On the formalization of termination techniques based on multiset orderings*

René Thiemann[1], Guillaume Allais[2], and Julian Nagele[1]

1 **University of Innsbruck, 6020 Innsbruck, Austria**
2 **University of Strathclyde, Glasgow G1 1XQ, Scotland**

Abstract

Multiset orderings are a key ingredient in certain termination techniques like the recursive path ordering and a variant of size-change termination. In order to integrate these techniques in a certifier for termination proofs, we have added them to the *Isabelle Formalization of Rewriting*. To this end, it was required to extend the existing formalization on multiset orderings towards a generalized multiset ordering. Afterwards, the soundness proofs of both techniques have been established, although only after fixing some definitions.

Concerning efficiency, it is known that the search for suitable parameters for both techniques is NP-hard. We show that checking the correct application of the techniques—where all parameters are provided—is also NP-hard, since the problem of deciding the generalized multiset ordering is NP-hard.

1998 ACM Subject Classification F.4.2 Grammars and Other Rewriting Systems

Keywords and phrases formalization, term rewriting, termination, orderings

Category Regular Research Paper

1 Introduction

The multiset ordering has been invented in the '70s to prove termination of programs [10]. It is an ingredient for important termination techniques like the multiset path ordering (MPO) [8], the recursive path ordering (RPO) [9], or recently [3, 6] it has been used in combination with the size-change principle of [19], in the form of SCNP reduction pairs.

The original version of the multiset ordering w.r.t. some base ordering $\succ$ can be defined as $M \succ_{ms} N$ iff it is possible to obtain N from M by replacing at least one element of M by several strictly smaller elements in N.

In other papers—like [3, 6, 22]—a generalization of the multiset ordering is used (denoted by $\succ_{gms}$). To define $\succ_{gms}$ one assumes that in addition to $\succ$ there is a compatible non-strict ordering $\succsim$. Then $\succ_{gms}$ is like $\succ_{ms}$, but in the multiset comparison it is additionally allowed to replace each element by a smaller one (w.r.t. $\succsim$). To illustrate the difference between $\succ_{ms}$ and $\succ_{gms}$, we take $\succ$ and $\succsim$ as the standard orderings on polynomials over the naturals. Then $\succ_{gms}$ is strictly more powerful than $\succ_{ms}$: for example, $\{\{x+1, 2y\}\} \succ_{gms} \{\{y, x\}\}$ since $x + 1 \succ x$ and $2y \succsim y$, but $\{\{x + 1, 2y\}\} \succ_{ms} \{\{y, x\}\}$ does not hold, since $2y \neq y$ and also $2y \not\succ y$ as y can be instantiated by 0.

Note that $\succ_{gms}$ is indeed used in powerful termination tools like AProVE [11]. Hence, for the certification of termination proofs which make use of SCNP reduction pairs or RPOs which are defined via $\succ_{gms}$, we need a formalization of this multiset ordering.

* This research is supported by FWF (Austrian Science Fund) project P22767.

However, in the literature and in formalizations, often only $\succ_{ms}$ is considered. And even those papers that use $\succ_{gms}$ only shortly list the differences between $\succ_{ms}$ and $\succ_{gms}$—if at all—and afterwards just assume that both multiset orderings have similar properties.

To change this situation, as one new contribution of this paper, we give the first formalization of $\succ_{gms}$, and could indeed show that $\succ_{ms}$ and $\succ_{gms}$ behave quite similar. However, we also found one essential difference between both orderings. It is well known that deciding $\succ_{ms}$ is easy, one just removes identical elements and afterwards has to find for each element in the one set a larger element in the other. In contrast, we detected and proved that deciding $\succ_{gms}$ is an NP-complete problem.

Note that the original definition of RPO misses a feature that is present in other orderings like polynomial interpretations where it is possible to compare variables against a least element: $x \succsim 0$. This feature was already integrated in other orderings like KBO [17], and it can also be integrated into RPO where one allows to compare $x \succsim c$ if c is a constant of least precedence. For example, the internal definition of RPO in AProVE is the one of [22]—which is based on $\succ_{gms}$—with the additional inference rule of "$x \succsim c$". As a second new contribution we formalized this RPO variant and fixed its definition, as it turned out that it is not stable. Moreover, we show that the change from $\succ_{ms}$ to $\succ_{gms}$ increases the complexity of RPO: From [18] it is known that deciding whether two terms are in relation w.r.t. a given RPO or MPO is in P. However, if one uses the definitions of MPO and RPO in this paper which are based on $\succ_{gms}$, then the same decision problem becomes NP-complete.

As third new contribution we also give the first formalization of SCNP reduction pairs where we could establish the main soundness theorem, although several definitions had to be fixed. Again, we have shown that the usage of $\succ_{gms}$ makes certification for SCNP reduction pairs an NP-complete problem. As a consequence, the search for suitable SCNP reduction pairs and the problem whether two terms are in relation for a given SCNP reduction pair belong to the same complexity class, as they are both NP-complete.

All our formalizations are using the proof assistant Isabelle/HOL [21]. They are available within the IsaFoR library (Isabelle Formalization of Rewriting, [24]). The new parts of the corresponding proof checker CeTA—which is just obtained by applying the code generator of Isabelle [13] on IsaFoR—have been tested on the examples of the experiments performed in [22]. After fixing some output bugs, eventually all proofs could be certified. Both IsaFoR and CeTA are freely available at:

`http://cl-informatik.uibk.ac.at/software/ceta`

The paper is structured as follows. We give preliminaries and the exact definitions for $\succ_{ms}$ and $\succ_{gms}$ in Sec. 2. Afterwards we discuss the formalization of $\succ_{gms}$ in Sec. 3. Different variants of RPO are discussed in Sec. 4. Here, we also show that certifying constraints for the RPO variant used in AProVE is NP-complete. In Sec. 5 we discuss the formalization of SCNP reduction pairs. Finally, in Sec. 6 we elaborate on our certified algorithms for checking whether two terms are in relation w.r.t. RPO or the orderings from an SCNP reduction pair, and we report on our experimental results.

2 Preliminaries

We refer to [2] for basic notions and notations of rewriting. A *signature* is a set of symbols $\mathcal{F} = \{f, g, F, G, \dots\}$, each associated with an *arity*. We write $\mathcal{T}(\mathcal{F}, \mathcal{V})$ for the set of terms over signature $\mathcal{F}$ and set of variables $\mathcal{V}$. We write $\mathcal{V}(t)$ for the set of variables occurring in t. A relation $\succ$ on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ is *stable* iff it is closed under substitutions, and it is *monotone*, iff it

is closed under contexts. A *term rewrite system* (TRS) is a set of *rules* $\ell \to r$. The rewrite relation $\to_{\mathcal{R}}$ of a TRS $\mathcal{R}$ is the smallest stable and monotone relation containing $\mathcal{R}$.

We write Id for the identity relation, and for each relation $\succ$, let $\succeq$ be its reflexive closure.

▶ **Definition 2.1** (Ordering pair, reduction pair)**.** The pair $(\succsim, \succ)$ is an *ordering pair* (over carrier A) iff $\succsim$ is a quasi-ordering over A, $\succ$ is a transitive and well-founded relation over A, and $\succ$ and $\succsim$ are compatible, i.e., $\succ \circ \succsim \subseteq \succ$ and $\succsim \circ \succ \subseteq \succ$.

The pair $(\succsim, \succ)$ is a *non-monotone reduction pair* iff $(\succsim, \succ)$ is an ordering pair over $\mathcal{T}(\mathcal{F}, \mathcal{V})$ where both $\succ$ and $\succsim$ are stable. If additionally $\succsim$ is monotone, then $(\succsim, \succ)$ is a *reduction pair*, and if both $\succ$ and $\succsim$ are monotone, then $(\succsim, \succ)$ is a *monotone reduction pair*.

Throughout this paper we only consider finite multisets $\{\{x_1, \ldots, x_n\}\}$ and we write $\mathfrak{P}(A)$ for the set of all multisets with elements from A. For every function $f : A \to A$ and every multiset $M \in \mathfrak{P}(A)$ we define the image of f on M as $f[M] = \{\{f(x) \mid x \in M\}\}$.

▶ **Definition 2.2** (Multiset orderings)**.** Let $\succ$ and $\succsim$ be relations over A. We define the *multiset ordering* ($\succ_{ms}$), the *generalized multiset ordering* ($\succ_{gms}$), and the corresponding non-strict ordering ($\succsim_{gms}$) over $\mathfrak{P}(A)$ as follows: $M_1 \succ_{ms} / \succ_{gms} / \succsim_{gms} M_2$ iff there are S_i and E_i such that $M_1 = S_1 \cup E_1$, $M_2 = S_2 \cup E_2$, and

- conditions 1, 2, and 4 are satisfied: $M \succ_{ms} N$
- conditions 1, 3, and 4 are satisfied: $M \succ_{gms} N$
- conditions 1 and 3 are satisfied: $M \succsim_{gms} N$

where conditions 1–4 are defined by:

1. for each $y \in S_2$ there is some $x \in S_1$ with $x \succ y$
2. $E_1 = E_2$
3. $E_1 = \{\{x_1, \ldots, x_n\}\}$, $E_2 = \{\{y_1, \ldots, y_n\}\}$, and $x_i \succsim y_i$ for all $1 \le i \le n$
4. $S_1 \neq \emptyset$

Whenever M_i is split into $S_i \cup E_i$ we call S_i the *strict part* and E_i the *non-strict part*.

