A Simple Method for Building Bimachines from Functional Finite-State Transducers

Stefan Gerdjikov[1,2], Stoyan Mihov[2(✉)], and Klaus U. Schulz[3]

[1] Faculty of Mathematics and Informatics, Sofia University, 5, James Borchier Blvd., 1164 Sofia, Bulgaria
stefangerdzhikov@fmi.uni-sofia.bg

[2] Institute of Information and Communication Technologies, Bulgarian Academy of Sciences, 25A, Acad. G. Bonchev Street, 1113 Sofia, Bulgaria
stoyan@lml.bas.bg

[3] Centrum für Informations-und Sprachverarbeitung (CIS), Ludwig-Maximilians-Universität München, Oettingenstr. 67, 80538 München, Germany
schulz@cis.uni-muenchen.de

Abstract. The standard construction of a bimachine from a functional transducer involves a preparation step for converting the transducer into an unambiguous transducer (A transducer is unambiguous if there exists at most one successful path for each label.). The conversion involves a specialized determinization. We introduce a new construction principle where the transducer is directly translated into a bimachine. For any input word accepted by the transducer the bimachine exactly imitates one successful path of the transducer. For some classes of transducers the new construction can build a bimachine with an exponentially lower number of states compared to the standard construction. We first present a simple and generic variant of the construction. A second specialized version leads to better complexity bounds in terms of the size of the bimachine.

Keywords: Bimachines · Transducers · Rational functions

1 Introduction

Finite-state transducers are used for a large spectrum of translation tasks in text analysis and natural language processing [4–7]. Many practical translation tasks are functional in the sense that a given input needs to be transformed into a unique output. While (non-deterministic versions of) finite-state transducers can model arbitrary "regular" (s.b.) functions between strings, many regular functions cannot be recognized by deterministic finite-state transducers. In contrast, bimachines as a more powerful type of finite-state device enable a fully deterministic processing of arbitrary regular string functions [11].

For a given regular string function f it is often simple to find a non-deterministic finite-state transducer that represents f. Since a deterministic processing via bimachines is more efficient, there is an obvious interest in general methods for converting functional finite-state transducers into bimachines or equivalent devices [10,12]. The classical algorithm, described in [7], starts with a preparation step for converting the transducer into an unambiguous transducer. The conversion requires that the source transducer is "pseudo-deterministic". Afterwards it uses a specialized determinization for discarding unwanted paths. Essentially, only the least accepting paths under some lexicographical order are left. This construction can be applied to arbitrary output monoids after introducing a linear order on the outputs of single transitions.

Here we introduce a new single-step method that can be applied to any functional real-time transducer with output (codomain) in an arbitrary monoid. States of the right deterministic automaton of the bimachine are sets R of active states obtained when using inversed transitions of the functional input transducer $\mathcal{T}$, starting from final states. States of the left deterministic automaton of the bimachine are sets L of active states of $\mathcal{T}$ that are enhanced by a special function. Using this enhancement the bimachine satisfies the "path reconstruction" principle: (i) At each step, the bimachine output m represents the output of a single transducer transition step $\langle q, \langle a, m\rangle, q'\rangle$ for some $q \in L \cap R$. (ii) for any input w: the sequence of bimachine outputs w is given by the sequence of outputs of $\mathcal{T}$ for w on a specific path.

After formal preliminaries in Sect. 2 the new construction is described in Sect. 3. We start with a generic and flexible version that is conceptually simple. Afterwards a specialized version is added which leads to better complexity bounds for the number of states of the left and right deterministic automata of the bimachine. Correctness proofs are given. For the sake of comparison we sketch the classical bimachine construction in Sect. 4. A class of examples is given where the new construction leads to an exponentially lower number of states. A conclusion is presented in Sect. 5.

2 Formal Preliminaries

We assume that the reader is familiar with the basic notions of words over an alphabet and monoids (see e.g. [2]). The set Σ^* with concatenation as monoid operation and the empty word ε as unit element is called the *free monoid* over Σ. We list notions needed for the discussion of the paper. A *monoidal finite-state automaton* is a tuple of the form $\mathcal{A} = \langle \mathcal{M}, Q, I, F, \Delta\rangle$ where

– $\mathcal{M} = \langle M, \circ, e\rangle$ is a monoid,
– Q is a finite set called the set of states,
– $I \subseteq Q$ is the set of initial states,
– $F \subseteq Q$ is the set of final states, and
– $\Delta \subseteq Q \times M \times Q$ is a finite set of transitions called the transition relation.

A *proper path* in $\mathcal{A}$ is a finite sequence of $k > 0$ transitions, denoted

$$\pi = q_0 \to^{a_1} q_1 \ldots \to^{a_k} q_k$$

where $\langle q_{i-1}, a_i, q_i \rangle \in \Delta$ for $i = 1 \ldots k$. The monoid element $w = a_1 \circ \ldots \circ a_k$ is called the *label* of π. A *successful path* is a path starting in an initial state and ending in a final state.
The *generalized transition relation* Δ^* is defined as the smallest subset of $Q \times M \times Q$ with the following closure properties:

- for all $q \in Q$ we have $\langle q, e, q \rangle \in \Delta^*$.
- For all $q_1, q_2, q_3 \in Q$ and $w, a \in M$: if $\langle q_1, w, q_2 \rangle \in \Delta^*$ and $\langle q_2, a, q_3 \rangle \in \Delta$, then also $\langle q_1, w \circ a, q_3 \rangle \in \Delta^*$.

