Automatic Termination

Johannes Waldmann
Hochschule fr Technik, Wirtschaft und Kultur (FH) Leipzig
Fakultt IMN, PF 30 11 66, D-04251 Leipzig, Germany

Invited talk at RTA 2009 Brasilia.
The final publication is available at Springer via
https://doi.org/10.1007/978-3-642-02348-4_1

Abstract

We give an overview of applications of weighted finite *automata* to *automatically* prove termination of rewriting. Instances of this approach are: standard and arctic matrix interpretations, and the match bound technique. These methods have been developed in recent years, and they are being used by today's leading automated termination prover software.

1 Introduction

A rewriting system R defines a relation $\to_R$ on terms. Considering this as a model of computation, we are interested in derivations $i \to_R^* o$ from input i to output o. We actually want to obtain some output in finite time, so termination is a natural requirement: there is no infinite derivation starting from any i.

Another important application of rewriting is equational reasoning. Here one is concerned with the equivalence defined by the reachability relation $\leftrightarrow_R^*$. Termination comes into play since one wants to express this equivalence via a confluent and terminating rewriting system R' because this makes the reachability problem decidable. In fact, this can be seen as the historic motivation for developing methods for proving termination of rewriting, see Knuth-Bendix completion [24].

In recent years, there are increased efforts to prove termination of programs (logic, functional, imperative) via transformation to a rewriting termination problem, or applying methods from rewriting termination to the original problem.

Special focus is on obtaining termination proofs automatically. Such automatic provers then can be built into tools for completion or program analysis.

The present paper reports on termination methods that use weighted finite automata. They are being developed since 2003 (match-bound method) and it was realized only in 2006 (standard matrix method) that they have a uniform automata theoretic explanation. Once this was understood, the arctic matrix method could be derived "automatically" in 2007.

There has always been a strong connection between rewriting and the theory of automata and formal languages. Indeed, formal grammars are rewriting systems (plus a device for intersection with regular languages to remove remains of intermediate derivation steps since only the result is interesting). Much of the classic theory then is concerned with equivalent

ways of describing sets of descendants (reachable by applying grammar rules), of which we mention logic (e.g., monadic second order logic with one successor), algebra (language operations, e.g., regular expressions), and (finite) automata (pre-images of homomorphisms into (finite) algebras).

We list some connections between automata theory and rewriting (grammars): all the classes of the Chomsky hierarchy are closed w.r.t. intersection with regular languages, and in fact this can be proved, in each case, by constructing a grammar of the appropriate type that represents the intersection. More to the point, some language classes are known to be closed w.r.t. (many-step) rewriting: e.g., the image of a regular language under a monadic rewriting is again regular [6]. This result is proved by representing the language by a finite automaton, and then applying some closure construction.

2 Automata, Rewriting, . . . and Termination?

Indeed this leads us near the topic of the paper. The earliest of the methods under consideration here, namely match bounds, was obtained around 2002, when Dieter Hofbauer and the author visited Alfons Geser (then in Hampton/ Virginia), and together were trying to generalize the following well-known observation on solvable positions in the one-person game *Solitaire*. The object of this game is to remove pegs from a board, where one peg jumps over an adjacent peg, thereby removing it. A one-dimensional board is a string over the alphabet $\Sigma = \{O, X\}$, where X denotes a cell occupied by a peg, and O denotes an empty cell. Then a move of the game is an application of a rewrite step w.r.t. the system $S = \{XXO \to OOX, OXX \to XOO\}$. A solitaire position (a string $i \in \Sigma^*$) is *solvable* if there is a sequence of moves that leads to a position with only one peg, that is, to a string o in $L = O^*XO^*$. It is a folklore theorem that the set of solvable positions is a regular language. In other words, the set of (many-step) predecessors of L w.r.t. $\to_S$ is regular. It is not too hard to guess the corresponding regular expression, and verify it by a careful case analysis. Cristopher Moore and David Eppstein, who give the regular expression in [28], add: "[regularity] was already shown in 1991 by Thane Plambeck and appeared as an exercise in a 1974 book [Mathematical Theory of Computation] by Zohar Manna". Exercise III.5.5 in [4] by Jean Berstel requires to show that the congruence generated by "one-sided solitaire" $XXO \to OOX$ is a rational transduction (thus, regularity preserving).

Bala Ravikumar in [30] proved regularity of solvable solitaire positions on two-dimensional boards with fixed height (but unbounded width). He replaced the guessing of the regular expression by a constructive proof of the theorem that "the solitaire rewriting system preserves regularity of languages", and for the proof he introduced the idea of *change heights*. For a derivation in a length (in fact, shape) preserving rewriting system, the change height of a position measures the number of rewrite steps that touch this position. A system is change-bounded if there is a global bound on change heights, independent of the start term and the actual derivation sequence. Ravikumar's theorem is that each change-bounded string rewriting system preserves regularity of languages, and it can be shown that the change bound for the solitaire system is 4. This raised the natural question: is there a similar method that works for systems that are not length-preserving?

Note that at this point, the question was to give a constructive proof that certain rewriting systems preserve regularity of languages. From the proof, one could also infer that the rewriting system is terminating, but this was a side effect. In particular, for the solitaire system, termination is trivial (in each step, one peg is removed). With the

positive answer given by the *match bound* method, the side effect of proving termination soon became the main attraction.