Note that $\succ_{gms}$ indeed generalizes $\succ_{ms}$, since $\succ_{gms} = \succ_{ms}$ if $\succsim = Id$.

3 Formalization of the Generalized Multiset Ordering

Replacing condition 2 by 3 makes the formalization of $\succ_{gms}$ a bit more involved than the one of $\succ_{ms}$: the simple operation of (multiset) equality $E_1 = E_2$ is replaced by demanding that both E_1 and E_2 can be enumerated in such a way that $x_i \succsim y_i$ for all $1 \le i \le n$. Note that instead of enumerations one can equivalently demand that there is a bijection $f : E_1 \to E_2$ such that for all $x \in E_1$ we have $x \succsim f(x)$.

There is one main advantage of formalizing condition 3 via enumerations instead of bijections. It will be rather easy to develop an algorithm deciding $\succ_{gms}$. However, using enumerations we also observe a drawback: the proof that $\succ_{gms}$ and $\succsim_{gms}$ are compatible and transitive orderings will be harder than using bijections since composing bijections is easier than combining enumerations.

The formalization of the following (expected) properties for $\succ_{gms}$ and $\succsim_{gms}$ has been rather simple.

▶ **Lemma 3.1.** *$\succ_{gms}$ and $\succsim_{gms}$ have the following properties:*

1. *the empty set is the unique minimum of $\succ_{gms}$ and $\succsim_{gms}$*
2. *if $\succsim$ is reflexive then so is $\succsim_{gms}$*
3. *if $\succ$ is irreflexive and compatible with $\succsim$ then $\succ_{gms}$ is irreflexive*
4. *if $\succ$ and $\succsim$ are closed under an operation op then $\succ_{gms}$ and $\succsim_{gms}$ are closed under $op[\cdot]$.*

In contrast, the formalization of transitivity and compatibility was more tedious.

▶ **Lemma 3.2.** *If $\succ$ and $\succsim$ are compatible and transitive then so are $\succ_{gms}$ and $\succsim_{gms}$.*

Proof. For the sake of simplicity, we will work here with the definition of $(\succ_{gms}, \succsim_{gms})$ using bijections rather than enumerations: all technical details when using the representation with enumerations are available in IsaFoR, theory *Multiset-Extension*. Given that this lemma states four results that are pretty similar, we will only prove transitivity of $\succsim_{gms}$ and let the reader see how the proof could be slightly modified in the other cases.

Let M, N and P be three multisets such that $M \succsim_{gms} N$ (1) and $N \succsim_{gms} P$ (2). From (1) we get the partitions $M = M' \cup E$ and $N = N' \cup F$ and the bijection $f_{MN} : E \to F$. From (2) we get the partitions $N = N'' \cup F'$ and $P = P' \cup G$ and the bijection $f_{NP} : F' \to G$. We define the multiset I as $I = F \cap F'$ and claim that the partitions $M = (M \setminus f_{MN}^{-1}[I]) \cup f_{MN}^{-1}[I]$ and $P = (P \setminus f_{NP}[I]) \cup f_{NP}[I]$ and the bijection $f_{NP} \circ f_{MN}$ are the ones needed to prove $M \succsim_{gms} P$.

If $p \in P \setminus f_{NP}[I]$ then we have to find some $m \in M \setminus m \notin f_{MN}^{-1}[I]$ satisfying $m \succ p$. We distinguish two cases. In the first case, $p \in P'$ and thus there is an $n \in N''$ such that $n \succ p$ and $n \notin I$. If $n \in N'$ then by (1) there is some $m \in M'$ with $m \succ n$ and hence, $m \succ p$ by transitivity of $\succ$. Moreover, since $M' \subseteq M \setminus f_{MN}^{-1}[I]$ we found the desired element m. Otherwise, $n \in F$ and hence, for $m = f_{MN}^{-1}(n)$ we know $m \succsim n$ using (1). By compatibility, also $m \succ p$. Again, $m \in M \setminus f_{MN}^{-1}[I]$ since $n \notin I$. In the second case, $p \in G$ and thus for $n = f_{NP}^{-1}(p)$ we conclude $n \in F'' \setminus I$ and $n \succsim p$, and hence $n \notin F$. Thus, there is an element $m \in M'$ which satisfies $m \succ n$. By compatibility we again achieve $m \succ p$.

If $p \in f_{NP}[I]$, we have an element $n \in F'$ such that $n \succsim p$; n is also in F and therefore we have an element $m \in f_{MN}^{-1}[I]$ such that $m \succsim n \succsim p$ and hence, $m \succsim p$. ◀

Here lies the main difference to the formalization of $\succ_{ms}$ in [16]. While just transitivity needs to be shown for $\succ_{ms}$, we had to show transitivity of both $\succ_{gms}$ and $\succsim_{gms}$ as well as compatibility from both sides. Moreover the formalized proofs of these facts get more complicated since we cannot simply use bijections, set intersections, and differences, but have to deal with enumerations.

After having established Lem. 3.1 and Lem. 3.2 it remains to prove the most interesting and also most complicated property of $\succ_{gms}$, namely strong normalization. In the remainder of this section we assume that $\succ$ and $\succsim$ are compatible and transitive, and that $\succ$ is strongly normalizing. Our proof is closely related to the one for $\succ_{ms}$ [20] which is due to Buchholz: we first introduce a more atomic relation $\succ_{gms\text{-}step}$ which has $\succ_{gms}$ as its transitive closure. Then it suffices to prove that $\succ_{gms\text{-}step}$ is well-founded.

▶ **Definition 3.3.** We define $\succ_{gms\text{-}step}$ as $M \succ_{gms\text{-}step} N$ iff $M \succ_{gms} N$ where in the split $M = S \cup E$ the size of S is exactly one.

▶ **Lemma 3.4.** *$\succ_{gms}$ is the transitive closure of $\succ_{gms\text{-}step}$.*

For strong normalization we perform an accessibility-style proof, i.e., we define $\mathcal{A}$ as the set of strongly normalizing elements w.r.t. $\succ_{gms\text{-}step}$ and show that $\mathcal{A}$ contains all multisets. Note that to show $M \in \mathcal{A}$ it suffices to prove strong normalization for all N with $M \succ_{gms\text{-}step} N$. Moreover, since $\succ_{gms\text{-}step}$ is strongly normalizing on $\mathcal{A}$, one can use the following induction principle to prove some property P for all elements in $\mathcal{A}$.

$$(\forall M.(\forall N \in \mathcal{A}.M \succ_{gms\text{-}step} N \to P(N)) \to P(M)) \to (\forall M \in \mathcal{A}.P(M)) \qquad (\star)$$

We will later on apply this induction scheme using the first of the following two predicates:

- $P(x)$ is defined as $\forall M.M \in \mathcal{A} \to M \cup \{\{x\}\} \in \mathcal{A}$
- $Q(M,x)$ is defined as $\forall b.x \gtrsim b \to M \cup \{\{b\}\} \in \mathcal{A}$

For showing that all multisets belong to $\mathcal{A}$, we will require the following technical lemma.

▶ **Lemma 3.5.** *For all multisets $M \in \mathcal{A}$ and for all elements a, if $\forall N.M \succ_{gms\text{-}step} N \to Q(N,a)$ and $\forall b.a \succ b \to P(b)$ then $Q(M,a)$ holds.*

Proof. To prove $Q(M,a)$, let b be an element such that $a \gtrsim b$ where we have to show $M \cup \{\{b\}\} \in \mathcal{A}$. To prove the latter, we consider an arbitrary N with $M \cup \{\{b\}\} \succ_{gms\text{-}step} N$ and have to show $N \in \mathcal{A}$. From $M \cup \{\{b\}\} \succ_{gms\text{-}step} N$ we obtain suitable $m_1, \ldots, m_k$, $n_1, \ldots, n_k$, m, and N' to perform the splits $M \cup \{\{b\}\} = \{\{m\}\} \cup \{\{m_1, \ldots, m_k\}\}$ and $N = N' \cup \{\{n_1, \ldots, n_k\}\}$. We distinguish two cases: either b is part of $\{\{m_1, \ldots, m_k\}\}$ or equal to m.

If $b = m_i$ for some i then $M \succ_{gms\text{-}step} N \setminus \{\{n_i\}\}$. Thanks to the first hypothesis and the fact that $a \gtrsim n_i$ (because $a \gtrsim b = m_i \gtrsim n_i$), we can conclude that $N \in \mathcal{A}$.

If $b = m$ then one can prove that $\{\{n_1, \ldots, n_k\}\} \in \mathcal{A}$ using the fact that $\{\{m_1, \ldots, m_k\}\} = M \in \mathcal{A}$ and $\forall i.m_i \gtrsim n_i$. By induction on the size of N' and thanks to the second hypothesis and the fact that for any $p \in N'$, $a \gtrsim b \succ p$, we deduce that $\{\{n_1, \ldots, n_k\}\} \cup N' \in \mathcal{A}$ which concludes the proof. ◀

▶ **Lemma 3.6.** $\forall M.M \in \mathcal{A}$.

