Certification of Proving Termination of Term Rewriting by Matrix Interpretations

Adam Koprowski Hans Zantema

August 2007

Abstract

We develop a Coq formalization of the matrix interpretation method, which is a recently developed, powerful approach to proving termination of term rewriting. Our formalization is a contribution to the CoLoR project and allows to automatically certify matrix interpretation proofs produced by tools for proving termination. Thanks to this development the combination of CoLoR and our tool, TPA, was the winner in 2007 in the new certified category of the annual Termination Competition.

1 Introduction

Termination is an important concept in term rewriting. Many methods for proving termination have been proposed over the years. Recently the emphasis in this area is on automation and a number of tools have been developed for that purpose. One of such tools is TPA [15] developed by the first author.

To evaluate termination tools and stimulate their improvement the annual Termination Competition [3] is organized, where such tools compete on a set of problems from the Termination Problems Database (TPDB), [4]. This competition has become a de-facto standard in evaluation of new termination techniques and developments of termination tools.

However, every year termination tools are becoming more and more complex and are changing rapidly as new techniques are being developed and old ones re-implemented. Therefore ensuring correctness of such tools is a challenging task. This was one of the motivations to start the CoLoR [6] project, initiated by Frédéric Blanqui in 2004. The goal of the project is to use the Coq [1] theorem prover to fully automatically verify results produced by tools for proving termination.

The main subject of this paper is our contribution to the CoLoR project, namely formalization of the matrix interpretation method [10]. This recent method turned out to be very powerful for proving termination and was incorporated into many modern termination provers.

This year in the termination competition the new certified category has been introduced, where tools must not only find a termination proof but also ensure its correctness by stating and proving it in an established theorem prover. Our contribution to CoLoR allowed the combined entry of TPA+CoLoR to win the 2007 edition of the competition in this newly introduced category.

Concerning related work in the first place we should mention the Coccinelle library which uses approach similar to the one employed by CoLoR and also uses Coq theorem prover. We will say more about it in Section 4, where we evaluate the results of CoLoR in the context of the termination competition.

The recent work of Alexander Krauss [16] is another effort toward certified termination. It is different in several aspects. Its main aim is to automatically generate certified termination proofs for recursive functions used in Isabelle/HOL theorem prover. However external termination provers are not involved and the only termination technique supported by this method is the size-change principle.

The rest of this paper is organized as follows. First in Section 2 we recapitulate the theory of matrix interpretations from [10]. Section 3 presents an overview of the Coq formalization of the theoretical results from the preceding section. It is followed by Section 4 where the method is evaluated in the context of the Termination Competition. We conclude in Section 5.

2 Theory of Matrix Interpretations

In this section we recall what we need from the theory of matrix interpretations as it is presented in [10, 11]. To keep the presentation self-contained we start by preliminaries on term rewriting.

2.1 Preliminaries

Let Σ be a signature, being a set of operation symbols each having a fixed arity in $\mathbb{N}$. For a set of variable symbols $\mathcal{V}$, let $\mathcal{T}(\Sigma, \mathcal{V})$ be the set of terms over Σ and $\mathcal{V}$, that is, the smallest set satisfying

- $x \in \mathcal{T}(\Sigma, \mathcal{V})$ for all $x \in \mathcal{V}$, and
- if the arity of $f \in \Sigma$ is n and $t_i \in \mathcal{T}(\Sigma, \mathcal{V})$ for $i = 1, \ldots, n$, then $f(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma, \mathcal{V})$.

A *term rewriting system* (TRS) $\mathcal{R}$ over $\Sigma, \mathcal{V}$ is a set of pairs $(\ell, r) \in \mathcal{T}(\Sigma, \mathcal{V}) \times \mathcal{T}(\Sigma, \mathcal{V})$, for which $\ell \notin \mathcal{V}$ and all variables in r occur in ℓ. Pairs (ℓ, r) are called *rewrite rules* and are usually written as $\ell \to r$.

For a substitution $\sigma : \mathcal{V} \to \mathcal{T}(\Sigma, \mathcal{V})$ and a term t the application of σ to t, denoted by $t\sigma$, is a term defined inductively as

- $x\sigma = \sigma(x)$ for all $x \in \mathcal{V}$, and

- $f(t_1, \ldots, t_n)\sigma = f(t_1\sigma, \ldots, t_n\sigma)$.

For a TRS $\mathcal{R}$ the *top rewrite relation* $\xrightarrow{\text{top}}_{\mathcal{R}}$ on $\mathcal{T}(\Sigma, \mathcal{V})$ is defined by $t \xrightarrow{\text{top}}_{\mathcal{R}} u$ if and only if there is a rewrite rule $\ell \to r \in \mathcal{R}$ and a substitution $\sigma : \mathcal{V} \to \mathcal{T}(\Sigma, \mathcal{V})$ such that $t = \ell\sigma$ and $u = r\sigma$. The *rewrite relation* $\to_{\mathcal{R}}$ is defined to be the smallest relation satisfying

- if $t \xrightarrow{\text{top}}_{\mathcal{R}} u$ then $t \to_{\mathcal{R}} u$, and
- if $t_i \to_{\mathcal{R}} u_i$ and $t_j = u_j$ for $j \neq i$, then $f(t_1, \ldots, t_n) \to_{\mathcal{R}} f(u_1, \ldots, u_n)$ for every $f \in \Sigma$ of arity n.

A binary relation $\to$ is called *terminating* or *strongly normalizing*, notation $\mathsf{SN}(\to)$, if it is well-founded, i.e., no infinite sequence $t_1, t_2, \ldots$ exists satisfying $t_i \to t_{i+1}$ for all $i \in \mathbb{N}$. A TRS $\mathcal{R}$ is called terminating if $\mathsf{SN}(\to_{\mathcal{R}})$ holds, shortly written as $\mathsf{SN}(\mathcal{R})$

Example 2.1. Consider the TRS consisting of the following single rule:

$$\mathrm{a}(\mathrm{a}(x)) \to \mathrm{a}(\mathrm{b}(\mathrm{a}(x)))$$

We will use this example to illustrate the method of matrix interpretations. It is worth noting that this TRS is not simply terminating and hence simplification orders are bound to fail for it.

A binary relation $\to_1$ is called *terminating relative to* a binary relation $\to_2$, written as $\mathsf{SN}(\to_1 \,/\, \to_2)$, if there is no infinite sequence $t_1, t_2, t_3, \ldots$ such that

- $t_i \to_1 t_{i+1}$ for infinitely many values of i, and
- $t_i \to_2 t_{i+1}$ for all other values of i.

