Finite Tree Automata

with Cost Functions

Helmut Seidl

Fachbereich Informatik
Universität des Saarlandes
Im Stadtwald
D-6600 Saarbrücken 11
Federal Republic of Germany

ABSTRACT

Cost functions for tree automata are mappings from transitions to (tuples of) polynomials over some semiring. We consider four semirings, namely **N** the semiring of nonnegative integers, **A** the "arctical semiring", **T** the tropical semiring and **F** the semiring of finite subsets of nonnegative integers. We show: for semirings **N** and **A** it is decidable in polynomial time whether or not the costs of accepting computations is bounded; for **F** it is decidable in polynomial time whether or not the *cardinality* of occurring cost sets is bounded. In all three cases we derive explicit upper bounds. For semiring **T** we are able to derive similar results at least in case of polynomials of degree at most 1.
For **N** and **A** we extend our results to multi-dimensional cost functions.

0. Introduction

Finite tree automata are finite state devices which operate on labeled ordered trees. *Cost functions* **c** for tree automata map transitions to (tuples of) polynomials over some semiring **R** such that every computation obtains a *cost* $\mathbf{c}(\phi)$ in **R** (resp. $\mathbf{R}^d$ in the multi-dimensional case). A pair of a finite tree automaton and a cost function is called cost automaton. There are two reasons why we are interested in finite tree automata with cost functions.

First, finite tree automata are an important tool of compiler generating systems like OPTRAN where they are applied for generating code selectors from descriptions of target machine assembly languages [GieSch88, WeiWi88]. A generated tree automaton A is meant to traverse the abstract syntax tree of an input program. The different accepting computations of A correspond to different possibilities of target machine code generation. From these the "cheapest" one is selected according to some suitable cost measure. In fact, these measures are of a type similar to the cost functions we consider here.

The second reason why we are interested in cost automata is of a more theoretical nature. In [CouMo90] Courcelle and Mosbah investigated MS (i.e. *M*onadic *S*econd-Order) evaluations

on graphs and came up with a general method to translate these to evaluations of graph expressions over suitable semirings. Their method allows to derive polynomial algorithms for a huge class of problems on families of graphs like series parallel graphs, graphs definable by contextfree hyperedge replacement grammars, etc.. To be more specific consider, e.g., the family SP(Σ) of *series parallel graphs* with edge labels from Σ. It consists of acyclic digraphs with one source and one sink whose edges are labeled by elements from Σ. SP(Σ) inductively can be defined as follows:

(1) A single directed edge e labeled by some $a \in \Sigma$ is in SP(Σ);

(2) If $G_1, G_2 \in$ SP(Σ) then both the series and the parallel composition of G_1 and G_2 are elements from SP(Σ). The series composition is obtained from G_1 and G_2 by pasting the sink of G_1 with the source of G_2 whereas the parallel composition is obtained by pasting the two sources as well as the two sinks.

Every series parallel graph can be represented by a graph expression, i.e. some tree over the ranked alphabet $\Sigma \cup \{\|, \cdot\}$ where symbols $a \in \Sigma$ represent edges labeled by a, $\cdot$ and $\|$ have rank 2 and denote series and parallel composition respectively.
Let G denote the graph represented by the graph expression g (the binary operators written in infix notation):

$$g = a \,\|\, ((a \cdot d) \,\|\, b) \cdot c)$$

If we are interested in the *number* of paths from the source to the sink of G we use an evaluation of g over the semiring **N** of naturals. Namely, the leaves of the expression tree (i.e. the edges of G) are counted 1; whereas $\|$ and $\cdot$ are viewed as the polynomials $x_1 + x_2$ and $x_1 x_2$. Evaluating g, we obtain 3 as a result.
To compute the *length* of the shortest path from the source to the sink of G we employ the *tropical* semiring **T** where addition is $\sqcap$ and multiplication is +. We attach cost one to the leaves; to $\|$ and $\cdot$ we attach the polynomials $x_1 \sqcap x_2$ and $x_1 + x_2$ respectively. The result is 1.
The maximal *breadth* of G is computed over the "arctical" semiring **A**. Here, addition is $\sqcup$ instead of $\sqcap$ whereas multiplication is again +. As above, cost one is attached to the leaves of the expression, to $\|$ we now attach the polynomial $x_1 + x_2$, and to $\cdot$ the polynomial $x_1 \sqcup x_2$. Thus, evaluation of g yields 3.
Finally, to compute the *set of path lengthes* of G we need semiring **F**. **F** consists of all finite subsets of $\mathbb{N}_0$ where addition is set union and multiplication is addition of numbers extended to sets. We attach cost $\{1\}$ to the leaves of the expression. $\|$ is interpreted as the polynomial $x_1 \cup x_2$, and $\cdot$ as the polynomial $x_1 + x_2$. The resulting set is $\{1, 2, 3\}$.
We are interested in decision problems like this:

- "Is there a global bound to the breadth of graphs described by graph expressions from some tree language L?"
- "Is there a global bound to the length of the shortest path in graphs described by graph expressions from some tree language L?"

Provided the tree language L is the language accepted by some finite tree automaton A the evaluation functions in question turn out to be cost functions for A over a suitable semiring **R**. Thus, decision problems of the above type correspond to boundedness problems for tree automata with cost functions over **R**. In [HaKre89] decidability was proven for the case of cost functions over **N** or **A**. We improve their results by giving *polynomial time* algorithms for

boundedness in these cases. Our algorithms are based on *syntactic* properties which are easy to test but which (at least for "parameter-reduced" automata) precisely characterize boundedness. We furthermore consider semirings T and F. In case of T we are able to decide boundedness - provided the polynomials occurring in the cost function are of degree at most 1. For this, we introduce a new technique to pump up different (even overlapping) subparts of a computation simultaneously. In case of F we show that it can be decided in polynomial time whether or not the *cardinalities* of cost sets of accepting computations are bounded or not.

The paper is organized as follows. In Section 1, we introduce our basic notation concerning trees and tree automata. In Section 2, we introduce semirings, our notation concerning polynomials over semirings and cost functions. We give a general method how to parameter-reduce R-cost automata (Theorem 1). In Section 3, we consider the case of 1-dimensional cost functions for each of the semirings N, A, T and F (Theorems 2, 3, 4 and 5). We derive polynomial time algorithms deciding boundedness for each of these semirings - in case of T however, only provided the polynomials occurring in the cost function have degree at most 1. Also, we provide explicit upper bounds depending only on structural parameters of the automata and cost functions in question. Section 4 extends the results of Section 2 and 3 to multi-dimensional cost functions for the semirings N and A (Theorems 6, 7 and 8). Section 5 concludes with open problems.

1. Basics

A *ranked alphabet* or *signature* is a pair (Σ,ρ) where Σ is a finite alphabet and $\rho: \Sigma \to \mathbb{N}_0$ is a function mapping symbols to their rank. Usually, if ρ is understood we write Σ for short and define $\Sigma_j = \rho^{-1}(j)$. The maximal j such that $\Sigma_j \neq \emptyset$ is called the *rank* of Σ.

T_Σ denotes the free Σ-algebra of (finite ordered Σ-labeled) *trees*, i.e. T_Σ is the smallest set T satisfying (i) $\Sigma_0 \subseteq T$, and (ii) if $a \in \Sigma_m$ and $t_1,...,t_m \in T$, then $a(t_1,...,t_m) \in T$. Note: (i) can be viewed as the subcase of (ii) where m = 0.

The set of *nodes* of t , O(t) is the subset of $\mathbb{N}^*$ defined by $O(t) = \{\varepsilon\} \cup \bigcup_{j=1}^{m} j \cdot O(t_j)$ where $t = a(t_1,...,t_m)$ for some $a \in \Sigma_m$, $m \geq 0$. The *size* |t| of t is defined as the cardinality of O(t). For nodes r, r' we write $r < r'$ to denote that r is a proper prefix of r'. If neither $r < r'$, $r' < r$ nor $r = r$ then we call r and r' *incomparable* and write $r \between r'$.

t defines maps $t(\_): O(t) \to \Sigma$ and $t/\_: O(t) \to T_\Sigma$ mapping the nodes o of t to their labels or the subtree of t with root o, respectively. We have

$$t(o) = \begin{cases} a & \text{if } o=\varepsilon \\ t_j(o') & \text{if } o=j\cdot o' \end{cases} \quad \text{and} \quad t/o = \begin{cases} t & \text{if } o=\varepsilon \\ t_j/o' & \text{if } o=j\cdot o' \end{cases}$$

Let X denote a set of variables of rank 0. Define $T_\Sigma(X) = T_{\Sigma \cup X}$. We use this different notation in order to indicate which variables are to be substituted. (Clearly, $T_\Sigma \subseteq T_\Sigma(X)$.) Assume $t \in T_\Sigma(X)$. t is called X-*proper* iff every $x \in X$ occurs in t exactly once. If $X = \{x\}$ we write x-proper instead of {x}-proper, and if X is understood we skip the prefix X.