Yet we postpone the discussion of match bounds because we try to structure the paper not historically, but systematically, as follows. In Section 3, we review automata with weights (in some semiring), as a generalization of classical automata (which are weighted in the Boolean semiring), and explain in Section 4 in general terms how they can certify termination of rewriting. In Section 5, we present the "(standard) matrix method" as an instance of automata with weights in $(\mathbb{N}, +, \cdot)$. For easier exposition, we start with string rewriting. The methods apply to term rewriting as well. We need weighted tree automata, introduced in Section 6. Then in Section 7, we show that the method also works for automata with weights from the arctic semiring $(\{-\infty\} \cup \mathbb{N}, \max, +)$. Using another semiring $(\mathbb{N}, \max, \min)$, we explain the match-bound method in Section 8. We then turn to the question of how do we actually find such certificates of termination for given rewrite systems. We discuss the general *constraint solving* approach in Section 9, and an *automata completion* approach that works especially well for match-bounds in Section 10. We review some open problems in Section 11, where we refer to the notion of matrix termination hierarchy. We close the paper with Section 12 by showing that weighted automata contribute to recent developments on derivational complexity of rewriting.

We do not attempt to give a complete and general overview of methods and techniques in automated termination. We focus on various instances of "matrix methods", by explaining them in the weighted automaton setting. Even there, the presentation will be somewhat biased: we do plan to cover the map, but at the same time emphasize some "forgotten" ideas and point to ongoing work and open problems.

3 Weighted Automata ...

A (classical) finite automaton $A = (\Sigma, Q, I, F, \delta)$ consists of a signature Σ, a set of states Q, sets $I, F \subseteq Q$ of initial and final states, respectively, and a transition relation $\delta \subseteq Q \times \Sigma \times Q$. We imagine the automaton as a directed graph on Q. For each $(p, c, q) \in \delta$, there is an edge from p to q labelled c. There may be loops and parallel edges. A path is a connected sequence of edges, and the label of a path is the word obtained as the concatenation of its edge labels. Write $A(p, w, q)$ for the statement "there is a path with label w from state p to state q". The automaton computes a Boolean-valued function $A : \Sigma^* \to \mathbb{B}$ with $A(w) = \bigvee\{A(p, w, q) \mid p \in I, q \in F\}$. Paths are combined sequentially and in parallel. Along a path, all edges must be present (conjunction); while there may be several paths with identical label, one of which is enough (disjunction).

This can be generalized: we can replace the Boolean domain $(\mathbb{B}, \vee, \wedge)$ with any other suitable structure. Then, an edge (still labelled with a letter) is not just present or absent, but it carries a weight, where weights are taken from a *semiring*. The semiring provides operations of addition (used in parallel composition) and multiplication (for sequential composition), and fulfill several axioms that just map naturally to the idea of composing paths and their weights.

Formally, a W-weighted automaton A consists of $(\Sigma, Q, I, F, \delta)$ where function $\delta \subseteq Q \times \Sigma \times Q \to W$ assigns weights to transitions, and the sets of initial (final, resp.) states are replaced by initial (final, resp.) weight assignments I (F, resp.) Each path from p to q with label u now has a weight $A(p, u, q) \in W$, computed as the product of its edge weights. The automaton assigns to a word $u \in \Sigma^*$ the value $A(u) = \sum\{I(p) \cdot A(p, u, q) \cdot F(q) \mid p, q \in Q\}$.

It is a nice coincidence that the talk will be given in Brazil: much of today's knowledge on weighted automata builds on work by Brazilian scientist Imre Simon (`http://www.ime.usp.br/~is/`) who used finite automata over the $(\mathbb{N}, \min, +)$ semiring in a decision procedure for the Finite Power Property of regular languages [31], and other problems in formal languages (for an more recent overview, see [32]). In fact this semiring was later named with reference to him the *tropical semiring*. We use its counterpart, the arctic semiring, in Section 7.

4 . . . for Termination of Rewriting

If the weight semiring W of the automaton A is well-founded w.r.t. some order $>$, and A fulfills the following condition w.r.t. a rewriting system R, called *global compatibility*: $u \to_R v$ implies $A(u) > A(v)$, then R is terminating. This is a trivial statement and it is not effective since in general there seems to be no way to test global compatibility, since we need to check this condition for all terms and rewrite steps. We turn this approach into a useful termination method by giving a *local compatibility* condition on the automaton that implies global compatibility and that can be checked effectively, e.g., by inspection of a finite number of cases. Roughly speaking, the infinite number of rewrite steps of the system R is partitioned into a finite number of classes by considering all possible locations of the rewrite step "in the automaton".

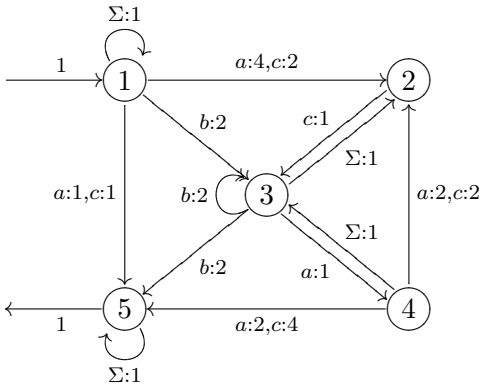
We will strive to formulate local compatibility in such a way that for a given rewrite system, an automaton that fulfill the conditions can (in principle) be found by automated constraint solver software. Such an automaton is then indeed a (finite) certificate for termination. Its validity can be checked independently from the way it was obtained. One application is automated verification of termination proofs, which is done in a two-step process: the underlying theorems (local compatibility $\Rightarrow$ global compatibility $\Rightarrow$ termination) for standard [26] and arctic [25] matrix interpretations have been formalized and proved by Adam Koprowski and are now part of the Color library of certified termination techniques [5]. Then for each concrete termination certificate, the correct application of the theorem has to be checked. With our notions of local compatibility, this is (conceptually) easy, since it consists of checking the solution of a constraint system, see Section 9.

5 Matrix Interpretations

As a first instance of the general scheme, we describe how automata with weights in the "standard" semiring of natural numbers $(\mathbb{N}, +, \cdot)$ are used for proving termination of string rewriting. (For term rewriting, see Section 6.)

Example 1 *This is the automaton from [21] that helped to sell the matrix method, because it certifies termination of* $Z_{086} = \{a^2 \to bc, b^2 \to ac, c^2 \to ab\}$*, which solved an open problem. (Here,* Z_{086} *is the name of this problem in the Termination Problem Data Base*

(TPDB), where "Z" refers to its author Hans Zantema.)