We use the notation $\to_1 \,/\, \to_2$ to denote $\to_2^* \cdot \to_1$ ([1]); it is easy to see that $\mathsf{SN}(\to_1 \,/\, \to_2)$ coincides with well-foundedness of $\to_1 \,/\, \to_2$. Obviously for every binary relation $\to$ the property $\mathsf{SN}(\emptyset/\to)$ holds, and $\mathsf{SN}(\to /\emptyset)$ is equivalent to $\mathsf{SN}(\to)$. We write $\mathsf{SN}(\mathcal{R}/\mathcal{S})$ as a shorthand for $\mathsf{SN}(\to_{\mathcal{R}} \,/\, \to_{\mathcal{S}})$, and we write $\mathsf{SN}(\mathcal{R}_{\text{top}}/\mathcal{S})$ as a shorthand for $\mathsf{SN}(\xrightarrow{\text{top}}_{\mathcal{R}} \,/\, \to_{\mathcal{S}})$.

For a TRS $\mathcal{R}$ a symbol $f \in \Sigma$ is called a *defined symbol* if f is the root symbol of a left hand side of a rule from $\mathcal{R}$. For every defined symbol $f \in \Sigma$ we add to the signature a new marked symbol $f_\#$ with the same arity as f. If $f(s_1, \ldots, s_n) \to r$ is a rule in $\mathcal{R}$ and $g(t_1, \ldots, t_m)$ is a subterm of r for g being a defined symbol of $\mathcal{R}$, then the rewrite rule $f_\#(s_1, \ldots, s_n) \to g_\#(t_1, \ldots, t_m)$ is called a *dependency pair* of $\mathcal{R}$. The TRS consisting of all dependency pairs of $\mathcal{R}$ is denoted by $\mathsf{DP}(\mathcal{R})$.

[1] Other texts define $\to_1 \,/\, \to_2$ to denote $\to_2^* \cdot \to_1 \cdot \to_2^*$; since $\mathsf{SN}(\to_2^* \cdot \to_1)$ and $\mathsf{SN}(\to_2^* \cdot \to_1 \cdot \to_2^*)$ are easily seen to be equivalent, this does not cause an essential difference, while for our purpose $\to_2^* \cdot \to_1$ is more convenient

Example 2.2. Consider again the TRS from Example 2.1. The only defined symbol is " a" and there are two dependency pairs:

$$\mathrm{a}^\sharp(\mathrm{a}(x)) \to \mathrm{a}^\sharp(\mathrm{b}(\mathrm{a}(x))) \qquad \mathrm{a}^\sharp(\mathrm{a}(x)) \to \mathrm{a}^\sharp(x)$$

The main theorem about dependency pairs is the following, due to Arts and Giesl [5].

Theorem 2.3. *Let $\mathcal{R}$ be an arbitrary TRS. Then $\mathsf{SN}(\mathcal{R})$ if and only if $\mathsf{SN}(\mathsf{DP}(\mathcal{R})_{\mathrm{top}}/\mathcal{R})$.*

2.2 Monotone algebras

Here we summarize the monotone algebra theory as presented in [10]. There is one difference: in contrast to [10] we do not consider many-sortedness. It is not essential for certification as every proof in the many-sorted setting can be trivially translated to the one-sorted setting. The reason for this more complex setup in [10] is that it allows for an optimization in the search for termination proofs using matrix interpretations.

The monotone algebra approach works for all non-empty sets A; when using matrix interpretations the set A always consists of the set of vectors over $\mathbb{N}$ of a fixed dimension.

Definition 2.4. *An operation $[f] : A \times \cdots \times A \to A$ is* monotone *with respect to a binary relation $\to$ on A if for all $a_i, b_i \in A$ for $i = 1, \ldots, n$ with $a_i \to b_i$ for some i and $a_j = b_j$ for all $j \neq i$ we have $[f](a_1, \ldots, a_n) \to [f](b_1, \ldots, b_n)$.*

A weakly monotone Σ-algebra $(A, [\cdot], >, \gtrsim)$ *is a Σ-algebra $(A, [\cdot])$ equipped with two binary relations $>$, $\gtrsim$ on A such that*

- *$>$ is well-founded;*
- *$> \cdot \gtrsim \subseteq >$;*
- *for every $f \in \Sigma$ the operation $[f]$ is monotone with respect to $\gtrsim$.*

An extended monotone Σ-algebra $(A, [\cdot], >, \gtrsim)$ *is a weakly monotone Σ-algebra $(A, [\cdot], >, \gtrsim)$ in which moreover for every $f \in \Sigma$ the operation $[f]$ is monotone with respect to $>$.*

Up to presentation details the following theorem is the one-sorted version of the main theorem for the matrix interpretations from [10, Theorem 2].

Theorem 2.5. *Let $\mathcal{R}, \mathcal{R}', \mathcal{S}, \mathcal{S}'$ be TRSs over a signature Σ.*

1. *Let $(A, [\cdot], >, \gtrsim)$ be an extended monotone Σ-algebra such that $[\ell, \alpha] \gtrsim [r, \alpha]$ for every rule $\ell \to r$ in $\mathcal{R} \cup \mathcal{S}$ and $[\ell, \alpha] > [r, \alpha]$ for every rule $\ell \to r$ in $\mathcal{R}' \cup \mathcal{S}'$, for every $\alpha : \mathcal{V} \to A$.*

   Then $\mathsf{SN}(\to_{\mathcal{R}}/\to_{\mathcal{S}})$ implies $\mathsf{SN}(\to_{\mathcal{R}} \cup \to_{\mathcal{R}'} / \to_{\mathcal{S}} \cup \to_{\mathcal{S}'})$.

2. *Let* $(A, [\cdot], >, \gtrsim)$ *be a weakly monotone* Σ*-algebra such that* $[\ell, \alpha] \gtrsim [r, \alpha]$ *for every rule* $\ell \to r$ *in* $\mathcal{R} \cup \mathcal{S}$ *and* $[\ell, \alpha] > [r, \alpha]$ *for every rule* $\ell \to r$ *in* $\mathcal{R}'$*, for every* $\alpha : \mathcal{V} \to A$*.*

   Then $\mathsf{SN}(\to_{\mathcal{R}_{\mathrm{top}}} / \to_{\mathcal{S}})$ *implies* $\mathsf{SN}((\to_{\mathcal{R}} \cup \to_{\mathcal{R}'})_{\mathrm{top}} / \to_{\mathcal{S}})$*.*

2.3 Matrix interpretations

Now we present matrix interpretations with a fixed dimension d as an instance of monotone algebras. For the interpretation $[f]$ of a symbol $f \in \Sigma$ of arity n we choose a vector $\vec{f} \in \mathbb{N}^d$ and n matrices $F_1, F_2, \ldots, F_n$ over $\mathbb{N}$, each of size $d \times d$, such that the upper left elements $(F_i)_{1,1}$ are positive for all $i = 1, 2, \ldots, n$. Now we define

$$[f](\vec{v_1}, \ldots, \vec{v_n}) \;=\; F_1\vec{v_1} + \cdots + F_n\vec{v_n} + \vec{f} \tag{1}$$

for all $\vec{v_1}, \ldots, \vec{v_n} \in A$.

