Tuple Interpretations for Termination of Term Rewriting

Akihisa Yamada

Cyber Physical Security Research Center (CPSEC), National Institute of Advanced Industrial Science and Technology (AIST), 2-3-26, Aomi, Koto-ku, Tokyo, 135-0064, Japan.

Contributing authors: akihisa.yamada@aist.go.jp;

Abstract

Interpretation methods constitute a foundation of the termination analysis of term rewriting. From time to time remarkable instances of interpretation methods appeared, such as polynomial interpretations, matrix interpretations, arctic interpretations, and their variants. In this paper we introduce a general framework, the tuple interpretation method, that subsumes these variants as well as many previously unknown interpretation methods as instances. Employing the notion of derivers, we prove the soundness of the proposed method in an elegant way. We implement the proposed method in the termination prover NaTT and verify its significance through experiments.

1 Introduction

Term rewriting [2] is a formalism for reasoning about function definitions or functional programs. For instance, a term rewrite system (TRS) $\mathcal{R}_{\mathtt{fact}}$ [8] consisting of the following rewrite rules defines the factorial function:

$$\mathtt{fact}(\mathtt{0}) \to \mathtt{s}(\mathtt{0}) \quad \mathtt{fact}(\mathtt{s}(x)) \to \mathtt{mul}(\mathtt{s}(x), \mathtt{fact}(\mathtt{p}(\mathtt{s}(x)))) \quad \mathtt{p}(\mathtt{s}(x)) \to x$$

assuming that `s`, `p`, and `mul` are interpreted respectively as the successor, predecessor, and multiplication functions.[1]

[1] For clarity, we omit the rules that define `mul`; the analyses we make on $\mathcal{R}_{\mathtt{fact}}$ later on can be easily extended for coping with those rules.

Analyzing whether a TRS *terminates*, meaning that the corresponding functional program responds or the function is well defined, has been an active research area for decades. Consequently, several fully automatic termination provers have been developed, e.g., AProVE [11], TTT2 [23], CiME [6], MU-TERM [15], and NaTT [37], and have been competing in the annual Termination Competitions (termCOMP) [12].

Throughout their history, interpretation methods [27] have been foundational in termination analysis. They are categorized by the choice of well-founded carriers and the class of functions as which symbols are interpreted. *Polynomial interpretations* [25] use the natural numbers $\mathbb{N}$ as the carrier and interpretations are monotone polynomials, i.e., every variable has coefficient at least 1. Weakly monotone polynomials, i.e., zero coefficients, are allowed in the *dependency pair* method [1]. *Negative constants* are allowed using the max operator [17]. General combinations of polynomials and the max operator are proposed in both the standard [40] and the dependency pair settings [10]. *Negative coefficients* and thus non-monotone polynomials are also allowed, but in a more elaborated theoretical framework [17, 10].

These methods share the common carrier $\mathbb{N}$. In contrast, *matrix interpretations* [18, 9] choose vectors over $\mathbb{N}$ as the carrier, and interpret symbols as affine maps over it. Although the carrier is generalized, matrix interpretations do not properly generalize polynomial interpretations, since not all polynomials are affine. This gap can be filled by *improved matrix interpretations* [7], that further generalize the carrier to square matrices, so that natural polynomial interpretations can be subsumed by matrix polynomials over 1×1 matrices. In *arctic interpretations* [22], the carrier consists of vectors over arctic naturals ($\mathbb{N} \cup \{-\infty\}$) or integers ($\mathbb{Z} \cup \{-\infty\}$), and interpretations are affine maps over it, where affinity is with respect to the *max/plus semiring*.

Having this many variations would be welcome if you are a user of a termination tool in which someone else has already implemented all of them. It would not be so if you are the developer of a termination tool in which you will have to implement all of them. Also, to ultimately trust termination tools, one needs to formalize proof methods using proof assistants and obtain a trusted certifier that validates outputs of termination tools, see, e.g., the IsaFoR/CeTA [33] or CoLoR/Rainbow [5] frameworks. Although some interpretation methods have already been formalized [30, 32], adding missing variants one by one would cost a significant effort.

In this paper, we introduce a general framework for interpretation methods, which subsumes most of the above-mentioned methods as instances, namely, (max-)polynomial interpretations (with negative constants), (improved) matrix interpretations, and arctic interpretations, as well as a syntactic method called *argument filtering* [1, 24]. Moreover, we obtain a bunch of previously unexplored interpretation methods as other instances.

After preliminaries, we start with a convenient fact about *reduction pairs*, a central tool in termination proving with dependency pairs (Section 3).

The first step to the main contribution is the use of *derivers* [26, 35], which allow us to abstract away the mathematical details of polynomials or max-polynomials. We will obtain a key soundness result that derivers derive monotone interpretations from monotone interpretations (Section 4).

The second step is to extend interpretations to *tuple interpretations* (Section 5). We show that the tuple interpretation induced by a monotone interpretation is again monotone. This setting further generalizes (improved) matrix interpretations, so that max-polynomials, negative constants, and negative entries are allowed. It will also be indicated that tuple interpretations can emulate the effect of negative coefficients, although they are not strictly subsumed.

As *strict monotonicity* is crucial for proving termination without dependency pairs, and is still useful with dependency pairs, we will see how to ensure strict monotonicity of tuple interpretations (Section 6). At this point, the convenient fact we have seen in Section 3 becomes crucial.

We also show that our approach subsumes arctic interpretations by adding a treatment for $-\infty$ (Section 7). Although the original formulation by Koprowski and Waldmann [22] has some intricacies, we will present a simpler formulation and prove that our formulation is sufficient in the dependency pair setting.

Then we present a template-based search method for tuple interpretations, and show how the search is reduced to satisfiability modulo theory (SMT) solving (Section 8). The proposed method is implemented in the termination prover NaTT, and experimental results are reported (Section 9). We evaluate various instances of tuple interpretations, some corresponding to known interpretation methods and many others not. We choose two new instances to integrate to the NaTT strategy. The new strategy proved the termination of 20 more benchmarks than the old one, and six of them were not proved by any tool in termCOMP 2020.

A preliminary version of this paper was presented at the 28th International Conference on Automated Deduction (CADE-28) [36]. In the present version, besides improvements in notations, we stated and proved that our method subsumes existing ones (Corollaries 2, 3, etc.), introduced *tuple algebras* in order to formulate tuple interpretations elegantly (Definition 6), concluded that negative coefficients are not subsumed (Example 9), presented the formal account of the template-based implementation (Section 8), and provided extensive report on experiments (Section 9). We adopt the term "tuple interpretation" coined by Kop and Vale for essentially the same idea, independently in the context of higher-order complexity analysis [21]. Their method analyzes many-sorted TRSs using single-sorted tuple algebras, while we analyze single-sorted TRSs using many-sorted tuple algebras.

2 Preliminaries

We start with *order-sorted* algebras. Let $\mathcal{S} = \langle S, \sqsubseteq \rangle$ be a partially ordered set, where elements in S are called *sorts* and $\sqsubseteq$ is called the *subsort relation*. An

$\mathcal{S}$-*sorted set* is an S-indexed family $A = \{A_\sigma\}_{\sigma \in S}$ such that $\sigma \sqsubseteq \tau$ implies $A_\sigma \subseteq A_\tau$. A sorted set may be viewed as a set, where $a : \sigma \in A$ means $a \in A_\sigma$. A *sorted map* between $\mathcal{S}$-sorted sets A and B is a mapping f, written $f : A \to B$, such that $x : \sigma \in A$ implies $f(x) : \sigma \in B$.

An $\mathcal{S}$-*sorted signature* is an $S^* \times S$-indexed family $\mathcal{F} = \{\mathcal{F}_{\vec{\sigma},\tau}\}_{\langle \vec{\sigma},\tau\rangle \in S^* \times S}$ of function symbols.[2] We also write $f : \sigma_1 \times \cdots \times \sigma_n \to \tau \in \mathcal{F}$ to mean $f \in \mathcal{F}_{(\sigma_1,\ldots,\sigma_n),\tau}$, and in this case, we say f has *rank* $\sigma_1 \times \cdots \times \sigma_n \to \tau$ and *arity* n in $\mathcal{F}$. We may also write $f : \tau \in \mathcal{F}$ when $n = 0$.

Example 1 Consider sort $\mathtt{Nat}$. We define the following $\{\mathtt{Nat}\}$-sorted signatures:

- $\mathcal{N} := \{\mathtt{0} : \mathtt{Nat},\ \mathtt{1} : \mathtt{Nat},\ \mathtt{2} : \mathtt{Nat},\ \ldots\}$
- $\mathcal{N}_* := \mathcal{N} \cup \{\mathtt{*} : \mathtt{Nat} \times \mathtt{Nat} \to \mathtt{Nat}\}$
- $\mathcal{N}_+ := \mathcal{N} \cup \{\mathtt{+} : \mathtt{Nat} \times \mathtt{Nat} \to \mathtt{Nat}\}$
- $\mathcal{N}_{\mathtt{max}} := \mathcal{N} \cup \{\mathtt{max} : \mathtt{Nat} \times \mathtt{Nat} \to \mathtt{Nat}\}$

Let us abbreviate unions of signatures by concatenations of subscripts: for instance $\mathcal{N}_{*+\mathtt{max}}$ denotes $\mathcal{N}_* \cup \mathcal{N}_+ \cup \mathcal{N}_{\mathtt{max}}$. Next consider sorts $\mathtt{Neg}$ and $\mathtt{Int}$ with $\mathtt{Nat}, \mathtt{Neg} \sqsubseteq \mathtt{Int}$. We define the following $\{\mathtt{Nat}, \mathtt{Neg}, \mathtt{Int}\}$-sorted signatures:

- $\mathcal{Z} := \mathcal{N} \cup \{\mathtt{0} : \mathtt{Neg},\ \mathtt{-1} : \mathtt{Neg},\ \mathtt{-2} : \mathtt{Neg},\ \ldots\}$
- $\mathcal{Z}_* := \mathcal{Z} \cup \mathcal{N}_* \cup \{\mathtt{*} : \mathtt{Neg} \times \mathtt{Neg} \to \mathtt{Nat},\ \mathtt{*} : \mathtt{Int} \times \mathtt{Int} \to \mathtt{Int}\}$
- $\mathcal{Z}_+ := \mathcal{Z} \cup \mathcal{N}_+ \cup \{\mathtt{+} : \mathtt{Neg} \times \mathtt{Neg} \to \mathtt{Neg},\ \mathtt{+} : \mathtt{Int} \times \mathtt{Int} \to \mathtt{Int}\}$
- $\mathcal{Z}_{\mathtt{max}} := \mathcal{Z} \cup \mathcal{N}_{\mathtt{max}} \cup$
  $\{\mathtt{max} : \mathtt{Nat} \times \mathtt{Int} \to \mathtt{Nat},\ \mathtt{max} : \mathtt{Int} \times \mathtt{Nat} \to \mathtt{Nat},\ \mathtt{max} : \mathtt{Int} \times \mathtt{Int} \to \mathtt{Int}\}$

For an $\mathcal{S}$-sorted signature $\mathcal{F}$, an $\mathcal{F}$-*algebra* $\langle A, [\cdot]\rangle$ consists of an $\mathcal{S}$-sorted set A called the *carrier* and a family $[\cdot]$ of mappings called the *interpretation* such that $[f] : A_{\sigma_1} \times \cdots \times A_{\sigma_n} \to A_\tau$ whenever $f : \sigma_1 \times \cdots \times \sigma_n \to \tau \in \mathcal{F}$.

Example 2 We consider the following *standard* interpretation $[\![\cdot]\!]$:

$$\cdots \quad [\![\mathtt{-2}]\!] := -2 \quad [\![\mathtt{-1}]\!] := -1 \quad [\![\mathtt{0}]\!] := 0 \quad [\![\mathtt{1}]\!] := 1 \quad [\![\mathtt{2}]\!] := 2 \quad \cdots$$
$$[\![\mathtt{*}]\!](a,b) := a \cdot b \qquad [\![\mathtt{+}]\!](a,b) := a + b \qquad [\![\mathtt{max}]\!](a,b) := \max(a,b)$$

Notice that $\langle \mathbb{N}, [\![\cdot]\!]\rangle$ is an $\mathcal{N}_{*+\mathtt{max}}$-algebra and $\langle \mathbb{Z}, [\![\cdot]\!]\rangle$ is a $\mathcal{Z}_{*+\mathtt{max}}$-algebra. Here, the $\{\mathtt{Nat}\}$-sorted set $\mathbb{N}$ is defined by $\mathbb{N}_{\mathtt{Nat}} := \mathbb{N}$ and the $\{\mathtt{Nat}, \mathtt{Neg}, \mathtt{Int}\}$-sorted set $\mathbb{Z}$ is defined by $\mathbb{Z}_{\mathtt{Nat}} := \mathbb{N}$, $\mathbb{Z}_{\mathtt{Neg}} := \{0, -1, -2, \ldots\}$ and $\mathbb{Z}_{\mathtt{Int}} := \mathbb{Z}$.