Proof. The proof of the lemma is done by induction on the size of the multiset M.

If M is the empty multiset, then obviously, it is strongly normalizing (hence in $\mathcal{A}$).

Otherwise we have to prove $\forall a.\forall M \in \mathcal{A}.M \cup \{\{a\}\} \in \mathcal{A}$ which is the same as $\forall a.P(a)$. We perform a well-founded induction on a (w.r.t. $\succ$) using the property P and are left to prove $P(a)$ assuming that $\forall b.a \succ b \to P(b)$ holds. We pick a multiset $M \in \mathcal{A}$ and perform an induction on M using $(\star)$ to prove the property $Q(M,a)$ (which entails $P(a)$ because $\gtrsim$ is reflexive): for any $M \in \mathcal{A}$ we have to prove $Q(M,a)$ given the induction hypothesis $\forall N.M \succ_{gms\text{-}step} N \to Q(N,a)$. But this result is a trivial application of Lem. 3.5 using the two induction hypotheses that we just generated. ◀

Using Lem. 3.4 and Lem. 3.6, strong normalization of $\succ_{gms}$ immediately follows.

▶ **Theorem 3.7.** *$\succ_{gms}$ is strongly normalizing.*

4 Multiset and Recursive Path Ordering

In this section we study variations of MPO and RPO. To this end, throughout this section we assume that $\geqslant$ is some precedence for the signature $\mathcal{F}$. We write $> = \geqslant \setminus \leqslant$ for the strict part of the precedence, and $\approx = \geqslant \cap \leqslant$ for its equivalence relation.

A standard version of MPO allowing quasi-precedences can be defined by the following inference rules.

$$\frac{s_i \succeq^{mpo} t}{f(\vec{s}) \succ^{mpo} t} \qquad \frac{\{\{\vec{s}\}\} \succ^{mpo}_{ms} \{\{\vec{t}\}\}}{f(\vec{s}) \succ^{mpo} g(\vec{t})}\, f \approx g \qquad \frac{f(\vec{s}) \succ^{mpo} t_i \text{ for all } i}{f(\vec{s}) \succ^{mpo} g(\vec{t})}\, f > g$$

For example for the precedence where $\mathsf{g} \approx \mathsf{h}$ we conclude that $\mathsf{f}(y, \mathsf{s}(x)) \succ^{mpo} \mathsf{f}(x,y)$ and $\mathsf{g}(\mathsf{s}(x)) \succ^{mpo} \mathsf{h}(x)$, but $\mathsf{f}(\mathsf{g}(y), \mathsf{s}(x)) \not\succ^{mpo} \mathsf{f}(x, \mathsf{h}(y))$ since $\{\{\mathsf{g}(y), \mathsf{s}(x)\}\} \not\succ^{mpo}_{ms} \{\{x, \mathsf{h}(y)\}\}$ as $\mathsf{g}(y) \not\succeq^{mpo} \mathsf{h}(y)$.

To increase the power of MPO, the following inference rules are sometimes used which generalize MPO by defining two orderings $\succ^{gmpo}$ and $\succsim^{gmpo}$ where the multiset extension is done via $\succ_{gms}$. This variant of MPO is the one that is internally used within AProVE, and if one removes the last inference rule ($x \succsim^{gmpo} c$), then it is equivalent to the MPO definition of [22].

$$(1)\ \frac{s_i \succsim^{gmpo} t}{f(\vec{s}) \succ^{gmpo} t} \qquad (2)\ \frac{\{\{\vec{s}\}\} \succ^{gmpo}_{gms} \{\{\vec{t}\}\}}{f(\vec{s}) \succ^{gmpo} g(\vec{t})}\, f \approx g \qquad (3)\ \frac{f(\vec{s}) \succ^{gmpo} t_i \text{ for all } i}{f(\vec{s}) \succ^{gmpo} g(\vec{t})}\, f > g$$

$$(4)\ \frac{s \succeq^{gmpo} t}{s \succsim^{gmpo} t} \qquad (5)\ \frac{\{\{\vec{s}\}\} \succsim^{gmpo}_{gms} \{\{\vec{t}\}\}}{f(\vec{s}) \succsim^{gmpo} g(\vec{t})}\, f \approx g \qquad (6)\ \frac{}{x \succsim^{gmpo} c}\, \text{if } \forall f \in \mathcal{F} : f \geqslant c$$

Hence, there are two differences between $\succeq^{gmpo}$ (the reflexive closure of $\succ^{gmpo}$) and $\succsim^{gmpo}$: only in $\succsim^{gmpo}$ one can compare multisets using the non-strict multiset ordering, and one can compare variables against constants of least precedence. This latter feature is similar to polynomial orderings where $x \geqslant 0$, and it was also added to the Knuth-Bendix-Ordering [17].

The increase of power of $\succ^{gmpo}$ in comparison to $\succ^{mpo}$ is due to both new features in the non-strict relation. For example, using the same precedence as before, $\mathsf{f}(\mathsf{g}(y), \mathsf{s}(x)) \succ^{gmpo} \mathsf{f}(x, \mathsf{h}(y))$ as $\mathsf{g}(y) \succsim^{gmpo} \mathsf{h}(y)$ and $\mathsf{s}(x) \succ^{gmpo} x$. Moreover, $\mathsf{f}(\mathsf{f}(y, z), \mathsf{s}(x)) \succ^{gmpo} \mathsf{f}(x, \mathsf{f}(\mathsf{a}, z))$ if a has least precedence, where this decrease is only possible due to the comparison $y \succsim^{gmpo} \mathsf{a}$.

Note that $\succsim^{gmpo}$ is a strict superset of the equivalence relation where equality is defined modulo $\approx$ and modulo permutations. For this inclusion, indeed all three inference rules (4-6) of $\succsim^{gmpo}$ are essential. With (4) and (5) we completely subsume the equivalence relation, and (6) exceeds the equivalence relation. However, then also (5) adds additional power by using $\succsim_{gms}$ instead of multiset equality w.r.t. the equivalence relation. As an example, consider $\mathsf{g}(x) \succsim^{gmpo} \mathsf{g}(\mathsf{a})$.

Finally note that our definition does not require any explicit definition of an equivalence relation, one can just use $\succsim^{gmpo} \cap \precsim^{gmpo}$. This is in contrast to the RPO in related formalizations of CoLoR [5] and Coccinelle [7]. In Coccinelle first an equivalence relation for RPO is defined explicitly, before defining the strict ordering,[1] and the RPO of CoLoR currently just supports syntactic equality.[2]

The reason that AProVE uses $\succ^{gmpo}$ for its MPO implementation is easily understood, as $\succ^{gmpo}$ is more powerful than $\succ^{mpo}$, and the SAT/SMT-encodings of $\succ^{mpo}$ and $\succ^{gmpo}$ to find a suitable precedence are quite similar, so there is not much overhead. Hence, in order to be able to certify AProVE's MPO proofs—which then also allows to certify weaker variants of MPO—we have formally proved that $(\succsim^{gmpo}, \succ^{gmpo})$ is a monotone reduction pair. Concerning strong normalization of $\succ^{gmpo}$, we did not use Kruskal's tree theorem, but we performed a proof similar to the strong normalization proof of the (higher-order) recursive path ordering as in [5, 7, 15, 16] (which is based on reducibility predicates of Tait and Girard.) By following this proof and by using the results of Sec. 3, it was an easy—but tedious—task to formalize the following main result. Here—as for the generalized multiset ordering—the transitivity proof became more complex as one has to prove transitivity and compatibility of $\succsim^{gmpo}$ and $\succ^{gmpo}$ at the same time, i.e., within one large inductive proof.

▶ **Theorem 4.1.** *The pair $(\succsim^{gmpo}, \succ^{gmpo})$ is a monotone reduction pair.*

Whereas Thm. 4.1 was to be expected—$\succ^{gmpo}$ is just an extension of $\succ^{mpo}$, and it is well known that $(\succeq^{mpo}, \succ^{mpo})$ is a monotone reduction pair—we detected a major difference

[1] See `http://www.lri.fr/~contejea/Coccinelle/doc/term_orderings.rpo.html`.
[2] See `http://color.inria.fr/doc/CoLoR.RPO.VRPO_Type.html`.

when trying to certify existing proofs where one has to compute for a given precedence whether two terms are in relation. This problem is in P for $\succ^{mpo}$ but it turns out to be NP-complete for $\succ^{gmpo}$.

▶ **Theorem 4.2.** *Let there be some fixed precedence. The problem of deciding $\ell \succ^{gmpo} r$ for two terms ℓ and r is NP-complete.*

Proof. Membership in NP is easily proved. Since the size of every proof-tree for $\ell \succ^{mpo} r$ is bounded by $2 \cdot |\ell| \cdot |r|$, one can just guess how the inference rules for $\succ^{mpo}$ have to be applied; and for the multiset comparisons one can also just guess the splitting.

