Branching Automata with Costs – A Way of Reflecting Parallelism in Costs

Dietrich Kuske and Ingmar Meinecke

Institut für Algebra, Technische Universität Dresden
D-01062 Dresden, Germany
{kuske,meinecke}@math.tu-dresden.de

Abstract. Extending work by Lodaya and Weil, we propose a model of *branching automata with costs* in which the calculation of the cost of a parallel composition is handled differently from the calculation of the cost of a sequential composition. Our main result characterizes the behavior of these automata in the spirit of Kleene's and Schützenberger's theorems.
The technical report [12] that this extended abstract is based on contains complete proofs and can be accessed at the net.

1 Introduction

This paper reports on our research into parallel systems with costs in the setting of series-parallel posets. One of its roots is the line of research initiated by Grabowski [7] and Gischer [6]. They extended previous ideas by Kleene on sequential systems build by nondeterministic choice, iteration and sequential composition. Gischer proposed, in order to model parallel systems, in addition a parallel composition. It turned out that series-parallel posets are ideally suited to describe executions of such systems. Later, Lodaya and Weil [14, 13] proposed a finite-state device capable of accepting series-parallel posets. These automata model parallelism by branching - hence the name "branching automata".

Suppose we wanted to calculate the minimal duration of a run in a modularly constructed system. The execution time of a sequential composition is the sum of the durations, and that of a parallel composition is the maximum of the durations of the arguments, possibly increased by some duration for the necessary fork and join operations at the beginning and end. A given series-parallel poset can be executed in different ways and we should consider the minimal duration of all possible executions. In order to accompany this situation, we introduce bisemirings, *i.e.*, structures consisting of two semirings on a joint domain with the same additive operation. Costs of executions in our model of branching automata with costs are then evaluated in such bisemirings and the behavior of a branching automaton with costs is a function that associates with any series-parallel poset an element from the bisemiring.

It is the aim of this paper to characterize those functions that are associated with branching automata with costs. For this, we employ and extend the machinery from the theory of weighted automata (see [16, 10, 2, 9] for expositions).

O.H. Ibarra and Z. Dang (Eds.): CIAA 2003, LNCS 2759, pp. 150–162, 2003.

In this field, one starts from a nondeterministic finite (word or tree) automaton and provides its transitions with weights, costs or multiplicities (depending on the community). M. Droste raised the question whether branching automata can be provided with costs in a semiring. Our conceptional contribution in this respect is the observation that one should not just consider cost structures with *one* multiplication, but that several multiplications are necessary to model the phenomena of parallelism.

We characterize the behavior of branching automata with costs in the style of theorems by Kleene [8] and Schützenberger [17] stating the equivalence of regularity and rationality. Several related results are known: for trees, there is a wealth of results of different generality [1, 9, 5, 15]; for Mazurkiewicz traces, Droste and Gastin proved a theorem á la Schützenberger [3]; and for infinite words, Droste and Kuske showed a result in the spirit of the theorems by Büchi and by Schützenberger [4]. When Lodaya and Weil considered languages accepted by branching automata, they observed that unbounded parallelism cannot be captured completely by rational operations, their main results hold for languages of bounded width [13]. Since, in a parallel system, the width corresponds to the number of independent processes, this boundedness restriction seems natural to us. Therefore and similarly, our characterization holds for branching automata with costs only that generate functions with support of bounded width. For this class, we get as our main result the equivalence of acceptance by branching automata with costs and rationality (see Theorem 5.2).

2 Basic Definitions

Let Σ be a finite alphabet. A *Σ-labeled poset* $(V, \leq, \tau)$ is a finite poset[1] $(V, \leq)$ equipped with a labeling function $\tau : V \longrightarrow \Sigma$. The *width* $\mathrm{wd}(t)$ of t is the maximal size of a subset of V whose elements are mutually incomparable.

The *sequential product* $t_1 \cdot t_2$ of $t_1 = (V_1, \leq_1, \lambda_1)$ and t_2 is the Σ-labeled poset

$$(V_1 \dot\cup V_2, \leq_1 \cup (V_1 \times V_2) \cup \leq_2, \tau_1 \cup \tau_2).$$

Graphically, t_2 is put on top of t_1. The *parallel product* $t_1 \| t_2$ is defined as $(V_1 \dot\cup V_2, \leq_1 \cup \leq_2, \tau_1 \cup \tau_2)$, *i.e.*, the two partial orders are put side by side. SP denotes the least class of Σ-labeled posets containing all labeled singletons and closed under the application of the sequential and the parallel product, its elements are *series-parallel posets* or *sp-posets* for short. We say that t is *sequential* if it cannot be written as a parallel product $t = u \| v$. Dually, t is called *parallel* if it cannot be written as a sequential product $t = u \cdot v$ of $u, v \in \mathrm{SP}$. The only sp-posets which are both sequential and parallel are the singleton posets that we identify with the elements of Σ. By Gischer [6], every $t \in \mathrm{SP}$ admits a maximal *parallel factorization* $t = t_1 \| \ldots \| t_n$ (which is unique up to commutativity) where $n \geq 1$ and each $t_i \in \mathrm{SP}$ $(i = 1, \ldots, n)$ is sequential, and a unique maximal *sequential decomposition* $t = t'_1 \cdot \ldots \cdot t'_m$ where $m \geq 1$ and each $t'_i \in \mathrm{SP}$ $(i = 1, \ldots, m)$

[1] In this paper, we consider even only nonempty posets.

is parallel. Hence, SP is freely generated by Σ subject to associativity of both operations and commutativity of the parallel product.