Sorted Terms:

Given an $\mathcal{S}$-sorted signature $\mathcal{F}$ and an $\mathcal{S}$-sorted set $\mathcal{V}$ of *variables*, the $\mathcal{S}$-sorted set $\mathcal{T}(\mathcal{F}, \mathcal{V})$ of *terms* is inductively defined as follows:

[2] In the literature, sorted signatures are given more assumptions such as monotonicity or regularity. For the purpose of this paper, these assumptions are not necessary.

- $v : \sigma \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ if $v : \sigma \in \mathcal{V}$;
- $f(s_1, \ldots, s_n) : \rho \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ if $s_1 : \sigma_1, \ldots, s_n : \sigma_n \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, $f : \sigma_1 \times \cdots \times \sigma_n \to \tau \in \mathcal{F}$, and $\tau \sqsubseteq \rho$.

An interpretation $[\cdot]$ is extended to terms as follows: given a sorted map $\alpha : \mathcal{V} \to A$, $[v]\alpha := \alpha(v)$ if $v : \sigma \in \mathcal{V}$, and $[f(s_1, \ldots, s_n)]\alpha := [f]([s_1]\alpha, \ldots, [s_n]\alpha)$. The *term algebra* is the $\mathcal{F}$-algebra $\langle \mathcal{T}(\mathcal{F}, \mathcal{V}), \cdot \rangle$, interpreting f as the mapping that takes $(s_1, \ldots, s_n)$ and returns $f(s_1, \ldots, s_n)$. A *substitution* is a sorted map $\theta : \mathcal{V} \to \mathcal{T}(\mathcal{F}, \mathcal{V})$, and the term obtained by replacing every variable x by $\theta(x)$ in s is $s\theta$.

Term Rewriting:

This paper is concerned with the termination analysis of *plain* term rewriting. In this setting, there is only one sort o, and we may identify a $\{\mathsf{o}\}$-sorted set A and the set A_{o}. The set of variables appearing in a term s is denoted by $\mathsf{Var}(s)$. A *context* is a term $C \in \mathcal{T}(\mathcal{F}, \mathcal{V} \cup \{\Box\})$ where a special variable $\Box$ occurs exactly once. Given $s \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, we denote by $C[s]$ the term $C(\Box \mapsto s) \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, i.e., the term obtained by substituting $\Box$ by s in C.

A *rewrite rule* is a pair of terms l and r, written $l \to r$, such that $l \notin \mathcal{V}$, $\mathsf{Var}(l) \supseteq \mathsf{Var}(r)$. A *term rewrite system (TRS)* is a set $\mathcal{R}$ of rewrite rules. It induces the *root rewrite step* $\xrightarrow[\mathcal{R}]{\epsilon}$ and the *rewrite step* $\xrightarrow[\mathcal{R}]{}$ as the least relations such that $l\theta \xrightarrow[\mathcal{R}]{\epsilon} r\theta$ and $C[l\theta] \xrightarrow[\mathcal{R}]{} C[r\theta]$, for any rule $l \to r \in \mathcal{R}$, substitution $\theta : \mathcal{V} \to \mathcal{T}(\mathcal{F}, \mathcal{V})$, and context C. A TRS $\mathcal{R}$ is *terminating* iff no infinite rewriting $s_1 \xrightarrow[\mathcal{R}]{} s_2 \xrightarrow[\mathcal{R}]{} s_3 \xrightarrow[\mathcal{R}]{} \cdots$ is possible.

The dependency pair (DP) framework [1, 16, 14]

is a *de facto* standard among automated termination provers for term rewriting. Here we briefly recapitulate its essence. The *root symbol* of a term $s = f(s_1, \ldots, s_n)$ is f and is denoted by $\mathsf{root}(s)$. The set of *defined* symbols in $\mathcal{R}$ is $\mathcal{D}_{\mathcal{R}} := \{\mathsf{root}(l) \mid l \to r \in \mathcal{R}\}$. We assume a fresh *marked* symbol $f^\sharp$ for every $f \in \mathcal{D}_{\mathcal{R}}$, and write $s^\sharp$ to denote the term $f^\sharp(s_1, \ldots, s_n)$ for $s = f(s_1, \ldots, s_n)$. A *dependency pair* of a TRS $\mathcal{R}$ is a rule $l^\sharp \to r^\sharp$ such that $\mathsf{root}(r) \in \mathcal{D}_{\mathcal{R}}$ and $l \to C[r] \in \mathcal{R}$ for some context C. The set of all dependency pairs of $\mathcal{R}$ is denoted by $\mathsf{DP}(\mathcal{R})$. A *DP problem* $\langle \mathcal{P}, \mathcal{R} \rangle$ is just a pair of TRSs.

Theorem 1 ([1]) *A TRS $\mathcal{R}$ is terminating iff the DP problem $\langle \mathsf{DP}(\mathcal{R}), \mathcal{R} \rangle$ is* finite*, i.e., there is no infinite chain $s_0 \xrightarrow[\mathsf{DP}(\mathcal{R})]{\epsilon} t_0 \xrightarrow[\mathcal{R}]{}^* s_1 \xrightarrow[\mathsf{DP}(\mathcal{R})]{\epsilon} t_1 \xrightarrow[\mathcal{R}]{}^* \cdots$.*

A number of techniques called *DP processors* that simplify or decompose DP problems are proposed; see [14] for a list of such processors. Among them, the central technique for concluding the finiteness of DP problems is the *reduction pair* processor, which will be reformulated in the next section.

3 Notes on Reduction Pairs

A reduction pair is a pair $\langle \succsim, \succ \rangle$ of order-like relations over terms with some conditions. Here we introduce two formulations of reduction pairs, one demanding natural assumptions of orderings, and the other, reduction pair seed, demanding only essential requirements. The first formulation is useful when proving properties of reduction pairs, while the latter is useful when devising new reduction pairs. We will show that the two notions are essentially equivalent: one can always extend a reduction pair seed into a reduction pair of the former sense. Existing formulations of reduction pairs lie strictly in between the two.

Definition 1 (reduction pair) An *order pair* $\langle \succsim, \succ \rangle$ is a pair of a quasi-order $\succsim$ and an irreflexive relation $\succ \subseteq \succsim$ satisfying *compatibility*: $\succsim \circ \succ \circ \succsim \subseteq \succ$, where $\circ$ denotes the relation composition. The order pair is *well-founded* if $\succ$ is well-founded.

A *reduction pair* is a well-founded order pair $\langle \succsim, \succ \rangle$ on terms, such that both $\succsim$ and $\succ$ are closed under substitutions, and $\succsim$ is closed under contexts. Here, a relation $\sqsupset$ is *closed under substitutions (resp. contexts)* iff $s \sqsupset t$ implies $s\theta \sqsupset t\theta$ for every substitution θ (resp. $C[s] \sqsupset C[t]$ for every context C).

The above formulation of reduction pairs is strictly subsumed by standard definitions (e.g., [1, 16, 14]), where $\succ$ is not necessarily a subset of $\succsim$, and compatibility is weakened to either $\succsim \circ \succ \subseteq \succ$ or $\succ \circ \succsim \subseteq \succ$. Instead, $\succ$ is required to be transitive but this follows from our assumptions $\succ \subseteq \succsim$ and compatibility: $\succ \circ \succ \subseteq \succsim \circ \succ \subseteq \succ$. On one hand, this means that we can safely import existing results of reduction pairs into our formulation.

Theorem 2 (reduction pair processor [16, 14]) *Let $\langle \mathcal{P}, \mathcal{R} \rangle$ be a DP problem and $\langle \succsim, \succ \rangle$ be a reduction pair such that $\mathcal{P} \cup \mathcal{R} \subseteq \succsim$. Then the DP problem $\langle \mathcal{P}, \mathcal{R} \rangle$ is finite if and only if $\langle \mathcal{P} \setminus \succ, \mathcal{R} \rangle$ is.*

Example 3 Consider again the TRS $\mathcal{R}_{\mathtt{fact}}$ of the introduction. Proving that $\mathcal{R}_{\mathtt{fact}}$ terminates in the DP framework boils down to finding a reduction pair $\langle \succsim, \succ \rangle$ satisfying (considering *usable rules* [1]):

$$\mathtt{p}(\mathtt{s}(x)) \succsim x \qquad\qquad \mathtt{fact}^\sharp(\mathtt{s}(x)) \succ \mathtt{fact}^\sharp(\mathtt{p}(\mathtt{s}(x)))$$

On the other hand, one may wonder whether Definition 1 might be too restrictive. We justify our formulation by uniformly extending general "reduction pairs" into reduction pairs that comply with Definition 1. This is possible for even more general pairs of relations than standard reduction pairs.

Definition 2 (reduction pair seed) A *well-founded order seed* is a pair $\langle W, S\rangle$ of relations such that S is well-founded and $S \circ W \subseteq S^+$. A *reduction pair seed* is a well-founded order seed on terms such that both W and S are closed under substitutions, and W is closed under contexts.

Now we show that every reduction pair seed $\langle W, S\rangle$ can be extended to a reduction pair $\langle \succsim, \succ\rangle$ such that $W \subseteq \succsim$ and $S \subseteq \succ$. Before that, the assumption $S \circ W \subseteq S^+$ of Definition 2 is generalized as follows.

Lemma 1 *If $\langle W, S\rangle$ is a well-founded order seed, then $S \circ W^* \subseteq S^+$.*

Proof By induction on the number of W steps. □

Theorem 3 *Let $\langle W, S\rangle$ be a well-founded order seed. Then $\langle \succsim, \succ\rangle$ is a well-founded order pair, where $\succsim := (W \cup S)^*$ and $\succ := (W^* \circ S)^+$.*

Proof It is trivial that $\succsim$ is a quasi-order and $\succ \subseteq \succsim$ by definition. We show the well-foundedness of $\succ$ as follows: Suppose on the contrary we have an infinite sequence:

$$a_1\ W^*\ b_1\ S\ a_2\ W^*\ b_2\ S\ a_3\ W^*\ b_3\ S \cdots$$

Then using Lemma 1 ($S \circ W^* \subseteq S^+$) we obtain $a_1\ W^*\ b_1\ S^+\ b_2\ S^+ \cdots$, which contradicts the well-foundedness of S.

Now we show compatibility. By definition we have $\succsim \circ \succ \subseteq \succ$, so it suffices to show $\succ \circ \succsim \subseteq \succ$. By induction we reduce the claim to $\succ \circ (W \cup S) \subseteq \succ$, that is, both $\succ \circ W \subseteq \succ$ and $\succ \circ S \subseteq \succ$. Using $S \circ W \subseteq S^+ = S \circ S^*$ we have

$$\begin{aligned} \succ \circ W \;=\; (W^* \circ S)^+ \circ W \;&=\; (W^* \circ S)^* \circ W^* \circ S \circ W \\ &\subseteq\; (W^* \circ S)^* \circ W^* \circ S \circ S^* \;\subseteq\; \succ \end{aligned}$$

The other case $\succ \circ S \subseteq \succ$ is easy from the definition. □

Now we obtain the following corollary of Theorem 2 and Theorem 3.

Corollary 1 *Let $\langle \mathcal{P}, \mathcal{R}\rangle$ be a DP problem and $\langle W, S\rangle$ a reduction pair seed such that $\mathcal{P} \cup \mathcal{R} \subseteq W$. Then $\langle \mathcal{P}, \mathcal{R}\rangle$ is finite if and only if $\langle \mathcal{P} \setminus S, \mathcal{R}\rangle$ is.*

Notice that Definition 2 does not demand any order-like property, most notably transitivity. This is beneficial when developing new reduction pairs; for instance, *higher-order recursive path orders* [19] are known to be non-transitive, but form a reduction pair seed with their reflexive closure. Throughout the paper we use Definition 1, since it provides more useful and natural properties of orderings, which becomes crucial in Section 6.