For all pairs (p, q) of states, and rules $(l \to r) \in R$, we consider the weights $A(p, l, q)$ and $A(p, r, q)$ computed by the automaton. We require weak local compatibility everywhere: $A(p, l, q) \geq A(p, r, q)$, this already ensures that $u \to_R v$ implies $A(u) \geq A(v)$. In the example, we have,e.g., $2 = A(4, bb, 3) \geq A(4, ac, 3) = 2$. □

Additionally, we need a strict decrease in a few well-chosen places. The original matrix paper [21] treated this with a theory based on positive cones in rings, which we here present in an automata theoretic setting:

Proposition 1 *For any string rewriting system $R = \{l_1 \to r_1, \ldots, l_n \to r_n\}$ over Σ, and $(\mathbb{N}, +, \cdot)$-weighted automaton $A = (\Sigma, Q, I, F, \delta)$ that is weakly locally compatible with R, define the language $D \subseteq \Sigma^* \cdot \{1, \ldots, n\} \cdot \Sigma^*$ as*

$$\left\{ u \cdot k \cdot v \mid \exists i, p, q, f \in Q : \begin{array}{lllll} I(i) \geq 1 & \wedge & A(i, u, p) \geq 1 & & \\ & \wedge & A(p, l_k, q) > A(p, r_k, q) & & \\ & \wedge & A(q, v, f) \geq 1 & \wedge F(f) \geq 1 & \end{array} \right\}.$$

If $D = \Sigma^* \cdot \{1, \ldots, n\} \cdot \Sigma^*$, then A is globally compatible with R. This condition on D is decidable. □

The language D encodes the set of all reachable and productive redex/contractum positions for which the automaton computes a decrease, so the first part of the statement is immediate. The second part is seen as follows: if we map 0 to "false", and each positive number to "true", then we have a semiring morphism from $(\mathbb{N}, +, \cdot)$ to $(\mathbb{B}, \vee, \wedge)$ that also respects the ordering (where "false" $<$ "true"). This means the languages $\{u \mid A(I, u, p) > 0\}$ and $\{v \mid A(q, v, F) > 0\}$ are effectively regular. The triplets $T = \{(p, k, q) \mid A(p, l_k, q) > A(p, r_k, q)\}$ can be found by considering finitely many cases, so the condition can be decided with methods from classical automata theory. We remark that this involves (roughly) a check whether a non-deterministic finite automaton accepts Σ^*, which is expensive (in fact, PSPACE-complete).

Example 1 meets the conditions of the above proposition, since $\{(1, k, 5) \mid 1 \leq k \leq 3\} \subseteq T$, e.g., $4 = A(1, bb, 5) > A(1, ac, 5) = 2$, and for each $u, v \in \Sigma^*$ we have $A(1, u, 1) > 0$ and $A(5, v, 5) > 0$ because of the loops in the respective states. We remark that the paper [21] contains other realizations of strict local compatibility, but none of them seems to have been used seriously in implementations.

If our goal is proving top termination, e.g., because we applied the dependency pairs transformation [2], then the above scheme can be adapted easily by restricting $u = \epsilon$ in Proposition 1.

As described in work with Andreas Gebhardt and Dieter Hofbauer [14], these methods work even in larger semirings like the non-negative rational, algebraic or real numbers. The natural order $>$ on those domains is not well-founded, so we replace it by $x >_\epsilon y \iff x > \epsilon + y$, for some fixed $\epsilon > 0$. The nice thing is that we do not need to change Proposition 1, since we carefully wrote ≥ 1 which is equivalent to > 0 on the naturals, but does the right thing on the dense domains. Note that we can take

$$\epsilon = \min\{d \mid (l \to r) \in R, d = A(p, l, q) - A(p, r, q), d > 0\}$$

which is positive since R is finite.

6 Weighted Tree Automata

For easier exposition, so far we only considered string rewriting. Since most "real" data is (tree-)structured, there is some interest in term rewriting, and we show how our methods generalize. We need the concept of weighted tree automaton [10, 9]. This is a finite state device that computes a mapping from trees over some signature into some semiring. This computational model is obtained from classical (Boolean) tree automata by assigning weights to transitions.

Formally, a W-weighted tree automaton is a tuple $A = (\Sigma, Q, \delta, F)$ where W is a semiring, Q is a finite set of states, Σ is a ranked signature, δ is a transition function that assigns to any k-ary symbol $f \in \Sigma_k$ a function $\delta_f : Q^k \times Q \to W$ and F is a mapping $Q \to W$. The idea is that $\delta_f(q_1, \ldots, q_k, q)$ gives the weight of the transition from $(q_1, \ldots, q_k)$ to q, and $F(q)$ gives the weight of the final state q.

The semantics of a weighted tree automaton is defined as follows: a run of A on a tree t is a mapping from positions of t to states of A, the weight of a run is the product of the weights of its transitions, and the weight of the term is the sum of the weight of all its runs. We emphasize here another, equivalent approach: the automaton is a Σ-algebra where the carrier set consists of weight vectors, indexed by states. Let $V = (Q \to W)$ be the set of such vectors. Then for each k-ary symbol f, the transition δ_f computes a function $[\delta_f] : V^k \to V$ by $[\delta_f](\vec{v}_1, \ldots, \vec{v}_k) = \vec{w}$ where

$$\vec{w}_q = \sum\{\delta_f(q_1, \ldots, q_k, q) \cdot \vec{v}_{1,q_1} \cdot \ldots \cdot \vec{v}_{k,q_k} \mid q_1, \ldots, q_k \in Q\}.$$

A weighted tree automaton realizes a multilinear algebra: each function $[\delta_f]$ is linear in each argument.