Definition 2.1. *A* bisemiring $\mathbb{K} = (K, \oplus, \circ, \diamond, 0, 1)$ *is a set* K *equipped with three binary operations called addition* $\oplus$*, sequential multiplication* $\circ$ *and parallel multiplication* $\diamond$ *such that:*

1. $(K, \oplus, 0)$ *is a commutative monoid,* $(K, \circ, 1)$ *a monoid, and* $(K, \diamond)$ *a commutative semigroup,*
2. *both* $\circ$ *and* $\diamond$ *distribute over* $\oplus$*, and*
3. 0 *is absorbing for* $\circ$ *and* $\diamond$*, i.e.,* $k \circ 0 = 0 \circ k = k \diamond 0 = 0$ *for all* $k \in K$*.*

The structure $(K, \oplus, \circ, 0, 1)$ is a semiring. Moreover, $(K, \oplus, \diamond, 0)$ is almost a semiring; only the parallel multiplication does not have to admit a unit. Any commutative semiring can be seen as a bisemiring by identifying sequential and parallel multiplication. Under these trivial bisemirings we only mention the Boolean bisemiring $\mathbb{B} = (\{0, 1\}, \vee, \wedge, \wedge, 0, 1)$.

Example 2.2. The structure $(\mathbb{R} \cup \{+\infty\}, \min, +, \max, +\infty, 0)$ is a bisemiring that we referred to in the introduction. Here, 0 is the unit for the sequential multiplication $+$ and $+\infty$ is the absorbing zero of the bisemiring.

Let $a \in \Sigma$. We interpret a as some action and assume a has a duration of $\mathrm{time}(a)$. Let $\mathrm{time}(a) = +\infty$ if a cannot be performed. For any $t = t_1 \cdot \ldots \cdot t_n \in \mathrm{SP}$ we put $\mathrm{time}(t) = \mathrm{time}(t_1) + \ldots + \mathrm{time}(t_n)$, and for $t = t_1 \| \ldots \| t_m \in \mathrm{SP}$ we put $\mathrm{time}(t) = \max\{\mathrm{time}(t_1), \ldots, \mathrm{time}(t_m)\}$. Hence, $\mathrm{time} : (\mathrm{SP}, \cdot, \|) \to (\mathbb{R} \cup \{+\infty\}, +, \max)$ is a homomorphism and can be interpreted as the duration time of an sp-poset t. In Example 3.2, we will present an automaton that computes the minimal execution time of an sp-poset using the semiring $(\mathbb{R} \cup \{+\infty\}, \min, +, \max, +\infty, 0)$.

Example 2.3. Let Σ be a fixed finite alphabet and let $\mathrm{SP}^1 = \mathrm{SP} \,\dot\cup\, \{\varepsilon\}$ where ε acts as unit w.r.t. $\cdot$ and $\|$. Then the class of sp-languages $(\mathfrak{P}(\mathrm{SP}^1), \cup, \cdot, \|, \emptyset, \{\varepsilon\})$ is a bisemiring. Here the multiplications $\cdot$ and $\|$ are defined elementwise.

3 Branching Automata with Costs

In this section we introduce a model of automata generalizing the concept of branching automata by Lodaya and Weil [13] and their behavior. We fix an alphabet Σ and a bisemiring $\mathbb{K}$. By $\mathfrak{P}_2(Q)$ we denote the collection of subsets of Q of cardinality 2.

Definition 3.1. *A* branching automaton with costs *from* $\mathbb{K}$ *over the alphabet* Σ*, or a* BRAC *for short, is a tuple* $\mathcal{A} = (Q, T_{\mathrm{seq}}, T_{\mathrm{fork}}, T_{\mathrm{join}}, \lambda, \gamma)$ *where*

- Q *is a finite set of* states*,*
- $T_{\mathrm{seq}} : Q \times \Sigma \times Q \longrightarrow \mathbb{K}$ *is the* sequential transition function*,*
- $T_{\mathrm{fork}} : Q \times \mathfrak{P}_2(Q) \longrightarrow \mathbb{K}$ *is the* fork transition function*,*
- $T_{\mathrm{join}} : \mathfrak{P}_2(Q) \times Q \longrightarrow \mathbb{K}$ *is the* join transition function*,*

– $\lambda, \gamma : Q \longrightarrow \mathbb{K}$ *are the* initial *and the* final cost function*, respectively.*

We write $p \xrightarrow{a}_k q$ if $T_{\text{seq}}(p, a, q) = k \neq 0$ and call it a *sequential transition*; if it only matters that the costs are distinct from 0, we write $p \xrightarrow{a} q$. Similarly, we write $p \to_k \{p_1, p_2\}$ and $p \to \{p_1, p_2\}$ if $T_{\text{fork}}(p, \{p_1, p_2\}) = k \neq 0$. In the same way, we understand $\{q_1, q_2\} \to_k q$ and $\{q_1, q_2\} \to q$. A state $q \in Q$ is an *initial state* if $\lambda(q) \neq 0$. Dually, q is a *final state* if $\gamma(q) \neq 0$.