So we fix a monotone algebra with $A = \mathbb{N}^d$, interpretations $[\cdot]$ defined as above and we use the following orders on algebra elements:

$$\begin{aligned}(u_1, \ldots, u_d) \gtrsim (v_1, \ldots, v_d) &\iff \forall i : u_i \geq_{\mathbb{N}} v_i \\ (u_1, \ldots, u_d) > (v_1, \ldots, v_d) &\iff (u_1, \ldots, u_d) \gtrsim (v_1, \ldots, v_d) \wedge u_1 >_{\mathbb{N}} v_1\end{aligned}$$

One easily checks that $(A, [\cdot], >, \gtrsim)$ is an extended monotone Σ-algebra.

Let $x_1, \ldots, x_k$ be the variables occurring in ℓ, r. Then due to the linear shape of the functions $[f]$ we can compute matrices $L_1, \ldots, L_k, R_1, \ldots, R_k$ and vectors $\vec{l}, \vec{r}$ such that

$$\begin{aligned}[\ell, \alpha] &= L_1\vec{x_1} + \cdots + L_k\vec{x_k} + \vec{l} \\ [r, \alpha] &= R_1\vec{x_1} + \cdots + R_k\vec{x_k} + \vec{r}\end{aligned} \tag{2}$$

where $\alpha(x_i) = \vec{x_i}$ for $i = 1, \ldots, k$.

For matrices $B, C \in \mathbb{N}^{d \times d}$ write

$$B \succcurlyeq C \iff \forall i, j : (B)_{i,j} \geq (C)_{i,j}.$$

The following lemma provides a decision procedure for orders $>$ and $\gtrsim$ lifted to terms as used in Theorem 2.5.

Lemma 2.6. *Let* ℓ, r *be terms and let matrices* $L_1, \ldots, L_k, R_1, \ldots, R_k$ *and vectors* $\vec{l}, \vec{r}$ *be defined as above. Then:*

- $\forall \alpha : \mathcal{V} \to A, [\ell, \alpha] \gtrsim [r, \alpha] \iff \vec{l} \gtrsim \vec{r} \wedge \forall i : L_i \succcurlyeq R_i$*, and*
- $\forall \alpha : \mathcal{V} \to A, [\ell, \alpha] > [r, \alpha] \iff \vec{l} > \vec{r} \wedge \forall i : L_i \succcurlyeq R_i$*.*

Now the approach of applying Theorem 2.5, part 1, for proving $\mathsf{SN}(\mathcal{R}/\mathcal{S})$ is as follows (for proving $\mathsf{SN}(\mathcal{R})$ this coincides with choosing $\mathcal{S} = \emptyset$):

- Fix a dimension d.
- For every symbol $f \in \Sigma$ choose a vector $\vec{f} \in \mathbb{N}^d$ and matrices $F_i \in \mathbb{N}^{d\times d}$ for $i = 1, 2, \ldots, n$ for n being the arity of f, such that the upper left elements $(F_i)_{1,1}$ are positive for all $i = 1, 2, \ldots, n$.
- For every rule $\ell \to r \in \mathcal{R} \cup \mathcal{S}$ check that $L_i \succcurlyeq R_i$ for $i = 1, \ldots, k$ and $\vec{l} \gtrsim \vec{r}$ for the corresponding matrices L_i, R_i and vectors $\vec{l}$, $\vec{r}$ as defined above.
- Remove all rules from $\mathcal{R}$ and $\mathcal{S}$ moreover satisfying $l_1 > r_1$.
- If the remaining $\mathcal{R}$ is empty we are finished since $\mathsf{SN}(\emptyset/\mathcal{S})$ trivially holds, otherwise the process is repeated for the reduced TRSs $\mathcal{R}$, $\mathcal{S}$.

We illustrate this on an example:

Example 2.7. Consider again the TRS from Example 2.1. Now we choose dimension $d = 2$ and the following interpretation of symbols:

$$[\mathrm{a}(x)] = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$
$$[\mathrm{b}(x)] = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

We proceed by computing interpretation of the left and right hand side of the single rule.

$$[\mathrm{a}(\mathrm{a}(x))] = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \left(\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) + \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$
$$[\mathrm{a}(\mathrm{b}(\mathrm{a}(x)))] = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \left(\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \left(\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right) + \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Evaluating that expressions to linear form, as in Equation 2 yields:

$$[\mathrm{a}(\mathrm{a}(x))] = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$
$$[\mathrm{a}(\mathrm{b}(\mathrm{a}(x)))] = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

We observe that coefficients standing by x are equal and for the constant terms we have $\begin{bmatrix} 1 \\ 1 \end{bmatrix} > \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ as we have strict decrease in the first position and equality in the second. Hence by Lemma 2.6 we conclude that $\forall \alpha : \mathcal{V} \to A, [\mathrm{a}(\mathrm{a}(x)), \alpha] > [\mathrm{a}(\mathrm{b}(\mathrm{a}(x))), \alpha]$. Application of Theorem 2.5, part 1 allows us to remove this rule. As this is the only rule we have proven termination of this one rule TRS.

For proving termination of a TRS $\mathcal{R}$ by dependency pairs, according to Theorem 2.3, we have to prove $\mathsf{SN}(\mathsf{DP}(\mathcal{R})_{\mathrm{top}}/\mathcal{R})$. For this we apply Theorem 2.5, part 2. A similar scheme as sketched above is used, with the following difference:

- Since we only require to have a weakly monotone algebra, we may choose arbitrary matrices $F_i \in \mathbb{N}^{d\times d}$ without the restricton of positiveness of the upper left element $(F_i)_{1,1}$.
- For proving $\mathsf{SN}(\mathsf{DP}(\mathcal{R})_{\mathrm{top}}/\mathcal{R})$ only rules from $\mathsf{DP}(\mathcal{R})$ are removed until nothing remains, and all rules from $\mathcal{R}$ are kept.

We conclude this section by illustrating how a termination of a very challenging system can be easily proven with the matrix interpretation method.

Example 2.8. Consider the TRS (Zantema/z086.srs from the TPDB [4]) consisting of the following three rules:

$$\mathrm{a}(\mathrm{a}(x)) \to \mathrm{c}(\mathrm{b}(x)), \quad \mathrm{b}(\mathrm{b}(x)) \to \mathrm{c}(\mathrm{a}(x)), \quad \mathrm{c}(\mathrm{c}(x)) \to \mathrm{b}(\mathrm{a}(x)).$$

Until recently termination of this innocent looking system was an open problem [2, Problem 104]. Attempts to prove its termination gave birth to the matrix interpretation method.