The monoidal language *accepted* (or *recognized*) by $\mathcal{A}$ is defined as $L(\mathcal{A}) := \{w \in M \mid \exists p \in I\ \exists q \in F : \langle p, w, q \rangle \in \Delta^*\}$.
A monoidal finite-state automaton $\mathcal{A}$ is *unambiguous* iff for every element $m \in M$ there exists at most one successful path in $\mathcal{A}$ with label m.
A state $q \in Q$ is *accessible* if q is the ending of a path of $\mathcal{A}$ starting from an initial state. A state $q \in Q$ is *co-accessible* if q is the starting of a path of $\mathcal{A}$ ending in a final state. A monoidal finite-state automaton $\mathcal{A}$ is *trimmed* iff each state $q \in Q$ is accessible and co-accessible.
A *deterministic finite-state automaton* is a monoidal finite-state automaton over the free monoid $\mathcal{A} = \langle \Sigma^*, Q, I, F, \Delta \rangle$, such that $|I| = 1$ and Δ is a graph of a (partial) function with domain $dom(\Delta) \subseteq Q \times \Sigma$. In this case we identify Δ with the function $\Delta : Q \times \Sigma \to Q$ that it represents. The *reversed finite-state automaton* for $\mathcal{A}$ is $\mathcal{A}^{rev} = \langle \Sigma^*, Q, F, I, \Delta^{rev} \rangle$, where $\Delta^{rev} = \{\langle q, a^{rev}, p \rangle \mid \langle p, a, q \rangle \in \Delta\}$.

Definition 1. *A monoidal finite-state automaton $\mathcal{T}$ over a monoid $\mathcal{M}$ is a* monoidal finite-state transducer *iff $\mathcal{M}$ can be represented as the Cartesian product of a free monoid Σ^* with another monoid $\mathcal{M}'$, i.e. $\mathcal{M} = \Sigma^* \times \mathcal{M}'$. For a monoidal finite-state transducer $\mathcal{T} = \langle \Sigma^* \times \mathcal{M}, Q, I, F, \Delta \rangle$* the underlying finite-state automaton *is the monoidal finite-state automaton $\mathcal{A}_\mathcal{T} = \langle \Sigma^*, Q, I, F, \Delta_\Sigma \rangle$ where $\Delta_\Sigma = \{\langle p, a, q \rangle \mid \exists m \in M(\langle p, \langle a, m \rangle, q \rangle \in \Delta\}$. A monoidal finite-state transducer $\mathcal{T} = \langle \Sigma^* \times \mathcal{M}', Q, I, F, \Delta \rangle$ is said to be* real-time *if $\Delta \subseteq Q \times (\Sigma \times M') \times Q$.*

Let $\mathcal{M}$ be a monoid. A set $L \subseteq M$ is *rational* iff it is accepted by a monoidal finite-state automaton. If M is a Cartesian product, then rational sets are relations. A *rational function* is a rational set that is a function.

Definition 2. *A* bimachine *is a tuple $\mathcal{B} = \langle \mathcal{M}, \mathcal{A}_L, \mathcal{A}_R, \psi \rangle$, where:*

- *$\mathcal{A}_L = \langle \Sigma, L, s_L, L, \delta_L \rangle$ and $\mathcal{A}_R = \langle \Sigma, R, s_R, R, \delta_R \rangle$ are deterministic finite-state automata called the* left *and* right automaton *of the bimachine;*
- *$\mathcal{M} = \langle M, \circ, e \rangle$ is the output monoid and $\psi : (L \times \Sigma \times R) \to M$ is a partial function called the* output function.

Note that all states of $\mathcal{A}_L$ and $\mathcal{A}_L$ are final. The function ψ is naturally extended to the generalized output function ψ^* *as follows:*

- $\psi^*(l, \varepsilon, r) = e$ *for all* $l \in L, r \in R$;
- $\psi^*(l, t\sigma, r) = \psi^*(l, t, \delta_R(r, \sigma)) \circ \psi(\delta_L^*(l, t), \sigma, r)$ *for* $l \in L, r \in R, t \in \Sigma^*, \sigma \in \Sigma$.

The function represented by the bimachine *is*

$$O_{\mathcal{B}} : \Sigma^* \to M : t \mapsto \psi^*(s_L, t, s_R).$$

If $O_{\mathcal{B}}(t) = t'$ *we say that the bimachine* $\mathcal{B}$ translates t *into* t'.

Note that for any states $p, q \in Q$ of a monoidal finite-state transducer $\mathcal{T} = \langle \Sigma^* \times \mathcal{M}, Q, I, F, \Delta \rangle$ and word $w \in \Sigma^*$ holds $\exists m \in M : \langle p, \langle w, m \rangle, q \rangle \in \Delta^* \iff \langle p, w, q \rangle \in \Delta_\Sigma^*$, where Δ_Σ is the transition relation of its underlying automaton.

If $\mathcal{A}^{rev} = \langle \Sigma^*, Q, F, I, \Delta^{rev} \rangle$ is the reversed finite-state automaton of $\mathcal{A} = \langle \Sigma^*, Q, I, F, \Delta \rangle$, then for any states $q, p \in Q$ and any word $w \in \Sigma^*$ we have $\langle p, w, q \rangle \in \Delta^* \iff \langle q, w^{rev}, p \rangle \in \Delta^{rev*}$.

After applying the power-set construction to transform a nondeterministic automaton $\mathcal{A} = \langle \Sigma^*, Q, I, F, \Delta \rangle$ into an equivalent deterministic one $\mathcal{A}_D = \langle \Sigma^*, Q_D, \{I\}, F_D, \delta_D \rangle$ with states $Q_D \subseteq 2^Q$ the following holds:

$$\forall w \in \Sigma^* \; \forall P \in Q_D : \delta_D^*(P, w) = \{q \,|\, \exists p \in P : \langle p, w, q \rangle \in \Delta^*\}.$$

Proposition 1. *(Cf. e.g. [7]) Let* $\mathcal{A} = \langle \Sigma^* \times \mathcal{M}, Q, I, F, \Delta \rangle$ *be a trimmed monoidal transducer. If* $\mathcal{A}$ *does not contain any cycle of the form* $\langle p, \langle \varepsilon, m \rangle, p \rangle \in \Delta^*$ *with* $m \neq e$, *then* $\mathcal{A}$ *can be effectively transformed into a real-time transducer* $\mathcal{A}'$ *such that* $L(\mathcal{A}) \cap (\Sigma^+ \times M) = L(\mathcal{A}') \cap (\Sigma^+ \times M)$. *Furthermore, we can effectively compute the set* $\{m \,|\, \langle \varepsilon, m \rangle \in L(\mathcal{A})\}$.

