $\beta\eta$-reduction. Hence, we can transform the infinite sequence $s_1 \to_\mathcal{R} s_2 \cdots$ into the infinite descending sequence $s_1 \sqsupset^+ s_2 \cdots$ which contradicts well-foundedness of $\sqsupset$. Thus, $\mathcal{R}$ is terminating. □

3 Ingredients of the Order

We start by recalling the notion of nonversatile terms from [11, Definition 4.1]. Intuitively, a term is nonversatile if the application of any substitution together with the subsequent $\beta\eta$-normalization only affects its proper subterms. As this is needed in the inductive proof of $\beta\eta$-normal stability, an NCPO comparison $s > t$ is only defined for nonversatile s. The sufficient condition for nonversatility given in the subsequent lemma is an improvement over the one given in [11, Lemma 4.2].[2]

Definition 3. *The set* $\Lambda_{\mathsf{nv}} \subseteq \mathsf{NF}(\beta\eta)$ *of* nonversatile *terms is defined as follows:*

[2] In fact, subgoal 3 of Example 4.8 in [11] cannot be handled with the corresponding sufficient condition.

- $\mathcal{V} \cap \Lambda_{\mathsf{nv}} = \varnothing$,
- $f(\bar{t}) \in \Lambda_{\mathsf{nv}}$ *for all* $f \in \mathcal{F}$ *and* $\bar{t} \in \Lambda^{\mathsf{ar}(f)}$,
- $uv \in \Lambda_{\mathsf{nv}}$ *if* $(uv)\sigma{\downarrow} = u\sigma{\downarrow}v\sigma{\downarrow}$ *for all substitutions* σ,
- $\lambda x.u \in \Lambda_{\mathsf{nv}}$ *if* $(\lambda x.u)\sigma{\downarrow} = \lambda x.u\sigma{\downarrow}$ *for all substitutions* σ.

Terms which are not nonversatile are said to be versatile.

Example 3. The terms $\mathsf{c}x$ and $\lambda x.\mathsf{f}(yx)$ are nonversatile while the terms yx and $\lambda x.\mathsf{f}(yx)x$ are versatile as can be seen by taking the substitution $\{y \mapsto \lambda z.\mathsf{d}\}$.

Lemma 1. *The following terms are nonversatile if they are* $\beta\eta$*-normal forms:*

(i) *applied function symbols* $f(\bar{t})$,
(ii) *applications* uv *with nonversatile* u,
(iii) *abstractions* $\lambda x.ux$ *where* $ux \trianglerighteq vw$ *implies that* $vw \in \Lambda_{\mathsf{nv}}$,
(iv) *abstractions* $\lambda x.u$ *with* $u \neq vx$ *where*
  - $u \in \mathcal{V} \cup \Lambda_{\mathsf{nv}}$,
  - *if* $u = v(w_1w_2)$ *then* $w_1w_2 \in \Lambda_{\mathsf{nv}}$,
  - *if* $u = v(\lambda y.w)$ *then* $\lambda y.w \in \Lambda_{\mathsf{nv}}$.

Besides the usual inference rules on terms, reduction orders derived from HORPO [10] require appropriate orders on types. The next definition recalls the notion of admissible type orders from CPO [2].

Definition 4. *We define the left (right) argument relation on types* $\rhd_l$ $(\rhd_r)$ *as follows:* $T \to U \rhd_l T$ $(T \to U \rhd_r U)$. *An order* $\succ_{\mathcal{T}}$ *on types is admissible if*

- $\rhd_r \subseteq \succ_{\mathcal{T}}$,
- $\succcurlyeq = (\succ_{\mathcal{T}} \cup \rhd_l)^+$ *is well-founded,*
- *if* $T \to U \succ_{\mathcal{T}} V$ *then* $U \succeq_{\mathcal{T}} V$ *or* $V = T \to U'$ *with* $U \succ_{\mathcal{T}} U'$.

Given a type T *and* $a \in \mathcal{B}$ *we write* $a \succcurlyeq_{\mathcal{B}} T$ $(a \succcurlyeq_{\mathcal{B}} T)$ *if* $a \succcurlyeq b$ $(a \succcurlyeq b)$ *for every* $b \in \mathcal{B}$ *occurring in* T.

Lemma 2 (Lemma 2.3 from [2]). *Given a well-founded order* $\succ_{\mathcal{B}}$ *on base types, let* $\succ_{\mathcal{T}}$ *be the smallest order on types containing* $\succ_{\mathcal{B}}$ *and* $\rhd_r$ *such that* $V \succ_{\mathcal{T}} V'$ *implies* $U \to V \succ_{\mathcal{T}} U \to V'$ *for all* U, V, V'. *Then,* $>_{\mathcal{T}}$ *is admissible.*

Example 4. Let $\mathcal{B} = \{a, b, c\}$ where $a \succ_{\mathcal{B}} b \succ_{\mathcal{B}} c$ and consider the order $\succ_{\mathcal{T}}$ defined in the previous lemma. We have $a \to b \succ_{\mathcal{T}} a \to c$ because $b \succ_{\mathcal{T}} c$. However, $c \succ_{\mathcal{T}} b \to b$ does not hold because $b \to b \rhd_r b$ and therefore $c \succ_{\mathcal{T}} b \to b \succ_{\mathcal{T}} b$ which contradicts well-foundedness of $\succ_{\mathcal{T}}$ together with $b \succ_{\mathcal{T}} c$.

Unlike their first-order versions, higher-order recursive path orders do not enjoy the subterm property because we cannot have $\mathsf{f}(\mathsf{g}(x)) > x$ in general: If $\mathsf{f} \in \mathcal{F}_{a \to a \to a}$ and $\mathsf{g} \in \mathcal{F}_{(a \to a) \to a}$ for some $a \in \mathcal{B}$ then this is an encoding of the untyped lambda calculus (where f represents application and g represents abstraction) and therefore not terminating. Thus, if we want to recursively define $f(\bar{t}) > v$ by $t_i \succcurlyeq v$ for some i, we usually have to check whether $\tau(t_i) \succeq_{\mathcal{T}} \tau(v)$

holds for the given admissible type order $\succ_{\mathcal{T}}$. In particular, this means that establishing $s > t$ by choosing some $s \triangleright s'$ and showing $s' \geqslant t$ is only possible if we check that there is a weak decrease in the admissible type order for all intermediate terms in the recursive definition of $\triangleright$. CPO extends HORPO by allowing these checks to be dismissed for the special case of accessible subterms which are determined by type-related properties. To this end, we start by introducing the concepts of (positive and negative) base type positions in a type taken from [2, Definition 7.2].

