Formalizing Monotone Algebras for Certification of Termination and Complexity Proofs*

Christian Sternagel and René Thiemann

Institute of Computer Science, University of Innsbruck, Austria
{christian.sternagel,rene.thiemann}@uibk.ac.at

Abstract. Monotone algebras are frequently used to generate reduction orders in automated termination and complexity proofs. To be able to certify these proofs, we formalized several kinds of interpretations in the proof assistant Isabelle/HOL. We report on our integration of matrix interpretations, arctic interpretations, and nonlinear polynomial interpretations over various domains, including the reals.

1 Introduction

Since the first termination competition[1] in 2004 it is of great interest whether a proof – that has been automatically generated by a termination or complexity tool – is indeed correct. The increasing complexity of generated proofs makes certification (i.e., checking correctness) more and more tedious for humans. Hence the interest in automated certification of termination and complexity proofs. This led to the general approach of using proof assistants for certification.

In this paper we present one of the key ingredients of our certifier, CeTA [34], namely the machinery for checking order constraints for (weakly) monotone algebras in the form of polynomial, matrix, and arctic interpretations. These constraints frequently arise in both termination and complexity proofs. For example, during the full run on the termination problem database in 2013, 3759 certifiable proofs have been generated. In 3170 of these proofs interpretations are used. Hence, they would not be certifiable by CeTA without the results of this paper.

In order to properly certify such proofs – and not just implement an independent but untrusted machinery for constraint checking – we take a two-phase approach. In the first phase, we prove general properties in Isabelle/HOL [25]. For example, we show that indeed all of the above interpretations are sound, i.e., they may be used for termination proofs.

In the second phase, we have to check concrete applications of interpretations on a concrete set of constraints. For example, at this point we need to ensure monotonicity of a given polynomial or to validate the growth rate of some matrix interpretation. To this end, we develop appropriate algorithms, prove them correct within Isabelle/HOL, and then invoke Isabelle's code generator [11] to obtain our certifier CeTA. The consequences of applying code generation are twofold:

* Supported by the Austrian Science Fund (FWF) projects P22767 and J3202.

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

while we obtain a high execution speed, there is the additional requirement that all the algorithms that are used for certification have to be fully executable (in the sense of functional programming).

Contribution and Overview. After giving some preliminaries in Section 2, we present our main contributions. In Section 3 we start with a generic formalization of polynomial and matrix interpretations that may be instantiated by several carriers like the naturals, the rationals, and the reals; we also support recent monotonicity criteria that are not present in other certifiers. Our integration of arctic interpretations (Section 4) reveals how the theory on arctic naturals and arctic integers can be unified. Moreover, it shows how to support arctic interpretations which are monotone in presence of a fresh binary symbol, a novelty. We further report on how we achieve executability for an interesting subset of the real numbers (Section 5). At this point, we also have to develop algorithms for computing n-th roots of numbers in order to efficiently factor numbers. Afterwards, we present our work on certifying complexity proofs (Section 6): as far as we know CeTA is the first certifier which supports complexity proofs at all. We finally conclude in Section 7.

All of the proofs that are presented (or omitted) in the following have been made available in the archive of formal proofs [29,30,33] or in IsaFoR,[2] an Isabelle/HOL formalization of rewriting. We further provide example termination and complexity proofs[3] which show applications of the various kinds of interpretations and can all be certified by CeTA.

2 Preliminaries

We assume familiarity with term rewriting (see, e.g., Baader and Nipkow [1]) but briefly recall notions that are used in the following. Terms are defined inductively: a *term* is either a variable x or is constructed by applying a function symbol f from the *signature* $\mathcal{F}$ to a list of argument terms $f(t_1, \ldots, t_n)$.

A pair of terms (s, t) is sometimes considered a *(rewrite) rule*, then we write $s \to t$. A set $\mathcal{R}$ of rules is called a *term rewrite system* (TRS for short). In contrast to many authors, we do not assume any a priori restrictions on rules of TRSs (the most frequent ones being that the left-hand side of a rule is not a variable and that rules do not introduce fresh variables on their right-hand side; both or either of the previous conditions are sometimes referred to as *variable condition* in the literature). Whenever there are restrictions, we mention them explicitly. TRSs induce a *rewrite relation* by closing their rules under contexts and substitutions. More precisely the rewrite relation of $\mathcal{R}$, denoted by $\to_{\mathcal{R}}$, is defined inductively by $s \to_{\mathcal{R}} t$ whenever there are a rule $\ell \to r \in \mathcal{R}$, a context C, and a substitution σ such that $s = C[\ell\sigma]$ and $t = C[r\sigma]$.

[2] http://cl-informatik.uibk.ac.at/software/ceta
[3] http://cl-informatik.uibk.ac.at/software/ceta/experiments/interpretations

A *semiring* is a structure $(A, +, \cdot, 0, 1)$ such that $(A, +, 0)$ is a commutative monoid with neutral element 0 and $(A, \cdot, 1)$ is a monoid with neutral element 1. Moreover, $\cdot$ distributes over $+$, $0 \neq 1$, and $0 \cdot x = x \cdot 0 = 0$ for all $x \in A$.

An *($\mathcal{F}$-)algebra* $\mathcal{A}$ is a carrier set A equipped with an interpretation function $f_{\mathcal{A}} : A^n \to A$ for every n-ary $f \in \mathcal{F}$. We call an algebra $\mathcal{A}$ *monotone* w.r.t. a binary relation $>$ when all interpretation functions are monotone, i.e., $f_{\mathcal{A}}(\ldots, a, \ldots) > f_{\mathcal{A}}(\ldots, b, \ldots)$ whenever $a > b$. A *well-founded monotone algebra* is a monotone algebra $(\mathcal{A}, >)$ such that $>$ is well-founded. For any algebra $\mathcal{A}$, terms can be interpreted w.r.t. an assignment α, written $[t]_\alpha$. Then, $s >_{\mathcal{A}} t$ denotes $[s]_\alpha > [t]_\alpha$ for all α.

A binary relation $\to$ is *terminating* (or *well-founded*) if there are no infinite derivations $a_1 \to a_2 \to a_3 \to \cdots$. Given two binary relations $\to_\alpha$, $\to_\beta$ we write $\to_\alpha / \to_\beta$ to abbreviate $\to_\beta^* \cdot \to_\alpha \cdot \to_\beta^*$, i.e., the rewrite relation of $\to_\alpha$ *relative to* $\to_\beta$. Termination of $\to_\alpha / \to_\beta$ is also called relative termination of $\to_\alpha$ w.r.t. $\to_\beta$.

We call a pair of two orders on terms $(\succ, \succeq)$ a *reduction pair* whenever it satisfies the following requirements: $\succ$ is well-founded, $\succeq$ and $\succ$ are compatible (i.e., $\succeq \cdot \succ \subseteq \succ$) and stable (i.e., closed under substitutions), and $\succeq$ is monotone (i.e., closed under contexts). If in addition $\succ$ is monotone, we call $(\succ, \succeq)$ a *monotone* reduction pair. Reduction pairs are employed for termination proofs inside the dependency pair framework, monotone reduction pairs for direct termination and complexity proofs.

