International Conference on Rewriting Techniques and Applications 2010 (Edinburgh), pp. 357-372
http://rewriting.loria.fr/rta/

POLYNOMIALLY BOUNDED MATRIX INTERPRETATIONS

JOHANNES WALDMANN

Hochschule für Technik, Wirtschaft und Kultur (FH) Leipzig, Fakultät IMN, PF 30 11 66, D-04251 Leipzig, Germany.
E-mail address: waldmann@imn.htwk-leipzig.de

Abstract. Matrix interpretations can be used to bound the derivational complexity of rewrite systems. We present a criterion that completely characterizes matrix interpretations that are polynomially bounded. It includes the method of upper triangular interpretations as a special case, and we prove that the inclusion is strict. The criterion can be expressed as a finite domain constraint system. It translates to a Boolean constraint system with a size that is polynomial in the dimension of the interpretation. We report on performance of an implementation.

1. Introduction

Algorithms with polynomial complexity are widely accepted as practical. Since rewriting is a model of computation, we are interested in polynomial derivational complexity of rewriting. The derivational complexity of a (terminating) rewrite system is the length of a longest derivation in the system, measured as a function of the size of its initial term.

For a given terminating rewrite system, one can estimate its derivational complexity by looking at the proof method that established termination. We investigate the method of matrix interpretations [Hof06, End08]. If a rewrite system admits a matrix interpretation that is strictly compatible with all rules, then its derivational complexity is at most exponential. By restricting the shape of the matrices, we can lower this bound: if the matrices are upper triangular, derivational complexity is polynomial [Mos08].

Matrix interpretations are in fact weighted finite tree automata [Wal09], and an upper bound on the derivational complexity of the rewrite system is obtained from a bound for the growth of of the weight function computed by the automaton. So it is natural to use automata-theoretic results for a more detailed analysis. We can apply methods for the determination of (non-)ambiguity of classical (non-weighted) automata. The connection is immediate since a weighted automaton (over the standard semi-ring of natural numbers with standard addition and multiplication) can be seen as a path-counting device for an underlying classical automaton.

1998 ACM Subject Classification: F.4.2 [Grammars and Other Rewriting Systems] Term Rewriting, F.1.1 [Models of Computation] Weighted Automata, F.1.3 [Complexity Measures and Classes] Machine-independent Complexity, D.3.3 [Language Constructs and Features] Constraints.

Key words and phrases: derivational complexity of rewriting, matrix interpretation, weighted automata, ambiguity of automata, finite domain constraints.

Proceedings of the 21st International Conference on Rewriting Techniques and Applications, Edinburgh, July, 2010
Editors: Christopher Lynch
LIPIcs - Leibniz International Proceedings in Informatics. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany
Digital Object Identifier: 10.4230/LIPIcs.RTA.2010.357

The core of the paper is organized as follows. We show in Section 3 that it is enough to consider weighted word automata even if the object is term rewriting. In Section 4 we reduce the question of growth of a weighted automaton to the question of ambiguity of non-weighted automata. Then in Section 5 we give an algorithm that decides polynomial growth of an automaton. In Section 6 we show a rewriting system that has a polynomially bounded matrix interpretation, but no triangular matrix interpretation. In Section 7 we discuss the degree of the polynomial growth bounds. We then explain in Section 8 how the decision algorithms can be realized by finite domain constraint systems, and how these can be transformed to constraints in propositional logic. In Section 9 we report on the performance of an implementation of our method.

2. Notation and Preliminaries

We recall some notions and notations.

Terms and Rewriting [Baa98]. For a ranked signature $\Sigma = \Sigma_0 \cup \ldots \cup \Sigma_k$, we denote by $\mathsf{Term}(\Sigma, V)$ the set of terms over Σ with variables from a set V, and $\mathsf{Term}(\Sigma) := \mathsf{Term}(\Sigma, \emptyset)$. The size of a ground term is the total number of symbol occurrences: if $f \in \Sigma_k$, then $|f(t_1, \ldots, t_k)| = 1 + |t_1| + \ldots + |t_k|$.

We use paths to address subterms. A path is a sequence of steps, and a step is a pair of a function symbol and a number. The number indicates in which subtree the path continues. Formally, from the given ranked signature Σ, we construct a path signature $\Sigma' \subseteq \Sigma \times \mathbb{N}$ of unary symbols: for each $f \in \Sigma$ of arity k, we have symbols $(f,1), \ldots, (f,k)$ in Σ'. We often abbreviate (f,i) by f_i. For $t \in \mathsf{Term}(\Sigma, V)$, the set of all paths from the root of t to any node is

$$\mathsf{Path}(f(t_1, \ldots, t_k)) = \{\epsilon\} \cup \bigcup \{(f,i) \cdot p \mid 1 \leq i \leq k, p \in \mathsf{Path}(t_i)\}.$$

E.g., $\mathsf{Path}(f(a, g(b))) = \{\epsilon, (f,1), (f,2), (f,2)(g,1)\}$. Note that $|t| = |\mathsf{Path}(t)|$, the size of t is the number of paths in t. We write t_p for the function symbol that is reached by following the path $p \in \mathsf{Path}(t)$:

$$(f(\ldots))_\epsilon = f, \ (f(t_1, \ldots, t_k))_{(f,i)\cdot w} = (t_i)_w.$$

A rewrite system R is a set of pairs of terms with variables, and it defines a rewrite relation $\to_R$ in the usual way.

The *derivational complexity* [Hof89] of a terminating rewrite system R is the function

$$\mathsf{dc}_R : \mathbb{N}_+ \to \mathbb{N} : n \mapsto \max\{k \mid \exists t_1, t_2 \in \mathsf{Term}(\Sigma) : |t_1| \leq n \wedge t_1 \to_R^k t_2\}.$$

Matrix Interpretations [End08]. Let $\mathbb{N}^d$ denote the set of d-dimensional vectors with entries in $\mathbb{N}$. We picture these as column vectors. We use these orders on $\mathbb{N}^d$:

$$x \geq y \iff x_1 \geq y_1 \wedge \ldots \wedge x_d \geq y_d, \quad x > y \iff x \geq y \wedge x_1 > y_1.$$

We use k-ary linear functions $F : (\mathbb{N}^d)^k \to \mathbb{N}^d$ that are given by k square matrices $M_1, \ldots, M_k$ and a vector v via

$$F : (x_1, \ldots, x_k) \mapsto M_1 x_1 + \ldots + M_k x_k + v.$$

We call v the *absolute part* of F, and write $v = \mathsf{abs}(F)$. A linear function is *monotone* (with respect to $>$, in each argument separately) iff for each i, the top left entry of M_i is ≥ 1.

