Certified ACKBO

Alexander Lochmann
Department of Computer Science
University of Innsbruck
Innsbruck, Austria
alexander.lochmann@uibk.ac.at

Christian Sternagel
Department of Computer Science
University of Innsbruck
Innsbruck, Austria
christian.sternagel@uibk.ac.at

Abstract

Term rewriting in the presence of associative and commutative function symbols constitutes a highly expressive model of computation, which is for example well suited to reason about parallel computations.

However, it is well known that the standard notion of termination does not apply any more: any term rewrite system containing a commutativity rule is nonterminating. Thus, instead of adding AC-rules to a rewrite system, we switch to the notion of AC-termination. AC-termination can for example be shown using AC-compatible reduction orders. One specific example of such an order is ACKBO.

We present our Isabelle/HOL formalization of the ACKBO order. On an abstract level this gives us a mechanized proof of the fact that ACKBO is indeed an AC-compatible reduction order. Moreover, we integrated corresponding check functions into the verified certifier CeTA. This has the more practical consequence of enabling the machine certification of AC-termination proofs generated by automated termination tools.

CCS Concepts • **Theory of computation** → **Equational logic and rewriting**; Higher order logic; • **Software and its engineering** → *Formal software verification.*

Keywords ACKBO, AC-termination, certification, formalization, Isabelle/HOL, term rewriting

ACM Reference Format:
Alexander Lochmann and Christian Sternagel. 2019. Certified ACKBO. In *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP '19), January 14–15, 2019, Cascais, Portugal.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3293880.3294099

1 Introduction

Rewriting in the presence of associative and commutative symbols is useful when reasoning about parallel computations. To illustrate this point, here is an example that we adapted from *All About Maude* [3] (a book about the Maude framework, which employs rewriting logic to specify and reason about parallel systems):

Example 1.1. Consider a distributed banking system, where an account $\mathsf{account}(i, x)$ is identified by its ID i and has a current balance x. The two basic operations are to withdraw an amount y from an account i (written $\mathsf{withdraw}(i, y)$) or to deposit an amount y into an account i (written $\mathsf{deposit}(i, y)$). The behavior of the banking system should be captured by the following two rewrite rules (assuming appropriate rules for addition and subtraction):

$$\mathsf{account}(i, x) \parallel \mathsf{deposit}(i, y) \to \mathsf{account}(i, x + y)$$
$$\mathsf{account}(i, x) \parallel \mathsf{withdraw}(i, y) \to \mathsf{account}(i, x - y)$$

Since we want to be able to represent an arbitrary state of a distributed system, we moreover require $\parallel$ to be associative and commutative. Then the "global" state can be represented by a term $o_1 \parallel o_2 \parallel \cdots \parallel o_n$, where each o_i is either an account or an operation. However, if we try to ensure associativity and commutativity of $\parallel$ by explicitly adding corresponding rewrite rules, we immediately end up with nontermination: $x \parallel y \to y \parallel x \to \cdots$ (due to the commutativity rule of $\parallel$).

Thus, we instead require $\parallel$ to be associative and commutative on a meta-level and thereby move from standard rewriting to *rewriting modulo associativity and commutativity* (AC-rewriting, for short).

On the one hand, in the previous example we saw that adding AC-rules to a rewrite system always introduces nontermination. On the other hand, switching from standard rewriting to AC-rewriting comes with its own problems: it is not possible to use standard termination techniques to conclude AC-termination of a term rewrite system (TRS), as demonstrated by the following example.

Example 1.2. Consider the TRS consisting of the single rule $\mathsf{a} + \mathsf{b} \to \mathsf{b} + \mathsf{a}$ where a and b are constants. Disregarding commutativity of $+$, we can conclude termination by most any technique that is available: for example using the Knuth-Bendix order with precedence $+ > \mathsf{a} > \mathsf{b}$ and all weights set to 1.

However, the above TRS is not AC-terminating, since $\mathsf{a} + \mathsf{b}$ is AC-equivalent to $\mathsf{b} + \mathsf{a}$ and thus AC-rewriting allows us to construct the infinite derivation $\mathsf{a} + \mathsf{b} \to \mathsf{a} + \mathsf{b} \to \cdots$

Hence the demand for termination techniques that allow us to conclude AC-termination of a given TRS. One class of such termination techniques is induced by AC-compatible reduction orders, with ACKBO as specific representative.

ACKBO is a variant of Knuth and Bendix's seminal order [4] that is AC-compatible and implemented in termination tools like TT₂ [5]. Moreover, a recent trend for termination tools is to formalize the underlying techniques in a proof assistant and thereby support formal verification of generated proofs. In the same spirit we formalized ACKBO.

More specifically, our contributions are as follows:

- We present our Isabelle/HOL formalization of ACKBO, the first formalization of an AC-compatible reduction order ever. After recalling the definition of ACKBO (Section 2), we give an overview of its key properties (Section 3) when it comes to proving AC-termination.
- Moreover, we develop verified executable check functions that allow us to rigorously check the correct application of ACKBO to concrete examples (Section 4).
- Finally, we evaluate the resulting formally verified certifier CeTA on a benchmark of certificates that are generated by the automated termination tool $\mathsf{T_TT_2}$ (Section 5). In the process we reveal a bug in $\mathsf{T_TT_2}$'s SMT encoding of ACKBO.

The presented work is part of the *Isabelle Formalization of Rewriting* (or IsaFoR for short)[1] since version 2.35, and compatible with Isabelle2018 [6].

In the remainder we aim to give a high-level overview of our formalization, but provide pointers (marked by ☑) to an HTML rendering of our formalization for those who want to dive right into the actual Isabelle code.

2 ACKBO

In this section we fix some required preliminaries, before we recall the definition of ACKBO that we use in our formalization. Throughout we assume basic familiarity with term rewriting and equational reasoning [1].

Preliminaries. In the remainder, let F be the set of function symbols and V be the set of variables over which we build our terms. Moreover, let AC denote the set of AC-symbols (that is, those function symbols, for which we assume associativity and commutativity). The corresponding equivalence relation, written $=_{\mathsf{AC}}$, is the symmetric, transitive, and reflexive closure of the rewrite relation induced by the rules

$$f(x, y) \to f(y, x) \qquad f(f(x, y), z) \to f(x, f(y, z)) \qquad ☑$$

for each $f \in AC$. When using infix symbols like $+$ and $\cdot$ for AC-symbols we assume the usual operator priority $\cdot > +$ in order to save some parentheses.