We choose dimension $d = 4$ and the following interpretation:

$$[\mathrm{a}(x)] = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 1 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 2 \\ 0 \end{bmatrix}$$

$$[\mathrm{b}(x)] = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 2 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$[\mathrm{c}(x)] = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Computing interpretation for the first rule $\mathrm{a}(\mathrm{a}(x)) \to \mathrm{c}(\mathrm{b}(x))$ yields:

$$\begin{aligned} \mathrm{a}(\mathrm{a}(x)) &= \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 1 & 0 & 0 \end{bmatrix} \left(\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 1 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 2 \\ 0 \end{bmatrix} \right) + \begin{bmatrix} 0 \\ 0 \\ 2 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 2 \\ 0 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} \mathrm{c}(\mathrm{b}(x)) &= \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 0 \end{bmatrix} \left(\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 2 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \right) + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 2 \\ 0 \end{bmatrix} \end{aligned}$$

We conclude that $\forall \alpha : \mathcal{V} \to A, [\mathrm{a}(\mathrm{a}(x)), \alpha] \gtrsim [\mathrm{c}(\mathrm{b}(x)), \alpha]$. Repeating this procedure for the remaining two rules we observe that the third rule is also oriented weakly, whereas the second rule is oriented strictly. Application of Theorem 2.5, part 1 allows us to remove this rule and proceed with proving termination of the simplified system:

$$\mathrm{a}(\mathrm{a}(x)) \to \mathrm{c}(\mathrm{b}(x)), \quad \mathrm{c}(\mathrm{c}(x)) \to \mathrm{b}(\mathrm{a}(x)),$$

which is easy and can be established with any standard method.

3 Coq Formalization

Our formalization was developed within the CoLoR project, so we begin by a short introduction of CoLoR in Section 3.1. Then we continue with a description of the formalization of matrix interpretations, which consists of several parts. The formalization of monotone algebras, introduced in Section 2.2, is presented in Section 3.2. To deal with matrices we had to develop a Coq library of matrices; this is the subject of Section 3.3. Then in Section 3.4 we present the formalization of the matrix interpretations method, corresponding to the theory developed in Section 2.3. Finally, in Section 3.5, we shortly explain how the results concerning polynomial interpretations, already present in CoLoR, could be expressed in the setting of monotone algebras and how they were improved by doing so.

3.1 CoLoR: Certification of Termination

The CoLoR [6] project was founded by Frédéric Blanqui in March 2004, with the goal of certification of termination proofs found by termination provers in Coq. It is available at the following address:

`http://color.loria.fr`

It essentially consists of three parts:

- TPG (Termination Proofs Grammar): a formal grammar for the termination proofs.
- CoLoR (Coq Library on Rewriting and Termination): a library of results on termination of rewriting, formalized in Coq.
- Rainbow: a tool for transforming termination proofs in the TPG format into Coq scripts certifying termination by employing results from CoLoR.

The general approach to certifying termination with CoLoR is presented in Figure 1. For a given TRS $\mathcal{R}$ some termination prover is called. If it succeeds in proving termination, it outputs a termination proof in the TPG format. Such an encoding of a proof is given to Rainbow which translates it into a Coq script containing a formal proof of

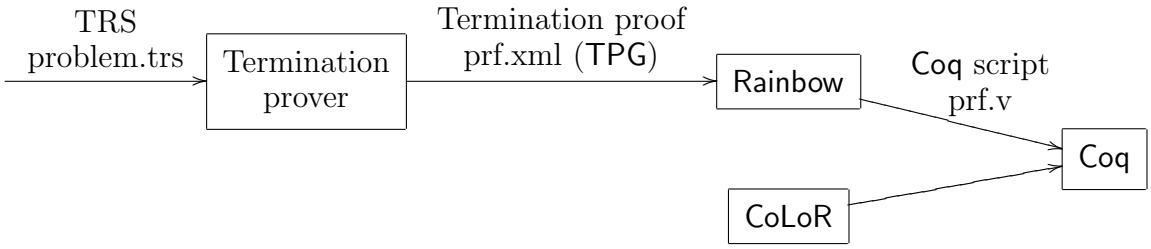

Figure 1: Certifying termination with CoLoR.

the claim that $\mathcal{R}$ is terminating by using results from the CoLoR library. Then Coq is executed on such a script to verify that the termination proof found by the termination tool is indeed correct.

Before presenting a glimpse of how the TPG format looks like we would like to begin with two general remarks concerning the format. Firstly, owing to the use of XML as an underlying markup language, the proof format is rather verbose. This does not seem to be a problem, however, as such proofs are both produced and consumed by programs and there is hardly ever any need for human to consult such file. The advantages of using XML are clear, as it is very popular, enjoys a lot of extensions (for instance allowing to easily transform XML documents) and is heavily supported by tools.

Second important observation is that the TRS under consideration is not part of the proof. Indeed at the moment the choice in CoLoR was to separate the problem and the corresponding proof in two different files. This is likely to change in future versions of CoLoR, though.[2]

We conclude this section with an example of how the proofs in the TPG format look like.

Example 3.1. Figure 2 presents a complete termination proof of the TRS introduced in Example 2.1, in the TPG format.[3]

We try to give a short overview of the structure of the proof. It starts on line `[01]`; the first step of the proof is application of the Manna-Ness criterion (`[02]`). This is the standard criterion stating that if a TRS is included in a reduction ordering (a well-founded ordering on terms closed under substitutions and contexts) then it is terminating. It is closely related to the approach of monotone algebras (monotone algebras giving rise to a particular class of reduction orderings). In this case the Manna-Ness criterion is instantiated to the matrix interpretation ordering (`[03-04]`), with dimension 2 (`[05]`). The interpretations of function symbols follow (`[06-39]`). For instance the constant term of the interpretation of " b" is given by vector $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$ (`[10-12]`). Then we have a number of matrix entries – one per every argument. The matrices are given in a row-by-row order, so lines `[14-19]` encode the matrix $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$. Similarly lines `[23-38]` encode $[\mathrm{a}(x)] = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. Finally lines `[42-44]` contain the

[2]Indeed in our opinion this was not a fortunate choice.

[3]Line numbers in bracket are included for presentation purposes only and are not part of the format.

```
[01] <proof>
[02]   <manna_ness>
[03]     <order>
[04]       <matrix_int>
[05]         <dimension>2</dimension>
[06]         <mi_map>
[07]           <mapping>
[08]             <fun>b</fun>
[09]             <mi_fun>
[10]               <const>
[11]                 <velem>0</velem> <velem>0</velem>
[12]               </const>
[13]               <arg>
[14]                 <row>
[15]                   <velem>1</velem> <velem>0</velem>
[16]                 </row>
[17]                 <row>
[18]                   <velem>0</velem> <velem>0</velem>
[19]                 </row>
[20]               </arg>
[21]             </mi_fun>
[22]           </mapping>
[23]           <mapping>
[24]             <fun>a</fun>
[25]             <mi_fun>
[26]               <const>
[27]                 <velem>0</velem> <velem>2</velem>
[28]               </const>
[29]               <arg>
[30]                 <row>
[31]                   <velem>1</velem> <velem>1</velem>
[32]                 </row>
[33]                 <row>
[34]                   <velem>0</velem> <velem>0</velem>
[35]                 </row>
[36]               </arg>
[37]             </mi_fun>
[38]           </mapping>
[39]         </mi_map>
[40]       </matrix_int>
[41]     </order>
[42]     <proof>
[43]       <trivial/>
[44]     </proof>
[45]   </manna_ness>
[46] </proof>
```

Figure 2: Termination proof for the $\mathrm{a}(\mathrm{a}(x)) \to \mathrm{a}(\mathrm{b}(\mathrm{a}(x)))$ TRS in the TPG format used by Rainbow.

termination proof for the simplified TRS resulting from the application of Theorem 2.5. In this particular case no rules remain (the only rule is oriented strictly and removed) so the remaining part of the proof is indeed `<trivial>`. It is worth noting, however, that the proof obligations (in the form of a TRS under consideration) at every point in the proof are implicit. They follow from the transformations and simplifications preceding in the proof so from this point of view they can be considered redundant. This is another point where CoLoR is likely to change in the future.