3 Polynomial and Matrix Interpretations

Two widely used approaches to synthesize reduction pairs are polynomial interpretations (Lankford [18]) and matrix interpretations (Endrullis et al. [8]).

To support polynomial interpretations within CeTA, we formalized nonlinear multivariate polynomials [30] within Isabelle/HOL. Since similar tasks have already been conducted CoLoR [3] and Coccinelle [6] (using the approach of CiME [7]), we just shortly mention two distinguishing features of our work.

A formalization of polynomial orders has already been described by Blanqui and Koprowski [3]. Whereas their formalization fixes the carrier to $\mathbb{N}$, our polynomial orders are parametric in the carrier, cf. theory `Poly_Order`. Hence, we can treat polynomial orders over $\mathbb{N}$, $\mathbb{Q}$, and $\mathbb{R}$ within the same framework by just instantiating the carrier to the respective ordered semiring. Here, for both $\mathbb{Q}$ and $\mathbb{R}$ we use *δ-orders* as the strict order to achieve well-foundedness – as described by Lucas [20]: $x >_\delta y := x - y \geq \delta \wedge y \geq 0$ where δ is some fixed positive number. Notice that each carrier may have its own specialties, e.g., x^2 is monotone over $\mathbb{N}$, but not monotone for δ-orders if $\delta < 1$. This required the addition of properties in the parametric setting. For example, we added a Boolean parameter *power-mono* which describes whether polynomials like x^k with $k > 1$ are strictly monotone; it is always satisfied for $\mathbb{N}$ but demands $\delta \geq 1$ for $\mathbb{Q}$ and $\mathbb{R}$.

As far as we know we provide the first formalization of the improved monotonicity criteria for polynomials over $\mathbb{N}$ of Neurauter et al. [24] that can ensure

monotonicity of polynomials like $2x^2 - x$ which are not monotone over $\mathbb{Q}$ and $\mathbb{R}$. We made them available in the archive of formal proofs [30], theory `NZM`.

To support matrix interpretations, we basically follow the ideas of Courtieu et al. [7], i.e., we integrate matrix interpretations as linear polynomial interpretations where the carrier consists of matrices. To this end, we first developed a list-based and executable formalization of matrices [29] within Isabelle/HOL and afterwards connected it in the theory `Linear_Poly_Order` to obtain matrix interpretations in IsaFoR. Again, one of the distinguishing features of our work is the parametric carrier – for example in [7] the carrier is fixed to $\mathbb{N}$. Note that the demand for other carriers like $\mathbb{Q}$ and $\mathbb{R}$ was clearly shown by Neurauter and Middeldorp [23]: matrix interpretations over $\mathbb{R}$ are strictly more powerful than those over $\mathbb{Q}$ which in turn are strictly more powerful than those over $\mathbb{N}$.

Having developed the abstract results on these interpretations, it was easy to integrate executable criteria within CeTA that check applications of polynomial or matrix interpretations within concrete termination proofs – if the carrier consists of (matrices over) rational or natural numbers. However, more work had to be done for the reals. Before we discuss these problems in Section 5, we consider another kind of semiring in the next section.

4 Arctic Interpretations

The semirings $(\mathbb{A}_{\mathbb{N}}, \max, +, -\infty, 0)$ and $(\mathbb{A}_{\mathbb{Z}}, \max, +, -\infty, 0)$ are called *arctic semiring* and *arctic semiring below zero*, respectively. Here, $\mathbb{A}_A$ denotes the extension of A by the element $-\infty$, i.e., $A \cup \{-\infty\}$, $\max\{x, -\infty\} = x$, and $x + -\infty = -\infty + x = -\infty$ for all x. Waldmann and Koprowski [16] first used these semirings in the well-founded monotone algebra setting.

In the following we unify and extend (see Sternagel and Thiemann [28] for an earlier account) the arctic interpretations introduced by Waldmann and Koprowski. To do so, we first introduce the notion of an ordered arctic semiring.

Definition 1. *Let $(A, +, \cdot, 0, 1)$ be a semiring. Then an* ordered arctic semiring, *denoted by $(A, +, \cdot, 0, 1, >, \geq, \mathsf{pos})$, satisfies the additional requirements:*

- *$\geq$ is reflexive and transitive; $> \cdot \geq \subseteq >$ and $\geq \cdot > \subseteq >$*
- *$1 \geq 0$; $\neg\mathsf{pos}(0)$; $\mathsf{pos}(1)$; $x > 0$; $x \geq 0$; and $x = 0$ whenever $0 > x$*
- *$+$ is left-monotone w.r.t. $\geq$, i.e., $x + z \geq y + z$ whenever $x \geq y$*
- *$+$ is monotone w.r.t. $>$, i.e., $w + x > y + z$ whenever $w > y$ and $x > z$*
- *$\cdot$ is left- and right-monotone w.r.t. $\geq$ and left-monotone w.r.t. $>$*
- *$\mathsf{pos}(x + y)$ whenever $\mathsf{pos}(x)$; and $\mathsf{pos}(x \cdot y)$ whenever $\mathsf{pos}(x)$ and $\mathsf{pos}(y)$*
- *$\{(x, y) \mid x > y \wedge \mathsf{pos}(y)\}$ is well-founded*

Interpretation into an ordered arctic semiring yields a reduction pair.

Theorem 2. *Let $\mathcal{A}$ be an algebra over an ordered arctic semiring with interpretations $f_{\mathcal{A}}(x_1, \ldots, x_n) = f_0 + f_1 \cdot x_1 + \cdots + f_n \cdot x_n$ such that $\mathsf{pos}(f_i)$ for some $0 \leq i \leq n$. Then $(\succ_{\mathcal{A}}, \succeq_{\mathcal{A}})$ is a reduction pair.*

Examples for ordered arctic semirings are given in the following:

Example 3. The arctic semiring, the arctic semiring below zero, and the arctic rational semiring $(\mathbb{A}_{\mathbb{Q}}, \max, +, -\infty, 0)$ are ordered arctic semirings for $x > y := (y = -\infty \vee (x \neq -\infty \wedge x >_A y))$ (where $>_A$ is $>_{\mathbb{N}}$ and $>_{\mathbb{Z}}$ for naturals and integers, respectively; and $x >_\delta y$ for some $\delta > 0$ for the rationals), $x \geq y := (y = -\infty \vee (x \neq -\infty \wedge x \geq_{\mathbb{N}/\mathbb{Z}/\mathbb{Q}} y))$, and $\mathsf{pos}(x) := (x \neq -\infty \wedge x \geq_{\mathbb{N}/\mathbb{Z}/\mathbb{Q}} 0)$.