Given a TRS R with AC-symbols, its induced *AC-rewrite relation* $\to_{R/\mathsf{AC}}$ is given by: $s \to_{R/\mathsf{AC}} t$ iff $s =_{\mathsf{AC}} \cdot \to_R \cdot =_{\mathsf{AC}} t$ (where $\cdot$ denotes relation composition). Intuitively, this can be thought of as rewriting on AC-equivalence classes of terms.

A relation $>$ is *terminating* (or *strongly normalizing*) if it does not admit any infinite descending sequences $x_1 > x_2 > x_3 > \cdots$ We sometimes write $\mathsf{SN}_>(T)$ and $\mathsf{SN}_>(t)$ to express that there are no infinite descending sequences with respect to relation $>$ that start from terms in the set T and the term t, respectively. In such a case we also say that $>$ is strongly normalizing *on* the set T or the term t. Also note that there is no real difference between well-foundedness and termination/strong normalization of a relation as far as our formalization is concerned.[2] Nevertheless, for specific results, we usually stick to the parlance that is used in the literature.

If, given a TRS R, the relation $\to_{R/\mathsf{AC}}$ is terminating (that is, does not allow any infinite rewrite sequences), we say that R is *AC-terminating*.

We use standard set-notation like $\varnothing$ and $\{e_1, \ldots, e_n\}$ for *finite multisets*, but write $M + N$ for the *sum* of two multisets M and N, and $M - N$ for their *difference*. The *size* of a multiset M, written $|M|$, is the number of elements it contains.

Given an AC-symbol f, its *AC top* (sometimes also called *top flattening*) is the multiset given by

$$\begin{aligned} \nabla_f(f(\vec{t}_n)) &= \nabla_f(t_1) + \cdots + \nabla_f(t_n) \qquad ☑ \\ \nabla_f(t) &= \{t\} \end{aligned}$$

Which is to say that we collect the subterms remaining after removing the topmost layer of f symbols. (Since f is assumed to be AC, a multiset is an appropriate representation for its arguments.)

Given an arbitrary relation $>$ we write $>^{mul}$ and $>^{lex}$ for its *multiset extension* ☑ and *lexicographic extension* ☑, respectively.

Given a multiset of terms T, we write $T{\restriction}^P$ for its *restriction* to elements whose root symbols satisfy predicate P and $T{\restriction}_V$ for its restriction to variable terms. For example, given that $\mathsf{f} > \mathsf{g}$, we have $\{\mathsf{f}(x), \mathsf{g}(y)\}{\restriction}^{\not<\mathsf{f}} = \{\mathsf{f}(x)\}$ (the submultiset of elements whose root symbols are not less than f) and $\{\mathsf{f}(x), y\}{\restriction}_V = \{y\}$ (the submultiset of elements which are variables).

[1] http://cl-informatik.uibk.ac.at/isafor

[2] The only difference we are aware of is that the predicate SN of IsaFoR assumes that relations decrease to the right—as is typical for term rewriting applications—while, maybe for historical reasons, the predicate wf of Isabelle/HOL assumes that relations decrease to the left. This sometimes requires tedious conversions between the two notions inside a formalization but is otherwise irrelevant for our presentation.

Given a function symbol $f \in F$ and an arbitrary binary relation $>$ (we will for example use $>_{\mathrm{ACKBO}}$ and $=_{\mathrm{AC}}$ later on), we write $>^f$ for the order on multisets defined by $S >^f T$ iff $S{\restriction}^{\not< f} >^{mul} T{\restriction}^{\not< f} + T{\restriction}_V - S{\restriction}_V$.

To save some space, we sometimes write $\vec{x}_n$ instead of $x_1, \ldots, x_n$.

ACKBO. A *weight function* consists of two components: a *base weight* w_0 that is used for variables and a *weight* $w(f)$ for each function symbol $f \in F$. The weight ☑ of a term is computed recursively using the two equations $w(x) = w_0$ for all $x \in V$ and $w(f(\vec{t}_n)) = w(f) + w(t_1) + \cdots + w(t_n)$ for all function symbols $f \in F$.

A weight function is called *admissible* ☑ (with respect to a precedence $>$) whenever it satisfies the following properties:

1. $w(c) \geq w_0 > 0$ for all constants $c \in F$, and
2. $w(f) = 0$ and $f \neq g$ implies $f < g$, for all unary $f \in F$.

The latter allows for a unique unary function symbol f with weight 0 as long as it is least in the precedence. (This is for example required to accommodate the inverse operation of groups [4].)

Definition 2.1 (ACKBO ☑)**.** Let $>$ be a precedence and (w, w_0) an admissible weight function. The order $>_{\mathrm{ACKBO}}$ is inductively defined as follows: $s >_{\mathrm{ACKBO}} t$ if $|s|_x \geq |t|_x$ for all $x \in V$ and either $w(s) > w(t)$ or $w(s) = w(t)$ and one of the following holds

0. $s = f^k(t)$, $t \in V$ for some $k > 0$ or
1. $s = f(\vec{s}_m)$, $t = g(\vec{t}_n)$, $f > g$ or
2. $s = f(\vec{s}_m)$, $t = f(\vec{t}_m)$, $f \notin AC$, $(\vec{s}_m) >^{lex}_{\mathrm{ACKBO}} (\vec{t}_m)$ or
3. $s = f(s_1, s_2)$, $t = f(t_1, t_2)$, $f \in AC$, $S = \nabla_f(s)$, $T = \nabla_f(t)$,
   a. $S >^f_{\mathrm{ACKBO}} T$ or
   b. $S =^f_{\mathrm{AC}} T$ and $|S| > |T|$ or
   c. $S =^f_{\mathrm{AC}} T$, $|S| = |T|$, and $S{\restriction}^{<f} >^{mul}_{\mathrm{ACKBO}} T{\restriction}^{<f}$.