3.1 The Behavior of Branching Automata

In order to calculate the cost of an sp-poset $t \in \text{SP}$ in a BRAC $\mathcal{A}$ we introduce the notion of a *path* in $\mathcal{A}$. We consider labeled directed graphs $G = (V, E, \nu, \eta)$ with a unique source $\text{src}(G)$ and a unique sink $\text{sk}(G)$ where $\nu : V \longrightarrow Q$ is a total and $\eta : E \rightarrow\!\!\!\!\!\rightarrow \Sigma$ is a partial function. The labeled graph G with $V = \{v_1, v_2\}$, $E = \{(v_1, v_2)\}$, $\nu(v_i) = p_i$ $(i = 1, 2)$ and $\eta(v_1, v_2) = a$ is an *atomic* path if $p_1 \xrightarrow{a} p_2$ is a sequential transition of $\mathcal{A}$. Now let $G_i = (V_i, E_i, \nu_i, \eta_i)$ be paths for $i = 1, 2$. If $\nu_1(\text{sk}(G_1)) = \nu_2(\text{src}(G_2))$, we define the *sequential product* $G = G_1 \cdot G_2$ of G_1 and G_2. At that, G is the union of both graphs where the sink of G_1 and the source of G_2 (which have the same label) are fused to one vertex. If $p \to \{\nu_1(\text{src}(G_1)), \nu_2(\text{src}(G_2))\}$ is a fork and $\{\nu_1(\text{sk}(G_1)), \nu_2(\text{sk}(G_2))\} \to q$ a join transition of $\mathcal{A}$ then the *p-q-parallel product* $G_1 \|_{p,q} G_2 = (V, E, \nu, \eta)$ is a path with vertices $V = V_1 \,\dot\cup\, V_2 \,\dot\cup\, \{u, w\}$ and edges $E = E_1 \,\dot\cup\, E_2 \,\dot\cup\, \{(u, \text{src}(G_i)), (\text{sk}(G_i), w) \mid i = 1, 2\}$. For $v \in V_i$ we put $\nu(v) = \nu_i(v)$ $(i = 1, 2)$, furthermore $\nu(u) = p$ and $\nu(w) = q$, and $\eta = \eta_1 \,\dot\cup\, \eta_2$. The sequential product is associative, and every *p-q*-parallel product is commutative, but not associative. The *set* $\text{PT}(\mathcal{A})$ *of paths* of the BRAC $\mathcal{A}$ is the smallest set of labeled directed graphs G that contains all atomic paths of $\mathcal{A}$ and is closed under the sequential product and under all *p-q*-parallel products with $p, q \in Q$ as defined above.

Inductively, we define the label $\text{lab}(G) \in \text{SP}$ and the cost $\text{cost}(G) \in \mathbb{K}$ for any path G. For an atomic path $G : p \xrightarrow{a} q$, we put $\text{lab}(G) = a$ and $\text{cost}(G) = T_{\text{seq}}(p, a, q)$. Further, $\text{lab}(G_1 \cdot G_2) = \text{lab}(G_1) \cdot \text{lab}(G_2)$, $\text{cost}(G_1 \cdot G_2) = \text{cost}(G_1) \circ \text{cost}(G_2)$, and $\text{lab}(G_1 \|_{p,q} G_2) = \text{lab}(G_1) \| \text{lab}(G_2)$. To define the cost of $G = G_1 \|_{p,q} G_2$, let $p_i = \nu_i(\text{src}(G_i))$ and $q_i = \nu_i(\text{sk}(G_i))$ for $i = 1, 2$. Then

$$\text{cost}(G) = T_{\text{fork}}(p, \{p_1, p_2\}) \circ \Big[\text{cost}(G_1) \diamond \text{cost}(G_2)\Big] \circ T_{\text{join}}(\{q_1, q_2\}, q) \ .$$

The cost of such a parallel path can be interpreted as follows. At first we have a cost for branching the process, then the cost for the two subprocesses and finally the cost for joining the subprocesses. These costs come up one after the other and, therefore, are multiplied sequentially. On the other hand, the costs of the two subprocesses are multiplied in parallel.

