Squaring transducers

An efficient procedure for deciding functionality and sequentiality of transducers

Marie-Pierre Béal[1], Olivier Carton[1],
Christophe Prieur[2], and Jacques Sakarovitch[3]

[1] Institut Gaspard Monge, Université Marne-La Vallée
[2] LIAFA, Université Paris 7 / CNRS
[3] Laboratoire Traitement et Communication de l'Information, ENST / CNRS

Abstract. We described here a construction on transducers that give a new conceptual proof for two classical decidability results on transducers: it is decidable whether a finite transducer realizes a functional relation, and whether a finite transducer realizes a sequential relation. A better complexity follows then for the two decision procedures.

In this paper we give a new presentation and a conceptual proof for two classical decision results on finite transducers.

Transducers are finite automata with input and output; they realize thus relations between words, the so-called *rational relations*. Eventhough they are a very simple model of machines that compute relations — they can be seen as 2-tape *1-way* Turing machines — most of the problems such as equivalence or intersection are easily shown to be equivalent to the Post Correspondence Problem and thus undecidable. The situation is drastically different for transducers that are *functional*, that is, transducers that realize functions, and the above problems become then easily decidable. And this is of interest because of the following result.

Theorem 1. [12] *Functionality is a decidable property for finite transducers.*

Among the functional transducers, those which are *deterministic in the input* (they are called *sequential*) are probably the most interesting, both from a pratical and from a theoretical point of view: they correspond to machines that can really and easily be implemented. A rational function is *sequential* if it can be realized by a sequential transducer. Of course, a non sequential transducer may realize a sequential function and this occurrence is known to be decidable.

Theorem 2. [7] *Sequentiality is a decidable property for rational functions.*

The original proofs of these two theorems are based on what could be called a "pumping" principle, implying that a word which contradicts the property may be chosen of a bounded length, and providing thus directly decision procedures of exponential complexity. Theorem 1 was published again in [4], with exactly the same proof, hence the same complexity.

Later, it was proved that the functionality of a transducer can be decided in polynomial time, as a particular case of a result obtained by reduction to another decision problem on another class of automata ([10, Theorem 2]).

With this communication, we shall see how a very natural construction performed on the *square of the transducer* yields a decision procedure for the two properties, that is, it can be read on the result of the construction whether the property holds or not.

The size of the object constructed for deciding functionality is *quadratic* in the size of the considered transducer. In the case of sequentiality, one has to be more subtle for the constructed object may be too large. But it is shown that it can be decided in *polynomial* time whether this object has the desired property.

Due to the short space available on the proceedings, the proofs of the results are omited here and will be published in a forthcoming paper.

1 Preliminaries

We basically follow the definitions and notation of [9, 2] for automata.

The set of *words* over a finite alphabet A, *i.e. the free monoid* over A, is denoted by A^*. Its identity, or *empty word* is denoted by 1_{A^*}.

An *automaton $\mathcal{A}$ over a finite alphabet A*, noted $\mathcal{A} = \langle Q, A, E, I, T \rangle$, is a directed graph labelled by elements of A; Q is the set of vertices, called *states*, $I \subset Q$ is the set of *initial* states, $T \subset Q$ is the set of *terminal* states and $E \subset Q \times A \times Q$ is the set of labelled *edges* called *transitions*. The automaton $\mathcal{A}$ is *finite* if Q is finite.

The definition of automata as labelled graphs extends readily to automata over any monoid: an *automaton $\mathcal{A}$ over M*, noted $\mathcal{A} = \langle Q, M, E, I, T \rangle$, is a directed graph the edges of which are labelled by elements of the monoid M. A *computation* is a path in the graph $\mathcal{A}$; its label is the product of the label of its transitions. A computation is *successful* if it begins with an initial state and ends with a final state. The *behaviour* of $\mathcal{A}$ is the subset of M consisting of the labels of the successful computations of $\mathcal{A}$.

A state of $\mathcal{A}$ is said to be *accessible* if it belongs to a computation that begins with an initial state; it is *useful* if it belongs to a successful computation. The automaton $\mathcal{A}$ is *trim* if all of its states are useful. The accessible part and the useful part of a finite automaton $\mathcal{A}$ are easily computable from $\mathcal{A}$.

An automaton $\mathcal{T} = \langle Q, A^* \times B^*, E, I, T \rangle$ over a direct product $A^* \times B^*$ of two free monoids is called *transducer* from A^* to B^*. The behaviour of a transducer $\mathcal{T}$ is thus (the graph of) a relation α from A^* into B^*: α is said to be *realized* by $\mathcal{T}$. A relation is *rational* (*i.e.* its graph is a rational subset of $A^* \times B^*$) if and only if it is realized by a finite transducer.