3.2 Monotone Algebras

While doing this formalization we faced a number of design choices. The essential question was whether to simply formalize matrix interpretations as they are or to try to make the development as general as possible, such that hopefully (parts of) it could be reused for other techniques and also extensions to the technique itself would be feasible. We opted for the latter. Hence we formalized monotone algebras in their full generality and only later instantiated them to matrix interpretations; as in the theory presented in Sections 2.2 and 2.3. This, later on, allowed us to easily express the technique of polynomial interpretations in the setting of monotone algebras, making it more powerful and more generally applicable. We will see more about that in Section 3.5.

To achieve such a generic formalization we found the module mechanism of Coq especially useful. It allows for mass abstraction by encapsulating a number of declarations and definitions in modules. Such modules can be parameterized by means of functors, that is functions from modules to modules. For instance we formalized monotone algebras in Coq as a functor, which takes as an argument the following structure describing a weakly monotone Σ-algebra instance:

```
Module Type MonotoneAlgebraType.
  Parameter Sig : Signature.
  Parameter I : interpretation Sig.
  Notation A := (domain I).
  Parameters (succ succeq : relation A).
```

So a monotone algebra structure consists of: a signature `Sig`, interpretation `I` for all function symbols from this signature and two relations over the domain `A` of the monotone algebra: `succ` and `succeq`. Moreover it contains the following requirements on those components:

```
  Parameter monotone_succeq : monotone I succeq.
  Parameter succ_wf : WF succ.
  Parameter succ_succeq_compat : absorb succ succeq.
```

So we demand that `succeq` is monotone (`monotone_succeq`), `succ` is well-founded (`succ_wf`) and we also require the compatibility condition between `succ` and `succeq`

(succ_succeq_compat); all the standard requirements of monotone algebras, as presented in Section 2.2. Note that typically succ and succeq will be orders, but this is not required.

There is however one more thing that we need in order to be able to deal with concrete examples. For an application of Theorem 2.5 we need to check for arbitrary terms ℓ and r whether $[\ell, \alpha] > [r, \alpha]$ for every $\alpha : \mathcal{V} \to A$ and similarly for $\gtrsim$. Our first approach was as follows:

```
  Parameter succ_dec : rel_dec IR_succ.
  Parameter succeq_dec : rel_dec IR_succeq.
```

IR_succ and IR_succeq are relations on terms obtained from succ and succeq by requiring the respective relation to hold for an arbitrary instantiation of variables. Now we require those lifted relations on terms to be decidable, that is we require a proof that for two arbitrary elements the relation between them either holds or not. Such decidability results proven in the constructive logic of Coq provide a decision procedure. By making proofs transparent and hence allowing to reduce associated proof terms, one effectively obtains an algorithm for checking whether two given terms can be oriented with the given relation.

This approach however has one limitation: we require a decidability proof, so indeed the relations in question must be decidable. This is the case for matrix interpretations due to the characterization of Lemma 2.6 but it is not so for instance for non-linear polynomial interpretations. Therefore to make our development more general the actual requirements are as follows:

```
  Parameters (succ' : relation term) (succeq' : relation term).
  Parameter (succ'_sub : succ' << IR_succ).
  Parameter (succeq'_sub : succeq' << IR_succeq).
  Parameter succ'_dec : rel_dec succ'.
  Parameter succeq'_dec : rel_dec succeq'.
End MonotoneAlgebraType.
```

So essentially we must provide two decidable relations succ' and succeq' that are refinements of succ and succeq (<< being the CoLoR notation for the inclusion of relations), respectively, and those relations are used in application of Theorem 2.5 to check whether a rule can be (weakly) oriented. The fact that they are subsets of succ and succeq ensures soundness of this approach. But there is no completeness requirement allowing to use some heuristics in cases where the intended relations are not decidable, such as in case of polynomial interpretations; see Section 3.5.

Given a monotone algebra instance, specified by means of a module described above (MonotoneAlgebraType), we build a module with results about such a monotone algebra and machinery for proving termination of concrete examples with its help as a functor:

```
Module MonotoneAlgebraResults (MA : MonotoneAlgebraType).
```

To give a feeling of how theorems from Section 2.2 are stated in the theorem prover we present the Coq equivalent of Theorem 2.5, part 1.

```
Lemma ma_relative_termination:
  let S_gt := partition part_succ S in
  let S_ge := partition part_succeq S in
  let R_gt := partition part_succ R in
  let R_ge := partition part_succeq R in
    monotone I succ ->
    snd R_ge = nil ->
    snd S_ge = nil ->
    WF (red_mod (snd S_gt) (snd R_gt)) ->
    WF (red_mod S R).
```

Let us try to explain the components of this statement. To begin with `partition P l` is a function that given a predicate `P` and a list `l`, splits this list into two parts and returns them as a pair `l1, l2`, such that `P` holds for every element of the list `l1` and does not hold for every element of `l2`.

Now `part_succ` and `part_succeq` are predicates for the `partition` function, corresponding to the relations `succ` and `succeq`. We demand `succ` to be monotone, `monotone I succ`. This is because in order to have one uniform module to deal with all types of problems (termination, relative termination, relative-top termination) we do not introduce extended weakly monotone algebras as a separate construct, but rather postulate this additional monotonicity property of extended algebras where needed. Now we require the second component of the pairs `R_ge` and `S_ge` to be empty, hence all the rules of `R` and `S` must be weakly oriented. Finally this theorem states that we can conclude `WF (red_mod S R)` if, on top of all the other requirements that we mentioned, we can prove `WF (red_mod (snd S_gt) (snd R_gt))` so of the relative problem consisting of the rules from `S` and `R` that could not be oriented strictly. Stating this problem in such "operational" style allows us to easily apply it for concrete instances of termination problems.

Our formalized proof of the theorem `ma_relative_termination` mentioned above (corresponding to Theorem 2.5) is constructive and hence slightly differs from the proof in [10]. It is based on the following lemma.

Lemma 3.2. *Let* $\to_{\mathcal{R}}, \to_{\mathcal{S}}, \to_{\mathcal{R}}', \to_{\mathcal{S}}'$ *be binary relations for which* $\to_{\mathcal{S}}^* \cdot \to_{\mathcal{R}}$ *and* $(\to_{\mathcal{R}} \cup \to_{\mathcal{S}})^* \cdot (\to_{\mathcal{R}}' \cup \to_{\mathcal{S}}')$ *are well-founded. Then* $(\to_{\mathcal{S}} \cup \to_{\mathcal{S}}')^* \cdot (\to_{\mathcal{R}} \cup \to_{\mathcal{R}}')$ *is well-founded.*

When thinking in terms of infinite sequences this lemma can easily be proven by truncating the initial part of such, supposedly, infinite sequences and observing that the remaining part must be finite. Working in the constructive setting of Coq brings a slightly different kind of reasoning, where the focus is on providing a relation that is decreasing along the sequence and performing induction with respect to it.

