Certification of Proving Termination of Term Rewriting by Matrix Interpretations *

Adam Koprowski and Hans Zantema

Eindhoven University of Technology
Department of Computer Science
P.O. Box 513, 5600 MB, Eindhoven, The Netherlands
{A.Koprowski,H.Zantema}@tue.nl

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 [13] 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 [9]. This recent method turned out to be very powerful for proving termination and was incorporated into many modern termination provers. Due to space restrictions we present this

* Some preliminary results of this paper were first announced in [14]

development for termination only, but it is also applicable to relative termination problems and in the setting of dependency pairs [5]. For a more detailed description we refer to [15].

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 [9]. 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

2.1 Preliminaries

Let Σ be a signature. For a set of variable symbols $\mathcal{V}$, let $\mathcal{T}(\Sigma, \mathcal{V})$ be the set of terms over Σ and $\mathcal{V}$. We denote application of a substitution $\sigma : \mathcal{V} \to \mathcal{T}(\Sigma, \mathcal{V})$ to a term t by $t\sigma$.

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 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 such that $\xrightarrow{\text{top}}_{\mathcal{R}} \subseteq \to_{\mathcal{R}}$ 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. A TRS $\mathcal{R}$ is called terminating if $\mathsf{SN}(\to_{\mathcal{R}})$ holds, shortly written as $\mathsf{SN}(\mathcal{R})$

2.2 Monotone algebras

Here we summarize the monotone algebra theory as presented in [9, 10]. There is one difference: in contrast to [9] 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 [9] 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 1. *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* $[f](a_1, \ldots, a_n) \to [f](b_1, \ldots, b_n)$.

An extended 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,* $> \cdot \gtrsim \subseteq >$ *and 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 [9, Theorem 2].

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

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

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 $\boldsymbol{f} \in \mathbb{N}^d$ 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

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

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

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:

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

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 $\boldsymbol{l}, \boldsymbol{r}$ such that

$$[\ell, \alpha] = L_1\boldsymbol{x_1} + \cdots + L_k\boldsymbol{x_k} + \boldsymbol{l}, \qquad [r, \alpha] = R_1\boldsymbol{x_1} + \cdots + R_k\boldsymbol{x_k} + \boldsymbol{r} \tag{2}$$

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

For matrices $B, C \in \mathbb{N}^{d \times d}$ write $B \succcurlyeq C$ as a shorthand for $\forall i, j : (B)_{i,j} \geq (C)_{i,j}$. The following lemma provides a decision procedure for orders $>$ and $\gtrsim$ lifted to terms as used in Theorem 2.

Lemma 3. *Let ℓ, r be terms and let matrices $L_1, \ldots, L_k, R_1, \ldots, R_k$ and vectors $\boldsymbol{l}, \boldsymbol{r}$ be defined as above. Then $\forall \alpha : \mathcal{V} \to A, [\ell, \alpha] \gtrsim [r, \alpha]$ (resp. $[\ell, \alpha] > [r, \alpha]$) iff $\forall i : L_i \succcurlyeq R_i$ and $\boldsymbol{l} \gtrsim \boldsymbol{r}$ (resp. $\boldsymbol{l} > \boldsymbol{r}$)*

Now the approach of applying Theorem 2, for proving $\mathsf{SN}(\mathcal{R})$ is as follows:

- Fix a dimension d.
- For every symbol $f \in \Sigma$ choose a vector $\boldsymbol{f} \in \mathbb{N}^d$ 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 $\ell \to r \in \mathcal{R}$ check that $L_i \succcurlyeq R_i$ for $i = 1, \ldots, k$ and $\boldsymbol{l} \gtrsim \boldsymbol{r}$ for the corresponding matrices L_i, R_i and vectors $\boldsymbol{l}$, $\boldsymbol{r}$ as defined above.
- Remove all rules from $\mathcal{R}$ moreover satisfying $l_1 > r_1$.
- If the remaining $\mathcal{R}$ is empty we are finished, otherwise the process is repeated for the reduced TRS $\mathcal{R}$.

Example 4. We illustrate the above procedure on an example. Consider the TRS consisting of the following single rule: $\mathrm{a}(\mathrm{a}(x)) \to \mathrm{a}(\mathrm{b}(\mathrm{a}(x)))$. It is worth noting that this TRS is not simply terminating and hence simplification orders are bound to fail for it. For the approach of matrix interpretations we choose dimension $d = 2$ and the following interpretation of symbols:

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

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:

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

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 3 we conclude that $[\mathrm{a}(\mathrm{a}(x)), \alpha] > [\mathrm{a}(\mathrm{b}(\mathrm{a}(x))), \alpha]$ for all $\alpha : \mathcal{V} \to A$. Application of Theorem 2 allows us to remove this rule. As this is the only rule we have proven termination of this one rule TRS.

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.

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 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.

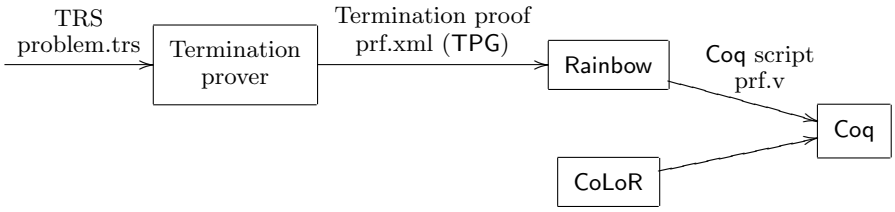

Fig. 1. Certifying termination with CoLoR.