In order to use tree automata for automated termination, the local compatibility condition should be easy. Therefore we only consider automata whose transition functions can be written as sums of unary linear functions. These are matrices, so we arrive at the notion of matrix interpretation, using functions $V^k \to V$ of shape

$$(\vec{v_1}, \ldots, \vec{v_k}) \mapsto M_1 \cdot \vec{v_1} + \ldots + M_k \cdot \vec{v_k} + \vec{a}, \tag{1}$$

where each M_i is a square matrix, and $\vec{a}$ is a vector, and all vectors are column vectors.

The corresponding tree automata are called *path-separated* because their semantics can be computed as the sum of matrix products along all paths of the input tree, and the values along different paths do not influence each other.

With these preparations, we can apply the monotone algebra approach [13] for proving termination of term rewriting, where the algebra is given by a path-separated weighted tree automaton. The order on the vector domain depends on the chosen weight semiring, and on the question whether we want closure under contexts (we don't need this for top rewriting). In each case, we can take some modification of the pointwise extension of the semiring order.

We briefly discuss the relation to polynomial interpretations [7], which are a well-known previous instance of monotone algebras for termination. The distinctive features are that polynomial interpretations can be non-linear while matrix interpretations are linear, but this is complemented by the fact that polynomial interpretation use a totally ordered domain (of natural or real numbers) while the domain for matrix interpretations consists of vectors with the point-wise ordering, which is non-total.

7 Half-Strict Semirings

The formulation of a sufficient local compatibility condition depends on properties of the semiring operations. We call an operation $\circ$ *strict* w.r.t. an order $>$ if $x_1 > x_2$ implies $x_1 \circ y > x_2 \circ y$. On naturals, standard addition is strict, and standard multiplication is strict for $y \neq 0$.

We intend to use the arctic semiring $(\{-\infty\} \cup \mathbb{N}, \max, +)$, and we immediately notice that the "max" operation is not strict. Still it does have the following property: $x_1 > x_2 \wedge y_1 > y_2$ implies $\max(x_1, y_1) > \max(x_2, y_2)$. We call this *half strict*, since we need two strict decreases in the arguments to get one strict decrease in the result.

Now look at one strict decrease for a redex, $A(p, l, q) > A(p, r, q)$, in some weighted string automaton A. Strict multiplication produces from that a strict decrease when we apply a context (left and right). The automaton adds the weight of all paths with identical label. If addition is strict, then *one* decreasing path is enough, and that was the idea behind Proposition 1.

Now in the arctic semiring, addition is half-strict, so a sufficient condition for a global weight decrease is that *all* redex paths must be decreasing. We can make an exception for redex paths of weight zero (i.e., "missing" paths), since the max operation is strict when one argument is $-\infty$. Therefore we compare arctic weights by $x \gg y \iff x > y \vee x = y = -\infty$, and arctic vectors by the pointwise extension of that. Since $\gg$ is not well-founded, we need to make sure that we never produce a total weight vector $-\infty^d$. This can be achieved by requiring that the first vector element is "positive" (that is, $> -\infty$). This restricts the domain of the algebra, so the operations have to respect that. It is enough to require that the upper left entry of the matrices is positive as well. We remark here that the focus on the first vector component is just one way of reaching the goal, and the derivation of a more general criterion in the spirit of Proposition 1 is left as an exercise.

Example 2 *Again, we prove termination of* $Z_{086} = \{a^2 \to bc, b^2 \to ac, c^2 \to ab\}$. *The nontrivial dependency pairs are* $\{Aa \to Bc, Bb \to Ac\}$. *Take the arctic two-state automaton with transition matrices*

$$[a] = \begin{pmatrix} 0 & 3 \\ 2 & 1 \end{pmatrix}, [b] = \begin{pmatrix} 3 & 2 \\ 1 & -\infty \end{pmatrix}, [c] = \begin{pmatrix} 0 & 1 \\ 3 & 2 \end{pmatrix}, [A] = [B] = (0 \ \ -\infty),$$

where we use a reduced matrix shape for the top symbols (only the transitions from the initial

state). Then we have these weak compatibilities

$$[a^2] = \begin{pmatrix} 5 & 4 \\ 3 & 5 \end{pmatrix} \geq \begin{pmatrix} 5 & 4 \\ 1 & 2 \end{pmatrix} = [bc], \qquad [b^2] = \begin{pmatrix} 6 & 5 \\ 4 & 3 \end{pmatrix} = [ac]$$

$$[c^2] = \begin{pmatrix} 4 & 3 \\ 5 & 4 \end{pmatrix} \geq \begin{pmatrix} 4 & 2 \\ 5 & 4 \end{pmatrix} = [ab], \qquad [Aa] = (0\ 3) \geq (0\ 1) = [Bc]$$

and the strict compatibility $[Bb] = (3\ 2) > (0\ 1) = [Ac]$. *This allows to remove one rule and the rest is trivial (counting symbols).* □

Another hard termination problem was $\{b^3 \to a^3, a^3 \to aba\}$. Termination could not be established automatically by any of the programs (nor their authors) taking part in the competition 2006. Then, Aleksey Nogin and Carl Witty produced a handwritten proof, that was later generalized to the method of *quasi-periodic interpretations* [35]. It is now known that quasi-periodic interpretations of slope one over unary signatures can be translated into arctic matrix interpretations (see full version of [25], submitted). In fact, Matchbox used this translation in the termination competition of 2008.

We now turn to arctic interpretations for proving termination of term rewriting. We restrict to path-separated automata, as described earlier. At each position of a tree, values from subtrees are added. The redex position might be in any of the subtrees, and this creates a problem: we want then a decrease of the value of the tree, but arctic addition is not strict. This rules out the possibility of using path-separated arctic tree automata for proving full termination. They are still useful: the subtree problem does not appear when there are no redexes in subtrees, and this happens exactly when the redex position is in the root. So, arctic tree automata are applicable for proving top termination. (Of course there are non-top redexes but we don't need a strict decrease when reducing them.) This plan has been carried out and it is described in [25]. This paper also contains an extension to arctic numbers "below zero", i.e., the semiring $(\{-\infty\} \cup \mathbb{Z}, \max, +)$. Restricting the first component of the vectors to be positive works again.