4 Interpretation Methods as Derivers

Interpretation methods construct reduction pairs from $\mathcal{F}$-algebras, where $\mathcal{F}$ is the $\{\mathtt{o}\}$-sorted signature of an input TRS or DP problem, and the carrier is a mathematical structure where a well-founded ordering $>$ is known. In the DP framework, weakly monotone $\mathcal{F}$-algebras play an important role.

Definition 3 (weakly monotone algebra) A mapping $f : A_1 \times \cdots \times A_n \to B$ is *monotone* with respect to $\sqsupset$ if $f(a_1, \ldots, a_i, \ldots, a_n) \sqsupset f(a_1, \ldots, a'_i, \ldots, a_n)$ whenever $a_1 \in A_1, \ldots, a_n \in A_n$, $a'_i \in A_i$, and $a_i \sqsupset a'_i$. A *weakly monotone $\mathcal{F}$-algebra* $\langle A, [\cdot], \geq, > \rangle$ consists of an $\mathcal{F}$-algebra $\langle A, [\cdot] \rangle$ and an order pair $\langle \geq, > \rangle$ such that every $[f]$ is monotone with respect to $\geq$.

Example 4 Continuing Example 2, $\langle \mathbb{N}, [\![\cdot]\!], \geq, > \rangle$ is a weakly monotone $\mathcal{N}_{\mathtt{*+max}}$-algebra with the standard ordering $\langle \geq, > \rangle$. Notice that $\langle \mathbb{Z}, [\![\cdot]\!], \geq, > \rangle$ is not a weakly monotone $\mathcal{Z}_{\mathtt{*+max}}$-algebra, since multiplication on integers is not necessarily monotone. Nevertheless, it is a weakly monotone $\mathcal{Z}_{\mathtt{+max}} \cup \mathcal{N}_{\mathtt{*}}$-algebra.

To ease presentation, from now on we assume that $\mathcal{F}$ is a $\{\mathtt{o}\}$-sorted signature, while $\mathcal{G}$ is an $\mathcal{S}$-sorted signature. It is easy nevertheless to generalize our results to an arbitrary order-sorted signature $\mathcal{F}$.

Theorem 4 ([16]) *Let $\langle A, [\cdot], \geq, > \rangle$ be a weakly monotone $\mathcal{F}$-algebra such that $>$ is well-founded in A. Then $\langle \geq_{[\cdot]}, >_{[\cdot]} \rangle$ is a reduction pair on $\mathcal{T}(\mathcal{F}, \mathcal{V})$, where $s \sqsupset_{[\cdot]} t \iff \forall \alpha : \mathcal{V} \to A.\ [s]\alpha \sqsupset [t]\alpha$.*

Moreover, using the term algebra any reduction pair $\langle \succsim, \succ \rangle$ on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ can be seen as a well-founded $\mathcal{F}$-algebra $\langle \mathcal{T}(\mathcal{F}, \mathcal{V}), \cdot, \succsim, \succ \rangle$.

Example 5 Continuing Example 4, $\langle \geq_{[\![\cdot]\!]}, >_{[\![\cdot]\!]} \rangle$ forms a reduction pair for signature $\mathcal{N}_{\mathtt{*+max}}$. Notice that it does not for $\mathcal{Z}_{\mathtt{+max}} \cup \mathcal{N}_{\mathtt{*}}$, essentially because $>$ is not well-founded in $\mathbb{Z}$.

In order to prove the finiteness of a given DP problem, we need a weakly monotone $\mathcal{F}$-algebra for the signature $\mathcal{F}$ indicated by this problem, rather than for a predefined signature like $\mathcal{N}_{\mathtt{*+max}}$. We fill the gap by employing the notion of *derivers* [26, 35] that turns an algebra into another of a different signature. We assume $\mathtt{x}_1, \mathtt{x}_2, \ldots$ are distinct variable symbols.

Definition 4 (deriver) Let $\delta \in \mathcal{S}$. A δ-sorted *$\mathcal{F}/\mathcal{G}$-deriver* is a mapping d, such that $d(f) : \delta \in \mathcal{T}(\mathcal{G}, \{\mathtt{x}_1 : \delta, \ldots, \mathtt{x}_n : \delta\})$ when f has arity n in $\mathcal{F}$. Given

a *base* $\mathcal{G}$-algebra $\langle A, [\cdot]\rangle$, we define the *derived* $\mathcal{F}$-algebra $\langle A_\delta, d[\cdot]\rangle$ by

$$d[f](a_1, \ldots, a_n) := [d(f)](\mathtt{x}_1 \mapsto a_1, \ldots, \mathtt{x}_n \mapsto a_n)$$

Example 6 Let us denote the signature $\{\mathtt{fact}^\sharp : \mathtt{o} \to \mathtt{o}, \mathtt{s} : \mathtt{o} \to \mathtt{o}, \mathtt{p} : \mathtt{o} \to \mathtt{o}\}$ by $\mathcal{F}_{\mathtt{fact}}$ and define a $\mathtt{Nat}$-sorted $\mathcal{F}_{\mathtt{fact}}/\mathcal{Z}_{+\mathtt{max}}$-deriver d by

$$d(\mathtt{fact}^\sharp) := \mathtt{x}_1 \qquad d(\mathtt{s}) := \mathtt{x}_1 + 1 \qquad d(\mathtt{p}) := \mathtt{max}(\mathtt{x}_1 - 1, 0)$$

Note that $d(\mathtt{p})$ has sort $\mathtt{Nat}$, thanks to the rank $\mathtt{Int} \times \mathtt{Nat} \to \mathtt{Nat}$ of $\mathtt{max}$ in $\mathcal{Z}_{\mathtt{max}}$. The order pair $\langle \geq_{d[\![\cdot]\!]}, >_{d[\![\cdot]\!]}\rangle$ satisfies the constraints given in Example 3.

Now we show that an $\mathcal{F}/\mathcal{G}$-deriver yields a weakly monotone $\mathcal{F}$-algebra if the base $\mathcal{G}$-algebra is known to be weakly monotone. Thus, Example 6 proves that $\mathcal{R}_{\mathtt{fact}}$ is terminating. The next result about monotonicity is folklore:

Lemma 2 *A mapping $f : A_1 \times \cdots \times A_n \to B$ is monotone with respect to a quasi-order $\geq$ if and only if $a_1 \geq b_1, \ldots, a_n \geq b_n$ implies $f(a_1, \ldots, a_n) \geq f(b_1, \ldots, b_n)$.*

Proof The "if" direction is due to the reflexivity of $\geq$, and the "only if" direction is easy by induction on n and the transitivity of $\geq$. □

Then monotonicity is carried over to the interpretation of terms, in the following sense. For two sorted maps $\alpha : X \to A$ and $\beta : X \to A$, we write $\alpha \geq \beta$ to mean that $\alpha(x) \geq \beta(x)$ whenever $x : \sigma \in X$.

Lemma 3 *Let $\langle A, [\cdot], \geq, >\rangle$ be a weakly monotone $\mathcal{G}$-algebra and $s : \sigma \in \mathcal{T}(\mathcal{G}, \mathcal{V})$. If $\alpha \geq \beta$ then $[s]\alpha \geq [s]\beta$.*

Proof By structural induction on s. The claim is trivial if s is a variable. Consider $s = f(s_1, \ldots, s_n)$. We have $[s_i]\alpha \geq [s_i]\beta$ for each $i \in \{1, \ldots, n\}$ by the induction hypothesis. With Lemma 2 and the monotonicity of $[f]$, we conclude:

$$[s]\alpha = [f]([s_1]\alpha, \ldots, [s_n]\alpha) \;\geq\; [f]([s_1]\beta, \ldots, [s_n]\beta) = [s]\beta$$ □

Lemma 4 *Let $\langle A, [\cdot], \geq, >\rangle$ be a weakly monotone $\mathcal{G}$-algebra and d a δ-sorted $\mathcal{F}/\mathcal{G}$-deriver. Then $\langle A_\delta, d[\cdot], \geq, >\rangle$ is a weakly monotone $\mathcal{F}$-algebra.*

Proof Suppose that f has arity n in $\mathcal{F}$, and for every $i \in \{1, \ldots, n\}$ that $a_i, b_i \in A_\delta$ and $a_i \geq b_i$. Then from Lemma 3,

$$\begin{aligned} d[f](a_1, \ldots, a_n) &= [d(f)](\mathtt{x}_1 \mapsto a_1, \ldots, \mathtt{x}_n \mapsto a_n) \\ &\geq [d(f)](\mathtt{x}_1 \mapsto b_1, \ldots, \mathtt{x}_n \mapsto b_n) = d[f](b_1, \ldots, b_n) \end{aligned}$$

With Lemma 2 we conclude that every $d[f]$ is monotone with respect to $\geq$, and hence $\langle A_\delta, d[\cdot], \geq, > \rangle$ is a weakly monotone $\mathcal{F}$-algebra. □

Thus we conclude the soundness of the deriver-based interpretation method:

Theorem 5 *If $\langle A, [\cdot], \geq, > \rangle$ is a weakly monotone $\mathcal{G}$-algebra, $>$ is well-founded in A_δ, and d is a δ-sorted $\mathcal{F}/\mathcal{G}$-deriver, then $\langle \geq_{d[\cdot]}, >_{d[\cdot]} \rangle$ is a reduction pair.*

Proof Immediate consequence of Lemma 4 and Theorem 4. □

Below we see that some existing methods are subsumed by Theorem 5.

Corollary 2 (polynomial interpretation [25, 1]) *If for every n-ary f in $\mathcal{F}$, $[f]_{\mathcal{P}ol}(x_1, \ldots, x_n)$ is a polynomial*

$$\sum_{k_1 + \cdots + k_n \leq m} c_{k_1, \ldots, k_n} \cdot x_1^{k_1} \cdots x_n^{k_n} \tag{1}$$

where $m \in \mathbb{N}$, $k_1, \ldots, k_n$ range over $\mathbb{N}$ and $c_{k_1, \ldots, k_n} \in \mathbb{N}$, then $\langle \geq_{[\cdot]_{\mathcal{P}ol}}, >_{[\cdot]_{\mathcal{P}ol}} \rangle$ forms a reduction pair.

Proof Let $[f]_{\mathcal{P}ol}(x_1, \ldots, x_n)$ be given by (1). Then define $d(f)$ by

$$d(f) := \mathtt{sum}\left\{ c_{k_1, \ldots, k_n} * \mathtt{x}_1^{k_1} * \cdots * \mathtt{x}_n^{k_n} \;\middle|\; k_1 + \cdots + k_n \leq m \right\}$$

where $\mathtt{sum}\, S$ denotes the term $s_1 + \cdots + s_k$ when $S = \{s_1, \ldots, s_k\}$ and $s_1 \prec \cdots \prec s_k$ for an arbitrary fixed total ordering $\prec$ on terms, and s^k is recursively defined by $s^0 = \mathtt{1}$ and $s^{k+1} = s^k * s$. Then d is a $\mathtt{Nat}$-sorted $\mathcal{F}/\mathcal{N}_{*+}$-deriver and $d[\![\cdot]\!]$ coincides with $[\cdot]_{\mathcal{P}ol}$. Hence Theorem 5 concludes the proof. □

Corollary 3 (negative constants [17, Lemma 4]) *If for every n-ary f in $\mathcal{F}$, $[f](x_1, \ldots, x_n)$ is written as*

$$\max\left(a + \textstyle\sum_{1 \leq k_1 + \cdots + k_n \leq m} c_{k_1, \ldots, k_n} \cdot x_1^{k_1} \cdots x_n^{k_n}, 0\right) \tag{2}$$

where $a \in \mathbb{Z}$, $m \in \mathbb{N}$, $k_1, \ldots, k_n$ *range over* $\mathbb{N}$, *and* $c_{k_1,\ldots,k_n} \in \mathbb{N}$, *then* $\langle \geq_{[\cdot]}, >_{[\cdot]} \rangle$ *forms a reduction pair.*

Proof Let $[f](x_1, \ldots, x_n)$ be given by (2). Then define $d(f)$ as

$$\mathtt{max}\Big(a + \mathtt{sum}\big\{c_{k_1,\ldots,k_n} * \mathtt{x}_1^{k_1} * \cdots * \mathtt{x}_n^{k_n} \;\big|\; 1 \leq k_1 + \cdots + k_n \leq m\big\}, \mathtt{0}\Big)$$

Note that this term has sort $\mathtt{Nat}$ due to the rank $\mathtt{max} : \mathtt{Int} \times \mathtt{Nat} \to \mathtt{Nat}$. Thus d is a $\mathtt{Nat}$-sorted $\mathcal{F}/(\mathcal{N}_* \cup \mathcal{Z}_{+\mathtt{max}})$-deriver and $d[\![\cdot]\!]$ coincides with $[\cdot]$. Hence Theorem 5 concludes the proof. □

Theorem 5 gives a slightly more general fact that one can mix max and negative constants and still get a reduction pair. As far as the author knows, this fact has not been reported elsewhere, although natural max-polynomials without negative constants are known to yield reduction pairs [10, Section 4.1].