To show NP-hardness we perform a reduction from SAT. So let ϕ be some Boolean formula over variables $\{x_1, \ldots, x_n\}$ represented as a set of clauses $\{C_1, \ldots, C_m\}$ where each clause is a set of literals, and each literal l is variable x_i or a negated variable $\overline{x_i}$. W.l.o.g. we assume $n \geq 2$.

In the following we will construct one constraint $\ell \succ^{gmpo} r$ for terms $\ell, r \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ where $\mathcal{F} = \{\mathsf{a}, \mathsf{f}, \mathsf{g}, \mathsf{h}\}$ and $\mathcal{V} = \{x_1, \ldots, x_n, y_1, \ldots, y_m\}$. To this end, we define $s(l, C_j) = y_j$, if $l \in C_j$, and $s(l, C_j) = \mathsf{a}$, otherwise. Moreover, $t_x^+ = \mathsf{f}(x, s(x, C_1), \ldots, s(x, C_m))$, $t_x^- = \mathsf{f}(x, s(\overline{x}, C_1), \ldots, s(\overline{x}, C_m))$, $t_x = \mathsf{f}(x, \mathsf{a}, \ldots, \mathsf{a})$. We define $L = \{\{t_{x_1}^+, t_{x_1}^-, \ldots, t_{x_n}^+, t_{x_n}^-\}\}$ and $R = \{\{t_{x_1}, \ldots, t_{x_n}, y_1, \ldots, y_m\}\}$. Finally, we define $\ell = \mathsf{g}(L)$ and $r = \mathsf{h}(R)$—where here we abuse notation and interpret L and R as lists of terms.

We prove that ϕ is satisfiable iff $\ell \succ^{gmpo} r$ for the precedence where $\mathsf{a} \approx \mathsf{f} \approx \mathsf{g} \approx \mathsf{h}$. For this precedence, $\ell \succ^{gmpo} r$ iff $L \succ_{gms}^{gmpo} R$ since there is only one inference rule that can successfully be applied. The reason is that $\mathsf{f} \not\succ \mathsf{g}$ and for each term $t_{x_i}^{\pm}$ of L we have $t_{x_i}^{\pm} \not\succsim^{gmpo} r$ where $t_{x_i}^{\pm}$ represents one of the terms $t_{x_i}^+$ or $t_{x_i}^-$. To see the latter, assume $t_{x_i}^{\pm} \succsim^{gmpo} r$ would hold. Then $\{x_i, y_1, \ldots, y_m\} \supseteq \mathcal{V}(t_{x_i}^{\pm}) \supseteq \mathcal{V}(r) \supseteq \{x_1, \ldots, x_n\}$ being a contradiction to $n \geq 2$.

To examine whether $L \succ_{gms}^{gmpo} R$ can hold, let us consider an arbitrary splitting of R into a strict part S' and a non-strict part E'. Notice that $t_x^+ \succsim^{gmpo} t_x$ and $t_x^- \succsim^{gmpo} t_x$, but neither $t_x^+ \succ^{gmpo} t_x$ nor $t_x^- \succ^{gmpo} t_x$: the reason is that for each l and C_j we get $s(l, C_j) \succsim^{gmpo} \mathsf{a}$ but not $s(l, C_j) \succ^{gmpo} \mathsf{a}$. Hence, each t_x of R must be contained in E'. As moreover, $t_{x_i}^{\pm} \succ^{gmpo} y_j$ iff $t_{x_i}^{\pm} \succsim^{gmpo} y_j$ we can w.l.o.g. assume that each $y_j \in S'$. In total, if $L \succ_{gms}^{gmpo} R$, then R must be split into $S' \cup E'$ where $S' = \{\{y_1, \ldots, y_m\}\}$ and $E' = \{\{t_{x_1}, \ldots, t_{x_n}\}\}$ and L must be split into $S \cup E$ such that all conditions of $\succ_{gms}^{gmpo}$ are satisfied.

At this point we consider both directions to show that ϕ is satisfiable iff $L \succ_{gms}^{gmpo} R$.

First assume that ϕ is satisfiable, so let α be some satisfying assignment. Then we choose $S = \{\{t_x^+ \mid \alpha(x) = \top\}\} \cup \{\{t_x^- \mid \alpha(x) = \bot\}\}$ and $E = L \setminus S$. Notice that for each x exactly one of t_x^+ and t_x^- is in S, and the other is in E. Hence, for each $t_x \in E'$ there is a corresponding $t_x^{\pm} \in E$ with $t_x^{\pm} \succsim^{gmpo} t_x$. Next, we have to find for each $y_j \in S'$ some term in S which is larger than y_j. To this end, notice that α is a satisfying assignment, thus there is some literal x_i or $\overline{x_i}$ in C_j which evaluates to true. If $x_i \in C_j$ then $\alpha(x_i) = \top$ and hence, $t_{x_i}^+ \in S$ where $t_{x_i}^+ = \mathsf{f}(\ldots, y_j, \ldots) \succ^{gmpo} y_j$. Otherwise, $\overline{x_i} \in C_j$ and $\alpha(x_i) = \bot$ and hence, $t_{x_i}^- \in S$ where $t_{x_i}^- = \mathsf{f}(\ldots, y_j, \ldots) \succ^{gmpo} y_j$. Thus, in both cases there is some term in S being larger than y_j. Moreover, $S \neq \emptyset$ since $|S| = n \geq 2$.

For the other direction assume that S and E could be found such that $L = S \cup E$ and all conditions of $\succ_{gms}^{gmpo}$ are satisfied. Hence for each $t_{x_i} \in E'$ there is some term $t \in E$ satisfying $t \succsim^{gmpo} t_{x_i}$. Then, t can only be $t_{x_i}^+$ or $t_{x_i}^-$ since $x_i \in \mathcal{V}(t_x)$ and each other term $t_{x_j}^{\pm}$ with $i \neq j$ does not contain the variable x_i. Thus, for each i exactly one of the terms $t_{x_i}^+$ and $t_{x_i}^-$ is contained in E and the other is contained in S. We define the assignment α where $\alpha(x_i) = \top$ iff $t_{x_i}^+ \in S$. It remains to show that α is a satisfying assignment for ϕ. So let C_j be some

clause of ϕ. Since $y_j \in S'$ we know that there is some $t \in S$ with $t \succ^{gmpo} y_j$, i.e., $y_j \in \mathcal{V}(t)$. There are two cases. First, if $t = t^+_{x_i}$ for some i, then by the definition of t^+ we know that $y_j \in \mathcal{V}(t^+_{x_i})$ implies $x_i \in C_j$, and hence C_j is evaluated to true, since $\alpha(x_i) = \top$ by definition of α. Otherwise, $t = t^-_{x_i}$ for some i where now $\overline{x_i} \in C_j$. Moreover, as $t^-_{x_i} \in S$, we know that $t^+_{x_i} \notin S$ and hence, $\alpha(x_i) = \bot$. Together with $\overline{x_i} \in C_j$ this again shows that C_j is evaluated to true. Hence, all clauses evaluate to true using α which proves that ϕ is satisfiable. ◀

The following corollary states that NP-completeness is essentially due to fact that $\succ_{gms}$ and $\succsim_{gms}$ are hard to compute, even if all comparisons of the elements in the multiset are given. It can be seen within the previous proof, where the important part of the reduction from SAT was to define for each formula ϕ the multisets L and R such that ϕ is satisfiable iff $L \succ^{gmpo}_{gms} R$ (and also iff $L \succsim^{gmpo}_{gms} R$).

▶ **Corollary 4.3.** *Given two orderings $\succ$ and $\succsim$, two multisets M and N, and the set $\{(x, y, x \succ y, x \succsim y) \mid x \in M, y \in N\}$, deciding $M \succ_{gms} N$ and $M \succsim_{gms} N$ is NP-complete.*

Of course, if the splitting for the multiset comparison is given, then deciding $\succ_{gms}$ and $\succsim_{gms}$ becomes polynomial.

All our results have also been generalized to RPO where for every function symbol there is a status function τ which determines whether the arguments of each function f should be compared lexicographically ($\tau(f) = lex$) or via multisets ($\tau(f) = mul$).[3] Here, the existing inference rules of $\succ^{gmpo}$ are modified that instead of $f \approx g$ it is additionally demanded that $\tau(f) = \tau(g) = mul$. Moreover, there are two additional inference rules for $f \approx g$ and $\tau(f) = \tau(g) = lex$, one for the strict ordering $\succ^{grpo}$ and one for the non-strict ordering $\succsim^{grpo}$.

Again, $\succ^{grpo}$ is used in AProVE instead of the standard definition of RPO ($\succ^{rpo}$,[9]). However, during our formalization we have detected that in contrast to $(\succsim^{gmpo}, \succ^{gmpo})$, the pair $(\succsim^{grpo}, \succ^{grpo})$ is not a reduction pair, as the orderings are not stable. To see this, consider a precedence where a and b have least precedence, and $\tau(\mathsf{a}) \neq \tau(\mathsf{b})$. Then $x \succsim^{grpo} \mathsf{a}$, but $\mathsf{b} \not\succsim^{grpo} \mathsf{a}$.