Definition 5. *The sets* $\mathcal{P}\mathsf{os}^+(T)$, $\mathcal{P}\mathsf{os}^-(T)$ *and* $\mathcal{P}\mathsf{os}_a(T)$ *of* positive base type positions, negative base type positions *and* positions of $a \in \mathcal{B}$ *in a type* $T \in \mathcal{T}$ *are inductively defined as sets of finite strings over* $\{1,2\}$ *(where* ϵ *denotes the empty string) as follows:*

$$\mathcal{P}\mathsf{os}^+(a) = \mathcal{P}\mathsf{os}_a(a) = \{\epsilon\} \qquad \mathcal{P}\mathsf{os}^-(a) = \varnothing \qquad \mathcal{P}\mathsf{os}_a(b) = \varnothing \text{ if } a \neq b$$
$$\mathcal{P}\mathsf{os}_a(U \to V) = \{1p \mid p \in \mathcal{P}\mathsf{os}_a(U)\} \cup \{2p \mid p \in \mathcal{P}\mathsf{os}_a(V)\}$$
$$\mathcal{P}\mathsf{os}^+(U \to V) = \{1p \mid p \in \mathcal{P}\mathsf{os}^-(U)\} \cup \{2p \mid p \in \mathcal{P}\mathsf{os}^+(V)\}$$
$$\mathcal{P}\mathsf{os}^-(U \to V) = \{1p \mid p \in \mathcal{P}\mathsf{os}^+(U)\} \cup \{2p \mid p \in \mathcal{P}\mathsf{os}^-(V)\}$$

Example 5. Let $\mathcal{B} = \{a, b\}$ and consider $T = (a \to b) \to (a \to b)$. We have $\mathcal{P}\mathsf{os}^+(T) = \{11, 22\}$ and $\mathcal{P}\mathsf{os}^-(T) = \{12, 21\}$. Furthermore, $\mathcal{P}\mathsf{os}_a(T) = \{11, 21\}$ and $\mathcal{P}\mathsf{os}_b(T) = \{12, 22\}$.

Next, we define the notions of accessible arguments of function symbols and basic base types as given in [2, Definitions 7.3 & 7.4].

Definition 6. *With every* $f \in \mathcal{F}_{T_1 \to \cdots \to T_n \to a}$ *we associate a set* $\mathsf{Acc}(f)$ *of* accessible *arguments of* f *such that* $i \in \mathsf{Acc}(f)$ *implies* $a \geqslant_{\mathcal{B}} T_i$ *and* $\mathcal{P}\mathsf{os}_a(T_i) \subseteq \mathcal{P}\mathsf{os}^+(T_i)$ *for all* $1 \leqslant i \leqslant n$. *Furthermore, we say that* $a \in \mathcal{B}$ *is* basic *if the following conditions hold:*

- $T \prec_{\mathcal{T}} a$ *implies that* T *is a basic base type,*
- *for all* $f \in \mathcal{F}_{T_1 \to \cdots \to T_n \to a}$ *and* $i \in \mathsf{Acc}(f)$, $T_i = a$ *or* T_i *is a basic base type.*

Note that the condition $T \prec_{\mathcal{T}} a$ is straightforward to check with the admissible type order from Lemma 2 as only base types can be smaller than base types. In general, it is possible to have base types which are bigger than function types while retaining admissibility of the type order [2].

Next, we define the notion of nonversatilely accessible subterms. The definition closely follows the one given in [2, Definition 7.5] but with appropriate restrictions regarding nonversatility.

Definition 7. *We write* $s \triangleright_{\mathsf{b}}^{\mathsf{nv}} t$ *if* t *is a subterm of* $s \in \Lambda_{\mathsf{nv}}$ *reachable by nonversatile intermediate terms in the recursive definition of* $\triangleright$, t *is of basic base type and* $\mathsf{FV}(t) \subseteq \mathsf{FV}(s)$, *i.e., no bound variables have become free. Furthermore,* $s \triangleright_{\mathsf{a}} t$ *if there are* $a \in \mathcal{B}$, $f \in \mathcal{F}_{T_1 \to \cdots \to T_n \to a}$, $s_i \in \Lambda_{T_i}$ *for* $1 \leqslant i \leqslant n$ *and* $j \in \mathsf{Acc}(f)$ *such that* $s = f(s_1, \ldots, s_{\mathsf{ar}(f)}) s_{\mathsf{ar}(f)+1} \cdots s_n$ *and* $s_j \trianglerighteq_{\mathsf{a}} t$. *Here,* $\triangleright_{\mathsf{b}}^{\mathsf{nv}}, \triangleright_{\mathsf{a}} \subseteq \mathsf{NF}(\beta\eta) \times \mathsf{NF}(\beta\eta)$. *A term* t *is* nonversatilely accessible *in a nonversatile term* s *if* $s \triangleright_{\mathsf{b}}^{\mathsf{nv}} t$ *or* $s \triangleright_{\mathsf{a}} t$.

Finally, we define the notion of structurally smaller terms with respect to a set of variables X. It is an important ingredient of CPO with accessible subterms as it facilitates a way of using the set X with which the order is parameterized in places where it would otherwise lead to non-termination. Once again, we can immediately use the corresponding definition given in [2, Definition 7.8].

Definition 8. *Let X be a finite set of variables. We say that a term t is* structurally smaller *than a term s, written $s \triangleright_{@}^{X} t$, if there are $a \in \mathcal{B}$, $x_1, \dots, x_k \in X$ and $u \in \Lambda$ such that $\tau(s) = \tau(t) = a$, $t = ux_1 \cdots x_k$, $s \triangleright_{\mathsf{a}} u$ and $\mathcal{P}\mathsf{os}_a(\tau(x_i)) = \varnothing$ for all $1 \leqslant i \leqslant k$. Here, $\triangleright_{@}^{X} \subseteq \mathsf{NF}(\beta\eta) \times \Lambda$.*

Note that if $s \triangleright_{@}^{X} t$ then t may not be in $\beta\eta$-normal form. The following important result is required for $\beta\eta$-normal stability and explains why nonversatility is needed in $\triangleright_{\mathsf{b}}^{\mathsf{nv}}$ but not in $\triangleright_{\mathsf{a}}$ and $\triangleright_{@}^{X}$ where it is guaranteed by the original definition.

Lemma 3. *The following statements hold:*

(i) If $s \triangleright_{\mathsf{b}}^{\mathsf{nv}} t$ then $s\sigma{\downarrow} \triangleright_{\mathsf{b}}^{\mathsf{nv}} t\sigma{\downarrow}$ for all substitutions σ.
(ii) If $s \triangleright_{\mathsf{a}} t$ then $s\sigma{\downarrow} \triangleright_{\mathsf{a}} t\sigma{\downarrow}$ for all substitutions σ.
(iii) If $s \triangleright_{@}^{X} t$ then $s\sigma{\downarrow} \triangleright_{@}^{X} t'$ for some $t' \to_{\beta\eta}^{!} t\sigma{\downarrow}$ whenever σ is away from X.
(iv) If $s \triangleright_{@}^{X} t$ and $t \in \Lambda_{\mathsf{nv}}$ then $s\sigma{\downarrow} \triangleright_{@}^{X} t\sigma{\downarrow}$ for all substitutions σ away from X.

4 The Beta-Eta-Normal Computability Path Order

First, we briefly recapitulate the definition of the multiset and lexicographic extension of orders. Given an order $>$, let $(s_1, \dots, s_n) >_{\mathsf{lex}} (t_1, \dots, t_m)$ if there exists an $i \leqslant \mathsf{min}(n, m)$ such that $s_i > t_i$ and $s_j = t_j$ for all $j < i$. Given two multisets M and N we write $M >_{\mathsf{mul}} N$ if $N = (M \setminus X) \uplus Y$ where $\varnothing \neq X \subseteq M$ and for all $y \in Y$ there exists an $x \in X$ such that $x > y$. It is well-known that these extensions preserve well-foundedness.

