Incremental Checking of Well-Founded Recursive Specifications Modulo Axioms*

Felix Schernhammer[1] and José Meseguer[2]

[1] Vienna University of Technology, Austria, felixs@logic.at
[2] University of Illinois at Urbana-Champaign, USA, meseguer@uiuc.com

Abstract. We introduce the notion of well-founded recursive order-sorted equational logic (OS) theories modulo axioms. Such theories define functions by well-founded recursion and are inherently terminating. Moreover, for well-founded recursive theories important properties such as confluence and sufficient completeness are modular for so-called fair extensions. This enables us to incrementally check these properties for hierarchies of such theories that occur naturally in modular rule-based functional programs. Well-founded recursive OS theories modulo axioms contain only commutativity and associativity-commutativity axioms. In order to support arbitrary combinations of associativity, commutativity and identity axioms, we show how to eliminate identity and (under certain conditions) associativity (without commutativity) axioms by theory transformations in the last part of the paper.

1 Introduction

Scalability is a big, unsolved challenge in formal reasoning about executable algebraic specifications. When using such specifications as programs and reasoning about their correctness, we often need to check basic properties such as confluence, termination, and sufficient completeness. This is quite manageable for small specifications, but when dealing with larger specifications corresponding to realistic programs, we can encounter severe tool performance barriers. For example, a non-built-in specification in Maude of the natural numbers, which is the exact counterpart of Maude's built-in `NAT` module, cannot be proved terminating by the MTT tool, which performs a relatively simple transformation to make the order-sorted specification unsorted and then invokes the AProVE tool with a 900 second timeout. Likewise, Mu-term cannot prove the same specification terminating with the same timeout, even though both AProVE and Mu-Term are state-of-the-art tools. In a similar way, particularly in the presence of AC axioms, a large number of critical pairs is often generated when checking the local confluence of specifications. For example, a small AC specification of hereditarily finite sets with only 26 equations already generates 1027 critical pairs when using the Maude Church-Rosser Checker [10]. Modularity is crucial.

Modular methods for termination and confluence (for a good survey up to 2002 see [24]) are certainly helpful. However: (i) some of these methods make quite strong requirements (e.g., disjointness) on the kind of modularity they allow; (ii) little seems to be known about the modularity of sufficient completeness; and (iii) the modularity results we are aware of do not deal with sorts and subsorts, nor (except for, e.g., [23, 19]) with rewriting modulo axioms, which are key features of state-of-the-art rule-based languages such as ASF+SDF [30], ELAN [5], CafeOBJ [11], and Maude [6].

Our Approach is based on the observation that in practice algebraic specifications are often *recursive function definitions* based on *constructor patterns*, and whose right-hand sides involve *recursive calls* to the same and/or previously defined functions on *smaller arguments* in the *well-founded* subterm ordering. This includes, but goes beyond, the very common case of primitive-recursive definitions. For example, the equations defining Ackerman's function,

$$A(0,n) = s(n) \qquad A(s(m),0) = A(m,1)$$
$$A(s(m),s(n)) = A(m,A(s(m),n))$$

* The first author has been supported by the Austrian Academy of Sciences under grant number 22.361 and by the Austrian Marshall Plan Foundation in the Marshall Plan Scholarship Program under grant number 154.265.20.3.2010. The second author has been supported by NSF Grants CNS 07-16638 and 09-04749, and CCF 09-05584.

exemplify a well-founded recursive function definition based on natural number constructor patterns which is not primitive-recursive. Such specifications define *total* (i.e., terminating) functions on the set of constructor terms. Furthermore, they naturally form *hierarchies*, so that previously-defined functions can be used to define more complex ones. For example, natural number exponentiation can be recursively defined in terms of multiplication, which can in turn be recursively defined in terms of addition.

The main goal of this work is to reduce the checking of *confluence, termination, and sufficient completeness* for algebraic specifications based on well-founded recursive function definitions to relatively simple *incremental checks* on the module hierarchies containing such definitions. However, in order to be practically useful for rule-based languages, the notion of well-founded recursive function definition *needs to be generalized* to support: (a) mutually recursive definitions; (b) sorts, subsorts, and subsort overloading of function symbols; and (c) rewriting modulo axioms such as associativity and/or commutativity and/or identity. Such a generalization is non-trivial. Support for (a) is the least problematic, but support for (b) means that, because of subsort overloading, a function f can never be considered to be defined *once and for all:* it can always be *extended* to bigger sorts. For example, we can first define a + function in a `NAT` module, and then extend its domain of definition in `INT`, `RAT`, and `COMPLEX` modules. Support for (c) is the least obvious, because the notion of "well-founded recursive function definition" does not have a straightforward extension to the modulo case. For example, if f is an associative-commutative (AC) function symbol, a definition of f based on a binary constructor g and a constructor constant a might include an equation $f(g(x,y),a) = g(f(x,y),g(a,a))$, which syntactically satisfies all the expected requirements of well-founded recursive function definitions, yet is non-AC-terminating (cf. Example 5 in Section 3). A related difficulty for axioms like AC is that the usual *syntactic* characterizations of classes of recursive functions (e.g., primitive recursive) are no longer adequate, because of the much greater flexibility in the constructor patterns that can be used. For example, the definition of the cardinality function `card` in the `MSET-NAT` module below could just as well have used an equation `card(MS,MS')= card(MS) + card(MS')` with `MS, MS'` of sort `MSet`, instead of the equation `card(N,MS)= s(0) + card(MS)` with `N` of sort `Nat` below. This work provides a notion of well-founded recursive function definition supporting features (a)–(b)–(c). We show in Section 3.1 that our approach generalizes an already very general notion of many-sorted well-founded recursive function.

To make the approach scalable, the cost of each incremental check should be small. This can be achieved by taking advantage of modular methodologies which ensure that in an immediate submodule inclusion $(\Sigma, E \cup Ax) \subset (\Sigma \cup \Sigma_\Delta, (E \cup E_\Delta) \cup (Ax \cup Ax_\Delta))$, while both modules *can be arbitrarily large*, the *incremental additions* Σ_Δ to the signature, E_Δ to the defining equations, and Ax_Δ to the axioms, are *small*. Such increments being big is a clear sign of bad software engineering practice, since usually a more modular design can be achieved by module refactoring. The incremental proof methods we propose are scalable precisely because they are based on checking the typically small increments $(\Sigma_\Delta, E_\Delta \cup Ax_\Delta)$ and not the, potentially very large, theory $(\Sigma \cup \Sigma_\Delta, (E \cup E_\Delta) \cup (Ax \cup Ax_\Delta))$.

A Running Example. Throughout the paper we use the following running example in Maude. Although small, it illustrates all the key features supported: mutual recursion, order-sortedness, and rewriting modulo axioms.

```
fmod NATURAL is pr TRUTH-VALUE .
 sort Nat .
 op 0 : -> Nat [ctor] .
 op s : Nat -> Nat [ctor] .
 op _+_ : Nat Nat -> Nat [comm id: 0] .
 ops even odd : Nat -> Bool .
 vars N M : Nat .
 eq s(N) + s(M) = s(s(N + M)) .
 eq even(0) = true .
 eq even(s(N)) = odd(N) .
 eq odd(0) = false .
 eq odd(s(N)) = even(N) .
endfm
```

```
fmod MSET-NAT is pr NATURAL .
 sort MSet .
 subsort Nat < MSet .
 op _,_ : MSet MSet -> MSet [ctor assoc comm id: null] .
 op null : -> MSet [ctor] .
 op card : MSet -> Nat .
 var MS : MSet .
 var N : Nat .
 eq card(null) = 0 .
 eq card(N,MS)= s(0) + card(MS) .
endfm

fmod LIST-MSET-NAT is pr MSET-NAT .
 sorts List NeList .
 subsorts MSet < NeList < List .
 op nil : -> List [ctor] .
 op U : List -> MSet .
 op _;_ : List List -> List [assoc id: nil] .
 op _;_ : MSet NeList -> NeList [ctor assoc id: nil] .
 var MS : MSet .
 var NL : NeList .
 eq U(nil) = null .
 eq U(MS) = MS .
 eq U(MS ; NL) = MS, U(NL) .
endfm
```

The `NATURAL` module defines the natural numbers with addition and with `even` and `odd` predicates. The `MSET-NAT` module defines multisets of naturals and cardinality of multisets. Finally, the `LIST-MSET-NAT` module forms lists of multisets of numbers and defines a multiset union operator on such lists. Associativity, commutativity and identity axioms are specified with the `assoc`, `comm`, and `id:` attributes. All constructor operators are declared with the `ctor` keyword. As illustrated for `_;_`, an operator can be a constructor for some typing (`NeList`) and a defined symbol for a looser typing: `(0,0) ; nil` and `nil ; nil` are *not* constructor terms.

The paper is organized as follows. Section 2 gives background on order-sorted rewriting. Section 3 introduces well-founded recursive theories. Section 4 describes and justifies the incremental checking methods modulo C and AC axioms. Section 5 extends the approach to other combinations of A, C and I (identity) axioms. Finally, Section 6 discusses related work and presents some conclusions. The proofs of all theorems can be found in the appendix.

2 Background on Order-sorted Term Rewriting

We summarize here material from [14, 21] on order-sorted algebra and order-sorted rewriting. For standard notions and notations of ordinary term rewriting we refer to [3, 4]. We start with a partially ordered set $(S, \leq)$ of *sorts*, where $s \leq s'$ is interpreted as *subsort inclusion*. The *connected components* of $(S, \leq)$ are the equivalence classes $[s]$ corresponding to the least equivalence relation $\equiv_{\leq}$ containing $\leq$. When a connected component $[s]$ has a top element, we will also denote by $[s]$ such a top element. An order-sorted signature $\Sigma = (S, \leq, F)$ consists of a poset of sorts $(S, \leq)$ and a $S^* \times S$-indexed family of sets $F = \{F_{w,s}\}_{(w,s) \in S^* \times S}$, which are *function symbols* with given string of argument sorts and result sort. If $f \in F_{s_1 \ldots s_n, s}$, we declare the function symbol f as $f : s_1 \ldots s_n \longrightarrow s$. Some of these symbols f can be *subsort-overloaded*, i.e., they can have several declarations related in the $\leq$ ordering [14].

Given an S-sorted set $\mathcal{X} = \{\mathcal{X}_s \mid s \in S\}$ of *disjoint* sets of variables and an order-sorted (OS) signature $\Sigma = (S, \leq, F)$, the set $\mathcal{T}(\Sigma, \mathcal{X})_s$ of terms of sort s is the least set such that $\mathcal{X}_s \subseteq \mathcal{T}(\Sigma, \mathcal{X})_s$; if $s' \leq s$, then $\mathcal{T}(\Sigma, \mathcal{X})_{s'} \subseteq \mathcal{T}(\Sigma, \mathcal{X})_s$; and if $f : s_1 \ldots s_n \longrightarrow s$ is a declaration for symbol f and $t_i \in \mathcal{T}(\Sigma, \mathcal{X})_{s_i}$ for $1 \leq i \leq n$, then $f(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma, \mathcal{X})_s$. The set $\mathcal{T}(\Sigma, \mathcal{X})$ of order-sorted terms is $\mathcal{T}(\Sigma, \mathcal{X}) = \cup_{s \in S} \mathcal{T}(\Sigma, \mathcal{X})_s$. An element of any set $\mathcal{T}(\Sigma, \mathcal{X})_s$ is called a *well-formed* term. A simple syntactic condition on Σ called *preregularity* [14] ensures that each well-formed term t has always a *least-sort* possible among all sorts in S, which is denoted $ls(t)$.

Furthermore, Σ is *monotonic* if for every two declarations $f : s_1 \dots s_n \longrightarrow s$ and $f : s'_1 \dots s'_n \longrightarrow s'$, $s_1 \dots s_n \geqslant s'_1 \dots s'_n$ implies $s > s'$, where $s_1, \dots, s_n \geqslant s'_1, \dots, s'_n$ means $s_i \geq s'_i$ for all $1 \leq i \leq n$ and $s_i > s'_i$ for some $1 \leq i \leq n$. Throughout this paper we assume that all order-sorted signatures are preregular and monotonic. Terms are viewed as labeled trees in the usual way. Positions $p, q, \dots$ are represented by chains of positive natural numbers used to address subterm positions of t. The set of positions of a term t is denoted $\mathcal{P}os(t)$. Positions of non-variable symbols in t are denoted as $\mathcal{P}os_\Sigma(t)$, and $\mathcal{P}os_\mathcal{X}(t)$ are the positions of variables. The subterm at position p of t is denoted as $t|_p$ and $t[u]_p$ is the term t with the subterm at position p replaced by u. We write $t \trianglerighteq u$, read u *is a subterm of* t, if $u = t|_p$ for some $p \in \mathcal{P}os(t)$ and $t \triangleright u$ if $t \trianglerighteq u$ and $t \neq u$.

An order-sorted substitution σ is an S-sorted mapping $\sigma = \{\sigma : \mathcal{X}_s \to \mathcal{T}(\Sigma, \mathcal{X})_s\}_{s \in S}$ from variables to terms. A *specialization* ν is an OS-substitution that maps a variable x of sort s to a variable x' of sort $s' \leq s$. We denote $\mathcal{D}om(\sigma)$ and $\mathcal{R}ng(\sigma)$ the domain and range of a substitution σ. An (order-sorted) rewrite rule is an ordered pair (l, r), written $l \to r$, with $l, r \in \mathcal{T}(\Sigma, \mathcal{X})$, $l \notin \mathcal{X}$, $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$ (and $ls(l) \equiv_\leq ls(r)$ for order-sorted rules). If for all specializations ν, $ls(\nu(l)) \geq ls(\nu(r))$, then we say that the OS-rule $l \to r$ is *sort-decreasing*. A term $t \in \mathcal{T}(\Sigma, \mathcal{X})$ rewrites to u (at position $p \in \mathcal{P}os(t)$ and using the rule $l \to r$), written $t \xrightarrow{p}_{l \to r} s$ (or just $t \to_\mathcal{R} s$ or even $t \to s$ if no confusion arises), if $t|_p = \sigma(l)$ and $s = t[\sigma(r)]_p$, for some OS-substitution σ; if $l \to r$ is *not* sort-decreasing, we also require that $t[\sigma(r)]_p$ is a well-formed term.

An *order-sorted theory* (OS theory) is a triple $\mathcal{E} = (\Sigma, B, R)$ with Σ a preregular order-sorted signature such that each connected component has a top sort, B a set of unconditional Σ-equations, and R a set of unconditional Σ-rules. In this paper B will always be a combination of associativity and/or commutativity and/or identity axioms for some of the operators in Σ. Moreover, associative and commutative operators f are always typed $f\colon s\, s' \longrightarrow s$ for some sorts s, s' where $s' \leq s$. By Σ_{AC} (resp. Σ_C) we denote the subsignature of Σ where all function symbols are associative and commutative but do not have an identity (resp. where all function symbols are commutative but not associative and do not have an identity). Furthermore, we assume[3] that in each equation $u = v$ in B the variables $\{x_1, \dots, x_n\} = \mathcal{V}ar(u) = \mathcal{V}ar(v)$ have *top sorts* $[s_1]$, $\dots$, $[s_n]$.

Given an OS theory $\mathcal{E}$ as above, $t \to_{R/B} t'$ iff there exist u, v such that $t =_B u$ and $u \to_R v$ and $v =_B t'$. We say that (Σ, B, R) is *B-confluent*, resp. *B-terminating*, if the relation $\to_{R/B}$ is confluent, resp. terminating. By $[w]_B$ we denote that equivalence class of terms that are B-equal to w. We call an order-sorted signature *B-preregular* if the set of sorts $\{s \in S \mid \exists w' \in [w]_B \text{ s.t. } w' \in \mathcal{T}(\Sigma, \mathcal{X})_s\}$ has a least upper bound, denoted $ls[w]_B$ which can be effectively computed.[4] If (Σ, B, R) is B-confluent, B-terminating, B-preregular, and sort-decreasing, then the initial algebra $\mathcal{T}_{\Sigma/R \cup B}$, where the rules R are interpreted as equations, is isomorphic to the canonical term algebra $\mathcal{C}_{\Sigma/R,B}$, whose elements are B-equivalence classes in $\to_{R/B}$-canonical form. An order-sorted subsignature $\Omega \subseteq \Sigma$ with the same poset of sorts as Σ is called a *constructor subsignature* iff for each ground Σ-term t there is a ground Ω-term u such that $t \to^*_{R/B} u$. Terms from $\mathcal{T}(\Omega, \mathcal{X})$ are called *Ω-constructor terms*, or just constructor terms if Ω is clear from the context. We then say that (Σ, B, R) is *sufficiently complete* with respect to Ω. Intuitively this means that the functions defined by the rules R have been fully defined. For instance, the operators declared with the `ctor` attribute in our running example define a constructor subsignature, so that the specification is sufficiently complete. Assuming that (Σ, B, R) is B-confluent, B-terminating, B-preregular, and sort-decreasing, and that Ω is a constructor subsignature, if for any $t \to t'$ in R, whenever t is an Ω-term then t' is also an Ω-term, we are then guaranteed that all the elements in the canonical term algebra $\mathcal{C}_{\Sigma/R,B}$ are B-equivalence classes of ground Ω-terms. If, in addition, any ground Ω-term t is in $\to_{R/B}$-canonical form, then we call Ω a signature of *free constructors modulo B*.

Given an OS theory $\mathcal{E} = (\Sigma, B, R)$, we call an unsorted theory $\mathcal{E}' = (\Sigma', B', R')$ a *sound reflection* of $\mathcal{E}$ if there exists a mapping $\mathcal{M}$ from $\mathcal{T}(\Sigma, \mathcal{X})$ to a set of unsorted terms such that $t \to_{R/B} t' \Rightarrow \mathcal{M}(t) \to^+_{R'/B'} \mathcal{M}(t')$ for all terms $t, t' \in \mathcal{T}(\Sigma, \mathcal{X})$ (in that case we say that $\mathcal{E}'$ is a

[3] This assumption makes the technical treatment simpler, but it does not involve loss of generality and does not have to be specified explicitly: we can always assume that a new top sort $[s]$ is added to each connected component so that a binary operator f to which some axiom in B apply is overloaded for the top sort as $f : [s]\ [s] \longrightarrow [s]$. Maude adds these "kind" sorts $[s]$ automatically to any specification.

[4] Maude automatically checks the B-preregularity of a signature Σ for any combination of associativity, commutativity and identity axioms (see [6, Chapter 22.2.5]).

sound reflection of $\mathcal{E}$ w.r.t. $\mathcal{M}$; cf. [25]). Given a strict order $\succ$ on some domain D, the lexicographic extension of $\succ$ to n-tuples over D is defined as $\langle d_1, \ldots, d_n \rangle \succ^{lex} \langle d'_1, \ldots, d'_n \rangle$ if there exists an $i \leq n$ such that $d_j = d'_j$ for all $j < i$ and $d_i \succ d'_i$. Moreover, the multiset extension of $\succ$ to multisets over D is defined by $D_1 \succ^{mul} D_2$ if $D_1 \neq D_2$ and for each $d \in D_2 \setminus D_1$ there exists a $d' \in D_1 \setminus D_2$ such that $d' \succ d$.

3 Well-founded Recursive Theories

In this section we introduce the notion of well-founded recursive OS theories modulo axioms. The basic idea is to impose conditions on the equations of such theories, that guarantee finiteness of rewrite derivations. These conditions are based on the notion of *recursive dependency* of function symbols. Intuitively, a function symbol f recursively depends on g if there is a rule $f(t_1, \ldots, t_n) \to r$ in the OS theory with $root(r|_p) = g$ for some position p of r.

Definition 1 (recursive dependency). *Assume the axioms B_0 of the theory $\mathcal{E} = (\Sigma, B_0, R)$ are only commutativity and associativity-commutativity axioms.*[5]

Let G be the names of function symbols in Σ. The relation $\blacktriangleright^1_{\mathcal{E}} \subseteq G \times G$ is defined as $f \blacktriangleright^1_{\mathcal{E}} g$ whenever, there is a rule $l \to r \in R$ and a position $p \in Pos(r)$ such that $root(l) = f$ and $root(r|_p) = g$. The preorder $\blacktriangleright_{\mathcal{E}} \subseteq G \times G$ is obtained as the reflexive and transitive closure of $\blacktriangleright^1_{\mathcal{E}}$.