In addition, a syntactic technique known as *argument filtering* [1, 24] is also a special case of Theorem 5. In the context of higher-order rewriting, Kop and van Raamsdonk generalized argument filters into *argument functions* [20, Definition 7.7], which, in the first-order case, correspond to derivers with $\mathcal{G}$ being a variant of $\mathcal{F}$. In these applications, base signatures and algebras are not *a priori* known, but are subject to be synthesized and analyzed.

5 Tuple Interpretations

The *matrix interpretation method* [9] uses a well-founded weakly monotone algebra $\langle \mathbb{N}^m, [\cdot]_{\mathcal{M}at}, \gtrsim, \gg \rangle$ over natural vectors, with an affine interpretation:

$$[f]_{\mathcal{M}at}(\vec{a}_1, \ldots, \vec{a}_n) = C_1\vec{a}_1 + \cdots + C_n\vec{a}_n + \vec{c} \tag{3}$$

where $C_1, \ldots, C_n \in \mathbb{N}^{m \times m}$ and $\vec{c} \in \mathbb{N}^m$, and the following ordering:

Definition 5 ([9]) Given an order pair $\langle \geq, > \rangle$ on A and a dimension $m \in \mathbb{N}$, we define the order pair $\langle \gtrsim, \gg \rangle$ on A^m as follows:

$$(a_1, \ldots, a_m) \;{}_{(}\gtrsim_{)}\; (b_1, \ldots, b_m) \;:\Longleftrightarrow\; a_1 \;{}_{(}\geq_{)}\; b_1 \wedge a_2 \geq b_2 \wedge \cdots \wedge a_m \geq b_m$$

Improved matrix interpretations [7] consider square matrices instead of vectors, and thus, in principle, matrix polynomials can be considered. Now we generalize these methods by extending derivers to multi-dimensional ones.

Definition 6 (tuple algebra) For an $\mathcal{S}$-sorted set A, we define the $\mathcal{S}^*$-sorted set A^* by $(A^*)_{(\sigma_1,\ldots,\sigma_n)} := A_{\sigma_1} \times \cdots \times A_{\sigma_n}$. For an $\mathcal{S}$-sorted signature $\mathcal{G}$, we

define the $\mathcal{S}^*$-sorted signature $\mathcal{G}^*$ by extending $\mathcal{G}$ with

$$\mathtt{tp} : \sigma_1 \times \cdots \times \sigma_n \to (\sigma_1, \ldots, \sigma_n) \qquad (\cdot)_i : (\sigma_1, \ldots, \sigma_i, \ldots, \sigma_n) \to \sigma_i$$

for all $\sigma_1, \ldots, \sigma_n \in \mathcal{S}$. The *tuple algebra* induced by a $\mathcal{G}$-algebra $\langle A, [\cdot]\rangle$ is the $\mathcal{G}^*$-algebra $\langle A^*, [\cdot]\rangle$, where $[\cdot]$ is extended with

$$[\mathtt{tp}](a_1, \ldots, a_n) = (a_1, \ldots, a_n) \qquad [(\cdot)_i]((a_1, \ldots, a_i, \ldots, a_n)) = a_i$$

Using the tuple algebra (induced by a $\mathcal{G}$-algebra), one can derive an $\mathcal{F}$-algebra over tuples via an $\mathcal{F}/\mathcal{G}^*$-deriver.

Example 7 ([9, Example 1]) The singleton TRS $\{\mathtt{f}(\mathtt{f}(x)) \to \mathtt{f}(\mathtt{g}(\mathtt{f}(x)))\}$ over the signature $\mathcal{F}_{\mathtt{fg}} := \{\mathtt{f} : \mathtt{o} \to \mathtt{o}, \mathtt{g} : \mathtt{o} \to \mathtt{o}\}$ can be shown terminating by the following 2-dimensional matrix interpretation:

$$[\mathtt{f}]_{\mathcal{M}at}(\vec{a}) = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \vec{a} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad [\mathtt{g}]_{\mathcal{M}at}(\vec{a}) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \vec{a} + \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Notice that

$$[\mathtt{f}]_{\mathcal{M}at}(\vec{a}) = \begin{pmatrix} (\vec{a})_1 + (\vec{a})_2 \\ 1 \end{pmatrix} \qquad [\mathtt{g}]_{\mathcal{M}at}(\vec{a}) = \begin{pmatrix} (\vec{a})_1 \\ 0 \end{pmatrix}$$

Clearly, the $(\mathtt{Nat}, \mathtt{Nat})$-sorted $\mathcal{F}_{\mathtt{fg}}/\mathcal{N}^*_+$-deriver $\vec{d}$ defined by

$$\vec{d}(\mathtt{f}) = \mathtt{tp}\begin{pmatrix} (\mathtt{x}_1)_1 \mathtt{+} (\mathtt{x}_1)_2 \\ \mathtt{1} \end{pmatrix} \qquad \vec{d}(\mathtt{g}) = \mathtt{tp}\begin{pmatrix} (\mathtt{x}_1)_1 \\ \mathtt{0} \end{pmatrix}$$

represents $[\cdot]_{\mathcal{M}at}$ as $\vec{d}[\![\cdot]\!]$, that is, $[f]_{\mathcal{M}at} = \vec{d}[\![f]\!]$ and $[g]_{\mathcal{M}at} = \vec{d}[\![g]\!]$.

The following lemma is one of the main results of this paper, which is somewhat surprisingly easy to prove.

Lemma 5 *If a $\mathcal{G}$-algebra $\langle A, [\cdot]\rangle$ is monotone with respect to a reflexive relation $\geq$, then $\langle A^*, [\cdot]\rangle$ is monotone with respect to $\gtrapprox$.*

Proof By assumption, $[f]$ is monotone if $f : \sigma_1 \times \cdots \times \sigma_n \to \tau \in \mathcal{G}$. It is also immediate that $[(\cdot)_i]$ is monotone by definition. Finally, $[\mathtt{tp}]$ is monotone since $a_i \geq a'_i$ implies

$$(a_1, \ldots, a_i, \ldots, a_n) \gtrapprox (a_1, \ldots, a'_i, \ldots, a_n)$$

thanks to the reflexivity of $\geq$. □

Theorem 6 *If $\vec{d}$ is a $(\delta_1, \ldots, \delta_n)$-sorted $\mathcal{F}/\mathcal{G}^*$-deriver, $\langle A, [\cdot], \geq, > \rangle$ is a weakly monotone $\mathcal{G}$-algebra, and $>$ is well-founded in A_{δ_1}, then $\langle \gtrsim_{\vec{d}[\cdot]}, \gg_{\vec{d}[\cdot]} \rangle$ is a reduction pair.*

Proof Thanks to Theorem 5 and Lemma 5, it suffices to show that $\gg$ is well-founded in $A^*_{(\delta_1,\ldots,\delta_n)}$. Suppose on the contrary that there exists an infinite sequence $\vec{a}_1 \gg \vec{a}_2 \gg \cdots$ with $\vec{a}_1, \vec{a}_2, \ldots \in A^*_{(\delta_1,\ldots,\delta_n)}$. Then we have $(\vec{a}_1)_1 > (\vec{a}_2)_1 > \cdots$ and $(\vec{a}_1)_1, (\vec{a}_2)_1, \ldots \in A_{\delta_1}$, contradicting the well-foundedness of $>$ in A_{δ_1}. □

Clearly, every matrix interpretation can be expressed as an $\mathcal{F}/\mathcal{N}^*_{*+}$-deriver.

Corollary 4 (matrix interpretations [9]) *Let $[\cdot]_{\mathcal{M}at}$ be a matrix interpretation. Then $\langle \gg_{[\cdot]_{\mathcal{M}at}}, \gtrsim_{[\cdot]_{\mathcal{M}at}} \rangle$ is a reduction pair.*

Proof We define the $(\mathtt{Nat}, \ldots, \mathtt{Nat})$-sorted $\mathcal{F}/\mathcal{N}^*_{*+}$-deriver $\vec{d}$ as follows: Let f be a symbol in $\mathcal{F}$ and $[f]_{\mathcal{M}at}$ be given by (3). Then define

$$\vec{d}(f) := \mathtt{tp}\begin{pmatrix} s(C_1, \mathtt{x}_1, 1) + \cdots + s(C_n, \mathtt{x}_n, 1) + (\vec{c})_1 \\ \vdots \\ s(C_1, \mathtt{x}_1, m) + \cdots + s(C_n, \mathtt{x}_n, m) + (\vec{c})_m \end{pmatrix}$$

where $s(C, x, i) := (C)_{i,1} * (x)_1 + \cdots + (C)_{i,m} * (x)_m$. It is easy to see that $\vec{d}[\![f]\!](\vec{a}_1, \ldots, \vec{a}_n) = [\![\vec{d}(f)]\!](\mathtt{x}_1 \mapsto \vec{a}_1, \ldots, \mathtt{x}_n \mapsto \vec{a}_n) = [f]_{\mathcal{M}at}(\vec{a}_1, \ldots, \vec{a}_n)$ by definition. Hence, Theorem 6 concludes the proof. □

Corollary 5 (improved matrix interpretations [7]) *Let $\langle \mathbb{N}^{m\times m}, [\cdot]_{\mathcal{M}at} \rangle$ be an $\mathcal{F}$-algebra such that for each f of arity n in $\mathcal{F}$,*

$$[f]_{\mathcal{M}at}(A_1, \ldots, A_n) = C_1A_1 + \cdots + C_nA_n + C_0$$

where $C_0, C_1, \ldots, C_n \in \mathbb{N}^{m\times m}$. Then $\langle \gg_{[\cdot]_{\mathcal{M}at}}, \gtrsim_{[\cdot]_{\mathcal{M}at}} \rangle$ is a reduction pair.

Proof We represent an $m \times m$-matrix A by an m^2-tuple

$$\overline{A} := ((A)_{1,1}, \ldots, (A)_{1,m}, \ldots, (A)_{m,1}, \ldots, (A)_{m,m})$$

Define $s(C, x, i, j) := (C)_{i,1} * (x)_{(j-1)m+1} + \cdots + (C)_{i,m} * (x)_{(j-1)m+m}$. Then we have $(CA_k)_{i,j} = [\![s(C, \mathtt{x}_k, i, j)]\!](\mathtt{x}_1 \mapsto \overline{A}_1, \ldots, \mathtt{x}_n \mapsto \overline{A}_n)$. Using this fact, it

is easy to see that $\vec{d}[\![f]\!](\overline{A}_1, \ldots, \overline{A}_n) = \overline{[f]_{\mathcal{M}at}(A_1, \ldots, A_n)}$, where

$$\vec{d}(f) := \mathtt{tp}(t_{1,1}, \ldots, t_{1,m}, \ldots, t_{m,1}, \ldots, t_{m,m})$$

and $t_{i,j} := s(C_1, \mathtt{x}_1, i, j) + \cdots + s(C_n, \mathtt{x}_n, i, j) + (C_0)_{i,j}$. Hence, Theorem 6 concludes the proof. □

There are two more important consequences of Theorem 6: First, we can interpret symbols as non-affine maps even including max-polynomials. Second, since $>$ is not required to be well-founded in $A_{\delta_2}, \ldots, A_{\delta_m}$, examples that previously required non-monotone interpretations—and hence a stronger condition than Theorem 2—can be handled.