Every map $\theta: X \to T_\Sigma(X)$ can be extended to a map $\theta: T_\Sigma(X) \to T_\Sigma(X)$ by $t\theta = a(t_1\theta,...,t_m\theta)$ whenever $t = a(t_1,...,t_m)$. θ is called X-*substitution* or simply substitution if X is understood. If $X = \{x_1,...,x_m\}$ and $x_i\theta = t_i$, we denote $t\theta$ also by $t[t_1,...,t_m]$. Of special importance is the case where the set X of variables which are to be substituted consists of just one element x.

Assume $x\theta = t_2$ and $t_1 \in T_\Sigma(x) = T_\Sigma(\{x\})$. Then we write $t_1\theta = t_1t_2$. The set $T_\Sigma(x)$ is a monoid w.r.t. x-substitution. (The neutral element is x).
For $t \in T_\Sigma(X)$ we define the k-th power t^k of t by $t^1 = t$ and for $k > 1$, $t^k = t\theta$ where $x\theta = t^{k-1}$ for every $x \in X$. Observe that even if t was proper t^k does not need to be proper as well.

A *finite tree automaton* (FTA for short) is a 4-tuple $A = (Q,\Sigma,\delta,Q_F)$ where Q is a finite set of *states*, $Q_F \subseteq Q$ is the set of *final* states, Σ is the signature of *input trees*, and $\delta \subseteq \bigcup_{m\geq 0} Q\times\Sigma_m\times Q^m$ is the set of *transitions* of A; the transitions in $\delta \cap \bigcup_{m\geq 0} \{q\}\times\Sigma_m\times Q^m$ are also called q-*transitions*.
For Sections 1, 2 and 3 let X denote the set of variables $\{x_j \mid j \in \mathbb{N}\}$, and X_k the set of variables $\{x_j \mid j \in [1,k]\}$ for $k \in \mathbb{N}$. Let $t = a(t_1,...,t_m) \in T_\Sigma(X_k)$ and $q, q_1,..., q_k \in Q$. A $(q,q_1...q_k)$-*computation* ϕ of A for t starts at variables x_j in states q_j and consists of $(p_j,q_1...q_k)$-computations of A for the subtrees t_j, $j = 1,...,m$, together with a transition $(q,a,p_1...p_m) \in \delta$ for the root. We write the state at the root to the left of the states at the variable leaves. This convention is chosen in accordance with our prefix notation of trees and the left-to-right order of substitutions. Formally, we represent ϕ as a tree over signature δ and set of variables X_k as follows. If $t = x_j$ and $q = q_j$ then $\phi = x_j$. If $t = a(t_1,...,t_m)$ then $\phi = \tau(\phi_1,...,\phi_m)$ where $\tau = (q,a,p_1...p_m) \in \delta$ for suitable states $p_1,...,p_m \in Q$ and ϕ_j is a $(p_j,q_1...q_k)$-computation for t_j, $j = 1,...,m$.
The set of all $(q,q_1...q_k)$-computations is denoted by $\Phi^{(q,q_1...q_k)}$.
Assume $t \in T_\Sigma(X_k)$ and $t = t_0[t_1,...,t_k]$. Assume ϕ_0 is a $(q,p_1...p_k)$-computation for t_0, and ϕ_i are $(p_i,q_1...q_m)$-computations for t_i , $i = 1,...,k$. Then $\phi_0[\phi_1,...,\phi_k]$ is a $(q,q_1...q_m)$-computation ϕ of A for t. Conversely, if t_0 contains exactly one occurrence of any x_j , $j = 1,...,k$ (i.e. is X_k-proper), then every $(q,q_1...q_m)$-computation ϕ for t uniquely can be decomposed into a $(q,p_1...p_k)$-computation ϕ_0 for t_0, and $(p_i,q_1...q_m)$-computations ϕ_i for t_i (for suitable states p_i) such that $\phi = \phi_0[\phi_1,...,\phi_k]$. ϕ_i is called *subcomputation* of ϕ on t_i.
A (q,ε)-computation is also called q-computation. A q-computation is called *accepting*, iff $q \in Q_F$. $L(A) = \{t \in T_\Sigma \mid$ there is an accepting computation of A for t$\}$ is the *language* accepted by A. The *size* of A, $|A|$, is defined by $|A| = \sum_{(q,a,q_1...q_m)\in\delta} (m+2)$. For estimating the complexity of our algorithms we always assume that the input signature Σ is *fixed*. Only the sets of states and transitions vary. Thus especially, the rank of Σ is viewed as a *constant*.
The algorithms for tree automata we consider mainly run on a data structure called *trace graph*. For FTA $A = (Q,\Sigma,\delta,Q_F)$ the *trace graph* of A is the edge-labeled digraph $G(A) = (V,E)$ where the set of vertices V equals Q, and the set of edges E consists of all triples $(q,<\tau,j>,q')$ where $\tau = (q,a,q_1...q_m)$ is a q-transition in δ with $q_j = q'$.

Call a state $q \in Q$ *useless* if there is no accepting computation in which there occurs a q-transition, and *useful* otherwise. An FTA A is called *reduced* iff A has no useless states. Useless states can be removed without changing the "behavior" of A. We have:

Proposition 1
For every FTA $A = (Q,\Sigma,\delta,Q_F)$ there is an FTA $A_r = (Q_r,\Sigma,\delta_r,Q_{r,F})$ with $L(A) = L(A_r)$ such that

- $Q_r \subseteq Q$, $\delta_r \subseteq \delta$ and $Q_{r,F} \subseteq Q_F$;
- A_r is reduced; and

A_r can be constructed from A in polynomial time. □

2. Cost Automata

In this section we consider polynomials over semirings and introduce cost automata. A (commutative) *semiring* **R** (with 0 and 1) is a 5-tuple $\mathbf{R} = (R,+,\cdot,0,1)$ where $(R,+,0)$ and $(R,\cdot,1)$ are commutative monoids with neutral elements 0 and 1 respectively such that

$$0 \cdot a = 0 \quad \text{and} \quad a \cdot (b + c) = (a \cdot b) + (a \cdot c)$$

for all $a, b, c \in R$. R is also called the *carrier* of **R**. As usual, we also write $r \in \mathbf{R}$ iff r is an element of the carrier of **R**. Especially, we consider the following four semirings.

(1) The *naturals* $\mathbf{N} = (\mathbb{N}_0,+,\cdot,0,1)$ with usual addition and multiplication;

(2) The *arctical semiring* **A** with carrier $\mathbb{N}_0 \cup \{-\infty\}$ where addition is $\sqcup$ and multiplication is + on integers extended by $x + -\infty = -\infty + x = -\infty$;

(3) The *tropical semiring* **T** with carrier $\mathbb{N}_0 \cup \{\infty\}$ where addition is $\sqcap$ and multiplication is + on integers extended by $x + \infty = \infty + x = \infty$;

(4) The semiring **F** whose carrier consists of all finite subsets of nonnegative integers where addition is set union, and multiplication is addition extended to sets i.e. $A + B = \{a+b \mid a \in A, b \in B\}$.

Observe that all four semirings have a natural (in case of **F** partial) ordering which in case of **N**, **T** and **A** is derived from the ordering $-\infty < 0 < 1 < 2 < \ldots < n < \ldots < \infty$. The ordering of **F** is given by set inclusion. For these orderings all finite least upper bounds exist. In **N**, **A** and **T**, we denote the least upper bound of an unbounded set of semiring elements by ∞. For **F**, the least upper bound of a finite number of semiring elements is their union. A least upper bound of an infinite number of elements from **F** is defined as their union as well - although it may no longer be an element of **F**.

The set of polynomials over **R** with variables from X is denoted by R[X]. A *monomial* m is a polynomial of the form $h \cdot x_1^{\varepsilon_1} \cdot \ldots \cdot x_k^{\varepsilon_k}$. h is called the *coefficient*, and $\varepsilon_1 + \ldots + \varepsilon_k$ the *degree* of m. The *size* of m, $|m|$ is defined as $|m| = 1 + \#\{j \mid \varepsilon_j \neq 0\}$. The latter definition refers to the intuition that both every semiring element and every exponent can be stored in one storage cell of a Random Access Machine (with uniform cost measure). For simplicity we assume that such a machine can execute comparisons and semiring operations of two elements $r, r' \in \mathbf{R}$ in one step. Thus, a polynomial algorithm (say, for a Turing machine) is only obtained from a polynomial algorithm for such a RAM if each involved comparison and semiring operation can be executed in polynomial time.

For our purposes we assume that a polynomial $p \in \mathbf{R}[X_m]$ always is given as a sum of monomials where no monomial has a coefficient 0. The degree of p is the maximal degree of a monomial in p whereas we choose the *size* of p as the sum of the sizes of the monomials in p.