For order-sorted theories, and in particular in the presence of subsort overloaded function symbols, it is advantageous to distinguish between subsort overloaded variants of function symbols, because by doing so one obtains a more fine-grained notion of recursive dependency. This more fine-grained notion is needed because recursive dependencies exclusively based on the *names* of overloaded function symbols are too coarse to faithfully capture the actual dependencies involved in order-sorted rewriting.

A straightforward approach to achieve this disambiguation of subsort overloaded function symbols is to label them with the sorts of their arguments. This approach was used for instance by Ölveczky et al. ([25][Definitions 2 and 3]) to obtain an unsorted *reflection* of order-sorted rewrite systems. Unfortunately, in the presence of associativity axioms the unsorted rewrite system obtained by this labeling may not reflect the original order-sorted one.

Example 1. Consider an OS theory $\mathcal{E}$ with sorts $A < B$, a function symbol f which is subsort overloaded with typings $f\colon A\,A \to A$, and $f\colon B\,B \to B$, a unary function symbol s with typing $s\colon B \to A$, and a constant b of sort B. The symbol f is associative and commutative and the rules are

$$\begin{aligned} f(b, x) &\to f(s(b), s(b)) \\ f(b, s(x)) &\to f(b, x) \end{aligned}$$

where the sort of the variable x is B. By labeling the function symbols according to the sorts of their arguments in the corresponding order-sorted rewrite system, one does not obtain a sound reflection. The labeled versions of the above rules (including specializations) are

$$\begin{aligned} f_{B,B}(b, x) &\to f_{A,A}(s_B(b), s_B(b)) \\ f_{B,A}(b, x) &\to f_{A,A}(s_B(b), s_B(b)) \\ f_{B,A}(b, s_B(x)) &\to f_{B,B}(b, x) \\ f_{B,A}(b, s_A(x)) &\to f_{B,A}(b, x) \end{aligned}$$

In [25], additionally rules that decrease the sort labelings of function symbols are needed. Here, these rules are

$$\begin{aligned} f_{B,B}(x, y) &\to f_{A,B}(x, y) \\ f_{B,B}(x, y) &\to f_{B,A}(x, y) \\ f_{A,B}(x, y) &\to f_{A,A}(x, y) \\ f_{B,A}(x, y) &\to f_{A,A}(x, y) \\ s_B(x) &\to s_A(x) \end{aligned}$$

[5] The subscript 0 in B_0 indicates that only C and AC axioms are present. In Section 5 below also other combinations of A, C and I axioms will be considered.

The symbols $f_{B,B}, f_{B,A}$ and $f_{A,A}$ are considered to be associative and commutative. The following cyclic reduction sequence cannot be reflected.

$$\begin{aligned} f(f(b,s(b)),b) &\to f(f(s(b),s(b)),b) =_{AC} \\ &=_{AC} f(f(b,s(b)),s(b)) \to \\ &\to f(f(b,b),s(b)) =_{AC} f(f(b,s(b)),b) \end{aligned}$$

In fact the labeled rewrite system is terminating (which can automatically be proved by AProVE [13]).

The reason for the inability of the labeled rewrite system to simulate correctly the order-sorted rewriting of Example 1 is the complex interaction between sorts and structural axioms. More precisely, in the term $f(f(s(b),s(b)),b)$ the sort of the arguments of the inner f symbol is A. However, in the AC-equal term $f(f(b,s(b)),s(b))$ the two arguments of the inner f symbol have sorts B and A. Hence, there is an increase in the sorts of the arguments caused by the associativity axiom. Note that in the labeled version of the term $f(f(s(b),s(b)),b)$ which is $f_{A,B}(f_{A,A}(s_B(b_B),s_B(b_B)),b_B)$ no associativity equation is applicable since $f_{A,B} \neq f_{A,A}$.

The problem, therefore, is to find a C- and AC-compatible disambiguation scheme on which we can express order-sorted recursive dependencies. The solution to this problem is to label AC function symbols not by pairs of sorts, but by the multisets of sorts of arguments of the flattened versions of the terms in question. Commutative (but not associative) function symbols are labeled by unordered pairs of sorts of their arguments.

Definition 2 ((top) flattening). *Let Σ be an unsorted signature containing free and AC-function symbols and let f be an AC symbol. Then,*

$$flat(t,f) = \begin{cases} x & \text{if } t = x, \text{ a variable} \\ g(t_1,\ldots,t_n) & \text{if } t = g(t_1,\ldots,t_n),\ g \neq f \\ f(T_1 \cup T_2) & \text{if } t = f(t_1,t_2) \end{cases}$$

where

$$T_i = \begin{cases} \{u_1,\ldots,u_m\} \text{ if } flat(t_i,f) = f(u_1,\ldots,u_m) \\ \{flat(t_i,f)\} \quad \text{otherwise} \end{cases}$$

Definition 3 (labeled signature, lab). *Let $\Sigma = (S,<,F)$ be an order-sorted signature containing AC, C and free function symbols. Its associated unsorted labeled signature Σ^{os} is given by*

$$\begin{aligned} \{f_\Psi \mid f\colon A\,B \to A \in \Sigma_{AC},\ & \\ \Psi \text{ a finite multiset of sorts} \leq A\} \cup & \\ \{f_{[A',B']} \mid f\colon A\,B \to C \in \Sigma_C, [A',B'] \text{ an unordered} & \\ \text{pair of sorts } A' \leq A, B' \leq B\} \cup & \\ \{f_{A'_1,\ldots,A'_n} \mid f\colon A_1 \ldots A_n \to C \in \Sigma \setminus (\Sigma_C \cup \Sigma_{AC}), & \\ A'_i \leq A_i \text{ for all } 1 \leq i \leq n\}. & \end{aligned}$$

For a function symbol $f_{\mathcal{A}}$ of Σ^{os} where $\mathcal{A}$ is a multiset, an unordered pair or a sequence of sorts, we denote by $erase(f_{\mathcal{A}})$ the unlabeled function symbol f and by $lab(f_{\mathcal{A}})$ the label $\mathcal{A}$.

Note that Σ^{os} is countably infinite in general if Σ is countable. Next we define a mapping from terms over Σ to terms over the labeled signature Σ^{os}.

Definition 4 (labeling terms). *Let $\Sigma = (S,<,F)$ be an order-sorted signature containing AC, C and free function symbols which is preregular modulo the AC and C axioms. The mapping*

$\overline{\cdot}: \mathcal{T}(\Sigma, V) \to \mathcal{T}(\Sigma^{os}, V)$ *is defined by*

$$\overline{t} = \begin{cases} x & \text{if } t = x \in V \\ f_{ls(t_1),\ldots,ls(t_n)}(\overline{t_1}, \ldots, \overline{t_n}) & \text{if } t = f(t_1, \ldots, t_n), \\ & f \in \Sigma \setminus (\Sigma_C \cup \Sigma_{AC}) \\ f_{[ls(t_1),ls(t_2)]}(\overline{t_1}, \overline{t_2}) & \text{if } t = f(t_1, t_2),\ f \in \Sigma_C \\ f_\Psi(\lambda(t_1, f, \Psi), \lambda(t_2, f, \Psi)) & \text{if } t = f(v_1, v_2),\ f \in \Sigma_{AC}, \\ & flat(t, f) = f(u_1, \ldots, u_m), \\ & \Psi = \{ls(u_1), \ldots, ls(u_m)\} \end{cases}$$

where

$$\lambda(u, f, \Psi) = f_\Psi(\lambda(u_1, f, \Psi), \lambda(u_2, f, \Psi))$$

if $u = f(u_1, u_2)$ *and* $\overline{u}$ *otherwise.*

Note that, by definition, constants are always *labeled by the empty sequence* ϵ*. Thus, for notational simplicity we omit the label of constants if no confusion arises. Slightly abusing notation, we denote by erase the inverse mapping of* $\overline{\cdot}$*, which erases the labels of function symbols and is defined for terms* $f(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma^{os}, V)$ *as* $erase(f)(erase(t_1), \ldots, erase(t_n))$*.*

Example 2. Consider the signature of $\mathcal{E}$ in Example 1. The labeled term $\overline{f(f(s(b), s(b)), b)}$ is

$$f_{\{A,A,B\}}(f_{\{A,A,B\}}(s_B(b), s_B(b)), b).$$

By labeling terms in equations of an OS theory, we obtain a theory transformation that maps OS theories modulo axioms to unsorted theories modulo axioms.

Definition 5 (labeled theory). *Let* $\mathcal{E} = (\Sigma, B_0, R)$ *be an OS theory with axioms* B_0 *including only C and AC axioms. By* $\overline{\mathcal{E}}$ *we denote the unsorted theory* $(\Sigma^{os}, \overline{B_0}, \overline{R})$ *where*

$$\overline{B_0} = \{\overline{l\theta} = \overline{r\theta} \mid l = r \in B_0,\ \theta \text{ a sort specialization}\}$$
$$\overline{R} = \{\overline{l\theta} \to \overline{r\theta} \mid l \to r \in R,\ \theta \text{ a sort specialization}\}$$

Example 3. Consider the module `LIST-MSET-NAT` of our running example of Section 1 and the equation

```
(MS ; NL) ; L = MS ; (NL ; L)
```

that is added by the theory transformation of Section 5 below, thus eliminating the associativity axiom for ";". The sorts of the variables are $MS\colon MSet, NL\colon NeList$ and $L\colon List$. Hence, the labeled version of this equation (considering the identity specialization) is

$$\texttt{(MS ;}_{\texttt{MSet,NeList}}\ \texttt{NL) ;}_{\texttt{NeList,List}}\ \texttt{L} = \texttt{MS ;}_{\texttt{MSet,List}}\ \texttt{(NL ;}_{\texttt{NeList,List}}\ \texttt{L).}$$

Thus, the various occurrences of the symbol ";" in the equation are explicitly disambiguated. Moreover, there are 24 specializations of this equation (including the identity), because the variables can be specialized in the following way:

$$MS \text{ to } MSet, Nat$$
$$NL \text{ to } NeList, MSet, Nat$$
$$L \text{ to } List, NeList, MSet, Nat$$

Hence, according to Definition 5, our equation is transformed into 24 labeled equations given by

$$\texttt{(MS ;}_{\texttt{X,Y}}\ \texttt{NL) ;}_{\texttt{Y,Z}}\ \texttt{L} = \texttt{MS ;}_{\texttt{X,Z}}\ \texttt{(NL ;}_{\texttt{Y,Z}}\ \texttt{L)}$$
where $X \in \{MSet, Nat\}, Y \in \{NeList, MSet, Nat\}$,
$Z \in \{List, NeList, MSet, Nat\}$.

As shown by Example 4 below, the theory transformation of Definition 5 is *not* a sound reflection (w.r.t. $\bar{\cdot}$). However, it can be extended to a theory transformation that is a sound reflection (w.r.t. $\bar{\cdot}$). This is detailed in the appendix of this report. Nevertheless, in this short version of the paper we use labeled versions of rewrite rules exclusively to derive a "sort-aware" recursive dependency relation (cf. Definition 8 below). For this purpose, it suffices to use the simpler theory transformation of Definition 5. Hence, for the sake of simplicity, we use only this transformation in the rest of this section.

Example 4. Consider the theory $\mathcal{E} = (\Sigma, B_0, R)$ of Example 1. The rules of $\overline{\mathcal{E}} = (\Sigma^{os}, \overline{B_0}, \overline{R})$ are

$$\begin{aligned} f_{\{B,B\}}(b,x) &\to f_{\{A,A\}}(s_B(b), s_B(b)) \\ f_{\{A,B\}}(b, s_B(x)) &\to f_{\{B,B\}}(b,x) \\ f_{\{A,B\}}(b,x) &\to f_{\{A,A\}}(s_B(b), s_B(b)) \\ f_{\{A,B\}}(b, s_A(x)) &\to f_{\{A,B\}}(b,x) \end{aligned}$$

In the unlabeled system, we have $s = f(b, f(b,b)) \to_R f(s(b), s(b)) = t$, but after labeling the corresponding reduction is impossible: $\overline{s} = f_{\{B,B,B\}}(b_\epsilon, f_{\{B,B,B\}}(b_\epsilon, b_\epsilon)) \not\to_{\overline{R}} f_{\{A,A\}}(s_B(b), s_B(b)) = \overline{t}$.

Based on the notion of recursive dependency and the labeling of function symbols, we now define well-founded recursive OS theories modulo axioms. These well-founded recursive theories are guaranteed to be terminating and properties like confluence and sufficient completeness can be verified incrementally. Left-hand sides of equations in well-founded OS theories are linear *patterns*, or linear constructor terms with constructor right-hand sides in case constructors are not B_0-free ($\Omega \subseteq \Sigma$ is B_0-free iff for each specialization $l\nu \to r\nu$ with $l \to r \in R$, $l\nu$ is not an Ω-term).

Definition 6 (pattern). *Let $\mathcal{E} = (\Sigma, B_0, E)$ be an OS theory where $\Sigma = \mathcal{D} \uplus \Omega$ is partitioned into defined function symbols and constructors.*[6] *A term t is a* pattern *if it is linear and every proper subterm is from $\mathcal{T}(\Omega, V)$.*

In order to obtain termination of well-founded recursive theories, arguments of functions called recursively by other mutually recursively dependent functions have to decrease. For example, when considering a rewrite rule $f(s_1, s_2, s_3) \to C[f(t_1, t_2, t_3)]$ for some context C, we demand that the tuple $\langle s_1, s_2, s_3 \rangle$ resp. the multiset $\{s_1, s_2, s_3\}$ is greater than $\langle t_1, t_2, t_3 \rangle$ resp. $\{t_1, t_2, t_3\}$ w.r.t. some extension of the subterm ordering to tuples or multisets that preserves well-foundedness. Whether arguments of non-commutative functions are compared as tuples (thus using a lexicographic ordering *lex*) or multisets (thus using a multiset ordering *mul* or a more specialized version *tup* for AC functions) is determined by a status function. This provides a maximum of flexibility, since arguments of different functions can be compared in different ways (unless the functions are mutually recursive). The idea of recursive calls to functions with smaller collections of arguments is formalized by the notion of *argument decreasing rules*, where all recursive function calls are to functions with smaller argument collections.

Definition 7 (decreasing rule). *Let $\mathcal{E} = (\Sigma, B_0, R)$ be an OS theory where B_0 consists exclusively of AC and C axioms and let $stat\colon \Sigma \to \{lex, mul\}$ be a status function on Σ. Moreover, let $l \to r$ be a rule of R and g a function symbol such that $root(l)$ and g are either both AC or none of them is AC. We say that $l \to r$ is* g-argument decreasing *if $stat(root(l)) = stat(g)$ and for each subterm $r|_p$ of r with $root(r|_p) = g$ there are terms $l' =_{B_0} l$ and $w' =_{B_0} r|_p$ such that*

$$\{l''|_1, \ldots, l''|_{ar(root(l''))}\} \rhd^{tup} \{w''|_1, \ldots, w''|_{ar(root(w''))}\}$$

if $root(l'), root(w') \in \Sigma_{AC}$ where $l'' = flat(l', root(l'))$ and $w'' = flat(w', root(w'))$[7]*; and*

$$\{l'|_1, \ldots, l'|_{ar(root(l'))}\} \rhd^{mul} \{w'|_1, \ldots, w'|_{ar(root(w'))}\}$$

[6] But note that we can have $f\colon s_1, \ldots, s_n \to s \in \mathcal{D}$ and another $f\colon s'_1, \ldots, s'_n \to s' \in \Omega$, as illustrated for $f = \_;\_$ by our running example.

[7] $A \rhd^{tup} B$ (where $\rhd$ is the proper subterm relation of terms) for multisets A and B of terms means that $A = A' \cup C$, $B = B' \cup C$, $A' \neq \emptyset$ and there is a (possibly partial) surjective mapping $\phi\colon A' \to B'$, such that $\phi(a) = b$ implies $a \rhd b$.

if $root(l'), root(w') \notin \Sigma_{AC}$ *and* $stat(root(l)) = mul$*; and*

$$\langle l'|_1, \ldots, l'|_{ar(root(l'))}\rangle \rhd^{lex} \langle w'|_1, \ldots, w'|_{ar(root(w'))}\rangle$$

if $root(l'), root(w') \notin \Sigma_{AC}$ *and* $stat(root(l)) = lex$*.*

Note that the status function does not assign the *tup* extension of the subterm ordering to non-AC function symbols, as the *mul* extension is more general and thus subsumes the use of the *tup* extension. For AC function symbols it is crucial to compare multisets of arguments by $\rhd^{tup}$ instead of $\rhd^{mul}$ in order to obtain a well-founded decrease of argument multisets. Example 5 shows that theories may be non-terminating even if multisets of mutually recursive functions decrease w.r.t $\rhd^{mul}$.

Example 5. Consider the unsorted theory $\mathcal{E}$ (already used in Section 1) using a defined binary AC-function symbol f, a binary constructor g and a constructor constant a. The single rule is $f(g(x,y),a) \to g(f(x,y),g(a,a))$.

We have $f \blacktriangleright_{\mathcal{E}} g$ but not $g \blacktriangleright_{\mathcal{E}} f$. Hence, we compare the arguments of $flat(f(g(x,y),a),f) = f(g(x,y),a)$ and $flat(f(x,y),f) = f(x,y)$. We have $\{g(x,y),a\} \rhd^{mul} \{x,y\}$.

Indeed, $\mathcal{E}$ is not AC-terminating:

$$\begin{aligned} f(g(f(a,a),g(a,a)),a) &\to \\ \to\ g(f(f(a,a),g(a,a)),g(a,a)) &=_{AC} \\ =_{AC}\ g(f(f(g(a,a),a),a),g(a,a)) &\to \\ \to\ g(\underline{f(g(f(a,a),g(a,a)),a)},g(a,a)) & \end{aligned}$$

Note however that $\{g(x,y),a\} \not\rhd^{tup} \{x,y\}$.

Now we are ready to define well-founded recursive OS theories modulo axioms. Ultimately, our goal is to show that well-founded recursive OS theories are compatible with a recursive path ordering that is compatible with C and AC axioms (i.e. an $ACRPO$). Due of the presence of rules like $c(c(x,y),z) \to c(x,c(y,z))$ (cf. e.g. Example 9 and Section 5 below) and commutative function symbols, it is crucial to compare arguments of some functions lexicographically and others by multiset orders. Moreover, for rules involving subsort-overloaded AC function symbols, sorts may or may not be crucial and it may even be advantageous to ignore sorts of certain function symbols (cf. Example 6 below). Hence, the notion of well-founded recursive OS theory modulo axioms is *parameterized* by two status functions *stat* and $stat_{ac}$. Note, however, that this does not compromise the syntactic and easy-to-check character of well-founded recursion, since the possible choices for these status functions are finite for each finite OS theory. However, finding working status functions may be computationally hard. To solve this problem we show in Section 4 below that these status functions can be computed incrementally when checking hierarchies of order-sorted theories for being well-founded recursive. Hence, provided that the theory extensions in such a hierarchy are small, the task of choosing suitable status functions is feasible.

Definition 8 (well-founded theories). *Let* $\mathcal{E} = (\Sigma, B_0, R)$ *be an OS theory with constructors* $\Omega \subseteq \Sigma$ *where the structural axioms* B_0 *are either AC or C axioms. Let* $stat\colon \Sigma \to \{lex, mul\}$ *be a status function where* $stat(f) = mul$ *for all* $f \in \Sigma_C \cup \Sigma_{AC}$ *and where* $f \blacktriangleright_{\mathcal{E}} g, g \blacktriangleright_{\mathcal{E}} f$ *implies* $stat(f) = stat(g)$ *and. Let* $stat_{ac}\colon \Sigma_{AC} \to \{s, us\}$[8] *be a status function where* $f, g \in \Sigma_{AC}$ *and* $f \blacktriangleright_{\mathcal{E}} g, g \blacktriangleright_{\mathcal{E}} f$ *implies* $stat_{ac}(f) = stat_{ac}(g)$*.*

$\mathcal{E}$ *is well-founded recursive iff (i)* $g \blacktriangleright_{\mathcal{E}} h$ *and* $h \blacktriangleright_{\mathcal{E}} g$ *(*$h, g \in \Sigma$*) implies that* h *and* g *are either both AC symbols or both non-AC-symbols, and (ii) there are status function stat and* $stat_{ac}$ *such that for each rule* $l \to r$ *and each specialization* θ *the following properties hold:*

1. *Either* l *is a linear constructor term, or if not then* l *is a pattern in case* $root(l) \in \Sigma \setminus \Sigma_{AC}$ *and* $flat(l, root(l))$ *is a pattern in case* $root(l) \in \Sigma_{AC}$*.*

[8] The function $stat_{ac}$ determines for an AC function symbol f whether sorts are taken into account (s for sorted) or not (us for unsorted) when comparing labeled versions $f_\Psi, f_{\Psi'}$ of this function in the $ACRPO$ we are going to use to prove termination of well-founded recursive theories (cf. Theorem 1 below).