Our solution to this problem is to add the following further inference rules which allow comparisons of terms $f(\vec{s})$ with $g(\vec{t})$ where $\tau(f) \neq \tau(g)$. In detail, we require that $\vec{t}$ is empty and for a strict decrease, additionally $\vec{s}$ must be non-empty.

$$\frac{|\vec{s}| > 0 \quad |\vec{t}| = 0}{f(\vec{s}) \succ^{grpo} g(\vec{t})} f \approx g, \tau(f) \neq \tau(g) \qquad \frac{|\vec{t}| = 0}{f(\vec{s}) \succsim^{grpo} g(\vec{t})} f \approx g, \tau(f) \neq \tau(g)$$

If these inference rules are added, then indeed $(\succsim^{grpo}, \succ^{grpo})$ is a monotone reduction pair. As a consequence, one can interpret AProVE's version of RPO as a sound, but non-stable under-approximation of $\succ^{grpo}$.

We also tried to to relax the preconditions further, e.g. by allowing vectors $\vec{t}$ of length at most one. But no matter whether we also restrict the length of $\vec{s}$ in some way or not, and no matter whether we compared the arguments $\vec{s}$ and $\vec{t}$ lexicographically or via multisets, the outcome was always that transitivity or strong normalization are lost.

For example, if we add the inference rule that $\{\{\vec{s}\}\} \succ^{grpo}_{gms} \{\{\vec{t}\}\}$, $|\vec{t}| \leq 1$, $f \approx g$, and $\tau(f) \neq \tau(g)$ implies $f(s) \succ^{grpo} g(t)$, then strong normalization is lost: assume the precedence

[3] We do not consider permutations for lexicographic comparisons in the definition of RPO as this feature can be simulated by generating reduction pairs using an RPO (without permutations) in combination with argument filterings as defined in [1]. In this way, we only have to formalize permutations once and we can reuse them for other orderings like KBO.

is defined by $\mathsf{f} \approx \mathsf{g} \approx \mathsf{h}$ and $3 > 2 > 1 > 0$, and the status is defined by $\tau(\mathsf{f}) = \tau(\mathsf{h}) = \mathit{lex}$ and $\tau(\mathsf{g}) = \mathit{mul}$. Then $\mathsf{f}(0,3) \succ^{grpo} \mathsf{g}(2) \succ^{grpo} \mathsf{h}(1) \succ^{grpo} \mathsf{f}(0,3)$ clearly shows that the resulting ordering is not strongly normalizing anymore.

To summarize, $\succ^{gmpo}$ and $\succ^{grpo}$ are strictly more powerful reduction orderings than the standard definitions of MPO and RPO ($\succ^{mpo}$ and $\succ^{rpo}$). The price for the increased power is that checking constraints for $\succ^{gmpo}$ and $\succ^{grpo}$ becomes NP-complete whereas it is in P for $\succ^{mpo}$ and $\succ^{rpo}$.

Note that if one would provide all required splittings for the multiset comparisons in $\succ^{gmpo}$ and $\succ^{grpo}$, then constraint checking again becomes polynomial. However, this would make certificates more bulky, and since in practice the arities of function symbols are rather small, certification can efficiently be done even without additional splitting information.

5 SCNP Reduction Pairs

The size-change criterion of [19] to prove termination of programs can be seen as a graph-theoretical problem: given a set of graphs—encoding for each recursive call the decrease in size of each argument—one tries to decide the *size-change termination condition* (SCT condition) on the graphs, namely that in every infinite sequence of graphs one can find an argument whose size is strictly decreased infinitely often. If the condition is satisfied, then termination is proved.

Concerning automation of the size-change principle, there are two problems: first, the base ordering (or size-measure or ranking function) must be provided to construct the graphs, and second, even for a given base ordering, deciding the SCT condition is PSPACE-complete.

To overcome these problems, in [3] and [6] sufficient criteria have been developed. They approximate the SCT condition in a way that can be encoded into SAT.

Another benefit of [6] is an integration of the approximated SCT condition as a reduction pair in the dependency pair framework (DP framework) [12], called *SCNP reduction pair*.

Although IsaFoR contains already a full formalization of size-change termination as it is used in [23], an integration of SCNP reduction pairs in the certification process might be beneficial for two reasons:

- since deciding the SCT condition is PSPACE hard, whereas the approximated condition can be encoded into SAT, the certificates might be easier to check
- the approximated SCT condition is not fully subsumed by the SCT condition as only the former allows incremental termination proofs in the DP framework

Before we describe our formalization of SCNP reduction pairs, we shortly recall some notions of the DP framework, a popular framework to perform modular termination proofs for TRSs. The main data structure are *DP problems* $(\mathcal{P}, \mathcal{R})$ consisting of two TRSs where all rules in $\mathcal{P}$ are of the form $F(\dots) \to G(\dots)$ where F, G are symbols that do not occur in $\mathcal{R}$. The main task is to prove finiteness of the a given DP problem $(\mathcal{P}, \mathcal{R})$, i.e., absence of *infinite minimal chains* $s_1\sigma \to t_1\sigma \to^*_{\mathcal{R}} s_2\sigma \to t_2\sigma \dots$ where all $s_i \to t_i \in \mathcal{P}$ and all $t_i\sigma$ are terminating w.r.t. $\mathcal{R}$. To this end, one uses various *processors* to simplify the initial DP problem for a TRS until the $\mathcal{P}$-component is empty.

One of the most important processors is the reduction pair processor [12, 14]—where here we only present its basic version without other refinements like usable rules w.r.t. an argument filtering [12]. It can remove strictly decreasing rules from $\mathcal{P}$, provided that both $\mathcal{P}$ and $\mathcal{R}$ are at least weakly decreasing.

▶ **Theorem 5.1** (Reduction pair processor). *Let $(\succsim, \succ)$ be a reduction pair. If $\mathcal{P} \subseteq \succ \cup \succsim$ and $\mathcal{R} \subseteq \succsim$ then $(\mathcal{P}, \mathcal{R})$ is finite if $(\mathcal{P} \setminus \succ, \mathcal{R})$ is finite.*

Hence, to prove termination it suffices to find different reduction pairs to iteratively remove rules from $\mathcal{P}$ until all rules of $\mathcal{P}$ have been removed. Thus, with SCNP reduction pairs it is possible to choose different base orderings to remove different rules of $\mathcal{P}$.

In the following we report on details of SCNP reduction pairs as defined in [6] and on their formalization. Essentially, SCNP reduction pairs are generated from reduction pairs $(\succsim, \succ)$ via a multiset extension of a lexicographic combination of $\succ$ with the standard ordering on the naturals.

▶ **Definition 5.2** (Multiset extension)**.** We define that μ is a *multiset extension* iff for each ordering pair $(\succsim, \succ)$ over A, the pair $(\succsim_\mu, \succ_\mu)$ is an ordering pair over $\mathfrak{P}(A)$. Moreover, whenever $\succ$ and $\succsim$ are closed under an operator op ($x \mathrel{(\succsim)} y$ implies $op(x) \mathrel{(\succsim)} op(y)$ for all x, y), then $\succ_\mu$ and $\succsim_\mu$ must also be closed under the image of op on multisets ($M \mathrel{(\succsim)_\mu} N$ implies $op[M] \mathrel{(\succsim)_\mu} op[N]$).

We also call $(\succsim_\mu, \succ_\mu)$ the *multiset extension* of $(\succsim, \succ)$ w.r.t. μ.

For example, *gms* is a multiset extension and in [3] and [6] in total four extensions are listed to compare multisets (*gms*, *min*, *max*, and *dms*). The extension *dms* is the dual multiset extension [4] where our formulation is equivalent to the definition in [6].[4] The strict relation is defined as $M \succ_{dms} N$ iff $M = \{\{x_1, \ldots, x_m\}\} \cup \{\{z_1, \ldots, z_k\}\}$, $N = \{\{y_1, \ldots, y_n\}\} \cup \{\{z'_1, \ldots, z'_k\}\}$, $n > 0$, $\forall i. z_i \succsim z'_i$, and $\forall x_i. \exists y_j. x_i \succ y_j$; the non-strict relation $\succsim_{dms}$ is defined like $\succ_{dms}$ except that the condition $n > 0$ is omitted.

For SCNP reduction pairs, multisets are compared via one of the four multiset extensions. And to generate multisets from terms, the notion of a level mapping is used.

▶ **Definition 5.3** (Level mapping [6])**.** For each $f \in \mathcal{F}$ with arity n, let $\pi(f) \in \mathfrak{P}(\{1, \ldots, n\} \times \mathbb{N})$.[5] We define the *level-mapping* $\mathcal{L} : \mathcal{T}(\mathcal{F}, \mathcal{V}) \to \mathfrak{P}(\mathcal{T}(\mathcal{F}, \mathcal{V}) \times \mathbb{N})$ where $\mathcal{L}(f(s_1, \ldots, s_n)) = \{\{\langle s_i, k\rangle \mid (i, k) \in \pi(f)\}\}$.[6]

▶ **Definition 5.4.** Let $(\succsim, \succ)$ be a reduction pair for terms in $\mathcal{T}(\mathcal{F}, \mathcal{V})$. Let μ be a multiset extension. The ordering pair $(\succsim^{\mathbb{N}}, \succ^{\mathbb{N}})$ over $\mathcal{T}(\mathcal{F}, \mathcal{V}) \times \mathbb{N}$ is defined as the lexicographic combination of $(\succsim, \succ)$ with the $>$-ordering on the naturals: $\langle s, n\rangle \succsim^{\mathbb{N}} \langle t, m\rangle$ iff $s \succ t \vee (s \succsim t \wedge n \geq m)$, and $\langle s, n\rangle \succ^{\mathbb{N}} \langle t, m\rangle$ iff $s \succ t \vee (s \succsim t \wedge n > m)$. The ordering pair $(\succsim^{\mathbb{N}}_\mu, \succ^{\mathbb{N}}_\mu)$ is defined as the multiset extension of $(\succsim^{\mathbb{N}}, \succ^{\mathbb{N}})$ w.r.t. μ.