Here $=_{\mathrm{AC}}$ is used as preorder in $>^{lex}_{\mathrm{ACKBO}}$ and $>^{mul}_{\mathrm{ACKBO}}$.

Example 2.2. Recall Example 1.1. By adding the following rules for addition and subtraction (here, s stands for *successor* and p for *predecessor*) we arrive at the complete system:

$$\begin{array}{lll}
0 + y \to y & x - 0 \to x & \mathsf{p}(\mathsf{s}(x)) \to x \\
\mathsf{s}(x) + y \to \mathsf{s}(x + y) & x - \mathsf{s}(y) \to \mathsf{p}(x - y) & \mathsf{s}(\mathsf{p}(x)) \to x \\
\mathsf{p}(x) + y \to \mathsf{p}(x + y) & x - \mathsf{p}(y) \to \mathsf{s}(x - y) &
\end{array}$$

All rules can be oriented from left to right using ACKBO with precedence

$$0 > \mathsf{account} > \| > \mathsf{withdraw} > + > - > \mathsf{p} > \mathsf{deposit} > \mathsf{s}$$

and weights

$$w_0 = w(\mathsf{p}) = w(\mathsf{s}) = w(0) = w(\mathsf{deposit}) = w(\|) = 1$$

and

$$w(-) = w(\mathsf{withdraw}) = w(+) = w(\mathsf{account}) = 0.$$

In the next section we establish that ACKBO can be used to show AC-termination.

3 Formalization

In this section we give an overview of the key results of our formalization that allow us to conclude the desired result, namely that $>_{\mathrm{ACKBO}}$ is an AC-compatible reduction order.

Before we continue, we summarize some concepts that we will need later in this section.

Preliminaries. A pair of relations $(\geq, >)$ is called an *order pair* ☑ if both $\geq$ and $>$ are transitive, $\geq$ is reflexive, and the two orders are *compatible*, that is, $\geq \cdot > \cdot \geq \subseteq >$.

A relation $>$ is *AC-compatible* if $=_{\mathrm{AC}} \cdot > \cdot =_{\mathrm{AC}} \subseteq >$. Note that this implies for every order pair of the shape $(=_{\mathrm{AC}}, >)$ that $>$ is AC-compatible.

A *reduction order* is a well-founded, transitive, binary relation on terms that is closed under contexts and under substitutions.

If a term t is a subterm of a term s, we write $s \rhd t$. A relation $>$ has the *subterm property*, whenever $s \rhd t$ implies $s > t$ for all terms s and t.

Proving AC-Termination. To see why AC-compatible reduction orders imply AC-termination, assume for the sake of a contradiction that a TRS R is oriented by just such an order $>$, that is, we have $\ell > r$ for all rules $\ell \to r \in R$, but in addition R admits an infinite AC-rewrite sequence:

$$t_1 \to_{R/\mathrm{AC}} t_2 \to_{R/\mathrm{AC}} t_3 \to_{R/\mathrm{AC}} \cdots$$

First note that each step $t_i \to_{R/\mathrm{AC}} t_{i+1}$ of this sequence can be decomposed into

$$t_i =_{\mathrm{AC}} C[\ell\sigma] \to_{R/\mathrm{AC}} C[r\sigma] =_{\mathrm{AC}} t_{i+1}$$

for some rule $\ell \to r \in R$, context C, and substitution σ. Now since all rules of R are oriented by $>$, we clearly have $\ell > r$. Moreover, since $>$ is a reduction order, it is closed under contexts and substitutions, and therefore we have $C[\ell\sigma] > C[r\sigma]$. Finally, AC-compatibility of $>$ yields $t_i > t_{i+1}$, which contradicts the well-foundedness of $>$.

The Formalization. Our formalization mostly follows an earlier pen-and-paper proof by Yamada et al. [14] which is interesting from a formalization perspective for at least the following two reasons: (1) proofs are most of the time spelled out in great detail, and (2) Yamada et al. prove general properties of the structure that is shared among several AC-compatible reduction orders.

We formalize these general properties so that in principle it is possible to reuse our results also for the other AC-compatible reduction orders mentioned by Yamada et al.

Concerning well-foundedness, we actually have formalized two different proofs with slightly different implications. Our first proof follows Yamada et al. in establishing that ACKBO is a simplification order (which implies well-foundedness) and relies on an earlier formalization of *Kruskal's tree theorem* [7]. However, this result only holds for TRSs over a finite set of function symbols. While in principal this restriction is no problem for certification of termination tools (which usually take finite TRS as inputs), the existing framework for reduction orders in IsaFoR/CeTA demands well-foundedness over arbitrary signatures (including infinite ones). Therefore we also present a direct well-foundedness proof of $>_{\mathsf{ACKBO}}$ that holds for arbitrary signatures.

In the remainder of this section we mostly concentrate on proofs that deviate from the approach by Yamada et al. [14].

The following result gives us sufficient conditions for combining a countable collection of order pairs into a single order pair.

Lemma 3.1 ([14, Lemma A.3] ☑). *Assume that for all $k \in \mathbb{N}$ we have that $(\geq, >_k)$ is an order pair. Moreover assume that $>_k \subseteq >_{k+1}$ for all $k \in \mathbb{N}$. Then also $(\geq, \bigcup\{>_k \mid k \in \mathbb{N}\})$ is an order pair.*

The next lemma gives us a way to obtain an order pair on multisets of terms from an order pair of terms.