2. *If l is a constructor term, then so is r.*
3. *For every (not necessarily proper) subterm $r|_p$ of r,*

$$root(\overline{r\theta}|_p) \blacktriangleright_{\overline{\mathcal{E}}} root(\overline{l\theta})$$

(resp. $root(r|_p) \blacktriangleright_{\mathcal{E}} root(l) \in \Sigma_{AC}$ and $stat_{ac} = us$) implies that

$$\overline{l\theta} \to \overline{r\theta} \text{ is } root(\overline{r\theta}|_p) \text{ argument decreasing (w.r.t. } stat)$$

(resp. that $l \to r$ is $root(r|_p)$ argument decreasing).
4. *Assume $root(l) = root(r|_p)$ for some $p \in Pos(r)$, $root(r|_p) \blacktriangleright_{\mathcal{E}} root(l)$ and $stat_{ac}(root(l)) = s$ and consider the multiset S of arguments of $root(\overline{l\theta})$ in the term $\overline{l\theta}$ as well as the multiset T of arguments of $root(\overline{r|_p\theta})$ in the term $\overline{r|_p\theta}$. For every variable $x \in T \setminus S$, there exists a term $s \in S \setminus T$, such that $ls(s) > ls(x)$. Moreover, $lab(root(\overline{l\theta})) \geq^{mul} lab(root(\overline{r|_p\theta}))$.*
5. *If l is a constructor term and $root(l)$ is associative and commutative, then*

$$root(flat(l, root(l))|_p) \not\blacktriangleright_{\mathcal{E}} root(l)$$

for all positions $p \in Pos_\Sigma(flat(l, root(l)))$ with $p > \epsilon$.

The status function *stat* in Definition 8 determines whether arguments of function symbols are compared lexicographically or by multiset comparison. Mutually recursive function symbols must have the same status. Moreover, arguments of commutative function symbols may only be compared by multiset orders. Hence, the problem of finding suitable statuses for non-commutative functions is very similar to the problem of finding suitable statuses for functions when checking TRSs for RPOS compatibility for which efficient methods exist (cf. e.g. [28]). These methods can be used to determine the status function when checking theories for being well-founded recursive. The other status function $stat_{ac}$ determines whether sorts are taken into account when comparing arguments of AC-function symbols. The reason why we make this distinction is that in the presence of AC function symbols it is not always desirable to take sorts into account, because the labels of AC-function symbols appearing in equations may change through instantiations.

Example 6. Consider an OS theory containing two sorts A, B with $A < B$, an AC function symbol $g\colon B, B \to B$, two unary function $h\colon B \to B$ and $t\colon B \to A$ and a constant $a : B$. Consider a rule

$$g(h(x), h(y)) \to g(x, y).$$

We have

$$root(\overline{g(h(x), h(y))}) = g_{\{B,B\}}$$

and indeed $root(\overline{g(h(x), h(y))\sigma}) = g_{\{B,B\}}$ for every substitution σ. On the other hand, we have e.g. $root(\overline{g(x, y)\sigma}) = g_{\{B,B,B\}}$ if $x\sigma = g(a, a), y\sigma = y$. Hence, when instantiating the rule, there may be an increase in the multiset of sorts of the root symbol of the right-hand side compared to that of the root symbol of the left-hand side of the rule. In this case it is preferable to consider labeled occurrences of g as equal, since there is a decrease in the arguments of the recursive function call. We would have $stat_{ac}(g) = us$ in this case.

On the other hand, consider a rule

$$g(a, a) \to g(h(a), h(a)).$$

We have

$$root(\overline{g(a, a)}) = g_{\{B,B\}}$$
$$root(\overline{g(h(a), h(a))}) = g_{\{A,A\}}.$$

Thus, in order to orient this rule, e.g. by an $(AC)RPO$, it is preferable to consider the symbols $g_{\{B,B\}}$ and $g_{\{A,A\}}$ as different ones, so that $g_{\{B,B\}}$ can be larger in the precedence of function symbols than $g_{\{A,A\}}$. We would have $stat_{ac}(g) = s$ in this case.

The concrete value of $stat_{ac}$ for AC functions can be determined by checking a theory for possible increases in the multisets of sorts (w.r.t. the multiset extension of the subsort ordering) of the arguments in recursive calls to other (or the same) AC functions. If there are no such increases $stat_{ac}$ of the corresponding function should be set to s, otherwise it should be set to us.

Example 7. Consider the functional Maude module `NATURAL` of the running example of Section 1. It contains an identity axiom so it is outside the scope of well-founded recursive theories. However, by the semantics-preserving theory transformation described in Section 5 below, we obtain the following module which, considered as an OS theory modulo C, is sort-decreasing and well-founded recursive.

```
fmod TR-NATURAL is pr TRUTH-VALUE .
 sort Nat .
 op 0 : -> Nat [ctor] .
 op s : Nat -> Nat [ctor] .
 op _+_ : Nat Nat -> Nat [comm] .
 ops even odd : Nat -> Bool .
 vars N M : Nat .
 eq N + 0 = N .
 eq s(N) + s(M) = s(s(N + M)) .
 eq even(0) = true .
 eq odd(0) = false .
 eq odd(s(N)) = even(N) .
 eq even(s(N)) = odd(N) .
endfm
```

Note that, since there is only one sort in this module, there are no non-trivial sort specializations and thus the equations of the labeled theory are identical with those of the unlabeled one (modulo names of function symbols; cf. Definition 5). Moreover, there are no AC axioms. Hence, the status function $stat_{ac}$ is irrelevant. Finally, the choice of the status function $stat$ is completely arbitrary since the module is well-founded recursive w.r.t. every choice of the status function.

As for $stat$, mutually recursive AC function symbols have to agree on $stat_{ac}$ in well-founded recursive OS theories. In the presence of AC function symbols $f \in \Sigma$ with $stat_{ac}(f) = s$ in a well-founded recursive OS theory $\mathcal{E}$, two additional complications, compared to non-AC function symbols or those with a $stat_{ac}$ of us, may arise, but, as we explain below, these two potential complications do not cause any problems.

First, since Σ^{os} is infinite, there might be infinite decreasing $\blacktriangleright_{\overline{\mathcal{E}}}$ chains that are not looping. However, by Item (4) of Definition 8 we have $\Psi >^{mul} \Psi'$ whenever, $f_\Psi \blacktriangleright_{\overline{\mathcal{E}}} g_{\Psi'}$ ($f_\Psi \neq g_{\Psi'}$ and $f, g \in \Sigma_{AC}$) where $<$ is the (well-founded) subsort ordering. Hence, there are no infinite non-looping $\blacktriangleright_{\overline{\mathcal{E}}}$ chains.

The second potential complication is that the labeling is not-stable under substitutions as illustrated by Example 6. Item (4) of Definition 8 ensures that this stability is restored by ensuring that the sort of every variable occurring directly under an AC function symbols in the (subterm of the) right-hand side in question is dominated by a larger sort in the left-hand side.

The key result of this section is that well-founded recursive sort-decreasing OS theories modulo axioms are terminating. Therefore, our notion of well-founded recursive OS-theories provides: (i) a new formal definition that extends to the order-sorted and modulo C and AC cases the intuitive notion of "specification of a set of well-founded recursive functions"; (ii) a machine-checkable way of ascertaining whether a specification is indeed well-founded recursive; and (iii) a proof that such specifications are always terminating. Furthermore, as shown in Section 4, the checking that a specification is well-founded recursive can be made in a modular and *incremental* way. In practice, of course, we will want our well-founded recursive specifications to be also *confluent*, and *sufficiently complete* with respect to their constructors. These extra properties can also be checked incrementally, as explained in Section 4.

Theorem 1. *Let $\mathcal{E} = (\Sigma, B_0, R)$ be a sort-decreasing well-founded recursive OS theory where the structural axioms B_0 are either AC or C axioms. Then $\mathcal{E}$ is B_0-terminating.*

Note that the sort-decreasingness requirement is essential in Theorem 1, as shown by the following example.

Example 8. Consider the following OS theory $\mathcal{E}$ without structural axioms. We have sorts s_1 and s_2 where $s_1 < s_2$. Moreover, there is a unary function symbol f typed $f\colon s_2 \to s_2$, another unary function symbol g typed $g\colon s_2 \to s_1$ and a constant a of sort s_2. The rules are

$$f(a) \to f(g(a)),$$
$$g(x) \to x$$

where x is a variable of sort s_2. This theory is well-founded recursive. For the problematic first rule we have

$$\overline{f(a)} = f_{s_2}(a_\epsilon)$$

and

$$\overline{f(g(a))} = f_{s_1}(g_{s_2}(a_\epsilon)).$$

Moreover, $f_{s_1} \not\blacktriangleright_{\overline{\mathcal{E}}} f_{s_2}$ and $g_{s_2} \not\blacktriangleright_{\overline{\mathcal{E}}} f_{s_2}$.

However the theory is non-terminating as is witnessed by the cyclic reduction sequence

$$f(a) \to f(g(a)) \to f(a).$$

The problem here is that $\mathcal{E}$ is not sort-decreasing, since $ls(g(x)) = s_1 \not\geq s_2 = ls(x)$ for the second rule if x is of sort s_2.

3.1 Many-Sorted Well-Founded Functions as a Special Case of Well-Founded Theories

To further explain the generality of our notion of well-founded recursive theories we show in detail how it captures as a special case a very general notion of well-founded recursive definition in the many-sorted case without axioms. In a sense this is the most general comparison we can make with previous notions, since to the best of our knowledge the notion has not been previously studied in the order-sorted and modulo cases.

To simplify the exposition we focus on the case of recursive definitions without mutual recursion. It is well-known that by adding extra data constructors, such as product types, several mutually recursive functions can be expressed as a single function.

Definition 9. *Let Ω be a many sorted signature of constructors. A* well-founded recursive tower *is a sequence*

$$(f_1\colon s_1^1 \dots s_{n_1}^1 \to s_1, R_{f_1}) \dots (f_m\colon s_1^m \dots s_{n_m}^m \to s_m, R_{f_m})$$

consisting of fresh function symbols $f_1, \dots, f_m$ not in Ω and of sets of rules R_{f_i}, $1 \leq i \leq m$, defining each f_i such that the rules in R_{f_i} are of the form

$$f_i(t_1, \dots, t_{n_i}) \to C[f_i(u_1^1, \dots, u_{n_i}^1) \dots f_i(u_1^k, \dots, u_{n_i}^k)]$$

with $t_1, \dots, t_{n_i}$ Ω-terms and where:

(i) $f_i(t_1, \dots, t_{n_i})$ is linear and the right-hand side of the rule involves only variables occurring in $f(t_1, \dots, t_{n_i})$.
(ii) The context C is a term in the signature $\Omega \cup \{f_1, \dots, f_{i-1}\}$.
(iii) $k \geq 0$, and for each $1 \leq j \leq k$ the set $\{1, \dots, n_i\}$ can be split into two disjoint subsets $A \uplus B = \{1, \dots, n_i\}$ such that $1 \in A$, and
  (1) for each $a \in A$, $t_a \trianglerighteq u_a^j$;
  (2) for each $b \in B$, either
    * $u_b^j \in \mathcal{T}(\Omega \cup \{f_1, \dots, f_{i-1}\}, \mathcal{X})$, *or*
    * $u_b^j = f_i(v_1^b, \dots, v_{n_i}^b)$ *with* $(t_1, \dots, t_{n_i}) \rhd (v_1^b, \dots v_{n_i}^b)$; *and*
  (3) there is an $a \in A$ with $a < min(B)$ such that $(t_1, \dots, t_a) \rhd (u_1^j, \dots, u_a^j)$

where by definition $(t_1, \ldots, t_l) \rhd (t'_1, \ldots, t'_l)$ *iff for each* $1 \leq i \leq l$, $t_i \trianglerighteq t'_i$, *and there is a* $j \in \{1, \ldots, l\}$ *such that* $t_j \rhd t'_j$.

Note that this definition includes as a special case all primitive recursive functions, where the terms $f_i(u^j_1, \ldots, u^j_{n_i})$ are such that $(t_1, \ldots, t_{n_i}) \rhd (u^j_1, \ldots, u^j_{n_i})$. Note also that the equations for Ackerman's function in the introduction are a special instance of the above definition. In practice, two more conditions are required of such recursive towers:

(1) *disjoint patterns*, that is, if $f_i(t_1, \ldots, t_{n_i})$ and $f_i(t'_1, \ldots, t'_{n_i})$ are two different left-hand sides in R_{f_i}, which we may assume have distinct variables, then the patterns do not unify.
(2) *sufficient completeness*, that is, the collection of patterns

$$\{(t_1, \ldots, t_{n_i}) \mid \exists t' s.t.\ f_i(t_1, \ldots, t_{n_i}) = t' \in R_{f_i}\}$$

cover the product sort $s^i_1 \times \ldots \times s^i_{n_i}$, in the sense that any ground term in that product is an instance of one of the patterns.

Those are precisely the conditions of confluence and sufficient completeness that we show how to check incrementally in Section 4. The main result is now:

Theorem 2. *For any well-founded recursive tower as in Definition 9, the equational theory* $(\Omega \cup \{f_1, \ldots, f_m\}, R_{f_1} \cup \ldots \cup R_{f_m})$ *is a well-founded recursive many-sorted theory.*

4 Verifying Properties of OS Theories Incrementally

For well-founded recursive OS theories modulo axioms we can check important properties like termination, confluence, sort-decreasingness and sufficient completeness incrementally in the presence of theory hierarchies that satisfy reasonable conditions. These conditions are formalized in the notion of *fair extension*. The basic idea of fair extensions is that extending modules do not interfere with their base modules, i.e., they do not introduce new constructors of sorts of the base module and they do not redefine existing functions.

Definition 10 (fair extension). *Assume that* $\mathcal{E}_1 = (\Sigma_1, B^1_0, R_1)$ *and* $\mathcal{E}_2 = (\Sigma_1 \cup \Sigma_2, B^1_0 \cup B^2_0, R_1 \cup R_2)$ *are OS theories where the* $B^i_0 s$ *are* C *or* AC *axioms for* $i \in \{1, 2\}$. Σ_1 *and* $\Sigma_1 \cup \Sigma_2$ *are order-sorted signatures. We write* $\Sigma_1 = (S_1, <_1, F_1)$ *and* $\Sigma_1 \cup \Sigma_2 = (S_1 \cup S_2, <_1 \cup <_2, F_1 \cup F_2)$. *Furthermore,* F_i *is divided into constructors* Ω_i *and defined function symbols* $\mathcal{D}_i$ *for both* $i \in \{1, 2\}$. $\mathcal{E}_2$ *is a* fair extension *of* $\mathcal{E}_1$ *iff:*

1. *every function symbol* f *from* Σ_1 *is* AC *(resp.* C*) in* $\mathcal{E}_2$ *iff it is* AC *(resp.* C*) in* $\mathcal{E}_1$*;*
2. Σ_2 *does not introduce subsorts of sorts of* Σ_1*, i.e.* $s \in S_1 \wedge s' <_1 \cup <_2 s$ *for some* $s' \in S_1 \cup S_2$ *implies* $s' <_1 s$*;*
3. Σ_2 *does not contain new constructors of some sort of* S_1*, i.e.* $f\colon s_1, \ldots, s_n \to s \in \Omega_2$ *implies* $s \notin S_1$*;*
4. *for every rule* $l \to r \in R_2$ *and every function symbol* $f\colon s_1, \ldots, s_n \to s \in F_1$, l *and the term* $f(x^1_{s_1}, \ldots, x^n_{s_n})$ *do not unify in an order-sorted fashion modulo axioms (where* x_s *denotes a variable of sort* s*).*
5. *if* f *is a defined* AC *symbol in* $\mathcal{E}_1$ *and* $f \blacktriangleright_{\mathcal{E}_1} g$, $g \not\blacktriangleright_{\mathcal{E}_1} f$*, then* $g \not\blacktriangleright_{\mathcal{E}_2} f$*.*
6. *if* $c \in \Omega_1$ *and there is a rule* $l \to r$ *from* R_1 *such that* $root(l) = c$*, then* l *does not overlap (order-sorted modulo axioms) with the left-hand side of any rule* $l' \to r'$ *of* R_2 *in case* $root(l')$ *is a defined symbol in* $\Sigma_1 \cup \Sigma_2$*, and* c *does not occur below the root of* l' *in case* $root(l')$ *is an associative-commutative constructor.*

The first item of Definition 10 ensures that overloaded function symbols have the same set of attached axioms. Items 2 – 4 ensure that no new subsorts and constructors of sorts of the base module are introduced and no functions of the base module are redefined. Item 5 makes sure that no additional mutual recursive dependency of AC symbols is introduced by the extending module, and item 6 is needed to prevent overlaps of rules from R_1 that have constructor terms as left-hand sides with rules from R_2.

In the rest of this section we denote by $\mathcal{E}_1 = (\Sigma_1, B_0^1, R_1)$ an OS theory modulo axioms B_0^1 and by $\mathcal{E}_2 = (\Sigma_1 \cup \Sigma_2, B_0^1 \cup B_0^2, R_1 \cup R_2)$ a fair extension of $\mathcal{E}_1$. By $\mathcal{E}_2'$ we denote the OS theory $(\Sigma_1 \cup \Sigma_2, B_0^1 \cup B_0^2, R_2)$. First, we show modularity of sort-decreasingness. Then we show that the property of being well-founded recursive itself is modular, provided that the base and extending theory agree on the status functions.

Theorem 3 (modularity of sort-decreasingness). *If $\mathcal{E}_1$ and $\mathcal{E}_2'$ are both sort-decreasing, then so is $\mathcal{E}_2$.*

Theorem 4 (modularity of well-founded recursion). *If $\mathcal{E}_1$ and $\mathcal{E}_2'$ are well-founded recursive w.r.t. to compatible functions $stat^1, stat^1_{ac}$ and $stat^2, stat^2_{ac}$, then so is $\mathcal{E}_2$*[9].

Note that we require that the status functions of the base theory and the extending theory are compatible. In a naive mechanization of incremental checks for well-founded recursiveness this could necessitate backtracking, i.e., modifying the status function of a base module depending on an extending theory. To avoid this backtracking, we propose to compute the status functions incrementally in a "by need" fashion. This means that a specific status is assigned to a function symbol (resp. an AC symbol) only if this status is crucial for the theory in question to be well-founded recursive. Otherwise, the status is left open, so that it can be set later when incrementally checking an extending theory for well-founded recursiveness. For example, consider the theory of Example 7. It is well-founded recursive w.r.t. every status function *stat*. Hence, the status of functions can later be set arbitrarily when checking an extending module. A fully general implementation of this idea could, for example, compute a set of status functions for which a module is well-founded recursive. Then when checking an extending module for well-founded recursiveness one could choose suitable status functions from this set of possible ones. Next we show that confluence is modular for fair extensions of well-founded recursive theories.

Theorem 5 (modularity of confluence). *Assume $\mathcal{E}_1$ and $\mathcal{E}_2'$ are well-founded recursive w.r.t. to compatible functions $stat^1, stat^1_{ac}$ and $stat^2, stat^2_{ac}$. If $\mathcal{E}_1$ and $\mathcal{E}_2'$ are confluent then so is $\mathcal{E}_2$.*

Note that for sufficient completeness the adequate notion of modular check consists of checking the property only for *new* defined function symbols.