In principle, $\succ^{\mathbb{N}}_\mu \cup \succsim^{\mathbb{N}}_\mu$ is the part of a SCNP reduction pair that should be used to compare left- and right-hand sides of $\mathcal{P}$ within the reduction pair processor. However, one also needs to orient the rules in $\mathcal{R}$ via $\succsim$. To this end, two types are introduced in [6] so that the final ordering incorporates both $\succ^{\mathbb{N}}_\mu \cup \succsim^{\mathbb{N}}_\mu$ for $\mathcal{P}$ and $\succsim$ for $\mathcal{R}$. In detail, the signature $\mathcal{F}$ is partitioned into $\mathcal{F}^b \uplus \mathcal{F}^\sharp$ consisting of base symbols $\mathcal{F}^b$ and tuple-symbols $\mathcal{F}^\sharp$. The set of *base terms* is $\mathcal{T}(\mathcal{F}^b, \mathcal{V})$, and a *tuple term* is a term of the form $F(t_1, \ldots, t_n)$ where $F \in \mathcal{F}^\sharp$ and each t_i is a base term.

Notice that for a DP problem $(\mathcal{P}, \mathcal{R})$ all terms in $\mathcal{P}$ are tuple terms and all terms in $\mathcal{R}$ are base terms if one chooses $\mathcal{F}^\sharp$ to be the set of root symbols of terms in $\mathcal{P}$. Therefore, SCNP reduction pairs are defined in a way that the ordering depends on whether tuple terms or base terms are compared.

---

[4] There is a difference in the definition of the dual multiset extension in [3,4] to the definition in [6] which is similar to the difference between $\succ_{ms}$ and $\succ_{gms}$.

[5] In [6] there was an additional condition that $\pi(f)$ may not contain two entries $\langle j, k_1\rangle$ and $\langle j, k_2\rangle$. It turned out that this condition is not required for soundness.

[6] We use the notation $\mathcal{L}$ instead of ℓ for level mappings as in this paper, ℓ are left-hand sides of rules.

▶ **Definition 5.5** (SCNP reduction pair [6])**.** Let μ be a multiset extension and $\mathcal{L}$ be a level mapping. Let $(\succsim, \succ)$ be a reduction pair over $\mathcal{T}(\mathcal{F}, \mathcal{V})$. The *SCNP reduction pair* is the pair $(\succsim^{\mathcal{L},\mu}, \succ^{\mathcal{L},\mu})$ where the relations $\succsim^{\mathcal{L},\mu}$ and $\succ^{\mathcal{L},\mu}$ over $\mathcal{T}(\mathcal{F}, \mathcal{V})$ are defined as follows. If t and s are tuple terms, then $t \succ^{\mathcal{L},\mu} s$ iff $\mathcal{L}(t) \succ_\mu^{\mathbb{N}} \mathcal{L}(s)$, and $t \succsim^{\mathcal{L},\mu} s$ iff $\mathcal{L}(t) \succsim_\mu^{\mathbb{N}} \mathcal{L}(s)$. Otherwise, if t and s are base terms, then $t \succsim^{\mathcal{L},\mu} s$ iff $t \succsim s$, and $t \succ^{\mathcal{L},\mu} s$ iff $t \succ s$.

Before stating the major theorem of [6] that SCNP reduction pairs are reduction pairs, we first have to clarify the notion of reduction pair: In [6, Sec. 2] a non standard definition of a reduction pair is used which differs from Def. 2.1. To distinguish between both kinds of reduction pairs we call the ones of [6] *typed reduction pairs*.

▶ **Definition 5.6** (Typed reduction pair [6])**.** A *typed reduction pair* is a reduction pair $(\succsim, \succ)$ over $\mathcal{T}(\mathcal{F}, \mathcal{V})$ with the additional condition that $\succsim$ compares either two tuple terms or two base terms.

So, the major theorem of [6] states that whenever $(\succsim, \succ)$ is a typed reduction pair, then so is $(\succsim^{\mathcal{L},\mu}, \succ^{\mathcal{L},\mu})$ where μ is one of the four mentioned multiset extensions.

The major problem in the formalization of exactly this theorem required a link between the two notions of reduction pairs since all other theorems working with reduction pairs in IsaFoR are using Def. 2.1.

At this point, it turned out that there are problems with Def. 5.6. Of course, to use the major theorem of [6] one needs a typed reduction pair to start from. Unfortunately, common reduction pairs like RPO are not typed reduction pairs w.r.t. Def. 5.6. For example, if $F \in \mathcal{F}^\sharp$ then $F(F(x)) \succsim^{grpo} F(F(x))$, but then $\succsim$ does not satisfy the additional condition of Def. 5.6, since here two terms are in relation which are neither base terms nor tuple terms.

A possible solution might be to require that $(\succsim, \succ)$ is just a reduction pair and then show that $(\succsim^{\mathcal{L},\mu}, \succ^{\mathcal{L},\mu})$ is a typed reduction pair. However, even with this adaptation the problems remain, since Def. 5.6 itself is flawed: assume $(\succsim, \succ)$ is a typed reduction pair and F is a non-constant tuple symbol. Since $\succsim$ is a quasi-ordering (on tuple-terms) it is reflexive, and thus $F(\vec{t}) \succsim F(\vec{t})$ for every list $\vec{t}$ of base terms. By monotonicity of $\succsim$ also $F(\dots, F(\vec{t}), \dots) \succsim F(\dots, F(\vec{t}), \dots)$ must hold, in contradiction to the condition that $\succsim$ only compares base terms or tuple terms. So, there is a severe problem in demanding both monotonicity of $\succsim$ and the additional condition of Def. 5.6.

Repairing Def. 5.6 by only requiring monotonicity w.r.t. $\mathcal{F}^b$ is also no solution, since for using reduction pairs in the reduction pair processor, it is essential that $\succsim$ is also closed under $\mathcal{F}^\sharp$-contexts.

For a proper fix of SCNP reduction pairs, note that the distinction between tuple terms and base terms in [6] is solely performed, to have two kinds of orderings: $\succsim$ for orienting rules in $\mathcal{R}$, and $\succsim_\mu^{\mathbb{N}}$ and $\succ_\mu^{\mathbb{N}}$ for orienting rules of $\mathcal{P}$.

However, there is already a notion which allows the usage of different orderings for orientation of $\mathcal{P}$ and $\mathcal{R}$, namely *reduction triples*. The advantage of this notion is the fact that it does not require any distinction between base and tuple terms.

▶ **Definition 5.7** (Reduction triple, [14])**.** A *reduction triple* is a triple $(\succsim, \succsim_\top, \succ_\top)$ such that $(\succsim, \succ_\top)$ is a reduction pair and $(\succsim_\top, \succ_\top)$ is a non-monotone reduction pair.

Reduction triples can be used instead of reduction pairs in Thm. 5.1: in [14] it is shown that whenever $\mathcal{P} \subseteq \succ_\top \cup \succsim_\top$ and $\mathcal{R} \subseteq \succsim$ then $\mathcal{P}$ can be replaced by $\mathcal{P} \setminus \succ_\top$ in the DP problem $(\mathcal{P}, \mathcal{R})$. The proof is similar to the one of Thm. 5.1 and also in IsaFoR it was easy to switch from reduction pairs to reduction triples. Hence, the obvious attempt is to define SCNP reduction pairs as reduction triples. As there is no distinction of base terms and tuple terms, we will not run into problems that are caused by the required monotonicity of $\succsim$.

▶ **Definition 5.8** (SCNP reduction triple)**.** Let μ be a multiset extension and $\mathcal{L}$ be a level mapping. Let $(\succsim, \succ)$ be a reduction pair.

We define $\succsim_\top$ and $\succ_\top$ as $t \succsim_\top s$ iff $\mathcal{L}(t) \succsim^{\mathbb{N}}_\mu \mathcal{L}(s)$, and $t \succ_\top s$ iff $\mathcal{L}(t) \succ^{\mathbb{N}}_\mu \mathcal{L}(s)$. Then the *SCNP reduction triple* is defined as $(\succsim, \succsim_\top, \succ_\top)$.