In the following, let $\succ_{\mathcal{T}}$ be an admissible order on types and $\succsim_{\mathcal{F}}$ a preorder on $\mathcal{F}$ called *precedence* with a well-founded strict part $\succ_{\mathcal{F}} = \succsim_{\mathcal{F}} \setminus \precsim_{\mathcal{F}}$ and the equivalence relation $\simeq_{\mathcal{F}} = \succsim_{\mathcal{F}} \cap \precsim_{\mathcal{F}}$. Furthermore, for every $f \in \mathcal{F}$ we fix a status $\mathsf{stat}(f) \in \{\mathsf{mul}, \mathsf{lex}\}$ such that symbols equivalent in $\simeq_{\mathcal{F}}$ have the same status. We also assume that $\mathcal{F}$ is partitioned into sets $\mathcal{F}_{\mathsf{b}}$ and $\mathcal{F}_{\mathsf{s}}$ of *big* and *small* symbols such that the following conditions hold: $\mathcal{F}_{\mathsf{b}} \cap \mathcal{F}_{\mathsf{s}} = \varnothing$, if $f \succsim_{\mathcal{F}} g$ and $g \in \mathcal{F}_{\mathsf{b}}$ then $f \in \mathcal{F}_{\mathsf{b}}$, and whenever $f \in \mathcal{F}_{T_1 \to \cdots T_n \to a} \cap \mathcal{F}_{\mathsf{s}}$ then

- $\mathsf{ar}(f) = n$ implies $a \trianglerighteqslant_{\mathcal{B}} T_i$ and $\mathsf{S}\mathcal{P}\mathsf{os}_a(T_i) = \varnothing$ for all $1 \leqslant i \leqslant n$ where $\mathsf{S}\mathcal{P}\mathsf{os}_a(\cdot)$ is defined as in [2, Definition 8.9],
- $\mathsf{ar}(f) < n$ implies $\mathsf{Acc}(f) = \varnothing$ as well as $a \trianglerighteqslant_{\mathcal{B}} T_i$ and $T_{\mathsf{ar}(f)+1} \to \cdots \to T_n \to a \trianglerighteqslant T_i$ for all $1 \leqslant i \leqslant \mathsf{ar}(f)$.

We are now able to lift CPO with accessible subterms and small symbols [2] to an appropriate component of a $\beta\eta$-normal higher-order reduction order.

$\langle \mathcal{F}_\mathsf{b} \triangleright \rangle$ $f(\bar{t}) >^X v$ if $f \in \mathcal{F}_\mathsf{b}$ and $t_i \trianglerighteq^{\mathsf{nv}}_\mathsf{b} \cdot \trianglerighteq_\mathsf{a} \cdot \geqslant_\tau v$ for some i

$\langle \mathcal{F}_\mathsf{b} = \rangle$ $f(\bar{t}) >^X g(\bar{u})$ if $f \in \mathcal{F}_\mathsf{b}$, $f \simeq_\mathcal{F} g$, $f(\bar{t}) >^X u_i$ for all i and $\bar{t}\ (>_\tau \cup \triangleright^X_@ \cdot \geqslant_\tau)_{\mathsf{stat}(f)}\ \bar{u}$

$\langle \mathcal{F}_\mathsf{b} \succ \rangle$ $f(\bar{t}) >^X g(\bar{u})$ if $f \in \mathcal{F}_\mathsf{b}$, $f \succ_\mathcal{F} g$ and $f(\bar{t}) >^X u_i$ for all i

$\langle \mathcal{F}_\mathsf{b} @ \rangle$ $f(\bar{t}) >^X uv$ if $f \in \mathcal{F}_\mathsf{b}$, $f(\bar{t}) >^X u$ and $f(\bar{t}) >^X v$

$\langle \mathcal{F}_\mathsf{b} \lambda \rangle$ $f(\bar{t}) >^X \lambda y.v$ if $f \in \mathcal{F}_\mathsf{b}$, $f(\bar{t}) >^{X \cup \{z\}} v[y/z]$, $\tau(y) = \tau(z)$ and z fresh

$\langle \mathcal{F}_\mathsf{b} \mathcal{V} \rangle$ $f(\bar{t}) >^X y$ if $f \in \mathcal{F}_\mathsf{b}$ and $y \in X$

$\langle @ \triangleright \rangle$ $tu >^X v$ if $t \geqslant^X v$ or $u \geqslant^X_\tau v$

$\langle @ = \rangle$ $tu >^X t'u'$ if $t = t'$ and $u >^X u'$ or $tu >^X_@ t'$ and $tu >^X_@ u'$ where $tu >^X_@ v$ if $t >^X_\tau v$ or $u \geqslant^X_\tau v$ or $tu >^X_\tau v$

$\langle @ \lambda \rangle$ $tu >^X \lambda y.v$ if $tu >^X v[y/z]$, $\tau(y) = \tau(z)$ and z fresh

$\langle @ \mathcal{F}_\mathsf{s} \rangle$ $tu >^X f(\bar{v})$ if $f \in \mathcal{F}_\mathsf{s}$ and $tu >^X_\tau v_i$ for all i

$\langle @ \mathcal{V} \rangle$ $tu >^X y$ if $y \in X$

$\langle \lambda \triangleright \rangle$ $\lambda x.t >^X v$ if $t[x/z] \geqslant^X_\tau v$, $\tau(x) = \tau(z)$ and z fresh

$\langle \lambda \triangleright \eta \rangle$ $\lambda x.t >^X v$ if $t[x/z] \geqslant^X_\tau vz$, $\tau(x) = \tau(z)$ and z fresh

$\langle \lambda = \rangle$ $\lambda x.t >^X \lambda y.v$ if $t[x/z] >^X v[y/z]$, $\tau(x) = \tau(y) = \tau(z)$ and z fresh

$\langle \lambda \neq \rangle$ $\lambda x.t >^X \lambda y.v$ if $\lambda x.t >^X v[y/z]$, $\tau(x) \neq \tau(y) = \tau(z)$ and z fresh

$\langle \lambda \mathcal{F}_\mathsf{s} \rangle$ $\lambda x.t >^X f(\bar{v})$ if $f \in \mathcal{F}_\mathsf{s}$ and $\lambda x.t >^X_\tau v_i$ for all i

$\langle \lambda \mathcal{V} \rangle$ $\lambda x.t >^X y$ if $y \in X$

$\langle \mathcal{F}_\mathsf{s} \triangleright \rangle$ $f(\bar{t}) >^X v$ if $f \in \mathcal{F}_\mathsf{s}$ and $t_i \geqslant_\tau v$ for some i

$\langle \mathcal{F}_\mathsf{s} = \rangle$ $f(\bar{t}) >^X g(\bar{u})$ if $f \in \mathcal{F}_\mathsf{s}$, $f \simeq_\mathcal{F} g$, $f(\bar{t}) >^X_\tau u_i$ for all i and $\bar{t}\ (>_\tau)_{\mathsf{stat}(f)}\ \bar{u}$

$\langle \mathcal{F}_\mathsf{s} \succ \rangle$ $f(\bar{t}) >^X g(\bar{u})$ if $f \in \mathcal{F}_\mathsf{s}$, $f \succ_\mathcal{F} g$ and $f(\bar{t}) >^X_\tau u_i$ for all i

$\langle \mathcal{F}_\mathsf{s} @ \rangle$ $f(\bar{t}) >^X uv$ if $f \in \mathcal{F}_\mathsf{s}$, $f(\bar{t}) >^X_\tau u$ and $f(\bar{t}) >^X_\tau v$

$\langle \mathcal{F}_\mathsf{s} \mathcal{V} \rangle$ $f(\bar{t}) >^X y$ if $f \in \mathcal{F}_\mathsf{s}$ and $y \in X$

Fig. 1. Rules of NCPO.