Lemma 3.2 ([14, Lemma A.2] ☑). *Let $f \in F$ and $(\geq, >)$ be an order pair. Then $(\geq^f, >^f)$ is an order pair too.*

Proof. We were not able to reconstruct the proof of this lemma from [14, Lemma A.2]. Here is an alternative proof. Suppose we have

- $S{\restriction}^{\not> f} \geq^{mul} T{\restriction}^{\not> f} + T{\restriction}_V - S{\restriction}_V$
- $T{\restriction}^{\not> f} >^{mul} U{\restriction}^{\not> f} + U{\restriction}_V - T{\restriction}_V$

From the fact that $A \geq^{mul} B$ and $B >^{mul} C$ implies $A >^{mul} C$, together with $T{\restriction}^{\not> f} \cap S{\restriction}_V = \emptyset$, we obtain

$$S{\restriction}^{\not> f} >^{mul} U{\restriction}^{\not> f} + U{\restriction}_V - T{\restriction}_V + T{\restriction}_V - S{\restriction}_V.$$

Moreover, we have

$$U{\restriction}_V - S{\restriction}_V \subseteq U{\restriction}_V - T{\restriction}_V + T{\restriction}_V - S{\restriction}_V,$$

and therefore we can show

$$S{\restriction}^{\not> f} >^{mul} U{\restriction}^{\not> f} + U{\restriction}_V - S{\restriction}_V.$$

In total this yields that $S \geq^f T$ and $T >^f U$ implies $S >^f U$ for all multisets S, T, and U. At this point, compatibility and transitivity follows trivially from the fact that $S >^f T$ implies $S \geq^f T$. □

We once more follow Yamada et al. [14] by decomposing $>_{\mathsf{ACKBO}}$ into several orders that facilitate easier proofs of AC-compatibility, transitivity, etc. Here, the indices indicate which cases of Definition 2.1 are covered.

- ☑ $s >_{01} t$ if $|s|_x \geq |t|_x$ for all $x \in V$ and either $w(s) > w(t)$ or $w(s) = w(t)$ and case 0 or case 1 of Definition 2.1 applies.
- ☑ $s >_{23,k} t$ if $|s|, |t| \leq k$, $|s|_x \geq |t|_x$ for all $x \in V$, $w(s) = w(t)$ and case 2 or case 3 of Definition 2.1 applies.

This decomposition induces a stratification of the set of terms that are oriented by the recursive calls of $>_{\mathsf{ACKBO}}$. In the following, let $>_k = >_{01} \cup >_{23,k}$.

Lemma 3.3 ([14, Lemma A.1]). *The following statements hold:*

1. ☑ $>_{ACKBO} = \bigcup\{>_k \mid k \in \mathbb{N}\}$
2. ☑ $(=_{AC}, >_{01})$ *is an order pair*
3. ☑ $(>_{01} \cdot >_k) \cup (>_k \cdot >_{01}) \subseteq >_{01}$

Lemma 3.4 ([14, Proof of Lemma 5.4] ☑). *For all $k \in \mathbb{N}$, we have that $(=_{AC}, >_k)$ is an order pair.*

Proof. This can be shown by induction on k. The base case is a direct consequence of Lemma 3.3(2) and the fact that for $k = 0$ we have $>_k = >_{01}$. For the case $k = n + 1$ we have that $(=_{\mathsf{AC}}, >_n)$ is an order pair. From Lemma 3.3(3) and Lemma 3.3(2) we get that $(=_{\mathsf{AC}}, >_k)$ is an order pair whenever $(=_{\mathsf{AC}}, >_{23,k})$ is an order pair. It is well known that lexicographic extension and multiset extension preserve compatibility and transitivity. Therefore, we only need to consider case 3 of Definition 2.1 which follows from Lemma 3.2. □

Lemma 3.5 ([14, Lemma 5.4] ☑). *The pair $(=_{AC}, >_{ACKBO})$ is an order pair.*

Proof. From Lemma 3.3(1) we have that $>_{\mathsf{ACKBO}}$ is the union of all $>_k$ for $k \in \mathbb{N}$. Moreover, $>_k \subseteq >_{k+1}$ by construction and $(=_{\mathsf{AC}}, >_k)$ is an order pair by Lemma 3.4 for all $k \in \mathbb{N}$. Thus, the desired result follows directly from Lemma 3.1. □

Remember that being an order pair where the first component is $=_{\mathsf{AC}}$ implies AC-compatibility. Hence, $>_{\mathsf{ACKBO}}$ is AC-compatible.

Next, we turn towards the question whether $>_{\mathsf{ACKBO}}$ is a reduction order. Since Lemma 3.5 already gives us transitivity of $>_{\mathsf{ACKBO}}$, it remains to establish well-foundedness, closure under contexts and closure under substitutions. We defer well-foundedness to the end of this section.

The hard part of establishing closure under contexts is covered by the following result for AC-symbols.

Lemma 3.6 (☑). *Let $+$ be an AC-symbol and $s >_{ACKBO} t$, then $u + s >_{ACKBO} u + t$ and $s + u >_{ACKBO} t + u$ for all terms u.*

Proof sketch. This can be shown using admissibility, closure under multiset sum of case 3 of Definition 2.1 and case analysis. The cases where $w(s) > w(t)$ and t is a variable are trivial. The remaining cases are covered by considering the relation between the root symbols of s and t and $+$ with respect to the precedence relation. □

At this point we obtain closure under contexts of $>_{\mathsf{ACKBO}}$.

Lemma 3.7 (Closure under Contexts ☑). *If $s >_{ACKBO} t$, then $C[s] >_{ACKBO} C[t]$, for arbitrary contexts C.*

Proof. First we prove the following statement: if $s >_{\mathsf{ACKBO}} t$ then $f(s_1, \ldots, s_n) >_{\mathsf{ACKBO}} f(t_1, \ldots, t_n)$ where $s_i = s, t_i = t$ and $\forall j \in \{1, \ldots, n\} \setminus \{i\}.\, s_j = t_j$. Consider $f \in AC$ then case 3 of Definition 2.1 applies which follows directly from Lemma 3.6. If $f \notin AC$ then we have $\forall j < i.\, s_j =_{\mathsf{AC}} t_j$ and from the assumption $s_i >_{\mathsf{ACKBO}} t_i$. Hence, case 2 of Definition 2.1 applies.

Finally, we prove closure under contexts by induction on the context. □

It turns out that the subterm property is useful for proving closure under substitutions. Thus, we cover it first.

Lemma 3.8 (Subterm Property ☑). *Whenever $s \rhd t$ then also $s >_{ACKBO} t$.*

Towards closure under substitutions we moreover provide a general lemma for the structure of case 3 of Definition 2.1 for arbitrary strict and non strict relations.