We define orderings on these functions. For F given by $(M_1, \ldots, M_k, v)$ and F' given by $(M'_1, \ldots, M'_k, v')$, we write

$$\begin{aligned} F \geq F' &\iff v \geq v' \wedge \forall 1 \leq i \leq k : M_i \geq M'_i \\ F > F' &\iff v > v' \wedge F \geq F' \end{aligned}$$

For any tuple of argument vectors $\vec{x} = (x_1, \ldots, x_k)$, we have $F \geq F' \Rightarrow F(\vec{x}) \geq F'(\vec{x})$ and $F > F' \Rightarrow F(\vec{x}) > F'(\vec{x})$.

A matrix interpretation assigns to each k-ary function symbol $f \in \Sigma_k$ a k-ary linear function $[f] : (\mathbb{N}^d)^k \to \mathbb{N}^d$. Since linear functions of this shape are closed with respect to composition (substitution), an interpretation can be extended from function symbols to terms (with variables).

We say an interpretation $[\cdot]$ is *compatible* with a rewrite rule $l \to r$ iff $[l] > [r]$.

Example 2.1. Take $\Sigma = \Sigma_1 = \{a, b\}$, and the monotone interpretation

$$[a] : x \mapsto \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} x, \quad [b] : x \mapsto \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} x + \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

that is compatible with $R = \{ab \to ba\}$, since

$$[ab] : x \mapsto \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} x + \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad > \quad [ba] : x \mapsto \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} x + \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \qquad \blacksquare$$

If a monotone interpretation is compatible with each rule of a rewrite system R, then $t_1 \to t_2$ implies $[t_1] > [t_2]$ and since $>$ is well-founded on $\mathbb{N}^d$, the system R is terminating. More specifically, the length of each rewrite sequence starting in some $t \in \mathsf{Term}(\Sigma)$ is bounded by the first (top) component of $[t]$. This follows from the definition of $>$ on $\mathbb{N}^d$. We define the growth of the matrix interpretation $[\cdot]$ by $\mathsf{growth}_{[]} : n \mapsto \max\{[t]_1 \mid t \in \mathsf{Term}(\Sigma), |t| \leq n\}$. Then the derivational complexity of a rewriting system R is bounded by the growth of any matrix interpretation that is compatible with R.

Weighted Automata [Dro09]. We use automata with weights in $\mathbb{N}$, corresponding to matrix interpretations. We only need word (not tree) automata.

A $\mathbb{N}$-weighted word automaton $A = (Q, \lambda, \mu, \delta)$ over signature Σ consists of mappings

$$\lambda : Q \to \mathbb{N}, \mu : \Sigma \to Q^2 \to \mathbb{N}, \delta : Q \to \mathbb{N},$$

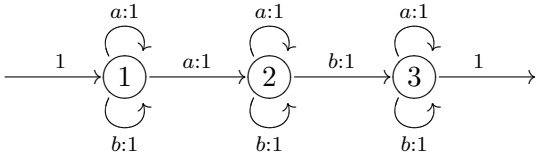
where we picture states as numbers, $Q = \{1, \ldots, d\}$, and $\lambda \in \mathbb{N}^{1 \times d}$ is the (row) vector of *initial* weights, for each letter $c \in \Sigma$, $\mu(c) \in \mathbb{N}^{d \times d}$ is a (square) *transition* matrix, and $\delta \in \mathbb{N}^{d \times 1}$ is the (column) vector of *final* weights. We extend μ homomorphically from letters to words by $\mu(u \cdot v) = \mu(u) \cdot \mu(v)$. For a word $w \in \Sigma^*$, we denote by $A(p, w, q)$ the entry at position (p, q) in the matrix $\mu(w)$. If $a = A(p, w, q)$, then we also write $p \xrightarrow{w:a}_A q$, and we define $p \xrightarrow{w}_A q$ as $A(p, w, q) > 0$. The weight $A(w)$ computed by A for a word $w \in \Sigma^*$ is given by $\lambda \cdot \mu(w) \cdot \delta$. The *growth function* growth_A of an $\mathbb{N}$-weighted automaton A over Σ is defined as the function $n \mapsto \max\{A(w) \mid w \in \Sigma^n\}$.

For a signature of unary function symbols (as in string rewriting), a d-dimensional matrix interpretation is a weighted automaton in this sense. It has states $Q = \{1, \ldots, d, d+1\}$. We have $\lambda = (1, 0, \ldots, 0)$ (the initial state is 1) and $\delta = (0, \ldots, 0, 1)^T$ (the final state is $d + 1$), and for $c \in \Sigma$, we construct $\mu(c)$ as follows: The interpretation of c is given by $[c] : x \mapsto M_1 x + v$, for a matrix $M_1 \in \mathbb{N}^{d \times d}$ and a vector $v \in \mathbb{N}^d$. From that we define

$\mu(c) = \begin{pmatrix} M & v \\ 0 \dots 0 & 1 \end{pmatrix} \in \mathbb{N}^{(d+1)\times(d+1)}$. Then for any $w \in \Sigma^*$, the weight $A(w)$ computed by the automaton is equal to the first (top) entry of the value $[w]$ of w under the interpretation.