3 New Bimachine Construction

From now on we assume that $\mathcal{T} = \langle \Sigma^* \times \mathcal{M}, Q, I, F, \Delta \rangle$ is any trimmed real-time functional monoidal transducer. We assume that $\langle \varepsilon, e \rangle \in L(\mathcal{T})$. Let $\mathcal{A}_{\mathcal{T}} = \langle \Sigma^*, Q, I, F, \Delta_\Sigma \rangle$ be the underlying finite-state automaton of $\mathcal{T}$ and $\mathcal{A}_{\mathcal{T}}^{rev} = \langle \Sigma^*, Q, F, I, \Delta_\Sigma^{rev} \rangle$ be the reverse finite-state automaton of $\mathcal{A}_{\mathcal{T}}$. Let $\mathcal{A}_{\mathcal{T}D} = \langle \Sigma^*, 2^Q, \{I\}, F_D, \delta_{\Sigma D} \rangle$ and ${\mathcal{A}_{\mathcal{T}}^{rev}}_D = \langle \Sigma^*, 2^Q, \{F\}, I_D, {\delta_\Sigma^{rev}}_D \rangle$ be the deterministic finite-state automata for $\mathcal{A}_{\mathcal{T}}$ and $\mathcal{A}_{\mathcal{T}}^{rev}$, respectively.

For each set of states $P \subseteq Q$ and $w \in \Sigma^*$, we define the set of w-successors and w-predecessors of P as

$$\begin{aligned} Succ_w(P) &:= {\delta_\Sigma}_D^*(P, w) = \{q \in Q \,|\, \exists p \in P, m \in M : \langle p, \langle w, m \rangle, q \rangle \in \Delta^*\} \\ Pred_w(P) &:= {\delta_\Sigma^{rev}}_D^*(P, w) = \{q \in Q \,|\, \exists p \in P, m \in M : \langle q, \langle w^{rev}, m \rangle, p \rangle \in \Delta^*\}. \end{aligned}$$

Note that the first (second) clause is based on a left-to-right (right-to-left) reading order.

Lemma 1 (Butterfly Lemma). *Let $\mathcal{T}$ be as above. Let $u, v \in \Sigma^*$, $a \in \Sigma$, let $L := Succ_u(I)$, $L' := Succ_{ua}(I)$, $R' := Pred_v(F)$ and $R := Pred_{av}(F)$. Then*

1. *for all $q \in L \cap R$ there is $q' \in L' \cap R'$ and $m \in M$ such that $\langle q, \langle a, m\rangle, q'\rangle \in \Delta$,*
2. *for all $q' \in L' \cap R'$ there is $q \in L \cap R$ and $m \in M$ such that $\langle q, \langle a, m\rangle, q'\rangle \in \Delta$,*
3. *$L \cap R \neq \emptyset$ iff $L' \cap R' \neq \emptyset$.*

Proof. As to 1, let $q \in L \cap R$. Since $R = Pred_a(R')$ there exists a transition of the form $\langle q, \langle a, m\rangle, q'\rangle \in \Delta$ such that $q' \in R$. Since $q \in L$ we have $q' \in L'$. 2 follows by a symmetric argument. 3 directly follows from 1 and 2. $\square$

3.1 Generic Construction

We now show how to build an equivalent bimachine $\mathcal{B} = \langle \mathcal{M}, \mathcal{A}_L, \mathcal{A}_R, \psi\rangle$, given the transducer $\mathcal{T}$ as input. First, we construct the **right automaton** $\mathcal{A}_R$ applying a determinization procedure to the reversed underlying automaton of $\mathcal{T}$. Let

$$\mathcal{A}_R = \mathcal{A}_{\mathcal{T}}^{rev}{}_D = \langle \Sigma^*, Q_R, s_R, F_R, \delta_R\rangle.$$

By definition $s_R = \{F\}$ and $\delta_R(R, a) = \delta_{\Sigma}^{rev}{}_D(R, a) = Pred_a(R)$ for $R \in Q_R$ and $a \in \Sigma$. The idea for the **left automaton** is to use the accessible sets in $\mathcal{A}_{\mathcal{T}D}$

$$Q'_L := \{\delta_{\Sigma}{}^*_D(I, w) \mid w \in \Sigma^*\} = \{Succ_w(I) \mid w \in \Sigma^*\}$$

as a “core” part of the states, but to enrich this core part by additional information that enables the reconstruction of successful paths in $\mathcal{T}$. Let $L \in Q'_L$. An *L-centered state selector function* is a partial function $\phi : Q_R \to Q$ such that the following conditions hold for any state of the right automaton $R \in Q_R$:

1. $\phi(R)$ is defined iff $R \cap L \neq \emptyset$ and
2. if $\phi(R)$ is defined, then $\phi(R) \in R \cap L$.

A state of the left automaton $\mathcal{A}_L = \langle \Sigma, Q_L, s_L, Q_L, \delta_L\rangle$ is a pair $\langle L, \phi\rangle$ where $L \in Q'_L$ and ϕ is an L-centered state selector function. The following induction defines s_L, the set of states Q_L, and the transition function δ_L.