Theorem 6 (modularity of sufficient completeness). *Assume $\mathcal{E}_1$ and $\mathcal{E}_2'$ are well-founded recursive w.r.t. to compatible functions $stat^1, stat^1_{ac}$ and $stat^2, stat^2_{ac}$. If $\mathcal{E}_1$ is sufficiently complete and for every function $f\colon s_1, \ldots, s_n \to s \in \mathcal{D}_2 \setminus \mathcal{D}_1$ and every ground substitution σ mapping variables to irreducible constructor terms, $f(x^1_{s_1}, \ldots, x^n_{s_n})\sigma$ is either $\mathcal{E}_2$-reducible or a constructor term (x_s denotes a variable of sort s), then $\mathcal{E}_2$ is sufficiently complete.*

This way of incrementally checking sufficient completeness is compatible with existing automated methods to check the property. Roughly, the idea of these methods is to check whether ground terms rooted by a defined function symbol and having only constructor terms as proper subterms are either reducible, or constructor terms (which is possible as the root symbol might be subsort overloaded). This is done by describing the respective languages of terms by (propositional) tree automata and then reducing the problem to an emptiness problem for tree automata (we refer to [15] and [16] for more details). The method is suitable for incremental checks following Theorem 6, since it can easily be adapted to consider only terms rooted by defined function symbols of the extending theory instead of all.

Corollary 1. *Assume $\mathcal{E}_1$ and $\mathcal{E}_2'$ are well-founded recursive w.r.t. to functions $stat^1, stat^1_{ac}$ and $stat^2, stat^2_{ac}$ that are compatible. If $\mathcal{E}_1$ and $\mathcal{E}_2'$ are sort-decreasing and confluent and moreover, $\mathcal{E}_1$ is sufficiently complete and for every function $f\colon s_1, \ldots, s_n \to s \in \mathcal{D}_2 \setminus \mathcal{D}_1$ and every irreducible ground substitution σ (that maps variables only to constructor terms) $f(x^1_{s_1}, \ldots, x^n_{s_n})\sigma$ is $\mathcal{E}_2$-reducible (or a constructor term), then $\mathcal{E}_2$ is sort-decreasing, well-founded recursive (thus terminating), confluent and sufficiently complete.*

[9] Compatibility of two functions f_1 and f_2 here means that $f_1(a) = f_2(a)$ whenever $a \in Dom(f_1) \cap Dom(f_2)$.

Example 9. Consider the running example of Section 1. In order to apply our methods to the modules of this example, the identity axioms and those axioms specifying associativity for a non-commutative function symbol have to be eliminated. Indeed, we can eliminate these problematic axioms by the theory transformation presented in Section 5. This transformation yields the module of Example 7 for the module `NATURAL` and the following two transformed theories for the modules `MSET-NAT` and `LIST-MSAT-NAT`.

```
fmod TR-MSET-NAT is pr TR-NATURAL .
 sort MSet .
 subsort Nat < MSet .
 op _,_ : MSet MSet -> MSet [ctor assoc comm] .
 op null : -> MSet [ctor] .
 op card : MSet -> Nat .
 var MS : MSet .
 var N : Nat .
 var X : [MSet] .
 eq X , null = X .
 eq card(null) = 0 .
 eq card(N) = s(0) + card(null) .
 eq card(N,MS) = s(0) + card(MS) .
endfm

fmod TR-LIST-MSET-NAT is pr TR-MSET-NAT .
 sorts List NeList .
 subsorts MSet < NeList < List .
 op nil : -> List [ctor] .
 op _;_ : List List -> List .
 op _;_ : MSet NeList -> NeList [ctor] .
 op U : List -> MSet .
 var MS : MSet .
 var NL : NeList .
 var L : List.
 var Y : [List] .
 eq Y ; nil = Y .
 eq nil ; Y = Y .
 eq (MS ; NL) ; L = MS ; (NL ; L) .
 eq U(nil) = null .
 eq U(MS) = MS .
 eq U(MS ; NL) = MS, U(NL) .
endfm
```

We already established that the `TR-NATURAL` module is sort-decreasing and well-founded recursive in Example 7. Moreover, it is non-overlapping and thus (by termination) confluent. Sufficient completeness can automatically be verified by the Maude sufficient completeness checker (cf. e.g. [15]). The module `TR-MSET-NAT`, restricted to equations explicitly defined in the module and particularly not including the ones from the `TR-NATURAL` module, is sort-decreasing and well-founded recursive as well. This is seen for instance by using the status functions $stat(f) = mul$ for all f and $stat_{ac}(\_,\_) = us$. Confluence of the equations of `TR-MSET-NAT` follows again from non-overlappingness. All ground instances of $card(x)$ are reducible. Furthermore, `TR-MSET-NAT` is a fair extension of `TR-NATURAL`. Hence, it is sort-decreasing, well-founded recursive, confluent and sufficiently complete.

Finally, consider the module `TR-LIST-MSET-NAT` restricted to equations explicitly defined in the module and particularly not including the ones from the `TR-MSET-NAT` module. It is sort-decreasing and well-founded recursive (e.g. $stat(;) = lex$ and $stat(f) = mul$ for all other functions f). Furthermore, it is confluent because all critical pairs are joinable. All ground instances of $(x_1; x_2)$ are either reducible or constructor terms and all ground instances of $U(x)$ are reducible. As `TR-LIST-MSET-NAT` is a fair extension of `TR-MSET-NAT` it is thus sort-decreasing, well-founded recursive (thus terminating), confluent and sufficiently complete.

5 A Variant-Based Theory Transformation

So far, our incremental methods for checking the sort-decreasingness, confluence, termination, and sufficient completeness of order-sorted well-founded recursive specifications modulo B have been developed for the case where B can only have commutativity and/or associativity-commutativity axioms. But we are interested in checking the confluence, termination, and sufficient completeness of more general order-sorted specifications $\mathcal{E} = (\Sigma, B, R)$ where B can have any combination of associativity and/or commutativity and/or identity axioms (with some restrictions on the case of associativity without commutativity as explained below). The extension of our method to this more general case is accomplished by an automatic theory transformation $(\Sigma, B, R) \mapsto (\Sigma, B_0, \widehat{R} \cup \Delta)$ such that: (i) B_0 only involves commutativity and associativity-commutativity axioms; (ii) the theories $R \cup B$ and $B_0 \cup \widehat{R} \cup \Delta$ are semantically equivalent (as inductive theories, see below); and (iii) (Σ, B, R) is confluent, terminating, and sufficiently complete for Ω modulo B iff $(\Sigma, B_0, \widehat{R} \cup \Delta)$ has the same properties modulo B_0. Here we summarize and extend the basic ideas of the transformation and refer to [9] for further details.

The first key idea is to decompose B as a disjoint union $B = B_0 \cup \Delta$ so that (Σ, B_0, Δ) is confluent and terminating modulo B_0, and Δ contains all its B_0-extensions (cf. e.g. [26, Definition 10.4]). The second key idea is to generate the transformed rules $\widehat{R}$ by computing the most general Δ, B-*variants* ([7]) of the left-hand sides l for the rules $l \to r$ in R. Given a term t, a Δ, B-variant of t is a Δ, B-canonical form u of an instance of t by some substitution θ; more precisely, it is a pair (u, θ). Some variants are more general than others, so that variants form a preorder in a natural way. The set $\widehat{R}$ then consists of all rules $\widehat{l} \to r\theta$ such that $(\widehat{l}, \theta)$ is a maximal variant of l for $l \to r$ a rule in R. Our transformation $(\Sigma, B, R) \mapsto (\Sigma, B_0, \widehat{R} \cup \Delta)$ is actually the composition of two simpler transformations of this kind:

$$(\Sigma, B, R) \mapsto (\Sigma, B_1, \widehat{R}_1 \cup \Delta_1) \mapsto (\Sigma, B_0, \widehat{R} \cup \Delta)$$

where B_1 is obtained by removing *all identity axioms*[10] Δ_1 from B, and B_0 is obtained by removing from B_1 all axioms that are associative but not commutative, so that Δ is the union of Δ_1 and such associativity axioms oriented (in one of the two directions) as rules. In this way, B_0 only contains commutativity and/or associativity-commutativity axioms. We then incrementally check the confluence, termination, and sufficient completeness of (Σ, B, R) modulo B by checking the same properties modulo B_0 for the semantically equivalent theory $(\Sigma, B_0, \widehat{R} \cup \Delta)$ according to the methods already developed in Sections 3 and 4.

For the first transformation $(\Sigma, B, R) \mapsto (\Sigma, B_1, \widehat{R}_1 \cup \Delta_1)$ we are always guaranteed that the set of rules $\widehat{R}_1$ is finite if R is (see [9]). However, for the second transformation $(\Sigma, B_1, \widehat{R}_1 \cup \Delta_1) \mapsto (\Sigma, B_0, \widehat{R} \cup \Delta)$, which removes associative but not commutative axioms from B_1, we cannot in general guarantee that $(\Sigma, B_0, \widehat{R} \cup \Delta)$ is a finite theory. However, the *use of subsorts* can make it often the case in practice that $(\Sigma, B_0, \widehat{R} \cup \Delta)$ is finite. We can illustrate this interesting phenomenon with our running example. The first transformation, removing identities, leaves the equation `U(MS ; NL) = MS, U(NL)` unchanged because, since `NL` has sort `NeList`, the identity rules for `_;_` cannot be applied to any instance of `MS ; NL`. By orienting the associativity axiom as a rule `(L ; P); Q` $\to$ `L; (P ; Q)`, the only variant of the equation `U(MS ; NL) = MS, U(NL)` is itself, since the left-hand side of the associativity rule fails to have an order-sorted unifier with the subterm `MS ; NL`. Therefore, the second transformation also succeeds in our running example (for the resulting transformed modules see Examples 7 and 9).

For well-founded recursive specifications containing operators f that are associative but not commutative (with or without identity) we need to impose some conditions on such f and slightly modify the version in [9] of the second transformation $(\Sigma, B_1, \widehat{R}_1 \cup \Delta_1) \mapsto (\Sigma, B_0, \widehat{R} \cup \Delta)$. There should be only one such operator per connected component, with only two overloadings, which must be

[10] By adding a fresh top sort to each connected component as explained in Footnote 3, we only need to add a pair of identity rules $f(x, e) \to x$ and $f(e, x) \to x$, with x of sort $[s]$, for each connected component $[s]$ involving such axioms.

1. either of the form $f : \mathit{List}\ \mathit{List} \to \mathit{List}$, $f : \mathit{Elt}\ \mathit{NeList} \to \mathit{NeList}$ $[\mathit{ctor}]$, with $\mathit{Elt} < \mathit{NeList} < \mathit{List}$,
2. or of the form $f : \mathit{List}\ \mathit{List} \to \mathit{List}$, $f : \mathit{NeList}\ \mathit{Elt} \to \mathit{NeList}$ $[\mathit{ctor}]$, with $\mathit{Elt} < \mathit{NeList} < \mathit{List}$.

Moreover, there may be no other constructors of sort *List* or lower except those of sort *Elt* or lower. The names *Elt*, *NeList*, and *List* are immaterial and are only used to respectively suggest sorts for list elements, nonempty lists, and general lists. Furthermore, in order to make sure that the associativity equations introduced by the second transformation have constructor patterns below their top function symbol (so that the conditions in Section 3 apply to the transformed theory $(\Sigma, B_0, \widehat{R} \cup \Delta)$), instead of introducing an associativity rule

$$f(f(L,P),Q) \to f(L,f(P,Q))$$

for case (1) (resp. $f(L,f(P,Q)) \to f(f(L,P),Q)$ for case (2)) with L, P, Q of sort *List*, we introduce a more restricted rule

$$f(f(E,NL),Q) \to f(E,f(NL,Q))$$

for case (1) (resp. $f(Q,f(NL,E)) \to f(f(Q,NL),E)$ for case (2)) with E of sort *Elt*, NL of sort *NeList*, and Q of sort *List*. It is then easy to check that: (i) the left-hand sides of these more restricted rules have constructor patterns below and have no nontrivial overlaps with themselves; (ii) f so defined is sufficiently complete; and (iii) the unrestricted associativity equations are *inductive theorems* of the specification based on the more restricted associativity equations; that is, with this modified second transformation the theories $(\Sigma, B_1, \widehat{R}_1 \cup \Delta_1)$ and $(\Sigma, B_0, \widehat{R} \cup \Delta)$, although no longer equivalent as OS theories, are nevertheless *inductively equivalent* in the sense that their initial algebras $\mathcal{T}_{\Sigma, B_1 \cup \widehat{R}_1 \cup \Delta_1}$ and $\mathcal{T}_{\Sigma, B_0 \cup \widehat{R} \cup \Delta}$ are isomorphic. Indeed, we have

Lemma 1. *Under the above restrictions on the first typing of an associative operator f, the associativity equation $f(f(L,P),Q) = f(L,f(P,Q))$ is an inductive consequence of the restricted associativity equation*

$$f(f(E,NL),Q) = f(E,f(NL,Q))$$

. Likewise, under the second typing the associativity equation $f(L,f(P,Q)) = f(f(L,P),Q)$ is an inductive consequence of the restricted associativity equation

$$f(Q,f(NL,E)) = f(f(Q,NL),E))$$

.

In practice these restrictions are not too strong, since we can automatically ensure typings (1) or (2) by introducing them through a *parameterized module for lists*. Furthermore, the restriction of having only one typing of type (1) or (2) per connected component for each associative f can be relaxed to allow several such typings, provided that the corresponding sorts $\mathit{Elt} < \mathit{NeList} < \mathit{List}$ and $\mathit{Elt}' < \mathit{NeList}' < \mathit{List}'$ involved in two different typings are incomparable.

Example 10. We use our running example to illustrate the two theory transformations

$$(\Sigma, B, R) \mapsto (\Sigma, B_1, \widehat{R}_1 \cup \Delta_1) \mapsto (\Sigma, B_0, \widehat{R} \cup \Delta)$$

The first transformation, adding identity axioms as explicit equations and computing the variants of rules with respect to identities, gives us the modules:

```
fmod TR1-NATURAL is pr TRUTH-VALUE .
 sort Nat .
 op 0 : -> Nat [ctor] .
 op s : Nat -> Nat [ctor] .
 op _+_ : Nat Nat -> Nat [comm] .
 ops even odd : Nat -> Bool .
 vars N M : Nat .
 eq N + 0 = N .
```

```
 eq s(N) + s(M) = s(s(N + M)) .
 eq even(0) = true .
 eq odd(0) = false .
 eq odd(s(N)) = even(N) .
 eq even(s(N)) = odd(N) .
endfm

fmod TR1-MSET-NAT is pr TR1-NATURAL .
 sort MSet .
 subsort Nat < MSet .
 op _,_ : MSet MSet -> MSet [ctor assoc comm] .
 op null : -> MSet [ctor] .
 op card : MSet -> Nat .
 var MS : MSet .
 var N : Nat .
 var X : [MSet] .
 eq X , null = X .
 eq card(null) = 0 .
 eq card(N)= s(0) + card(null) .
 eq card(N,MS)= s(0) + card(MS) .
endfm

fmod TR1-LIST-MSET-NAT is pr TR1-MSET-NAT .
 sorts List NeList .
 subsorts MSet < NeList < List .
 op nil : -> List [ctor] .
 op _;_ : List List -> List [assoc] .
 op _;_ : MSet NeList -> NeList [ctor assoc] .
 op U : List -> MSet .
 var MS : MSet .
 var NL : NeList .
 var L : List.
 var Y : [List] .
 eq Y ; nil = Y .
 eq nil ; Y = Y .
 eq U(nil) = null .
 eq U(MS) = MS .
 eq U(MS ; NL) = MS, U(NL) .
endfm
```

The way the variants of an equation with respect to the identities modulo the C and AC axioms are computed can be illustrated by the equation `card(N,MS)= s(0) + card(MS)` in the original module `MSET-NAT`. Since the variable `MS` could collapse by instantiating it to the identity element `null`, the equation's left-hand side has two most general variants: (i) itself, so that the original equation is kept, and (ii) the term `card(N)`, leading to the new variant equation `card(N)= s(0) + card(null)` added to `TR1-MSET-NAT`. The result of the second stage of the theory transformation, denoted above as $(\Sigma, B_1, \widehat{R}_1 \cup \Delta_1) \mapsto (\Sigma, B_0, \widehat{R} \cup \Delta)$, has already been described in detail in Examples 7 and 9. Note that, since they do not involve associative but not commutative axioms, the modules `TR1-NAT` and `TR1-MSET-NAT` are not changed by the second transformation.

6 Related Work and Conclusions

Our work is related to modularity methods for confluence and/or termination of TRSs. A very good survey of the literature on such methods up to 2002 can be found in [24]. One key difference is that, to the best of our knowledge, such work does not address sorts and subsorts, nor (except for, e.g., [23, 19]) rewriting modulo axioms. Another difference is that in some cases the modularity conditions imposed are quite strong, requiring for example disjointness, which is relatively rare in practical module hierarchies. Perhaps the earliest work most closely related to ours is the work on *proper extensions* of term rewriting systems of [24] (cf. also [8] and [27]). The basic idea behind

proper extensions is that calls to functions f in right-hand sides of rewrite rules $l \to r$ where $root(l)$ and f are mutually recursive, do not involve defined function symbols from the base theory (or from the extending theory that recursively depend on functions from the base theory) in the arguments of the function call. Our notion of fair extensions of well-founded recursive theories is even more restrictive in this respect, since the arguments of calls to functions in right-hand sides have to be constructor terms if the function in question is mutually recursive with the root of the left-hand side of the rule. Note however, that the advantage of our more restrictive definition is not just its ability to deal with sorts and structural axioms, but also in our case general termination is modular instead of the weaker notion of $C_{\mathcal{E}}$-termination as for proper extensions.

Our work is also related to the hierarchical termination approach of Urbain and Marché ([29, 23]), with their notion of hierarchical extension being similar to ours of fair extension. In some ways our notion is more general, since for us function symbols can appear in both a submodule and a supermodule, but of course our incremental conditions are in other ways stronger so as to ensure termination, whereas in [29, 23] a modular approach to dependency pairs is developed. Furthermore [23] covers the AC case. There is also a rich body of related work on rewriting modulo axioms, e.g. [17, 26, 18, 2, 22, 31]. For termination modulo, related papers include, e.g., [12, 23, 9, 1].

When using well-founded recursive OS theories and fair extensions to create hierarchies of theories, one can verify important properties such as sort-decreasingness, termination, confluence and sufficient completeness incrementally. Hence, at a practical level, when developing equational programs (such as functional modules in Maude), one can follow a programming discipline ensuring that modules are well-founded recursive and module extensions are fair extensions. Sticking to this programming discipline then guarantees that the verification complexity of the properties in question grows roughly linearly with the number of distinct modules. This is a significant improvement compared to existing methods used for the verification of, e.g., termination where experiments show that in practice the verification complexity grows rapidly with increasing size of theories (see also [29]).

Obvious future work includes the mechanization of all the incremental checks described above in a tool, experimentation with such a tool, and the extension of our results to conditional and context-sensitive theories, which are also supported in Maude. Moreover, recent developments in the termination analysis of rewrite systems modulo axioms (cf. e.g. [1]) might allow us to relax the conditions in the notion of well-founded recursion, thus making our approach more widely applicable.

References

1. B. Alarcón, S. Lucas, and J. Meseguer. A dependency pair framework for A∨C-termination. In P. Ölveczky editor, Proceedings of the 8th International Workshop on Rewriting Logic and its Applications (WRLA'10), *LNCS* 6381, pages 35–51. Springer, 2010.
2. L. Bachmair and N. Dershowitz. Completion for rewriting modulo a congruence. *Theoretical Computer Science*, 67(2&3):173–201, 1989.
3. F. Baader and T. Nipkow. *Term rewriting and All That.* Cambridge University Press, 1998.
4. M. Bezem, J. Klop, and R. Vrijer, editors. *Term Rewriting Systems.* Cambridge Tracts in Theoretical Computer Science 55. Cambridge University Press, 2003.
5. P. Borovanský, C. Kirchner, H. Kirchner, and P.-E. Moreau. ELAN from a rewriting logic point of view. *Theoretical Computer Science*, 285:155–185, 2002.
6. M. Clavel, F. Durán, S. Eker, J. Meseguer, P. Lincoln, N. Martí-Oliet and C. Talcott. *All About Maude – A High-Performance Logical Framework.* LNCS 4350, 2007.
7. H. Comon-Lundh and S. Delaune. The finite variant property: how to get rid of some algebraic properties. In J. Giesl editor, Proceedings of the 16th International Conference on Rewriting Techniques and Applications (RTA'05), LNCS 3467, 294–307, 2005.
8. N. Dershowitz. Hierarchical termination. In N. Dershowitz and N. Lindenstrauss editors, Proceedings of the 4th International Workshop on Conditional and Typed Rewriting Systems (CTRS-94), *LNCS 968*, pages 89–105. Springer, 1995.
9. F. Durán, S. Lucas, and J. Meseguer. Termination modulo combinations of equational theories. In S. Ghilardi and R. Sebastiani editors, Proceedings of the 7th International Symposium on Frontiers of Combining Systems (FroCoS'09), *LNAI* 5749, pages 246–262. Springer, 2009.