Example 2.2. [continued] The transition matrices of the automaton are given on the left, and a pictorial representation is shown on the right:

$$\mu(a) = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \mu(b) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix},$$



One can now see that A (as a classical automaton) corresponds to the regular expression $\Sigma^* a \Sigma^* b \Sigma^*$, and as a weighted automaton it computes, for input w, the number of index pairs (i, j) with $i < j$ such that $w_i = a \wedge w_j = b$, equivalently, the number of inversions (with respect to $b < a$). This is exactly the function that is needed in the termination proof of $R = \{ab \to ba\}$. ∎

3. Terms and Words

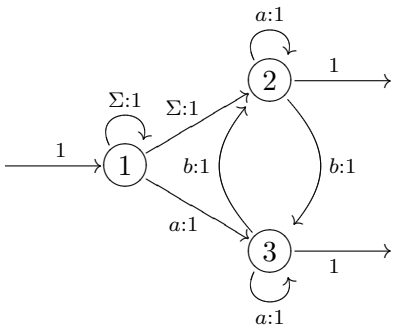
In this section we show that in order to bound the growth of a matrix interpretation (for a term rewriting system), it is sufficient to bound the growth of a $\mathbb{N}$-weighted word automaton. The reason is that a matrix interpretation corresponds to a rather restricted form of tree automaton, called *path-separated* [Kop09].

From a d-dimensional matrix interpretation $[\cdot]$ over Σ we construct a weighted word automaton A over the path signature $\Sigma' \subset \Sigma \times \mathbb{N}$ with states $Q_A = \{1, \dots, d\}$ by taking F_i (the matrix that is the factor for the i-th argument in the linear function $[f]$) as the transition matrix $\mu_A(f_i)$. The initial weight vector λ_A is $(1, 0, \dots, 0)$, and the final weight vector δ_A is obtained as follows. Denote by S the set of absolute parts of the interpretation $\{\mathsf{abs}[f] \mid f \in \Sigma\}$. Then $\delta_A(i)$ is 1 if there is some $v \in S$ with $v(i) > 0$. Otherwise, $\delta_A(i) = 0$. This automaton A can be used to bound the growth of the first (top) component $[t]_1$ of the interpretation of a term t.

Example 3.1. From the interpretation (for the unary signature $\{a, b\}$) on the left, we construct the automaton on the right:

$$[a] \quad : \quad x \mapsto \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot x + \begin{pmatrix} 0 \\ 2 \\ 2 \end{pmatrix}$$

$$[b] \quad : \quad x \mapsto \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \cdot x + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$



The final weight vector (indicated by outgoing arrows) is $\delta_A = (0, 1, 1)^T$. State 1 is not final because the top components of both absolute parts are zero. Note that the absolute parts of the interpretation are ignored except for their signum.

■

The following proposition formalizes an argument given in [Mos08] (before Theorem 6).

Proposition 3.2. *For a matrix interpretation* $[\cdot]$ *and the corresponding automaton* A*, there is a constant* C *such that for all* $t \in \mathsf{Term}(\Sigma)$ *we have* $[t]_1 \leq |t| \cdot C \cdot \mathsf{growth}_A(|t|)$.

Proof. By distributivity of matrix multiplication (over addition), the value of the matrix interpretation of a term t can be written as the sum of the values of matrix products along paths—and that is exactly what the automaton A computes:

$$[t] = \sum_{p \in \mathsf{Path}(t)} \mu_A(p) \cdot \mathsf{abs}[t_p].$$

We take C as the maximal entry of vectors in S. Then each $v \in S$ is point-wise smaller or equal to $C \cdot \delta_A$. Taking the first (top) component of $[t]$ corresponds to multiplication by λ_A from the left. In all, $[t]_1 = \lambda_A \cdot [t] \leq \sum_{p \in \mathsf{Path}(t)} C \cdot A(p) \leq \sum_{p \in \mathsf{Path}(t)} C \cdot \mathsf{growth}_A(|p|) \leq |t| \cdot C \cdot \mathsf{growth}_A(|t|)$, since $|p| \leq |t|$ (the length of a path compared to the size of the term). So the claim follows. ■

Since Proposition 3.2 introduces a factor $|t|$, we obtain the following

Theorem 3.3. *If* growth_A *(constructed from the matrix interpretation* $[\cdot]$*) is bounded by a polynomial of degree* g*, then* $\mathsf{growth}_{[]}$ *is bounded by a polynomial of degree* $g + 1$. ■

We also have a converse. For any $p \in \Sigma'^*$, there is a set T of terms $t \in \mathsf{Term}(\Sigma)$ with $|t| \leq D(1 + |p|)$ and $p \in \mathsf{Path}(t)$ and $\delta_A \leq \sum_{t \in T} \mathsf{abs}[t_p]$. Here, D is the maximal arity of Σ, and $|T| \leq d$, the dimension of the interpretation. The terms in T have the path p as their "spine", and some additional nullary symbols. At the end of the spine, there is some symbol to "cover" some non-zero entry of δ_A. Then $A(p) \leq \sum_{t \in T} \mu_A(p) \cdot \mathsf{abs}[t_p] \leq \sum_{t \in T} [t]_1$. That is, $\mathsf{growth}_A(n) \leq |T| \cdot \mathsf{growth}_{[]}(D(1 + n))$. If $\mathsf{growth}_{[]}$ is polynomially bounded, then growth_A is polynomially bounded.

Remark 3.4. The given translation ignores the entries in the absolute parts of the matrix interpretation. Indeed they do not influence the degree of the growth polynomial. Referring to Example 2.2, the present construction would remove state 3 which effectively acts as a "sink" state (no transition leaves this state). One may wonder whether state 1 could be ignored as well. In general, this may alter the degree of growth since there could be transitions from states > 1 to state 1.

4. Growth and Ambiguity

By Theorem 3.3, we will restrict our attention to weighted word automata. In the present section, we connect the weight function of a weighted automaton to the ambiguity of a non-weighted automaton. The *ambiguity* of a (non-weighted) automaton A is the function amb_A that maps each word $w \in \Sigma^*$ to the number of accepting computations (paths) of A on w.

Definition 4.1. Let A by an $\mathbb{N}$-weighted automaton. Obtain the skeleton $A' = \mathsf{skel}(A)$ by removing all weights. That is, (p, q) is an edge in $\mathsf{skel}(A)$ with label $c \in \Sigma$ exactly if $A(p, c, q) > 0$. If $\lambda_A(p) > 0$, then p is initial in A. If $\delta_A(p) > 0$, then p is final in A.

The following observation is immediate.