Lemma 3.9 ([14, Lemma 5.8] ☑). *Let $S = \nabla_f(s)$, $T = \nabla_f(t)$, $S' = \nabla_f(s \cdot \sigma)$, and $T' = \nabla_f(t \cdot \sigma)$. Moreover, assume that $(\geq, >)$ is an order pair, $\geq$ and $>$ are closed under substitutions, $>$ has the subterm property, and $f \in AC$. Then the following statements hold:*

1. *If $S{\restriction}^{\not< f} >^{mul} T{\restriction}^{\not< f} + T{\restriction}_V - S{\restriction}_V$ then $S'{\restriction}^{\not< f} >^{mul} T'{\restriction}^{\not< f} + T'{\restriction}_V - S'{\restriction}_V$*
2. *If $S{\restriction}^{\not< f} \geq^{mul} T{\restriction}^{\not< f} + T{\restriction}_V - S{\restriction}_V$ then $S'{\restriction}^{\not< f} >^{mul} T'{\restriction}^{\not< f} + T'{\restriction}_V - S'{\restriction}_V$ or*
   - *$S'{\restriction}^{\not< f} \geq^{mul} T'{\restriction}^{\not< f} + T'{\restriction}_V - S'{\restriction}_V$,*
   - *$|S| - |T| \leq |S'| - |T'|$, and*
   - *if $S{\restriction}^{< f} >^{mul} T{\restriction}^{< f}$ then $S'{\restriction}^{< f} \geq^{mul} T'{\restriction}^{< f}$*

In the proof of this lemma from [14] the following intermediate statement is claimed:

$$\nabla_f((T - S) \restriction_v \sigma) \restriction_v = \nabla_f(T \restriction_v \sigma) \restriction_v - \nabla_f(S \restriction_v \sigma) \restriction_v.$$

However, this does not hold in general as can be seen from the following counter example. Consider $\sigma = \{x \mapsto y\}$, $S = \{y\}$ and $T = \{x\}$. Then we have

$$\nabla_f((T - S) \restriction_v \sigma) \restriction_v = \{y\}$$

and

$$\nabla_f(T \restriction_v \sigma) \restriction_v - \nabla_f(S \restriction_v \sigma) \restriction_v = \{\}.$$

Thus, instead of the above statement, we prove the following subset relationship:

$$\nabla_f(T \restriction_v \sigma) \restriction_v - \nabla_f(S \restriction_v \sigma) \restriction_v \subseteq \nabla_f((T - S) \restriction_v \sigma) \restriction_v.$$

The remainder of the proof of [14, Lemma 5.8] goes through if we perform this replacement.

Lemma 3.10 (Closure under Substitutions ☑). *For arbitrary substitutions σ, we have that whenever $s >_{ACKBO} t$, then also $s\sigma >_{ACKBO} t\sigma$.*

Proof. We show this by well-founded induction on s with respect to the subterm relation. The variable case is trivial. Consider the case where s is a function symbol applied to arguments. Then we get that $>_{\mathsf{ACKBO}}$ is closed under substitution for all subterms of s and t using the subterm property Lemma 3.8 and the induction hypothesis. If $s >_{\mathsf{ACKBO}} t$ is not obtained by case 3 of Definition 2.1 we conclude $s\sigma >_{\mathsf{ACKBO}} t\sigma$ from the standard proof of KBO. If $s >_{\mathsf{ACKBO}} t$ holds by cases 3a or 3b then we get $s\sigma >_{\mathsf{ACKBO}} t\sigma$ from Lemma 3.9. If $s >_{\mathsf{ACKBO}} t$ holds by case 3c and cases 3a and 3b are not applicable then we get $|\nabla_f(s\sigma)| = |\nabla_f(t\sigma)|$ and $\nabla_f(s\sigma){\restriction}^{< f} \geq^{mul} \nabla_f(t\sigma){\restriction}^{< f}$ from Lemma 3.9. Hence we have $s\sigma >_{\mathsf{ACKBO}} t\sigma$. □

Taken together, the earlier results of this section establish that $>_{\mathsf{ACKBO}}$ is an AC-compatible rewrite order as well as the fact that $>_{\mathsf{ACKBO}}$ is a simplification order (any rewrite order that has the subterm property is). Now, for finite signatures F, being a simplification order already yields well-foundedness.

However, as we argued before, we are interested in well-foundedness for arbitrary signatures. Therefore, we spend the remainder of this section to establish well-foundedness of $>_{\mathsf{ACKBO}}$.

Well-foundedness. The well-foundedness proof of $>_{\mathsf{ACKBO}}$ mainly follows the structure of a similar proof by Sternagel and Thiemann for plain KBO [8, Theorem 3.7].

We start by a definition that covers case 3 of Definition 2.1, but for arbitrary base relations $>_R$ in place of $>_{\mathsf{ACKBO}}$.

Definition 3.11. We write $s >_R^{\mathsf{CASE3}} t$ if $s = f(s_1, s_2)$, $t = f(t_1, t_2)$, $f \in AC$, $S = \nabla_f(s)$, $T = \nabla_f(t)$, and

1. $S{\restriction}^{\not< f} >_{\mathsf{R}}^{mul} T{\restriction}^{\not< f} + T{\restriction}_V - S{\restriction}_V$ or
2. $S =_{\mathsf{AC}}^f T$ and $|S| > |T|$ or
3. $S =_{\mathsf{AC}}^f T$, $|S| = |T|$, and $S{\restriction}^{< f} >_{\mathsf{R}}^{mul} T{\restriction}^{< f}$

Before we prove well-foundedness of $>_{\mathsf{ACKBO}}$ below, we provide the following auxiliary results that will allow us to extend Sternagel and Thiemann's proof to the AC-case.

Lemma 3.12 (☑). *If $(\geq, >_R)$ is an order pair and $>_R$ is strongly normalizing, then $>_R^{CASE3}$ is also strongly normalizing.*

Lemma 3.13 (☑). *The relation*

$$\{(s, t) \mid (\forall u \lhd s.\, SN_R(u)) \land s >_R^{CASE3} t\}$$

is strongly normalizing whenever $(\geq, >_R)$ is an order pair and $\geq$ is symmetric.

Proof. We prove the statement by constructing a strongly normalizing order from $>_R$ that preserves the order pair property and orients arbitrary terms s and t whenever $s >_R^{\mathsf{CASE3}} t$:

$$>_U = \{(s, t) \mid (\forall u \lhd s.\, \mathsf{SN}_R(u)) \land s >_R^{\mathsf{CASE3}} t\}$$

We proceed by contradiction and assume that $>_U$ is *not* strongly normalizing.