Example 8 (Excerpt of `AProVE_08/log`) Consider the TRS $\mathcal{R}_/$ consisting of

$$\begin{array}{cc} x \mathtt{ - } \mathtt{0} \to x & \mathtt{0} \mathtt{ / } y \to \mathtt{0} \\ \mathtt{s}(x) \mathtt{ - } \mathtt{s}(y) \to x \mathtt{ - } y & \mathtt{s}(x) \mathtt{ / } \mathtt{s}(y) \to (\mathtt{s}(x) \mathtt{ - } \mathtt{s}(y)) \mathtt{ / } \mathtt{s}(y) \end{array}$$

which defines (for simplicity, rounded up) natural division. Proving $\mathcal{R}_/$ terminating using dependency pairs boils down to finding a reduction pair $\langle \succeq, \succ \rangle$ such that (again considering usable rules)

$$x \mathtt{ - } \mathtt{0} \succeq x \quad \mathtt{s}(x) \mathtt{ - } \mathtt{s}(y) \succeq x - y \quad \mathtt{s}(x) \mathtt{ /^\sharp } \mathtt{s}(y) \succ (\mathtt{s}(x) \mathtt{ - } \mathtt{s}(y)) \mathtt{ /^\sharp } \mathtt{s}(y) \tag{4}$$

An interpretation $[\cdot]_{\mathsf{HM}}$ such that

$$[\mathtt{0}]_{\mathsf{HM}} = 0 \quad [\mathtt{s}]_{\mathsf{HM}}(x) = x + 1 \quad [\mathtt{-}]_{\mathsf{HM}}(x, y) = \max(x - y, 0) \quad [\mathtt{/^\sharp}]_{\mathsf{HM}}(x, y) = x$$

satisfies (4), but this is not enough since $[\mathtt{-}]_{\mathsf{HM}}(x, y)$ is not weakly monotone in y. Instead, one must validate the requirement of [17, Theorem 11], that is, $l =_{[\cdot]_{\mathsf{HM}}} r$ for every $l \to r \in \mathcal{R}_/$. This is possible in this example, e.g., by setting $[/]_{\mathsf{HM}}(x, y) = 0$, but it is clearly beneficial to stay in the DP framework and combine techniques there (e.g., the usable rules).

In our setting, a $(\mathtt{Nat}, \mathtt{Neg})$-sorted $\mathcal{F}/\mathcal{Z}^*_{\mathtt{+max}}$-deriver $\vec{d}$ such that

$$\begin{array}{ll} \vec{d}(\mathtt{0}) = \mathtt{tp}(0, 0) & \vec{d}(\mathtt{s}) = \mathtt{tp}((\mathtt{x}_1)_1 \mathtt{ + } 1, (\mathtt{x}_1)_2 \mathtt{ - } 1) \\ \vec{d}(\mathtt{-}) = \mathtt{tp}(\mathtt{max}((\mathtt{x}_1)_1 \mathtt{ + } (\mathtt{x}_2)_2, 0), 0) & \vec{d}(\mathtt{/^\sharp}) = \mathtt{tp}((\mathtt{x}_1)_1, 0) \end{array}$$

yields a reduction pair satisfying constraints (4), where $\mathcal{F} = \{\mathtt{0} : \mathtt{o},\ \mathtt{s} : \mathtt{o} \to \mathtt{o},\ \mathtt{-}, \mathtt{/^\sharp} : \mathtt{o} \times \mathtt{o} \to \mathtt{o}\}$. Hence, by Theorem 2, $\mathcal{R}_/$ is terminating.

The intuition here is that $\mathtt{s}^n(\mathtt{0})$ is interpreted as a pair $(n, -n)$ so that one does not have to reconstruct $-n$ from n using the non-monotonic minus operation. However, it is not always possible to represent polynomial interpretations with negative coefficients by a tuple interpretation using this idea.

Example 9 Consider $[\cdot]_{\mathsf{HM}}$ and $\vec{d}$ of the above example. On one hand we have $[\mathtt{0}]_{\mathsf{HM}} = [x - x]_{\mathsf{HM}}$, since both sides are constant zero. On the other hand, $\vec{d}[\![\mathtt{0}]\!]$ and $\vec{d}[\![x - x]\!]$ are incomparable: while $\vec{d}[\![\mathtt{0}]\!]$ is the constant $(0, 0)$, notice that $\vec{d}[\![x - x]\!](x \mapsto (1, 0)) = (1, 0)$.

6 Strict Monotonicity

Before the invention of dependency pairs [1], strictly monotone algebras were necessary for proving termination by interpretation methods, and they constitute a sound and complete method for proving termination of TRSs.

Definition 7 A *strictly* monotone $\mathcal{F}$-algebra is a weakly monotone $\mathcal{F}$-algebra $\langle A, [\cdot], \geq, > \rangle$ such that $\langle A, [\cdot] \rangle$ is monotone with respect to both $\geq$ and $>$.

Theorem 7 (cf. [39]) *A TRS $\mathcal{R}$ is terminating if and only if there is a strictly monotone well-founded $\mathcal{F}$-algebra $\langle A, [\cdot], \geq, > \rangle$ such that $\mathcal{R} \subseteq >_{[\cdot]}$.*

Moreover, strict monotonicity is a desirable property in the DP framework as it allows one to remove not only dependency pairs but also rewrite rules.

Theorem 8 ([13]) *A DP problem $\langle \mathcal{P}, \mathcal{R} \rangle$ is finite if $\langle \mathcal{P} \setminus >_{[\cdot]}, \mathcal{R} \setminus >_{[\cdot]} \rangle$ is, where $\langle A, [\cdot], \geq, > \rangle$ is a strictly monotone well-founded $\mathcal{F}$-algebra such that $\mathcal{P} \cup \mathcal{R} \subseteq \geq_{[\cdot]}$.*

We now state a criterion that ensures the strict monotonicity of tuple interpretation obtained via derivers. Here the new assumption $> \subseteq \geq$ in Definition 1 is crucial.

Theorem 9 *Let $\vec{d}$ be a $\vec{\delta}$-sorted $\mathcal{F}/\mathcal{G}^*$-deriver and $\langle A, [\cdot], \geq, > \rangle$ a weakly monotone $\mathcal{G}$-algebra, such that for any n-ary f in $\mathcal{F}$, $\vec{a}_1, \ldots, \vec{a}_n \in A_{\vec{\delta}}$, and $a' \in A_{(\vec{\delta})_1}$, $(\vec{a}_i)_1 > a'$ implies*

$$\left(\vec{d}[f](\vec{a}_1, \ldots, \vec{a}_i, \ldots, \vec{a}_n)\right)_1 > \left(\vec{d}[f](\vec{a}_1, \ldots, (a', (\vec{a}_i)_2, \ldots, (\vec{a}_i)_m), \ldots, \vec{a}_n)\right)_1$$

Then $\langle A_{\vec{\delta}}, \vec{d}[\cdot], \gtrdot, \gg \rangle$ is a strictly monotone $\mathcal{F}$-algebra.

Proof We only prove strict monotonicity as we already know weak monotonicity by Lemma 4. So suppose that f has arity n in $\mathcal{F}$, $\vec{a}_1, \ldots, \vec{a}_i, \ldots, \vec{a}_n, \vec{a}'_i \in A_{\vec{\delta}}$ and $\vec{a}_i \gg \vec{a}'_i$. For the first coordinate, using the assumption and then the weak monotonicity, we conclude

$$\begin{aligned}\left(\vec{d}[f](\vec{a}_1 \ldots, \vec{a}_i, \ldots, \vec{a}_n)\right)_1 &> \left(\vec{d}[f](\vec{a}_1, \ldots, ((\vec{a}'_i)_1, (\vec{a}_i)_2, \ldots, (\vec{a}_i)_m), \ldots, \vec{a}_n)\right)_1 \\ &\geq \left(\vec{d}[f](\vec{a}_1, \ldots, \vec{a}'_i, \ldots, \vec{a}_n)\right)_1\end{aligned}$$

For the other coordinates, since $> \subseteq \geq$ we have $\vec{a}_i \gtrsim \vec{a}'_i$. Then the weak monotonicity ensures

$$\vec{d}[f](\vec{a}_1, \ldots, \vec{a}_i, \ldots \vec{a}_n) \gtrsim \vec{d}[f](\vec{a}_1, \ldots, \vec{a}'_i, \ldots, \vec{a}_n)$$

from which we deduce for each $j \in \{2, \ldots, m\}$,

$$\left(\vec{d}[f](\vec{a}_1, \ldots, \vec{a}_i, \ldots, \vec{a}_n)\right)_j \geq \left(\vec{d}[f](\vec{a}_1, \ldots, \vec{a}'_i, \ldots, \vec{a}_n)\right)_j \qquad \square$$

Although the above result and proof do not look surprising, it would be worth noticing that the statement is false in the standard formulation allowing $> \not\subseteq \geq$ (as even in [9]).

Example 10 Consider the following apparently monotone matrix interpretation:

$$[\mathtt{f}]\left(\begin{pmatrix} a_1 \\ a_2 \end{pmatrix}\right) := \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} a_1 \\ a_1 \end{pmatrix}$$

If one had $a_1 > b_1$ but $a_1 \not\gtrsim b_1$, then

$$[\mathtt{f}]\left(\begin{pmatrix} a_1 \\ a_2 \end{pmatrix}\right) = \begin{pmatrix} a_1 \\ a_1 \end{pmatrix} \begin{matrix} > \\ \not\geq \end{matrix} \begin{pmatrix} b_1 \\ b_1 \end{pmatrix} = [\mathtt{f}]\left(\begin{pmatrix} b_1 \\ a_2 \end{pmatrix}\right) \quad \text{even though} \quad \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \gg \begin{pmatrix} b_1 \\ a_2 \end{pmatrix}.$$

So $[\mathtt{f}]$ would not be monotone with respect to $\gg$.

7 Arctic Interpretations

An *arctic interpretation* [22] is a matrix interpretation $[\cdot]_{\mathcal{A}}$ on the *arctic semiring*; that is, every interpretation $[f]_{\mathcal{A}}(\vec{a}_1, \ldots, \vec{a}_n)$ is of the form

$$C_1 \otimes \vec{a}_1 \oplus \cdots \oplus C_n \otimes \vec{a}_n \oplus \vec{c} \tag{5}$$

where $\otimes$ denotes the matrix multiplication in which the scalar addition is replaced by the max operation, and the scalar multiplication by addition; $\oplus$ is the supremum operator (pointwise max), and entries of C_i and $\vec{c}$ are *arctic*

naturals ($\mathbb{N}_{-\infty} := \mathbb{N} \cup \{-\infty\}$) or *arctic integers* ($\mathbb{Z}_{-\infty} := \mathbb{Z} \cup \{-\infty\}$). An arctic interpretation is said to be *absolutely positive* if (5) satisfies $(\vec{c})_1 \geq 0$.

Theorem 10 (cf. [22]) *If $[\cdot]_{\mathcal{A}}$ is an absolutely positive arctic interpretation, then $\langle \gtrsim_{[\cdot]_{\mathcal{A}}}, \gg_{[\cdot]_{\mathcal{A}}} \rangle$ is a reduction pair.*

The above formulation deviates from the original [22] in two ways. First, we use the standard ordering of matrix interpretations, while Koprowski and Waldmann used the pointwise extension of a special relation $>_{\mathsf{KW}}$, defined by $a >_{\mathsf{KW}} b :\iff a > b \vee a = b = -\infty$. Although their definition is necessary for yielding a classic strictly monotone algebra, it works only if no function symbol has arity more than one, e.g., in string rewriting. In the dependency pair setting, our formulation is simply more general:

Proposition 1 *If $\vec{x} \in \mathbb{N} \times \mathbb{Z}_{-\infty}^{m-1}$ and $\vec{x} >_{\mathsf{KW}} \vec{y}$, then $\vec{x} \gg \vec{y}$.*

Proof Since $(\vec{x})_1 \in \mathbb{N}$, $(\vec{x})_1 >_{\mathsf{KW}} (\vec{y})_1$ implies $(\vec{x})_1 > (\vec{y})_1$. For $i = 2, \ldots, m$, $(\vec{x})_i >_{\mathsf{KW}} (\vec{y})_i$ implies $(\vec{x})_i \geq (\vec{y})_i$. We conclude $\vec{x} \gg \vec{y}$. □

The second difference is that for arctic natural interpretations, Koprowski and Waldmann relaxed absolute positiveness to *somewhere finiteness*: $(\vec{c})_1 \neq -\infty$ or $(C_i)_{1,1} \neq -\infty$ for some i. Sternagel and Thiemann showed that a similar assumption (where "$\neq -\infty$" is refined to "≥ 0") is sufficient for arctic integers [30]. However, these assumptions turn out equivalent.