We denote by $G : p \xrightarrow{t} q$ that G is a path from state p to state q with label $t \in \text{SP}$. Then the cost of some $t \in \text{SP}$ from p to q in $\mathcal{A}$ is given by summing up

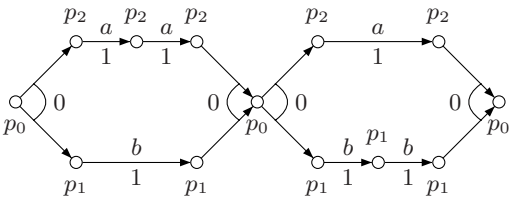

Fig. 1. A path measuring the height

the costs of all possible paths from p to q with label t:

$$\mathrm{cost}_{p,q}(t) = \bigoplus_{G:p\xrightarrow{t}q} \mathrm{cost}(G) \ .$$

The cost of $t \in \mathrm{SP}$ in $\mathcal{A}$ is defined as

$$(\mathcal{S}(\mathcal{A}), t) = \mathrm{cost}_{\mathcal{A}}(t) = \bigoplus_{p,q\in Q} \lambda(p) \circ \mathrm{cost}_{p,q}(t) \circ \gamma(q) \ .$$

Then $\mathcal{S}(\mathcal{A}) : \mathrm{SP} \longrightarrow \mathbb{K}$ is the *behavior of* $\mathcal{A}$ or, equivalently, is *recognized* by $\mathcal{A}$. A function $S : \mathrm{SP} \longrightarrow \mathbb{K}$ is *regular* if there is a BRAC $\mathcal{A}$ such that $S = \mathcal{S}(\mathcal{A})$.

Example 3.2. In this example, we define a branching automaton with costs $\mathcal{A}$ whose behavior measures the height of a pomset, *i.e.*, $(\mathcal{S}(\mathcal{A}), t) = \mathrm{height}(t)$ for any sp-pomset t. For this to work, we use the bisemiring $(\mathbb{R}\cup\{+\infty\}, \min, +, \max)$ from Example 2.2. The automaton has just three states p_0, p_1, p_2. Any of these states can fork into the other two at cost 0; similarly, any two distinct of these states can be joined into the remaining one at cost 0. In any state, we can execute any action at cost 1 (without changing the state). Figure 1 depicts a run of this BRAC on the sp-poset $t = (aa\|b)(a\|bb)$. In that picture, join- and fork-transitions are indicated by a semi-circle between the edges involved. Next to these semi-circles, we denote the cost of the corresponding transition. The path is the sequential product of two "bubbles" whose costs we calculate first. The first bubble is the parallel product of an atomic b-transition and the sequential aa-path. Since the join- and fork-transitions involved in this product have cost 0, the cost of a bubble is $0+\max(1+1, 1)+0 = 2$. Since this holds for both bubbles, the total cost is $2 + 2 = 4$ which equals the height of the poset $(aa\|b)(a\|bb)$.

If any action is executed in one unit of time, then the height of the poset measures the execution time of the sp-poset. This example can be refined in such a way that atomic actions can have different execution times that can even depend on the state they are executed in. Using the bisemiring $(\mathbb{R} \cup \{-\infty, +\infty\}, \min, \max, +)$, the same BRAC as above computes the width of a poset (*i.e.*, the maximal number of parallel sub-processes). In [12], we give another example of a BRAC, this time over the bisemiring of subsets of Σ^*. It calculates, from an sp-poset t, the set of words that label a maximal linearly ordered subset of t.

3.2 Formal Power Series over SP-Posets

To characterize the possible behavior of branching automata with costs, we introduce the notion of formal power series over sp-posets with values in a bisemiring. This concept is both a generalization of the well known formal power series over words (*cf.* [16]) and the sp-languages as introduced by Lodaya and Weil [13].

A *formal power series over* SP *with values in the bisemiring* $\mathbb{K}$ or *sp-series* is a function $S : \mathrm{SP} \longrightarrow \mathbb{K}$. With $(S,t) := S(t)$, it is written as a formal sum:

$$S = \sum_{t \in \mathrm{SP}} (S,t)t.$$

The *support* of S is $\operatorname{supp} S := \{t \in \mathrm{SP} \mid (S,t) \neq 0\}$. Formal power series with support a singleton are called *monomials*. The class of all formal power series over SP with values in $\mathbb{K}$ is denoted by $\mathbb{K}\langle\langle \mathrm{SP} \rangle\rangle$.

Now we introduce some operations for sp-series. Let $S, T \in \mathbb{K}\langle\langle \mathrm{SP} \rangle\rangle$. We define:

1. the *sum* $S + T$ by $(S+T,t) := (S,t) \oplus (T,t)$,
2. the *scalar products* $k \cdot S$ and $S \cdot k$ for $k \in \mathbb{K}$ by $(k \cdot S, t) := k \circ (S,t)$ and $(S \cdot k, t) := (S,t) \circ k$,
3. the *sequential product* $S \cdot T$ by $(S \cdot T, t) := \bigoplus_{t = u \cdot v} (S,u) \circ (T,v)$ where the sum is taken over all sequential factorizations $t = u \cdot v$ with $u, v \in \mathrm{SP}$,
4. the *parallel product* $S \| T$ by $(S \| T, t) := \bigoplus_{(u,v)\,:\, t = u \| v} (S,u) \diamond (T,v)$ where we add over all pairs (u,v) such that $t = u \| v$ with $u, v \in \mathrm{SP}$. Because of the commutativity of $\|$ in SP, both $(S,u) \diamond (T,v)$ and $(S,v) \diamond (T,u)$ contribute to the sum.
5. the *sequential iteration* S^+ of an sp-series S by

$$(S^+, t) := \bigoplus_{1 \leq n \leq |t|} \; \bigoplus_{t = u_1 \cdot \ldots \cdot u_n} (S,u_1) \circ \ldots \circ (S,u_n)$$

where we sum up over all possible sequential factorizations of t.