Then we know that there is an infinite sequence of the form $t_1 >_U t_2 >_U \cdots$ in which all terms share the same root symbol, which is an AC-symbol.

Let $T = \bigcup\{t_i \mid i \in \mathbb{N}\}$ denote the set of all terms of this infinite sequence, $S = \{u \mid u \lhd s \land s \in T\}$ denote the set of all subterms of these terms, and $S' = \{s \mid t \in S \land s \geq t\}$.

We have $\mathsf{SN}_R(S)$ from the fact that S contains the collection of all subterms of terms in T. Moreover, from $\mathsf{SN}_R(t)$ we know that for all u where $t >_R u$ it follows that $\mathsf{SN}_R(u)$.

Therefore, we can conclude $\mathsf{SN}_R(S')$, because if $s \geq t$ then $t \geq s$ follows by symmetry and for all u where $s >_R u$ it follows that $t >_R u$ by compatibility.

Now we can conclude that

$$>_{R'} = \{(s,t) \mid s \in S' \wedge t \in S' \wedge s >_R t\}$$

is strongly normalizing.

We can show that $t_1 >^{\mathsf{CASE3}}_{R'} t_2 >^{\mathsf{CASE3}}_{R'} \cdots$ holds using our assumption and by case analysis. At this point, the fact that $(\geq, >_{R'})$ is an order pair follows trivially from the fact that $(\geq, >_R)$ is an order pair together with the construction of $>_{R'}$. Hence, using Lemma 3.12, we deduce that $>^{\mathsf{CASE3}}_{R'}$ is strongly normalizing. However, we have the following infinite chain $t_1 >^{\mathsf{CASE3}}_{R'} t_2 >^{\mathsf{CASE3}}_{R'} \cdots$ which contradicts the strong normalization of $>^{\mathsf{CASE3}}_{R'}$. □

Lemma 3.14 (Well-foundedness ☑). *The order $>_{ACKBO}$ is well-founded.*

As mentioned before, our proof follows the structure of [8, Theorem 3.7]. For the sake of self-containedness and readability, we reproduce most of the proof and outline the differences.

Proof. Let s be an arbitrary term. We show that $\mathsf{SN}_{>_{\mathsf{ACKBO}}}(s)$ holds. We prove this by well-founded induction on s with respect to the subterm relation $\rhd$; in contrast to the original proof, where induction on the term structure was employed. (The main difference being that in the latter case we only obtain the IH for *direct* subterms, that is, arguments of a term, but not for arbitrary subterms.)

This gives us the induction hypothesis

$$\forall u \lhd s.\, \mathsf{SN}_{>_{\mathsf{ACKBO}}}(u) \Longrightarrow \mathsf{SN}_{>_{\mathsf{ACKBO}}}(s) \tag{1}$$

If s is a variable then s is in normal form with respect to $>_{\mathsf{ACKBO}}$. Otherwise, if $s = f(s_1, \ldots, s_n)$ then we have to show that $\forall u \lhd s.\, \mathsf{SN}_{>_{\mathsf{ACKBO}}}(u)$ holds.

This is shown by the second, nested induction on f and $s_1, \ldots, s_n$ where the induction relation compares the pair $(w(f(s_1, \ldots, s_n)), f)$ lexicographically. For the first component the standard order on natural numbers and for the second one the precedence $>$ is used.

This gives us the induction hypothesis

$$\begin{gathered} w(g(t_1 \ldots, t_n)) \leq w(f(s_1, \ldots, s_n)) \wedge g < f \wedge \\ \{u \mid u \lhd g(t_1, \ldots, t_n)\} \subseteq \{s \mid \mathsf{SN}_{>_{\mathsf{ACKBO}}}(s)\} \Longrightarrow \\ \mathsf{SN}_{>_{\mathsf{ACKBO}}}(g(t_1, \ldots, t_n)) \end{gathered} \tag{2}$$

The proof for case 2 of Definition 2.1 is analogous to the proof in [8], it differs by the additional assumption that the root symbol is not in AC. Both cases are mutually exclusive. We focus on case 3 of Definition 2.1, because this extends the theorem.

With a further induction we add a third component to the lexicographic comparisons, namely

$$\{(s,t) \mid (\forall u \lhd s.\, \mathsf{SN}_{>_{\mathsf{ACKBO}}}(u)) \wedge s >^{\mathsf{CASE3}}_{\mathsf{ACKBO}} t\}.$$

From Lemma 3.13 we know that this relation is strongly normalizing and we prove the following property for all $g \in AC$ and for all $t_1, \ldots, t_n$.

$$\begin{gathered} f \geq g \wedge w(f(s_1, \ldots, s_n)) \geq w(g(t_1, \ldots, t_n)) \wedge \\ \mathsf{SN}_{>_{\mathsf{ACKBO}}}\{u \mid u \lhd g(t_1, \ldots, t_n)\} \Rightarrow \\ \mathsf{SN}_{>_{\mathsf{ACKBO}}}(g(t_1, \ldots, t_n)) \end{gathered}$$

To prove this we assume the precondition for an arbitrary g and $t_1, \ldots, t_n$ and show that $\mathsf{SN}_{>_{\mathsf{ACKBO}}}(g(t_1, \ldots, t_n))$ follows. We can close the second induction after we have shown this (the case where $g \notin AC$ is described in [8] as mentioned before). This will be done by proving that for all terms u with $g(t_1, \ldots, t_n) >_{\mathsf{ACKBO}} u$ it follows that $\mathsf{SN}_{>_{\mathsf{ACKBO}}}(u)$. We do this by a fourth induction on the term u using well-founded induction with respect to the subterm relation.

The variable case is trivial. For the function case let $u = h(h_1, \ldots, h_n)$. From the fourth induction hypothesis we know that $\forall s \lhd u.\, g(t_1, \ldots, t_n) >_{\mathsf{ACKBO}} s \Rightarrow \mathsf{SN}_{>_{\mathsf{ACKBO}}}(s)$. Using the structure of u, the subterm property of $>_{\mathsf{ACKBO}}$, Lemma 3.8, and $g(t_1, \ldots, t_n) >_{\mathsf{ACKBO}} h(h_1, \ldots, h_n)$, we finally obtain $\mathsf{SN}_{>_{\mathsf{ACKBO}}}\{h \mid h \lhd u\}$.