If we are able to prove that every SCNP reduction triple is indeed a reduction triple, then we are done. Unfortunately, it turns out that an SCNP reduction triple is not a reduction triple, where the new problem is that compatibility between $\succ_\top$ and $\succsim$ cannot be ensured. As an example, consider a reduction pair $(\succsim, \succ)$ which is defined via an RPO with precedence $b > a$ and status $\tau(F) = lex$. Moreover, let the level-mapping be defined via $\pi(F) = \{\{\langle 2, 0\rangle\}\}$ and take $\mu = gms$. Then $F(a, b) \succ_\top F(b, a)$ since $\mathcal{L}(F(a, b)) = \{\{\langle b, 0\rangle\}\} \succ^{\mathbb{N}}_{gms} \{\{\langle a, 0\rangle\}\} = \mathcal{L}(F(b, a))$. Furthermore, $F(b, a) \succsim F(a, b)$. However, if $\succ_\top$ and $\succsim$ were compatible, then we would be able to conclude $F(a, b) \succ_\top F(a, b)$, a contradiction.

To this end, we finally have defined a weaker notion than reduction triples which can still be used like reduction triples.

▶ **Definition 5.9** (Root reduction triple)**.** A *root reduction triple* is a triple $(\succsim, \succsim_\top, \succ_\top)$ such that $(\succsim_\top, \succ_\top)$ is a non-monotone reduction pair, $\succsim$ is a stable and monotone quasi-ordering, and whenever $s \succsim t$ then $f(v_1, \ldots, v_{i-1}, s, v_{i+1}, \ldots, v_n) \succsim_\top f(v_1, \ldots, v_{i-1}, t, v_{i+1}, \ldots, v_n)$.

Note that every reduction triple $(\succsim, \succsim_\top, \succ_\top)$ is also a root reduction triple provided that $\succsim \subseteq \succsim_\top$—and as far as we know this condition is satisfied for all reduction triples that are currently used in termination tools. Moreover, root reduction triples can be used in the same way as reduction triples to remove pairs which has been proved in IsaFoR.

▶ **Theorem 5.10** (Root reduction triple processor)**.** *Let $(\succsim, \succsim_\top, \succ_\top)$ be a root reduction triple. Whenever $\mathcal{P} \subseteq \succ_\top \cup \succsim_\top$, $\mathcal{R} \subseteq \succsim$, and $(\mathcal{P} \setminus \succ_\top, \mathcal{R})$ is finite, then $(\mathcal{P}, \mathcal{R})$ is finite.*

Hence, the notion of root reduction triple seems useful for termination proving. And indeed, it turns out that each SCNP reduction triple is a root reduction triple which finally shows that SCNP reduction triples can be used to remove pairs from DP problems.[7]

▶ **Theorem 5.11.** *Every SCNP reduction triple is a root reduction triple.*

Thm. 5.11 is formally proved within IsaFoR, theory *SCNP*. Note that this formalization was straightforward, once the notion of root reduction triple was available: the whole formalization takes only 310 lines. It also includes the feature of ϵ-arguments—an extension of size-change graphs which is mentioned in both [23] and [6]—and it contains results on C_e-compatibility and compatibility w.r.t. to argument filterings. The latter results are important when dealing with usable rules, cf. [12] for further details.

Regarding the formalization of multiset extensions—which is orthogonal to the formalization of SCNP reduction triple—we were able to integrate three of the four mentioned multiset extensions. However, for the multiset extension *dms* it is essential that the signature $\mathcal{F}$ is finite as otherwise strong normalization is not necessarily preserved, cf. Ex. 5.12. Here, the essential issue is that without an explicit bound on the sizes of the multiset, strong normalization is lost (such a bound is explicitly demanded in [3], but not in [6]).

▶ **Example 5.12.** In [6] it is assumed that the *initial* TRS is finite. Hence, also the initial signature is finite which in turn gives a bound on the sizes of the constructed multisets.

[7] An alternative—but unpublished—fix to properly define SCNP reduction pairs has been developed by Carsten Fuhs. It is based on typed term rewriting. (private communication)

However, for the soundness of SCNP reduction pairs it is essential that the signature of the *current* system is finite, since otherwise the following steps would be a valid termination proof for the TRS $\mathcal{R} = \{\mathsf{a} \to \mathsf{a}\}$ as there is no bound on the sizes of multisets.

- Build the initial DP problem $(\{\mathsf{A} \to \mathsf{A}\}, \mathcal{R})$.
- Replace this DP problem by the DP problem $\mathcal{D} = (\{\mathsf{A}_i(x, \ldots, x) \to \mathsf{A}_{i+1}(x, \ldots, x) \mid i \in \mathbb{N}\}, \emptyset)$ where the arity of each A_i is i. This step is sound, as $\mathcal{D}$ is not finite.
- Replace $\mathcal{D}$ by $(\emptyset, \emptyset)$. This can be done using the SCNP reduction pair where $\mu = dms$, $\pi(\mathsf{A}_i) = \{\langle j, 0\rangle \mid 1 \le j \le i\}$, and $(\succsim, \succ)$ is any reduction pair. The reason is that

$$\{\{\underbrace{\langle x, 0\rangle, \ldots, \langle x, 0\rangle}_{i \text{ times}}\}\} \succ^{dms} \{\{\underbrace{\langle x, 0\rangle, \ldots, \langle x, 0\rangle}_{i+1 \text{ times}}\}\}.$$

The demand for a bound on the signature for *dms* resulted in a small problem in our formalization, since in IsaFoR we never have finite signatures: for each symbol f we directly include infinitely many f's, one for each possible arity in $\mathbb{N}$. So, in principle there might not be any bound on the size of the multisets that are constructed by the level mapping. Hence, we use a slightly different version of *dms* which includes some fixed bound n on the size of the multisets: we define $M \mathrel{(\succsim)_{dms\text{-}n}} N$ iff $M \mathrel{(\succsim)_{dms}} N$ and $|N| \le n \vee |N| = |M|$. Hence, *dms-n* is a restriction of *dms* where either the sizes of the multisets are bounded by n or where multisets of the same sizes are compared.

Note that the additional explicit restriction of $|N| \le n$ ensures strong normalization even for infinite $\mathcal{F}$ or unbounded multisets. The other alternative $|N| = |M|$ is added, as otherwise reflexivity of $\succsim_{dms\text{-}n}$ is lost, for example $\{\{\underbrace{x, \ldots, x}_{n+1}\}\} \succsim_{dms\text{-}n} \{\{\underbrace{x, \ldots, x}_{n+1}\}\}$ would no longer hold.

In practice, the difference between *dms* and *dms-n* can be neglected. As for the certification we only consider finite TRSs, we just precompute a suitable large enough number n such that for the resulting constraints *dms* and *dms-n* coincide.

We conclude this section by showing that certification of SCNP reduction triples is NP-hard in the case of $\mu \in \{gms, dms\}$ which also implies that deciding $\succ_{dms}$ is NP-hard.

▶ **Theorem 5.13.** *Given a SCNP reduction triple* $(\succsim, \succsim_\top, \succ_\top)$ *and* $\mu \in \{gms, dms\}$*, the problem of deciding* $\ell \succ_\top r$ *for two terms* ℓ *and* r *is NP-hard.*

Proof. For $\mu = gms$ we use nearly the identical reduction from SAT as we used in the proof of Thm. 4.2. To be more precise, for each formula ϕ we use the exactly the same terms ℓ and r and the same multisets L and R as before. Moreover, we define the level-mapping by choosing $\pi(\mathsf{g}) = \{\langle i, 0\rangle \mid 1 \le i \le 2n\}$ and $\pi(\mathsf{h}) = \{\langle i, 0\rangle \mid 1 \le i \le n + m\}$. Then for $L' = \{\{\langle s, 0\rangle \mid s \in L\}\}$ and $R' = \{\{\langle s, 0\rangle \mid s \in R\}\}$ we obtain $\ell \succ_\top r$ iff $L' \succ^{\mathbb{N}}_{gms} R'$ iff $L \succ_{gms} R$. It remains to choose $\succ$ as the MPO within the proof of Thm. 4.2. Then $L \succ_{gms} R$ iff ϕ is satisfiable, so in total, $\ell \succ_\top r$ iff ϕ is satisfiable.

Furthermore, it can easily be argued that NP-hardness is not a result of our extended definition of MPO: we also achieve $L \succ_{gms} R$ iff ϕ is satisfiable if we define $\succ$ by a polynomial interpretation $\mathcal{P}ol$ with $\mathcal{P}ol(\mathsf{f}(x_0, \ldots, x_m)) = 1 + x_0 + \cdots + x_m$ and $\mathcal{P}ol(\mathsf{a}) = 0$.

For $\mu = dms$ one can use a similar reduction from SAT: satisfiability of ϕ is equivalent to $\ell \succ_\top r$ using the same level-mapping as before, but where now $\ell = \mathsf{h}(x_1 \vee \overline{x_1}, \ldots, x_n \vee \overline{x_n}, \mathsf{s}(C'_1), \ldots, \mathsf{s}(C'_m))$, $r = \mathsf{g}(x_1, \overline{x_1}, \ldots, x_n, \overline{x_n})$, and $\succ$ is defined as the polynomial interpretation $\mathcal{P}ol$ where $\mathcal{P}ol(\mathsf{s}(x)) = 1 + x$ and $\mathcal{P}ol(x \vee y) = x + y$. Here, each C'_i is a representation of clause C_i as disjunction. ◀

6 Certification Algorithms

For the certification of existing proofs using RPO and SCNP reduction pairs there are in principle two possibilities, which we will explain using RPO.