Note that the ordered arctic semiring over $\mathbb{A}_{\mathbb{Q}}$ together with Theorem 2, unifies and extends Theorems 12 and 14 of Waldmann and Koprowski [16]. The main advantage of our approach is that we only require interpretations to have at least one positive f_i (instead of always requiring the constant part f_0 to be positive). Although our result is slightly more general, we could completely reuse the original proof structure of [16] to formalize Theorem 2.

Waldmann and Koprowski also showed that for string rewriting (i.e, terms over a signature of function symbols that are at most unary) arctic interpretations are monotone and thus may be used for rule removal on standard termination problems. In order to apply this technique in the dependency pair framework together with *usable rules* we also need $\mathcal{C}_\mathcal{E}$-compatibility, i.e., the rules of the TRS $\mathcal{C}_\mathcal{E} = \{\mathsf{c}(x, y) \to x, \mathsf{c}(x, y) \to y\}$ must be oriented where c is some fresh symbol. But by considering $\mathcal{C}_\mathcal{E}$ we leave the domain of string rewriting.

Nevertheless, we want to obtain $\mathcal{C}_\mathcal{E}$-compatibility for monotone arctic interpretations. As an application consider the technique of Giesl et al. [9, Thm. 28] which allows us to remove all non-usable rules and all strictly oriented rules from a dependency pair problem, provided that the dependency pairs and rules are weakly oriented, the rules in $\mathcal{C}_\mathcal{E}$ are strictly oriented, and $\succ$ is monotone. To this end we first need signature extensions for relative termination (see also [31]).

Theorem 4 (Signature Extensions Preserve Relative Termination). *Let $\mathcal{R}$ and $\mathcal{S}$ be TRSs over a common signature $\mathcal{F}$. Moreover, suppose that no right-hand side of a rule in $\mathcal{S}$ introduces fresh variables. Then $\to_\mathcal{R}/\to_\mathcal{S}$ terminates for terms over arbitrary extensions of $\mathcal{F}$, whenever it does so for terms over $\mathcal{F}$.*

The above statement is not true when $\mathcal{S}$ violates the variable condition.

Example 5. Consider $\mathcal{R} = \{\mathsf{a} \to \mathsf{b}\}$ relative to $\mathcal{S} = \{\mathsf{c} \to x\}$. Over the common signature $\mathcal{F} = \{\mathsf{a}/0, \mathsf{b}/0, \mathsf{c}/0\}$ we have relative termination. However, extending $\mathcal{F}$ by $\{\mathsf{f}/2\}$ yields the infinite derivation where $C = \mathsf{f}(\mathsf{b}, \Box)$.

$$s = \mathsf{f}(\mathsf{a}, \mathsf{c}) \to_\mathcal{R} C[\mathsf{c}] \to^*_\mathcal{S} C[s] \to_\mathcal{R} C[C[\mathsf{c}]] \to^*_\mathcal{S} \cdots$$

Finally, we have to show that for monotone $\succ$ we get $\mathcal{C}_\mathcal{E}$-compatibility.

Lemma 6. *Consider a reduction pair $(\succ, \succeq)$ and TRSs $\mathcal{R}$, $\mathcal{S}$ over a common signature of at most unary function symbols $\mathcal{F}$ such that no rule of $\mathcal{S}$ introduces fresh variables. Moreover, let $\mathcal{R} \subseteq \succ$ and $\mathcal{S} \subseteq \succeq$. Then monotonicity of $\succ$ implies termination of $\to_{\mathcal{C}_\mathcal{E} \cup \mathcal{R}}/\to_{\mathcal{C}_\mathcal{E} \cup \mathcal{S}}$.*

Proof. By Theorem 4 together with monotonicity of $\succ$, we obtain termination of $\to_{\mathcal{R}}/\to_{\mathcal{S}}$ for arbitrary extensions of $\mathcal{F}$. Consider a lexicographic path order where all symbols are equal in precedence in combination with an argument filter that projects unary function symbols to their argument and keeps all other symbols unchanged. Then, this combination yields a monotone reduction pair $(>, \geq)$. Moreover, $\mathcal{R}$ and $\mathcal{S}$ are compatible with $\geq$ (since all terms are collapsed to a single variable or constant). Since also $\mathcal{C}_\mathcal{E} \subseteq >$ we obtain relative termination of $\to_{\mathcal{C}_\mathcal{E}}$ w.r.t. $\to_{\mathcal{R}\cup\mathcal{S}}$ and thus termination of $\to_{\mathcal{C}_\mathcal{E}\cup\mathcal{R}}/\to_{\mathcal{C}_\mathcal{E}\cup\mathcal{S}}$. □

While the above lemma does not quite yield $\mathcal{C}_\mathcal{E}$-compatibility, it can be used to show that from every reduction pair $(\succ, \succeq)$ that satisfies the above conditions we obtain a corresponding $\mathcal{C}_\mathcal{E}$-compatible reduction pair $(\succ', \succeq')$. More specifically, take $\succ' = (\to_{\mathcal{C}_\mathcal{E}\cup\mathcal{R}}/\to_{\mathcal{C}_\mathcal{E}\cup\mathcal{S}})^+$ and $\succeq' = \to^*_{\mathcal{C}_\mathcal{E}\cup\mathcal{S}}$.

Now if we start from a monotone reduction pair $(\succ, \succeq)$, and a set of rules $\mathcal{P}$ over an at most unary signature and take $\mathcal{R} = \mathcal{P} \cap \succ$ and $\mathcal{S} = \mathcal{P} \cap \succeq$, then the resulting reduction pair $(\succ', \succeq')$ is $\mathcal{C}_\mathcal{E}$-compatible, monotone, and orients all rules of $\mathcal{P}$ that were also oriented by the original reduction pair.

5 Interpretations over the Reals

Whereas all basic operations on $\mathbb{Q}$ are executable, this is not the case for $\mathbb{R}$. To solve this problem, automated tools only work on a subset of the real numbers [21,36,37]. For example, in the setting of Zankl and Middeldorp [36] numbers may be chosen from $\mathbb{Q}[\sqrt{2}]$, the field extension of $\mathbb{Q}$ by $\sqrt{2}$. All these numbers are of the form $p + q\sqrt{2}$ where p and q range over $\mathbb{Q}$. In [37], Zankl et al. allow even more generic forms, e.g., where $\sqrt{2}$ may be replaced by $\sqrt{b}$ for a fixed natural number b with $\sqrt{b} \notin \mathbb{Q}$, i.e., we consider $\mathbb{Q}[\sqrt{b}]$.