Proposition 2 *An arctic natural interpretation has a somewhere finite representation iff it has an absolute positive one.*

Proof Clearly, absolute positiveness implies somewhere finiteness. For the other direction, consider an interpretation $[\cdot]_{\mathcal{A}}$ with a somewhere finite representation (5). Since $(\vec{c})_1 \neq -\infty$ trivially implies absolute positiveness, suppose that $(\vec{c})_1 = -\infty$ and $(C_i)_{1,1} \neq -\infty$ for some i. We then know $(\vec{y})_1 \geq 0$, where $\vec{y} := C_1 \otimes \vec{x}_1 \oplus \cdots \oplus C_n \otimes \vec{x}_n$. Hence, by $\vec{c}' := (0, (\vec{c})_2, \ldots, (\vec{c})_m)$, we have $[f]_{\mathcal{A}}(\vec{x}_1, \ldots, \vec{x}_n) = \vec{y} \oplus \vec{c}'$, and this representation is absolutely positive. □

This partly answers a question posed in [22]. However, it does not mean that arctic natural interpretations are subsumed by integer ones, since orderings on terms with respect to the latter must take the possibility of negative values into account.

One can easily obtain arctic interpretations via derivers: consider a sort `ANat` with `Nat` $\sqsubseteq$ `ANat` and {`Nat`, `ANat`}-sorted signature $\mathcal{N}_{\texttt{+max-}\infty}$, extending

$\mathcal{N}_{\mathtt{+max}}$ with

$$\mathtt{-\infty} : \mathtt{ANat} \qquad \mathtt{+} : \mathtt{ANat} \times \mathtt{ANat} \to \mathtt{ANat} \qquad \mathtt{max} : \mathtt{Nat} \times \mathtt{ANat} \to \mathtt{Nat}$$
$$\mathtt{max} : \mathtt{ANat} \times \mathtt{Nat} \to \mathtt{Nat} \qquad \mathtt{max} : \mathtt{ANat} \times \mathtt{ANat} \to \mathtt{ANat}$$

and extend the standard interpretation $[\![\cdot]\!]$ accordingly. We omit the easy proof of the following fact and the counterpart for arctic integer interpretations.

Proposition 3 *Every absolute positive arctic natural interpretation* $[\cdot]_{\mathcal{A}}$ *is represented as* $\vec{d}[\![\cdot]\!]$ *via a* $(\mathtt{Nat}, \mathtt{ANat}, \ldots, \mathtt{ANat})$*-sorted* $\mathcal{F}/\mathcal{N}^*_{\mathtt{+max-\infty}}$*-deriver* $\vec{d}$*.*

In practice, however, this requires us to deal with $-\infty$ by ourselves since there is no standard SMT theory [3] that supports arithmetic with $-\infty$.

8 Template-Based Implementation

The search for a deriver is implemented in the termination prover NaTT using a *template*-based approach. Below we fix an $\mathcal{S}$-sorted set $\mathcal{W}$ of fresh *template* variables.

Definition 8 (deriver template) A $\vec{\delta}$-sorted $\mathcal{F}/\mathcal{G}^*$*-deriver template* is a mapping $\vec{d}$ such that $\vec{d}(f) : \vec{\delta} \in \mathcal{T}\big(\mathcal{G}^*, \{\mathtt{x}_1 : \vec{\delta}, \ldots, \mathtt{x}_n : \vec{\delta}\} \uplus \mathcal{W}\big)$ for every n-ary f in $\mathcal{F}$. Its *instance* according to a substitution $\theta : \mathcal{W} \to \mathcal{T}(\mathcal{G}, \emptyset)$ is the $\vec{\delta}$-sorted $\mathcal{F}/\mathcal{G}^*$-deriver $\vec{d}\theta$ defined by $\vec{d}\theta(f) := \vec{d}(f)\theta$.

Example 11 Consider again the signature $\mathcal{F}$ of Example 8, and let $\mathcal{W}$ consist of template variables p_1, p_2, p_3, p_4 of sort $\mathtt{Nat}$ and n_1, n_2, n_3, n_4 of sort $\mathtt{Neg}$. We define a $(\mathtt{Nat}, \mathtt{Neg})$-sorted $\mathcal{F}/\mathcal{N}^*_{\mathtt{+max}}$-deriver template $\vec{d}$ as follows:

$$\vec{d}(\mathtt{0}) = \mathtt{tp}(p_1, n_1) \qquad \vec{d}(\mathtt{s}) = \mathtt{tp}((\mathtt{x}_1)_1 + p_2, (\mathtt{x}_1)_2 + n_2)$$
$$\vec{d}(\mathtt{-}) = \mathtt{tp}(\mathtt{max}((\mathtt{x}_1)_1 + (\mathtt{x}_2)_2, p_3), n_3) \qquad \vec{d}(\mathtt{/^\sharp}) = \mathtt{tp}((\mathtt{x}_1)_1 + p_4, n_4)$$

Given a $\vec{\delta}$-sorted $\mathcal{F}/\mathcal{G}^*$-deriver template $\vec{d}$ and a $\mathcal{G}$-algebra $\langle A, [\cdot]\rangle$, our interest is to find a substitution $\theta : \mathcal{W} \to \mathcal{T}(\mathcal{G}, \emptyset)$ that satisfies $\mathcal{P} \cup \mathcal{R} \subseteq \gtrsim_{\vec{d}\theta[\cdot]}$ for the DP problem $\langle \mathcal{P}, \mathcal{R}\rangle$ of concern. We reduce the problem of finding such a substitution to an SMT problem in several steps.

First, we extend $\vec{d}$ to $\vec{d} : \mathcal{T}(\mathcal{F}, \mathcal{V}) \to \mathcal{T}\big(\mathcal{G}^*, \{v : \vec{\delta} \mid v \in \mathcal{V}\}\big)_{\vec{\delta}}$ by $\vec{d}(v) = v$ if $v \in \mathcal{V}$, and $\vec{d}(f(s_1, \ldots, s_n)) = \vec{d}(f)(\mathtt{x}_1 \mapsto \vec{d}(s_1), \ldots, \mathtt{x}_n \mapsto \vec{d}(s_n))$. Hereafter we assume that $\mathcal{V}$ is a ($\{\mathtt{o}\}$-sorted) set of variables disjoint from $\mathcal{W}$.

Lemma 6 *If $s \in \mathcal{T}(\mathcal{F},\mathcal{V})$ and $\theta : \mathcal{W} \to \mathcal{T}(\mathcal{G},\emptyset)$, then $\vec{d}\theta[s] = \big[\vec{d}(s)\theta\big]$.*

Proof We show $\vec{d}\theta[s]\alpha = \big[\vec{d}(s)\theta\big]\alpha$ for an arbitrary $\alpha : \mathcal{V} \to A_{\vec{\delta}}$ by structural induction on s. If $s \in \mathcal{V}$, then $\vec{d}\theta[s]\alpha = \alpha(s) = \big[\vec{d}(s\theta)\big]\alpha$. Note that $s\theta = s$ as $\mathcal{V}$ and $\mathcal{W}$ are disjoint. Next consider $s = f(s_1,\dots,s_n)$. Then

$$\begin{aligned}\vec{d}\theta[f(s_1,\dots,s_n)]\alpha &= \vec{d}\theta[f](\vec{d}\theta[s_1]\alpha,\dots,\vec{d}\theta[s_n]\alpha)\\ &= \big[\vec{d}(f)\theta\big](\mathtt{x}_1 \mapsto \vec{d}\theta[s_1]\alpha,\dots,\mathtt{x}_n \mapsto \vec{d}\theta[s_n]\alpha)\\ &= \big[\vec{d}(f)\theta\big](\mathtt{x}_1 \mapsto \big[\vec{d}(s_1)\theta\big]\alpha,\dots,\mathtt{x}_n \mapsto \big[\vec{d}(s_n)\theta\big]\alpha)\\ &= \big[\vec{d}(f(s_1,\dots,s_n))\theta\big]\alpha\end{aligned}$$

Here, the third equation is by the induction hypothesis. □

In this way, we can reduce $s \gtrsim_{\vec{d}\theta[\cdot]} t$ to $\vec{d}(s)\theta \gtrsim_{[\cdot]} \vec{d}(t)\theta$.

Example 12 Continue Example 11 and consider satisfying $x \mathtt{-} \mathtt{0} \gtrsim_{\vec{d}\theta[\![\cdot]\!]} x$, that is, $\vec{d}\theta[\![x \mathtt{-} \mathtt{0}]\!]\alpha \gtrsim \vec{d}\theta[\![x]\!]\alpha$ for every $\alpha : \{x : (\mathtt{Nat},\mathtt{Neg})\} \to \mathbb{Z}^*$. Due to Lemma 6, this is equivalent to $[\![l\theta]\!]\alpha \gtrsim [\![r\theta]\!]\alpha$, where

$$l = \vec{d}(x \mathtt{-} \mathtt{0}) = \mathtt{tp}(\mathtt{max}((x)_1 \mathtt{+} (\mathtt{tp}(p_1,n_1))_2,p_3),n_3) \qquad r = \vec{d}(x) = x$$

Since $\mathtt{tp}$ and $(\cdot)_i$ are not standard arithmetic operators, the next step is to resolve these operators.

Definition 9 (flattening) Let $\mathcal{X}$ be an $\mathcal{S}^*$-sorted set of variables. We write $\mathcal{X}^\flat$ for the $\mathcal{S}$-sorted set consisting of a fresh variable $x_{\flat i} : \delta_i \in \mathcal{X}^\flat$ for each $x : (\delta_1,\dots,\delta_m) \in \mathcal{X}$ and $i \in \{1,\dots,m\}$. For $s : (\delta_1,\dots,\delta_m) \in \mathcal{T}(\mathcal{G}^*,\mathcal{X})$, we define its *flattening* $s^\flat$ as follows:

- $x^\flat = (x_{\flat 1},\dots,x_{\flat m})$ if $x : (\delta_1,\dots,\delta_m) \in \mathcal{X}$,
- $f(s_1,\dots,s_n)^\flat = f(s_1^\flat,\dots,s_n^\flat)$ if $f : \sigma_1 \times \cdots \times \sigma_n \to \tau \in \mathcal{G}$,
- $\mathtt{tp}(s_1,\dots,s_m)^\flat = (s_1^\flat,\dots,s_m^\flat)$, and
- $(s)_i^\flat = t_i$ if $s^\flat = (t_1,\dots,t_i,\dots,t_n)$.

For $\alpha : \mathcal{X} \to A^*$, we define $\alpha^\flat : \mathcal{X}^\flat \to A$ by $\alpha^\flat(x_{\flat i}) := (\alpha(x))_i$.

Lemma 7 *If $s : (\delta_1,\dots,\delta_m) \in \mathcal{T}(\mathcal{G}^*,\mathcal{X})$, then $s^\flat = (s_1,\dots,s_m)$ for some $s_1 : \delta_1,\dots,s_m : \delta_m \in \mathcal{T}\big(\mathcal{G},\mathcal{X}^\flat\big)$. Moreover $[s]\alpha = ([s_1]\alpha^\flat,\dots,[s_m]\alpha^\flat)$ for any $\alpha : \mathcal{X} \to A^*$.*

Proof By structural induction on s. □

Lemma 8 *Let* $s,t : (\delta_1,\dots,\delta_m) \in \mathcal{T}(\mathcal{G}^*,\mathcal{X})$, $s^\flat = (s_1,\dots,s_m)$ *and* $t^\flat = (t_1,\dots,t_m)$. *If* $s_1 \geq_{[\cdot]} t_1 \wedge \cdots \wedge s_m \geq_{[\cdot]} t_m$, *then* $s \gtrsim_{[\cdot]} t$.