Definition 9. *Given a finite set X of variables, the order $>^X \subseteq \mathsf{NF}(\beta\eta) \times \mathsf{NF}(\beta\eta)$ is inductively defined in Figure 1 where we implicitly assume $s \in \Lambda_{\mathsf{nv}}$ whenever $s >^X t$. Furthermore, $s >^X_\tau t$ if $s >^X t$ and $\tau(s) \succeq_\mathcal{T} \tau(t)$, and $> (>_\tau)$ is a shorthand for $>^\varnothing (>^\varnothing_\tau)$.*

Note that $>^X$ is well-defined by induction on the tuple (s, t) with respect to the well-founded relation $(\triangleright, \triangleright)_{\mathsf{lex}}$. It is defined like CPO with accessible subterms and small symbols [2] but with the restriction to terms in $\beta\eta$-normal form where we additionally require that the first argument of $>^X$ as well as the first argument and all intermediate terms in the recursive definition of the subterm relation $\triangleright^{\mathsf{nv}}_\mathsf{b}$ are nonversatile. Furthermore, the rules $\langle @\beta \rangle$ and $\langle \lambda\eta \rangle$ which orient β- and η-reduction, respectively, have been removed. We also weakened the rule $\langle \mathcal{F}_\mathsf{s} = \rangle$ by disallowing $\triangleright^X_@ \cdot \geqslant_\tau$ in the comparison of the arguments as used in $\langle \mathcal{F}_\mathsf{b} = \rangle$. Finally, we added the rule $\langle \lambda \triangleright \eta \rangle$ which is inspired by rule (11) from NHORPO [11]. We refer to $>_\tau$ as the $\beta\eta$-normal computability path order (NCPO) and use the

symbol $\sqsupset$ with the same decorations as $>$ to denote CPO with accessible subterms and small symbols. We illustrate the definition on a number of examples and emphasize some differences between NCPO and NHORPO together with the transformation technique of neutralization. For our purposes, it is sufficient to know that the definition of NHORPO is similar to the one of NCPO without the extensions of accessible subterms and small symbols but with fewer rules, without a set X of variables and a weak decrease in the admissible type order in each recursive invocation. Furthermore, the transformation technique of neutralization maps terms to terms with the same termination behavior. The goal is to transform terms which are beyond the scope of NHORPO into terms which NHORPO can deal with. In particular, the transformation may apply a function symbol's arguments of nonbase type to special terms and apply β-reduction in order to minimize the number of lambda abstractions. A detailed definition can be found in [11].

Example 6. In this first example, we use the rule $\langle \lambda \triangleright \eta \rangle$ which is not part of CPO to prove termination of the symbolic differentiation example from [11]. Let $\mathcal{B} = \{\mathsf{r}\}$ and consider the function symbols $\mathsf{sin}, \mathsf{cos} \in \mathcal{F}_{\mathsf{r}\to\mathsf{r}}$, $\mathsf{diff} \in \mathcal{F}_{(\mathsf{r}\to\mathsf{r})\to\mathsf{r}\to\mathsf{r}}$ and $+, \times \in \mathcal{F}_{(\mathsf{r}\to\mathsf{r})\to(\mathsf{r}\to\mathsf{r})\to\mathsf{r}\to\mathsf{r}}$ as well as the variables $x \in \mathcal{V}_{\mathsf{r}}$ and $F, G \in \mathcal{F}_{\mathsf{r}\to\mathsf{r}}$. Furthermore, $\mathsf{ar}(\mathsf{sin}) = \mathsf{ar}(\mathsf{cos}) = \mathsf{ar}(\mathsf{diff}) = 1$ and $\mathsf{ar}(+) = \mathsf{ar}(\times) = 2$. We will use infix notation for $+$ and $\times$. The HRS $\mathcal{R}$ defining the symbolic differentiation of sin and $\times$ is defined by the following two rules:

$$\begin{aligned} \mathsf{diff}(\lambda x.\mathsf{sin}(Fx)) &\to (\lambda x.\mathsf{cos}(Fx)) \times \mathsf{diff}(F) \\ \mathsf{diff}(F \times G) &\to (\mathsf{diff}(F) \times G) + (F \times \mathsf{diff}(G)) \end{aligned}$$

For the termination proof with NCPO, all function symbols can be big with multiset status and we will use the precedence $\mathsf{diff} \succ_{\mathcal{F}} \mathsf{sin}, \mathsf{cos}, +, \times$. Note that all subterms of left-hand sides except for variables and the application Fx are nonversatile. Since $\mathcal{R}$ is an HRS, $\tau(\ell) = \tau(r)$ for all $\ell \to r \in \mathcal{R}$, so we only have to check $\ell > r$ for all $\ell \to r \in \mathcal{R}$. For the first rule, we apply $\langle \mathcal{F}_{\mathsf{b}} \succ \rangle$ to get the two proof obligations $\mathsf{diff}(\lambda x.\mathsf{sin}(Fx)) > \lambda x.\mathsf{cos}(Fx)$ and $\mathsf{diff}(\lambda x.\mathsf{sin}(Fx)) > \mathsf{diff}(F)$. For the former one, we proceed by $\langle \mathcal{F}_{\mathsf{b}} \lambda \rangle$, $\langle \mathcal{F}_{\mathsf{b}} \succ \rangle$ and $\langle \mathcal{F}_{\mathsf{b}} @ \rangle$ to obtain the subgoals $\mathsf{diff}(\lambda x.\mathsf{sin}(Fx)) >^{\{z\}} F$ and $\mathsf{diff}(\lambda x.\mathsf{sin}(Fx)) >^{\{z\}} z$. We can directly resolve the second subgoal with $\langle \mathcal{F}_{\mathsf{b}} \mathcal{V} \rangle$. For the first goal, applying $\langle \mathcal{F}_{\mathsf{b}} \triangleright \rangle$ and then $\langle \lambda \triangleright \eta \rangle$ yields the goal $\mathsf{sin}(Fy) > Fy$ which can be handled by $\langle \mathcal{F}_{\mathsf{b}} \triangleright \rangle$. For the remaining proof obligation $\mathsf{diff}(\lambda x.\mathsf{sin}(Fx)) > \mathsf{diff}(F)$ we apply $\langle \mathcal{F}_{\mathsf{b}} = \rangle$ and $\langle \lambda \triangleright \eta \rangle$ to obtain the goal $\mathsf{sin}(Fx) >_{\tau} Fx$ which is again handled by $\langle \mathcal{F}_{\mathsf{b}} \triangleright \rangle$.[3] Now consider the second rule. From two applications of $\langle \mathcal{F}_{\mathsf{b}} \succ \rangle$ we obtain the proof obligations $\mathsf{diff}(F \times G) > \mathsf{diff}(F)$, $\mathsf{diff}(F \times G) > G$, $\mathsf{diff}(F \times G) > F$ and $\mathsf{diff}(F \times G) > \mathsf{diff}(G)$ which can be resolved by two applications of $\langle \mathcal{F}_{\mathsf{b}} \triangleright \rangle$ or $\langle \mathcal{F}_{\mathsf{b}} = \rangle$ followed by one application of $\langle \mathcal{F}_{\mathsf{b}} \triangleright \rangle$, respectively.