Variable x_j *occurs* in p, or p *depends* on x_j iff p contains a monomial $h \cdot x_1^{\varepsilon_1} \cdot \ldots \cdot x_k^{\varepsilon_k}$ with $\varepsilon_j \neq 0$.

As for substitutions, a map $\theta: X \to \mathbf{R}[X]$ can be extended to a map $\mathbf{R}[X] \to \mathbf{R}[X]$ which is denoted by θ as well. For $f \in \mathbf{R}[X]$, $f\theta$ is defined by function composition: $f\theta$ is the function obtained from f by first applying the functions $x\theta$ and then applying f to the results. As for

trees, we call θ a *X-substitution* or substitution, if X is understood. If $X = \{x_1,...,x_m\}$ and $x_i\theta = f_i$ we denote $f\theta$ by $f[f_1,...,f_m]$ and also write $f\ f_1$ for $f[f_1]$.

Fact 0

Assume $R \in \{N, A, F\}$ and p is a polynomial in $R[x] = R[\{x\}]$. Then the following holds:

(1) If $b_1, b_2 \in R$ and $b_1 \leq b_2$ then $p[b_1] \leq p[b_2]$.

(2) Assume x occurs in p and $b \in R$.

- If $R \in \{N, A\}$, then $p(b) \geq b$.
- If $R = F$ then $\#p(b) \geq \#b$. □

Assume $A = (Q,\Sigma,\delta,Q_F)$ is an FTA and $R \in \{N, T, A, F\}$. A R-*cost function* for A is a mapping $c: \delta \to R[X]$ where $c(\tau) \in R[X_m]$ provided $\tau = (q,a,q_1...q_m)$.

$c(\_)$ is extended to computations ϕ in the natural way. If $\phi = x_j$ then $c(\phi) = x_j$. If $\phi = \tau(\phi_1,...,\phi_m)$ for some $\tau \in \delta$ then $c(\phi) = c(\tau)[c(\phi_1),...,c(\phi_m)]$. A R-*cost automaton* M is a pair $M = (A,c)$ where A is the FTA *underlying* M and $c: \delta \to R[X]$ is a R-cost function for A.

The *size* of M consists of the size of A together with the space to represent c. Hence we define $|M| = |A| + \sum_{\tau \in \delta} |c(\tau)|$.

The set of costs of A w.r.t. c, c(A), is defined by

$$c(A) = \{c(\phi) \mid \phi \text{ accepting computation of } A\}$$

In case $R \in \{N, A, N\}$, we are interested in whether the least upper bound of costs $\sqcup M = \sqcup\ c(A)$ is finite. In case $R = F$ we are interested in whether the least upper bound of cost cardinalities $\#M = \sqcup\ \{\#B \mid B \in c(A)\}$ is finite. If so, we say that the costs of M are *bounded*, or shorter: M ist bounded.

We would like to eliminate costs of subcomputations which do not contribute to the final cost. Consider, e.g., semiring N. A subcomputation may not contribute to the final cost if its cost is multiplied by 0. It turns out that we can decide "online" whether or not a given subcomputation has cost 0 (or, if we like, 1). In order to have a machinery general enough to cope with semirings A and F as well we introduce the notion of *faithful* subsets of semirings.

Assume $H: R \to R'$ is a surjective homomorphism of semirings where R' is finite. Subset $E \subseteq R$ is called faithful (via H) iff $E \subseteq \{r \in R \mid H^{-1}(H(r)) = \{r\}\}$.

Examples: For every $m \in \mathbb{N}$,

(1) $[0,m-1]$ is a faithful subset of N;

(2) $\{-\infty\} \cup [0,m-1]$ is a faithful subset of A;

(3) $[0,m-1] \cup \{\infty\}$ is a faithful subset of T; and

(4) $2^{[0,m-1]}$ is a faithful subset of F.

In all these four cases of faithful subsets $E \subseteq R$, the subset E is faithful via some homomorphism $H_m: R \to R_m$ where R_m is a finite semiring obtained from R by an appropriate "truncation".

(1) N_m is the semiring with carrier [0,m] where addition $+_m$ and multiplication $\cdot_m$ are defined by $x +_m y = m \sqcap (x + y)$ and $x \cdot_m y = m \sqcap (x \cdot y)$; such that $H_m(x) = x \sqcap m$.

(2) A_m is the semiring with carrier $\{-\infty\} \cup [0,m]$ ordered by $-\infty < 0 < 1 < ... < m$. As for A, addition is $\sqcup$ and multiplication is $+_m$ defined by $x +_m y = m \sqcap (x + y)$ where "+" here is ordinary addition extended by $-\infty + x = x + -\infty = x$. Again, $H_m(x) = x \sqcap m$.

(3) Analogously to A_m, T_m is the semiring with carrier $[0,m] \cup \{\infty\}$ ordered by $0 < 1 < ... < m < \infty$. Addition is now $\sqcap$ and multiplication is $+_m$ defined by: $x +_m y = m \sqcap (x + y)$ where "+" here is ordinary addition extended by $\infty + x = x + \infty = \infty$. Now, $H_m(x) = \infty$ if $x = \infty$ and $H_m(x) = x \sqcap m$ otherwise.

(4) Finally, F_m is the semiring with carrier $2^{[0,m-1]} \cup \{u\}$ for some new symbol u. Addition in F_m is set union extended by $x \cup u = u \cup x = u$, whereas multiplication $+_m$ is defined by

$$A +_m B = \begin{cases} \{a + b \mid a \in A, b \in B\} & \text{if } a + b < m \text{ for all } a \in A, b \in B \\ u & \text{otherwise} \end{cases}$$

Now, $H_m(B) = B$ if $B \subseteq [0,m-1]$ and $H_m(B) = u$ otherwise.

Since all the semirings R_m are finite we can incorporate the evaluation of cost functions H_m c into the computation of the tree automaton itself. Assume $M = (A,c)$ is a R-cost automaton where A is reduced and $E \subseteq R$ is faithful where $0 \in E$. M is called E-*reduced* iff there are sets $U_r(A,c)$, $r \in E$, such that a q-computation has cost r iff $q \in U_r(A,c)$.
We would like to use information about costs of subcomputations to "simplify" the polynomials involved. We call M E-*parameter-reduced* iff M is E-reduced and the following holds:

(1) For every $r \in E \setminus \{0\}$, $p \in U_r(A,c)$, $\tau = (q,a,q_1...q_k) \in \delta$ and $j \in [1,k]$, $p \neq q_j$.

(2) If $\tau \in \delta$ is a q-transition with $q \in U_r(A,c)$ then $c(\tau) = r$.

(3) If $\tau = (q,a,q_1...q_k) \in \delta$ and x_j does not occur in $c(\tau)$ then $q_j \in U_0(A,c)$.

For our characterizations of bounded costs we only will refer to E-parameter-reduced R-cost automata where $E = \{0, 1\}$. For simplicity, we skip the prefix "E-" in this case.
Before we explain the corresponding results we first convince ourselves that we w.l.o.g. always may assume that the cost automata in question are parameter-reduced. For this we prove Theorem 1:

Theorem 1
Assume $R \in \{N, T, A, F\}$, and $E \subseteq R$ is faithful via H_m: $R \to R_m$ with $0 \in E$.

(1) For every R-cost automaton $M = (A,c)$ there is an E-reduced R-cost automaton $M_E = (A_E,c_E)$ such that

- $L(A) = L(A_E)$ and $c(A) = c_E(A_E)$;
- if A has n states then A_E has at most $(\#R_m) \cdot n$ states and $|M_E| \leq |M| \cdot (\#R_m)^L$;
- M_E can be constructed by a RAM in time polynomial in $|M|$ and $\#R_m$.

(2) For every E-reduced R-cost automaton $M = (A,c)$ there is an E-parameter-reduced R-cost automaton $M_{E,r} = (A_{E,r},c_{E,r})$ such that

- $L(A) = L(A_E)$ and $c(A) = c_{E,r}(A_{E,r})$;
- if A has n states then $A_{E,r}$ has at most $3 \cdot n$ states and $|M_{E,r}| \leq |M| \cdot 3^L$
- $M_{E,r}$ can be constructed by a RAM in polynomial time. □

As a corollary we obtain:

Corollary 1

(1)

- Assume $R \in \{N, T, A\}$. It can be decided in polynomial time for a given R-cost automaton $M = (A,c)$ and $m \in \mathbb{N}$, whether or not $\sqcup M < m$.
- Assume $m \in \mathbb{N}$ is fixed. Then it can be decided in polynomial time for a given F-cost automaton $M = (A,c)$, whether or not $c(A) \subseteq 2^{[0,m-1]}$.