Fixing the base b, all numbers in $\mathbb{Q}[\sqrt{b}]$ can be represented by pairs (p, q) (encoding $p + q\sqrt{b}$), where all ordered semiring operations are again executable. For example, $(p_1, q_1) \cdot (p_2, q_2) := (p_1p_2 + bq_1q_2, p_1q_2 + p_2q_1)$ and $(p, q) > 0 := (p \geq 0 \wedge q > 0) \vee (p > 0 \wedge q \geq 0) \vee (p \geq 0 \wedge q < 0 \wedge p^2 > bq^2) \vee (p \leq 0 \wedge q > 0 \wedge p^2 < bq^2)$ where the definition of $(p, q) > 0$ can be used to decide the comparison $p_1 + q_1\sqrt{b} > p_2 + q_2\sqrt{b}$ by choosing $p = p_1 - p_2$ and $q = q_1 - q_2$ [37, Def. 10].

A larger subset of the real numbers, namely the algebraic real numbers, has been formalized by Cohen [5]. However, since this formalization has been conducted using Coq, it cannot easily be integrated into our Isabelle/HOL development. Moreover, as far as we know, all real numbers which are currently generated by automated termination tools are contained in $\mathbb{Q}[\sqrt{b}]$ for some fixed b which can be chosen in the configuration of the tool. Hence, for certification it suffices to formalize this subset of the real numbers. Although a full integration of real algebraic numbers in Isabelle might be welcome, we pursued the more lightweight approach which was sufficient to support $\mathbb{Q}[\sqrt{b}]$.

We consider some alternatives for representing $\mathbb{Q}[\sqrt{b}]$ in Isabelle/HOL. The first alternative is to fix some b and create a new type $pair_b$ consisting of pairs of rational numbers. Then addition, multiplication, comparison, etc. are defined

as above, and we have to prove that this new type forms an ordered semiring (one that is completely independent from the real numbers). The disadvantage of this approach is that for each b we have to define a new type. As we can only define finitely many types within a finite Isabelle formalization, our certifier will be limited to a fixed number of choices for b.

For higher flexibility, we can alternatively create a type *triples* that additionally takes the parameter b as third component. The problem in this approach is to give a total definition for all operations, e.g., what should be the result of $(1,1,2)\cdot(1,1,3)$, i.e., how can we represent the number $(1+1\cdot\sqrt{2})\cdot(1+1\cdot\sqrt{3}) = 1+\sqrt{2}+\sqrt{3}+\sqrt{6}$ as a triple $p+q\sqrt{b}$ for suitable values of $p, q \in \mathbb{Q}$ and $b \in \mathbb{N}$.

A third possibility would be to not create a type at all, but use locales [2] and explicit domains. We do not go into the details here, but just mention that this approach is currently not applicable, since other parts of the formalization – like the theories on nonlinear polynomial interpretations – utilize the type class for semirings, and do not support the more flexible locales.

Our final solution is to not define a new type to form the semirings $\mathbb{Q}[\sqrt{b}]$, but to perform data refinement [10] instead, i.e., provide an implementation type for the reals. This has the following advantages: For a start, the Isabelle distribution already contains the result that $\mathbb{R}$ is an ordered semiring. Thus all the properties of the reals can be used when formalizing monotone algebras over the reals. Moreover, our implementation can be partial, e.g., we do not have to support the multiplication of arbitrary numbers like $(1+\sqrt{2})\cdot(1+\sqrt{3})$. Finally, as soon as a better implementation is available, we can just replace the current one by the new one, and do not have to change the theories which show that the reals can be used to generate monotone algebras.

5.1 A first implementation of $\mathbb{R}$ via triples (p, q, b)

In the following we implement the reals by the type *mini-alg* containing all triples $(p, q, b) \in \mathbb{Q} \times \mathbb{Q} \times \mathbb{N}$ that satisfy the invariant $q = 0 \vee \sqrt{b} \notin \mathbb{Q}$. Such a quotient type is easily created and accessed via the lifting and transfer package [15].

For this data refinement, we first have to declare how *mini-alg* is mapped into the reals. This is done by a function *real-of* : *mini-alg* $\to \mathbb{R}$, defined as:

$$\textit{real-of}\ (p, q, b) = p + q\sqrt{b}$$

Next, we tell the code generator that *real-of* should be seen as the constructor for real numbers, i.e., from now on we consider the reals as being defined by the datatype definition *datatype* $\mathbb{R} =$ *real-of mini-alg* where *real-of* is the unique constructor which takes a triple as input. Afterwards, the desired operations on reals must be implemented via lemmas on this new "constructor" *real-of*. E.g., the unary minus operation is implemented by proving the following lemma:

$$-\textit{real-of}\ (p, q, b) = \textit{real-of}\ (-p, -q, b)$$

Often, we only implement partial operations, e.g., for some binary operations we require triples with compatible bases. For example, addition is defined by the

lemma

$$\begin{aligned}&\mathit{real\text{-}of}\ (p_1, q_1, b_1) + \mathit{real\text{-}of}\ (p_2, q_2, b_2) = \\ &\quad \mathit{if}\ \mathit{compatible}\ (p_1, q_1, b_1)\ (p_2, q_2, b_2) \\ &\quad \mathit{then}\ (\mathit{if}\ q_1 = 0\ \mathit{then}\ \mathit{real\text{-}of}\ (p_1 + p_2, q_2, b_2)\ \mathit{else}\ \mathit{real\text{-}of}\ (p_1 + p_2, q_1 + q_2, b_1)) \\ &\quad \mathit{else}\ \mathit{abort}\ (\lambda\_.\ \mathit{real\text{-}of}\ (p_1, q_1, b_1) + \mathit{real\text{-}of}\ (p_2, q_2, b_2))\ ()\end{aligned}$$

where $\mathit{compatible}\ (p_1, q_1, b_1)\ (p_1, q_2, b_2)$ is defined as $q_1 = 0 \vee q_2 = 0 \vee b_1 = b_2$, and $\mathit{abort}\ f\ x = f\ x$. That is, two triples are compatible iff one of them encodes a rational number, or the bases are identical. The equation for *abort* allows us to prove the above lemma, but is not used to generate code, since this would lead to nontermination in case of incompatible triples. Instead, the code generator issues an appropriate error in the target language at this point. This trick was already described by Lochbihler [19].

Above we defined several operations on reals like addition, multiplication, greater-than, and a mapping from $\mathbb{Q}$ into $\mathbb{R}$. Some of the binary operations are partial and require compatible triples as input. However, the above mentioned operations do not make use of the invariant of *mini-alg*. The invariant is required for operations like equality and inverse. For example, for the multiplicative inverse of a triple (p, q, b) we use the triple $(p/d, -(q/d), b)$ where the divisor is $d = p^2 - bq^2$. To ensure that $d \neq 0$ whenever $\mathit{real\text{-}of}\ (p, q, b) \neq 0$ we need the invariant that $\sqrt{b}$ is irrational. Similarly, also for equality – which is defined as $p_1 = p_2 \wedge q_1 = q_2 \wedge (q_1 = 0 \vee b_1 = b_2)$ for compatible triples (p_1, q_1, b_1) and (p_2, q_2, b_2) – we require the invariant that $\sqrt{b}$ is irrational. Otherwise, the above implementation of equality would return false for the inputs $(0, 1, 4)$ and $(2, 0, 4)$, but $\mathit{real\text{-}of}\ (0, 1, 4) = 0 + 1 \cdot \sqrt{4} = 2 = 2 + 0 \cdot \sqrt{4} = \mathit{real\text{-}of}\ (2, 0, 4)$.