[3] In [11], the same reasoning is used to handle this subcase but an application of the rule corresponding to $\langle \lambda \triangleright \eta \rangle$ is not allowed by their definition as F is a variable. Furthermore, as already mentioned, $\lambda x.\mathsf{sin}(Fx)$ is nonversatile by our sufficient condition but not by the one given in [11].

Note that the argumentation given for the last rule also works for NHORPO as can be confirmed by both our implementation of NHORPO as well as the prototype implementation linked from [11]. Hence, NHORPO does not need neutralization in order to prove termination of the preceding example even though this is claimed in [11]. However, as already mentioned, the original implementation of NHORPO cannot orient the first rule as the used sufficient condition for nonversatility is too weak. Interestingly, neutralization can make up for that, so the proof checked by the original implementation is correct but for different reasons than the ones given in [11]. The following example shows that NCPO can prove termination of systems where NHORPO succeeds only if neutralization is employed.

Example 7. In this example we use accessible subterms to prove termination of the computation of negation normal forms of formulas in first-order logic in the framework of $\beta\eta$-normal higher-order rewriting using NCPO. We need to distinguish between terms and formulas, so let the set of base types under consideration be $\mathcal{B} = \{t, f\}$. Furthermore, we consider the logical connectives represented by the function symbols $\neg \in \mathcal{F}_{f\to f}$, $\wedge, \vee \in \mathcal{F}_{f\to f\to f}$ and $\forall, \exists \in \mathcal{F}_{(t\to f)\to f}$ which are all considered to be big. We set $\mathsf{ar}(\neg) = \mathsf{ar}(\forall) = \mathsf{ar}(\exists) = 1$ and $\mathsf{ar}(\wedge) = \mathsf{ar}(\vee) = 2$ and allow us to use syntactic sugar in writing as few parentheses as possible by establishing that the unary function symbols bind stronger than the binary function symbols. Furthermore, we use infix notation for $\wedge$ and $\vee$. As variables, we will use $P, Q \in \mathcal{V}_f$ and $R \in \mathcal{V}_{t\to f}$. Hence, the HRS $\mathcal{R}$ for the computation of negational normal forms in first-order logic is given as the following set of rules:

$$\begin{array}{lll} \neg\neg P \to P & \neg(P \wedge Q) \to \neg P \vee \neg Q & \neg\forall R \to \exists(\lambda x.\neg(Rx)) \\ & \neg(P \vee Q) \to \neg P \wedge \neg Q & \neg\exists R \to \forall(\lambda x.\neg(Rx)) \end{array}$$

Note that all non-variable subterms of the left-hand sides are nonversatile. We choose $f \succ_{\mathcal{T}} t$ and let all function symbols have multiset status. Furthermore, set $\neg \succ_{\mathcal{F}} \wedge, \vee, \forall, \exists$. We have $\neg\neg P > P$ by applying $\langle \mathcal{F}_{\mathsf{b}} \triangleright \rangle$ twice. For $\neg(P \wedge Q) > \neg P \vee \neg Q$, applying $\langle \mathcal{F}_{\mathsf{b}} \succ \rangle$ yields the subgoals $\neg(P \wedge Q) > \neg P$ and $\neg(P \wedge Q) > \neg Q$ which are handled by $\langle \mathcal{F}_{\mathsf{b}} = \rangle$ since $P \wedge Q >_\tau P, Q$ by $\langle \mathcal{F}_{\mathsf{b}} \triangleright \rangle$. We obtain $\neg(P \vee Q) > \neg P \wedge \neg Q$ in the same way. This leaves us with establishing $\neg\forall R > \exists(\lambda x.\neg(Rx))$ as the final rule can again be oriented with the same strategy. By applying $\langle \mathcal{F}_{\mathsf{b}} \succ \rangle$, $\langle \mathcal{F}_{\mathsf{b}} \lambda \rangle$ and $\langle \mathcal{F}_{\mathsf{b}} = \rangle$ we obtain $\forall R >_\tau \cup \triangleright_{@}^{\{x\}} \cdot \geqslant_\tau Rx$ as proof obligation. Note that $\forall R >_\tau Rx$ is impossible as x does not occur in the left-hand side. However, we can establish $\forall R \triangleright_{@}^{\{x\}} Rx$ which is enough to fulfill the proof obligation. Since $f \geqslant_{\mathcal{B}} t \to f$ and $\{2\} = \mathcal{P}\mathsf{os}_f(t \to f) \subseteq \mathcal{P}\mathsf{os}^+(t \to f) = \{2\}$ we can set $1 \in \mathsf{Acc}(\forall)$. Hence, $\forall R \triangleright_{\mathsf{a}} R$. Furthermore, $\tau(\forall R) = \tau(Rx) = f$ and $\mathcal{P}\mathsf{os}_f(t) = \varnothing$, so $\forall R \triangleright_{@}^{\{x\}} Rx$ as desired.

Example 8. In this example, we demonstrate the usefulness of small function symbols by proving termination of the map function together with a function that increments lists of natural numbers in successor notation. This is a

slightly modified subsystem of `AotoYamada_05__014` from the termination problem database.[4] Proving termination of the full system for plain higher-order rewriting needs the transitive closure of CPO as shown in [2, Example 8.20], but this does not help in our setting as the "middle term" is not in $\beta\eta$-normal form.[5] Consider the set of base types $\mathcal{B} = \{\mathsf{a}, \mathsf{b}\}$ as well as the function symbols $\mathsf{0} \in \mathcal{F}_{\mathsf{b}}$, $\mathsf{nil} \in \mathcal{F}_{\mathsf{a}}$, $\mathsf{s} \in \mathcal{F}_{\mathsf{b}\to\mathsf{b}}$, $\mathsf{plus} \in \mathcal{F}_{\mathsf{b}\to\mathsf{b}\to\mathsf{b}}$, $\mathsf{inc} \in \mathcal{F}_{\mathsf{a}\to\mathsf{a}}$, $\mathsf{map} \in \mathcal{F}_{(\mathsf{b}\to\mathsf{b})\to\mathsf{a}\to\mathsf{a}}$ and $\mathsf{cons} \in \mathcal{F}_{\mathsf{b}\to\mathsf{a}\to\mathsf{a}}$. We set $\mathsf{ar}(\mathsf{0}) = \mathsf{ar}(\mathsf{nil}) = 0$, $\mathsf{ar}(\mathsf{s}) = \mathsf{ar}(\mathsf{plus}) = \mathsf{ar}(\mathsf{inc}) = 1$ and $\mathsf{ar}(\mathsf{map}) = \mathsf{ar}(\mathsf{cons}) = 2$. We will use the variables $x, y \in \mathcal{F}_{\mathsf{b}}$, $v \in \mathcal{F}_{\mathsf{a}}$ and $F \in \mathcal{V}_{\mathsf{b}\to\mathsf{b}}$. Consider the HRS $\mathcal{R}$ consisting of the following rules:

$$\begin{aligned}
\mathsf{plus}(\mathsf{0})\, x &\to x & \mathsf{plus}(\mathsf{s}(y))\, x &\to \mathsf{s}(\mathsf{plus}(y)\, x)\\
\mathsf{map}(F, \mathsf{nil}) &\to \mathsf{nil} & \mathsf{map}(F, \mathsf{cons}(x, v)) &\to \mathsf{cons}(Fx, \mathsf{map}(F, v))\\
\mathsf{inc}(v) &\to \mathsf{map}(\mathsf{plus}(\mathsf{s}(\mathsf{0})), v)
\end{aligned}$$