Proposition 4.2. *If all weights in A are from the set $\{0,1\}$, then the growth function of A and the ambiguity function of $\mathsf{skel}(A)$ are identical.*

Proof. We use the fact that $A(p,w,q)$ is the sum over the weights of all paths from p to q labelled w. Given the precondition of the proposition, the weight of a such a path in A is 1 or 0, respectively, if there exists a corresponding path in $\mathsf{skel}(A)$ or not, respectively. ∎

We observe now that the weight of an edge can be ignored if it is used at most once. To this end we define:

Definition 4.3. An edge (p,q) in $\mathsf{skel}(A)$ is called *recurrent* if there is a path from q to p in $\mathsf{skel}(A)$. All other edges are called *transitional*.

We will apply this notion to edges of weight > 0 in A as well.

Proposition 4.4. *Each path $p \xrightarrow{w} q$ uses each transitional edge of A at most once.*

Proof. Assume $p \to q$ is used twice. Then $p \to q \to^* p \to q$, and (p,q) is recurrent. ∎

Theorem 4.5. *The growth function of A behaves the same (up to a constant factor) as the growth function of the automaton A' obtained from A by giving weight 1 to each transitional edge.*

Proof. It is immediate that for all p,w,q: $A(p,w,q) \geq A'(p,w,q)$. Let N be the number of transitional edges in A. If $N = 0$, then there is nothing to show. For $N > 0$, let W be the maximal weight of a transitional edge. We claim that $A(p,w,q) \leq W^N \cdot A'(p,w,q)$. This follows since in each path, each transitional edge can be used at most once, and it contributes W (multiplicatively). ∎

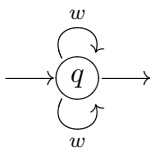
It is possible to reduce this constant W^N but we do not need this here.

We will see (Theorem 5.2, Item 1) that for A to be polynomially bounded, its recurrent edges have to have weight 1 (and not larger). Together with Proposition 4.5 this means that when we talk about polynomially bounded $\mathbb{N}$-weighted automata, we do not actually consider their edge weights in an essential way. This allows to apply known results on ambiguity, by Proposition 4.2.

5. Deciding Polynomial Growth

In this section, we give an algorithm to decide whether a given $\mathbb{N}$-weighted automaton has polynomially bounded behaviour. By applying the ideas from Section 4, we transform the problem to a question of ambiguity of non-weighted automata that can be solved with known methods. In particular we will apply

Theorem 5.1 ([Web91], Condition EDA)**.** *A trim automaton A over Σ is exponentially ambiguous if and only if there exist a state q of A and a word $w \in \Sigma^+$ such that there are at least two different paths $q \xrightarrow{w}_A q$.*



Here, a (classical) automaton is *trim* if each state is useful: it is accessible from some initial state, and it reaches some final state. For weighted automata A, we define the same concepts (trim, useful, accessible) by considering $\mathsf{skel}(A)$, that is, we use only paths of weight > 0.

A *strongly connected component* (SCC) of the automaton A is a maximal set C of states such that any two $p, q \in C$ are connected. Note that an edge (p, q) is recurrent (Definition 4.3) exactly if p and q belong to a common SCC.

A node is not necessarily connected to itself. Such nodes do not belong to any SCC, and they are called *transitional*. The incoming and outgoing edges of these nodes are transitional edges, as defined earlier. The automaton has a unique decomposition into SCCs and transitional nodes.

We call an automaton A *unambiguous* if it "contains no diamond" [Béa08]: there are no two paths with identical origin, end, and label.

We now characterize growth properties of $\mathbb{N}$-weighted automata:

Theorem 5.2. *For a trim $\mathbb{N}$-weighted automaton over Σ,*

1. *if there is a recurrent edge with weight > 1, then* $\mathsf{growth}(A)$ *is exponential.*
2. *if all recurring edges have weight one, and there is one SCC that is an ambiguous automaton, then* $\mathsf{growth}(A)$ *is exponential.*
3. *if all recurring edges have weight one and each SCC is an unambiguous automaton, then* $\mathsf{growth}(A)$ *is bounded by a polynomial.*

Proof. Item 1: assume there is some edge $p \xrightarrow{x:a} q$ with $a \geq 2$ in some SCC C. Since the edge is inside an SCC, there is also a path $q \xrightarrow{w:b} q$ with $b > 0$. Since the automaton is trim, there is a path $i \xrightarrow{w_i:a_i} p$ from some initial state i, and a path $p \xrightarrow{w_f:a_f} f$ to some final state. Then we can compose these paths, where xw gives a loop, and we obtain that for each $k \in \mathbb{N}$, the word $w_i(xw)^k w_f$ has at least weight 2^k.

In the following cases, we apply Proposition 4.2.

Item 2: assume there is some SCC C that is ambiguous. So it contains a diamond: there are states $p, q \in C$ and a non-empty word w such that there are two different paths from p to q labelled w. Since C is strongly connected, there is a path $q \xrightarrow{w'} p$. This implies that the condition of Theorem 5.1 holds true (for the state p and the word $w \circ w'$), and the automaton C is exponentially ambiguous.

Item 3: follows from Remark 7.3 and Proposition 7.4 below. There we will see that in this case it does not matter that the unambiguous components are strongly connected. ∎

Example 5.3. The interpretation shown in Example 3.1 is compatible with $\{ba \to ab, a^3 \to ba^2b, b^4 \to a\}$ (SRS/Zantema/z025). The conditions of Theorem 5.2 are fulfilled: SCCs are $\{1\}$ and $\{2, 3\}$. There are no edges with weight > 1. Each SCC is unambiguous. This is trivial for the singleton, and $\{2, 3\}$ is unambiguous since the restrictions of $\mu(a)$ and $\mu(b)$ to that component are permutation matrices. Any product of permutation matrices is again a permutation matrix, and has entries in $\{0, 1\}$ only. In general, the restriction to unambiguous components does not need to give a permutation matrix. ∎

6. Comparison to Triangular Method

We prove that our method for proving polynomial derivational complexity is strictly more powerful than the method of triangular interpretations [Mos08].

We recall that the matrices in a triangular interpretation must have zeroes below the main diagonal, and zeroes or ones on the main diagonal. The elements above the main diagonal are unrestricted. The interpretation in Example 2.1 is triangular.