10. F. Durán and J. Meseguer. A Church-Rosser checker tool for conditional order-sorted equational Maude specifications. In P. Ölveczky editor, Proceedings of the 8th International Workshop on Rewriting Logic and its Applications (WRLA'10), *LNCS* 6381, pages 69–85. Springer, 2010.
11. K. Futatsugi and R. Diaconescu. *CafeOBJ Report*. World Scientific, AMAST Series, 1998.
12. J. Giesl and D. Kapur. Dependency pairs for equational rewriting. In A. Middeldorp editor, Proceedings of the 12th International Conference on Rewriting Techniques and Applications (RTA'01), *LNCS* 2051, pages 93–108. Springer, 2001.
13. J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic termination proofs in the dependency pair framework. In U. Furbach and N. Shankar editors, Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR'06), *LNCS* 4130, pages 281–286. Springer, 2006.
14. J. Goguen and J. Meseguer. Order-sorted algebra I: Equational deduction for multiple inheritance, overloading, exceptions and partial operations. *Theoretical Computer Science*, 105:217–273, 1992.
15. J. Hendrix, J. Meseguer, and H. Ohsaki. A sufficient completeness checker for linear order-sorted specifications modulo axioms. In U. Furbach and N. Shankar editors, Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR'06), *LNCS* 4130, pages 151–155. Springer, 2006.
16. J. Hendrix, H. Ohsaki, and J. Meseguer. Sufficient completeness checking with propositional tree automata. Technical report, CS Department University of Illinois at Urbana-Champaign, 2005. `http://www.ideals.illinois.edu/handle/2142/11096`.
17. G. Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the Association for Computing Machinery*, 27:797–821, 1980. Preliminary version in *18th Symposium on Mathematical Foundations of Computer Science*, 1977.
18. J.-P. Jouannaud and H. Kirchner. Completion of a set of rules modulo a set of equations. *SIAM Journal of Computing*, 15(4):1055–1094, Nov. 1986.
19. J.-P. Jouannaud and Y. Toyama. Modular Church-Rosser modulo: the complete picture. *International Journal of Software and Informatics*, 2(1):61–75, 2008.
20. D. Kapur and G. Sivakumar. Proving associative-communicative termination using RPO-compatible orderings. In *Selected Papers from Automated Deduction in Classical and Non-Classical Logics LNCS 1761* pages 39–61, Springer 2000.
21. C. Kirchner, H. Kirchner, and J. Meseguer. Operational semantics of OBJ3. In T. Lepistö and A. Salomaa editors, Proceedings of the 15th International Colloquium on Automata, Languages and Programming (ICALP'88), *LNCS 317*, pages 287–301. Springer, 1988.
22. C. Marché. Normalised rewriting and normalised completion. In Proceedings of the 9th Annual Symposium on Logic in Computer Science (LICS'94), pages 394–403. IEEE, 1994.
23. C. Marché and X. Urbain. Modular and incremental proofs of AC-termination. *Journal of Symbolic Computation*, 38(1):873–897, 2004.
24. E. Ohlebusch. *Advanced Topics in Term Rewriting*. Springer Verlag, 2002.
25. P. Ölveczky and O. Lysne. Order-sorted termination: The unsorted way. In M. Hanus and M. Rodriguez-Artalejo editors, Proceedings of the 5th International Conference on Algebraic and Logic Programming (ALP'96), *LNCS 1139*, pages 92–106. Springer, 1996.
26. G. E. Peterson and M. E. Stickel. Complete sets of reductions for some equational theories. *Journal of the ACM*, 28(2):233–264, 1981.
27. M. R. K. K. Rao. Modular proofs for completeness of hierarchical term rewriting systems. *Theoretical Computer Science*, 151:487–512, 1995.
28. P. Schneider-Kamp, R. Thiemann, E. Annov M. Codish and J. Giesl. Proving Termination using Recursive Path Orders and SAT Solving. In B. Konev and F. Wolter editors, Proceedings of the 6th International Symposium on Frontiers of Combining Systems (FroCoS'07), *LNCS 4720*pages 267–282, 2007.
29. X. Urbain. Modular & incremental automated termination proofs. *J. Autom. Reasoning*, 32(4):315–355, 2004.
30. A. van Deursen, J. Heering, and P. Klint. *Language Prototyping: An Algebraic Specification Approach*. World Scientific, 1996.
31. P. Viry. Equational rules for rewriting logic. *Theoretical Computer Science*, 285:487–517, 2002.

A Termination of Well-founded OS Theories modulo axioms

In order to prove termination of well-founded recursive OS theories, we transform them into unsorted theories by labeling function symbols as in Definition 4. However, just labeling the rules and axioms as in Definition 5 does not yield an unsorted theory that is non-terminating whenever the sorted theory is. Hence, we need some additional and modified rules. The basic idea of our transformation is to transform a finite OS theory into an infinite unsorted theory. The unsorted theory is infinite, because we explicitly instantiate rules by terms built by AC symbols.

Definition 11 (Var_f). *$Var_f(t)$ is the set of those variables of t that occur as immediate argument of some function symbol f occurring in t. Formally, $Var_f(f(x,y)) = \{x,y\}$, $Var_f(t_1,x)) = \{x\} \cup Var_f(t_1)$ if $t_1 \notin V$, $Var_f(x,t_2)) = \{x\} \cup Var_f(t_2)$ if $t_2 \notin V$, $Var_f(f(t_1,t_2)) = Var_f(t_1) \cup Var_f(t_2)$ if $t_1 \notin V$ and $t_2 \notin V$, $Var(f)(x) = \emptyset$ and $Var_f(g(t_1,\ldots,t_k)) = Var_f(t_1) \cup \ldots \cup Var_f(t_k)$ if $g \neq f$.*

Definition 12 ($Inst_{AC}$). *Let $\mathcal{E} = (\Sigma, B_0, R)$ be an OS theory where B_0 are AC or C axioms. Then the set of instantiated rules $Inst_{AC}(l \to r)$ is given by*

$$\{l\sigma \to r\sigma \mid x_i\sigma \in \mathcal{T}(\{f\},V) \text{ if } x_i \in Var_f(l) \cup Var_f(r) \\ \text{and } f \in \Sigma_{AC}, y\sigma = y \text{ otherwise}\}.$$

Based on this instantiation we present the theory transformation from OS theories modulo AC and C axioms to unsorted theories modulo AC and C axioms that is a sound reflection.

Definition 13 (transformation). *Assume that $\mathcal{E} = (\Sigma, B_0, R)$ is an OS theory where the structural axioms B_0 are AC or C axioms. The unsorted theory $\widetilde{\mathcal{E}}$ is $(\Sigma^{os}, \widetilde{B_0}, \widetilde{R})$. $\widetilde{B_0}$ is given by $\widetilde{B_0}^{AC} \cup \widetilde{B_0}^{C}$ where*

$$\begin{aligned}\widetilde{B_0}^{AC} = \{&f_\Psi(f_\Psi(x,y),z) = f_\Psi(x,f_\Psi(y,z)),\\ &f_\Psi(x,y) = f_\Psi(y,x) \mid\\ &f\colon A\,B \to A \in \Sigma_{AC}, \Psi = \{s_1,\ldots,s_n\},\\ &s_i \leq A \text{ for all } 1 \leq i \leq n\}\end{aligned}$$

and

$$\begin{aligned}\widetilde{B_0}^{C} = \{&f_{[C,D]}(x,y) = f_{[C,D]}(y,x) \mid f\colon A\,B \to G \in \Sigma_C,\\ &C \leq A, D \leq B\}\end{aligned}$$

$\widetilde{R}$ is given by $\widetilde{R}_R \cup \widetilde{R}_j$[11] where $\widetilde{R}_R$ is

$$\begin{aligned}\{\overline{l'\theta} \to \overline{r'\theta} \mid &l \to r \in R,\ \theta \text{ a specialization},\\ &l' \to r' \in Inst_{AC}(f(l,x) \to f(r,x)),\\ &x \notin Var(l) \cup Var(r), f \in \Sigma_{AC}\} \cup\\ \{\overline{l'\theta} \to \overline{r'\theta} \mid &l \to r \in R, l' \to r' \in Inst_{AC}(l \to r),\\ &\theta \text{ a specialization}\}.\end{aligned}$$

[11] The R subscript refers to the rules R of $\mathcal{E}$ and the j subscript stands for "jump" because the rules model jumps from sorts to subsorts (cf. [25][Definition 4]).

and $\widetilde{R}_j$ *is*

$$\begin{aligned}
&\{\overline{l\theta} \to \overline{r\theta} \mid l \to r \in Inst_{AC}(f(x_1,x_2) \to f(x_1,x_2)),\\
&\qquad f \in \Sigma_{AC},\ \theta \text{ a specialization}, root(\overline{l\theta}) = f_\Psi,\\
&\qquad root(\overline{r\theta}) = f_{\Psi'}, \Psi >^{mul} \Psi'\} \cup\\
&\{\overline{l\theta} \to \overline{r\theta} \mid f \in \Sigma_C, l = f(x,y), r = f(x,y),\\
&\qquad \theta \text{ a specialization}, root(\overline{l\theta}) = f_{[A,B]},\\
&\qquad root(\overline{r\theta}) = f_{[A',B']}, (A,B \gg A',B')\} \cup\\
&\{\overline{l\theta} \to \overline{r\theta} \mid f \in \Sigma \setminus (\Sigma_C \cup \Sigma_{AC}), l = f(x_1,\ldots,x_n),\\
&\qquad r = f(x_1,\ldots,x_n),\ \theta \text{ a specialization},\\
&\qquad root(\overline{l\theta}) = f_{A_1,\ldots,A_n}, root(\overline{r\theta}) = f_{A'_1,\ldots,A'_n},\\
&\qquad (A_1,\ldots,A_n \gg A'_1,\ldots,A'_n)\}.
\end{aligned}$$

Note that in the theory $\widetilde{\mathcal{E}} = (\Sigma^{os}, \widetilde{B_0}, \widetilde{R})$ obtained from $\mathcal{E} = (\Sigma, A, E)$, $\Sigma^{os}, \widetilde{B_0}$ as well as $\widetilde{R}$ are countably infinite in general if Σ, B_0 and R are finite. The following lemma states the important fact that two B_0-equal terms are also $\widetilde{B_0}$-equal after they are labeled.

Lemma 2. *Let* $\mathcal{E} = (\Sigma, B_0, R)$ *be an OS theory where the structural axioms* B_0 *are AC or C axioms. If* $t =_{B_0} t'$*, then* $\overline{t} =_{\widetilde{B_0}} \overline{t'}$*.*

Proof. We prove the result by showing that $t =_{B_0} t'$ implies $\overline{t} =_{\widetilde{B_0}} \overline{t'}$ if t' is obtained from t by substituting an instance of one side of a single equation from B_0 by the corresponding instance of the other side. The general case follows by induction on the number of equality steps. We distinguish two cases, depending on whether the function symbol for which an equation is used is C or AC. First, assume a commutativity equation is used say for f and let $f \in \Sigma_C$. Let $t = C[f(t_1,t_2)]$ and $t' = C[f(t_2,t_1)]$. $\overline{t} = \overline{C}[f_{[A,B]}(\overline{t_1},\overline{t_2})]$ according to Definition 4 (where $ls(t_1) = A$ and $ls(t_2) = B$). According to Definition 13 there is an equation $f_{[A,B]}(x,y) = f_{[A,B]}(y,x) \in \widetilde{B_0}$. Hence, $\overline{t} =_{\widetilde{B_0}} \overline{C}[f_{[A,B]}(\overline{t_2},\overline{t_1})] = \overline{t'}$.

Second, assume that an axiom for an AC symbol was used and the used equation was an associativity equation. Let $t = C[f(f(t_1,t_2),t_3)]_p$ and $t' = C[f(t_1,f(t_2,t_3))]_p$. We write t as $C'[f(t'_1,t'_2)]_q$ where $q \le p$, $root(C|_o) = f$ for each $q \le o \le p$ and either $q = \epsilon$ or the function symbol right above q is not an f (i.e. $root(C'|_{q'}) \neq f$ if $q = q'.i$ and $i \in \mathbb{N}$). Then

$$\begin{aligned}
\overline{t} &= \overline{C'}[f_\Psi(\lambda(t'_1,f,\Psi),\lambda(t'_2,f,\Psi))]_q\\
&= C''[f_\Psi(f_\Psi(\lambda(t_1,f,\Psi),\lambda(t_2,f,\Psi)),\lambda(t_3,f,\Psi))]_p
\end{aligned}$$

which is $\widetilde{B_0}$-equal to

$$C''[f_\Psi(\lambda(t_1,f,\Psi),f_\Psi(\lambda(t_2,f,\Psi),\lambda(t_3,f,\Psi)))]_p$$

because there is an equation

$$f_\Psi(f_\Psi(x,y),z) = f_\Psi(x,f_\Psi(y,z)) \in \widetilde{B_0}$$

according to Definition 13. Finally, we have

$$C''[f_\Psi(\lambda(t_1,f,\Psi),f_\Psi(\lambda(t_2,f,\Psi),\lambda(t_3,f,\Psi)))]_p = \overline{t'}$$

because $flat(t|_q,f) = flat(t'|_q,f)$.

The cases where an associativity axiom is applied in the other direction and where a commutativity axiom is applied to an AC symbol are analogous.

Lemma 3. *Let* $\mathcal{E} = (\Sigma, B_0, R)$ *be a sort-decreasing OS theory where the structural axioms* B_0 *are AC or C axioms. Let* s *and* $s[t]_p$ *be terms of* $\mathcal{T}(\Sigma,V)$*, such that* $ls(s|_p) \ge ls(t)$ *and either* $p = \epsilon$ *or* $root(s|_{p'}) \neq root(t)$ *where* p' *is given by* $p = p'.i$ *for some* $i \in \mathbb{N}$*. Then,* $\overline{s}[\overline{t}]_p \to^*_{\widetilde{R}/\widetilde{B_0}} \overline{s[t]_p}$*.*

Proof. Note that since $root(s|_{p'}) \neq root(t)$ we have $\overline{s[t]_p}|_p = \overline{t}$ (cf. Definition 4). Hence, what needs to be done to derive $\overline{s[t]_p}$ from $\overline{s}[\overline{t}]_p$, is to modify the sort-labels of function symbols above (or potentially parallel to) p in $\overline{s}[\overline{t}]_p$.

To prove the result we use induction on the number of positions $o < p$ for which either $root(s|_0)$ is not an AC symbol or $root(s|_o) \neq root(s|_{o.i})$ where $i \in \mathbb{N}$ is uniquely determined by $o.i \leq p$.[12] If the number of these positions is 0, then $p = \epsilon$ and we are done because $\overline{s[t]_p}|_p = \overline{t}$. Otherwise, we distinguish two cases, depending on whether $root(s)$ is an AC symbol, or a C or free symbol.

If it is a free or C (but not AC) symbol, then we write $s[t]_p$ as $f(t_1, \ldots, s_j[t]_q, \ldots, t_n)$ where j and q are determined by $j.q = p$ and $j \in \mathbb{N}$. The induction hypothesis yields $\overline{s_j}[\overline{t}]_q \to^*_{\widetilde{R}/\widetilde{B_0}} \overline{s_j[t]_q}$. As $ls(s|_p) \geq ls(t)$ and by monotonicity of Σ we have $ls(s_j) \geq ls(s_j[t]_q)$. If $ls(s_j) = ls(s_j[t]_q)$, we have

$$\overline{s}[\overline{t}]_p = \overline{s}[\overline{s_j}[\overline{t}]_q]_j \overset{IH}{\to}^*_{\widetilde{R}/\widetilde{B_0}} \overline{s}[\overline{s_j[t]_q}]_j = \overline{s[s_j[t]_q]_j} = \overline{s[t]_p}.$$

Otherwise, if $ls(s_j) > ls(s_j[t]_q)$, we have

$$ls(s|_1), \ldots, ls(s|_j), \ldots, ls(s|_n) \succcurlyeq \\ ls(s|_1), \ldots, ls(s|_j[t]_q), \ldots, ls(s|_n)$$

and thus there exists a rule

$$f_{ls(s|_1),\ldots,ls(s|_j),\ldots,ls(s|_n)}(x_1, \ldots, x_n) \to \\ f_{ls(s|_1),\ldots,ls(s|_j[t]_q),\ldots,ls(s|_n)}(x_1, \ldots, x_n)$$

in case $root(s)$ is a free symbol, and a rule

$$f_{[ls(s|_1),ls(s|_2)]}(x, y) \to f_{[ls(s|_1[t]_q),ls(s|_2)]}(x, y)$$

in case $root(s)$ is commutative (assuming w.l.o.g. that $j = 1$ in this case) in $\widetilde{R}$ yielding

$$\overline{s}[\overline{t}]_p = \overline{s}[\overline{s_j}[\overline{t}]_q]_j \overset{IH}{\to}^*_{\widetilde{R}/\widetilde{B_0}} \overline{s}[\overline{s_j[t]_q}]_j \to_{\widetilde{R}} \overline{s[s_j[t]_q]_j} = \overline{s[t]_p}.$$

Next assume $root(s)$ is an AC symbol. We write

$$s[t]_p = C[t_1, \ldots, t_k[t]_q, \ldots, t_m]_{p_1,\ldots,p_k,\ldots,p_m}$$

where $C[x_1, \ldots, x_m]_{p_1,\ldots,p_m} \in \mathcal{T}(\{root(s)\}, V)$, $root(t_i) \neq root(s)$ for all $1 \leq i \leq m$ and $p_k.q = p$. The induction hypothesis yields $\overline{t_k}[\overline{t}]_q \to^*_{\widetilde{R}/\widetilde{B_0}} \overline{t_k[t]_q}$. As $ls(s|_p) \geq ls(t)$ and by monotonicity of Σ we have $ls(t_k) \geq ls(t_k[t]_q)$. If $ls(t_k) = ls(t_k[t]_q)$, we have

$$\overline{s}[\overline{t}]_p = \overline{s}[\overline{t_k}[\overline{t}]_q]_{p_k} \overset{IH}{\to}^*_{\widetilde{R}/\widetilde{B_0}} \overline{s}[\overline{t_k[t]_q}]_{p_k} = \overline{s[t_k[t]_q]_{p_k}} = \overline{s[t]_p}$$

Otherwise, we have $flat(s, root(s)) = root(s)(t_1, \ldots, t_m)$ and $root(\overline{flat(s, root(s))}) = f_\Psi$ where

$$\Psi = \{ls(t_1), \ldots, ls(t_k), \ldots, ls(t_m)\} >^{mul} \\ \{ls(t_1), \ldots, ls(t_k[t]_q), \ldots, ls(t_m)\} = \Psi',$$

and $root(\overline{flat(s[t]_p}, root(s))) = f_{\Psi'}$. Thus, there is a rule in $\widetilde{R}$ that is a labeled version of

$$C[x_1, \ldots, x_m] \to C[x_1, \ldots, x_m],$$

such that the (single) function symbol f occurring in the left-hand side is consistently labeled by Ψ and the function symbol on the right-hand side is consistently labeled by Ψ'. Hence, we obtain

$$\overline{s}[\overline{t}]_p = \overline{s}[\overline{t_k}[\overline{t}]_q]_{p_k} \overset{IH}{\to}^*_{\widetilde{R}/\widetilde{B_0}} \overline{s}[\overline{t_k[t]_q}]_{p_k} \to_{\widetilde{R}} \overline{s[t_k[t]_q]_{p_k}} = \overline{s[t]_p}$$

[12] The reason we cannot just use the plain number of such positions for the induction is that changing the labels of nested AC symbols has to be done in one step.

Lemma 4 (sound reflection property). *Let $\mathcal{E} = (\Sigma, B_0, R)$ be a left-linear sort-decreasing OS theory where the structural axioms B_0 are AC or C axioms. If $s \to_{R/B_0} t$, then $\overline{s} \to^{+}_{\widetilde{R}/\widetilde{B_0}} \overline{t}$.*

Proof. We write $s \to_{R/B_0} t$ as $s =_{B_0} s' \xrightarrow{p}_R t' =_{B_0} t$. Then $s'|_p = l\sigma$ for some $l \to r \in R$ and some substitution σ. By Lemma 2 we have $\overline{s} =_{\widetilde{B_0}} \overline{s'}$. We distinguish two cases depending on whether the function symbol immediately above p in s' (i.e. at position p' given by $p'.i = p$ for some $i \in \mathbb{N}$) is an AC symbol or not.