The first solution for certifying $s \succ^{grpo} t$ is to use a shallow embedding. In this approach, some untrusted tool figures out how the inference rules of RPO have to be applied and generates a proof script that can then be checked by the proof assistant. This solution cannot be used in our case, since CeTA is obtained from IsaFoR via code generation.

The second solution is to use a deep embedding where an algorithm for deciding RPO is developed within the proof assistant in combination with a soundness proof. Then this algorithm is amenable for code generation, and such an algorithm is also used in related certifiers [5, 7] where it is accessible via reflection. Hence, we had to develop a function for deciding RPO constraints within Isabelle. In our case, we have written a function $grpo^{\tau}_{>}$ for RPO, which is parametrized by a precedence $>$ and a status τ. It takes two terms s and t as input and returns the pair $(s \succ^{grpo} t, s \succsim^{grpo} t)$. In fact, we even defined RPO via $grpo^{\tau}_{>}$ and only later on derived the inference rules that have been presented in Sec. 4.

Since we proved several properties of RPO directly via $grpo^{\tau}_{>}$ (theory *RPO*), we implemented $grpo^{\tau}_{>}$ in a straightforward way as recursive function. As a result, $grpo^{\tau}_{>}$ has exponential runtime, since no sharing of identical subcalls is performed. To this end, we developed a second function for RPO, that has been proved to be equivalent to $grpo^{\tau}_{>}$, but where memoization is integrated—a well known technique where intermediate results are stored to avoid duplicate computations. In principle, this is an easy programming task, but since we use a deep embedding, we had to formally prove correctness of this optimization.

To stay as general as possible, the memoized function was implemented independent of the actual data structure used as memory. The interface we use for a memory is as follows. We call a memory valid w.r.t. to a function f if all entries in the memory are results of f. Moreover we require functionality for looking up a result in the memory and for storing a new result with the obvious soundness properties: storing a correct entry in a valid memory yields a valid memory and looking up an entry in a valid memory returns a correct result, i.e., the same result that would have been computed by f.

Assuming we have such a memory at our disposal the idea of memoizing is straightforward: before computing a result we do a lookup in the memory and if an entry is found we return it and leave the memory unchanged. Otherwise we compute as usual, store the result in the memory and return both.

The main difficulty was that all recursive calls of $grpo^{\tau}_{>}$ are indirect via higher-order functions like the computation of the multiset- and lexicographic extension of a relation. Consequently results of recursive calls to RPO are not available directly, but only in these higher-order functions. Thus, they have to take care of storing results in the memory and of passing it on to the next RPO call. Hence, new memoizing versions of all these higher-order functions had to be implemented and their soundness had to be proved. Soundness meaning that, when given equivalent functions as arguments, they compute the same results as their counterparts without memory handling, and that given a valid memory as input they return a valid memory in addition to their result. For further details we refer to theory *Efficient-RPO*.

Concerning the required decision procedures for *gms* and *dms* which have to find suitable splittings, we used a branch-and-bound approach.

We tested our certification algorithms by rerunning all experiments that have been performed in [6]. Here, AProVE tries to prove termination of 1,381 TRSs from the termination problem database (TPDB version 7.0) using 20 different strategies where in 12 cases SCNP

reduction pairs are used and full size-change termination is tried 4 times. Here, in 16 strategies RPO or weaker orderings have been tried. As in [6] we used a timeout of 60 seconds and although we used a different computer than in [6], on the $20 \times 1,381$ termination problems, there are only 8 differences in both experiments—all due to a timeout.

By performing the experiments we were able to detect a bug in the proof output of AProVE—the usable rules have not been computed correctly. After a corresponding fix indeed all 9,025 generated proofs could be certified by CeTA (version 2.2).

The experiments contain 432 proofs where the usage of $\succ_{gms}$ and $\succsim_{gms}$ was essential, i.e., where the constraints could not be oriented by $\succ_{ms}$ and $\succeq_{ms}$. If one allows equality modulo the equivalence relation induced by the ordering, then still 39 proofs require $\succ_{gms}$ and $\succsim_{gms}$.

Regarding the time required for certification, although it is NP-complete, in our experiments, certification is much faster than proof search. The reason is that our certification algorithms are only exponential in the arity of the function symbols, and in the experiments the maximal arity was 7. In numbers: AProVE required more than 31 hours ($\approx$ 4 seconds per example) whereas CeTA was done in below 6 minutes ($\approx$ 0.04 seconds per example).

The experiments also show that proofs using SCNP reduction pairs can indeed be certified faster than proofs using full size-change termination. In average, the latter proofs require 50 % more time for certification than the former.

All details of our experiments are available at `http://cl-informatik.uibk.ac.at/software/ceta/experiments/multisets/`.

7 Summary

We have studied the generalization of the multiset ordering which is generated by two orderings: a strict one and a compatible non-strict one. Indeed this generalization preserves most properties of the standard multiset ordering, where we only detected one difference: the decision problem becomes NP-complete.

Concerning termination techniques that depend on multiset orderings, we formalized and corrected an extended variant of RPO that is used within AProVE, and we formalized and corrected SCNP reduction pairs. Certification of both techniques is NP-hard.

Acknowledgments

We thank Carsten Fuhs for helpful discussions and his support in the experiments, and we thank the anonymous referees for their helpful comments.

References

1 T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theor. Comput. Sci.*, 236(1-2):133–178, 2000.

2 F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, Cambridge, 1998.

3 A. M. Ben-Amram and M. Codish. A SAT-based approach to size change termination with global ranking functions. In *Proc. TACAS '08*, volume 4963 of *LNCS*, pages 218–232, 2008.

4 A. M. Ben-Amram and C. Soon Lee. Program termination analysis in polynomial time. *ACM Trans. Program. Lang. Syst.*, 29(1), 2007.

5 F. Blanqui and A. Koprowski. CoLoR: a Coq library on well-founded rewrite relations and its application on the automated verification of termination certificates. *Mathematical Structures in Computer Science*, 21(4):827–859, 2011.

6 M. Codish, C. Fuhs, J. Giesl, and P. Schneider-Kamp. Lazy abstraction for size-change termination. In *Proc. LPAR '10*, volume 6397 of *LNCS*, pages 217–232, 2010.

7 E. Contejean, P. Courtieu, J. Forest, O. Pons, and X. Urbain. Certification of automated termination proofs. In *Proc. FroCoS '07*, volume 4720 of *LNAI*, pages 148–162, 2007.

8 N. Dershowitz. Orderings for term-rewriting systems. *Theor. Comput. Sci.*, 17:279–301, 1982.

9 N. Dershowitz. Termination of rewriting. *J. Symb. Comp.*, 3(1-2):69–116, 1987.

10 N. Dershowitz and Z. Manna. Proving termination with multiset orderings. *Comm. ACM*, 22(8):465–476, 1979.

11 J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic termination proofs in the dependency pair framework. In *Proc. IJCAR '06*, volume 4130 of *LNAI*, pages 281–286, 2006.

12 J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Mechanizing and improving dependency pairs. *J. Autom. Reason.*, 37(3):155–203, 2006.

13 F. Haftmann and T. Nipkow. Code generation via higher-order rewrite systems. In *Proc. FLOPS '10*, volume 6009 of *LNCS*. Springer, 2010.

14 N. Hirokawa and A. Middeldorp. Tyrolean Termination Tool: Techniques and features. *Inf. Comput.*, 205(4):474–511, 2007.

15 J.-P. Jouannaud and A. Rubio. The higher-order recursive path ordering. In *Proc. LICS '99*, pages 402–411. IEEE Computer Society Press, 1999.

16 A. Koprowski. Coq formalization of the higher-order recursive path ordering. *Appl. Algebra Eng. Commun. Comput.*, 20(5-6):379–425, 2009.

17 K. Korovin and A. Voronkov. Orienting rewrite rules with the Knuth-Bendix order. *Inf. Comput.*, 183(2):165–186, 2003.

18 M. S. Krishnamoorthy and P. Narendran. On recursive path ordering. *Theor. Comput. Sci.*, 40:323–328, 1985.

19 C. S. Lee, N. D. Jones, and A. M. Ben-Amram. The size-change principle for program termination. In *Proc. POPL '01*, pages 81–92, 2001.

20 T. Nipkow. An inductive proof of the wellfoundedness of the multiset order, 1998. Available at `http://www4.in.tum.de/~nipkow/misc/multiset.ps`.

21 T. Nipkow, L. Paulson, and M. Wenzel. *Isabelle/HOL – A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, Berlin-Heidelberg, 2002.

22 P. Schneider-Kamp, R. Thiemann, E. Annov, M. Codish, and J. Giesl. Proving termination using recursive path orders and SAT solving. In *Proc. FroCoS '07*, volume 4720 of *LNAI*, pages 267–282, 2007.

23 R. Thiemann and J. Giesl. The size-change principle and dependency pairs for termination of term rewriting. *Appl. Alg. Eng. Comm. Comput.*, 16(4):229–270, 2005.

24 R. Thiemann and C. Sternagel. Certification of termination proofs using CeTA. In *Proc. TPHOLs '09*, volume 5674 of *LNCS*, pages 452–468, 2009.