So far, we defined all required field operations and comparisons, each of which is implemented by a constant number of operations on rational numbers. However, we are lacking a way to really construct irrational numbers. To this end we provide a partial implementation of the square root function that is only defined for input triples encoding rational numbers. The definition for nonnegative rational numbers with numerator n and denominator d is

$$\mathit{sqrt}\ (\frac{n}{d}, 0, b) = \mathit{if}\ \sqrt{nd} \in \mathbb{Z}\ \mathit{then}\ (\frac{\sqrt{nd}}{d}, 0, 0)\ \mathit{else}\ (0, \frac{1}{d}, nd) \tag{1}$$

where the case-analysis is solely performed to satisfy the invariant of triples of type *mini-alg*. In (1) we make use of a square root function on integers which can decide for a given integer i whether $\sqrt{i} \in \mathbb{Z}$ or not. If so, it also returns the resulting number, cf. Thiemann [32, Thm. 14]. We modified this square root function such that it can additionally compute $\lfloor\sqrt{i}\rfloor$ and $\lceil\sqrt{i}\rceil$. In this way, we are able to implement $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ on triples of type *mini-alg*.

In total, our implementation provides the following operations on reals: $+$, $-$, $\times$, $\cdot^{-1}$, $>$, $\geq$, $=$, $\lfloor\cdot\rfloor$, $\lceil\cdot\rceil$, and $\sqrt{\cdot}$. Only the last three require the computation of square roots. All binary operations succeed if their operands are compatible (which is always the case if a fixed base b is chosen), and only the last operation is restricted to rational numbers as input. This implementation supports all operations that we require for monotone algebras except for one.

5.2 A second implementation of $\mathbb{R}$ via triples (p, q, b)

First note that CeTA not only accepts or rejects a termination proof, but also provides a detailed error message in case of rejection. To this end, we have to print numbers occurring in interpretations, i.e., we need a function $show : \mathbb{R} \to string$.

An easy solution might be to postulate such a function and provide an implementation via the axiom: $show\ (real\text{-}of\ (p, q, b)) =$ the string "p + q * sqrt(b)".

One might argue that adding this axiom is not really relevant, as it is only used for error messages. However, adding it immediately introduces an inconsistency in the logic:

$$\begin{aligned} &\text{“4 + 1 * sqrt(18)”} && = show\ (real\text{-}of\ (4, 1, 18)) && = show\ (4 + 1 \cdot \sqrt{18}) \\ = show\ (4 + 3 \cdot \sqrt{2}) && = show\ (real\text{-}of\ (4, 3, 2)) && = \text{“4 + 3 * sqrt(2)”} \end{aligned}$$

That is, the wrong fact that the first and the last string are identical is derivable.

As a consequence, we want to avoid this inconsistent axiom which stems from the fact that *real-of* is not injective, e.g., the number $\sqrt{18}$ can be represented by both $(0, 3, 2)$ and $(0, 1, 18)$. To this end, we define a new type of triples, *mini-alg-unique*. It is similar to *mini-alg* but adds another invariant: every triple (p, q, b) must satisfy $q = 0 \wedge b = 0 \vee q \neq 0 \wedge prime\text{-}product\ b$, where $prime\text{-}product\ b$ demands that b is a product of distinct primes. For example, 2 and $6 = 2 \cdot 3$ are prime products, but $18 = 2 \cdot 3 \cdot 3$ is not, since 3 occurs twice.

In the remainder of this section we assume that we perform data refinement of $\mathbb{R}$ by implementing it via triples of type *mini-alg-unique*. While most of the algorithms work as for *mini-alg*, we list the most important differences.

The main advantage of *mini-alg-unique* is that *real-of* is now injective. As a result, equality of reals can easily be implemented as equality of triples without checking for compatibility. For example, since $(1, 2, 3) \neq (2, 2, 2)$ we conclude $1 + 2 \cdot \sqrt{3} \neq 2 + 2 \cdot \sqrt{2}$. This also allows us to define a total function for comparisons which is implemented via $\lfloor \cdot \rfloor$: if the numbers are equal, then the result is determined, and otherwise we multiply both numbers iteratively by 1024 until there is a difference after applying $\lfloor \cdot \rfloor$. For example, the algorithm shows $1+2\cdot\sqrt{3} < 2+2\cdot\sqrt{2}$ since $\lfloor 1024\cdot(1+2\cdot\sqrt{3})\rfloor = 4571 < 4944 = \lfloor 1024\cdot(2+2\cdot\sqrt{2})\rfloor$.

As *real-of* is injective, it is now also possible to define *show* on the reals, and later on implement it for triples of type *mini-alg-unique*. To this end, assume we have already defined a function *mau-show* which pretty prints triples t as strings. The specification of *show* in the logic is

$$show\ x = (\mathit{if}\ \exists t.\ x = real\text{-}of\ t\ \mathit{then}\ mau\text{-}show\ (\mathit{THE}\ t.\ x = real\text{-}of\ t)\ \mathit{else}\ \text{“nothing”})$$

while its implementation is given by the lemma: $show\ (real\text{-}of\ t) = mau\text{-}show\ t$, where $\mathit{THE}\ t.\ P\ t$ results in the unique t satisfying $P\ t$, if such a t exists, and is undefined, otherwise. In the definition of *show*, existence of t is established before calling *mau-show* and uniqueness follows from the injectivity of *real-of*.

The last algorithm that requires an adaptation is the implementation of *sqrt*. The definition from (1) is not suitable any more, as $\sqrt{nd} \notin \mathbb{Z}$ does not guarantee nd to be a prime product. Therefore, a preprocessing is required which factors

every natural number m (like nd) into $m = s^2 \cdot p$ where $s, p \in \mathbb{N}$ and either p is a prime product or p is 1. In the latter case, $\sqrt{m}$ is the natural number s, and in the former case the triple $(0, s, p)$ represents the number $\sqrt{m}$ and also satisfies the invariant of *mini-alg-unique*.

For the factorization algorithm, we do not fully decompose m into prime factors – which would roughly require $\sqrt{m}$ iterations – but use the following algorithm, requiring only $\sqrt[3]{m}$ iterations. First check whether $\sqrt{m}$ is irrational. If not, then we are done by returning $(\sqrt{m}, 1)$. Otherwise, we check whether m has a factor between 2 and $\lfloor \sqrt[3]{m} \rfloor$. If we detect such a factor p, then we store p, and continue to search factors of m/p with a new upper bound of $\lfloor \sqrt[3]{m/p} \rfloor$. If there is no such factor, then we conclude that m is a prime product as follows: assume that m is not a prime product, i.e., $m = p \cdot p \cdot q$ for some prime p and natural number q. Then $q \neq 1$ since otherwise $\sqrt{m} = p \in \mathbb{N}$ is not irrational. Hence, m has both p and q as factors. But since we tested that m does not divide any of the numbers up to $\lfloor \sqrt[3]{m} \rfloor$, we know that both p and q are larger than $\sqrt[3]{m}$. Hence, $m = p \cdot p \cdot q > (\sqrt[3]{m})^3 = m$, a contradiction.