Now if we have that $w(f(s_1, \ldots, s_n)) > w(h(h_1, \ldots, h_n))$ or $f > h$ then we can use induction hypothesis (2) to conclude strong normalization. Otherwise, we can conclude that $f = h$ and $w(f(s_1, \ldots, s_n)) = w(h(h_1, \ldots, h_n))$ using the preconditions and $g(t_1, \ldots, t_n) >_{\mathsf{ACKBO}} h(h_1, \ldots, h_n)$. Moreover we know that $g(t_1, \ldots, t_n) >^{\mathsf{CASE3}}_{\mathsf{ACKBO}} h(h_1, \ldots, h_n)$ must hold, because $g \in AC$. Therefore, we can deduce that $\mathsf{SN}_{>_{\mathsf{ACKBO}}}(h(h_1, \ldots, h_n))$ using the third induction hypothesis. □

Example 3.15. At this point we can conclude AC-termination of the TRS from Example 1.1, since all its rules can be oriented by ACKBO as shown in Example 2.2 and moreover, we have established in this section that ACKBO is an AC-compatible reduction order.

4 Certification

In the previous section we proved all the abstract properties that are required for ACKBO so that it can in principle be used for proving termination.

Now, for certification, the setup is as follows. An external tool—like the automated termination prover TTT2—tries to show termination of a specific input TRS using ACKBO. This means that the tool tries to find a specific precedence and specific weights, such that the resulting ACKBO orients all the rules of the input TRS from left to right.

This information (the precedence and the weights) should now be provided in form of a certificate that can later be checked by an independent certifier (CeTA in our case). The agreed-upon format for termination proofs is the *certification problem format* [9] (an XML format which is also used in the annual international termination competition[3]). Therefore, a slight modification of the external tool is needed, so that it provides the required information in form of a CPF certificate.

On the side of the certifier CeTA, we need so called *check functions* that are able to take the certificate and make sure that it is indeed possible to orient all the rules of an input TRS by the corresponding ACKBO. While, so far, our formalization could employ arbitrary HOL formulas, a check function is required to be executable, since otherwise it is not possible to code-generate it into CeTA.

For our specific case of ACKBO this means that we need an executable function that corresponds to Definition 2.1 and can, given two terms s and t, decide whether $s >_{\mathsf{ACKBO}} t$ holds. It is mostly straight-forward to turn the inductive specification of Definition 2.1 ☑ into a recursive function. However, it is worth mentioning that in order to check $>^{lex}_{\mathsf{ACKBO}}$ and $>^{mul}_{\mathsf{ACKBO}}$, which take $=_{\mathsf{AC}}$ as preorder, we also need an executable way to check AC-equivalence ($=_{\mathsf{AC}}$) of two terms. The corresponding check function ☑ computes what we call *AC-normal forms* of two terms s and t, and then checks for syntactic equality. Computing the normal forms is achieved by recursive top flattening of AC-symbols, so that AC-symbols in the resulting terms take multisets of arguments instead of lists. For example, the normal form of the term $\mathsf{g}(y+z+\mathsf{a})+x$ is $+\{x, \mathsf{g}(+\{\mathsf{a}, y, z\})\}$. Finally, AC-normal forms are unique (and executable) as soon as we fix some arbitrary total order on terms.

5 Evaluation

In order to evaluate our addition of ACKBO to CeTA, we extended the existing TTT$_2$ implementation of ACKBO [14] by certificate generation. We then ran experiments on the same 145 TRSs with AC-symbols that were also used by Yamada et al. [14]. As expected, we obtained certificates for the same 32 TRSs that were also reported by Yamada et al. Of those, 31 can be certified by CeTA, while one certificate was rejected.

Example 5.1. The rejected certificate was generated for the TRS consisting of the following rules:

$$\mathsf{g}(\mathsf{g}(x)) \cdot x + \mathsf{g}(x+y) \to \mathsf{g}(x) \cdot \mathsf{g}(x) + (\mathsf{g}(x) + y)$$
$$(x+y) \cdot x \to x \cdot y + x$$
$$\mathsf{g}(x) + y \to \mathsf{g}(x+y)$$

In the certificate TTT$_2$ (version 1.17) wrongly claims that we can orient the first rule by $>_{\mathsf{ACKBO}}$ with the precedence $\mathsf{h} > + > \mathsf{g}$. However, we are in case 3 of ACKBO and computing

$$S = \{\mathsf{g}(\mathsf{g}(x)) \cdot x, \mathsf{g}(x+y)\},\ S{\restriction}^{\not<+} = \{\mathsf{g}(\mathsf{g}(x)) \cdot x\},\ S{\restriction}_V = \varnothing$$
$$T = \{\mathsf{g}(x) \cdot \mathsf{g}(x), \mathsf{g}(x), y\},\quad T{\restriction}^{\not<+} = \{\mathsf{g}(x) \cdot \mathsf{g}(x)\}, T{\restriction}_V = \{y\}$$

reveals that the variable y in the right-hand side cannot be covered when comparing the two multisets $S{\restriction}^{\not<+}$ and $T{\restriction}^{\not<+} + T{\restriction}_V - S{\restriction}_V$.

After we reported this problem it could be traced back to an error in the SMT encoding of ACKBO in TTT$_2$ (version 1.17) that was fixed in the meantime. The fixed version of TTT$_2$ (version 1.18) does no longer find an AC-termination proof for the above example.

Statistics. In total, our formalization comprises six new theory files (all in the subdirectory thys/Orderings), namely: AC_Aux, AC_Weight, Ackbo, Ackbo_split, Ackbo_simp_ord, and Ackbo_wf (the main entry point which contains the well-foundedness proof of ACKBO and imports the other theories).

Among these six theories we have 14 definitions that are used in proofs of 265 facts. The whole formalization spreads roughly across 5,000 lines of Isar [10] and took about three person-months to finish.