(2) Let $R \in \{N, T, A, F\}$. For every R-cost automaton $M = (A,c)$, a parameter-reduced R-cost automaton $M_r = (A_r,c_r)$ can be constructed in polynomial time with $L(A) = L(A_r)$ and $c(A) = c_r(A_r)$. □

3. Upper Bounds for Bounded Costs

In the next four subsections we successively consider semirings N, A, T and F. In case of the semirings N, A and T (here at least for cost functions of degree at most 1) we compute upper bounds for bounded costs and give polynomial time decision procedures for boundedness. In case of semiring F we derive an upper bound for the cardinality of occurring cost sets provided it is finite and prove that it also can be decided in polynomial time whether or not it is finite.

3.1. The Semiring *N*

In this section we consider costs in the semiring N.

Fact 1

Assume $p \in N[x]$ and $b \notin \{0,1\}$. Then either $p[b] > b$ or $p \in N \cup \{x\}$. □

The N-cost automaton $M = (A,c)$ has Property (N) iff

(N) If $(q,<\tau,j>,q')$ is an edge of a strong component of the trace graph G(A), then $c(\tau) \in \{0,x_j\}$.

Obviously, it can be decided in polynomial time whether or not M has Property (N). We have:

Theorem 2

For a parameter-reduced N-cost automaton $M = (A,c)$ the following three statements are equivalent:

(1) M is bounded;

(2) M has Property (N);

(3) $\sqcup M \leq \begin{cases} [(L+1)\cdot H]^n & \text{if } k = 1 \\ [(L+1)^k \cdot H]^{k^n} & \text{if } k > 1 \end{cases}$

where n is the number of states of A; L is the rank of the input signature; H and k are upper bounds for the coefficients and degrees resp. of polynomials occurring in c.
It can be decided in polynomial time whether or not M is bounded.

Proof: Assume M does not have Property (N). Then there is an accepting computation $\phi = \psi_1\psi_2\psi_3$ for proper (f,q)-computation ψ_1, proper (q,q)-computation ψ_2 and q-computation ψ_3 such that $c(\psi_2) \notin N$ and $c(\psi_2) \neq x_1$. Especially, it contains variable x_1. It follows that $c(\psi_3) \notin \{0,1\}$ since M is parameter-reduced. Hence by Fact 1 for every $k > 1$, $c(\psi_2^k\psi_3) = c(\psi_2)^k c(\psi_3) > k\text{-}1 + c(\psi_3) > k$. Since M is parameter-reduced, $c(\psi_1) \notin N$. Hence by Fact 0 (2), $c(\psi_1\psi_2^k\psi_3) = c(\psi_1)c(\psi_2)^k c(\psi_3) > k$; and the costs of M cannot be bounded. Therefore, (1) implies (2).
Since statement (3) trivially implies (1) it remains to show that (2) implies the upper bounds given in (3). Assume ϕ is an accepting computation of A. Define a *recursive decomposition* D of $\phi = \phi_\varepsilon$ by

$$\phi_o = \psi_o\tau_o(\phi_{o1},\ldots,\phi_{om_o}),\ o \in O(v)\ ,$$

for some $v \in T_\Sigma$ such that for every $o \in O(v)$:

(D1) $\tau_o \in \delta$;

(D2) ψ_o is a proper (q_o,q_o)-computation for some state $q_o \in Q$ which differs from every state $q_{o'}$ where o' is a proper prefix of o.

If M has Property (N) then for all $o \in O(v)$, $c(\psi_o) \in \{0, x_1\}$. Since M is parameter-reduced, this implies that for every $o \in O(v)$,

$$c(\phi_o) = c(\tau_o)[c(\phi_{o1}),\ldots,c(\phi_{om_o})].$$

By (D2), the depth of v is at most n-1. Since every polynomial $c(\tau_o)$ is a sum of monomials $h\cdot x_1^{\varepsilon_1}\cdot\ldots\cdot x_m^{\varepsilon_m}$ with $h \leq H$ and $\sum_{j=1}^{m}\varepsilon_j \leq k$, the upper bounds in statement (3) of the Theorem follow.

□

3.2. The Semiring *A*

Next, we consider costs in the semiring A.

Fact 2
Assume $p \in A[x]$ and $b \notin \{-\infty,0\}$. Then either $p^k[b] > k$ for all $k > 0$ or, p has one of the forms $p = a$ or $p = a \sqcup x$ for some $a \in A$. □

The A-cost automaton $M = (A,c)$ has Property (A) iff

(A) If $(q,\langle\tau,j\rangle,q')$ is an edge of a strong component of the trace graph G(A), then $c(\tau) \in A[X \setminus \{x_j\}]$ or $c(\tau) = p \sqcup x_j$ where $p \in A[X \setminus \{x_j\}]$.

Obviously, it can be decided in polynomial time whether or not M has Property (A). We have:

Theorem 3

Assume M = (A,c) is a parameter-reduced A-cost automaton. The following three statements are equivalent:

(1) M is bounded;

(2) M has Property (A);

(3) $\sqcup M \leq \begin{cases} n \cdot H & \text{if } k = 1 \\ 2 \cdot H \cdot k^n & \text{if } k > 1 \end{cases}$

where n is the number of states of A; L is the rank of the input signature; H and k are upper bounds for the coefficients and degrees resp. of polynomials occurring in c.
It can be decided in polynomial time whether or not M is bounded.

Proof: The proof of implication (1)=>(2) is analogous to the proof of the corresponding implication of the last theorem. Assume M does not have Property (A). Then there is an accepting computation $\phi = \psi_1\psi_2\psi_3$ for proper (f,q)-computation ψ_1, proper (q,q)-computation ψ_2 and q-computation ψ_3 such that $c(\psi_2)$ neither is in A nor in $A \sqcup x_1$. Especially, it contains an occurrence of x_1. Since M is parameter-reduced $c(\phi_3) \notin \{-\infty,0\}$. Hence by Fact 2 for every k > 1, $c(\psi_2^k\psi_3) = c(\psi_2)^k c(\psi_3) > k$. Also since M is parameter-reduced, x_1 occurs in $c(\psi_1)$. Hence by Fact 0 (2), $c(\psi_1\psi_2^k\psi_3) = c(\psi_1)c(\psi_2)^k c(\psi_3) > k$ and the costs of M cannot be bounded. Therefore, (1) implies (2).
Since implication (3)=>(1) is again trivial it only remains to deal with implication (2)=>(3). Assume ϕ is an accepting computation of A. As in the proof of Theorem 2, we consider recursive decompositions of ϕ. By Property (A) there exists such a recursive decomposition of $\phi = \phi_\varepsilon$, $\phi_o = \psi_o\tau_o(\phi_{o1},\ldots,\phi_{om_o})$, $o \in O(v)$, together with monomials m_o in $c(\tau_o)$, $o \in O(v)$, for some tree $v \in T_\Sigma$ of depth at most n-1 such that for all $o \in O(v)$:

$$c(\phi_o) = m_o[c(\phi_{o1}),\ldots,c(\phi_{om_o})]$$

Since every occurring monomial is of the form $m = h + \sum_{j=1}^{L} \varepsilon_j \cdot x_j$ with $h \leq H$ and $\sum_{j=1}^{L} \varepsilon_j \leq k$ we obtain the desired upper bounds. □

3.3. The Semiring *T*

In this subsection we consider costs in the semiring T. So, let M = (A,c) be a T-cost automaton where A = (Q,Σ,δ,Q_F). Here, we do not have such a simple characterization for bounded costs as in the two former cases. We consider only polynomials $p \in T[X]$ of degree at most 1, i.e. polynomials of the form $p = h_0 \sqcap \prod_{j=1}^{m}(h_j + x_j)$ for some $h_j \in T$. The set of these polynomials is denoted by $T^{(1)}[X]$.

Assume ϕ is an accepting computation of A and $c\phi(o) = h_{o,0} \sqcap \prod_{j=1}^{m_o} h_{o,j} + x_j$. The *cost of a node* $o \in O(\phi)$ (in ϕ w.r.t. c) consists of the cost of o itself together with the costs along the path in ϕ from the root of ϕ to r, i.e. $c(o) = h_{o,0} + \sum_{o'j \leq o} h_{o'j}$. We have:

$$c(\phi) = \prod_{o \in O(\phi)} c(o)$$

Clearly, we are only interested in nodes o where $c(o) < \infty$. These nodes are called ∞-*free*.
A node $o \in O(\phi)$ has a *costly cycle* iff $o = o_1o_2o_3$ such that both $\phi(o_1)$ and $\phi(o_1o_2)$ are p-transitions for some $p \in Q$, and there is a prefix o'j of o_2 such that $h_{o_1o'j} > 0$. The main result of this subsection is:

Theorem 4
Assume $M = (A,c)$ is a T-cost automaton where the polynomials in c have degree at most 1. Then the following statements are equivalent:

(1) M is bounded;

(2) For every accepting computation ϕ there is a ∞-free node r which has no costly cycle;

(3) $\sqcup M \leq H \cdot n$

where n is the number of states of A, and H is an upper bound of the coefficients of the polynomials in c.
It is decidable in polynomial time whether or not M is bounded.