– $s_L := \langle I, \phi_0\rangle$ where $\phi_0(R) := \begin{cases} \text{any element of } R \cap I & \text{if } R \cap I \neq \emptyset \\ \text{undefined} & \text{otherwise.} \end{cases}$

– For $\langle L, \phi\rangle \in Q_L$ and $a \in \Sigma$ we define $\delta_L(\langle L, \phi\rangle, a) := \langle L', \phi'\rangle$ where
  - $L' := Succ_a(L)$.
  - $\phi'(R') := \begin{cases} \text{any element of } \{q' \mid \exists m \in M : \langle q, \langle a, m\rangle, q'\rangle \in \Delta\} \\ \qquad\qquad\qquad \text{if } q = \phi(Pred_a(R')) \text{ is defined} \\ \text{undefined} \qquad \text{otherwise.} \end{cases}$

In the above notions we show that

1. for each state $\langle L, \phi\rangle$ always ϕ is an L-centered state selection function, and
2. if $\phi(Pred_a(R'))$ is defined, then $q' = \phi'(R')$ is also defined.

The proof is by induction. For $s_L := \langle I, \phi_0 \rangle$ clearly ϕ_0 is defined as an I-centered state selection function. For the induction step, given state $\langle L, \phi \rangle$ assume that ϕ is an L-centered state selection function. Let $R' \in Q_R$ and $R := \mathit{Pred}_a(R')$. First, if $q = \phi(R)$ is defined, then (ϕ is L-centered) $L \cap R \neq \emptyset$ and $q \in L \cap R$. By the Butterfly Lemma we have that $L' \cap R' \neq \emptyset$ and further there exists a transition $\langle q, \langle a, m \rangle, q' \rangle \in \Delta$ such that $q' \in L' \cap R'$. Therefore $q' = \phi'(R')$ is defined and $\phi'(R') \in L' \cap R'$. On the other hand, if $\phi(R)$ is undefined, then (ϕ is L-centered) $L \cap R = \emptyset$ and (Butterfly Lemma) $L' \cap R' = \emptyset$. It follows that ϕ' is L'-centered.

It remains to define the **output function** ψ of the bimachine. Given a pair of states $\langle L, \phi \rangle$ and R' of the left and right automaton and $a \in \Sigma$, let $\langle L', \phi' \rangle := \delta_L(\langle L, \phi \rangle, a)$ and $R := \mathit{Pred}_a(R') = \delta_R(R', a)$. Then

$$\psi(\langle L, \phi \rangle, a, R') := \begin{cases} \text{any element of } \{m \mid \langle \phi(R), \langle a, m \rangle, \phi'(R') \rangle \in \Delta\} & \text{if } !\phi(R) \\ \text{undefined} & \text{otherwise} \end{cases}$$

(We have shown above that there always exists a transition of the above form.)

Correctness. We now show that the function defined by the bimachine $\mathcal{B} = \langle \mathcal{A}_L, \mathcal{A}_R, \psi \rangle$ coincides with the language of the transducer $\mathcal{T}$.

Theorem 1. *Let $u = a_1 \dots a_k \in \mathit{dom}(\mathcal{T})$. For $i \in \{0, 1, \dots, k\}$ let $\langle L_i, \phi_i \rangle := \delta_L^*(s_L, a_1 \dots a_i)$ and $R_i := \delta_R^*(s_R, a_k, \dots, a_{i+1})$. Then for any $i \leq k$ the following hold:*

1. *$q_i := \phi_i(R_i)$ is defined.*
2. *$m_{i+1} := \psi(L_i, a_{i+1}, R_{i+1})$ is defined and $\langle q_i, \langle a_{i+1}, m_{i+1} \rangle, q_{i+1} \rangle \in \Delta$.*

Furthermore $O_{\mathcal{B}} = L(\mathcal{T})$.

Proof. Let $u_i = a_1 \dots a_i$ and $v_i = a_{i+1} \dots a_k$. Then we have that $L_i = \mathit{Succ}_{u_i}(I)$ and $R_i = \mathit{Pred}_{v_i}(F)$. Since $u_i v_i = u \in \mathit{dom}(\mathcal{T})$ it follows that $L_i \cap R_i \neq \emptyset$. Thus, since ϕ_i is L_i-centered we deduce that $q_i = \phi_i(R_i)$ is defined. Further, since $q_{i+1} = \phi_{i+1}(R_{i+1})$ is well-defined it follows that there is a transition $\langle q_i, \langle a_{i+1}, m_{i+1} \rangle, q_{i+1} \rangle \in \Delta$. As a consequence we obtain

$$\langle q_0, \langle u, m_1 \dots m_k \rangle, q_k \rangle \in \Delta^*.$$

Since $q_0 = \phi_0(R_0) \in L_0 = I$ and $q_k = \phi_k(R_k) \in R_k = F$ we have $\langle u, m_1 \dots m_k \rangle \in L(\mathcal{T})$. Furthermore in this case $O_{\mathcal{B}}(u) = m_1 \dots m_k = L(\mathcal{T})(u)$. This proves that if $u \in \mathit{dom}(\mathcal{T})$, then $u \in \mathit{dom}(\mathcal{B})$ and $L(\mathcal{T})(u) = O_{\mathcal{B}}(u)$.

Finally, if $u \notin \mathit{dom}(\mathcal{T})$, then $R_0 \cap I = \emptyset$ and therefore $\phi_0(R_0)$ is not defined. In particular, $O_{\mathcal{B}}(u)$ is not defined. Hence both functions have the same domain and coincide.

Remark 1. The construction can be applied to a non-functional transducer $\mathcal{T}$ and in this case for the output function of the bimachine we have $O_{\mathcal{B}} \subseteq L(\mathcal{T})$.

Applying the standard conversion of a bimachine to transducer we obtain the following corollary.