Note that, for implementing the factorization algorithm, we not only need the square root algorithm of [32, Sect. 6], but also require an algorithm to compute $\lfloor \sqrt[3]{m} \rfloor$. To this end, we extended the work of [32] to arbitrary n-th roots, i.e., we can check $\sqrt[n]{p} \in \mathbb{Q}$ and compute $\lfloor \sqrt[n]{p} \rfloor$ for every $n \in \mathbb{N}$ and $p \in \mathbb{Q}$. Here, Porter's [26] formalization of Cauchy's mean theorem was extremely helpful to show soundness of our n-th root algorithm. The algorithm itself uses a variant of Newton iteration to compute precise roots, which uses integer divisions where one usually works on rational or floating point numbers.

5.3 Summary

We performed data refinement to implement the subset $\mathbb{Q}[\sqrt{b}]$ of the reals as triples (p, q, b) representing $p + q \cdot \sqrt{b}$. The first implementation has the advantage of being more efficient, but several operations are only supported partially, where in binary operations the same basis $\sqrt{b}$ must be present. The second implementation is less partial and even allows to define a show function on reals, at the cost of having to perform a factorization of prime products, which we implemented via an algorithm with $\sqrt[3]{n}$ iterations. To this end, we also formalized a generic n-th root algorithm. This part of the formalization has been made available in the archive of formal proofs [33].

Using this formalization, we are able to certify each application of monotone algebras over the reals within termination proofs generated by T$_{\text{T}}$T$_2$ [17].

6 Complexity Proofs

Monotone algebras are not only a useful tool for termination analysis, but also for complexity analysis. In the following, we first introduce basic notions regarding complexity – including a modularity result of Zankl and Korp [35] – and afterwards provide details on complexity results for matrix interpretations and polynomial interpretations which have been integrated into IsaFoR and CeTA.

6.1 Complexity and modularity

To measure the complexity of a TRS $\mathcal{R}$, we use the following notions and ideas of [4,12,14]. The *derivation height* of a term w.r.t. some binary relation $\to$ on terms is defined as $\mathsf{dh}_{\to}(t) = \max\{n \mid \exists s.\, t \to^n s\}$ and measures the height of the derivation tree of t. The *derivational complexity* of $\to$ is given by $\mathsf{dc}_{\to}(n) = \max\{\mathsf{dh}_{\to}(t) \mid |t| \leq n\}$. That is, the maximal length of derivations with starting terms of size n is bounded by $\mathsf{dc}_{\to}(n)$. While in $\mathsf{dc}_{\to}(n)$ all starting terms are considered, allowing for terms like $\mathsf{double}^n(1)$, there is the alternative notion of *runtime complexity*, where starting terms are restricted to basic terms. In detail, $\mathsf{rc}_{\to,\mathcal{C}}(n) = \max\{\mathsf{dh}_{\to}(t) \mid |t| \leq n, t \in \mathcal{B}_{\mathcal{C}}\}$ where $\mathcal{B}_{\mathcal{C}}$ denotes the set of *basic terms*, i.e., terms of the form $f(c_1, \ldots, c_n)$, with all the c_i only built over symbols from $\mathcal{C}$. Here $\mathcal{C}$ typically is the set of constructors of the TRS of interest.

Note that all of $\mathsf{dh}_{\to}$, $\mathsf{dc}_{\to}$, and $\mathsf{rc}_{\to,\mathcal{C}}$ are only well-defined if max is applied to a finite set. However, this is not necessarily the case, as on the one hand $\to$ may be infinitely branching or nonterminating, and on the other hand, there might be infinitely many terms of size n if the signature is infinite. In order to avoid having to worry about these side-conditions within our formalization, we instead define the following function

$$\textit{deriv-bound-rel}_{\to}\,(SE, f) = (\forall n\ t.\, (t \in SE(n) \implies \nexists s.\, t \to^{f(n)+1} s))$$

checking whether a given function f is an upper bound for the complexity. Here, SE describes the set of starting elements depending on a natural number n, usually limiting the size of elements. We can easily model runtime and derivational complexity: $\textit{deriv-bound-rel}_{\to}\,(\lambda n.\, \{t \mid |t| \leq n\}, f)$ and $\textit{deriv-bound-rel}_{\to}\,(\lambda n.\, \{t \mid |t| \leq n, t \in \mathcal{B}_{\mathcal{C}}\}, f)$ express that $\mathsf{dc}_{\to}$ and $\mathsf{rc}_{\to,\mathcal{C}}$ are bounded by f, respectively.

The above definitions are contained in the theory `Complexity`, which also contains the first formalization of the modularity result by Zankl and Korp [35, Thm. 4.4]. Here, we stay in an abstract setting where $\to_1$, $\to_2$, and $\to_3$ are arbitrary binary relations (not necessarily ranging over terms), cf. the theorem *deriv-bound-relto-class-union* that is part of IsaFoR.

Theorem 7. *Let $\to_i$ be binary relations (with $i \in \{1,2,3\}$). Moreover, let $\textit{deriv-bound-rel}_{\to_1/(\to_2 \cup \to_3)}\,(SE, g_1)$ and $\textit{deriv-bound-rel}_{\to_2/(\to_1 \cup \to_3)}\,(SE, g_2)$ for two functions $g_1, g_2 \in \mathcal{O}(f)$. Then there is a function $g \in \mathcal{O}(f)$ such that $\textit{deriv-bound-rel}_{(\to_1 \cup \to_2)/\to_3}\,(SE, g)$.*

6.2 Complexity via monotone interpretations

In order to ensure complexity bounds via some monotone algebra $(\mathcal{A}, >)$ with carrier A, one first needs a function of type $A \to \mathbb{N}$ which bounds the number of decreases. To be more precise, in the generic setting for semirings within IsaFoR we require a function *bound* such that for each $a \in A$ there are no $a_1, a_2, \ldots$ such that $a > a_1 > \ldots > a_{bound(a)+1}$, and moreover *bound* has to grow linearly in its argument, cf. `Complexity_Carrier` for further details.

We defined various valid *bound* functions for the different kind of carriers. For example, we have chosen $bound_{\mathbb{N}}(n) = n$ for the naturals, $bound_{\delta}(x) = \lceil x/\delta \rceil$ for δ-orders on the rationals and reals, and $bound_{mat(A)}(m) = bound_A(||m||)$ for matrices over carrier A, where $||\cdot||$ denotes the linear norm of a matrix.