8 Match Heights

As mentioned in the introduction, the idea of annotating positions in strings by numbers that give some indication of their rewrite history, derives from Ravikumar's concept of change bounds. The restriction to length-preserving rewriting can be dropped by considering match-heights instead. Here, the match height in the contractum is 1 larger than the lowest match-height in the redex. It is proved in [16] that match-bounded string rewriting systems are terminating, and effectively preserve regularity of languages.

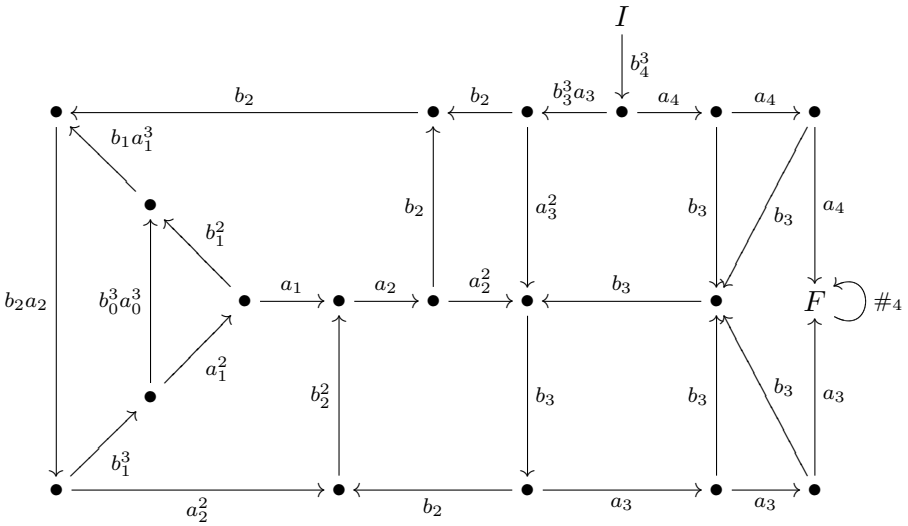
The relation to weighted automata was explained only some time later [33], and uses the $(\mathbb{N}, \min, \max)$ semiring. Here, none of the semiring operations is strict, so we expect complications. And indeed, both semiring addition and semiring multiplication are idempotent, and that means that for any finite weighted automaton A, the range $A(\Sigma^*)$ of the automaton's semantics function is finite. This also bounds the length of decreasing chains of weights by a constant, and that in turn bounds the length of derivations of the rewriting system we hoped the automaton to be strictly compatible with. This seems to rule out the use of this semiring completely, since every non-empty string rewriting system has at least linear, thus unbounded derivation lengths.

The solution is: we still apply a local compatibility condition in $(\mathbb{N}, \min, \max)$, but globally, we use a different semiring (where multiplication is not idempotent): its elements are the finite multisets of $\mathbb{N}$, semiring multiplication is multiset union, and semiring addition is the "min" operation w.r.t. the lexicographic ordering. Note that the multiplicative unit is the empty multiset, and for the additive unit we introduce an extra element ∞, larger than all others and absorbing for multiplication.

Now each $(\mathbb{N}, \min, \max)$-automaton can be lifted to an automaton in this multiset semiring, by changing each weight w into the multiset weight $\{w\}$, and the value that the automaton computes for some word u is the smallest (in the above sense) multiset of edge weights of a path labelled u.

The strict local compatibility condition (for the automaton in the original semiring) reads as expected: we require that for all $(l \to r) \in R, p, q \in Q : A(p, l, q) \gg A(p, r, q)$ where $x \gg y$ if $x > y \vee x = y = +\infty$. Basically, we replace the lexicographic comparison of multisets by the comparison of their respective maximum elements. That is, we ignore multiplicities.

Example 3 *The following automaton is strictly locally compatible with Zantema's system* $Z_{001} = \{a^2b^2 \to b^3a^3\}$*, and the picture (drawn by Dieter Hofbauer) is too nice to be omitted here.*



We use "edge compression", e.g., $\bullet \xrightarrow{a_2^2} \bullet$ *really means* $\bullet \xrightarrow{a_2} \bullet \xrightarrow{a_2} \bullet$ *containing one additional state. The indices on the letters indicate weights. These are from the semiring* $(\mathbb{N} \cup \{\infty\}, \min, \max)$*, since all "missing edges" have weight* ∞*. The symbol # is referring to the RFC method, see Section 10. The given automaton constituted (in 2003) the first automated termination proof for* Z_{001}*, while only "hand crafted" proofs were available [34].* □

The above presentation may not look like the standard version of match-bounded termination, but note that already there we used a multiset argument to bound lengths of derivations. The approach given here leads to extensions of the standard method, e.g., for relative termination. Note that the obvious $A(p, l, q) \geq A(p, r, q)$ for weak local compatibility in the

original automaton does not imply weak compatibility in the lifted automaton. But this can be repaired by replacing $\geq$ with $\gg'$ where $x \gg' y$ if $x > y \vee x = y = +\infty \vee x = y = -\infty$. That is, the "relative rules" must behave like strict rules (decrease weights) except they may keep the lowest weight ($-\infty$).

Is this useful for proving termination (without relative rules)? It is easy to show that a sequence of rule removals by relative match-bound proofs still implies a match-bound on the original system. Even so, the sequence of relative proofs may be easier to find automatically.

The idea of termination proofs via height annotations has been applied in term rewriting. There, one uses $(\mathbb{N}, \min, \max)$-weighted tree automata, again with a suitable local compatibility condition, where one has to take into account the position of variables in rewrite rules, resulting in the concept of roof-bounds [19].