Corollary 1. *For any functional monoidal finite-state transducer $\mathcal{T}$ there exists an unambiguous monoidal finite-state transducer $\mathcal{T}'$ such that $L(\mathcal{T}) = L(\mathcal{T}')$.*

Proof. After constructing the bimachine $\mathcal{B}$ we define the monoidal finite-state transducer $\mathcal{T}' = \langle \Sigma^* \times \mathcal{M}, Q_L \times Q_R, \{s_L\} \times Q_R, Q_L \times \{s_R\}, \Delta' \rangle$, where

$$\Delta := \{\langle\langle l, r\rangle, \langle a, m\rangle, \langle l', r'\rangle\rangle \mid l' = \delta_L(l, a), r = \delta_R(r', a), m = \psi(l, a, r')\}.$$

It can be shown that $L(\mathcal{T}) = L(\mathcal{T}')$.

3.2 Complexity Analysis and Specialized Construction

When using the generic construction presented above we obtain the bound $|Q_R| \leq |2^Q|$ for the number of states of the right automaton $\mathcal{A}_R$. The number of (partial) functions mapping Q_R to Q is $(|Q| + 1)^{|Q_R|}$. Hence the number of states of $\mathcal{A}_L$ satisfies

$$|Q_L| \leq 2^{|Q|}(|Q| + 1)^{|Q_R|} \leq 2^{|Q|}(|Q| + 1)^{2^{|Q|}} = 2^{|Q|+2^{|Q|}\log(|Q|+1)}.$$

A characteristics of the above generic construction is the arbitrariness of the selection of a state q' in the second clause of the inductive definition of the states of the left automaton. Since each new state selection function introduced during the construction produces its own swarm of followers the question arises if a more principled approach to select q' helps to avoid any unnecessary blow-up and to reduce the upper bound on the number of states of $\mathcal{A}_L$.

To this end we apply the idea to compare paths of transducers using the lexicographic ordering. It has been successfully used in different uniformization problems related to transducers [3,8,9,12]. In the context of bimachines, we use the idea to specialize the generic selection mechanism described in the previous section.

First, we define the states of the left automaton $\mathcal{A}_L$ as pairs $p = \langle L, <_p \rangle$, see also Algorithm 1. As before, the left component L is always an element of $Q'_L := \{Succ_w(I) \mid w \in \Sigma^*\}$. The second component $<_p$ is a strict linear order on L. The ordering $<_p$ induces a *canonical* state selector function $\phi_{<_p}(R)$: if $L \cap R \neq \emptyset$, then $\phi_{<_p}(R)$ is defined as the $<_p$-minimal element of $L \cap R$. Otherwise $\phi_{<_p}(R)$ is undefined. Note that in this way state selector functions are always L-centered. Still, in order to follow this line, we need a method for defining the a-successor $q = \langle L', <_q \rangle$ of a state $p = \langle L, <_p \rangle$ in such a way that the $<_q$-minimal element of $L' \cap R'$ always represents a state q' with $\langle q, \langle a, . \rangle, q' \rangle \in \Delta$.

Given $\langle L, <_p \rangle$ and a state $r' \in L' := Succ_a(L)$ the set of a-predecessors of r' in L is defined as $Pred_{a,L}(r') := L \cap Pred_a(\{r'\})$. Note that, by the definition of L', each set $Pred_{a,L}(r')$ where $r' \in L'$ is non-empty. The *$<_p$-minimal a-predecessor of r' in L*, denoted $min\_pred_{a,L}(r')$, is the minimal element of $Pred_{a,L}(r')$ with respect to the ordering $<_p$.

We define the initial state, s_L, the set of states, Q_L, and the new transition function, δ_L, for the new definition of the left automaton, $\mathcal{A}_L$, as follows:

Algorithm 1. Direct construction of a bimachine. SeqTrans computes the transition of the left automaton; SelectMinimal determines the least element in the left state that is an element of the right state. Out computes the output produced by a left state, input character, and a right state.

```
Project($\Delta$)
@1 return$\{\langle p,a,q\rangle | \exists m \langle p,\langle a,m\rangle,q\rangle \in \Delta\}$

Reverse($\Delta$)
@1 return $\{\langle q,a,p\rangle \mid \langle p,a,q\rangle \in \Delta\}$

SetTrans($\Delta, P, a$)
@1 return $\{q \mid \exists p \in P(\langle p,a,q\rangle \in \Delta)\}$

SeqTrans($\Delta, P, a$)
@1 $S \leftarrow \langle\rangle$; $i \leftarrow 0$
@2 for $j = 0$ to $|P|-1$ do
@3   for $\langle P[j],a,q\rangle \in \Delta$ do
@4     if $q \notin S[0..i-1]$ then
@5       $S[i] \leftarrow q$
@6       $i \leftarrow i+1$
@7     fi
@8 return $S$

DetGeneric($\mathcal{A}, CmpTrans, i\_state$)
@1 $\langle \Sigma, Q, I, F, \Delta\rangle \leftarrow \mathcal{A}$
@2 $Q_D^{(-1)} \leftarrow \emptyset$; $Q_D^{(0)} \leftarrow \{i\_state\}$
@3 $\delta_D \leftarrow \emptyset$; $i \leftarrow 0$;
@3 while $Q_D^{(i)} \neq Q_D^{(i-1)}$ do
@4   $Q_D^{(i+1)} \leftarrow Q_D^{(i)}$
@5   for $P \in Q_D^{(i)} \setminus Q_D^{(i-1)}$ do
@6     for $a \in \Sigma$ do
@7       $\delta_D(P,a) \leftarrow CmpTrans(\Delta, P, a)$
@8       $Q_D^{(i+1)} \leftarrow Q_D^{(i+1)} \cup \{\delta_D(P,a)\}$
@9   $i \leftarrow i+1$
@10 return $\langle \Sigma, Q_D^{(i)}, \{I\}, Q_D^{(i)}, \delta_D\rangle$

SelectMinimal($L, R$)
@1 for $i = 0$ to $|L|-1$ do
@2   if $L[i] \in R$ then
@3     return L[i]
@4   fi
@5 done
@6 return $\perp$

Out($L, \delta_L, a, R, \delta_R, \Delta$)
@1 $p \leftarrow SelectMinimal(L, \delta_R(R,a))$
@2 $q \leftarrow SelectMinimal(\delta_L(L,a), R)$
@3 if $p = \perp$ or $q = \perp$ then
@4   return $\perp$
@5 else
@6   let $\langle p, \langle a,m\rangle, q\rangle \in \Delta$
@7 return $m$

ComputeBimachine($\mathcal{T}$)
@1 $\langle \Sigma \times \mathcal{M}, Q, I, F, \Delta\rangle \leftarrow \mathcal{T}$
@2 $\Delta_\Sigma \leftarrow Project(\Delta)$
@3 $\mathcal{A} \leftarrow \langle \Sigma, Q, I, F, \Delta_\Sigma\rangle$
@4 $\mathcal{A}^r \leftarrow \langle \Sigma, Q, F, I, Reverse(\Delta_\Sigma)\rangle$
@5 $\mathcal{A}_R \leftarrow DetGeneric(\mathcal{A}^r, SetTrans, F)$
@6 $I' \leftarrow$ sequence of $I$
@6 $\mathcal{A}_L \leftarrow DetGeneric(\mathcal{A}, SeqTrans, I')$
@7 $\langle \Sigma, Q_L, s_L, Q_L, \delta_L\rangle \leftarrow \mathcal{A}_L$
@8 $\langle \Sigma, Q_R, s_R, Q_R, \delta_R\rangle \leftarrow \mathcal{A}_R$
@9 for $\langle L, a, R\rangle \in Q_L \times \Sigma \times Q_R$ do
@10   $\psi(L,a,R) \leftarrow Out(L, \delta_L, a, R, \delta_R, \Delta)$
@11 return $\langle \mathcal{M}, \mathcal{A}_L, \mathcal{A}_R, \psi\rangle$
```