It is a slight generalization — that does not increase the generating power of the model — to consider transducers $\mathcal{T} = \langle Q, A^* \times B^*, E, I, T \rangle$ where I and T are not *subsets* of Q (*i.e.* functions from Q into $\{0, 1\}$) but *functions* from Q

into $B^* \cup \emptyset$ (the classical transducers are those for which the image of a state by I or T is either $\emptyset$ or 1_{B^*}).

A transducer is said to be *real-time* if the label of every transition is a pair (a, v) where a is letter of A, the *input* of the transition, and v a word over B, the *output* of the transition, and if for any states p and q and any letter a there is at most one transition from p to q whose input is a. Using classical algorithms from automata theory, any transducer $\mathcal{T}$ can be transformed into a transducer that is real-time if $\mathcal{T}$ realizes a function ([9, Th. IX.5.1], [2, Prop. III.7.1]).

If $\mathcal{T} = \langle Q, A^* \times B^*, E, I, T \rangle$ is a real-time transducer, the *underlying input automaton* of $\mathcal{T}$ is the automaton $\mathcal{A}$ over A obtained from $\mathcal{T}$ by forgetting the second component of the label of every transition and by replacing the functions I and T by their respective domains. The language recognized by $\mathcal{A}$ is the domain of the relation realized by $\mathcal{T}$.

We call *sequential* a transducer that is real-time, functional, and whose underlying input automaton is *deterministic*. A function α from A^* into B^* is *sequential* if it can be realized by a sequential transducer. It has to be acknowlegded that *this is not the usual terminology*: what we call "sequential" (transducers or functions) have been called "*subsequential*" since the seminal paper by Schützenberger [13] — *cf.* [2, 5, 7, 8, 11, *etc.*]. There are good reasons for this change of terminology that has already been advocated by V. Bruyère and Ch. Reutenauer: "the word *subsequential* is unfortunate since these functions should be called simply *sequential*" ([5]). Someone has to make the first move.

2 Squaring automata and ambiguity

Before defining the square of a transducer, we recall what is the *square of an automaton* and how it can be used to decide whether an automaton is unambiguous or not. A trim automaton $\mathcal{A} = \langle Q, A, E, I, T \rangle$ is *unambiguous* if any word it accepts is the label of a unique successful computation in $\mathcal{A}$.

Let $\mathcal{A}' = \langle Q', A, E', I', T' \rangle$ and $\mathcal{A}'' = \langle Q'', A, E'', I'', T'' \rangle$ be two automata on A. The *Cartesian product* of $\mathcal{A}'$ and $\mathcal{A}''$ is the automaton $\mathcal{C}$ defined by

$$\mathcal{C} = \mathcal{A}' \times \mathcal{A}'' = \langle Q' \times Q'', A, E, I' \times I'', T' \times T'' \rangle$$

where E is the set of transitions defined by

$$E = \{((p', p''), a, (q', q'')) \mid (p', a, q') \in E' \quad \text{and} \quad (p'', a, q'') \in E''\} \ .$$

Let $\mathcal{A} \times \mathcal{A} = \langle Q \times Q, A, F, I \times I, T \times T \rangle$ be the Cartesian product of the automaton $\mathcal{A} = \langle Q, A, E, I, T \rangle$ with itself; the set F of transitions is defined by:

$$F = \{((p, r), a, (q, s)) \mid (p, a, q), (r, a, s) \in E\} \ .$$

Let us call *diagonal* of $\mathcal{A} \times \mathcal{A}$ the sub-automaton $\mathcal{D}$ of $\mathcal{A} \times \mathcal{A}$ determined by the diagonal D of $Q \times Q$, *i.e.* $D = \{(q, q) \mid q \in Q\}$, as set of states. The states and transitions of $\mathcal{A}$ and $\mathcal{D}$ are in bijection, hence $\mathcal{A}$ and $\mathcal{D}$ are equivalent.

Lemma 1. [3, Prop. IV.1.6] *A trim automaton* $\mathcal{A}$ *is* unambiguous *if and only if the* trim part *of* $\mathcal{A}\times\mathcal{A}$ *is equal to* $\mathcal{D}$*.* ■

Remark that as (un)ambiguity, *determinism* can also be described in terms of Cartesian square, by a simple rewording of the definition: *a trim automaton* $\mathcal{A}$ *is* deterministic *if and only if the* accessible part *of* $\mathcal{A}\times\mathcal{A}$ *is equal to* $\mathcal{D}$*.*

3 Product of an automaton by an action

We recall now what is an *action*, how an action can be seen as an automaton, and what can be then defined as the product of a (normal) automaton by an action. We end this section with the definition of the specific action that will be used in the sequel.