If this symbol is not an AC symbol (or p is the root position), then $\overline{s'} = \overline{s'[\overline{s'|_p}]_p}$. We inspect $s'|_p = l\sigma$ for some rule $l \to r \in R$. It can be written as $l\theta\sigma'$, where $x\theta \in \mathcal{T}(\{f\}, V)$ whenever $x \in Var_f(l) \cup Var_f(r)$ and f is an AC symbol (cf. Definition 12) and $root(x\sigma') \neq f$ whenever $x \in Var_f(l\theta) \cup Var_f(r)$. Then we have, $l\theta \to r\theta \in Inst_{AC}(l \to r)$ by Definition 12. Moreover, $\overline{l\theta\sigma'} = \overline{l\theta}\,\overline{\sigma'}$. Let τ be a specialization for the variables of $l\theta$ given by $x\colon S\tau = x\colon S'$ if $ls(x\sigma) = S'$. Then, by Definition 13, there is a rule $\overline{l\theta\tau} \to \overline{l\theta\tau} \in \widetilde{R}$. With this rule we have

$$\begin{aligned}\overline{s'} &= \overline{s'[\overline{s'|_p}]_p} = \overline{s'[\overline{l\sigma}]_p} = \overline{s'[\overline{l\theta\sigma'}]_p} = \\ &= \overline{s'[\overline{l\theta}\,\overline{\sigma'}]_p} \to_{\widetilde{R}} \overline{s'[\overline{r\theta}\,\overline{\sigma'}]_p} = \overline{s'[\overline{t'|_p}]_p}.\end{aligned}$$

By sort decreasingness of $\mathcal{E}$ we have $ls(t'|_p) \leq ls(s'|_p)$, hence Lemmas 3 and 2 yield $\overline{s'[\overline{t'|_p}]_p} \to^{*}_{\widetilde{R}/\widetilde{B_0}} \overline{t}$.

Now consider the case where the function symbol immediately above p is an AC symbol, say f. Then s' can be written as $s'[s'|_q]_q$ where $q < p$, the function symbol at the position right above q in s' is not f (or $q = \epsilon$) and $root(s'|_o = f)$ for all $q \leq o < p$. Then we have $\overline{s'} = \overline{s'[\overline{s'|_q}]_q}$. By associativity and commutativity of f and Lemma 2, there is a term $\overline{w} =_{B_0} \overline{s'|_q}$ such that $root(\overline{w}) = f$ and $\overline{w}|_1 = \overline{s'|_p}$. We can write w (i.e. $erase(\overline{w})$) as $f(l, x)\theta\sigma'$ where $l\theta\sigma' = s|_p$, $l \to r \in R$, $x \notin Var(f)$, $y\theta \in \mathcal{T}(\{f\}, V)$ whenever $y \in Var_f(f(l, x)) \cup Var_f(f(r, x))$ and $root(y\sigma') \neq f$ whenever $y \in Var_f(f(l, x)\theta) \cup Var_f(f(r, x)\theta)$. Moreover, let τ be a specialization for the variables of $f(l, x)\theta$ given by $x\colon S\tau = x\colon S'$ if $ls(x\sigma) = S'$. Then, by Definition 13, there is a rule $\overline{f(l, x)\theta\tau} \to \overline{f(r, x)\theta\tau} \in \widetilde{R}$. Thus, we have

$$\begin{aligned}\overline{s'} &= \overline{s'[\overline{s'|_q}]_q} =_{\widetilde{A}} \overline{s'[\overline{w}]_q} = \overline{s'[\overline{f(l, x)\theta\sigma'}]_q} = \\ &= \overline{s[\overline{f(l, x)\theta}\,\overline{\sigma'}]_q} \to_{\widetilde{R}} \overline{s'[\overline{f(r, x)\theta}\,\overline{\sigma'}]_q} =_{\widetilde{B_0}} \overline{s'[\overline{t'|_q}]_q}.\end{aligned}$$

Note that in the last equality we need to use $\widetilde{B_0}$ to inversely apply the axioms used to derive w from $s|_q$, which is possible e.g. by Lemma 2. By sort decreasingness of $\mathcal{E}$ and monotonicity of Σ, we have $ls(t'|_q) \leq ls(s'|_q)$, hence Lemmas 3 and 2 yield $\overline{s'[\overline{t'|_q}]_q} \to^{*}_{\widetilde{R}/\widetilde{B_0}} \overline{t}$.

Corollary 2. *Let $\mathcal{E} = (\Sigma, B_0, R)$ be a left-linear sort-decreasing OS theory where the structural axioms B_0 are AC or C axioms. If $\widetilde{\mathcal{E}}$ is $\widetilde{B_0}$-terminating, then $\mathcal{E}$ is B_0-terminating.*

Next we are going to prove that well-founded recursive OS theories are terminating, by showing that for a given well-founded recursive theory $\mathcal{E}$, $\widetilde{\mathcal{E}}$ is compatible with an $ACRPO$ as introduced by Kapur et. al. ([20]). Note that the presence of function symbols that are commutative but not associative is not a problem, since arguments of non associative function symbols are compared as multisets and hence terms $f(x, y)$ and $f(y, x)$ are always equivalent w.r.t. to an $ACRPO$ $\succ_{ac}$.

First, we prove some general properties of an $ACRPO$ $\succ_{ac}$, which are then used in Theorem 1 below. We show that if we have $s \rhd t$ for two terms s and t of a particular shape, then for each pair $\langle a, b\rangle \in cands(t, f)$ there is a pair $\langle a', b'\rangle \in cands(s, f)$ with $\langle a', b'\rangle \succ_c \langle a, b\rangle$ (cf. [20][Definitions 12 and 17]). Note that this result is stronger than $cands(s, f) \succ_c^{mul} cands(t, f)$ which is a consequence of Lemma 1 in Kapur et. al. ([20][Lemma 1]). In the following, whenever we mention an $ACRPO$ $\succ_{ac}$ we mean $\succ_{ac}$ as defined in Definition 17 by Kapur et. al. ([20][Definition 17]).

Lemma 5. *Let Σ be an unsorted signature and let $\succ_{ac}$ be an $ACRPO$ w.r.t. a precedence $>$ on function symbols. If $s \rhd t$ for terms s and t, where $root(s|_p) < f$ and $root(t|_q) < f$ for all $p \in Pos_\Sigma(s)$ and all $q \in Pos_\Sigma(t)$, then for each pair $\langle a, b\rangle \in cands(t, f)$, there is a pair $\langle a', b'\rangle \in cands(s, f)$ such that $\langle a', b'\rangle \succ_c \langle a, b\rangle$ (where $\succ_c$ is defined as in [20][Definition 17]).*

Proof. If s is a ground term, then $cands(s,f) = \{\langle\{a\},\{\langle\{a\},s\rangle\}\rangle\}$. In that case t is a ground term as well and we have $cands(t,f) = \{\langle\{a\},\{\langle\{a\},t\rangle\}\rangle\}$. Hence, $\langle\{a\},\{\langle\{a\},s\rangle\}\rangle \succ_c \langle\{a\},\{\langle\{a\},t\rangle\}\rangle$ proves the result.

Otherwise, we have

$$cands(s,f) = \{\langle\{y\},\{\langle\{y\},s\rangle\}\rangle \mid y \in Var(s)\},$$

where by $Var(s)$ we mean the multiset of variables in s and $cands(s,f)$ is a multiset, too. We have three possibilities for $cands(t,f)$:

First,

$$cands(t,f) = \{\langle\{x\},\emptyset\rangle\}$$

if t is the variable x. In that case there is a pair $\langle\{x\},\{\langle\{x\},s\rangle\}\rangle$ in $cands(s,f)$ and we have

$$\langle\{x\},\{\langle\{x\},s\rangle\}\rangle \succ_c \langle\{x\},\emptyset\rangle$$

(regardless of the used abstraction).

Second,

$$cands(t,f) = \{\langle\{a\},\{\langle\{a\},t\rangle\}\rangle\}$$

if t is a ground term. In that case for every pair $\langle\{x\},\{\langle\{x\},s\rangle\}\rangle \in cands(s,f)$ we have

$$\langle\{x\},\{\langle\{x\},s\rangle\}\rangle \succ_c \langle\{a\},\{\langle\{a\},t\rangle\}\rangle$$

.

Finally, if t is not a variable and not a ground term, we have

$$cands(t,f) = \{\langle\{y\},\{\langle\{y\},t\rangle\}\rangle \mid y \in Var(t)\}$$

As $s \rhd t$, $Var(t) \subseteq Var(s)$ and thus for every pair $\langle\{x\},\{\langle\{x\},t\rangle\}\rangle \in cands(t,f)$ there is a pair $\langle\{x\},\{\langle\{x\},s\rangle\}\rangle \in cands(s,f)$ such that we have

$$\langle\{x\},\{\langle\{x\},t\rangle\}\rangle \succ_c \langle\{x\},\{\langle\{x\},s\rangle\}\rangle$$

according to [20][Definition 17] (again regardless of the used abstraction; note that $s \rhd t$ implies $s \succ_{ac} t$, since $\succ_{ac}$ is a simplification ordering).

The next lemma states a property about the extension $\succ^{tup}$ of an ordering $\succ$ defined in Definition 8.

Lemma 6. *Let $S \succ^{tup} T$ for multisets S, T and some ordering $\succ$ and function ϕ. If $S = S' \cup (S \cap T)$ and $T = T' \cup (S \cap T)$ and $\phi(s') = t'$ for $s' \in S'$ and $t' \in T'$, then $S \setminus \{s'\} \succ^{tup} T \setminus \{t'\}$ or $S \setminus \{s'\} = T \setminus \{t'\}$.*

Proof. By the definition of $\succ^{tup}$.

The following lemma is the technical key to finally proving AC-termination of rewrite derivations w.r.t. a well-founded OS theory.

Lemma 7. *Let Σ be an unsorted signature and $\succ_{ac}$ be an ACRPO on $\mathcal{T}(\Sigma, V)$ where $<$ is the used precedence. Moreover, let*

$$Cands_1 = \{\langle c_1 \cup \ldots \cup c_n, p_1 \cup \ldots \cup p_n\rangle \mid \langle c_1, p_1\rangle \in cands(s_1,f), \ldots, \langle c_n, p_n\rangle \in cands(s_n,f)\}$$

and

$$Cands_2 = \{\langle c'_1 \cup \ldots \cup c'_m, p'_1 \cup \ldots \cup p'_m\rangle \mid \langle c'_1, p'_1\rangle \in cands(t_1,f), \ldots, \langle c'_m, p'_m\rangle \in cands(t_m,f)\}$$

for multisets $\{s_1,\ldots,s_n\}$ and $\{t_1,\ldots,t_m\}$ of terms from $\mathcal{T}(\Sigma^{<f}, V)$ each having pairwise disjoint variables and satisfying $\{s_1,\ldots,s_n\} \rhd^{tup} \{t_1,\ldots,t_m\}$.[13] *Then $Cands_1 \succ_c^{mul} Cands_2$.*

[13] $\Sigma^{<f}$ is the set of function symbols that are smaller than f in the precedence $<$.

Proof. $\{s_1, \ldots, s_n\} \rhd^{tup} \{t_1, \ldots, t_m\}$ means that

$$\{s_1, \ldots, s_n\} = \{s_{i_1}, \ldots, s_{i_{n'}}\} \cup \Pi$$
$$\{t_1, \ldots, t_m\} = \{t_{j_1}, \ldots, s_{j_{m'}}\} \cup \Pi$$

where $\Pi = \{s_1, \ldots, s_n\} \cap \{t_1, \ldots, t_m\}$, $n' \neq 0$ and there exists a (possibly partial) surjective function $\varphi : \{s_{i_1}, \ldots, s_{i_{n'}}\} \to \{t_{j_1}, \ldots, s_{j_{m'}}\}$ such that $\varphi(s) = t$ implies $s \rhd t$.

We prove $Cands_1 \succ_c^{mul} Cands_2$ by induction on m'. First, assume $m' = 0$, i.e. $\{t_1, \ldots, t_m\} = \Pi \subset \{s_1, \ldots, s_n\}$ (the strict subset inclusion is a consequence of $n' \neq 0$). This means that for each

$$\pi' = \langle c_1 \cup \ldots \cup c_m, p_1 \cup \ldots \cup p_m \rangle \in Cands_2$$

there is a

$$\pi = \langle c_1 \cup \ldots \cup c_m \cup c_{m+1} \cup \ldots \cup c_n, p_1 \cup \ldots \cup p_m \cup p_{m+1} \cup \ldots \cup p_n \rangle$$

from $Cands_1$, such that $\pi \succ_c \pi'$ because $c_1 \cup \ldots \cup c_m \cup c_{m+1} \cup \ldots \cup c_n \supset c_1 \cup \ldots \cup c_m$ and thus $c_1 \cup \ldots \cup c_m \cup c_{m+1} \cup \ldots \cup c_n \succ_{ac}^{mul} c_1 \cup \ldots \cup c_m$. Hence, we obtain $Cands_1 \succ_c^{mul} Cands_2$.

Now assume $m' > 0$. There exists a term $s' \in \{s_{i_1}, \ldots, s_{i_{n'}}\}$, such that $\varphi(s') = t'$ for some $t' \in \{t_{j_1}, \ldots, t_{j_{m'}}\}$ and such that, additionally, $s'' \not\rhd s'$ for all $s'' \in \{s_{i_1}, \ldots, s_{i_{n'}}\}$ for which $\varphi(s'')$ is defined (i.e. we choose a maximal element s' (w.r.t $\rhd$) from $\varphi^{-1}(\{t_{j_1}, \ldots, t_{j_{m'}}\})$). By Lemma 6, the induction hypothesis is applicable, yielding

$$Cands'_1 \succ_c^{mul} Cands'_2 \tag{1}$$

or

$$Cands'_1 = Cands'_2 \tag{2}$$

where $Cands'_1$ is

$$\{\langle c_{k_1} \cup \ldots \cup c_{k_{n-1}}, p_{k_1} \cup \ldots \cup p_{k_{n-1}} \rangle \mid \langle c_{k_1}, p_{k_1} \rangle \in cands(s_{k_1}, f), \ldots, \langle c_{k_{n-1}}, p_{k_{n-1}} \rangle \in cands(s_{k_{n-1}}, f)\},$$

$Cands'_2$ is

$$\{\langle c'_{k'_1} \cup \ldots \cup c'_{k'_{m-1}}, p'_{k'_1} \cup \ldots \cup p'_{k'_{m-1}} \rangle \mid \langle c'_{k'_1}, p'_{k'_1} \rangle \in cands(t_{k'_1}, f), \ldots, \langle c'_{k'_{m-1}}, p'_{k'_{m-1}} \rangle \in cands(t_{k'_{m-1}}, f)\}$$

and

$$\{s_{k_1}, \ldots, s_{k_{n-1}}\} = \{s_1, \ldots, s_n\} \setminus \{s'\}$$

resp.

$$\{t_{k'_1}, \ldots, t_{k'_{m-1}}\} = \{t_1, \ldots, t_m\} \setminus \{t'\}.$$

With this notation we can write $Cands_1$ and $Cands_2$ in the following way.

$$Cands_1 = \{\langle a_1 \cup a_2, b_1 \cup b_2 \rangle \mid \langle a_1, b_1 \rangle \in Cands'_1, \langle a_2, b_2 \rangle \in cands(s', f)\}$$

and

$$Cands_2 = \{\langle a'_1 \cup a'_2, b'_1 \cup b'_2 \rangle \mid \langle a'_1, b'_1 \rangle \in Cands'_2, \langle a'_2, b'_2 \rangle \in cands(t', f)\}.$$

We are going to show that for every pair α' from $Cands_2$ there exists a pair α from $Cands_1$ with $\alpha \succ_c \alpha'$ which concludes the proof as it implies $Cands_1 \succ_c^{mul} Cands_2$. So consider some arbitrary pair $\langle a'_1 \cup a'_2, b'_1 \cup b'_2 \rangle$ where $\langle a'_1, b'_1 \rangle \in Cands'_2$ and

$\langle a'_2, b'_2 \rangle \in cands(t', f)$. Now we distinguish several cases, depending on the kind of pairs contained in $Cands'_1$ and $cands(s', f)$.

First, assume there exists a pair $\langle a_2, b_2\rangle \in cands(s', f)$ with $a_2 \succ_{ac}^{mul} a_2'$. By our induction hypothesis, there must be a pair $\langle a_1, b_1\rangle \in Cands_1'$ with $a_1 = a_1'$ or $a_1 \succ_{ac}^{mul} a_1'$ (because otherwise $Cands_1' \not\succeq_c^{mul} Cands_2'$) and thus $a_1 \cup a_2 \succ_c^{mul} a_1' \cup a_2'$, yielding $\langle a_1 \cup a_2, b_1 \cup b_2\rangle \succ_c \langle a_1' \cup a_2', b_1' \cup b_2'\rangle$. Second, assume there exists a pair $\langle a_1, b_1\rangle \in Cands_1'$ with $a_1 \succ_{ac}^{mul} a_1'$. Then, by Lemma 5, there must be a pair $\langle a_2, b_2\rangle \in cands(s', f)$ such that $a_2 = a_2'$ or $a_2 \succ_{ac}^{mul} a_2'$. Thus, we have $a_1 \cup a_2 \succ_{ac}^{mul} a_1' \cup a_2'$, yielding $\langle a_1 \cup a_2, b_1 \cup b_2\rangle \succ_c \langle a_1' \cup a_2', b_1' \cup b_2'\rangle$. Third, assume there is no pair $\langle a_2, b_2\rangle \in cands(s', f)$ with $a_2 \succ_{ac}^{mul} a_2'$ and no pair $\langle a_1, b_1\rangle \in Cands_1'$ with $a_1 \succ_{ac}^{mul} a_1'$. The rest of the proof is dedicated to deal with this final case.

By the induction hypothesis there exists a pair $\langle a_1, b_1\rangle \in Cands_1'$ such that either $\langle a_1, b_1\rangle = \langle a_1', b_1'\rangle$ or $\langle a_1, b_1\rangle \succ_c \langle a_1', b_1'\rangle$. We deal with these two possibilities separately. Assume $\langle a_1, b_1\rangle = \langle a_1', b_1'\rangle$ first. We know that there exists a pair $\langle a_2, b_2\rangle \in cands(s', f)$ with $\langle a_2, b_2\rangle \succ_c \langle a_2', b_2'\rangle$ by Lemma 5. Moreover, as $a_2 \not\succ_{ac}^{mul} a_2'$, we have $a_2 = a_2'$ according to [20][Definition 17]. Hence, according to [20][Definition 17] $\langle a_2, b_2\rangle \succ_c \langle a_2', b_2'\rangle$ means that $b_2 \setminus b_2' \neq \emptyset$ and for each pair $\alpha' \in b_2' \setminus b_2$, there is a pair $\alpha \in b_2 \setminus b_2'$, with $\alpha \succ_p \alpha'$. The ordering $\succ_p$ is the ordering $\succ$ of [20][Definition 17].

We have $a_1 \cup a_2 = a_1' \cup a_2'$ and $b_1 \cup b_2 \setminus b_1' \cup b_2' \neq \emptyset$, because $b_1 = b_1'$. Moreover, for every pair $\alpha' \in b_1' \cup b_2' \setminus b_1 \cup b_2$ there is a pair $\alpha \in b_1 \cup b_2 \setminus b_1' \cup b_2'$ such that $\alpha \succ_p \alpha'$, because $b_1' \cup b_2' \setminus b_1 \cup b_2 = b_2' \setminus b_2$ and $b_1 \cup b_2 \setminus b_1' \cup b_2' = b_2 \setminus b_2'$, as $b_1 = b_1'$. Thus, we have $\langle a_1 \cup a_2, b_1 \cup b_2\rangle \succ_c \langle a_1' \cup a_2', b_1' \cup b_2'\rangle$.