We make the obvious observation that each triangular interpretation fulfills the conditions of Theorem 5.2, since the SCCs of the interpretation are singletons. All edges except loops are transitional.

The interesting statement is:

Theorem 6.1. *There is a rewriting system S with these properties:*

- *S has a compatible polynomially bounded matrix interpretation,*
- *S has no compatible triangular interpretation.*

The proof is contained in the rest of this section. The main technical result is a monotonicity property of triangular interpretations (Proposition 6.2).

We use signature $\Sigma = \{L, R, a, X\}$ and take the rewriting system

$$S = \{Raa \to aaR, RX \to LX, aaL \to Laa, XL \to XRa\}.$$

This is based on a system suggested by Jörg Endrullis for a related problem.

A typical S-derivation has R travelling right, and L travelling left: for any $k \geq 0$,

$$XRa^{2k}X \to^k Xa^{2k}RX \to Xa^{2k}LX \to^k XLa^{2k}X \to XRa^{2k+1}X \to^k Xa^{2k}RaX. \quad (6.1)$$

For termination, it is essential to count the length of blocks of a modulo 2. As the above derivation shows, $XRa^{\text{even}}X \to^* XRa^{\text{odd}}X$, but $XRa^{\text{odd}}X \not\to^* XRa^{\text{even}}X$.

There is a polynomially bounded matrix interpretation that is compatible with S, see Example 7.5 below.

We now prove that S has no compatible triangular interpretation of any dimension. Since the signature is unary, we consider the transition matrices of the weighted automaton corresponding to the interpretation, cf. Example 2.1 and Example 2.2. The relations $\geq, >$ for interpretations are expressed equivalently for matrices: we write $A \geq B$ if $\forall i, j : A_{i,j} \geq B_{i,j}$, and $A > B$ if $A \geq B$ and $A_{\text{top,right}} > B_{\text{top,right}}$.

Proposition 6.2. *For any upper triangular nonnegative integer matrix A of dimension d, the sequence $(A^d, A^{d+1}, \ldots)$ of powers of A is increasing: for all $k \geq d$ we have $A^k \leq A^{k+1}$.*

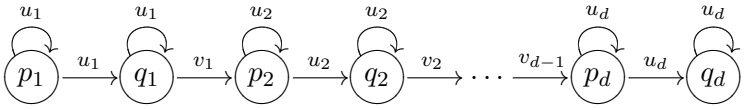
Proof. Take any $k \geq d$ and consider any pair of indices i, j with $1 \leq i, j \leq d$. Then $A^k_{i,j}$ is the sum of weights of paths of length k from i to j. For each such path p we construct a path p' (of length $k+1$) that contributes to $A^{k+1}_{i,j}$ in such a way that the construction $p \mapsto p'$ is injective and (weakly) weight-increasing. Since $|p| \geq d$, at least one vertex v is contained twice in p. Since A is upper triangular, occurrences of v must be consecutive in p. We construct p' from p by adding an edge $v \to v$ for the leftmost repeated index v of p. Since the weight of this new edge is ≥ 1, the weight of p' is at least the weight of p. By construction, the leftmost repeated vertex in p' occurs at least thrice. For any two non-equal paths p', q' constructed this way, we can delete the leftmost repetition and obtain two non-equal paths p, q of length k that contribute to $A^k_{i,j}$. ∎

Assume there is a triangular interpretation $[\cdot]$ of dimension d compatible with S. Take $k \geq d/2$. Then the interpretation must verify $[XRa^{2k}X] > [XRa^{2k+1}X]$ by the derivation 6.1. On the other hand Proposition 6.2 implies $[a^{2k}] \leq [a^{2k+1}]$. By monotonicity of multiplication, we obtain $[XRa^{2k}X] \leq [XRa^{2k+1}X]$, a contradiction. This proves Theorem 6.1.

7. Bounds for the Degree of the Growth Polynomial

The degree of ambiguity of a (classical) automaton can be determined by the following:

Theorem 7.1 ([Web91], Condition IDA$_d$)**.** *A trim automaton A over Σ is polynomially ambiguous of degree at least d if and only if there exist states $p_1, q_1, \ldots, p_d, q_d$ in A and words $u_1, \ldots, u_d \in \Sigma^+$ and $v_1, \ldots, v_{d-1} \in \Sigma^*$ such that $\forall 1 \leq i \leq d : p_i \xrightarrow{u_i}_A p_i \xrightarrow{u_i}_A q_i \xrightarrow{u_i}_A q_i$ and $\forall 1 \leq i < d : q_i \xrightarrow{v_i}_A p_{i+1}$.*



It is possible to check this condition in $O(|A|^6)$ steps, and also by a corresponding constraint system of this size. Still we found it to be infeasible submit this system to a constraint solver.

The following definition fits with the constraint system that we will describe later:

Definition 7.2. An unambiguous decomposition of an automaton A with state set Q is a system $U = \{U_1, \ldots, U_k\}$ of non-empty and pairwise disjoint subsets $U_i \subseteq Q$ such that

- each recurrent edge is contained in some U_i
- and for each i, the restriction of A to U_i is unambiguous.

The decomposition U defines a relation L_U on Q that consists of all pairs (p, q) such that p, q are in different components of U and there is a path from p to q.

The height of a decomposition is the height (length of a longest chain) of L_U.

Remark 7.3. The SCC decomposition in Theorem 5.2, Item 3 is an unambiguous decomposition. Its height is strictly smaller than the number of SCCs.