9 Constraint Solving

One method of finding matrix interpretations (weighted automata) is constraint programming: write down all the properties, and find a satisfying assignment by some constraint solver software. At top level, the unknowns are the coefficients of linear functions with the shape of Equation 1, and the constraints relate linear functions that represent interpretations of left-hand sides and right-hand sides of rewrite rules. Here, it comes in handy that these functions are closed under substitution. This is caused by path-separation and would not hold for general automata. Constraints for linear functions can be translated to constraints on matrices; and constraints for matrices can be translated to constraints on their elements. Here, we arrived at constraints for natural numbers, since all our semirings use numbers.

Numbers can be represented in binary notation, and numerical constraints should be formulated in the SMT (satisfiability modulo theories) language QB-BV (quantifier free bit vector arithmetics). Instead, the numerical constraint is translated into CNF-SAT and then a SAT solver is applied. It seems strange that "manual" translation of termination constraints to CNF-SAT should be more efficient than applying QF-BV solvers. One explanation could be that so far, the power of matrix termination provers is increased by taking larger matrices, rather than larger matrix entries. But this may be a self-fulfilling prophecy.

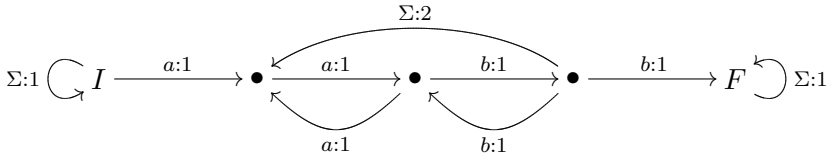
The constraint solver implementation in Matchbox in competition 2006 was internally dubbed "the one-bit wonder" since it used a bit width of *one* only. Today, Matchbox is using carefully optimized CNF-SAT encodings for binary arithmetics on standard and arctic naturals of bit widths 3 and 4. These were computed with the help of Peter Lietz. The translation of the Z_{001} and Z_{086} constraint systems for matrix dimension 4 and bit width 3 produces $\approx$ 3000 boolean variables and $\approx$ 20.000 clauses, of which Minisat [11] eliminates just one percent and then solves them in a few seconds.

It is an interesting idea to look for $(\mathbb{N}, \min, \max)$ interpretations via constraint solving. This may prove useful in connection with relative match-bound methods, see the discussion at the end of Section 8. E.g., we can prove that the TPDB systems SRS/Zantema06/$\{15, \ldots, 18\}$ are match-bounded. (This seems out of reach of completion methods.) Why would we want to do this? We could prove termination by other methods, but match-boundedness gives linear complexity, see Section 12.

10 Automata Completion

The constraint system that describes local compatibility requires certain inequalities between weights of paths in the automaton. We can fix the number of states, and then find appropriate edge weights, as described in the previous section. Another idea is to start with some candidate automaton, and then extend it by adding transitions *and states*, until all constraints are met. The problem is that additional states lead to additional constraints, and it is not clear how to organize this into a terminating algorithm.

Example 4 *Dieter Hofbauer's termination prover MultumNonMulta contains a version of weighted automata completion for the standard semiring* $(\mathbb{N}, +, \cdot)$*. It finds the following beautiful proof for (again)* Z_{001} *by starting with the redex path from left to right, and then adding three back edges.*



□

We discuss $(\mathbb{N} \cup \{\infty\}, \min, \max)$ now which we need for match-bounds. This semiring has the interesting property that its zero element (∞, which is neutral for the "min" operation) is maximal in the natural ordering. Consider local compatibility constraints $A(p, l, q) \gg A(p, r, q)$. We claim they can be read as "for every non-zero redex, there must be a smaller contractum". Indeed for zero-weight redexes, the constraint is true by the definition of $\gg$.

For a weighted automaton A, denote by $\mathrm{supp}(A)$ the set of words that get non-zero weight. If the weight semiring has zero as its maximum element, then local compatibility implies that $\mathrm{supp}(A)$ is closed under rewriting. Thus if we know that $L \subseteq \mathrm{supp}(A)$ and A is strictly locally compatible with R, then R is terminating on L. In other words, we have a method for proving *local termination* [8]. This allows to do termination proofs by considering right hand sides of forward closures [16]. They can be computed by the rewriting system

$$R' = R \cup \{l_1 \# \to r \mid (l_1 l_2 \to r) \in R, l_1 \neq \epsilon \neq l_2\},$$

starting with $L = \mathrm{rhs}(R)\#^*$, cf. Example 3.

Now, how do we realize the completion of automata? The basic approach is that whenever $A(p, l, q)$ is nonzero (= present), but $A(p, r, q)$ is zero (= missing), then we add a fresh path from p to q, labelled r and weighted appropriately. Several completion results for classical automata are available from the literature. In simple cases, they prove that the basic approach terminates. This happens, e.g., for monadic systems, where $|r| \leq 1$ and thus we never add states, only edges, and we will obtain a saturated automaton.

For rules with $|r| > 2$, this will not work. We should then employ some heuristics that tries to avoid the creation of too many fresh states [18]. A similar idea can be applied to tree automata [19] but in its basic version, it only works for left-linear rewriting. Non-linearities can be handled with quasi-deterministic automata as described by Martin Korp and Aart Middeldorp in [27]. The methods are sound but not complete (they may fail to find compatible automata). We now describe a complete method for string rewriting.