Second, assume $\langle a_1, b_1\rangle \succ_c \langle a_1', b_1'\rangle$. Since $a_1 \not\succ_{ac}^{mul} a_1'$ we have $a_1 = a_1'$. Hence, $\langle a_1, b_1\rangle \succ_c \langle a_1', b_1'\rangle$ means that $b_1 \setminus b_1' \neq \emptyset$ and for every pair $\alpha' \in b_1' \setminus b_1$, there exists a pair $\alpha \in b_1 \setminus b_1'$ such that $\alpha \succ_p \alpha'$. Moreover, there exists a pair $\langle a_2, b_2\rangle$ with $a_2 = a_2'$ and $\langle a_2, b_2\rangle \succ_c \langle a_2', b_2'\rangle$ according to Lemma 5.

We have $a_1 \cup a_2 = a_1' \cup a_2'$. In order to prove $\langle a_1 \cup a_2, b_1 \cup b_2\rangle \succ_c \langle a_1' \cup a_2', b_1' \cup b_2'\rangle$ we need to show $b_1 \cup b_2 \setminus b_1' \cup b_2' \neq \emptyset$ and that for every pair $\alpha' \in b_1' \cup b_2' \setminus b_1 \cup b_2$ there is a pair $\alpha \in b_1 \cup b_2 \setminus b_1' \cup b_2'$, with $\alpha \succ_p \alpha'$. We distinguish two cases to prove that

$$b_1 \cup b_2 \setminus b_1' \cup b_2' \supseteq b_2 \setminus b_2' \neq \emptyset : \tag{3}$$

If s' is a ground term, then it is a small term and thus $cands(s', f) = \{\langle\{a\}, \{\langle\{a\}, s'\rangle\}\rangle\}$ (i.e. $b_2 = \{\langle\{a\}, s'\rangle\}$). In that case t' is a small term as well and $cands(t', f) = \{\langle\{a\}, \{\langle\{a\}, t'\rangle\}\rangle\}$. Now because of the particular choice of s', the number of occurrences of s' in $\{t_{k_1'}, \ldots, t_{k_{m-1}'}\}$ must be less or equal than the occurrences of s' in $\{s_{k_1}, \ldots, s_{k_{n-1}}\}$. Hence, the number of pairs $\langle\{a\}, s'\rangle$ in b_1' must be less or equal than the number of these pairs in b_1. Now since the number of $\langle\{a\}, s'\rangle$ pairs in $b_1 \cup b_2$ equals the number of these pairs in $b_1 + 1$, while the number of these pairs in $b_1' \cup b_2'$ equals that in b_1', we have strictly more occurrences of $\langle\{a\}, s'\rangle$ in $b_1 \cup b_2$ than in $b_1' \cup b_2'$ and thus $b_1 \cup b_2 \setminus b_1' \cup b_2' \supseteq \{\langle\{a\}, s'\rangle\} = b_2 \setminus b_2'$.

Otherwise, s' is not a ground term. Then pairs in b_2 are of the shape $\langle\{x\}, s'\rangle$ for some variable x of s'. No term from $\{t_{k_1'}, \ldots, t_{k_{m-1}'}\}$ can contain any variable occurring in s', since otherwise, because of $\{s_{k_1}, \ldots, s_{k_{n-1}}\} \rhd^{tup} \{t_{k_1'}, \ldots, t_{k_{m-1}'}\}$, some term of $\{s_{k_1}, \ldots, s_{k_{n-1}}\}$ would have to contain the same variables and that would be a contradiction to variable disjointness of terms of $\{s_1, \ldots, s_n\} = \{s_{k_1}, \ldots, s_{k_{n-1}}\} \cup \{s'\}$. Hence, we have $b_2 \setminus b_1' = b_2$ and thus $b_1 \cup b_2 \setminus b_1' \cup b_2' \supseteq b_2 \setminus b_2'$.

Now, we show that for every pair $\alpha' \in b_1' \cup b_2' \setminus b_1 \cup b_2$ there is a pair $\alpha \in b_1 \cup b_2 \setminus b_1' \cup b_2'$, with $\alpha \succ_p \alpha'$. We distinguish several cases: First, assume $\alpha' \in b_2' \setminus b_1 \cup b_2$. Thus, there is a pair $\alpha \in b_2 \setminus b_2'$, such that $\alpha \succ_p \alpha'$. However, since $b_2 \setminus b_2' \subseteq b_1 \cup b_2 \setminus b_1' \cup b_2'$, we have $\alpha \in b_1 \cup b_2 \setminus b_1' \cup b_2'$.

Second, assume $\alpha' \in b_1' \setminus b_1 \cup b_2$. We further distinguish two cases, depending on whether α' has the shape $\langle\{a\}, t\rangle$ (i.e. it goes back to a small term t) or $\langle\{x\}, t\rangle$ (i.e. it goes back to a non-ground term t). Assume $\alpha' = \langle\{a\}, t\rangle$ for some term t. We know that there is a pair $\alpha \in b_1 \setminus b_1'$ with $\alpha \succ_p \alpha'$. If $\alpha \in b_1 \setminus b_1' \cup b_2'$ we are done. Otherwise, $b_2' = \{\alpha\}$ and $t = t'$ (cf. [20][Definition 12]). Moreover, we have $b_2 \setminus b_2' \supseteq \{\alpha''\}$ with $\alpha'' \succ_p \alpha$ and since $b_1 \cup b_2 \setminus b_1' \cup b_2' \supseteq b_2 \setminus b_2'$ (3) we have $\alpha'' \in b_1 \cup b_2 \setminus b_1' \cup b_2'$. By transitivity of $\succ_p$ we get $\alpha'' \succ_p \alpha'$ ($\alpha'' \succ_p \alpha \succ_p \alpha'$).

Now, assume $\alpha' = \langle\{x\}, t\rangle$ for some term t and variable $x \in t$. Then, there is a pair $\alpha \in b_1 \setminus b_1'$ with $\alpha \succ_p \alpha'$. $\langle a_1, b_1\rangle \in Cands_1'$. Hence, $x \in Var(s_{k_1}) \cup \ldots \cup Var(s_{k_{n-1}})$ and thus $x \notin s'$, because the terms in $\{s_1, \ldots, s_n\} = \{s_{k_1}, \ldots, s_{k_{n-1}}\} \cup \{s'\}$ are pairwise variable disjoint. This means $x \notin t'$, as $s' \rhd t'$ and thus there is no pair $\langle p_1, p_2\rangle \in cands(t', f)$ where $\alpha \in p_2$. Hence, $\alpha \in b_1 \setminus b_1' \cup b_2'$ and $\alpha \succ_p \alpha'$.

The final lemma justifies the use of flattened terms when comparing arguments of AC function symbols in Definition 8.

Lemma 8. *Let Σ be an unsorted signature and $\succ_{ac}$ be an ACRPO on $\mathcal{T}(\Sigma, V)$ where $<$ is the used precedence. Given a term t, such that all proper subterms of $flat(t, f)$ are from $\mathcal{T}(\Sigma^{<f}, V)$, then we have $cands(t, f) = cands(flat(t, f), f)$.*

Proof. Let $t' = flat(t, f)$. We prove the result by induction on the arity of $root(t')$ (denoted $ar(root(t'))$). If this arity is 2, then $t = t'$ according to Definitions 2 and 8. Otherwise, assume we have $ar(root(t')) = k > 2$. If $root(t) \neq f$, we have $t = t'$ and the result holds trivially. So assume $t = f(t_1, t_2)$. Then, if $t' = f(t'_1, \ldots, t'_k)$ we have $flat(t_1, f) = f(t_{i'_1}, \ldots, t'_{i_{l_1}})$ and $flat(t_2, f) = f(t_{j'_1}, \ldots, t'_{j_{l_2}})$ such that $\{t'_{i_1}, \ldots, t'_{i_{l_1}}, t'_{j_1}, \ldots t'_{j_{l_2}}\} = \{t_1, \ldots, t_k\}$ (here if l_1 or l_2 is one, then $f(t'_{i_1})$ (resp. $f(t'_{j_1})$) denotes t'_{i_1} (resp. t'_{j_1})).

By Definition 2 the arity of $root(flat(t_i, f))$ is less than k for both $i \in \{1, 2\}$. Hence, the induction hypothesis applies yielding $cands(t_1, f) = cands(flat(t_1, f), f)$ which is given by

$$\{c'_{i_1} \cup \ldots \cup c'_{i_{l_1}}, p'_{i_1} \cup \ldots \cup p''_{i_{l_1}} \mid \langle c'_{i_1}, p'_{i_1}\rangle \in cands(t'_{i_1}, f), \ldots, \langle c'_{i_{l_1}}, p'_{i_{l_1}}\rangle \in cands(t'_{i_{l_1}}, f)\}$$

and $cands(t_2, f) = cands(flat(t_2, f), f)$ which is given by

$$c'_{j_1} \cup \ldots \cup c'_{j_{l_2}}, p'_{j_1} \cup \ldots \cup p''_{j_{l_2}} \mid \langle c'_{j_1}, p'_{j_1}\rangle \in cands(t'_{j_1}, f), \ldots, \langle c'_{j_{l_2}}, p'_{j_{l_2}}\rangle \in cands(t'_{j_{l_2}}, f)\}.$$

By [20][Definition 12], we have

$$cands(t, f) = \{\langle c_1 \cup c_2, p_1 \cup p_2 \mid \langle c_1, p_1\rangle \in cands(t_1, f), \langle c_2, p_2\rangle \in cands(t_2, f)\},$$

which is thus the same as

$$\begin{aligned}&\{\langle (c'_{i_1} \cup \ldots \cup c'_{i_{l_1}}) \cup (c'_{j_1} \cup \ldots \cup c'_{j_{l_2}}),\\ &(p'_{i_1} \cup \ldots \cup p'_{i_{l_1}}) \cup (p'_{j_1} \cup \ldots \cup p'_{j_{l_2}})\rangle \mid\\ &\langle c'_{i_1}, p'_{i_1}\rangle \in cands(t'_{i_1}, f), \ldots, \langle c'_{i_{l_1}}, p'_{i_{l_1}}\rangle \in cands(t'_{i_{l_1}}, f)\\ &\langle c'_{j_1}, p'_{j_1}\rangle \in cands(t'_{j_1}, f), \ldots, \langle c'_{j_{l_2}}, p'_{j_{l_2}}\rangle \in cands(t'_{j_{l_2}}, f)\}.\end{aligned}$$

By associativity and commutativity of $\cup$ (for multisets) this further equals

$$\{\langle c'_1 \cup \ldots \cup c'_k, p'_1 \cup \ldots \cup p'_k \mid \langle c_1, p_1\rangle \in cands(t'_1, f), \ldots, \langle c_k, p_k\rangle \in cands(t'_k, f)\},$$

which is $cands(t', f)$ by [20][Definition 12].

The next lemma states stability of $\blacktriangleright$ under AC-instantiations.

Lemma 9. *Let Σ be an unsorted signature and let $s, t \in \mathcal{T}(\Sigma, V)$ such that $root(s) = root(t) = f \in \Sigma_{AC}$ and θ be a specialization such that $lab(root(\overline{s\theta})) \geq^{mul} lab(root(\overline{t\theta}))$. Consider the multiset S of arguments of $root(\overline{s\theta})$ in the term $\overline{s\theta}$ as well as the multiset T of arguments of $root(\overline{t\theta})$ in the term $\overline{t\theta}$. Assume that for every variable $x \in T \setminus S$, there exists a term $s' \in S \setminus T$, such that $ls(s') > ls(x)$. Moreover, let $root(\overline{s\theta}) = f_\Psi$ and $root(\overline{t\theta}) = f_{\Psi'}$. Then for every substitution σ with $ls(x\sigma) = ls(x\theta)$ for all $x \in Dom(\sigma)$ we have that $root(\overline{s\sigma}) = f_{\widetilde{\Psi}}$ and $root(\overline{t\sigma}) = f_{\widetilde{\Psi}'}$ implies $\widetilde{\Psi} \geq^{mul} \widetilde{\Psi}'$.*

Proof. If there is no variable in $T \setminus S$, then $\widetilde{\Psi}' \setminus \widetilde{\Psi} = lab(root(\overline{t\theta})) \setminus lab(root(\overline{s\theta}))$ and thus $\widetilde{\Psi} \geq^{mul} \widetilde{\Psi}'$.

Otherwise, for each sort $u \in \widetilde{\Psi}' \setminus \widetilde{\Psi}$ there is a sort $u' > u \in \widetilde{\Psi} \setminus \widetilde{\Psi}'$ and $\widetilde{\Psi} \setminus \widetilde{\Psi}'$ is non-empty, hence $\widetilde{\Psi} >^{mul} \widetilde{\Psi}'$.

Finally, we can prove finiteness of AC-rewrite derivations w.r.t. well-founded OS theories.

THEOREM 1. *Let $\mathcal{E} = (\Sigma, B_0, R)$ be a sort-decreasing well-founded recursive OS theory where the structural axioms B_0 are either AC or C axioms. Then $\mathcal{E}$ is B_0-terminating.*

Proof. We prove that $\widetilde{\mathcal{E}} = (\Sigma^{os}, \widetilde{B_0}, \widetilde{R})$ is compatible with an $ACRPO$ as defined in [20] (which implies termination of $\mathcal{E}$ by Lemma 4). Note that there is no lexicographic comparison of arguments of non-commutative function symbol in the definition of the $ACRPO$ in [20]. However, an inspection of the proofs reveals that the results remain valid also in the presence of lexicographic comparisons of arguments of non-commutative functions. This is also claimed in [20].

Assume $\mathcal{E}$ is well-founded recursive w.r.t. to the status functions $stat$ and $stat_{ac}$. The non-strict precedence $\gtrsim$ on function symbols of Σ^{os} that we use is the smallest precedence satisfying $f > g$ if:

- $f \blacktriangleright_{\overline{\mathcal{E}}} g$, $g \not\blacktriangleright_{\overline{\mathcal{E}}} f$ and either f or g is *not* an AC symbol; and
- f and g are both AC symbols, $f \blacktriangleright_{\overline{\mathcal{E}}} g$, $g \not\blacktriangleright_{\overline{\mathcal{E}}} f$ and either $erase(g) \not\blacktriangleright_{\mathcal{E}} erase(f)$ or $stat_{ac}(f) = stat_{ac}(g) = s$; and
- f is an AC symbol and there exists a rule $l \to r \in \widetilde{R}$ such that $root(l) = f$ and $root(l|_p) = g$ for some position $p \in Pos_{\Sigma}(l)$.

Moreover, two function symbols f and g are equal if

- $f \blacktriangleright_{\overline{\mathcal{E}}} g$, $g \blacktriangleright_{\overline{\mathcal{E}}} f$; and
- f and g are both AC symbols, $erase(f) \blacktriangleright_{\mathcal{E}} erase(g)$, $erase(g) \blacktriangleright_{\mathcal{E}} erase(f)$ and $stat_{ac}(f) = stat_{ac}(g) = us$.

The strict part of the precedence is well-founded because Σ is finite and whenever $f > g$ for AC symbols f and g, then $lab(f) >^{mul} lab(g)$ where $<$ is the subsort ordering of Σ.

Now, consider a rule $l \to r \in \widetilde{R}$ that is obtained through instantiation and labeling from an equation $l' \to r' \in R$. We prove that $l \succ_{ac} w$ for every (not necessarily proper) subterm w of r by induction on the depth of w. For the base case, let w be a variable. Since l is not a variable, we have $l \rhd w$ and since $\succ_{ac}$ is a simplification ordering we get $l \succ_{ac} w$.

In the step case, by Definitions 8 and 13, $root(w) = root(\overline{r'|_p\theta})$ for some specialization θ and position $p \in Pos(r')$ and thus $root(l) \gtrsim root(w)$ by Lemma 9. In case $root(l) > root(w)$ we have $l \succ_{ac} w|_1, \ldots, l \succ_{ac} w|_{ar(w)}$ by the induction hypothesis, and thus $l \succ_{ac} w$.

Otherwise $root(l) \sim root(w)$ and thus $root(l)$ and $root(w)$ are both AC symbols or both not AC symbols. By Definition 8 there are terms $l' =_{B_0} l$ and $w' =_{B_0} w$ such that

$$\{l''|_1, \ldots, l''|_{ar(root(l''))}\} \rhd^{tup} \{w''|_1, \ldots, w''|_{ar(root(w''))}\}$$

(where $l'' = flat(l', root(l')$ and $w'' = flat(w', root(w'))$) or

$$\{l'|_1, \ldots, l'|_{ar(root(l'))}\} \rhd^{mul} \{w'|_1, \ldots, w'|_{ar(root(w))}\}$$

respectively, depending on whether $root(l')$ and $root(w')$ are both AC-symbols or not, in case $stat(root(l)) = stat(root(w)) = mul$ or

$$\{l'|_1, \ldots, l'|_{ar(root(l'))}\} \rhd^{lex} \{w'|_1, \ldots, w'|_{ar(root(w))}\}$$

in case $stat(root(l)) = stat(root(w)) = lex$. By $AC-$compatibility of $\succ_{ac}$ it suffices to prove $l' \succ_{ac} w'$. We distinguish two cases, depending on whether $root(l')$ and $root(w')$ are AC or non-AC-symbols.

First, if they are not AC-symbols, we have

$$\{l'|_1, \ldots, l'|_{ar(root(l'))}\} \succ_{ac}^{mul} \{w'|_1, \ldots, w'|_{ar(root(w))}\}$$

in case $stat(root(l)) = stat(root(w)) = mul$ and

$$\{l'|_1, \ldots, l'|_{ar(root(l'))}\} \succ_{ac}^{lex} \{w'|_1, \ldots, w'|_{ar(root(w))}\}$$

in case $stat(root(l)) = stat(root(w)) = lex$ because by Definition 8 we have

$$\{l'|_1, \ldots, l'|_{ar(root(l'))}\} \rhd^{mul} \{w'|_1, \ldots, w'|_{ar(root(w))}\}$$

resp.

$$\{l'|_1, \ldots, l'|_{ar(root(l'))}\} \rhd^{lex} \{w'|_1, \ldots, w'|_{ar(root(w))}\}$$

and since $\succ_{ac} \supseteq \rhd$ ($\succ_{ac}$ is a simplification ordering) we get $l' \succ_{ac} w'$ according to [20][Definition 17].

Second, if both $root(w')$ and $root(l')$ are AC symbols, according to [20][Definition 17], we have to prove $cands(l', root(l')) \succ_c^{mul} cands(w', root(w'))$ (cf. [20][Definitions 12 and 17]). By, Lemma 8 this is the same as proving $cands(l'', root(l'')) \succ_c^{mul} cands(w'', root(w''))$ In the sequel let $l'' = f(s_1, \ldots, s_n)$ and $w'' = g(t_1, \ldots, t_m)$ (hence $root(l'') = f$ and $root(w'') = g$).

The multiset $cands(l'', f)$ is given by

$$\{\langle c_1 \cup \ldots \cup c_n, p_1 \cup \ldots \cup p_n\rangle \,|\langle c_1, p_1\rangle \in cands(s_1, f), \ldots, \langle c_n, p_n\rangle \in cands(s_n, f)\}.$$

Analogously, $cands(w', g)$ is given by

$$\{\langle c'_1 \cup \ldots \cup c'_m, p'_1 \cup \ldots \cup p'_m\rangle \,|\langle c'_1, p'_1\rangle \in cands(t'_1, f), \ldots, \langle c'_m, p'_m\rangle \in cands(t'_m, f)\}.$$

Note that $f \sim g$ means that $cands(s, f) = cands(s, g)$ for all terms s.

Since l'' and w'' are patterns and thus linear, the terms in the sets $\{s_1, \ldots, s_n\}$ and $\{t_1, \ldots, t_m\}$ are pairwise variable disjoint. Moreover, by Definition 8 all terms $s_1, \ldots, s_n, t_1, \ldots, t_m$ contain only function symbols smaller than f (and g) in the precedence. Hence, since $\{s_1, \ldots, s_n\} \rhd^{tup} \{t_1, \ldots, t_m\}$, Lemma 7 is applicable, yielding $cands(l'', f) \succ_c^{mul} cands(w'', g)$, and thus $l' \succ_{ac} w'$ according to [20][Definition 17].

THEOREM 2. *For any well-founded recursive tower as in Definition 9, the equational theory $(\Omega \cup \{f_1, \ldots, f_m\}, R_{f_1} \cup \ldots \cup R_{f_m})$ is a well-founded recursive many-sorted theory.*

Proof. We prove that $\mathcal{E} = (\Omega \cup \{f_1, \ldots, f_m\}, R_{f_1} \cup \ldots \cup R_{f_m})$ is well-founded recursive according to Definition 8. We use a status function $stat$ that maps every function symbol to lex, i.e. $stat(g) = lex$ for all $g \in \Omega \cup \{f_1, \ldots, f_m\}$. Since we do not have any axioms (and thus in particular no commutative and associative function symbols), $stat_{ac}$ is irrelevant.