The connection to the degree of a polynomial growth bound is:

Proposition 7.4. *If all recurring edges of a trim $\mathbb{N}$-weighted automaton A have weight one, and $\mathsf{skel}(A)$ admits an unambiguous decomposition of height g, then A is polynomially bounded with degree g.*

Proof. Assume that condition IDA$_d$ holds. Then denote by P_i (Q_i, resp.) the component of the decomposition that contains p_i (q_i, resp.) Note that these components exist since both p_i and q_i are incident to recurring edges. We observe $P_i \neq Q_i$. (Otherwise, the common component would be ambiguous.) This implies $L_U(P_i, Q_i)$. We also have $P_i \neq P_{i+1}$. (Otherwise, $L_U(P_i, Q_i) \wedge L_U(Q_i, P_i)$, in contradiction to the finite height of L_U.) This implies $L_U(P_i, P_{i+1})$, and we infer $d < g$. This shows that the ambiguity of $\mathsf{skel}(A)$ is bounded by a polynomial of degree g. By Theorem 4.5, the growth function of the weighted automaton A is bounded by a polynomial of the same degree g. ∎

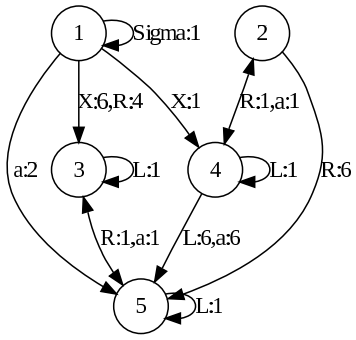
We remark that the statement in Proposition 7.4 is not an equivalence, in the sense that there may be decompositions U such that the height of L_U is larger than the degree of ambiguity. E.g., the SCC decomposition for Example 7.5.

Example 7.5. This interpretation is compatible with system S from Theorem 6.1:

$$[L] : x \mapsto \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot x + \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad [X] : x \mapsto \begin{pmatrix} 1 & 0 & 6 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \cdot x + \begin{pmatrix} 0 \\ 6 \\ 0 \\ 0 \\ 1 \end{pmatrix},$$

$$[R] : x \mapsto \begin{pmatrix} 1 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \cdot x + \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \quad [a] : x \mapsto \begin{pmatrix} 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 6 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \cdot x + \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}.$$

The translation from Section 3 turns this into a weighted automaton with 5 states, shown right. The SCCs of this automaton are $\{\{1\}, \{2, 4\}, \{3, 5\}\}$. Recurrent edges are the loops as well as the edges labelled R and a inside the two SCCs with two elements. The height of the SCC decomposition is two. Note that the only edge from 1 to $\{2, 4\}$ is labelled X, and this label does not occur inside $\{2, 4\}$. This implies that these two SCCs can be merged, resulting in the unambiguous decomposition $\{\{1, 2, 4\}, \{3, 5\}\}$. Its height is one, implying a quadratic bound on the derivational complexity of S. This bound is sharp since the rewrite system does admit derivation of quadratic length, e.g., $R^k(aa)^k \to^{k^2} (aa)^k R^k$. ∎



As an application of Proposition 7.4, we show how to modify triangular interpretations in order to improve (that is, reduce) the degree of their growth polynomial bound, in some cases.

Proposition 7.6. *For an upper triangular interpretation over alphabet Σ, let D be the set of indices p such that there exists $c \in \Sigma$ such that the entry at position p on the main diagonal of the linear term of the interpretation of c is positive. Then the interpretation is polynomially bounded with a degree of at most $|D|$.*

Proof. Each $p \in D$ constitutes a singleton SCC in the automaton. So all chains have length $\leq |D| - 1$, and by Theorem 3.3 the result follows. ∎

Example 7.7. (SRS/Zantema/z025) $\{ba \to ab, a^3 \to ba^2b, b^4 \to a\}$ is solved by the interpretation

$$[a] : x \mapsto \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot x + \begin{pmatrix} 0 \\ 1 \\ 2 \\ 3 \end{pmatrix}, \quad [b] : x \mapsto \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot x + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

where $D = \{1, 4\}$, from which we infer maximum degree $|D| = 2$. This bound is sharp since there are derivations of quadratic length because of the rule $ba \to ab$. ∎

8. Certificates for Polynomial Bounds

In this part of the paper we show how the conditions of Theorem 5.2 and Proposition 7.4 can be deduced from existence of a "certificate". We describe how to construct a (finite domain) constraint system that specifies validity of the certificate. The subjects of the constraints are relations on the set Q of states of the given $\mathbb{N}$-weighted (word) automaton A. These unknown relations are existentially quantified at the outer level.

We also build a constraint system that describes compatibility of an (unknown) linear interpretation $[\cdot]$ with a given rewrite system R.

We combine both systems and use a constraint solver to find the interpretation $[\cdot]$ and its polynomial growth certificate A at the same time.

The (by now) standard idea for constraint solving is to translate the constraint system into a propositional logic formula. In the present paper, we discuss this "SAT encoding" of the relational constraints only. For the encoding of compatibility constraints, we refer to [End08].

8.1. Encoding of Relations

A relation on finite domains is directly modelled as a matrix of propositional variables.

In the constraint systems we will need the identity relation, on some domain. We denote it by 1, and model it trivially with a matrix of propositional constants (True on the diagonal).

For relations R, S, the implication $R \subseteq S$ is modelled by point-wise propositional implication between the corresponding matrix entries. Intersection $R \cap S$ is modelled by point-wise "and", and union $R \cup S$ by point-wise "or".

We also need composition of relations $R \circ S$, and this corresponds to Boolean matrix multiplication; as well as the inverse R^- of a relation, corresponding to matrix transposition.

Then, we want to describe the image and pre-image of a relation w.r.t. a set. By abuse of notation, if we have a binary relation $R \subseteq A \times B$ and a unary relation (a set) $A' \subseteq A$, we write $A' \circ R$ for $\{b \mid \exists a \in A' : (a, b) \in R\}$, similarly for $R \circ B'$ for $B' \subseteq B$. This is realized by multiplying the Boolean matrix R with the Boolean vector A' (resp., B').

The cost of most operations is proportional to the square of the matrix dimension, except for composition (multiplication), where the cost is cubic. Here "cost" refers to the formula size, or, equivalently, to the number of additional propositional variables that will be created by conversion to an equisatisfiable conjunctive normal form.

8.2. Encoding SCCs

The automaton A defines for each $a \in \Sigma$ an "edge" relation

$$\mu_{>0}(a) = \{(p, q) : \mu(a)(p, q) > 0\},$$

and a "heavy edge" relation

$$\mu_{>1}(a) = \{(p, q) : \mu(a)(p, q) > 1\}.$$

An (over-)approximation $E \subseteq Q \times Q$ of the reachability relation of the automaton A is specified by the constraints

$$\bigcup\{\mu_{>0}(a) \mid a \in \Sigma\} \subseteq E \quad \wedge \quad E \circ E \subseteq E$$

Correctness claim: For any solution E of the constraint system: If there is any path $p \xrightarrow{w}_A q$ of length $|w| > 0$ and weight > 0, then $E(p, q)$ holds.