Proof: Certainly, (2) implies (3) and (3) implies (1). Also, provided the upper bound (3) holds we can by Corollary 1 decide in polynomial time whether or not M is bounded. Hence, it only remains to prove that (1) implies (2). The idea of the proof is the following. Assume (2) does not hold. Then there exists an accepting computation ϕ such that every minimal ∞-free node contains a costly cycle. We pump up all these cycles simultaneously to obtain an accepting computation of arbitrarily high costs. We interrupt the proof in order to introduce some machinery to treat *simultaneous pumping*. Note that some care has to be taken since pumping in one place may introduce new places where pumping has to be performed. To study this formally, we introduce the notion of *residual decompositions*.

Assume $\phi \in \Phi^{(p,\varepsilon)}$. A decomposition f of ϕ is given by proper computations $\phi_0 \in \Phi^{(p,q)}$, $\psi \in \Phi^{(q,q..q)}$ and $\phi_1,...,\phi_d \in \Phi^{(q,\varepsilon)}$ such that $\phi = \phi_0\psi[\phi_1,...,\phi_d]$.
Equivalently, the decomposition f of ϕ may be described by a pair $\langle r,R\rangle$ where $r \in O(\phi_0)$ is the leaf in u labeled by x_1 and R is the set of leaves of ψ labeled by a variable x_j. r is also called *root* of f.
A node o of ϕ is *factored* by $f = \langle r,R\rangle$, iff $o = rr'o'$ for some $r' \in R$.
Assume $k \geq 0$. Pumping ϕ up k-times w.r.t. f, we obtain computation $\phi^{f,k} = \phi_0\psi^k[\phi_1,...,\phi_d]$.
For ϕ and decomposition $f = \langle r,R\rangle$ of ϕ define r_* as the set of nodes o of ϕ of which r is *not* a prefix; R_0 is the set of proper prefixes of elements in R; R_1 is the set of nodes in ϕ/r which are incomparable to every node in R; and R_2 is the set of nodes o in ϕ/r which have a prefix in R.
Consider as an example $\phi = a(bc, a(a(c, c), c)) = \phi_0\psi[\phi_1,\phi_2]$ where $\phi_0 = a(bc, x_1)$; $\psi = a(a(x_1, c), x_2)$; $\phi_1 = c$ and $\phi_2 = c$. Then this decomposition can be written $f = \langle r,R\rangle$ with $r = 2$ and $R = \{1\cdot1, 2\}$.
We have $r_* = \{\varepsilon, 1, 1\cdot1\}$; $R_0 = \{\varepsilon, 1\}$; $R_1 = \{1\cdot2\}$; and $R_2 = \{1\cdot1, 2\}$.

Using these definitions we can partition the nodes of ϕ as follows:

$$O(\phi) = r_* \cup r\cdot R_0 \cup r\cdot R_1 \cup r\cdot R_2$$

Accordingly, the set of nodes of the k-th pump $\phi^{f,k}$ of ϕ is partitioned:

$$O(\phi^{f,k}) = r_{\aleph} \cup r{\cdot}R^{\leq k-1}{\cdot}R_0 \cup r{\cdot}R^{\leq k-1}{\cdot}R_1 \cup r{\cdot}R^{k}{\cdot}R_2$$

where $R^{\leq k-1} = \bigcup\{R^j \mid j \in [0,k-1]\}$.
Thus, every node $o \in O(\phi)$ gives rise to a set $O_o \subseteq O(\phi^{f,k})$, namely

$$O_o = \begin{cases} o & \text{if } o \in r_{\aleph} \\ rR^{\leq k-1}o_1 & \text{if } o = ro_1 \text{ with } o_1 \in R_0 \\ rR^{\leq k-1}o_1 & \text{if } o = ro_1 \text{ with } o_1 \in R_1 \\ rR^{k-1}o_1 & \text{if } o = ro_1 \text{ with } o_1 \in R_2 \end{cases}$$

such that $O(\phi^{f,k}) = \bigcup\{O_o \mid o \in O(\phi)\}$.
For our application we are not interested in nodes o_1 in O_o which have proper prefixes in O_o. Therefore, we define the set $RES_{f,k}(o)$ of *residuals* of o as the set of *minimal* elements in O_o, i.e.

$$RES_{f,k}(o) = \begin{cases} o & \text{if } o \in r_{\aleph} \\ ro_1 & \text{if } o = ro_1 \text{ with } o_1 \in R_0 \\ rR^{\leq k-1}o_1 & \text{if } o = ro_1 \text{ with } o_1 \in R_1 \\ rR^{k-1}o_1 & \text{if } o = ro_1 \text{ with } o_1 \in R_2 \end{cases}$$

Note that the definition of $RES_{f,k}(o)$ differs from O_o only in the second line. We find:

Proposition 2

(1) Assume $o, o' \in O(\phi)$.
If $o < o'$ then

$$\forall\, o_2 \in RES_{f,k}(o')\ \exists\, o_1 \in RES_{f,k}(o) : o_1 < o_2$$

If $o \aleph o'$ then

$$\forall\, o_2 \in RES_{f,k}(o')\ \forall\, o_1 \in RES_{f,k}(o) : o_1 \aleph o_2$$

(2) If o is ∞-free then $RES_{f,k}(o)$ consists also of ∞-free nodes.

(3) Every minimal ∞-free node of $\phi^{f,k}$ is the residual of some minimal ∞-free node of ϕ. □

Besides decomposition $f = \langle r,R\rangle$ consider a second decomposition $f' = \langle r',R'\rangle$ of ϕ. f' gives rise to a set $RES_{f,k}(f')$ of *residual decompositions*, short: *residuals*. This set of decompositions of $\phi^{f,k}$ is determined in two steps. First, we compute the sets

$$S_0 = RES_{f,k}(r') \text{ and } S_1 = RES_{f,k}(r'R') = \bigcup_{o \in r'R'} RES_{f,k}(o)$$

The set S_0 gives the set of roots of the residual decompositions. By Proposition 2 (1), the nodes in S_1 can be partitioned into sets $S_{1,w}$, $w \in S_0$, where $S_{1,w}$ contains all nodes from S_1 of which w is a prefix. Define $R_w = \{o_1 \mid wo_1 \in S_{1,w}\}$. Finally, put $RES_{f,k}(f') = \{\langle w,R_w\rangle \mid w \in S_0\}$.

We make the following simple observations:

- If r' ⋠ r then $RES_{f,k}(f') = \{f'\}$.
- If r' is a prefix of r then $RES_{f,k}(f') = \{<r',R''>\}$ for some set of nodes R".
- $RES_{f,k}(f) = \{<r,R^k>\}$;
- If $r' = ro_1 \in rR_2$ then $RES_{f,k}(f') = \{<o,R'> \mid o \in rR^{k-1}o_1\}$.

Proposition 3 follows from the above definition and Proposition 2 (1).

Proposition 3
If node o of ϕ is factored by f' then every $\overline{o} \in RES_{f,k}(o)$ is factored by some $g \in RES_{f,k}(f')$. □

Assume c: $\delta \to T^{(1)}[X]$ is a cost function, and $\phi \in \Phi^{(q,\varepsilon)}$. A decomposition $f = <r,R>$ of ϕ has *cost* at least k, iff $\sum_{o<r'} c(\phi(ro)) \geq k$ for every $r' \in R$.
Assume $F = \{f_1,...,f_m\}$ is a set of decompositions f_μ of ϕ. F is called *complete* (for ϕ w.r.t. c) iff every ∞-free node o of ϕ is factored by some decomposition f_μ in F. If furthermore every f_μ has cost at least k, then F is called *k-complete*. Observe that if ϕ admits a k-complete set F of decompositions then $c(\phi) \geq k$. From Propositions 2 and 3 we deduce:

Proposition 4

(1) If F is complete then $RES_{f,k}(F) = \bigcup_{f' \in F} RES_{f,k}(f')$ is complete as well.

(2) If f' has cost at least h then every decomposition in $RES_{f,k}(f')$ also has cost at least h. □

Proposition 5
Assume the accepting computation ϕ has a 1-complete set of decompositions. Then for every $k > 1$, an accepting computation ϕ' exists having a k-complete set of decompositions.

Proposition 5 can be applied to obtain a proof of Theorem 4.

Proof of Theorem 4 (continued): Assume statement (2) does not hold. Then there exists an accepting computation ϕ having a 1-complete set decompositions. By Proposition 5, this implies that for every k, there is a computation ϕ_k having a k-complete set of decompositions. Since $c(\phi_k) \geq k$, M cannot be bounded. □

The rest of this subsection is concerned with a proof of Proposition 5.

Proof of Proposition 5: Assume the 1-complete set of decompositions of ϕ is $F = \{f_1,...,f_m\}$ where r_μ is the root of f_μ. W.l.o.g. we assume that

(*) If $\mu < \mu'$ then r_μ is *not* a prefix of $r_{\mu'}$.

For h = 0,...,m, we inductively define computations ϕ_h with corresponding sets of decompositions F_h.