– $s_L := \langle I, <_0\rangle$ where $<_0$ is any fixed linear order of I.
– For $\langle L, <\rangle \in Q_L$ and $a \in \Sigma$ we define $\delta_L(\langle L, <\rangle, a) := (L', <')$ where $L' := Succ_a(L)$ and $<'$ is any linear order on L' satisfying the condition

$$\forall p', r' \in L' : p' \leq' r' \Rightarrow min\_pred_{a,L}(p') \leq min\_pred_{a,L}(r').$$

A linear order $<'$ of this form is obtained by starting with the elements of L' that have the $<$-minimal element q_{min} of L as their $<$-minimal a-predecessor (the ordering between these elements of L' is arbitrary). We then continue with the elements of L' that have the $<$-minimal element of $L \setminus \{q_{min}\}$ as their $<$-minimal a-predecessor, etc.

The following lemma shows that the new construction is a specialized version of the former construction described above.

Lemma 2. *Let $(L, <)$ and $(L', <')$ be as above. Let $\phi_<$ and $\phi_{<'}$ denote the canonical state selector functions corresponding to $<$ and $<'$, respectively. Let $R' \in Q_R$ and $R := Pred_a(R')$. Then $\phi_{<'}(R')$ is defined iff $\phi_<(R)$ is defined. Furthermore, if $q = \phi_<(R)$ and $q' = \phi'(R')$ are defined, then $\langle q, \langle a, m\rangle, q'\rangle \in \Delta$ for some $m \in M$.*

Proof. The Butterfly Lemma shows that

$$\phi_<(R) \text{ is defined } \overset{def}{\iff} L \cap R \neq \emptyset \underset{\text{Lemma}}{\overset{\text{Butterfly}}{\iff}} L' \cap R' \neq \emptyset \overset{def}{\iff} \phi_{<'}(R') \text{ is defined}$$

If $\phi_<(R)$ and $\phi_{<'}(R')$ are defined, then $q := \phi_<(R)$ is a $<$-minimal state of $L \cap R$ and $q' := \phi_{<'}(R')$ is a $<'$-minimal state of $L' \cap R'$. The Butterfly Lemma shows that there exist $p \in L \cap R$, $m \in M$, and a transition $\langle p, \langle a, m\rangle, q'\rangle \in \Delta$. Let p_0 be a $<$-minimal element of $L \cap R$ with this property. We claim that $p_0 = q$. The Butterfly Lemma shows that there exist $p' \in L' \cap R'$ and $m' \in M$ with $\langle q, \langle a, m'\rangle, p'\rangle \in \Delta$. From the minimality of q' we obtain $q' \leq' p'$, the definition of $\leq'$ shows that $p_0 \leq q$. Minimality of q implies that in fact $p_0 = q$. It follows that there exists a transition $\langle q, \langle a, m\rangle, q'\rangle \in \Delta$.

Theorem 2. *Given a functional real-time transducer $\mathcal{T} = \langle \Sigma, \mathcal{M}, Q, I, \Delta, F\rangle$ we can construct an equivalent bimachine $\mathcal{B} = \langle \mathcal{A}_L, \mathcal{A}_R, \psi\rangle$ such that the number of states of $\mathcal{A}_L$ is $O(|Q|!)$ and the number of states of $\mathcal{A}_R$ is $O(2^{|Q|})$.*

Proof. Clearly, the number of states of $\mathcal{A}_R$ is $O(2^{|Q|})$. Let $Seq(Q)$ denote the set of linearly ordered subsets of Q. In the specialized construction, the states of $\mathcal{A}_L$ can be represented as elements of $Seq(Q)$. We have

$$|Seq(Q)| = \sum_{k=0}^{|Q|} \binom{|Q|}{k} k! = \sum_{k=0}^{|Q|} \frac{|Q|!}{(|Q|-k)!} = 2|Q|! + \sum_{k=0}^{|Q|-2} \frac{|Q|!}{(|Q|-k)!}.$$