Note that we do require transitivity, but not reflexivity.

Also, we do not model reachability exactly. This would require to specify E as the smallest relation with the given properties. This is not easily expressed in the given logic, where we only have existential quantification. Over-approximation of reachability may even be helpful, as explained in Section 7.

The relation $S \subseteq Q \times Q$ (over-)approximates "being in the same SCC": $S = E \cap E^-$.

Correctness claim: for any solution (E, S) of the constraint system,

- if $p \to q$ is a recurring edge in A, then $S(p, q)$;

In particular $\neg S(p, p)$ implies that p is a transitional node.

Condition (1) of Theorem 5.2 is modelled by the constraint

$$S \cap \bigcup \{\mu_{>1}(a) \mid a \in \Sigma\} = \emptyset.$$

Correctness claim: if the constraint system has a solution, then each recurring edge of A has weight $= 1$.

The size of the constraint system is cubic in the number of states of A, since we need composition of relations (once).

8.3. Encoding Unambiguity

As in [Web91], we use the criterion that an automaton A is unambiguous if the reachable and productive states of the cross product automaton $A \times A$ are on its diagonal [Sak03].

We define a relation $T \subseteq (Q \times Q)^2$ that (over-)approximates the edge relation of the product automaton, by

$$\{((p, p'), (q, q')) \mid \exists a \in \Sigma : \mu_{>0}(a)(p, q) \wedge \mu_{>0}(a)(p', q')\} \subseteq T.$$

We use relations $R, P \subseteq Q \times Q$ with the intention that $R(p, q)$ holds true if the state $(p, q) \in A \times A$ is reachable, and $P(p, q)$ holds true if the state $(p, q) \in A \times A$ is productive. We specify:

- the diagonal states are reachable and productive: $1 \subseteq R \wedge 1 \subseteq P$,
- each successor of a reachable state is reachable: $(R \circ T) \cap S \subseteq R$,
  note that we restrict to transitions that stay inside the (approximated) SCCs.
- each predecessor of a productive state is productive: $(T \circ P) \cap S \subseteq P$,
- each state that is both reachable and productive, is on the diagonal: $R \cap P \subseteq 1$

Correctness claim: if the constraint system has a solution, then S describes an unambiguous decomposition U of A. Here, p and q are in the same component of U if $S(p, q)$. The height of L_U is finite. — Proof: By construction, each recurrent edge is contained in some component, and each component is unambiguous. A cycle of components is impossible since S is transitive.

The size of the constraint system is $\Theta(d^4)$ since we need to compute the (pre)image of a relation on Q^2.

Remark 8.1. If we collect all the constraints up to here, then we already have a method that is more powerful in proving polynomial complexity bounds than triangular interpretations. E.g., it finds a proof for the system in Theorem 6.1. In the following, we bound the degree of the growth polynomial.

8.4. Unary Encoding of (Small) Numbers

A number n is given by a unary relation N that we view as a subset of the range. In our case, $N \subseteq \{1, \ldots, d\}$ where d is the dimension of the interpretation = the number of states of the automaton. Equivalently, $N : \{1, \ldots, d\} \to \mathsf{Boolean}$. A value of k is encoded by the assignment $\{1, \ldots, k\} \mapsto \mathsf{True}, \{k+1, \ldots, d\} \mapsto \mathsf{False}$. That is, for any N encoding a number we have the constraint $\forall 1 \leq i < d : N(i) \Leftarrow N(i+1)$.

For the given application, we did not investigate other encodings (e.g., binary) as the range of numbers is small (it is the number of states of the automaton), and we do not need arithmetical operations, only comparison: $M > N$ is given by $\bigvee\{M(i) \wedge \neg N(i) \mid 1 \leq i \leq d\}$.

8.5. Encoding the Height of a Relation

The height of a relation $L \subseteq Q \times Q$ is the length of a longest L-chain. To express the condition "the height of L is at most g", we use a "height" function $h : Q \to \{0, 1, \ldots, g\}$ and the constraints

$$\forall p, q : L(p, q) \Rightarrow (h(p) > h(q)).$$

The range of h is implemented by unary numbers as discussed above.

The cost of this constraint is $\Theta(d^2 \cdot g)$, since we do d^2 comparisons that cost $\Theta(g)$ each.

We apply this for $g =$ the intended degree of polynomial growth, and the relation

$$L = \{(p, q) \mid S(p, p) \wedge \neg S(p, q) \wedge E(p, q) \wedge S(q, q)\}.$$

Correctness claim: if the constraint system has a solution, then the automaton A admits an unambiguous decomposition of height $\leq g$. — Proof: If p, q are recurrent nodes from distinct S-components such that A contains a path from p to q, then $L(p, q)$. Therefore, each L-chain is a chain of S-components, and each of them is unambiguous.

Remark 8.2. In the general case, an unambiguous component may contain recurrent and transitional edges, and the weight of transitional edges is irrelevant by Theorem 4.5. Since we use the relation S (for efficiency of implementation), we discard the possibility that unambiguous components contain transitional edges of weight > 1.

8.6. Encoding Improved Triangular Interpretations

We show that Proposition 7.6 can be implemented as a constraint system with little effort. We use binary variables $C_1, \ldots, C_d$ to encode membership in the set D:

$$\forall 1 \leq p \leq d : \forall c \in \Sigma : [c](p, p) > 0 \Rightarrow C_p$$

and we express that at most g of these variables are true, by a relation $Z \subseteq Q \times \{1, \ldots, g\}$ with the intention that $Z(p, h) =$ "at most h of $C_1, \ldots, C_p$ are true." This is specified by

$$Z(p, h) = (C_p \wedge Z(p-1, h-1)) \vee (\neg C_p \wedge Z(p-1, h)).$$

and obvious border cases. This constraint system is used in addition to the constraint system that describes compatibility of a triangular interpretation with the rewriting system. It is statically known that entries below the main diagonal are zero, so the multiplication of such matrices can be implemented with less effort than for full matrices. So we expect the constraint solver to be able to handle somewhat larger matrix dimensions. The interpretation in Example 7.7 was found this way.