For h = 0, we set $\phi_0 = \phi$ and $F_0 = F$. $\phi_1 = \phi_0^{f_1,k}$, i.e. ϕ_1 is obtained from ϕ_0 by pumping up ϕ k times w.r.t. f_1. Accordingly, $F_1 = RES_{f_1,k}(F_0)$. By assumption (*), the sets of residual decompositions of $f_2,...,f_m$ consist of just single elements, say $f_{1,2},...,f_{1,m}$ respectively, where the roots of $f_{1,\mu}$ and f_μ coincide. We can proceed with pumping up ϕ_1 k times w.r.t. $f_{1,2}$ to obtain ϕ_2 where $F_2 = RES_{f_{1,2},k}(F_1)$.

In general, assume ϕ_{h-1} and F_{h-1} are already computed and $f_{h-1,h},...,f_{h-1,m}$ are the decompositions in F_{h-1} with roots $r_h,...,r_m$. Then we define $\phi_h = \phi_{h-1}^{f,k}$ and $F_h = RES_{f,k}(F_{h-1})$ with $f = f_{h-1,h}$.

We claim that ϕ_m has a k-complete set of decompositions.

For a proof of this claim call a set F of decompositions of ϕ k-*complete up to* $F' \subseteq F$ iff F is complete and (1) and (2) holds:

(1) Every decomposition $f \in F \setminus F'$ has cost at least k.

(2) Every decomposition $f \in F'$ has cost at least 1.

We prove:

- For h = 0...,m, F_h is k-complete for ϕ_h up to $\{f_{h,h+1},...,f_{h,m}\}$.

Proposition 5 is the special instance of this statement where h = m . It is proved in three steps: For every $h \in \{0,...,m\}$,

- F_h is complete;
- Every decomposition in F_h has cost at least 1;
- Every decomposition in $F_h \setminus \{f_{h,h+1},...,f_{h,m}\}$ has cost at least k.

By assumption, $F_0 = F$ is 1-complete. Therefore, the first two statements follow from Proposition 4 (1) and (2) by induction.

The third statement is also proved by induction on h. For h = 0, it trivially holds. Assume the assertion holds for h-1. Then the inductive step "h-1 $\rightarrow$ h" follows from Proposition 4 (1) and the the fact that if decomposition f has cost at least 1 then the residual decomposition in $RES_{f,k}(f)$ has cost at least k. □

3.4. The Semiring *F*

Finally, we consider costs in the semiring **F**. Again, we would like to concentrate on the form of costs occurring in cycles of the trace graph. However, now there are several possibilities how the cardinality of costs may increase. We have:

Fact 3:

Assume $p \in \mathbf{F}[x]$ and $b \notin \{\emptyset, \{0\}\}$. Then $\#(p^k[b]) > k$ for every $k \in \mathbb{N}$ or $\#(p^k[b]) = \#(p[b])$ and one of the following statements hold:

- $p = a$ or $p = a \cup x$ for some $a \in \mathbf{F}$;
- $p = a + j \cdot x$ where $\#a = 1$ and $j = 1$;
- $p = a + j \cdot x$ where $\#a = 1$, $j > 1$ and $\#b = 1$. □

Assume $M = (A,c)$ is a **F-cost automaton** with $A = (Q,\Sigma,\delta,Q_F)$. For $q \in Q$, define M_q as the F-cost automaton obtained from M by replacing the set Q_F of accepting states with $\{q\}$. M has Property (F) iff

(F) Every strong component Q' of G(A) is of one of the following three types:
Type I: $Q' \subseteq \{q \in Q \mid \#M_q \leq 1\}$.
Type II: For every edge $\langle q,(\tau,j),q_j\rangle$ in Q' with $\tau = (q,a,q_1...q_m)$, $c(\tau) = x_j \cup p$ for some polynomial p in which x_j does not occur and where $\#(c(A_{q_i})) < \infty$ for every x_i occurring in p.
Type III: For every edge $\langle q,(\tau,j),q_j\rangle$ in Q' with $\tau = (q,a,q_1...q_m)$, $c(\tau) = x_j + p$ where $p = H+\sum_{i=1}^{m}\varepsilon_i \cdot x_i$ with $\#H = 1$, $\varepsilon_j = 0$ and $\#M_{q_i} \leq 1$ for every x_i occurring in p.

Opposed to Properties (N) and (A), it is not at all clear that Property (F) can be decided in polynomial time.

Theorem 5
For a parameter-reduced **F-cost automaton** $M = (A,c)$ the following three statements are equivalent:

(1) M is bounded;

(2) M has Property (F);

(3) $\#M \leq \begin{cases}[(L+1)\cdot n\cdot(Hn+1)]^n & \text{if } k = 1\\ [(L+1)^k\cdot 2H\cdot n\cdot k^n]^{k^n} & \text{if } k > 1\end{cases}$

where n is the number of states of A; L is the rank of the input signature; H and k are upper bounds for the cardinalities of the coefficients and degrees resp. of polynomials occurring in c. It can be decided in polynomial time whether or not M is bounded.

Proof: (3) trivially implies (1). To prove that (1) implies (2) we build on Fact 3. Consider for example $\tau = (q,a,q_1...q_m) \in \delta$ and an edge $\langle q,(\tau,j),q_j\rangle$ in a strong component which contradicts *type II*, i.e. for which $c(\tau) = x_j \cup p$ for some polynomial p in which x_j does not occur but in which some x_i occurs with $\#c(A_{q_i}) = \infty$. It follows that we can find a sequence $\psi_1,..,\psi_r,..$ of q_i-computations such that $c(\psi_r)$ contains some $x \geq r$. Since M is parameter-reduced there exist

- a proper (f,q)-computation ϕ_1 for some $f \in Q_F$ such that $c(\phi_1)$ depends on x_1;
- a proper $(q,q_i q)$-computation ϕ_2 such that $c(\phi_2) = p_1 \cup p_2$ where x_1 occurs only in p_1, and x_2 occurs only in p_2;
- a q-computation ϕ_3 such that $c(\phi_3) \notin \{\varnothing, \{0\}\}$.

For every $k > 0$, we construct an accepting computation $\phi^{(k)}$ by iterating ϕ_2 k times and inserting computations ψ_i into the different instances of ϕ_2. Precisely, define $\phi^{(k)} = \phi_1\tilde{\phi}^{(k)}$ where $\tilde{\phi}^{(0)} = \phi_3$ and for $i > 0$, $\tilde{\phi}^{(i)} = \phi_2[\psi_r,\tilde{\phi}^{(i-1)}]$ where r is larger than the maximal element in $p_2c(\tilde{\phi}^{(i-1)})$. Since p_1 depends on x_1, $p_1c(\psi_r)$ also contains an element $x \geq r$ which therefore is not in $p_2(\tilde{\phi}^{(i-1)})$. Hence, we have $\#\, c(\tilde{\phi}^{(0)}) > 0$, and for $i > 0$,

$$\#\, c(\tilde{\phi}^{(i)}) = \#\, c(\phi_2[\psi_r, \tilde{\phi}^{(i-1)}])$$
$$= \#\, [p_1(c(\phi_r)) \cup p_2(c(\tilde{\phi}^{(i-1)}))]$$

$$\geq 1 + \#\, p_2(c(\tilde{\phi}^{(i-1)}))$$

$$\geq 1 + \#\, c(\tilde{\phi}^{(i-1)}) > 1 + (i-1) = i.$$

Since x_1 occurs in $c(\phi_1)$ we conclude that for all $k > 0$,

$$\#\, c(\phi^{(k)}) = \#\, [c(\phi_1)\, c(\tilde{\phi}^{(k)})] > k$$

in contradiction to $\#M < \infty$.

It remains to show that (2) implies the upper bound for #M given by statement (3). For this observe that $\#(c(A_q)) < \infty$ implies that $\#M_q \leq Hn + 1$ in case $k = 1$ and $\#M_q \leq 2 \cdot H \cdot k^n$ otherwise.

Secondly, observe that $\#(A \cup B) \leq \#A + \#B$ and $\#(A + B) \leq \#A \cdot \#B$. These observations allow similar calculations as in the proof of Theorem 2 which gives the desired result.

In order to prove that it can be decided in polynomial time whether or not $\#M < \infty$ it remains to show that Property (F) can be decided in polynomial time. Property (F) in turn can be decided in polynomial time if we succeed to decide in polynomial time for every $q \in Q$, whether or not $\#(c(A_q)) < \infty$ as well as whether or not $\#M_q \leq 1$. This is the contents of subsequent Propositions 6 and 7 respectively. □

Proposition 6

It can be decided in polynomial time for a F-cost automaton M = (A,c) whether or not $\#(c(A)) < \infty$.