The `MonotoneAlgebraResults` module also contains tactics allowing to deal with proving termination for concrete examples. This means that for using a monotone algebra approach one only needs to provide a monotone algebra instance, a module of type `MonotoneAlgebraType`, and as a result one obtains all the results and a full machinery for proving termination. We will show in Section 3.4 how we did this for the monotone algebra corresponding to the matrix interpretation method. We will also shortly explain in Section 3.5 how we did the same for the polynomial interpretations approach, by using the existing CoLoR development of polynomial interpretations method and expressing it in the framework of monotone algebras.

3.3 Matrices

To begin with, the sole fact that we had to formalize matrices may be surprising — one would expect such a general notion to be readily available in a theorem prover. But it is not present in the Coq standard library. Moreover we could find only one Coq development where matrices were used: the contribution by Nicolas Magaud [17], where he proves ring properties of square matrices. We decided not to use this formalization for the reasons that we discuss at the end of this section.

To implement matrices we used a generic approach by allowing entries in the matrices to be arbitrary elements from some semi-ring structure. For that firstly we expressed semi-rings as a module type. Then we defined matrices as a functor taking as its argument such a semi-ring structure and as a result producing the structure of matrices of arbitrary size with entries from the semi-ring domain.

Internally we represent matrices as vectors of vectors. Vectors are defined in the standard library of Coq (Coq.Bool.BVector) as follows:

```
Variable A : Type.
Inductive vector : nat -> Type :=
| Vnil : vector 0
| Vcons : forall (a : A) (n : nat), vector n -> vector (S n).
```

So `vector A n` represents a vector of `n` elements of type `A`. Apart from this definition the Coq standard library provides only few basic properties and operations on this type. But on the other hand, building on that, the CoLoR project provides a rich set of results about vectors that were further extended in the course of this development. Some of these functions, which we will need later on in the presentation, are informally defined in Figure 3 and their corresponding Coq types are presented in Figure 4.

Ability to reuse those results was our main motivation to represent matrices in the following way:

```
Definition matrix (m n : nat) : matrix m n := vector (vector A n) m.
```

Then a number of operations on matrices was defined and some of its properties proven. The library is by no means complete and contains little more than the results needed for certification of matrix interpretations. The provided operations include:

$$
\begin{aligned}
\text{Vnth}\ [a_1; \ldots a_n]\ i &= a_i \\
\text{Vfold\_left}\ f\ [a_1; \ldots a_n]\ b &= f\, a_1\, (f \ldots (f\, a_n\, b) \ldots) \\
\text{Vmap}\ f\ [a_1; \ldots a_n] &= [f\, a_1; \ldots f\, a_n] \\
\text{Vmap2}\ f\ [a_1; \ldots a_n]\ [b_1; \ldots b_n] &= [f\, a_1\, b_1; \ldots f\, a_n\, b_n] \\
\text{Vforall2n}\ P\ [a_1; \ldots a_n]\ [b_1; \ldots b_n] &= P\, a_1\, b_1 \wedge \ldots \wedge P\, a_n\, b_n
\end{aligned}
$$

Figure 3: Informal definitions of some basic functions operating on vectors.

```
Vnth : forall (A : Set) (n : nat),
  vector A n -> forall i : nat, i < n -> A.
Vfold_left : forall (A B : Set) (f : B -> A -> B),
  B -> forall (n : nat), vector A n -> B.
Vmap : forall (A B : Set) (f : A -> B),
  forall n : nat, vector A n -> vector B n.
Vmap2 : forall (A B C : Set) (f : A -> B -> C),
  forall n : nat, vector A n -> vector B n -> vector C n.
Vforall2n : forall (A : Set) (P : A -> A -> Prop),
  forall n : nat, vector A n -> vector A n -> Prop.
```

Figure 4: Coq types of some basic functions operating on vectors.

matrix creation (given matrix size and a function providing values for all matrix entries), several accessor functions to retrieve matrix elements, columns and rows, conversions from vectors to 1-row and 1-column matrices and few standard matrix operations such as transposition, addition and multiplication. To show how reusing results about vectors substantially eased our task we present below the definition of multiplication.

First we need a few auxiliary functions on matrices. We begin with three accessor functions: `get_row`, `get_col` and `get_elem` to retrieve, respectively, the i'th row, the j'th column and element at position (i, j) of a given matrix. [4]

```
Definition get_row m n (M : matrix m n) i (ip : i < m) :=
  Vnth M ip.
Definition get_col m n (M : matrix m n) j (ip : j < n) :=
  Vmap (fun v => Vnth v ip) M.
Definition get_elem m n (M : matrix m n) i j (ip : i < m) (jp : j < n) :=
  Vnth (get_row M ip) jp.
```

Note that those functions are partial as indexes `i` and `j` must be within the boundaries of a matrix `M`. In Coq all functions are total and to deal with this we use additional

[4]Note that variables `m`, `n`, `i` and `j` below do not have type annotations as their types can be inferred by Coq and hence can be omitted. In this case all those variables range over natural numbers as a careful reader can easily check.

arguments for those functions, the so-called domain predicates, that ensure that the arguments are within the domain of the function.

Next we introduce the `mat_build` function, which constructs a $m \times n$ matrix from two natural numbers `m` and `n`, and a function `f` which, given a matrix position, returns the value of a matrix element to be placed at that position. Again, this function `f` is partial as it is defined only for coordinates i, j such that $0 \leq i < m$ and $0 \leq j < n$.[5] Defining function `mat_build` explicitly is not an easy task due to the presence of domain predicates and dependent types. Therefore we use Coq proving capabilities to prove existence of such a function using its specification.[6]

```
Definition mat_build_spec m n (gen : forall i j, i < m -> j < n -> A),
  { M : matrix m n | forall i j (ip : i < m) (jp : j < n),
    get_elem M ip jp = gen i j ip jp }.
Proof.
  ...
Defined.
```

and we extract the computational content from the above constructive proof to obtain the required function:

```
Definition mat_build m n gen : matrix m n :=
  proj1_sig (mat_build_spec gen).
```

Having all those auxiliary, general purpose functions on vectors and matrices defining matrix multiplication is fairly straightforward. First we introduce a dot product of two vectors as:

```
Definition dot_product (n : nat) (l r : vector A n) : vector A n :=
  Vfold_left Aplus A0 (Vmap2 Amult l r).
```

where `A0` is the zero element of the domain (the additive identity of the semi-ring) and `Aplus` is the addition. Then multiplication becomes:

```
Definition mat_mult m n p (L : matrix m n) (R : matrix n p) :=
  mat_build (fun i j ip jp => dot_product (get_row L ip) (get_col R jp)).
```

As can be seen from this example abstracting away natural operations on vectors and matrices and then using them for more complex constructs has big advantages. Not only the definitions became significantly simpler but also reasoning about them, as one can first prove properties about such auxiliary functions and then use them to reason about more complex constructs.