9. Results

The method described in this paper was implemented for the termination analyzer Matchbox. Given a rewriting system R, the implementation produces constraint systems for various values of d (matrix dimension) and g (degree of growth) and submits them to solvers in parallel. As soon as a solution for some g is found, all other attempts for degrees $\geq g$ are terminated.

In the 2009 Termination Competition, Matchbox entered the category of Derivational Complexity/Full rewriting. Of the 616 problems, it could prove polynomial derivational complexity for 58 of them. The winner CaT got 98 answers. (The results of the third participant TCT are currently not available.)

In 3 cases, Matchbox got a better degree bound than CaT, and in two cases, Matchbox could prove polynomial complexity where CaT couldn't. In 9 cases, CaT got a better degree bound than Matchbox; and in 36 cases, CaT proved polynomial complexity where Matchbox couldn't.

The differences in performance are mainly due to the different methods that the programs apply: Matchbox uses the method of the present paper exclusively, while CaT uses a combination of triangular interpretations, root labelling, relative termination, arctic matrix interpretations and match-bounds.

10. Discussion

There are several open question related to polynomially bounded matrix interpretations. We mention only a few.

Formal verification of certificates for polynomial derivational complexity is possible. As a certificate, we can take the relations that are specified by the constraint system. Then it is easy to formally verify their validity, since it only needs propositional logic. (If a SAT solver can find it, then it is easy to check.)

Does there exist a rewriting system with polynomially bounded complexity that does not admit a polynomially bounded matrix interpretation? It is easy to answer "yes" for term rewriting: it was already noted in [End08] that the ground rewriting system $\{f(a,b) \to f(b,b), f(b,a) \to f(a,a)\}$ does not admit a matrix interpretation. Its derivational complexity obviously is linear. The question seems much harder for string rewriting, even in the following restricted setting:

Do all match-bounded string rewriting systems have a polynomially (even linearly) bounded matrix interpretation? Experience in the recent termination competition suggests otherwise. Still, this may be due to vastly different search methods. Certificates for match-boundedness can be found by completion [End06, Kor09], and quite often this gives large automata quickly. On the other hand, matrix interpretations are usually found via constraint solving (via SAT, as described here), and this usually cannot handle much more than dimension 5.

From a "practical" viewpoint, one would be interested in a constraint system that describes a certificate for the exact degree of the growth polynomial with less than $O(n^6)$ size—or in an altogether different method for the construction of automata that are compatible with a given rewrite system, and polynomially bounded. As very concrete challenges, one could try to find polynomially bounded matrix interpretations for two famous rewrite

systems z001= $\{a^2b^2 \to b^3a^3\}$ and z086= $\{a^2 \to bc, b^2 \to ac, c^2 \to ab\}$. In both cases, matrix interpretations (of dimension 5) are known, but they grow exponentially.

What about using different weight domains for the interpretations? It is easy to generalize the matrix interpretation method to rational (or real) numbers [Geb07], but it seems harder to obtain complexity information from such interpretations. We would need to decide whether a given $\mathbb{Q}$-weighted automaton is polynomially bounded. The easy connections from Section 4 do not hold, as a polynomially growing automaton could contain recurrent edges of weight > 1 (e.g., if they are directly followed by edges of suitable weights < 1).

Acknowledgements. I appreciate the anonymous referees' careful reading and detailed discussion.

References

[Baa98] Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[Béa08] Marie-Pierre Béal, Eugen Czeizler, Jarkko Kari, and Dominique Perrin. Unambiguous automata. *Mathematics in Computer Science*, 1(4):625–638, 2008.

[Dro09] Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of Weighted Automata*. Springer, 2009.

[End06] Jörg Endrullis, Dieter Hofbauer, and Johannes Waldmann. Decomposing terminating rewriting relations. In *Workshop on Termination*. 2006.

[End08] Jörg Endrullis, Johannes Waldmann, and Hans Zantema. Matrix interpretations for proving termination of term rewriting. *J. Autom. Reasoning*, 40(2-3):195–220, 2008.

[Geb07] Andreas Gebhardt, Dieter Hofbauer, and Johannes Waldmann. Matrix evolutions. In Dieter Hofbauer and Alexander Serebrenik (eds.), *Proc. Workshop on Termination, Paris*. 2007.

[Hof89] Dieter Hofbauer and Clemens Lautemann. Termination proofs and the length of derivations. In Nachum Dershowitz (ed.), *RTA*, *Lecture Notes in Computer Science*, vol. 355, pp. 167–177. Springer, 1989.

[Hof06] Dieter Hofbauer and Johannes Waldmann. Termination of string rewriting with matrix interpretations. In Frank Pfenning (ed.), *RTA*, *Lecture Notes in Computer Science*, vol. 4098, pp. 328–342. Springer, 2006.

[Kop09] Adam Koprowski and Johannes Waldmann. Max/plus tree automata for termination of term rewriting. *Acta Cybern.*, 19(2):357–392, 2009.

[Kor09] Martin Korp and Aart Middeldorp. Match-bounds revisited. *Inf. Comput.*, 207(11):1259–1283, 2009.

[Mos08] Georg Moser, Andreas Schnabl, and Johannes Waldmann. Complexity analysis of term rewriting based on matrix and context dependent interpretations. In Ramesh Hariharan, Madhavan Mukund, and V. Vinay (eds.), *FSTTCS*, *LIPIcs*, vol. 08004. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2008.

[Sak03] Jacques Sakarovitch. *Eléments de théorie des automates*. Vuibert, Paris, 2003.

[Wal09] Johannes Waldmann. Automatic termination. In Ralf Treinen (ed.), *RTA*, *Lecture Notes in Computer Science*, vol. 5595, pp. 1–16. Springer, 2009.

[Web91] Andreas Weber and Helmut Seidl. On the degree of ambiguity of finite automata. *Theor. Comput. Sci.*, 88(2):325–349, 1991.