Obviously, whenever each reduction step $t \to s$ within a derivation corresponds to a decrease $[t]_\alpha > [s]_\alpha$ then $\mathsf{dh}_{\to}(t) \leq bound([t]_\alpha)$ for every assignment α and term t. Thus, $\mathsf{dh}_{\to_{\mathcal{R}}/\to_{\mathcal{S}}}(t) \leq bound([t]_\alpha)$ whenever $\ell >_{\mathcal{A}} r$ for each $\ell \to r \in \mathcal{R}$ and $\ell \geq_{\mathcal{A}} r$ for each $\ell \to r \in \mathcal{S}$. At this point in the formalization we fix α to be the zero-assignment α_0 where $\alpha_0(x) = 0$ for all x.

Since *bound* has to grow linearly in its argument, to get asymptotic bounds it suffices to estimate $[t]_{\alpha_0}$ for each $t \in SE(n)$, depending on n.

For *polynomial interpretations* we formalized the criterion of strongly linear interpretations of Hofbauer [13].

Theorem 8. *Let $\mathcal{F}$ be a subset of the signature. Whenever $f_{\mathcal{A}}(x_1, \ldots, x_n) = c_f + \sum_{i=1}^{n} x_i$ for each $f \in \mathcal{F}$ then*

- *$[t]_{\alpha_0}$ is linearly bounded in $|t|$ whenever $\mathcal{F}$ is the full signature.*
- *$[t]_{\alpha_0}$ is bounded by $\mathcal{O}(|t|^d)$ whenever $\mathcal{F} = \mathcal{C}$, $t \in \mathcal{B}_{\mathcal{C}}$, and d is the largest degree of a polynomial within the interpretation.*

The two alternatives have been formalized in `Poly_Order`, where the first one (*linear-bound*) is used for derivational complexity and the second one (*degree-bound*) for runtime complexity. Further note that the above theorem can be combined with several ordered semirings, so that currently CeTA can check complexity proofs involving polynomial interpretations over $\mathbb{N}$, $\mathbb{Q}$, and $\mathbb{R}$.

Furthermore, we also support complexity proofs via matrix interpretations. To be more precise, we provide the first formalization of the criterion of Moser, Schnabl, and Waldmann [22] that for upper triangular matrix interpretations we get an upper bound of $[t]_{\alpha_0} \in \mathcal{O}(|t|^d)$ where d is the dimension of the matrix.

To this end, we have first proven [22, Lem. 5] that $||m^n|| \in \mathcal{O}(n^{d-1})$ is satisfied for an upper triangular matrix m of dimension d. This fact has been made available as *upper-triangular-mat-pow-value* in `Matrix_Comparison` within the archive of formal proofs [29]. Here, we want to stress that the formalization has been much more verbose than the paper: in [22] the proof is two lines long, whereas the formalization takes 300 lines. However, this is not surprising since the two lines have been expanded to a more detailed paper proof (one full page) in Schnabl's PhD thesis [27], and even this proof contains a "straightforward" inner induction which is not spelled out.

In `Matrix_Comparison` we also prove that the linear norm is sub-multiplicative, i.e., $||m_1 \times m_2|| \leq ||m_1|| \cdot ||m_2||$, a property that is required to achieve [22, Thm. 6], but is not mentioned in the paper.

Again, all of our results have been proven in a generic way for several semirings, which includes the semirings on $\mathbb{N}$, $\mathbb{Q}$, and $\mathbb{R}$. In this way, we generalized [22, Thm. 6] which was only proven for the natural numbers. Especially the proof that the linear norm is sub-multiplicative required the development of a

completely new proof: at least three mathematical textbooks contain the same incomparable statement, which states the property for a whole class of norms, but only for the reals. However, we required the property only for the linear norm, but for matrices of type $A^{n\times m}$ where A is generic. Therefore, the proofs within the textbooks – which all use a limit construction on the reals – could not be formalized. Instead, we performed an inductive proof over the shared dimension of the matrices, cf. *linear-norm-submultiplicative* for more details.[4]

7 Conclusion

We presented an overview of our Isabelle/HOL formalization of interpretations over various carriers, which is part of the formalized library IsaFoR and employed in the fully verified certifier CeTA. The kinds of interpretations we support are linear polynomial interpretations, which also allow for matrix interpretations, and nonlinear polynomial interpretations. As we have shown above, supported carriers range from natural numbers, over integers and rational numbers, to real numbers, as well as corresponding arctic carriers. This unifies and extends previous work. Since CeTA needs to certify given proofs containing explicit numbers and interpretation functions we also had to take care that our formalization supports executable algorithms for all required operations (like addition, multiplication, various comparisons, the square root function, etc.). For real numbers this is not a trivial task. Our solution was to perform data refinement to a subset of the reals that suffices for our purposes. Finally we presented our formalization of complexity related results. In contrast to typical formulations in the literature, we only provide upper bounds, but in return do not have to care about well-definedness issues that would arise otherwise.

Acknowledgments. We are grateful to Bertram Felgenhauer for pointing us to Cauchy's mean theorem when proving soundness of our root algorithm.

The authors are listed in alphabetical order regardless of individual contributions or seniority.

References

1. F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.
2. C. Ballarin. Locales: A module system for mathematical theories. *J. Autom. Reasoning*, 52(2):123–153, 2014. doi:10.1007/s10817-013-9284-7.
3. F. Blanqui and A. Koprowski. CoLoR: a Coq library on well-founded rewrite relations and its application to the automated verification of termination certificates. *Math. Struct. Comp. Sci.*, 21(4):827–859, 2011. doi:10.1017/S0960129511000120.

[4] In the formalization there is a precondition that all elements have to be nonnegative. This is due to the fact that the linear norm is defined as the sum of all values in the matrix, without taking absolute values. This was done on purpose, so that we do not even have to require the existence of an absolute value function.