Collectively, we refer to these operations as the *series-rational operations of* $\mathbb{K}\langle\langle \mathrm{SP} \rangle\rangle$.

Similarly to the sequential iteration, one can define the parallel iteration. Already in the theory of sp-languages, this parallel iteration causes severe problems [14]. Smoother results are obtained if one does not allow the parallel iteration in rational expressions [13, 11].

The operations $+$, $\cdot$, and $\|$ are associative on $\mathbb{K}\langle\langle \mathrm{SP} \rangle\rangle$, and $+$ and $\|$ are even commutative. The series $\mathbf{0}$ with $(\mathbf{0}, t) = 0$ for all $t \in \mathrm{SP}$ is the unit w.r.t. $+$ and absorbing w.r.t. $\cdot$ and $\|$.

The class $\mathbb{K}^{\mathrm{s-rat}}\langle\langle \mathrm{SP} \rangle\rangle$ of *series-rational sp-series* over Σ with values in $\mathbb{K}$ is the smallest class containing all monomials that is closed under the series-rational operations of $\mathbb{K}\langle\langle \mathrm{SP} \rangle\rangle$.

Let $\mathbb{K}$ and $\mathbb{K}'$ be bisemirings, $h : \mathbb{K} \to \mathbb{K}'$ a bisemiring homomorphism, and $f : \mathrm{SP} \to \mathrm{SP}$ a function that commutes with $\cdot$ and $\|$. Further, let S be an sp-series over $\mathbb{K}$. For $t \in \mathrm{SP}$, define $(\overline{h}(S), t) = h(S, t)$ and $(\overleftarrow{f}(S), t) = \bigoplus_{f(s)=t}(S, s)$.

Proposition 3.3. *The functions $\overline{h}$ and $\overleftarrow{f}$ commute with the series-rational operations. In particular, they preserve series-rationality.*

We consider as a special case the Boolean bisemiring $\mathbb{B}$. An *sp-language* L is a subset of SP. Any sp-language $L \subseteq \mathrm{SP}$ can be identified with its *characteristic series* $\mathbf{1}_L$ where $\operatorname{supp} \mathbf{1}_L = L$. This isomorphism maps the class $\mathbb{B}^{\mathrm{s-rat}}\langle\langle \mathrm{SP} \rangle\rangle$ to the class of *series-rational sp-languages* $\mathrm{SP}^{\mathrm{s-rat}}$ (*cf.* [13] for the definition). Therefore, the theory of sp-series is a generalization of the theory of sp-languages as investigated by Lodaya and Weil [13].

An sp-language $L \subseteq \mathrm{SP}$ has *bounded width* if there exists an integer n such that for each element $t \in L$ we have $\mathrm{wd}(t) \leq n$. Similarly, we call $S \in \mathbb{K}\langle\langle \mathrm{SP} \rangle\rangle$ *width-bounded* if $\operatorname{supp} S$ has bounded width. From the definition of series-rational sp-series we get immediately that any series-rational sp-series has bounded width. As for sp-languages the opposite is not true.

3.3 Bounded Width and Bounded Depth

The bounded width of a regular sp-series is reflected by the "bounded depth" of a BRAC. Every atomic path is of depth 0, $\mathrm{dp}(G_1 \cdot G_2) = \max\{\mathrm{dp}(G_1), \mathrm{dp}(G_2)\}$ and $\mathrm{dp}(G_1 \|_{p,q} G_2) = 1 + \max\{\mathrm{dp}(G_1), \mathrm{dp}(G_2)\}$. Therefore, the depth of a path measures the nesting of branchings within the path. A BRAC $\mathcal{A}$ is of *bounded depth* if the depth of its paths is uniformly bounded (Lodaya and Weil [13] require this for successful paths, only). Any series recognized by a BRAC of bounded depth is of bounded width. The converse for sp-languages was shown in [11] by just counting and thereby limiting the depth of a path. That proof can be extended to bisemirings that do not allow an additive decomposition of 0. The problem in the general case arises from the existence of paths of different depths having the same label t. Then two such paths can have non-zero costs, but the sum of these costs can be 0. If now the path with larger depth is disabled, the cost of the sp-poset t changes (see the complete paper [12] for a more elaborated example). To overcome this problem, we will keep track of the actual width (and not just the depth) of a poset. This is achieved by a stack where the widths encountered up to the last fork transition are stored. In addition, we restrict the size of the stack as well as the natural numbers stored therein to the width of the support. This allows to perform the construction within the realm of finite-state systems. By $S_{|n}$ we denote the restriction of S to the sp-posets of width less than or equal to n.