6 Related Work

The Knuth-Bendix order (KBO) was first introduced by Knuth and Bendix in their seminal paper on completion [4]. Later several AC-compatible variants were developed. An overview of these orders is given by Yamada et al. [14]. Various versions of the original KBO are nowadays used in automatic termination tools. A particularly powerful variant that covers most implementations was formalized in Isabelle/HOL by Sternagel and Thiemann [8].

In another interesting line of research, Becker et al. [2] extended KBO to lambda-free higher-order terms and formalized the resulting order in Isabelle/HOL.

The only implementation of ACKBO that we are aware of is in TTT$_2$, which, in turn, is used in the multi-completion tool mkbTT [11].[4]

7 Conclusion and Future Work

We presented our Isabelle/HOL formalization of ACKBO, an AC-compatible variant of the well-known Knuth-Bendix order (KBO). Moreover, we integrated an executable check function for ACKBO proofs into the certifier CeTA and evaluated the resulting system on a benchmark of certificates that were generated by TTT$_2$. In the process we uncovered an error in the SMT encoding of ACKBO that was previously used by TTT$_2$.

While some existing techniques can already be used to certify AC-termination proofs (for example by explicitly adding AC-rules as constraints for polynomial interpretations), our formalization constitutes to the best of our knowledge, the first formalization of an AC-compatible reduction order using a proof assistant.

[3]http://termination-portal.org/wiki/Termination_Competition

[4]An older version of NaTT [12] used to implement ACKBO (without producing XML certificates though), but the current version does not.

Future Work. Since 2016, IsaFoR supports a variant of the *dependency pair framework* that applies to AC-termination problems [13]. Thus it would be interesting future work to integrate AC-dependency pairs into $\mathsf{T_TT_2}$ and evaluate how powerful the combination with ACKBO is.

Acknowledgments

We thank Sarah Winkler for fixing the bug in $\mathsf{T_TT_2}$ mentioned in Section 5. This material is based upon work supported by the Austrian Science Fund (FWF) through project P27502.

References

[1] Franz Baader and Tobias Nipkow. 1999. *Term Rewriting and All That.* Cambridge University Press.

[2] Heiko Becker, Jasmin Christian Blanchette, Uwe Waldmann, and Daniel Wand. 2017. A Transfinite Knuth-Bendix Order for Lambda-Free Higher-Order Terms. In *Proceedings of the 26th International Conference on Automated Deduction (CADE) (Lecture Notes in Computer Science)*, Vol. 10395. Springer, 432–453. https://doi.org/10.1007/978-3-319-63046-5_27

[3] Manuel Clavel Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Carolyn Talcott. 2007. *All About Maude – A High-Performance Logical Framework.* Lecture Notes in Computer Science, Vol. 4350. Springer. https://doi.org/10.1007/978-3-540-71999-1

[4] Donald E. Knuth and Peter B. Bendix. 1970. Simple Word Problems in Universal Algebras. In *Computational Problems in Abstract Algebra*, John Leech (Ed.). Pergamon, 263–297. https://doi.org/10.1016/B978-0-08-012975-4.50028-X

[5] Martin Korp, Christian Sternagel, Harald Zankl, and Aart Middeldorp. 2009. Tyrolean Termination Tool 2. In *Proceedings of the 20th International Conference on Rewriting Techniques and Applications (RTA) (Lecture Notes in Computer Science)*, Vol. 5595. Springer, 295–304. https://doi.org/10.1007/978-3-642-02348-4_21

[6] Tobias Nipkow, Markus Wenzel, and Lawrence C. Paulson. 2002. *Isabelle/HOL – A Proof Assistant for Higher-Order Logic.* Lecture Notes in Computer Science, Vol. 2283. Springer. https://doi.org/10.1007/3-540-45949-9

[7] Christian Sternagel. 2014. Certified Kruskal's Tree Theorem. *Journal of Formalized Reasoning* 7, 1 (2014), 45–62. https://doi.org/10.6092/issn.1972-5787/4213

[8] Christian Sternagel and René Thiemann. 2013. Formalizing Knuth-Bendix Orders and Knuth-Bendix Completion. In *Proceedings of the 24th International Conference on Rewriting Techniques and Applications (RTA) (Leibniz International Proceedings in Informatics)*, Vol. 21. Schloss Dagstuhl, 287–302. https://doi.org/10.4230/LIPIcs.RTA.2013287

[9] Christian Sternagel and René Thiemann. 2014. The Certification Problem Format. In *Proceedings of the 11th Workshop on User Interfaces for Theorem Provers (UITP) (Electronic Proceedings in Theoretical Computer Science)*, Vol. 167. 61–72. https://doi.org/10.4204/EPTCS.167.8

[10] Makarius Wenzel. 2002. *Isabelle/Isar - A Versatile Environment for Human-Readable Formal Proof Documents.* Ph.D. Dissertation. Technische Universität München, Institut für Informatik.

[11] Sarah Winkler, Haruhiko Sato, Aart Middeldorp, and Masahito Kurihara. 2013. Mult-Completion with Termination Tools. *Journal of Automated Reasoning* 50, 3 (2013), 317–354. https://doi.org/10.1007/s10817-012-9249-2

[12] Akihisa Yamada, Keiichirou Kusakari, and Toshiki Sakabe. 2014. Nagoya Termination Tool. In *Proceedings of the Joint 25th International Conference on Rewriting Techniques and Applications (RTA) and 12th International Conference on Typed Lambda Calculi and Applications (TLCA) (Lecture Notes in Computer Science)*, Vol. 8560. Springer, 466–475. https://doi.org/10.1007/978-3-319-08918-8_32

[13] Akihisa Yamada, Christian Sternagel, René Thiemann, and Keiichirou Kusakari. 2016. AC Dependency Pairs Revisited. In *Proceedings of the 25th EACSL Annual Conference on Computer Science Logic (CSL) (Leibniz International Proceedings in Informatics)*, Vol. 62. Schloss Dagstuhl, 8:1–8:16. https://doi.org/10.4230/LIPIcs.CSL.2016.8

[14] Akihisa Yamada, Sarah Winkler, Nao Hirokawa, and Aart Middeldorp. 2016. AC-KBO Revisited. *Theory and Practice of Logic Programming* 16, 2 (2016), 163–188. https://doi.org/10.1017/S1471068415000083