Instead of a W-weighted automaton A over Σ we can also consider a classical (Boolean) automaton B over $\Sigma \times W$. Local compatibility of A w.r.t. a rewriting system R then translates into local compatibility of B w.r.t. an annotated rewriting system R_W over

$\Sigma \times W$. For $W = (\mathbb{N}, \min, \max)$, this annotated system contains rules $l_W \to r_W$ such that the maximal annotation in the left is larger than the maximal annotation in the right. In other words, a rewrite step deletes the maximal letter. We have an instance of a *deleting* string rewriting system [20]: there is a well-founded order on the alphabet such that in reach rule $l \to r$, there is one letter in l that is larger than each letter in r. Deleting systems are terminating and preserve regularity of languages. The idea behind the proof is that for a deleting system R, the rewrite relation $\to_R^*$ can be represented as the composition of two rewrite relations $\to_C^* \circ \to_E^*$ over an auxiliary alphabet, where C is SN $\cap$ CF (not the French railway, but Strongly Normalizing and Context-Free: left-hand sides have length 1) and E is inverse context-free (right-hand sides have length ≤ 1).

The proof is constructive and indeed it was realized in the termination prover Matchbox (2003). However this implementation was soon outperformed by Hans Zantema's implementation of a completion heuristics in Torpa (2004). The situation was reversed again in 2006 when Jörg Endrullis found a substantial improvement for handling deleting systems [12] and implemented it for Jambox: the auxiliary alphabet can be small and the closures w.r.t. C and E can be computed in an interleaved manner. This gives us the best of both worlds: the construction is fast (it can build automata of $> 10^4$ states in < 10 seconds) and it is complete (if a match-bound certificate exists, it will be found). We remark that this seems to be the only instance of an weighted automata method for termination where we have a complete construction. This result also implies that the question "is a given rewriting R system match-bounded by k" is decidable. When k is not given, decidability remains open.

If we reverse all arrows in a (C, E) decomposition of a deleting system, we get some results on "inverse match-bounded" rewriting [17]. There should be a connection to the $(\mathbb{N}, \max, \min)$ semiring, but it is not immediate, e.g., the above multiset argument does not work, and indeed inverse match-boundedness does not imply termination, but termination is decidable. We observed [22] that match-boundedness, inverse match-boundedness, and change-boundedness are equivalent for length-preserving string rewriting systems. In particular this holds for the Solitaire rewriting system.

11 Matrix Termination Hierarchy

Typically, a proof of termination is obtained by a sequence of rule removals. In [15], Andreas Gebhardt and the author propose the notation $R \vdash S$ for rewriting systems R, S with $R \supseteq S$ and $(R \setminus S)$ is terminating relative to S, that is, from R, all non-S rules "could be removed". The relation $\vdash$ is indeed transitive, and $R \vdash^* \emptyset$ implies termination of R. Let $\mathfrak{M}(W, n)$ be the set of pairs of rewriting systems (R, S) such that if there is a W-weighted automaton with $\leq n$ states that is strictly compatible with $R \setminus S$ and weakly compatible with S. Thus $(R, S) \in \mathfrak{M}(W, n)$ implies $R \vdash S$ and therefore we also write $R \overset{\mathfrak{M}(W,n)}{\vdash} S$.

Since $\mathfrak{M}(W, n)$ is a relation on rewriting systems, we make use of standard operations on relations like composition, iteration (exponentiation) and (reflexive and) transitive closure. Then, the collection $\mathfrak{M}(W, n)^s$ is the "matrix termination hierarchy". We have some obvious (non-strict) inclusions for the levels of this hierarchy, considering the embeddings of natural $\subset$ rational $\subset$ algebraic $\subset$ real numbers (on the non-negative subset, with standard operations and ordering), and the monotonicity w.r.t. number of states and proof steps. Then, interesting questions can be raised, like, which levels of the hierarchy are inhabited, which are decidable, and which of the obvious inclusions are strict.

The paper [15] investigates weight domains that are sub-semirings of $\mathbb{R}_{\geq 0}$, and contains some concrete results, like $\mathfrak{M}(\mathbb{N},0) \subset \mathfrak{M}(\mathbb{N},1) \subset \mathfrak{M}(\mathbb{N},2) \subset \mathfrak{M}(\mathbb{N},3)$, and some general statements. E.g., the Amitsur-Levitski-Theorem [23] implies that the dimension hierarchy is infinite, and investigation of derivation lengths implies that the proof length hierarchy is infinite. Many questions remain open, and we did not even start to investigate this hierarchy for non-strict semirings.

12 Weighted Automata for Derivational Complexity

Recent work of Georg Moser et al. revived the idea of extending termination analysis of rewriting systems towards complexity analysis. Formally, the goal is to bound lengths of derivations by some function of the size of the starting term. Weighted automata methods contribute some results here. The basic idea is that if A is strictly compatible with R, then derivation lengths from t are bounded by the height of $A(t)$ in the order of the semiring.

For standard matrix interpretations, i.e., path-separated $(\mathbb{N},+,\cdot)$-automata, $A(t)$ is bounded by some exponential function of $\mathrm{depth}(t)$, since each matrix computes a linear function. By restricting the shape of the matrices, this bound can be lowered, and one instance is that upper triangular matrices give a polynomially bounded interpretation [29].

When we change the semiring, we get even lower bounds: arctic matrix interpretations imply a linear bound, and $(\mathbb{N},\min,\max)$ interpretations do as well. This holds even without any restrictions on the matrix shapes. In earlier work, the space complexity of computations was bounded by max/plus polynomial quasi-interpretations [1].

One challenge is to lift the "upper triangular" shape restriction for standard matrix interpretations, and still get polynomial bounds. A nice test case is Z_{086}. It is widely believed that this system has quadratic derivational complexity, but it has resisted all attempts to prove this. The interpretation computed by the automaton in Example 1 is not polynomially bounded, as it contains a cycle with weight > 1 (in state 3). Perhaps the arctic automaton in Example 2 helps?

References

[1] Roberto M. Amadio. Synthesis of max-plus quasi-interpretations. *Fundam. Inform.*, 65(1-2):29–60, 2005.

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

[3] Franz Baader, editor. *Term Rewriting and Applications, 18th International Conference, RTA 2007, Paris, France, June 26-28, 2007, Proceedings*, volume 4533 of *Lecture Notes in Computer Science*. Springer, 2007.