We choose $\mathsf{a} \succ_{\mathcal{T}} \mathsf{b}$, consider all function symbols except for s to be big and let them all have multiset status. The used precedence is $\mathsf{inc} \succ_{\mathcal{F}} \mathsf{map} \succ_{\mathcal{F}} \mathsf{cons}, \mathsf{nil}$ and $\mathsf{plus} \succ_{\mathcal{F}} \mathsf{s}$. Accessible arguments are not needed. Note that all non-variable subterms of the left-hand sides are nonversatile. The first rule follows directly from $\langle @\triangleright \rangle$. For the second rule, we apply $\langle @\mathcal{F}_{\mathsf{s}} \rangle$ to obtain the subgoal $\mathsf{plus}(\mathsf{s}(y))\, x >_\tau \mathsf{plus}(y)\, x$. We proceed by $\langle @= \rangle$ and choose the proof obligations $\mathsf{plus}(\mathsf{s}(y)) >_\tau \mathsf{plus}(y)$ as well as $x \geqslant_\tau x$. The second one is obviously true while the first rule can be resolved by $\langle \mathcal{F}_{\mathsf{b}}= \rangle$ and $\langle \mathcal{F}_{\mathsf{s}}\triangleright \rangle$. The third rule follows directly from $\langle \mathcal{F}_{\mathsf{b}}\triangleright \rangle$. For the fourth rule, we apply $\langle \mathcal{F}_{\mathsf{b}}\succ \rangle$ to obtain the subgoal $\mathsf{map}(F, \mathsf{cons}(x, v)) > Fx$ which follows from $\langle \mathcal{F}_{\mathsf{b}}@ \rangle$ and three applications of $\langle \mathcal{F}_{\mathsf{b}}\triangleright \rangle$ as well as the subgoal $\mathsf{map}(F, \mathsf{cons}(x, v)) > \mathsf{map}(F, v)$ which follows from $\langle \mathcal{F}_{\mathsf{b}}= \rangle$ and $\langle \mathcal{F}_{\mathsf{b}}\triangleright \rangle$. Finally, the fifth rule can be oriented by applying $\langle \mathcal{F}_{\mathsf{b}}\succ \rangle$ four times as well as $\langle \mathcal{F}_{\mathsf{b}}\triangleright \rangle$ once.

The previous example cannot be handled by NHORPO (with or without neutralization) as the rule for the recursive case of plus needs small symbols which are neither part of NHORPO nor added by neutralization.

5 Correctness Proof

We start by a technical result for CPO with accessible subterms which is needed for establishing $\beta\eta$-normal stability of $(>_\tau, \sqsupseteq_\tau)$. In particular, it states an important connection between $\triangleright^X_@$ and $\sqsupseteq^X$.

Lemma 4. *Let X be a finite set of variables and consider the term $f(t_1, \dots, t_n)$ with $f \in \mathcal{F}_{\mathsf{b}}$. If $t_i \triangleright^X_@ \cdot \sqsupseteq_\tau v$ for some $1 \leqslant i \leqslant n$ then $f(t_1, \dots, t_n) \sqsupseteq^X v$.*

Note that the previous lemma holds for both CPO with accessible subterms and NCPO. Hence, it allows us to do the usual optimizations in the implementation of the rule $\langle \mathcal{F}_{\mathsf{b}}= \rangle$ given in Figure 2. This can be justified in both cases

[4] https://github.com/TermCOMP/TPDB

[5] Actually, our modification also has to be applied to [2, Example 8.20] because admissible type orders as defined in [2] do not support equivalent base types.

$$\langle \mathcal{F}_{\mathsf{b}}{=}\mathsf{mul} \rangle \quad f(\bar{t}) >^X g(\bar{u}) \text{ if } f \simeq_{\mathcal{F}} g,\ \mathsf{stat}(f) = \mathsf{mul} \text{ and } \bar{t}\ (>_\tau \cup \rhd^X_{@} \cdot \geqslant_\tau)_{\mathsf{mul}}\ \bar{u}$$
$$\langle \mathcal{F}_{\mathsf{b}}{=}\mathsf{lex} \rangle \quad f(\bar{t}) >^X g(\bar{u}) \text{ if } f \simeq_{\mathcal{F}} g,\ \mathsf{stat}(f) = \mathsf{lex} \text{ and } \exists i.\, t_i >_\tau \cup \rhd^X_{@} \cdot \geqslant_\tau u_i \text{ and } \forall j < i.\, t_j = u_j \text{ and } \forall j > i.\, f(\bar{t}) >^X t_j$$

Fig. 2. Optimized Rules for the Implementation of $\langle \mathcal{F}_{\mathsf{b}}{=} \rangle$.

as follows: If $t_i \geqslant_\tau u_j$ then $f(\bar{t}) >^X u_j$ by $\langle \mathcal{F}_{\mathsf{b}}{\rhd} \rangle$ and if $t_i \rhd^X_{@} \cdot \geqslant_\tau u_j$ then $f(\bar{t}) >^X u_j$ by Lemma 4. The following lemma is the main result needed for establishing $\beta\eta$-normal stability of the pair $(>_\tau, \sqsupset_\tau)$.

Lemma 5. *Let $s >^X t$. For every $\beta\eta$-normal substitution σ away from X we have $s\sigma{\downarrow} \sqsupset^X t'$ for some $t\sigma \to^*_{\beta\eta} t' \to^!_{\beta\eta} t\sigma{\downarrow}$. Additionally, if $t = uv$ then $t' = u'v'$ with $u\sigma \to^*_{\beta\eta} u' \to^!_{\beta\eta} u\sigma{\downarrow}$ and $v\sigma \to^*_{\beta\eta} v' \to^!_{\beta\eta} v\sigma{\downarrow}$.*

Theorem 2. *The pair $(>_\tau, \sqsupset_\tau)$ is a $\beta\eta$-normal higher-order reduction order.*

Proof. Since $\sqsupset_\tau$ contains $\beta\eta$-reduction by definition, [2, Lemma 8.2] establishes its monotonicity and [2, Theorem 8.14] its well-foundedness, we are left to prove $\beta\eta$-normal stability. Let $s >_\tau t$ and σ be a substitution. By Lemma 5 we obtain $s\sigma{\downarrow} \sqsupset t'$ for some $t' \to^!_{\beta\eta} t\sigma{\downarrow}$. Since types are preserved under the application of substitutions as well as $\beta\eta$-conversions between well-typed terms, we obtain $s\sigma{\downarrow} \sqsupset_\tau t'$. Finally, we obtain $s\sigma{\downarrow} \sqsupset^+_\tau t\sigma{\downarrow}$ as $\beta\eta$-steps are included in $\sqsupset_\tau$. □

6 Implementation

A prototype implementation of NCPO is available at GitHub.[6] In contrast to the implementations of NHORPO with neutralization as well as CPO linked from [11] and [2], respectively, our implementation is not a mere termination checker which requires all parameters of the order as input but searches for suitable parameters using SAT/SMT which is standard practice for termination tools such as [6, 13]. Implementing the search for parameters also allows one to quickly find mistakes like in [2, Example 8.19] where it is claimed that small symbols are needed whereas this is not true. Running the prototype implementation linked from [2] with the parameters generated from our prototype implementation confirms our findings.