Taking into account that $(|Q|-k)! \geq 2^{|Q|-k}$ for $k \leq |Q|-2$ we obtain:

$$|Seq(Q)| = 2|Q|! + \sum_{k=0}^{|Q|-2} \frac{|Q|!}{(|Q|-k)!} \leq 2|Q|! + \sum_{k=0}^{|Q|-2} \frac{|Q|!}{2^{|Q|-k}} \leq 3|Q|!$$

thus showing that $|Q'_L| \leq |Seq(Q)| \leq 3|Q|!$. □

4 Remark on the Classical Bimachine Construction

The classical construction of bimachines [2] refers to the special case where $\mathcal{M} = \langle \Omega^*, \circ, \varepsilon\rangle$ is the free monoid generated by an alphabet Ω. As described in [7], but see also the proofs in [1,2,8], it departs from a pseudo-deterministic transducer, i.e. a transducer $\mathcal{T} = \langle \Sigma \times \Omega^*, Q, I, F, \Delta\rangle$ that can be considered as a deterministic finite-state automaton over the new alphabet $\Sigma \times \Omega^*$. This means that I contains a single state i and Δ is a finite graph of a function $Q \times (\Sigma \times \Omega^*) \to Q$.

The next step is the core of the construction. The goal is to construct an unambiguous transducer $\mathcal{T}'$ equivalent to $\mathcal{T}$. This is achieved by specializing the standard determinization construction for finite-state automata: the sets generated by the determinization procedure are split into two parts, a single *guessed positive state* – this is our positive hypothesis for the successful path to be followed, and a set of *negative states* – these are the alternative hypotheses that must all fail in order for our positive hypothesis to be confirmed. Formally, the states in the resulting transducer are pairs $\langle p, N\rangle \in Q \times 2^Q$. The initial state is $i' = \langle i, \emptyset\rangle$. The algorithm inductively defines transitions in Δ' and states in Q'. Let $\prec_{lex}$ denote the lexicographic order on Σ^*. For a generated state $\langle p, N\rangle$ and each transition $\langle p, \langle a, v\rangle, p'\rangle \in \Delta$ we obtain a transition

$$\begin{gathered}\langle\langle p, N\rangle, \langle a, v\rangle, \langle p', N'\rangle\rangle \in \Delta', \text{ where} \\ N' = Succ_a(N) \cup \{q \mid \exists v' \prec_{lex} v(\langle p, \langle a, v'\rangle, q\rangle \in \Delta\}.\end{gathered}$$

The pair $\langle p', N'\rangle$ is added to Q'. Intuitively, this transition makes a guess about the lexicographically smallest continuation of the output that can be followed to a final state $f \in F$. Accordingly, all transitions that have the same input character, a, but lexicographically smaller output, are implicitly assumed to fail. To reflect this, we add those states to the set of negative hypotheses, N'. To maintain the previously accumulated negative hypotheses along the path to $\langle p, N\rangle$ the a-successors of N are added to N'. Following these lines, the set of final states of $\mathcal{T}'$ is defined as:

$$F' = \{\langle f, N\rangle \mid f \in F \text{ and } N \cap F = \emptyset\}.$$

Note, that $\langle f, N\rangle$ becomes final only if $f \in F$ and there is no final state $n \in N$ reached with smaller output on a parallel path. It can be formally shown [7], that this construction indeed leads to an unambiguous transducer:

$$\mathcal{T}' = \langle \Sigma \times \Omega^*, Q', \{i'\}, F', \Delta'\rangle$$

equivalent to $\mathcal{T}$.

The final step is to convert the (trimmed part of) $\mathcal{T}'$ in an equivalent bimachine. This can be easily done by a determinization of $\mathcal{A}_L = \mathcal{A}_{\mathcal{T}',D}$ and $\mathcal{A}_R = \mathcal{A}^{rev}_{\mathcal{T}',D}$ and defining an appropriate output function $\psi : Q_L \times \Sigma \times Q_R \to \Omega^*$. The following points have to be stressed about this construction.

Remark 2. The states of the left automaton are sets $L \subseteq 2^{Q \times 2^Q}$. Yet, these sets have an inner structure that enables a non-trivial upper bound on their number, $|Q_L| = O(|Q|! \exp(|Q| + 1))$. We sketch the main points of the proof:

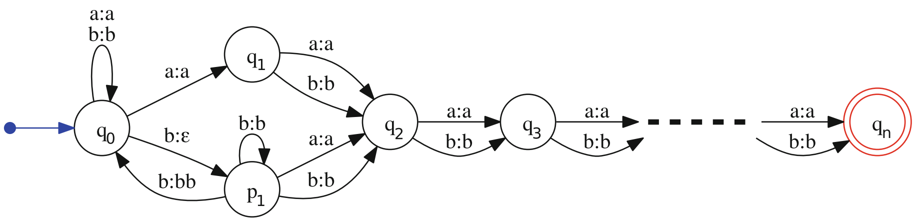
– First, if $q = \langle p', N'\rangle \in L$, then, since q is co-accessible in $\mathcal{T}'$, it follows that $p' \notin N'$. Assume now that $\langle p', N'\rangle, \langle p'', N''\rangle \in L$.
– Let $\langle p', N'\rangle \neq \langle p'', N''\rangle \in \mathcal{T}'$ be distinct states accessible via the same input word $u \in \Sigma^*$. Then either $\{p'\} \cup N' \subseteq N''$ or $\{p''\} \cup N'' \subseteq N'$ (the proof uses a simple induction on $|u|$).