[4] Jean Berstel. *Transductions and Context-Free Languages*. Teubner, Stuttgart, 1979.

[5] Frédéric Blanqui, Solange Coupet-Grimal, William Delobel, Sébastien Hinderer, and Adam Koprowski. CoLoR, a Coq library on rewriting and termination. In *Workshop on Termination*, 2006. `http://color.loria.fr`.

[6] Ronald V. Book, Matthias Jantzen, and Celia Wrathall. Monadic thue systems. *Theor. Comput. Sci.*, 19:231–251, 1982.

[7] Ahlem Ben Cherifa and Pierre Lescanne. Termination of rewriting systems by polynomial interpretations and its implementation. *Sci. Comput. Program.*, 9(2):137–159, 1987.

[8] Roel de Vrijer, Jörg Endrullis, and Johannes Waldmann. Local termination. In Ralf Treinen, editor, *Rewriting Techniques and Applications*, 2009.

[9] Manfred Droste, Christian Pech, and Heiko Vogler. A kleene theorem for weighted tree automata. *Theory Comput. Syst.*, 38(1):1–38, 2005.

[10] Manfred Droste and Heiko Vogler. Weighted tree automata and weighted logics. *Theor. Comput. Sci.*, 366(3):228–247, 2006.

[11] Niklas Eén and Armin Biere. Effective preprocessing in sat through variable and clause elimination. In Fahiem Bacchus and Toby Walsh, editors, *SAT*, volume 3569 of *Lecture Notes in Computer Science*, pages 61–75. Springer, 2005.

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

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

[14] Andreas Gebhardt, Dieter Hofbauer, and Johannes Waldmann. Matrix evolutions. In Dieter Hofbauer and Alexander Serebrenik, editors, *Proc. Workshop on Termination, Paris*, 2007.

[15] Andreas Gebhardt and Johannes Waldmann. Weighted automata define a hierarchy of terminating string rewriting systems. In Manfred Droste and Heiko Vogler, editors, *Proc. Weighted Automata Theory and Applications, Dresden*, pages 34–35, 2008.

[16] Alfons Geser, Dieter Hofbauer, and Johannes Waldmann. Match-bounded string rewriting systems. *Appl. Algebra Eng. Commun. Comput.*, 15(3-4):149–171, 2004.

[17] Alfons Geser, Dieter Hofbauer, and Johannes Waldmann. Termination proofs for string rewriting systems via inverse match-bounds. *J. Autom. Reasoning*, 34(4):365–385, 2005.

[18] Alfons Geser, Dieter Hofbauer, Johannes Waldmann, and Hans Zantema. Finding finite automata that certify termination of string rewriting systems. *Int. J. Found. Comput. Sci.*, 16(3):471–486, 2005.

[19] Alfons Geser, Dieter Hofbauer, Johannes Waldmann, and Hans Zantema. On tree automata that certify termination of left-linear term rewriting systems. *Inf. Comput.*, 205(4):512–534, 2007.

[20] Dieter Hofbauer and Johannes Waldmann. Deleting string rewriting systems preserve regularity. *Theor. Comput. Sci.*, 327(3):301–317, 2004.

[21] Dieter Hofbauer and Johannes Waldmann. Termination of string rewriting with matrix interpretations. In Frank Pfenning, editor, *RTA*, volume 4098 of *Lecture Notes in Computer Science*, pages 328–342. Springer, 2006.

[22] Dieter Hofbauer and Johannes Waldmann. Equivalence of match-boundedness, change-boundedness and inverse match-boundedness for length-preserving string rewriting. Theorietag der FG Automaten und Sprachen der GI, Leipzig, 2007. `http://www.imn.htwk-leipzig.de/~waldmann/talk/07/tt/`.

[23] Alexei Kanel-Belov and Louis Halle Rowen. *Computational Aspects of Polynomial Identities*. AK Peters, 2005.

[24] Donald E. Knuth and Peter B. Bendix. Simple word problems in universal algebra. In *Proc. Conf. Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, Oxford, 1967.

[25] Adam Koprowski and Johannes Waldmann. Arctic termination ...below zero. In Andrei Voronkov, editor, *RTA*, volume 5117 of *Lecture Notes in Computer Science*, pages 202–216. Springer, 2008.

[26] Adam Koprowski and Hans Zantema. Certification of proving termination of term rewriting by matrix interpretations. In Viliam Geffert, Juhani Karhumäki, Alberto Bertoni, Bart Preneel, Pavol Návrat, and Mária Bieliková, editors, *SOFSEM*, volume 4910 of *Lecture Notes in Computer Science*, pages 328–339. Springer, 2008.

[27] Martin Korp and Aart Middeldorp. Proving termination of rewrite systems using bounds. In Baader [3], pages 273–287.

[28] Cristopher Moore and David Eppstein. One-dimensional peg solitaire, and duotaire. *CoRR*, math.CO/0008172, 2000.

[29] 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, editors, *FSTTCS*, volume 08004 of *Dagstuhl Seminar Proceedings*. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2008.

[30] Bala Ravikumar. Peg-solitaire, string rewriting systems and finite automata. *Theor. Comput. Sci.*, 321(2-3):383–394, 2004.

[31] Imre Simon. Limited subsets of a free monoid. In *FOCS*, pages 143–150. IEEE, 1978.

[32] Imre Simon. On semigroups of matrices over the tropical semiring. *ITA*, 28(3-4):277–294, 1994.

[33] Johannes Waldmann. Weighted automata for proving termination of string rewriting. *Journal of Automata, Languages and Combinatorics*, 12(4):545–570, 2007.

[34] Hans Zantema and Alfons Geser. A complete characterization of termination of 0^{p} 1^{q}-> 1^{r} 0^{s}. *Appl. Algebra Eng. Commun. Comput.*, 11(1):1–25, 2000.

[35] Hans Zantema and Johannes Waldmann. Termination by quasi-periodic interpretations. In Baader [3], pages 404–418.