Our prototype is implemented in Haskell and uses the recent Hasmtlib package[7] to encode the termination problems into the SMT-LIB 2 format[8] and communicate with SMT solvers which have to be installed separately. We use the well-known TPTP THF format [24] as the input format of our prototype implementation. Our parser supports a suitable fragment of TPTP THF which consists of unit clauses where equality is the only allowed predicate. Variables in rules are handled by universal quantification. In particular, the chosen input

[6] https://github.com/niedjoh/hrsterm
[7] https://github.com/bruderj15/Hasmtlib
[8] https://smt-lib.org/

format does not support applied function symbols with fixed arities but solely relies on application. We think this is appropriate as in higher-order problems, fixed arities are a superfluous limitation in general and only needed as additional structure for termination techniques based on path orders. Hence, our implementation transforms the input into a representation using applied function symbols with fixed arities for the termination proof. The transformation chooses the maximal possible arity for each function symbol which depends on the minimal number of arguments which it is applied to in the problem. This gives us as much structure as possible without having to write the arities down explicitly.

The SMT encoding of NCPO contains more recursive cases and more global conditions than the one for NHORPO, but except for mild cases like $\triangleright^X_@$, terms only need to be decomposed. In particular, we can easily encode the search for precedences and admissible type orders of the class given in Lemma 2 by assigning an integer variable to each function symbol and base type. The remaining parameters regarding the status of each function symbol, which base types are basic, accessible arguments and small symbols can be encoded by boolean variables. Apart from some global conditions, the encoding then just needs to assert orientability of each rule in a given HRS by taking the disjunction of all applicable cases in the definition of NCPO and proceed recursively. Unlike stated in [11], this drastically changes for the transformation technique of normalization. In an efficient encoding, one would have to syntactically analyze $\beta\eta$-normal forms of transformed terms of the original system depending on parameters which are only given symbolically as the goal of the encoding is to search for suitable values for them. This seems to be impossible unless the concrete values of these parameters are hard-coded in a big disjunction which makes the encoding much more verbose than the usual encodings of precedences and the like. In our reimplementation of NHORPO and neutralization, we precomputed the terms resulting from the concrete parameters of neutralization in Haskell and used the SMT solver interactively to try to find a solution for each encoding in turn.

Apart from our reimplementation of NHORPO, we know of two other fully automated implementations of HORPO which are applicable to higher-order rewriting modulo $\beta\eta$, WANDA [13] and CSI^ho [19]. WANDA implements a variation of HORPO for the different rewrite formalism of AFSMs. AFSMs can model our $\beta\eta$-normal flavor of HRSs by replacing the free variables with meta-variables, their applications by meta-applications and employing a $\beta\eta$-first reduction strategy which rewrites a term to $\beta\eta$-normal form before performing a rewrite step. However, a formal proof of such a result is only available for a transformation of the subclass of pattern HRSs in $\beta\eta$-long normal form [12, Transformation 3.4] to AFSMs. Furthermore, WANDA only supports a β-first strategy, so in general the HORPO implementation in WANDA and our implementation of NHORPO/NCPO cannot be directly compared. CSI^ho is a confluence tool which implements a basic variant of HORPO which can be used for pattern HRSs [23] in $\beta\eta$-long normal form. Hence, it is also not possible to directly compare the HORPO implementation of CSI^ho with our prototype. Nevertheless,

Table 1. Experimental results.

problem	NCPO		NHORPO		NHORPO+neutralization	
Example 6 (extended)	✓	0.080 s	✓	0.022 s	✓	0.410 s
Example 7	✓	0.043 s	×	0.011 s	✓	2.286 s
Example 8	✓	0.020 s	×	0.010 s	×	0.322 s
[2, Example 5.2]	✓	0.015 s	×	0.009 s	✓	0.009 s
[2, Example 7.1]	✓	0.021 s	×	0.011 s	×	33.968 s
[2, Example 8.19]	✓	0.026 s	×	0.011 s	×	12.408 s
[11, Example 7.1]	✓	0.032 s	×	0.013 s	✓	0.498 s
[11, Example 7.2]	×	0.016 s	×	0.010 s	×	0.340 s
[11, Example 7.3]	✓	0.025 s	×	0.011 s	✓	0.166 s
`neutr.p`	×	0.012 s	×	0.008 s	✓	0.025 s
`neutrN.p`	✓	0.018 s	✓	0.010 s	✓	0.030 s

a comparison with both WANDA and CSIˆho is instructive as the investigated termination problems only differ in the treatment of η.

Table 1 compares our implementation of NCPO with our reimplementation of NHORPO with and without neutralization on a small set of problems. In all configurations, termination (✓) or the unsatisfiability of the encoding (×) are checked using the SMT solver Z3 [18]. Here, unsatisfiability of the encoding means that termination cannot be established with the given method, and does not imply nontermination. Furthermore, the table contains the execution time of each invocation on an Intel Core i7-7500U CPU running at a clock rate of 2.7 GHz with 15.5 GiB of main memory. As has already been pointed out in [11], NHORPO on its own is quite weak, so it is intended to be used together with neutralization. For four problems, neutralization on top of NHORPO is not enough: As already discussed, Example 8 needs small symbols and [2, Example 8.19] does not work because (N)HORPO enforces all subgoals to be weakly oriented in the admissible order on types while this is not the case for (N)CPO. Furthermore, [2, Example 7.1] shows that neutralization does not subsume accessible subterms. Finally, note that [11, Example 7.2] is not solvable by any of the three methods. In [11] it is claimed that NHORPO with neutralization is able to prove its termination but the proof given there does not work since it contains comparisons where the left-hand side is versatile. Indeed, our experiments confirm that there does not exist a suitable parameter assignment for neutralization which enables a termination proof with NHORPO. The existence of `neutr.p` shows that NCPO and NHORPO with neutralization have incomparable power. Note that NCPO succeeds in establishing termination of a neutralized version (`neutrN.p`) of this problem. With respect to execution time, it can be seen that a search for suitable parameters of NCPO does not take much more time than the corresponding search for NHORPO parameters. For neutralization, the number of parameters and therefore the execution time of our search method can explode quickly in the presence of function symbols with big arities which have more than one argument of nonbase type. However, there may be a more efficient encoding

of the search for the neutralization parameters than the one implemented by us. We also evaluated the HORPO implementations in WANDA and CSI^ho on the problems given in Table 1. The HORPO implementation in WANDA yields termination proofs for `neutr.p` and `neutrN.p` which shows that it is incomparable with NHORPO. The HORPO implementation in CSI^ho performs worse than NHORPO which is to be expected because it is based on a very basic version of HORPO. The examples as well as a script to reproduce our experiments are available on the GitHub repository of our prototype implementation.

7 Conclusion

In this paper, we lifted the computability path order with its extensions for accessible subterms and small symbols [2] from plain to $\beta\eta$-normal higher-order rewriting. In order to achieve that goal, we followed the approach from [11] based on $\beta\eta$-normal stability which allowed us to use monotonicity and well-foundedness of CPO instead of proving the corresponding properties once more for the new order. In addition, we gave an improved sufficient condition for terms to be nonversatile which is an important ingredient of the used approach. The resulting order, dubbed NCPO, can prove termination of systems which are out of reach for all other HORPO implementations targeting normal higher-order rewriting known to us. Moreover, NCPO and NHORPO with neutralization, its strongest competitor, are of incomparable power. Contrary to what is claimed in [11], we find it much more difficult to encode the search for suitable parameters of NHORPO with neutralization than for NCPO. Thus, we think that NCPO is a powerful and lightweight alternative to NHORPO with neutralization.