Proposition 3.4. *Let $n \in \mathbb{N}$ and S a regular sp-series. Then $S_{|n}$ can be recognized by a depth-bounded BRAC.*

Proof. Let $\mathcal{A}$ be a BRAC recognizing S. We put $[n] := \{1, \ldots, n\}$, and $[n]^+$ denotes the set of nonempty words over $[n]$. From $\mathcal{A}$ we construct a new automaton $\mathcal{A}'$ as follows. The states get a new component that we prefer to think of as a stack because of the operations that will be performed on it; its alphabet is $[n]$ and its size is restricted to depth n:

$$Q' = \{(p, w) \in Q \times [n]^+ \ : \ |w| \leq n\} \ .$$

A sequential transition does not change the stack:

$$T'_{\text{seq}}((p,u),a,(q,v)) = \begin{cases} T_{\text{seq}}(p,a,q) & ;\text{if } u = v \\ 0 & ;\text{otherwise} \end{cases} ,$$

but at any fork, we push a new top symbol onto the stack (if this is not forbidden by the size restriction):

$$T'_{\text{fork}}((p,u),\{(p_1,u_1),(p_2,u_2)\}) = \begin{cases} T_{\text{fork}}(p,\{p_1,p_2\}) & ;\text{if } \ u_1 = u_2 = u1 \\ 0 & ;\text{otherwise} \end{cases} .$$

The real work is done in a join transition. First, the two top symbols are summed up – they are meant to measure the width of the two subpaths, hence their sum is the width of the total path since the matching fork. Then, this sum is compared with the previous symbol on the stack (*i.e.*, the width of the path between the previous fork and join) and the larger of the two replaces them both:

$T'_{\text{join}}(\{(q_1,v_1),(q_2,v_2)\},(q,v)) = T_{\text{join}}(\{q_1,q_2\},q)$ if there are $w \in [n]^+$ and $x, y_1, y_2, z \in [n]$ such that $v_1 = wxy_1, v_2 = wxy_2, v = wz$ with $z = \max\{x, y_1 + y_2\}$. Otherwise, $T'_{\text{join}}(\{(q_1,v_1),(q_2,v_2)\},(q,v)) = 0$.

At the start, the newly constructed BRAC places a 1 in its stack and it is only allowed to stop if the stack contains precisely one symbol:

$$\lambda'(p,u) = \begin{cases} \lambda(p) & ;\text{if } u = 1 \\ 0 & ;\text{otherwise} \end{cases} , \qquad \gamma'(q,v) = \begin{cases} \gamma(q) & ;\text{if } |v| = 1 \\ 0 & ;\text{otherwise} \end{cases} .$$

The construction ensures that the symbol in the stack of a final state is the width of the poset executed by the automaton. □

Let S be a regular sp-series of bounded width. Then, as a consequence of this proposition, it can be recognized by a BRAC of bounded depth. Thus, width-boundedness of regular sp-series and depth-boundedness of BRACs are equivalent notions.

4 Closure Properties of Regular SP-Series

In the proofs of the following results, we use branching automata with restricted possibilities to enter and to leave the automaton. A BRAC $\mathcal{A}$ is called *normalized* if there are unique states i and f with $\lambda(i) \neq 0$, $\gamma(f) \neq 0$ and, moreover, we have for all $p, p_1, p_2 \in Q$ and $a \in \Sigma$

- $\lambda(i) = 1$ and $T_{\text{seq}}(p, a, i) = 0$ as well as $T_{\text{join}}(\{p_1, p_2\}, i) = 0 = T_{\text{fork}}(p_1, \{i, p_2\})$,
- $\gamma(f) = 1$ and $T_{\text{seq}}(f, a, p) = 0$ and $T_{\text{fork}}(f, \{p_1, p_2\}) = 0 = T_{\text{join}}(\{p_1, f\}, p_2)$.

Proposition 4.1. *Any regular sp-series is the behavior of some normalized BRAC.*

The closure of regular sp-series under + is shown by the classical construction for the union of regular languages: one considers the disjoint union of two automata. Closure under parallel composition is shown similarly: one considers the disjoint union of two BRACs and adds initial fork- and terminal join-transitions as appropriate. It turns out that this obvious idea works for normalized BRACs, only, which is, by the proposition above, no restriction on the sp-series.

Proposition 4.2. *Let S_1 and S_2 be regular sp-series. Then $S_1 + S_2$ and $S_1 \| S_2$ are regular.*

The rest of this section is devoted to the closure of the class of regular series under sequential multiplication and sequential iteration. It is tempting to believe that constructions familiar from the theory of finite automata work here as well. But, as already observed for sp-languages by Lodaya and Weil [13], the obvious variant of the classical construction for the product does not yield the correct result. The problem is that the newly constructed automaton can switch from $\mathcal{A}_1$ into $\mathcal{A}_2$ independently in parallel subpaths.

Lodaya and Weil showed that this problem does not arise when one restricts to "behaved automata". Then they show that one can transform any branching automaton into an equivalent behaved one. We proceed differently giving a direct construction for the sequential product. More precisely, we "send a signal" from the initial state along the path. In fork transitions, this signal is only propagated along one branch. In order not to duplicate paths, the signal is sent to the "smaller" of the two states that arise from the fork transition.[2] The newly constructed BRAC can only switch from $\mathcal{A}_1$ into $\mathcal{A}_2$ in the presence of this signal, and in any successful path, the signal has to be present at the final state.

Proposition 4.3. *Let $S_1, S_2 \in \mathbb{K}\langle\langle \text{SP} \rangle\rangle$ be two regular sp-series. Then $S_1 \cdot S_2$ is regular.*

Similarly to the sequential composition, the classical construction for the sequential iteration suggests itself - and yields an incorrect result as the following example shows.