For a more detailed description of the TPG format we refer to [15].

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. For more details we again refer to [15].

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 a structure describing a weakly monotone Σ-algebra instance, as introduced in Section 2.2.

For the formalization 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 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 to require the relations $>$ and $\gtrsim$ lifted to terms to be decidable, that is to 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 3 but it is not so for instance for non-linear polynomial interpretations. Therefore to make our development more general we actually require two decidable relations $\rhd$ and $\trianglerighteq$ such that $\rhd \subseteq >$ and $\trianglerighteq \subseteq \gtrsim$ and those relations are used in application of Theorem 2 to check whether a rule can be (weakly) oriented. The fact that they are subsets of $>$ and $\gtrsim$ 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 [15] for more details.

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.

```
Lemma ma_termination:
 let R_gt := partition part_succ R in
 let R_ge := partition part_succeq R in
   monotone I succ -> snd R_ge = nil -> WF (red (snd R_gt)) -> WF (red 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` ($\gtrsim$). We demand `succ` to be monotone, `monotone I succ`. Now we require the second component of the pair `R_ge` to be empty, hence all the rules of `R` must be weakly oriented. Finally this theorem states that we can conclude `WF (red R)` if, on top of all the other requirements that we mentioned, we can prove `WF (red (snd R_gt))` so of the TRS consisting of the rules from `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.

The monotone algebra module also contains Coq 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 and as a result one obtains all the results and a full machinery for proving termination. We will sketch in Section 3.4 how we instantiated monotone algebras to the matrix interpretation method. We also did that for polynomial interpretations; the interested reader is referred to [15].

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) with the type `vector A n` representing 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. Here we informally define some of these functions, which we will need later on in the presentation:

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

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: 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. [1]

```
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 arguments for those functions, the so-called domain predicates, which 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 $0 \leq j < n$.[2] 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.[3]

```
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 `mat_build` function.

---

[1] 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.

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

[3] 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`.

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 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 $\boldsymbol{f}$ 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 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 cannot present it here in details due to space limitations. The most difficult property was actually decidability of algebra relations $>$ and $\gtrsim$ lifted to terms. This corresponds to proving the 'if' part of Theorem 3. Note that we did not prove the 'only-if' part of that theorem, which state completeness of this characterization and which is 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 required a substantial effort.

4 Evaluation

We already mentioned the termination competition [3, 18], 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 [13], 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 [8] using the Coccinelle [7] 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 T$_{\text{T}}$T[12] 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 few systems were the certification time was substantially longer. During the competition verification for 4 problems reached the 5 minutes timeout. 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. Even more advantageous would be formalization of more involved refinements of the dependency pair framework [11]; a modular, powerful approach to proving termination, employed by most, if not all, successful modern termination provers. By supporting relative top termination problems our development is fully ready to benefit

from such refinements (see [15] for details), but the CoLoR library at the moment supports only the basic computation of dependency pairs. Implementing extensions such as usable rules or dependency graph approximations 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 *8th WST*, 2006.
7. É. Contejean, P. Courtieu, J. Forest, O. Pons, and X. Urbain. Certification of automated termination proofs. In *Proc. 6th FroCoS*, September 2007. To appear.
8. E. Contejean, C. Marché, B. Monate, and X. Urbain. The CiME rewrite tool. Available at `http://cime.lri.fr`.
9. J. Endrullis, J. Waldmann, and H. Zantema. Matrix interpretations for proving termination of term rewriting. In *Proc. 3rd IJCAR*, volume 4130 of *LNCS*, pages 574–588, 2006.
10. J. Endrullis, J. Waldmann, and H. Zantema. Matrix interpretations for proving termination of term rewriting. *Journal of Automated Reasoning*, 2007. Accepted.
11. 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 *LNCS*, pages 301–331, 2004.
12. N. Hirokawa and A. Middeldorp. Tyrolean termination tool: Techniques and features. *Information and Computation*, 205(4):474–511, 2007.
13. A. Koprowski. TPA: Termination proved automatically. In *Proc. 17th RTA*, volume 4098 of *LNCS*, pages 257–266, 2006.
14. A. Koprowski and H. Zantema. Certification of matrix interpretations in Coq. In *9th WST*, 2007.
15. A. Koprowski and H. Zantema. Certification of proving termination of term rewriting by matrix interpretations. Technical Report CS-Report 07/22, Eindhoven University of Technology, August 2007. Available at `http://www.win.tue.nl/~akoprows/papers/mint-cert-TR.pdf`.
16. A. Krauss. Certified size-change termination. In *Proc. 21st CADE*, volume 4603 of *LNAI*, pages 460–475, 2007.
17. N. Magaud. Ring properties for square matrices. Coq contributions, available at `http://coq.inria.fr/contribs-eng.html`.
18. C. Marché and H. Zantema. The Termination Competition 2007. In *Proc. 18th RTA*, volume 4533 of *LNCS*, pages 303–313, 2007.