In fact this was the main reason against using the development by Nicolas Magaud, mentioned at the beginning of this section. It provides nice results by proving the ring

[5] We index matrix rows and columns starting from 0.

[6] Please note that we are using the Coq mechanism of implicit arguments to skip arguments that can be inferred by Coq due to type dependencies. So for the function `get_elem M i j ip jp` arguments `i` and `j` can be inferred from the domain predicates `ip` and `jp`. For the reader aware of that, it also improves readability as the definitions get shorter and do not contain redundant information.

properties for square matrices. But the fact that it is stand-alone and does not provide this kind of separation as mentioned above, made it difficult to use in our setting. For instance a function for matrix addition is realized there by a relatively complex Fixpoint construct (which is 16 lines long), whereas we can simply write

```
Definition vec_plus n (L R : vector A n) := Vmap2 Aplus L R.
Definition mat_plus m n (L R : matrix m n) := Vmap2 (@vec_plus n) L R.
```

and use all CoLoR properties of `Vmap2` to prove properties of matrix addition. Similarly other operations could be expressed easily and concisely by using operations and properties of vectors available in CoLoR.

3.4 Matrix Interpretations

Now we will explain how monotone algebras are instantiated for the matrix interpretation method, so we will develop the Coq counter-part of the theory described in Section 2.3. First we introduce a data type representing a matrix interpretation of a function symbol:

```
Variables (Sig : Signature) (f : symbol Sig) (dim : nat).
Record matrixInt (argCnt : nat) : Type := mkMatrixInt {
  const : vector nat dim;
  args : vector (matrix dim dim) argCnt
}.
```

So `matrixInt n` is a type of matrix interpretation for a function symbol of arity `n`, defined as a record with two fields: `const` being a constant vector of the interpretation of size `dim` and `args` representing coefficients for the arguments with a `dim`×`dim` matrix per argument. Comparing with equation 1, `const` represents the $\vec{f}$ vector and `args` the list of matrices $F_1, \cdots, F_n$.

Now we enclose all the parameters required for the application of Theorem 2.5 specialized to the monotone algebra for matrix interpretations, in a module type:

```
Module Type TMatrixInt.
  Parameter sig : Signature.
  Parameter dim : nat.
  Parameter dim_pos : dim > 0.
  Parameter trsInt : forall f : sig, matrixInt dim (arity f).
End TMatrixInt.
```

So we take a signature `sig`, dimension for matrices (`dim`; d in Section 2.3), a proof that dimension is positive (`dim_pos`) and interpretations for all function symbols of the signature, with respective arities (`trsInt`).

Given those parameters we construct the respective monotone algebra. We begin by constructing the evaluation function `mi_eval`, which corresponds to computation of Equation 1, given values of vectors $\vec{v_1}, \ldots, \vec{v_n}$.

```
Notation vec := (vector A dim).
Definition mi_eval n (mi : matrixInt dim n)
   (v : vector vec n) : vec :=
  add_vectors (Vmap2 mat_times_vec (args mi) v) [+] const mi.
```

Note that `add_vectors` takes a list of vectors of equal size as an argument and returns their sum as output; `mat_times_vec M v` computes the vector resulting from multiplication of a matrix `M` by a vector `v` (both of appropriate sizes); and finally `[+]` is a notation we introduce for addition of vectors.

Now we can begin monotone algebra construction.

```
Module MatrixIntAlgebra <: MonotoneAlgebraType.
  Definition Sig := sig.
  Definition I := @mkInterpretation sig vec (@zero_vector dim)
    (fun f => mi_eval (trsInt f)).
  Definition succeq := @vec_ge dim.
  Definition succ v1 v2 := v1 >=v v2 /\ vec_at0 v1 > vec_at0 v2.
  ...
End MatrixIntAlgebra.
```

where `mkInterpretation sig A A0 Aeval` is a CoLoR function to build a function interpretation type given an interpretation domain `A`, a zero element of the domain `A0` and the evaluation function `Aeval`. `vec_at0` is a function returning the 0 coordinate of a vector and `vec_ge` is a greater equal relation on vectors defined by demanding vector elements to be pointwise related by $\geq$ relation, defined simply as:

```
Definition vec_ge := Vforall2n ge.
```

What remains to be done is to prove that those definitions meets the monotone algebra requirements. We cannot present this part of the development in great detail here due to space considerations. The most difficult property was actually decidability of algebra relations $>$ and $\gtrsim$ lifted to terms. This corresponds to proving the 'if' parts of Theorem 2.6. Note that we did not prove the 'only-if' parts of that theorem, which state completeness of this characterization and which are not needed for the correctness of the approach. Proving the 'if' part required performing linearization of the computation of a matrix interpretation, such as in equation 2. Then we proved that evaluating this linearized expression leads to the same result as simply evaluating this expression without any simplifications beforehand. Performing those two steps in Coq requires some effort.

3.5 Polynomial Interpretations and Monotone Algebras

Polynomial interpretations were contributed to CoLoR by Sébastien Hinderer [13]. Combining his contribution with out development we could easily construct a monotone algebra instance corresponding to polynomial interpretations method. This has the following advantages:

- Before, it was not possible to prove termination step-wise. So in order to prove termination one had to find a polynomial interpretation such that all the rules could be oriented strictly and then one could conclude termination of the whole system. The approach of monotone algebras on the other hand allows to orient strictly only some rules and, provided that the remaining rules can be weakly oriented, those strict rules can be removed and one may continue with proving termination for this simpler system. This brings a big improvement in the proving power of the method and corresponds to the way it is used in automatic termination provers.

- The development of polynomial interpretations of Sébastien Hinderer supported only termination, $\mathsf{SN}(\mathcal{R})$. The setting of monotone algebras supports also relative termination, $\mathsf{SN}(\mathcal{R}/\mathcal{S})$, and relative-top termination, $\mathsf{SN}(\mathcal{R}_{\mathrm{top}}/\mathcal{S})$. So by expressing polynomial interpretations as an instance of the monotone algebra approach we obtained the support for treating those more general problems with polynomial interpretations absolutely for free.

Instantiating the monotone algebra results to polynomial interpretations was straightforward. We achieved it in mere 117 sparse lines of code and with very minor modifications to the development of Sébastien Hinderer (essentially to get the relation for orienting rules weakly).

4 Evaluation

We already mentioned the termination competition [3, 7], the battlefield for termination provers, in Section 3.1. This year, for the first time, a new category of certified termination has been introduced, showing the recognition for the importance of certification efforts. Indeed ensuring reliability of constantly evolving and more and more complex tools is difficult and every year we observe some disqualifications due to erroneous proofs produced by some of the tools.

In this new category every claim made by a termination prover must be backed up by a full formal proof expressed and checked by some well established theorem prover (and not only by a textual informal description of such a proof, as is the case in the standard category). This makes the results reliable with the highest standards of reliability available in verification.