- Let $\langle p', N' \rangle \neq \langle p'', N'' \rangle \in L$ be distinct. Then, $\langle p', N' \rangle$ and $\langle p'', N'' \rangle$ are all accessible in $\mathcal{T}$ via a common word $u \in \Sigma$. By the above argument we can assume that $\{p'\} \cup N' \subseteq N''$. By the first argument we have that $p'' \notin N''$ and therefore $\{p'\} \cup N' \subsetneq \{p''\} \cup N''$.
- This proves that every left state $L = \{\langle p_i, N_i \rangle \mid i \leq |L|\}$ induces a linear order on $\{p_1, \ldots, p_{|L|}\}$ by defining $p_i < p_j$ if and only if $\{p_i\} \cup N_i \subsetneq \{p_j\} \cup N_j$. This shows that the left states L arise as linear orders of the states $\{p_1, \ldots, p_{|L|}\}$ and some additional elements $q \in Q \setminus \{p_1, \ldots, p_{|L|}\}$ that belong to some N_i. By the third point we can assign each such state q to the least N_i with $q \in N_i$. By the linear order it will belong to all the bigger sets $\{p_j\} \cup N_j$.
- With this remarks, the problem becomes a combinatorial one and using ideas similar to those in the proof of Theorem 2 one can prove that

$$|Q_L| \leq \sum_{k=1}^{|Q|} \binom{|Q|}{k} k!(k+1)^{|Q|-k} = |Q|! \sum_{k=1}^{|Q|} \frac{(k+1)^{|Q|-k}}{(|Q|-k)!}.$$

Looking at the term for $k = |Q|$, one sees that the upper bound for Q_L is at least $Q!$. On the other hand, since $k \leq |Q|$, substituting $k+1$ with $|Q|+1$ we easily get that: $|Q_L| \leq |Q|! \sum_{k=1}^{|Q|} \frac{(|Q|+1)^{|Q|-k}}{(|Q|-k)!} \leq |Q|! \exp(|Q|+1)$.

Remark 3. Since the transducer $\mathcal{T}'$ is unambiguous any two states $L \in Q_L$ and $R \in Q_R$ have at most one common element. This shows that for each $L \in Q_L$, there is a unique L-centered function ϕ_L and therefore our construction would find exactly this function if run on $\mathcal{T}'$. Thus in this case the output function $\psi : Q_L \times \Sigma \times Q_R \to \Omega^*$ will be defined in exactly the same way.



	$\lvert Q\rvert$	$\lvert Q_{\text{pseudo-det}}\rvert$	$\lvert Q_L\rvert$	$\lvert Q_R\rvert$
classical construction	$n+2$	$2^n + \binom{n+1}{2}$	$2^{n+1}-1$	$n+2$
new construction		$n.a.$	$2n+1$	$n+2$

Fig. 1. A class of ambiguous finite-state transducers representing the rational functions $\{\langle a,a\rangle, \langle b,b\rangle\}^*\{\langle a,a\rangle, \langle b,\varepsilon\rangle\}\{\langle a,a\rangle, \langle b,b\rangle\}^{n-1}$, which deletes the n-th character from right-to-left if it is a b. The table shows the number of states of the source transducer, the pseudo-deterministic transducer, the left and the right automaton of the bimachine built by the standard and the new constructions.

Remark 4. The classical construction is starting from a pseudo-deterministic transducer. However, if $\mathcal{T}$ is an arbitrary real-time transducer the initial conversion to a pseudo-deterministic transducer may cause an exponential blow-up. In contrast, our constructions can be applied directly to arbitrary real-time transducers and thus avoids this blow-up. See Fig. 1 for an example.

5 Conclusion

In this paper we introduced a new generic algorithm and a specialization for building bimachines from functional finite-state transducers. The generic procedure is conceptually simple. Both constructions avoid the preparatory steps used in the classical construction, namely pseudodeterminization and disambiguation.

For the specialized construction we derived an upper bound on the size of the bimachine. We showed that this construction is asymptotically not worse than the classical construction. Moreover we presented a class of transducers for which the classical construction generates a bimachine with exponentially more states than the new construction.

The generic construction described in Subsect. 3.1 is not based on any order of the successful paths. It provides a simple and general algorithmic scheme for bimachine constructions, leaving room for other specialization, with new path selection strategies that might lead to even smaller bimachines. The study of optimal path selection strategies is a point for future research.

References

1. Berstel, J.: Transductions and Context-Free Languages. Springer Fachmedien Wiesbaden GmbH, Wiesbaden (1979)
2. Eilenberg, S.: Automata, Languages and Machines. Academic Press, New York and London (1974)
3. Filiot, E., Servais, F.: Visibly pushdown transducers with look-ahead. In: Bieliková, M., Friedrich, G., Gottlob, G., Katzenbeisser, S., Turán, G. (eds.) SOFSEM 2012. LNCS, vol. 7147, pp. 251–263. Springer, Heidelberg (2012). doi:10.1007/978-3-642-27660-6_21
4. Kempe, A.: Part-of-speech tagging with two sequential transducers. In: Yu, S., Păun, A. (eds.) CIAA 2000. LNCS, vol. 2088, pp. 337–339. Springer, Heidelberg (2001). doi:10.1007/3-540-44674-5_34
5. Mohri, M.: On some applications of finite-state automata theory to natural language processing. J. Nat. Lang. Eng. **2**, 1–20 (1996)
6. Mohri, M.: Finite-state transducers in language and speech processing. Comput. Linguist. **23**(2), 269–311 (1997)
7. Roche, E., Schabes, Y.: Finite-State Language Processing. MIT Press, Cambridge (1997)
8. Sakarovitch, J.: Elements of Automata Theory. Cambridge University Press, Cambridge (2009)
9. Sakarovitch, J., de Souza, R.: Lexicographic decomposition of k-valued transducers. Theor. Comp. Sys. **47**(3), 758–785 (2010). http://dx.doi.org/10.1007/s00224-009-9206-6

10. Santean, N.: Bimachines and structurally-reversed automata. J. Automata Lang. Comb. **9**(1), 121–146 (2004)
11. Schützenberger, M.P.: A remark on finite transducers. Inf. Control **4**, 185–196 (1961)
12. Souza, R.: A note on bimachines. In: 1a Escola de Informática Teórica e Métodos Formais, Natal - RN, pp. 83–92, November 2016