Proof: There is a semiring morphism π: $\mathbf{F} \to \mathbf{A}$ given by: $\pi(B) = \bigsqcup_{b \in B} b$. Define $M_\pi = (A, c_\pi)$ as the A-cost automaton where $c_\pi = \pi\, c$. We have:

$\#(c(A)) < \infty$ iff $\#\bigcup\{c(\phi) \mid \phi \text{ accepting}\} < \infty$ iff $\bigsqcup\{\pi\, c(\phi) \mid \phi \text{ accepting}\} < \infty$ iff $\bigsqcup M_\pi < \infty$.

Since the latter can be decided in polynomial time by Theorem 3, we are done. □

Proposition 7

It can be decided in polynomial time for a given parameter-reduced F-cost automaton M, whether or not $\#M \leq 1$.

The method we apply here to prove Proposition 7 is inspired by techniques used in [Sei90] when dealing with single-valued transducers. Especially, we learn from [Sei90] that we may find a simple syntactical characterization of $\#M \leq 1$ provided M is in a special normal form.

A state $q \in Q$ is called *constant* iff $c(\phi) = c(\phi')$ for every two q-computations ϕ and ϕ'. Let Const(A,c) denote the set of all constant states, and define a function c: Const(A,c) $\to$ F by $c(q) = B$ iff $c(\phi) = B$ for some q-computation ϕ. A state $q \in Q$ is called *1-constant* iff q is constant and $\#c(q) \leq 1$. The set of all 1-constant states of A is denoted by $\mathrm{Const}_1(A,c)$. M = (A,c) is called *1-strongly reduced* iff M is parameter-reduced and $\mathrm{Const}_1(M) \cap [Q \setminus Q_F] \subseteq U_\varnothing(M)$. At least the set of 1-constant states can be computed in polynomial time. This gives rise to the following proposition.

Proposition 8

For every parameter-reduced F-cost automaton M = (A,c) exists a 1-strongly reduced F-cost

automaton $M_s = (A_s, c_s)$ such that $L(A) = L(A_s)$ and $c(A) = c_s(A_s)$. M_s can be computed in polynomial time.

Proof: First, we compute the 1-constant states. The algorithm doing so is a special instance of grammar flow analysis as introduced in [MöWi82].
Let $A = (Q,\Sigma,\delta,Q_F)$. W.l.o.g. assume M is already parameter-reduced and $f \in Q_F$ implies $f \neq q_j$ for every $(q,a,q_1,...,q_m) \in \delta$. Assume **R** is the ordered semiring with carrier $\{\bot, ERR, \varnothing\} \cup \{\{i\} \mid i \in \mathbb{N}_0\}$ where the ordering is given by $\bot < \varnothing < ERR$ and $\bot < \{i\} < ERR$ for $i \in \mathbb{N}_0$; addition is $\cup$ defined by $\bot \cup x = x \cup \bot = \bot$; $\varnothing \cup x = x \cup \varnothing = x$; $x \cup x = x$; and $\{x\} \cup \{y\} = ERR$ provided $x \neq y$. Multiplication is defined by: $\varnothing + x = x + \varnothing = \varnothing$; $\bot + x = x + \bot = \bot$ whenever $x \neq \varnothing$; $ERR + x = x + ERR = ERR$ whenever $x \notin \{\varnothing, \bot\}$; $\{x\} + \{y\} = \{x+y\}$. There is a semiring morphism $\tilde{(\_)}$: $\mathbf{F} \to \mathbf{R}$ defined by $\tilde{A} = A$ if $\#A \leq 1$ and $\tilde{A} = ERR$ otherwise.
Consider the trace graph $G(A) = (V,E)$ and define a map $OUT: V \to \mathbf{R}$ as the least upper bound of mappings OUT_i, $i \in \mathbb{N}_0$, where

$$OUT_0(q) = \bot; \text{ and}$$

$$OUT_i(q) = \bigsqcup_{(q,a,q_1\ldots q_m)\in\delta} \tilde{c}(\tau)[OUT_{i-1}(q_1),...,OUT_{i-1}(q_m)] \text{ for } i > 0.$$

We have:

(1) $q \in Q$ is 1-constant iff $OUT(q) \neq ERR$.

(2) $OUT(q) = c(q)$ whenever $q \in Const_1(M)$.

The map $OUT(\_)$ can be computed from M by a RAM in polynomial time which can perform additions and multiplications in $\mathbb{N}_0$ constant time. The values of $OUT(\_)$ different from ERR or $\varnothing$ are bounded by $2 \cdot H \cdot k^n$ where k is the maximal degree of a polynomial $c(\tau)$ and H is the maximal occurring coefficient in $\mathbb{N}_0$. Hence, the lengthes of the values are polynomial in the input size, and the algorithm runs in polynomial time.
Now, define $A_s = A$ and c_s as follows. Assume $\tau = (q,a,q_1\ldots q_m) \in \delta$. If $q \in Q_F \cap Const_1(M)$ then $c_s(\tau) = c(q)$. If $q \in Const_1(M) \setminus Q_F$ then $c_s(q) = \varnothing$. Otherwise, $c_s(\tau) = c(\tau)[s_1,...,s_m]$ where $s_j = \begin{cases} c(q_j) & \text{if } q_j \in Const_1(M) \\ x_j & \text{otherwise} \end{cases}$. □

Proof of Proposition 7: Assume $M = (A,c)$ is parameter-reduced. M has Property (U1) iff

(U1) For every $\tau = (q,a,q_1\ldots q_m) \in \delta$, $c(\tau) = \varnothing$ or $c(\tau) = \{h\} + \sum_{j=1}^{m} \varepsilon_j \cdot x_j$ for some $h \in \mathbb{N}_0$ and $\varepsilon_j \in \mathbb{N}_0$.

Clearly, it can be decided in polynomial time whether or not M has Property (U1). We show:

(*) If M is 1-strongly reduced then $\#M \leq 1$ iff M has Property (U1).

By Proposition 8, we w.l.o.g. may assume that M is 1-strongly reduced. Therefore, statement (*) implies Proposition 7. It remains to prove statement (*). If M has Property (U1) then clearly $\#M \leq 1$. For the converse implication we need the following observation about polynomials in $\mathbb{N}_0[X]$ of degree at most one.

(**) Assume for i = 1, 2, $p_i = h_i + \sum_{j=1}^{m} \varepsilon_{i,j} \cdot x_j$ for some $h_i \in \mathbb{N}_0$ and $\varepsilon_{i,j} \in \mathbb{N}_0$. Furthermore, $a_{1,j} \neq a_{2,j}$ for $j \in [1,m]$. If $p_1[a_{\mu(1),1},...,a_{\mu(m),m}] = p_2[a_{\mu(1),1},...,a_{\mu(m),m}]$ for all $\mu: [1,m] \to \{1, 2\}$ then $p_1 = p_2$.

Assume #M ≤ 1 and all coefficients occurring in the image of δ have cardinality at most 1. Thus (by ignoring set brackets) we can view the occurring monomials as polynomials from N[X] of degree at most 1. For a contradiction assume some $\tau = (q,a,q_1...q_m)$ δ exists such that $c(\tau) = p_1 \cup ... \cup p_r$ for monomials p_i where, e.g., $p_1 \neq p_2$ (as functions). Consider a variable x_j occurring either in p_1 or p_2. Since M is 1-strongly reduced and #M ≤ 1 by assumption there are q_j-computations $\phi_{1,j}$ and $\phi_{2,j}$ such that $c(\phi_{1,j}) \neq c(\phi_{2,j})$. Thus, observation (**) implies that some computation $\phi = \tau(\phi_1,...,\phi_m)$ exists with $p_1[c(\phi_1),...,c(\phi_m)] \neq p_2[c(\phi_1),...,c(\phi_m)]$ and hence, $\#c(\phi) > 1$. By parameter-reducedness of M and Fact 0 (2), this contradicts #M ≤ 1. We conclude that M has Property (U1) whenever #M ≤ 1. □

4. Multi-Dimensional Cost Automata

Although cost functions as considered so far may suffice for a lot of interesting cases (see, e.g., the examples in the introduction), MS-evaluations in general give rise to *multi-dimensional* cost functions [CouMo90]. Therefore in this section, we extend our methods to the multi-dimensional case. We succeed to prove results concerning boundedness at least in case of the semirings **N** and **A**.

For the following, we fix a dimension $d \in \mathbb{N}$. Let now X denote the *doubly indexed* set of variables $\{x_{i,j} \mid i \in \mathbb{N}, j \in [1,d]\}$, and $X_k = \{x_{i,j} \mid i \in [1,k], j \in [1,d]\}$ for every $k \in \mathbb{N}$. Assume $A = (Q,\Sigma,\delta,Q_F)$ is an FTA and **R** denotes a semiring. An d-*dimensional* **R**-*cost function* for A is a mapping $c: \delta \to R[X]^d$ where $c(\tau) \in R[X_k]^d$ provided $\tau = (q,a,q_1...q_k)$.

c(_) is extended to computations φ in the natural way. If $\phi = x_j$ then $c(\phi)_\mu = x_{j,\mu}$ for all $\mu \in [1,d]$. (As usual, we write the μ-th component of c(_) as $c(\_)_\mu$.) If $\phi = \tau(\phi_1,...,\phi_m)$ for some $\tau \in \delta$ then $c(\phi) = c(\tau)[c(\phi_1),...,c(\phi_m)]$ where substitution is extended to tuples in the natural way.