The combination of the CoLoR project (with Rainbow) and the termination prover TPA [15], developed by the first author, was the winning entry in this newly introduced category of the Termination Competition in 2007. It achieved the score of 354, meaning that for 354 out of the total 975 TRSs used in the competition, TPA could find a termination proof and using CoLoR correctness of this proof could be verified by Coq.

Due to the fact that this category was introduced only this year there were only two other participants. The termination prover C*i*ME [9] using the Coccinelle [8] library to certify termination results, again using Coq theorem prover. It got the second place

with a score of 317. The third participating tool was the entry of TTT[14] using CoLoR as the certifying back-end with a score of 289.

For comparison we would like to mention that in the standard category, which is run on the same set of problems, the scores ranged from 330 to 723. This shows that many proofs are beyond reach of the certification at the moment, which is completely understandable. But it also shows that for a substantial part of proofs we can not only produce them with termination tools but also fully automatically ensure their correctness, including difficult problems for which establishing termination results by human is very hard. We believe this is a big step forward and a very promising future for the termination results.

Considering evaluation of our contribution, every single termination proof produced by TPA in the competition was using matrix interpretations at some point. This is not so surprising given the fact that CoLoR, at the moment, is supporting only two basic orders: polynomial and matrix interpretations. But this also shows that for winning the competition, our contribution was crucial.

When it comes to performance finding a proof took TPA on average 2.0 sec and certification required 2.6 sec per system. There were however a few systems were the certification time was substantially longer. During the competition verification for 4 problems reached the 5 minutes timeout. We tried to certify termination proofs for those systems on our own, without a time limit. Three of those systems required 4, 11 and 27 minutes to complete on an Intel® Xeon™ 2.8GHz PC. For the forth problem, TRCSR/PALINDROME_complete_noand_FR.trs, Coq stopped with "out of memory" error after 86 minutes of computation. It is worth noting that for TPDB standards this is a large system with 58 function symbols and 76 rules. The proof of its termination required 57 successive applications of Theorem 2.5 and the generated file with Coq script was 8240 lines long and 408Kb in size, which is one third of the whole CoLoR library. Currently we are busy experimenting and trying to improve the performance of the verification routines but, although we did achieve some speedups, so far they were of rather minor effect.

5 Conclusions

We presented our contribution to the CoLoR project — a Coq formalization of matrix interpretations method for proving termination of rewriting. This allows us to fully automatically certify termination of non-trivial rewrite systems, such as the Zantema/z086.srs from the TPDB [4]:

$$\mathrm{a}(\mathrm{a}(x)) \to \mathrm{c}(\mathrm{b}(x)), \quad \mathrm{b}(\mathrm{b}(x)) \to \mathrm{c}(\mathrm{a}(x)), \quad \mathrm{c}(\mathrm{c}(x)) \to \mathrm{b}(\mathrm{a}(x))$$

Until recently termination of this innocent looking system was an open problem [2, Problem 104] and now not only it can be automatically proven terminating by termination tools but also that results can be warranted by Coq.

It is worth noting that typically Coq is used as a proof assistant, where the formalization is built by a human interacting with the system. It is not so in our application as the Coq script formalizing termination of a given system is generated fully automatically by Rainbow from a proof description produced by some termination prover; again, automatically. However the proof assistance capabilities of Coq are crucial for the development of CoLoR.

The natural way of continuing work on certification of termination is to formalize further termination techniques. Although matrix interpretations provide a very powerful base ordering, they do not subsume other orders. Consider for instance the following system:

$$\mathrm{f}(\mathrm{s}(x), \mathrm{a}) \to \mathrm{f}(x, \mathrm{f}(\mathrm{s}(x), \mathrm{b}))$$

which can be easily proven terminating with lexicographic path order. By a simple argument one can show that matrix interpretations are not applicable here. Even more advantageous would be formalization of the dependency pair framework [12], a modular, powerful approach to proving termination, employed by most, if not all, successful modern termination provers. This is on-going work.

Acknowledgements

We would like to thank Frédéric Blanqui for helpful comments and encouragement for this work.

References

[1] The Coq proof assistant. `http://coq.inria.fr`.

[2] The RTA list of open problems. `http://www.lsv.ens-cachan.fr/rtaloop`.

[3] Termination competition.
`http://www.lri.fr/~marche/termination-competition`.

[4] Termination problems data base. `http://www.lri.fr/~marche/tpdb`.

[5] T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236(1-2):133–178, 2000.

[6] F. Blanqui, W. Delobel, S. Coupet-Grimal, S. Hinderer, and A. Koprowski. CoLoR, a Coq library on rewriting and termination. In *Eighth International Workshop on Termination (WST '06)*, 2006.

[7] H. Zantema C. March. The Termination Competition 2007. In Franz Baader, editor, *Proceedings of the 18th Conference on Rewriting Techniques and Applications (RTA '07)*, volume 4533 of *Lecture Notes in Computer Science*, pages 303–313. Springer, 2007.

[8] É. Contejean, P. Courtieu, J. Forest, O. Pons, and X. Urbain. Certification of automated termination proofs. In *Proceedings of the 6th International Symposium on Frontiers of Combining Systems (FroCoS '05)*, September 2007. To appear.

[9] E. Contejean, C. Marché, B. Monate, and X. Urbain. The CiME rewrite tool. Available at `http://cime.lri.fr`.

[10] J. Endrullis, J. Waldmann, and H. Zantema. Matrix interpretations for proving termination of term rewriting. In *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR '06)*, volume 4130 of *Lecture Notes in Computer Science*, pages 574–588. Springer, 2006.

[11] J. Endrullis, J. Waldmann, and H. Zantema. Matrix interpretations for proving termination of term rewriting. *Journal of Automated Reasoning*, 2007. Accepted.

[12] J. Giesl, R. Thiemann, and P. Schneider-Kamp. The dependency pair framework: Combining techniques for automated termination proofs. In *Proceedings of the 11th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR '04)*, volume 3452 of *Lecture Notes in Computer Science*, pages 301–331. Springer, 2004.

[13] S. Hinderer. Certification of termination proofs based on polynomial interpretations, 2004.

[14] N. Hirokawa and A. Middeldorp. Tyrolean termination tool: Techniques and features. *Information and Computation*, 205(4):474–511, 2007.

[15] A. Koprowski. TPA: Termination proved automatically. In Frank Pfenning, editor, *Proceedings of the 17th Conference on Rewriting Techniques and Applications (RTA '06)*, volume 4098 of *Lecture Notes in Computer Science*, pages 257–266. Springer, 2006.

[16] A. Krauss. Certified size-change termination. In Frank Pfenning, editor, *Automated Deduction - CADE-21, 21st International Conference on Automated Deduction, Bremen Germany, July 2007, Proceedings*, volume 4603 of *Lecture Notes in Artificial Intelligence*, pages 460–475. Springer, 2007.

[17] N. Magaud. Ring properties for square matrices. Coq contributions, available at `http://coq.inria.fr/contribs-eng.html`.