As far as future work is concerned, transitivity of NCPO is an open question. For NHORPO and CPO, the inclusion of β-reduction in the order justifies that it is not transitive, but this is not the case for NCPO. If $>_\tau$ were not transitive, then merely checking $\ell >_\tau r$ for each rule would not be a complete method for establishing termination with $(>_\tau, \sqsupset_\tau)$ using Theorem 1. Furthermore, the logical next step towards a powerful termination technique for normal higher-order rewriting would be the integration of NCPO into a dependency pair technique for normal higher-order rewriting as defined for example in [15]. Finally, given the complex nature and wide-ranging pitfalls inherent in higher-order termination as well as the mistakes found especially in [11], formalizations of higher-order termination methods are needed. To the best of our knowledge, a formalization of a basic variant of HORPO without computability closure [14] is the only existing work in that direction.

Acknowledgments. We thank the anonymous reviewers for their valuable comments which improved the presentation of the paper.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Barendregt, H.P.: Lambda calculi with types. In: Abramsky, S., Gabbay, D.M., Maibaum, T.S.E. (eds.) Handbook of Logic in Computer Science, vol. 2, pp. 117–309. Oxford University Press (1992). https://doi.org/10.1093/oso/9780198537618.003.0002
2. Blanqui, F., Jouannaud, J.P., Rubio, A.: The computability path ordering. Logical Methods in Computer Science **11**(4), 3:1–3:45 (2015). https://doi.org/10.2168/LMCS-11(4:3)2015
3. Blanqui, F., Jouannaud, J.P., Okada, M.: The calculus of algebraic constructions. In: Narendran, P., Rusinowitch, M. (eds.) Proc. 10th International Conference on Rewriting Techniques and Applications. Lecture Notes in Computer Science, vol. 1631, pp. 301–316 (1999). https://doi.org/10.1007/3-540-48685-2_25
4. Church, A.: A formulation of the simple theory of types. Journal of Symbolic Logic **5**(2), 56–68 (1940). https://doi.org/10.2307/2266170
5. Fuhs, C., Kop, C.: Polynomial interpretations for higher-order rewriting. In: Tiwari, A. (ed.) Proc. 23rd International Conference on Rewriting Techniques and Applications. Leibniz International Proceedings in Informatics, vol. 15, pp. 176–192 (2012). https://doi.org/10.4230/LIPIcs.RTA.2012.176
6. Giesl, J., Aschermann, C., Brockschmidt, M., Emmes, F., Frohn, F., Fuhs, C., Hensel, J., Otto, C., Plücker, M., Schneider-Kamp, P., Ströder, T., Swiderski, S., Thiemann, R.: Analyzing program termination and complexity automatically with AProVE. Journal of Automated Reasoning **58**, 3–31 (2017). https://doi.org/10.1007/s10817-016-9388-y
7. Hirokawa, N., Middeldorp, A., Sternagel, C., Winkler, S.: Abstract completion, formalized. Logical Methods in Computer Science **15**(3), 19:1–19:42 (2019). https://doi.org/10.23638/LMCS-15(3:19)2019
8. Jouannaud, J.P.: Higher-order rewriting: Framework, confluence and termination. In: Middeldorp, A., van Oostrom, V., van Raamsdonk, F., de Vrijer, R. (eds.) Processes, Terms and Cycles: Steps on the Road to Infinity: Essays Dedicated to Jan Willem Klop on the Occasion of His 60th Birthday, Lecture Notes in Computer Science, vol. 3838, pp. 224–250. Springer Berlin Heideberg (2005). https://doi.org/10.1007/11601548_14
9. Jouannaud, J.P., Li, J.: Church–Rosser properties of normal rewriting. In: Cégielski, P., Durand, A. (eds.) Proc. 22nd International Conference on Rewriting Techniques and Applications. Leibniz International Proceedings in Informatics, vol. 16, pp. 350–365 (2012). https://doi.org/10.4230/LIPIcs.CSL.2012.350
10. Jouannaud, J.P., Rubio, A.: Polymorphic higher-order recursive path orderings. Journal of the ACM **54**(1), 1–48 (2007). https://doi.org/10.1145/1206035.1206037
11. Jouannaud, J.P., Rubio, A.: Normal higher-order termination. ACM Transactions on Computational Logic **16**(2), 13:1–13:38 (2015). https://doi.org/10.1145/2699913
12. Kop, C.: Higher Order Termination. Ph.D. thesis, Vrije Universiteit Amsterdam (2012)
13. Kop, C.: WANDA - a higher order termination tool (system description). In: Ariola, Z.M. (ed.) Proc. 5th International Conference on Formal Structures for Computation and Deduction. Leibniz International Proceedings in Informatics, vol. 167, pp. 36:1–36:19 (2020). https://doi.org/10.4230/LIPIcs.FSCD.2020.36
14. Koprowski, A.: Coq formalization of the higher-order recursive path ordering. Applicable Algebra in Engineering, Communication and Computing **20**(5), 379–425 (2009). https://doi.org/10.1007/s00200-009-0105-5

15. Kusakari, K., Isogai, Y., Sakai, M., Blanqui, F.: Static dependency pair method based on strong computability for higher-order rewrite systems. IEICE Transactions on Information and Systems **E92.D**(10), 2007–2015 (2009). https://doi.org/10.1587/transinf.E92.D.2007
16. Mayr, R., Nipkow, T.: Higher-order rewrite systems and their confluence. Theoretical Computer Science **192**(1), 3–29 (1998). https://doi.org/10.1016/S0304-3975(97)00143-6
17. Miller, D.: A logic programming language with lambda-abstraction, function variables, and simple unification. Journal of Logic and Computation **1**(4), 497–536 (1991). https://doi.org/10.1093/logcom/1.4.497
18. de Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) Proc. 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Lecture Notes in Computer Science, vol. 4963, pp. 337–340 (2008). https://doi.org/10.1007/978-3-540-78800-3_24
19. Nagele, J.: Mechanizing Confluence: Automated and Certified Analysis of First- and Higher-Order Rewrite Systems. Ph.D. thesis, University of Innsbruck (2017)
20. Niederhauser, J., Middeldorp, A.: The computability path order for beta-eta-normal higher-order-rewriting (full version). CoRR **abs/2505.20121** (2025). https://doi.org/10.48550/arXiv.2505.20121
21. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL — A Proof Assistant for Higher-Order Logic, Lecture Notes in Computer Science, vol. 2283. Springer Berlin Heideberg (2002). https://doi.org/10.1007/3-540-45949-9
22. van de Pol, J.: Termination of Higher-Order Rewrite Systems. Ph.D. thesis, Utrecht University (1996)
23. van Raamsdonk, F.: On termination of higher-order rewriting. In: Middeldorp, A. (ed.) Proc. 12th International Conference on Rewriting Techniques and Applications. Lecture Notes in Computer Science, vol. 2051, pp. 261–275 (2001). https://doi.org/10.1007/3-540-45127-7_20
24. Sutcliffe, G.: The TPTP problem library and associated infrastructure. Journal of Automated Reasoning **59**, 483–502 (2017). https://doi.org/10.1007/s10817-017-9407-7
25. Zantema, H.: Termination of term rewriting: Interpretation and type elimination. Journal of Symbolic Computation **17**(1), 23–50 (1994). https://doi.org/10.1006/jsco.1994.1003