A **R**-*cost automaton* M or dimension d is a pair M = (A,c) where A is the FTA underlying M, and $c: \delta \to R[X]^d$ is an d-dimensional **R**-cost function for A.

The *size* of M again consists of the size of A together with the space to represent c. Hence we define $|M| = |A| + \sum_{\tau \in \delta} |c(\tau)|$.

In case $R \in \{N, T, A, F\}$, we may define the set of R-costs of M, c(A), by

$$c(A) = \{c(\phi)_1 \mid \phi \text{ accepting computation of } A\}$$

Accordingly for $R \in \{N, A, N\}$, $\sqcup M = \sqcup c(A)$ is the least upper bound for R-costs of M. The notions of E-reducedness and E-parameter-reducedness can be extended to d-dimensional cost automata in a straight forward way. Especially, instead of sets of states $U_r(M)$ we need sets of states $U_{<r_1,...,r_d>}(M)$ for every d-tuple $<r_1,...,r_d> \in (E \cup \{\bot\})^d$ where symbol ⊥ denotes a cost ∉ E.

Theorem 6

Assume $R \in \{N, T, A, F\}$ and $E \subseteq R$ is faithful via $H_m: R \to R_m$ with $0 \in E$. Then the following holds:

(1) For every d-dimensional **R**-cost automaton $M = (A,\mathbf{c})$ with n states there is an E-reduced **R**-cost automaton $M_E = (A_E,\mathbf{c}_E)$ such that

- $L(A) = L(A_E)$ and $\mathbf{c}(A) = \mathbf{c}_E(A_E)$;
- A_E has at most $(\#\mathbf{R}_m)^d \cdot n$ states and $|M_E| \leq |M| \cdot (\#\mathbf{R}_m)^{dL}$;
- M_E can be constructed by a RAM in time polynomial in $|M|$ and $\#\mathbf{R}_m$.

(2) For every E-reduced **R**-cost automaton $M = (A,\mathbf{c})$ with n states there is an E-parameter-reduced **R**-cost automaton $M_{E,r} = (A_{E,r},\mathbf{c}_{E,r})$ such that

- $L(A) = L(A_{E,r})$ and $\mathbf{c}$ sub E,r $(A) = \mathbf{c}_{E,r}(A_{E,r})$;
- $A_{E,r}$ has at most $3^d \cdot n$ states and $|M_{E,r}| \leq |M| \cdot 3^{dL}$;
- If d is fixed, $M_{E,r}$ can be constructed by a RAM in polynomial time. □

From Theorem 6 we deduce:

Corollary 2

(1)

- Assume $\mathbf{R} \in \{\mathbf{N}, \mathbf{T}, \mathbf{A}\}$. It can be decided in polynomial time for $m \in \mathbb{N}$ and a given d-dimensional **R**-cost automaton $M = (A,\mathbf{c})$, whether or not $\sqcup M < m$.
- Assume $m \in \mathbb{N}$ is fixed. Then it can be decided in polynomial time for a given d-dimensional **F**-cost automaton $M = (A,\mathbf{c})$, whether or not $\mathbf{c}(A) \subseteq 2^{[0,m-1]}$.

(2) Let $\mathbf{R} \in \{\mathbf{N}, \mathbf{T}, \mathbf{A}, \mathbf{F}\}$. For every **R**-cost automaton $M = (A,\mathbf{c})$, a parameter-reduced **R**-cost automaton $M_r = (A_r,\mathbf{c}_r)$ can be constructed in polynomial time with $L(A) = L(A_r)$ and $\mathbf{c}(A) = \mathbf{c}_r(A_r)$. □

We extend Property (N) of Section 3 to the multi-dimensional case.

The d-dimensional **N**-cost automaton $M = (A,\mathbf{c})$ has Property (N) iff

(N) If $(q,<\tau,j>,q')$ is an edge of a strong component of the trace graph G(A), then either $\mathbf{c}(\tau)_\mu = 0$ or $\mathbf{c}(\tau)_\mu = x_{j,\nu}$ for some $\nu \in [1,d]$.

Obviously, it can be decided in polynomial time whether or not M has Property (N). We have:

Theorem 7

For a parameter-reduced **N**-cost automaton $M = (A,\mathbf{c})$ of dimension d the following three statements are equivalent:

(1) M is bounded;

(2) M has Property (N);

(3) $$\sqcup M \leq \begin{cases} [(L+1)\cdot H]^n & \text{if } k = 1 \\ [(L+1)^k \cdot H]^{k^n} & \text{if } k > 1 \end{cases}$$

where n is the number of states of A; L is the rank of the input signature; H and k are upper bounds to the coefficients and degrees resp. of polynomials occurring in **c**.

It can be decided in polynomial time whether or not M is bounded. □

Note that the characterization given in Theorem 7 can be viewed as a stronger version of the non-ramification lemma as described in [Sei89, Sei91] which gives a characterization of finitely ambiguous FTAs.

A syntactical characterization of bounded costs is also possible for parameter-reduced d-dimensional A-cost automata.

The d-dimensional A-cost automaton M = (A,c) has Property (A) iff

(A) If $(q,\langle\tau,j\rangle,q')$ is an edge of a strong component of the trace graph G(A), then for every $i \in [1,d]$, $c(\tau)_i$ has the form $p \sqcup \bigsqcup_{\mu\in J} x_{j,\mu}$ for some $J \subseteq [1,d]$ where p does not contain variables $x_{j,\nu}$ for any ν.

Clearly, it can be decided in polynomial time whether or not M has Property (A). We have:

Theorem 8

For a parameter-reduced A-cost automaton M = (A,c) of dimension d the following three statements are equivalent:

(1) M is bounded;

(2) M has Property (A);

(3) $\sqcup M \le \begin{cases} n\cdot H & \text{if } k = 1 \\ 2\cdot H\cdot k^n & \text{if } k > 1 \end{cases}$

where n is the number of states of A; L is the rank of the input signature; H and k are upper bounds to the coefficients and degrees resp. of polynomials occurring in c.
It can be decided in polynomial time whether or not M is bounded. □

Theorems 7 and 8 give efficient versions of the result of [HaKre89] proving that boundedness of costs is decidable. Note that Habel et al. prove a somewhat stronger result by allowing not only polynomials over N or A but also polynomials where all three operations ⊔, + and · occur. In fact, our methods allow to derive a polynomial decision procedure for boundedness of costs also in this slightly more general situation. It was for descriptional clearity only that we omitted to include a corresponding result.

5. Conclusion

We considered cost automata with cost in several semirings, derived upper bounds for bounded costs and gave polynomial time algorithms to decide boundedness. Not for all semirings in {N, T, A, F} we could prove results in full generality. Especially, it remains open whether an upper bound can be derived for costs in T when either the degree of the occurring polynomials or the dimension is > 1. Also, it is rather unclear how our result on boundedness in F can be extended to the multi-dimensional case.

References

[CouMo90] B. Courcelle, M. Mosbah: Monadic second-order evaluations on tree-decomposable graphs. Tech. Report LaBRI No. 90-110, Bordeaux, 1990

[GeSt84] F. Gecseg, M. Steinby: Tree automata. Akademiai Kiado, Budapest, 1984

[GiSch88] R. Giegerich, K. Schmal: Code selection techniques: pattern matching, tree parsing and inversion of derivors. Proc. of ESOP 1988, LNCS 300 pp. 245-268

[HaKre89] A. Habel, H.-J. Kreowski, W. Vogler: Decidable boundedness problems for hyperedge-replacement grammars. Proc. TAPSOFT '89 vol. 1, LNCS 352, pp. 275-289; long version to appear in TCS

[MöWi82] U.Möncke, R. Wilhelm: Iterative algorithms on grammar graphs. Proc. 8th Conf. on Graphtheoretic Concepts in Computer Science, Hanser Verlag 1982, pp. 177-194

[Paul78] W. Paul: Komplexitätstheorie. B.G. Teubner Verlag Stuttgart 1978

[Sei89] H. Seidl: On the finite degree of ambiguity of finite tree automata. Acta Inf. 26, pp. 527-542 (1989)

[Sei90] H. Seidl: Single-valuedness of tree transducers is decidable in polynomial time. To appear in: TCS, special issue of CAAP 90

[Sei91] H. Seidl: Ambiguity and Valuedness. To appear in: M. Nivat, A. Podelski (eds.): "Definability and Recognizability of Sets of Trees". Elsevier Amsterdam

[WeiWi88] Two tree pattern matchers for code selection. Proc. 2nd CCHSC Workshop 1988, LNCS 371, pp. 215-229