Example 4.4. We work with the Boolean bisemiring $\mathbb{B} = (\{0, 1\}, \vee, \wedge, \wedge, 0, 1)$, *i.e.*, in the setting of sp-languages. Consider the BRAC from Figure 2 (left) where we omitted the costs; any transition depicted has cost 1 and no further transitions

[2] This is actually the reason why we have to assume that these two states are different, *i.e.*, that we work with sets in the definition of fork- and join-transitions and not with multisets as Lodaya and Weil do.

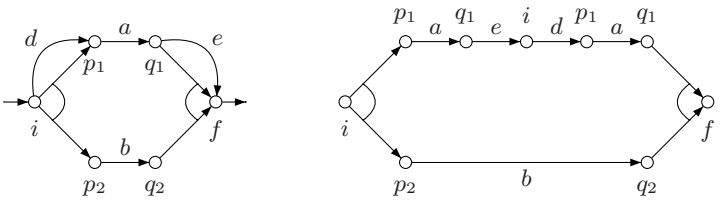

Fig. 2. A BRAC and a problematic path in the classical construction

have nonzero cost. The support of the recognized sp-series is $\{a\|b, dae\}$. The classical construction for the sequential iteration tells us to add, among other transitions, one of the form $q_1 \xrightarrow{e} i$ since there is a sequential transition $q_1 \xrightarrow{e} f$ in the BRAC in consideration. But then we get the path depicted in Figure 2 (right) whose label is $(aeda)\|b$ which does not belong to the sequential iteration of the sp-language generated by the BRAC we started with.

Lodaya and Weil's solution is, again, to use behaved automata. Our direct construction sends not just one, but two signals. These two signals travel along different ways: whenever they can separate in a fork transition, they do so. Then the newly constructed automaton is allowed to jump from the final state to the initial state only in case both signals are present. As before, in any successful path, both signals are present in the first and the last state.

Proposition 4.5. *If $S \in \mathbb{K}\langle\langle \mathrm{SP} \rangle\rangle$ is regular, then the sequential iteration S^+ is regular.*

Proof. Let $\mathcal{A}$ be a branching automaton with costs with behavior S. As explained above, we need two signals that travel along maximal ways in a path. This is modeled by extending the states of $\mathcal{A}$ by two binary digits, *i.e.*, we set $Q' = Q \times \{0,1\}^2$. In sequential transitions, the signals are simply propagated:

$$T'_{\mathrm{seq}}((p,x,x'),a,(q,y,y')) = \begin{cases} T_{\mathrm{seq}}(p,a,q) & ;\text{if } y = x, y' = x' \\ 0 & ;\text{otherwise} \end{cases} .$$

In a fork transition, the first signal is propagated to the smaller successor state and the second signal to the larger one. In particular, they separate in case both are present. For that, let $\leq$ be an arbitrary but fixed linear order on Q, and assume $p_1 \leq p_2$. Then we put:

$$T'_{\mathrm{fork}}((p,x,x'),\{(p_1,x_1,x'_1),(p_2,x_2,x'_2)\})$$

$$= \begin{cases} T_{\mathrm{fork}}(p,\{p_1,p_2\}) & ;\text{if } p_1 \neq p_2, x_1 = x, \\ & \quad x'_1 = 0, x_2 = 0, x'_2 = x' \\ 0 & ;\text{otherwise} \end{cases} .$$

Similarly to the sequential transitions, in a join transition both incoming signals are propagated to the unique successor state:

$$T'_{\text{join}}(\{(q_1,x_1,x'_1),(q_2,x_2,x'_2)\},(q,x,x')) = \begin{cases} T_{\text{join}}(\{q_1,q_2\},q) & ;\text{if } x'_1=0, x_2=0, x=x_1, x'=x'_2 \\ 0 & ;\text{otherwise} \end{cases}.$$

Finally, the automaton starts in the presence of both signals and is only allowed to finish when both signals are present:

$$\lambda'(p,x,x') = \begin{cases} \lambda(p) & ;\text{if } x=x'=1 \\ 0 & ;\text{otherwise} \end{cases}, \qquad \gamma'(q,x,x') = \begin{cases} \gamma(q) & ;\text{if } x=x'=1 \\ 0 & ;\text{otherwise} \end{cases}.$$

Due to Proposition 4.1, $\mathcal{A}$ can be assumed to be normalized. Then $\mathcal{A}'$ is normalized too. For any sequential or join transition of $\mathcal{A}'$ with final state f', we add the corresponding transition into i'.

The construction above ensures that any path of this new BRAC $\mathcal{A}^+$ that contributes to the cost is the sequential product of finitely many paths of the BRAC $\mathcal{A}$. Moreover, this gives a bijection between the sets of successful paths of $\mathcal{A}^+$ and tuples of successful paths of $\mathcal{A}$. □

Theorem 4.6. *Let $\mathbb{K}$ be an arbitrary bisemiring. Every series-rational sp-series $S \in \mathbb{K}\langle\langle \mathrm{SP} \rangle\rangle$ is regular; it is even recognized by a normalized BRAC of bounded depth.*

5 From Regular and Bounded Depth to Series-Rational

If $G = G_1 \|_{p,q} G_2$ is a path, f denotes the starting fork transition of G, and j the finishing join transition of G, then we say that (f,j) is a *matched pair*.