Proof By Lemma 7. □

Example 13 Continue Example 12. We have

$$l^\flat = (\mathtt{max}(x_{\flat 1} + n_1, p_3), n_3) \qquad r^\flat = (x_{\flat 1}, x_{\flat 2})$$

Thus, thanks to Lemma 8, $l\theta \gtrsim_{[\![\cdot]\!]} r\theta$ is equivalent to

$$\mathtt{max}(x_{\flat 1} + n_1\theta, p_3\theta) \geq_{[\![\cdot]\!]} x_{\flat 1} \qquad n_3\theta \geq_{[\![\cdot]\!]} x_{\flat 2} \tag{6}$$

Now we fix our interest to $\mathcal{G} = \mathcal{Z}_{+\mathtt{max}} \cup \mathcal{N}_*$ and $[\cdot] = [\![\cdot]\!]$ and reduce finding θ such that $s\theta \geq_{[\![\cdot]\!]} t\theta$ to an SMT problem. We need to resolve the `max` operator, which is not supported in the standard SMT arithmetic. To this end, terms are reduced according to the following four rules [32]:

$$\mathtt{max}(x,y) + z \to \mathtt{max}(x+z, y+z) \qquad x + \mathtt{max}(y,z) \to \mathtt{max}(x+y, x+z)$$
$$\mathtt{max}(x,y) * z \to \mathtt{max}(x*z, y*z) \qquad x * \mathtt{max}(y,z) \to \mathtt{max}(x*y, x*z)$$

Note that the distribution of $*$ over `max` is admissible because the only allowed rank of $*$ is $\mathtt{Nat} \times \mathtt{Nat} \to \mathtt{Nat}$. In this way, $s\theta \geq_{[\![\cdot]\!]} t$ is normalized into $\mathtt{max}_{i=1}^n s_i\theta \geq_{[\![\cdot]\!]} \mathtt{max}_{j=1}^m t_j\theta$, where `max` does not occur in $s_1,\dots,s_n,t_1,\dots,t_m$. Then the comparison of two such normal forms is translated as follows, cf. [4]:

$$\mathtt{max}_{i=1}^n s_i\theta \geq_{[\![\cdot]\!]} \mathtt{max}_{j=1}^m t_j\theta \iff \bigwedge_{j=1}^m \bigvee_{i=1}^n s_i\theta \geq_{[\![\cdot]\!]} t_j\theta$$

Finally, each $s_i\theta \geq_{[\![\cdot]\!]} t_j\theta$, where s_i and t_j are polynomials of variables in $\mathcal{V}^\flat$, is converted into the conjunction of comparison of monomials. Each monomial comparison $a\theta * x_{1\flat i_1} * \cdots * x_{n\flat i_n} \geq_{[\![\cdot]\!]} b\theta * x_{1\flat i_1} * \cdots * x_{n\flat i_n}$ is reduced to: (1) $a = b$ if $(\vec{\delta})_{i_j} = \mathtt{Int}$ for some j, (2) $a \geq b$ if $\left\|\{j \mid (\vec{\delta})_{i_j} = \mathtt{Neg}\}\right\|$ is even, and (3) $a \leq b$ otherwise. Due to the last step, having coordinates of sort `Int` leads to a stronger constraint when ordering terms. Finally, the resulting formula, containing only template variables, is passed to the SMT solver Z3 4.8.10 [28] and a satisfying solution $\theta : \mathcal{W} \to \mathbb{Z}$ is a desired substitution.

Example 14 Continue Example 13. After normalization, (6) is reduced to

$$x_{\flat 1} + n_1\theta \geq_{[\![\cdot]\!]} x_{\flat 1} \vee p_3\theta \geq_{[\![\cdot]\!]} x_{\flat 1} \qquad n_3\theta \geq_{[\![\cdot]\!]} x_{\flat 2}$$

which is then converted to the following SMT formula via comparison of coefficients:

$$((1 \geq 1 \wedge n_1 \geq 0) \vee (p_3 \geq 0 \wedge 0 \geq 1)) \wedge n_3 \geq 0 \wedge 1 \geq 0$$

A solution θ to the formula, in conjunction with other formulas resulting from (4), gives the desired deriver $\vec{d}\theta$ that proves the termination of $\mathcal{R}_{/}$ of Example 8.

9 Experiments

To evaluate the practical significance of the tuple interpretation method, we experimentally evaluate various templates in a simple dependency pair setting. For a function symbol f of arity n, we define $\vec{d}(f) := \mathtt{tp}(d_1(f), \ldots, d_m(f))$ and choose $d_k(f)$ from

- sum: $w + \sum_{i=1}^{n}(b * (\mathtt{x}_i)_k)$,
- max: $\mathtt{max}_{i=1}^{n}\, b * (w + (\mathtt{x}_i)_k)$,
- sum-sum: $w + \sum_{i=1}^{n}\sum_{j=1}^{m} b * (\mathtt{x}_i)_j$,
- max-max: $\mathtt{max}_{i=1}^{n}\,\mathtt{max}_{j=1}^{m}\, b * (w + (\mathtt{x}_i)_j)$,
- sum-max: $\sum_{i=1}^{n}\mathtt{max}_{j=1}^{m}\, b * (w + (\mathtt{x}_i)_j)$,
- max-sum: $\mathtt{max}_{i=1}^{n}(w + \sum_{j=1}^{m} b * (\mathtt{x}_i)_j)$,
- heuristic: a heuristic choice [38] between sum and max, and
- heuristic2: the choice between sum-sum and max-max,

where b and w introduce fresh template variables, b ranges over $\{0, 1\}$ and the sort of w is up to further choice. The sort of $d_1(f)$ is turned to `Nat` by applying $\mathtt{max}(\cdot, 0)$ if necessary. It is meaningless to take max of one or zero argument, so we use sum templates for symbols of arity $n \leq 1$.

All experiments are run on the StarExec environment [31]. The benchmarks are the 1508 TRSs from the TRS Standard category of the *Termination Problem DataBase* (TPDB) ver. 11.2 [34]. Due to the huge search space, we evaluate templates of dimensions up to 2. Full details of the experiments are made available at http://www.trs.cm.is.nagoya-u.ac.jp/NaTT/tuple/.

9.1 Evaluation of Templates

The purpose of this section is to evaluate the practical impact of the tuple interpretation method, as well as to search for new templates that are worth being incorporated into the default strategy of NaTT.

As a base line, in Table 1 we review the 1-dimensional (i.e., `Nat`-sorted) templates. The first two columns represent the choice of the template and the sort of w. The "YES" column indicates the number of successful termination

[3] This template is a subset of integer max-polynomials [10], although the fact that it yields a reduction pair is new.

Table 1 Evaluation of 1-dimensional templates.

d	w	YES	New	Time	T.O.	Known as
sum	`Nat`	512	-	39:12	6	polynomial [1]
max	`Nat`	473	-	54:08	7	natural max-polynomial [10]
heuristic	`Nat`	524	-	52:43	7	natural max-polynomial
sum	`Int`	559	-	49:57	6	negative constant [17]
max	`Int`	563	-	1:04:24	8	integer max-polynomial[3]
heuristic	`Int`	610	-	1:03:29	8	integer max-polynomial[3]

Table 2 Evaluation of d_1 = sum-sum.

w_1	d_2	w_2	YES	New	Time	T.O.	Known as
`Nat`	sum	`Nat`	599	-	1:27:55	12	triangular matrix [9]
`Nat`	sum-sum	`Nat`	637	-	2:58:25	22	matrix [9]
`Nat`	max	`Nat`	603	0	4:45:35	42	
`Nat`	max-max	`Nat`	591	0	23:51:46	246	
`Nat`	max-sum	`Nat`	619	1	10:44:04	104	
`Nat`	sum-max	`Nat`	572	0	1d 23:51:46	246	
`Nat`	heuristic	`Nat`	608	0	04:19:16	39	
`Nat`	heuristic2	`Nat`	620	0	13:19:16	39	
`Int`	sum	`Int`	579	0	2:44:12	21	
`Int`	sum-sum	`Int`	557	0	20:43:03	210	
`Int`	max	`Int`	573	0	10:30:38	97	
`Int`	max-max	`Int`	522	0	1d 15:28:04	414	
`Int`	max-sum	`Int`	559	1	1d 05:04:01	298	
`Int`	sum-max	`Int`	487	1	1d 23:36:04	501	
`Int`	heuristic	`Int`	566	0	9:44:29	89	
`Int`	heuristic2	`Int`	541	0	1d 03:46:22	285	
(a) `Int`	sum	`Neg`	605	10	2:53:48	19	
`Int`	max	`Neg`	588	9	11:39:59	101	
`Int`	heuristic	`Neg`	589	9	10:38:58	97	

proofs. The "New" column shows the number of termination proofs that the 2020 version of NaTT could not find. The "Time" column indicates the total computation time, and "T.O." indicates the number of timeouts, which is set to 300s as in termCOMP. The "Known as" column gives a brief description if the resulting reduction pair is already known. In fact, all the 1-dimensional templates we evaluate here have already been (basically) known and implemented in the 2020 version of NaTT.

We move on to the 2-dimensional templates. Table 2 presents the results when d_1 takes the shape sum-sum. The shape of the second coordinate is indicated in column d_2, and the sorts of w in d_1 and d_2 are specified in the columns w_1 and w_2, respectively.

In the rows where w_2 has sort `Nat`, we observe the impressive effectiveness of the matrix interpretation method (in row 1 and 2). The additional freedom enabled by tuple interpretations does not seem to give much benefit there.

Table 3 Evaluation of $d_1 = \text{max-max}$.

w_1	d_2	w_2	YES	New	Time	T.O.	Known as
`Nat`	sum	`Nat`	584	0	4:06:12	28	
`Nat`	sum-sum	`Nat`	575	0	1d 00:39:33	251	
`Nat`	max	`Nat`	528	0	4:41:00	32	natural arctic [22][4]
`Nat`	max-max	`Nat`	559	4	13:23:48	100	natural arctic
`Nat`	heuristic1	`Nat`	571	0	4:18:42	30	
`Nat`	heuristic2	`Nat`	576	0	19:23:38	186	
`Int`	sum	`Int`	557	0	4:30:14	34	
`Int`	sum-sum	`Int`	536	2	1d 07:17:34	319	
`Int`	max	`Int`	578	0	5:49:53	45	integer arctic
`Int`	max-max	`Int`	587	4	22:39:57	207	integer arctic
`Int`	heuristic1	`Int`	575	0	5:04:15	37	
`Int`	heuristic2	`Int`	564	1	1d 00:29:56	240	
`Int`	sum	`Neg`	587	3	7:35:08	62	
`Int`	heuristic	`Neg`	585	2	8:48:26	81	
`Int`	max	`Neg`	581	2	9:09:57	81	

Only the case $d_2 = \text{max-sum}$ proves one "New" result, but the runtime would be significantly affected. It does not seem to improve by considering `Int` as the sort of w_2 either.

The situation is different when w_2 is given sort `Neg` (last three rows). Especially, the template indicated by (a) proves ten "New" results with mild increase in runtime. Therefore, we choose (a) as the first candidate for incorporation into the NaTT strategy.

Table 3 presents the results when the shape of d_1 is max-max. Here, we obtain variants of arctic interpretations where addition of $-\infty$ is simulated by multiplication by zero. As arctic interpretations were not implemented in NaTT, both arctic natural and integer interpretations deliver four "New" results. However, at least in the current implementation, the overhead on runtime would be significant in the light of the efficiency of the current version of NaTT. Therefore, we do not choose a candidate from here for incorporation into the default strategy.

Table 4 presents the results for a selection of other templates that yield at least one "New" result. None of these templates results in a previously known reduction pair, as far as the author knows. The template indicated by (b) performed the best in terms of the number of successful termination proofs. Moreover, its runtime is relatively reasonable, and it proves nine "New" results that do not have much overlap with those of template (a). Hence we pick template (b) as the second candidate for incorporation into NaTT strategy.

[4]In our implementation, negative infinity is not supported. Instead, a similar effect is obtained by zero coefficients.

Table 4 Evaluation of selected other templates.

d_1	w_1	d_2	w_2	YES	New	Time	T.O.
max-sum	Nat	sum-sum	Nat	597	1	12:01:29	115
max-sum	Nat	max-max	Nat	521	1	1d 01:44:31	245
max-sum	Int	sum	Neg	592	4	5:59:52	52
max-sum	Int	max	Neg	578	1	13:08:37	123
max-sum	Int	heuristic	Neg	583	1	11:32:43	107
sum-max	Nat	sum	Nat	579	1	13:38:14	116
sum-max	Nat	max	Nat	578	1	17:15:29	151
sum-max	Nat	heuristic	Nat	586	1	16:36:56	143
sum-max	Nat	max-max	Nat	559	4	13:23:01	100
sum-max	Int	sum	Neg	588	9	1d 02:13:32	245
sum-max	Int	heuristic	Neg	580	9	1d 05:36:56	286
sum-max	Int	max	Neg	576	8	1d 07:18:22	300
(b) heuristic2	Int	sum	Neg	648	9	4:20:20	29
heuristic2	Int	heuristic	Neg	647	9	7:55:14	65
heuristic2	Int	max	Neg	645	9	8:57:10	72

Table 5 Experiments with combined strategies

Strategy	YES	NO	New to NaTT	New to termCOMP	Time	T.O.
2020 strategy	862	170	-	-	3:04:03	24
with (a)	876	170	14	4	3:22:46	25
with (b)	872	170	10	1	3:43:12	28
with both	883	170	20	6	3:59:17	29

9.2 Incorporation into NaTT Strategy

Finally, we evaluate the impact of incorporating templates (a) and (b) into NaTT. The results are summarized in Table 5. We observe that it is beneficial to add (a) and/or (b). Adding either (a) or (b) gives 14 or 10 new examples that NaTT could not solve before. Notice that these numbers are more than those which (a) and (b) alone could solve: Thanks to the divide-and-conquer nature of the DP framework, the new techniques may solve subproblems that only other techniques of NaTT could yield.