4. A. Cichon and P. Lescanne. Polynomial interpretations and the complexity of algorithms. In *Proc. 11th CADE*, volume 607 of *LNCS*, pages 139–147. Springer, 1992. doi:10.1007/3-540-55602-8_161.
5. C. Cohen. Construction of real algebraic numbers in Coq. In *Proc. 3rd ITP*, volume 7406 of *LNCS*, pages 67–82. Springer, 2012. doi:10.1007/978-3-642-32347-8_6.
6. Évelyne Contejean, Pierre Courtieu, Julien Forest, Olivier Pons, and Xavier Urbain. Automated certified proofs with CiME3. In *Proc. 22nd RTA*, volume 10 of *LIPIcs*, pages 21–30. Schloss Dagstuhl, 2011. doi:10.4230/LIPIcs.RTA.2011.21.
7. P. Courtieu, G. Gbedo, and O. Pons. Improved matrix interpretation. In *Proc. 36th SOFSEM*, volume 5901 of *LNCS*, pages 283–295. Springer, 2010. doi:10.1007/978-3-642-11266-9_24.
8. J. Endrullis, J. Waldmann, and H. Zantema. Matrix interpretations for proving termination of term rewriting. *J. Autom. Reasoning*, 40(2-3):195–220, 2008. doi:10.1007/s10817-007-9087-9.
9. J. Giesl, R. Thiemann, and P. Schneider-Kamp. The dependency pair framework: Combining techniques for automated termination proofs. In *Proc. 11th LPAR*, volume 3452 of *LNAI*, pages 301–331. Springer, 2005. doi:10.1007/978-3-540-32275-7_21.
10. F. Haftmann, A. Krauss, O. Kunčar, and T. Nipkow. Data refinement in Isabelle/HOL. In *Proc. 4th ITP*, volume 7998 of *LNCS*, pages 100–115. Springer, 2013. doi:10.1007/978-3-642-39634-2_10.
11. F. Haftmann and T. Nipkow. Code generation via higher-order rewrite systems. In *Proc. 10th FLOPS*, volume 6009 of *LNCS*, pages 103–117. Springer, 2010. doi:10.1007/978-3-642-12251-4_9.
12. N. Hirokawa and G. Moser. Automated complexity analysis based on the dependency pair method. In *Proc. 4th IJCAR*, volume 5195 of *LNCS*, pages 364–379. Springer, 2008. doi:10.1007/978-3-540-71070-7_32.
13. D. Hofbauer. *Termination Proofs and Derivation Lengths in Term Rewriting Systems*. Dissertation, Technische Universität Berlin, Germany, 1991. Available as Technical Report 92-46, TU Berlin, 1992.
14. D. Hofbauer and C. Lautemann. Termination proofs and the length of derivations. In *Proc. 3rd RTA*, volume 355 of *LNCS*, pages 167–177. Springer, 1989. doi:10.1007/3-540-51081-8_107.
15. B. Huffman and O. Kuncar. Lifting and transfer: A modular design for quotients in Isabelle/HOL. In *Proc. 3rd CPP*, volume 8307 of *LNCS*, pages 131–146. Springer, 2013. doi:10.1007/978-3-319-03545-1_9.
16. A. Koprowski and J. Waldmann. Arctic termination ... below zero. In *Proc. 24th RTA*, volume 5117 of *LNCS*, pages 202–216. Springer, 2008. doi:10.1007/978-3-540-70590-1_14.
17. M. Korp, C. Sternagel, H. Zankl, and A. Middeldorp. Tyrolean Termination Tool 2. In *Proc. 20th RTA*, volume 5595 of *LNCS*, pages 295–304. Springer, 2009. doi:10.1007/978-3-642-02348-4_21.
18. D. Lankford. On proving term rewriting systems are Noetherian. Technical Report MTP-3, Louisiana Technical University, Ruston, LA, USA, 1979.
19. A. Lochbihler. Light-weight containers for Isabelle: Efficient, extensible, nestable. In *Proc. 4th ITP*, volume 7998 of *LNCS*, pages 116–132. Springer, 2013. doi:10.1007/978-3-642-39634-2_11.
20. S. Lucas. On the relative power of polynomials with real, rational, and integer coefficients in proofs of termination of rewriting. *Appl. Algebr. Eng. Comm.*, 17(1):49–73, 2006. doi:10.1007/s00200-005-0189-5.

21. S. Lucas. Practical use of polynomials over the reals in proofs of termination. In *Proc. 9th PPDP*, pages 39–50. ACM, 2007. doi:10.1145/1273920.1273927.
22. G. Moser, A. Schnabl, and J. Waldmann. Complexity analysis of term rewriting based on matrix and context dependent interpretations. In *Proc. 28th FSTTCS*, volume 2 of *LIPIcs*, pages 304–315. Schloss Dagstuhl, 2008. doi:10.4230/LIPIcs.FSTTCS.2008.1762.
23. F. Neurauter and A. Middeldorp. On the domain and dimension hierarchy of matrix interpretations. In *Proc. 18th LPAR*, volume 7180 of *LNCS*, pages 320–334. Springer, 2012. doi:10.1007/978-3-642-28717-6_25.
24. F. Neurauter, H. Zankl, and A. Middeldorp. Monotonicity criteria for polynomial interpretations over the naturals. In *Proc. 5th IJCAR*, volume 6173 of *LNAI*, pages 502–517. Springer, 2010. doi:10.1007/978-3-642-14203-1_42.
25. T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL – A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002. doi:10.1007/3-540-45949-9.
26. B. Porter. Cauchy's mean theorem and the Cauchy-Schwarz inequality. *Archive of Formal Proofs*, March 2006. http://afp.sf.net/entries/Cauchy.shtml.
27. A. Schnabl. *Derivational Complexity Analysis Revisited*. PhD thesis, University of Innsbruck, Austria, 2011.
28. C. Sternagel and R. Thiemann. Certification extends termination techniques. In *Proc. 11th WST*, 2010. arXiv:1208.1594.
29. C. Sternagel and R. Thiemann. Executable matrix operations on matrices of arbitrary dimensions. *Archive of Formal Proofs*, June 2010. http://afp.sf.net/entries/Matrix.shtml.
30. C. Sternagel and R. Thiemann. Executable multivariate polynomials. *Archive of Formal Proofs*, August 2010. http://afp.sf.net/entries/Polynomials.shtml.
31. C. Sternagel and R. Thiemann. Signature extensions preserve termination. In *Proc. 19th CSL*, volume 6247 of *LNCS*, pages 514–528. Springer, 2010. doi:10.1007/978-3-642-15205-4_39.
32. R. Thiemann. Formalizing bounded increase. In *Proc. 4th ITP*, volume 7998 of *LNCS*, pages 245–260. Springer, 2013. doi:10.1007/978-3-642-39634-2_19.
33. R. Thiemann. Implementing field extensions of the form $\mathbb{Q}[\sqrt{b}]$. *Archive of Formal Proofs*, February 2014. http://afp.sf.net/entries/Real_Impl.shtml.
34. R. Thiemann and C. Sternagel. Certification of termination proofs using CeTA. In *Proc. 22nd TPHOLs*, volume 5674 of *LNCS*, pages 452–468. Springer, 2009. doi:10.1007/978-3-642-03359-9_31.
35. H. Zankl and M. Korp. Modular complexity analysis via relative complexity. In *Proc. 21st RTA*, volume 6 of *LIPIcs*, pages 385–400. Schloss Dagstuhl, 2010. doi:10.4230/LIPIcs.RTA.2010.385.
36. H. Zankl and A. Middeldorp. Satisfiability of non-linear (ir)rational arithmetic. In *Proc. 16th LPAR*, volume 6355 of *LNCS*, pages 481–500. Springer, 2010. doi:10.1007/978-3-642-17511-4_27.
37. H. Zankl, R. Thiemann, and A. Middeldorp. Satisfiability of non-linear arithmetic over algebraic numbers. In *Proc. SCSS*, volume 10-10 of *RISC-Linz Technical Report*, pages 19–24, 2010.