Theorem 5.1. *The behavior of any BRAC of bounded depth is series-rational.*

Proof. Let $\sqsubseteq$ be a relation on the set of matched pairs such that $(f,j) \sqsubseteq (f',j')$ whenever there exists a parallel path G starting with f' and ending with j' that contains (f,j) as a matched pair. Since $\mathcal{A}$ is of bounded depth, $\sqsubseteq$ can be extended to a linear order $\leq$.

Let (f,j) be a matched pair and $p,q \in Q$ states of $\mathcal{A}$. We denote by $S^{(f,j)}_{p,q}$ the series with

$$(S^{(f,j)}_{p,q},t) = \bigoplus_{G:p \xrightarrow{t} q} \mathrm{cost}(G)$$

where $t \in \mathrm{SP}$ and the paths G, over which the sum extends, use only matched pairs smaller than or equal to (f,j). By induction along $\leq$, one shows that $S^{(f,j)}_{p,q}$ is series-rational which yields the result since there are only finitely many matched pairs. □

The special case $\mathbb{K} = \mathbb{B}$ was shown by Lodaya and Weil [13]. Their proof uses a nested induction which we simplified here to just one induction along the linear order of matched pairs.

Now we can prove the main theorem about regular and series-rational sp-series.

Theorem 5.2. *Let $\mathbb{K}$ be an arbitrary bisemiring and $S \in \mathbb{K}\langle\langle \mathrm{SP} \rangle\rangle$. The following are equivalent:*

1. *S is series-rational.*
2. *S is recognized by a BRAC of bounded depth.*
3. *S is regular and has bounded width.*

Proof. Due to Theorem 4.6, (1) implies (2). By Theorem 5.1, (2) implies (1). Statement (3) implies (2) by Proposition 3.4, the remaining implication "(2) $\Rightarrow$ (3)" is obvious. □

By putting $\mathbb{K} = \mathbb{B}$, we get as a consequence of the last theorem the result of Lodaya and Weil [13] for sp-languages.

References

[1] J. Berstel and C. Reutenauer. Recognizable formal power series on trees. *Theoret. Comp. Sc.*, 18:115–148, 1982. 151

[2] J. Berstel and C. Reutenauer. *Rational Series and Their Languages*, volume 12 of *EATCS Monographs on Theoret. Comp. Sc.* Springer, 1988. 150

[3] M. Droste and P. Gastin. The Kleene-Schützenberger theorem for formal power series in partially commuting variables. *Information and Computation*, 153:47–80, 1999. 151

[4] M. Droste and D. Kuske. Skew and infinitary formal power series. In *ICALP 2003*, Lecture Notes in Computer Science. Springer, 2003. Accepted. 151

[5] M. Droste and H. Vogler. A Kleene theorem for weighted tree automata. Technical Report TUD-FI02-04, TU Dresden, June 2002. 151

[6] J. L. Gischer. The equational theory of pomsets. *Theoret. Comp. Sc.*, 61:199–224, 1988. 150, 151

[7] J. Grabowski. On partial languages. *Ann. Soc. Math. Pol. IV: Fund. Math.*, 4(2):427–498, 1981. 150

[8] S. E. Kleene. Representations of events in nerve nets and finite automata. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 3–42. Princeton University Press, 1956. 151

[9] W. Kuich. Formal power series over trees. In *Proc. of the 3rd International Conference Developments in Language Theory*, pages 60–101. Aristotle University of Thessaloniki, 1997. 150, 151

[10] W. Kuich and A. Salomaa. *Semirings, Automata, Languages*, volume 5 of *EATCS Monographs on Theoret. Comp. Sc.* Springer, 1986. 150

[11] D. Kuske. Towards a language theory for infinite N-free pomsets. *Theoret. Comp. Sc.*, 299:347–386, 2003. 155, 156

[12] D. Kuske and I. Meinecke. Branching automata with costs - a way of reflecting parallelism in costs. Technical Report MATH-AL-4-2003, TU Dresden, March 2003. see `www.math.tu-dresden.de/~meinecke`. 150, 154, 156

[13] K. Lodaya and P. Weil. Series-parallel languages and the bounded-width property. *Theoret. Comp. Sc.*, 237:347–380, 2000. 150, 151, 152, 155, 156, 158, 161

[14] K. Lodaya and P. Weil. Rationality in algebras with a series operation. *Information and Computation*, 171:269–293, 2001. 150, 155

[15] Ch. Pech. Kleene's theorem for weighted tree automata. In *14th International Symposium on Fundamentals of Computation Theory (FCT 2003)*, Lecture Notes in Computer Science. Springer, 2003. Accepted. 151

[16] A. Salomaa and M. Soittola. *Automata-Theoretic Aspects of Formal Power Series*. Texts and Monographs in Computer Science. Springer, 1978. 150, 155

[17] M. P. Schützenberger. On the definition of a family of automata. *Information and Control*, 4:245–270, 1961. 151