Incorporating both (a) and (b) gives 20 more examples solved, and six of them (`AProVE_09_Inductive/log` and five in `Transformed_CSR_04/`) were not solved by any tool in termCOMP 2020. In one of them (`Transformed_CSR_04/ExIntrod_GM99_iGM`), both (a) and (b) are applied in subproblems.

10 Conclusion

In this paper we introduced the tuple interpretation method based on the notion of derivers. The author expects that the result makes the relationships

between existing interpretation methods cleaner, and eases the task of developing and maintaining termination tools. Moreover, it yields many previously unknown interpretation methods as instances, proving the termination of some standard benchmarks that state-of-the-art termination provers could not.

Since this work significantly broadens the search space of interpretation methods, it is an interesting future work to heuristically search for derivers rather than fixing some templates. Derivers of higher dimensions seem also interesting to explore. To fully subsume arctic interpretations, the treatment of $-\infty$ has to be implemented yet. Finally, although the proposed method is implemented in the termination prover NaTT, there is no guarantee that the implementation is correct. In order to certify termination proofs that use tuple interpretations, one must formalize the proofs in this paper, extend the certifiable proof format [29], and implement a verified function to validate such proofs.

Acknowledgments

The author would like to thank Aart Middeldorp for discussions on the power of existing methods, and Aaron Stump and his team for the StarExec environment, without which the experiments reported in the paper would not be feasible. The author also thanks the anonymous reviewers of previous versions of the paper for their detailed comments that improved the presentation a lot. This work was partly supported by the Austrian Science Fund (FWF) projects Y757 and P27502, and the Japan Science and Technology Agency (JST) project ERATO MMSD.

References

[1] Arts, T., Giesl, J.: Termination of term rewriting using dependency pairs. Theor. Compt. Sci. **236**(1–2), 133–178 (2000). https://doi.org/10.1016/S0304-3975(99)00207-8

[2] Baader, F., Nipkow, T.: Term rewriting and all that. Cambridge University Press (1998)

[3] Barrett, C.W., Tinelli, C.: Satisfiability modulo theories. In: Clarke, E.M., Henzinger, T.A., Veith, H., Bloem, R. (eds.) Handbook of Model Checking, pp. 305–343. Springer (2018). https://doi.org/10.1007/978-3-319-10575-8_11

[4] Ben-Amram, A.M., Codish, M.: A SAT-based approach to size change termination with global ranking functions. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 218–232. Springer (2008). https://doi.org/10.1007/978-3-540-78800-3_16

[5] Blanqui, F., Koprowski, A.: CoLoR: a Coq library on well-founded rewrite relations and its application to the automated verification of termination certificates. Math. Struct. Comput. Sci. **21**(4), 827–859 (2011). https://doi.org/10.1017/S0960129511000120

[6] Contejean, É., Courtieu, P., Forest, J., Pons, O., Urbain, X.: Automated certified proofs with CiME3. In: Schmidt-Schauß, M. (ed.) RTA 2011. LIPIcs, vol. 10, pp. 21–30. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2011). https://doi.org/10.4230/LIPIcs.RTA.2011.21

[7] Courtieu, P., Gbedo, G., Pons, O.: Improved matrix interpretation. In: van Leeuwen, J., Muscholl, A., Peleg, D., Pokorný, J., Rumpe, B. (eds.) SOFSEM 2010. LNCS, vol. 5901, pp. 283–295. Springer (2010). https://doi.org/10.1007/978-3-642-11266-9_24

[8] Dershowitz, N.: 33 examples of termination. In: Comon, H., Jounnaud, J.P. (eds.) Term Rewriting. pp. 16–26. Springer (1995). https://doi.org/10.1007/3-540-59340-3_2

[9] Endrullis, J., Waldmann, J., Zantema, H.: Matrix interpretations for proving termination of term rewriting. J. Autom. Reason. **40**(2-3), 195–220 (2008). https://doi.org/10.1007/s10817-007-9087-9

[10] Fuhs, C., Giesl, J., Middeldorp, A., Schneider-Kamp, P., Thiemann, R., Zankl, H.: Maximal termination. In: Voronkov, A. (ed.) RTA 2008. LNCS, vol. 5117, pp. 110–125. Springer (2008). https://doi.org/10.1007/978-3-540-70590-1_8

[11] Giesl, J., Aschermann, C., Brockschmidt, M., Emmes, F., Frohn, F., Fuhs, C., Hensel, J., Otto, C., Plücker, M., Schneider-Kamp, P., Ströder, T., Swiderski, S., Thiemann, R.: Analyzing program termination and complexity automatically with AProVE. J. Autom. Reason. **58**(1), 3–31 (2017). https://doi.org/10.1007/s10817-016-9388-y

[12] Giesl, J., Rubio, A., Sternagel, C., Waldmann, J., Yamada, A.: The termination and complexity competition. In: Beyer, D., Huisman, M., Kordon, F., Steffen, B. (eds.) TACAS 2019 (3). LNCS, vol. 11429, pp. 156–166. Springer (2019). https://doi.org/10.1007/978-3-030-17502-3_10

[13] Giesl, J., Thiemann, R., Schneider-Kamp, P.: The dependency pair framework: Combining techniques for automated termination proofs. In: Baader, F., Voronkov, A. (eds.) LPAR 2004. LNCS, vol. 3452, pp. 301–331. Springer (2004). https://doi.org/10.1007/978-3-540-32275-7_21

[14] Giesl, J., Thiemann, R., Schneider-Kamp, P., Falke, S.: Mechanizing and improving dependency pairs. J. Autom. Reason. **37**(3), 155–203 (2006). https://doi.org/10.1007/s10817-006-9057-7

[15] Gutiérrez, R., Lucas, S.: mu-term: Verify termination properties automatically (system description). In: Peltier, N., Sofronie-Stokkermans, V. (eds.) IJCAR 2020 (2). LNCS, vol. 12167, pp. 436–447. Springer (2020). https://doi.org/10.1007/978-3-030-51054-1_28

[16] Hirokawa, N., Middeldorp, A.: Dependency pairs revisited. In: van Oostrom, V. (ed.) RTA 2004. LNCS, vol. 3091, pp. 249–268. Springer (2004). https://doi.org/10.1007/978-3-540-25979-4_18

[17] Hirokawa, N., Middeldorp, A.: Polynomial interpretations with negative coefficients. In: Buchberger, B., Campbell, J.A. (eds.) AISC 2004. LNAI, vol. 3249, pp. 185–198. Springer (2004). https://doi.org/10.1007/

978-3-540-30210-0_16

[18] Hofbauer, D., Waldmann, J.: Termination of string rewriting with matrix interpretations. In: Pfenning, F. (ed.) RTA 2006. LNCS, vol. 4098, pp. 328–342. Springer (2006). https://doi.org/10.1007/11805618_25

[19] Jouannaud, J., Rubio, A.: The higher-order recursive path ordering. In: LICS 1999. pp. 402–411. IEEE Computer Society (1999). https://doi.org/10.1109/LICS.1999.782635

[20] Kop, C., van Raamsdonk, F.: Dynamic dependency pairs for algebraic functional systems. Log. Methods Comput. Sci. **8**(2) (2012). https://doi.org/10.2168/LMCS-8(2:10)2012

[21] Kop, C., Vale, D.: Tuple interpretations for higher-order complexity. In: Kobayashi, N. (ed.) FSCD 2021. LIPIcs, vol. 195, pp. 31:1–31:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021). https://doi.org/10.4230/LIPIcs.FSCD.2021.31

[22] Koprowski, A., Waldmann, J.: Max/plus tree automata for termination of term rewriting. Acta Cybern. **19**(2), 357–392 (2009)

[23] Korp, M., Sternagel, C., Zankl, H., Middeldorp, A.: Tyrolean Termination Tool 2. In: Treinen, R. (ed.) RTA 2009. LNCS, vol. 5595, pp. 295–304. Springer (2009). https://doi.org/10.1007/978-3-642-02348-4_21

[24] Kusakari, K., Nakamura, M., Toyama, Y.: Argument filtering transformation. In: Nadathur, G. (ed.) PPDP 1999. LNCS, vol. 1702, pp. 47–61. Springer (1999). https://doi.org/10.1007/10704567_3

[25] Lankford, D.: Canonical algebraic simplification in computational logic. Tech. Rep. ATP-25, University of Texas (1975)

[26] Lucas, S., Gutiérrez, R.: Automatic synthesis of logical models for order-sorted first-order theories. J. Autom. Reason. **60**(4), 465–501 (2018). https://doi.org/10.1007/s10817-017-9419-3

[27] Manna, Z., Ness, S.: On the termination of Markov algorithms. In: the 3rd Hawaii International Conference on System Science. pp. 789–792 (1970)

[28] de Moura, L.M., Bjørner, N.: Z3: an efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer (2008). https://doi.org/10.1007/978-3-540-78800-3_24

[29] Sternagel, C., Thiemann, R.: The certification problem format. In: Benzmüller, C., Paleo, B.W. (eds.) UITP 2014. EPTCS, vol. 167, pp. 61–72 (2014). https://doi.org/10.4204/EPTCS.167.8

[30] Sternagel, C., Thiemann, R.: Formalizing monotone algebras for certification of termination and complexity proofs. In: Dowek, G. (ed.) RTA-TLCA 2014. LNCS, vol. 8560, pp. 441–455. Springer (2014). https://doi.org/10.1007/978-3-319-08918-8_30

[31] Stump, A., Sutcliffe, G., Tinelli, C.: StarExec: A cross-community infrastructure for logic solving. In: Demri, S., Kapur, D., Weidenbach, C. (eds.) IJCAR. LNCS, vol. 8562, pp. 367–373. Springer (2014). https://doi.org/10.1007/978-3-319-08587-6_28

[32] Thiemann, R., Schöpf, J., Sternagel, C., Yamada, A.: Certifying the Weighted Path Order (Invited Talk). In: Ariola, Z.M. (ed.) FSCD 2020.

LIPIcs, vol. 167, pp. 4:1–4:20. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2020). https://doi.org/10.4230/LIPIcs.FSCD.2020.4

[33] Thiemann, R., Sternagel, C.: Certification of termination proofs using CeTA. In: Berghofer, S., Nipkow, T., Urban, C., Wenzel, M. (eds.) TPHOLs 2009. LNCS, vol. 5674, pp. 452–468. Springer (2009). https://doi.org/10.1007/978-3-642-03359-9_31

[34] The termination problem data base, http://termination-portal.org/wiki/TPDB

[35] Watson, T., Goguen, J., Thatcher, J., Wagner, E.: An initial algebra approach to the specification, correctness, and implementation of abstract data types. In: Current Trends in Programming Methodology. Prentice Hall (1976)

[36] Yamada, A.: Multi-dimensional interpretations for termination of term rewriting. In: Platzer, A., Sutcliffe, G. (eds.) CADE-28. LNCS, vol. 12699, pp. 273–290. Springer (2021). https://doi.org/10.1007/978-3-030-79876-5_16

[37] Yamada, A., Kusakari, K., Sakabe, T.: Nagoya Termination Tool. In: Dowek, G. (ed.) RTA-TLCA 2014. LNCS, vol. 8560, pp. 466–475. Springer (2014). https://doi.org/10.1007/978-3-319-08918-8_32

[38] Yamada, A., Kusakari, K., Sakabe, T.: A unified order for termination proving. Sci. Comput. Program. **111**, 110–134 (2015). https://doi.org/10.1016/j.scico.2014.07.009

[39] Zantema, H.: Termination of term rewriting: interpretation and type elimination. J. Symb. Comput. **17**(1), 23–50 (1994). https://doi.org/10.1006/jsco.1994.1003

[40] Zantema, H.: The termination hierarchy for term rewriting. Appl. Algebr. Eng. Comm. Compt. **12**(1/2), 3–19 (2001). https://doi.org/10.1007/s002000100061