Since there are no mutually recursive functions, the constraints for $stat$ in Definition 8 are trivially satisfied.

Moreover, item 1. of Definition 8 is satisfied since left-hand sides of equations in $\mathcal{E}$ are linear patterns according to Definition 9 item i. Item 2. is trivially satisfied, since left-hand sides of rules cannot be constructor terms. Regarding item 3., note first that since we are in a many-sorted setting, there are no non-trivial specializations. Furthermore, since we do not have any mutually recursive functions in $\mathcal{E}$, $root(\overline{r}|_p) \blacktriangleright_{\overline{\mathcal{E}}} root(\overline{l})$ implies $root(r|_p) = root(l)$ for all rules $l \to r$ of $\mathcal{E}$ and all positions $p \in Pos(r)$. Hence, in order to prove that $\mathcal{E}$ satisfies item 3. of Definition 8 it suffices to show that every rule $l \to r$ of $\mathcal{E}$ is $root(l)$ argument decreasing. Thus, consider such a rule. According to Definition 9 is has the shape

$$f_i(t_1, \ldots, t_{n_i}) \to C[f_i(u_1^1, \ldots, u_{n_i}^1) \ldots f_i(u_1^k, \ldots, u_{n_i}^k)].$$

Since $stat(f_i) = lex$, we need to show that $(t_1, \ldots, t_{n_i}) \rhd^{lex} (v_1, \ldots, v_{n_i})$ for every occurrence of a term of the shape $f_i(v_1, \ldots, v_{n_i})$ in r. By definition the context C does not contain terms rooted by f_i (item ii of Definition 9). Hence, consider the term $f_i(u_1^j, \ldots, u_{n_i}^j)$ for an arbitrary $1 \leq j \leq k$. By Definition 9 item iii.3, we have $(t_1, \ldots, t_{n_i}) \rhd^{lex} (u_1^j, \ldots, u_{n_i}^j)$. Now consider the immediate (proper) subterms of $f_i(u_1^j, \ldots, u_{n_i}^j)$. Again, by Definition 9 items iii.1 and iii.2, these terms are either terms over $\mathcal{T}(\Omega \cup \{f_1, \ldots, f_{i-1}, \mathcal{X})$ or are of the shape $f_i(v_1, \ldots, v_{n_i})$ such that $(t_1, \ldots, t_{n_i}) \trianglerighteq (v_1, \ldots, v_{n_i})$, which on the one hand implies $(t_1, \ldots, t_{n_i}) \rhd^{lex} (v_1, \ldots, v_{n_i})$ and on the other hand implies that $f_i(v_1, \ldots, v_{n_i})$ does not contain a proper f_i-rooted subterm. Hence, $l \to r$ is $root(l)$ argument decreasing and thus item 3. of Definition 8 is satisfied.

Finally, items 4. and 5. of Definition 8 are trivially satisfied, because $\mathcal{E}$ does not contain structural axioms.

B Missing Proofs of Section 4

THEOREM 3 (MODULARITY OF SORT-DECREASINGNESS). *Let $\mathcal{E}_1 = (\Sigma_1, B_0^1, R_1)$ and $\mathcal{E}_2 = (\Sigma_1 \cup \Sigma_2, B_0^1 \cup B_0^2, R_1 \cup R_2)$ be OS theories where the B_0^is are C or AC axioms for both $i \in \{1,2\}$. If $\mathcal{E}_1$ and $\mathcal{E}_2' = (\Sigma_1 \cup \Sigma_2, B_0^1 \cup B_0^2, R_2)$ are both sort-decreasing, then so is $\mathcal{E}_2$.*

Proof. We need to show that $ls(l\theta) \geq ls(r\theta)$ for all rules $l \to r \in R_1 \cup R_2$ and all specializations θ. First, assume $l \to r \in R_1$. Then because Σ_2 introduces no new subsorts of sorts in Σ_1 (by Definition 10), $ls(l\theta) \geq ls(r\theta)$ is implied by sort-decreasingness of $\mathcal{E}_1$. Second, if $l \to r \in R_2$, $ls(l\theta) \geq ls(r\theta)$ by sort decreasingness of $\mathcal{E}_2'$.

THEOREM 4 (MODULARITY OF WELL-FOUNDED RECURSION). *Let $\mathcal{E}_1 = (\Sigma_1, B_0^1, R_1)$ and $\mathcal{E}_2 = (\Sigma_1 \cup \Sigma_2, B_0^1 \cup B_0^2, R_1 \cup R_2)$ be OS theories such that $\mathcal{E}_2$ fairly extends $\mathcal{E}_1$ and the B_0^is are C or AC axioms for both $i \in \{1,2\}$. If $\mathcal{E}_1$ and $\mathcal{E}_2' = (\Sigma_1 \cup \Sigma_2, B_0^1 \cup B_0^2, R_2)$ are well-founded recursive w.r.t. to functions $stat^1, stat^1_{ac}$ and $stat^2, stat^2_{ac}$ that are compatible, then so is $\mathcal{E}_2$.*

Proof. We have to show that $\mathcal{E}_2 = (\Sigma_1 \cup \Sigma_2, B_0^1 \cup B_0^2, R_1 \cup R_2)$ is well-founded recursive. Let $\Sigma_1 = (S_1, <_1, F_1 = \Omega_1 \uplus \mathcal{D}_1)$ and $\Sigma_2 = (S_2, <_2, F_2 = \Omega_2 \uplus \mathcal{D}_2)$. We prove well-foundedness of $\mathcal{E}_2$ by looking at the rule of $R_1 \cup R_2$. First, consider a rule $l \to r$ of R_1. Since the function symbols occurring in l or the proper subterms of l resp. $flat(l, root(l))$ are from Ω_1 if l is a free or C, resp. an AC symbol, they are also constructors in $\mathcal{E}_2$ since $\Omega_2 \subseteq \Omega_1 \cup \Omega_2$ and thus l is a pattern in $\mathcal{E}_2$. Hence, if r was a constructor term w.r.t. Σ_1, then it is also a constructor term w.r.t. $\Sigma_1 \cup \Sigma_2$.

Moreover, consider a subterm t of r and a specialization θ where $root(\overline{t\theta}) \blacktriangleright_{\overline{\mathcal{E}_2'}} root(\overline{l\theta})$. We distinguish three possible cases according to the whether $root(\overline{t\theta}) \blacktriangleright_{\overline{\mathcal{E}_1}} root(\overline{l\theta})$ or not and whether $stat_{ac}(root(l))$ is s or us.

- In case $root(\overline{t\theta}) \blacktriangleright_{\overline{\mathcal{E}_1}} root(\overline{l\theta})$ we are done, because $\mathcal{E}_1$ is well-founded recursive and hence $\overline{l\theta} \to \overline{r\theta}$ is $root(\overline{t\theta})$ argument decreasing.
- If $root(\overline{t\theta}) \not\blacktriangleright_{\overline{\mathcal{E}_1}} root(\overline{l\theta})$ and $stat_{ac}(root(l)) = us$, then either $\overline{l\theta} \to \overline{r\theta}$ is $root(\overline{t\theta})$ is argument decreasing in which case we are done or $root(t) \not\blacktriangleright_{\mathcal{E}_1} root(l)$ and thus we have $root(t) \not\blacktriangleright_{\mathcal{E}_2'} root(l)$ which is a contradiction to $root(\overline{t\theta}) \blacktriangleright_{\overline{\mathcal{E}_2'}} root(\overline{l\theta})$.
- Finally, assume that $root(\overline{t\theta}) \not\blacktriangleright_{\overline{\mathcal{E}_1}} root(\overline{l\theta})$ and $stat_{ac}(root(l)) = s$ (or $root(l)$ is not an AC symbol). There is a rule $l' \to r' \in R_2$ with $root(\overline{l'\theta}) = root(\overline{t\theta})$. Hence, l' unifies in an order-sorted way with $root(t)(x_1, \ldots, x_n)$ where the sort of $x_i = s_i$ and $root(t)\colon s_1, \ldots, s_n \to s \in \Sigma_1$. Hence, we get a contradiction to $\mathcal{E}_2$ being a fair extension of $\mathcal{E}_1$.

If $root(l)$ is an AC constructor (for some specialization) we additionally need to show that $root(l|_p) \not\blacktriangleright_{\mathcal{E}_2} root(l)$ for all positions $p \in Pos_\Sigma(l)$, $p > \epsilon$. We have $root(l|_p) \not\blacktriangleright_{\mathcal{E}_1} root(l)$, because $\mathcal{E}_1$ is well-founded recursive. If $root(l|_p) \blacktriangleright_{\mathcal{E}_2} root(l)$, then there is a rule $l' \to r'$ in R_2 such that l' and $root(l|_p)(x^1_{s_1}, \ldots, x^n_{s_n})$ unify (order-sorted modulo axioms) where $root(l|_p)\colon s_1, \ldots, s_n \to s \in \Sigma_1$ and thus we derive a contradiction to $\mathcal{E}_2$ being a fair extension of $\mathcal{E}_1$.

Now consider a rule $l \to r$ from R_2. Since $\mathcal{E}_2'$ is well-founded recursive l is a pattern or constructor term and $flat(l, root(l))$ is a pattern in $\mathcal{E}_2$.

Now consider a subterm t of r and a specialization θ where $root(\overline{t\theta}) \blacktriangleright_{\overline{\mathcal{E}_1}} root(\overline{l\theta})$. We distinguish three possible cases, according to the whether $root(\overline{t\theta}) \blacktriangleright_{\overline{\mathcal{E}_1}} root(\overline{l\theta})$ or not and $stat_{ac}(root(l))$ is s or us.

- In case $root(\overline{t\theta}) \blacktriangleright_{\overline{\mathcal{E}_2'}} root(\overline{l\theta})$ we are done, because $\mathcal{E}_2'$ is well-founded recursive and hence $\overline{l\theta} \to \overline{r\theta}$ is $root(\overline{t\theta})$ argument decreasing.
- If $root(\overline{t\theta}) \not\blacktriangleright_{\overline{\mathcal{E}_2'}} root(\overline{l\theta})$ $stat_{ac}(root(l)) = us$, then either $\overline{l\theta} \to \overline{r\theta}$ is $root(\overline{t\theta})$ argument decreasing in which case we are done or $root(t) \not\blacktriangleright_{\mathcal{E}_2'} root(t)$ and thus we have $root(t) \not\blacktriangleright_{\mathcal{E}_1} root(l)$ which is a contradiction to $root(\overline{t\theta}) \blacktriangleright_{\overline{\mathcal{E}_1}} root(\overline{l\theta})$.
- Finally, assume that $root(\overline{t\theta}) \not\blacktriangleright_{\overline{\mathcal{E}_2'}} root(\overline{l\theta})$ and $stat_{ac}(root(l)) = s$ (or $root(l)$ is not an AC symbol). There is a rule $l' \to r' \in R_1$ with $root(\overline{r|'_q\theta}) = root(\overline{l\theta})$ for some position q. Hence, l unifies in an order-sorted way with $root(r'|_q)(x_1, \ldots, x_n)$ where the sort of $x_i = s_i$ and $root(r'|_p)$ is typed $root(r'|_p)\colon s_1, \ldots, s_n \to s \in \Sigma_1$. Hence, we get a contradiction to $\mathcal{E}_2$ being a fair extension of $\mathcal{E}_1$.

If $root(l)$ is an AC constructor (for some specialization) we additionally need to show that $root(l|_p) \not\blacktriangleright_{\mathcal{E}_2} root(l)$ for all positions $p \in Pos_\Sigma(l)$, $p > \epsilon$. We have $root(l|_p) \not\blacktriangleright_{\mathcal{E}_2'} root(l)$, since $\mathcal{E}_2'$ is well-founded recursive. If $root(l|_p) \blacktriangleright_{\mathcal{E}_1} root(l)$, then there exists a rule $l' \to r'$ in R_1 with $root(l') = root(l|_p)$ and thus we get a contradiction to $\mathcal{E}_2$ being a fair extension of $\mathcal{E}_1$.

Regarding Item 4 in Definition 8, rules in $\mathcal{E}_2$ satisfy this property because since no subsorts of sorts present in Σ_1 are introduced, the possible specializations of rules are the same as for $\mathcal{E}_1$ and $\mathcal{E}_2$.

Finally, if $h \blacktriangleright_{\mathcal{E}_2} g$ and $g \blacktriangleright_{\mathcal{E}_2} h$, then either $h \blacktriangleright_{\mathcal{E}_1} g$ and $g \blacktriangleright_{\mathcal{E}_1} h$ or $h \blacktriangleright_{\mathcal{E}_2'} g$ and $g \blacktriangleright_{\mathcal{E}_2'} h$, and thus either both symbols are AC or both are not AC.

THEOREM 5 (MODULARITY OF CONFLUENCE). *Let $\mathcal{E}_1 = (\Sigma_1, B_0^1, R_1)$ and $\mathcal{E}_2 = (\Sigma_1 \cup \Sigma_2, B_0^1 \cup B_0^2, R_1 \cup R_2)$ be OS theories such that $\mathcal{E}_2$ fairly extends $\mathcal{E}_1$ and the $B_0^i s$ are C or AC axioms for both $i \in \{1, 2\}$. Moreover, let $\mathcal{E}_1$ and $\mathcal{E}_2' = (\Sigma_1 \cup \Sigma_2, B_0^1 \cup B_0^2, R_2)$ be well-founded recursive w.r.t. to functions $stat^1, stat_{ac}^1$ and $stat^2, stat_{ac}^2$ that are compatible. If $\mathcal{E}_1$ and $\mathcal{E}_2'$ are confluent then so is $\mathcal{E}_2$.*

Proof. Since $\mathcal{E}_1, \mathcal{E}_2'$ and $\mathcal{E}_2$ are all terminating modulo the respective axioms according to Theorems 4 and 1, confluence of either theory is equivalent to joinability of all critical pairs (modulo axioms). To prove confluence of $\mathcal{E}_2$ we thus consider critical pairs of $\mathcal{E}_2$. If the rules used for the critical pair are either both from R_1 or both from R_2, joinability follows from confluence of $\mathcal{E}_1$ resp. $\mathcal{E}_2$. Otherwise, we distinguish two cases, depending on the type of overlap of left-hand sides of rules the critical pair originate from.

- First, assume that a left-hand side l of some rule of R_1 unifies (order-sorted modulo axioms) with a proper (non-variable) subterm t of some lhs l' of a rule of R_2 with some unifier θ. Since l' is a pattern in $\mathcal{E}_2'$, $root(t\theta)$ is a constructor in $\mathcal{E}_2'$ and thus $root(l\theta)$ is a constructor in $\mathcal{E}_1$, because $root(t\theta) = root(l\theta)$ and there are no new constructors of sorts of S_1 in $\mathcal{E}_2$ according to Definition 10. Hence, we get a contradiction to $\mathcal{E}_2$ being a fair extension of $\mathcal{E}_1$ since $root(l)$ is a constructor and l overlaps the left-hand side of a rule from R_2. Thus, there are no overlaps of this kind.
- Second, assume that a left-hand side l of some rule of R_2 unifies (order-sorted modulo axioms) with a (non-variable and not necessarily proper) subterm t of some lhs l' of a rule of R_1 with some unifier θ. Let $root(t\theta) : s_1, \ldots, s_n \to s$ be a declaration of the operator $root(t)$ in Σ_1. Since t and l unify, also $root(t)(x_{s_1}^1, \ldots, x_{s_n}^n)$ and l unify and we have a contradiction to $\mathcal{E}_2$ being a proper extension of $\mathcal{E}_1$. Hence, there are no overlaps of this kind as well.

Hence, all critical pairs of $\mathcal{E}_2$ are joinable and we deduce confluence from termination of $\mathcal{E}_2$.

THEOREM 6 (MODULARITY OF SUFFICIENT COMPLETENESS). *Let $\mathcal{E}_1 = (\Sigma_1, B_0^1, R_1)$ and $\mathcal{E}_2 = (\Sigma_1 \cup \Sigma_2, B_0^1 \cup B_0^2, R_1 \cup R_2)$ be OS theories ($\Sigma_i = \mathcal{D}_i \cup \Omega_i$) such that $\mathcal{E}_2$ fairly extends $\mathcal{E}_1$ and the $B_0^i s$ are C or AC axioms for both $i \in \{1, 2\}$. Moreover, let $\mathcal{E}_1$ and $\mathcal{E}_2' = (\Sigma_1 \cup \Sigma_2, B_0^1 \cup B_0^2, R_2)$ be well-founded recursive w.r.t. to functions $stat^1, stat_{ac}^1$ and $stat^2, stat_{ac}^2$ that are compatible. If $\mathcal{E}_1$ is sufficiently complete and for every function $f\colon s_1, \ldots, s_n \to s \in \mathcal{D}_2 \setminus \mathcal{D}_1$ and every ground substitution σ that maps variables to irreducible constructor terms, $f(x_{s_1}^1, \ldots, x_{s_n}^n)\sigma$ is either $\mathcal{E}_2$-reducible or a constructor term (where x_s is a variable of sort s), then $\mathcal{E}_2$ is sufficiently complete.*

Proof. Assume towards a contradiction that $\mathcal{E}_2$ is not sufficiently complete. Then because of termination of $\mathcal{E}_2$ (which holds by Theorem 4 and Theorem 1), there exists a term ground t, that is not a constructor term and is $\mathcal{E}_2$-irreducible. Consider a (not necessarily proper) subterm s of t, where $root(s)$ is a defined symbol and every proper subterm of s is a constructor term. If $root(s) \in \Sigma_1$, then $s \in \mathcal{T}(\Sigma_1, V)$, because there are no constructors in Σ_2 of sorts of Σ_1 by Definition 10. Hence, we obtain a contradiction to sufficient completeness of $\mathcal{E}_1$ by $\mathcal{E}_1$-irreducibility of s.

Otherwise, $root(s) \in \Sigma_2 \setminus \Sigma_1$. Then s is an instance of the term $root(s)(x_1, \ldots, x_n)$ where x_i is a variable of sort s_i and $root(s)$ is typed $root(s)\colon s_1, \ldots, s_n \to s$. Thus, we get a contradiction to $\mathcal{E}_2$-irreducibility of s.

C Missing Proofs of Section 5

LEMMA 6 (MODULARITY OF SUFFICIENT COMPLETENESS). *Under the above restrictions on the first typing of an associative operator f, the associativity equation $f(f(L, P), Q) = f(L, f(P, Q))$ is an*

inductive consequence of the restricted associativity equation $f(f(E,NL),Q) = f(E,f(NL,Q))$. *Likewise, under the second typing the associativity equation* $f(L,f(P,Q)) = f(f(L,P),Q)$ *is an inductive consequence of the restricted associativity equation* $f(Q,f(NL,E)) = f(f(Q,NL),E))$.

Proof. The proof is by constructor-based structural induction. Without loss of generality we prove the result for the fist typing, which has a constructor declaration $f : \mathit{Elt}\ \mathit{NeList} \rightarrow \mathit{NeList}\ [\mathit{ctor}]$ (the proof for the second typing is entirely similar). By assumption the only constructors of type *NeList* are *nil*, the above constructor for f, and whatever constructors may exist of sort *Elt*. These latter constructors may be ignored, since the canonical form of any term of sort *List* by the identity equations is either *nil*, or an element E, or a constructor term of the form $f(E,NL)$. By inducting on L get three inductive subgoals: (i) $f(f(nil,P),Q) = f(nil,f(P,Q))$, (ii) $f(f(E,P),Q) = f(E,f(P,Q))$, and (iii) $f(f(f(E,NL),P),Q) = f(f(E,NL),f(P,Q))$. The proof of (i) is trivial. The proof of (ii) is straightforward by structural induction on P. Let us focus on proving (iii) under the induction hypotheses $f(f(E,P),Q) = f(E,f(P,Q))$ and $f(f(NL,P),Q) = f(NL,f(P,Q))$. To begin with, (iii) can be immediately simplified with the restricted associativity equation to: $f(f(E,f(NL,P)),Q) = f(E,f(NL,f(P,Q)))$. But this goal can then be discharged by applying the first and second induction hypotheses to simplify the left term to the right term of the equation.