Actions. A (right) *action of a monoid* M *on a set* S is a mapping $\delta\colon S\times M \to S$ which is consistent with the multiplication in M:

$$\forall s \in S\,,\ \forall m, m' \in M \qquad \delta(s, 1_M) = s \quad \text{and} \quad \delta(\delta(s,m), m') = \delta(s, m\,m') \ .$$

We write $s \cdot m$ rather than $\delta(s,m)$ when it causes no ambiguity.

Actions as automata. An action δ of M on a set S with s_0 as distinguished element may then be seen as an automaton on M (without terminal states):

$$\mathcal{G}_\delta = \langle\, S, M, E, s_0 \,\rangle$$

is defined by the set of transitions $E = \{(s, m, s\cdot m) \mid s \in S\,,\ m \in M\}$.

Note that, as both S and M are usually infinite, the automaton $\mathcal{G}_\delta$ is "doubly" infinite: the *set of states* is infinite, and, for every state s, the *set of transitions* whose origin is s is infinite as well.

Product of an automaton by an action. Let $\mathcal{A} = \langle\, Q, M, E, I, T \,\rangle$ be a (finite trim) automaton on a monoid M and δ an action of M on a (possibly infinite) set S. The product of $\mathcal{A}$ and $\mathcal{G}_\delta$ is the automaton on M:

$$\mathcal{A}\times\mathcal{G}_\delta = \langle\, Q\times S, M, F, I\times\{s_0\}, T\times S \,\rangle$$

the transitions of which are defined by

$$F = \{((p,s), m, (q, s\cdot m)) \mid s \in S\,,\ (p,m,q) \in E\} \ .$$

We shall call *product of* $\mathcal{A}$ *by* δ, and denote by $\mathcal{A}\times\delta$, the *accessible part* of $\mathcal{A}\times\mathcal{G}_\delta$.

The projection on the first component induces a bijection between the transitions of $\mathcal{A}$ whose origin is p and the transitions of $\mathcal{A}\times\delta$ whose origin is (p,s), for any p in Q and any (p,s) in $\mathcal{A}\times\delta$. The following holds (by induction on the length of the computations):

$$(p,s) \xrightarrow[\mathcal{A}\times\delta]{m} (q,t) \quad \Longrightarrow \quad t = s\cdot m \ .$$

We call *value of a state* (p, s) of $\mathcal{A}\times\delta$ the element s of S. We shall say that the product $\mathcal{A}\times\delta$ itself *is a valuation* if the projection on the first component is a 1-to-1 mapping between the states of $\mathcal{A}\times\delta$ and the states of $\mathcal{A}$.

Remark 1. Let us stress again the fact that $\mathcal{A}\times\delta$ is the *accessible part* of $\mathcal{A}\times\mathcal{G}_\delta$. This makes possible that it may happen that $\mathcal{A}\times\delta$ is finite eventhough $\mathcal{G}_\delta$ is infinite (*cf.* Theorem 5).

The "Advance or Delay" action. Let B^* be a free monoid and let us denote by H_B the set $H_B = (B^*\times 1_{B^*}) \cup (1_{B^*}\times B^*) \cup \{\mathbf{0}\}$. A mapping $\psi\colon B^*\times B^* \to H_B$ is defined by:

$$\forall u, v \in B^* \qquad \psi(u, v) = \begin{cases} (v^{-1}u, 1_{B^*}) & \text{if} \quad v \text{ is a prefix of } u \\ (1_{B^*}, u^{-1}v) & \text{if} \quad u \text{ is a prefix of } v \\ \mathbf{0} & \text{otherwise} \end{cases}$$

Intuitively, $\psi(u, v)$ tells either how much the first component u is ahead of the second component v, or how much it is late, or if u and v are not prefixes of a common word. In particular, $\psi(u, v) = (1_{B^*}\times 1_{B^*})$ if, and only if, $u = v$.

Lemma 2. *The mapping ω_B from $H_B\times(B^*\times B^*)$ into H_B defined by:*

$$\forall (f, g) \in H_B \setminus \mathbf{0} \qquad \omega_B((f, g), (u, v)) = \psi(f\,u, g\,v) \quad \textit{and} \quad \omega_B(\mathbf{0}, (u, v)) = \mathbf{0}$$

*is an action, which will be called the "*Advance or Delay*" (or "*AD *") action (relative to B^*) and will thus be denoted henceforth by a dot.* ∎

Remark 2. The transition monoid of ω_B is isomorphic to $B^*\times B^*$ if B has at least two letters, to $\mathbb{Z}$ if it has only one letter. (We have denoted by $\mathbf{0}$ the absorbing element of H_B under ω_B in order to avoid confusion with 0, the identity element of the monoid $\mathbb{Z}$).

4 Deciding functionality

Let $\mathcal{T} = \langle Q, A^*\times B^*, E, I, T\rangle$ be a real-time trim transducer such that the output of every transition is a single word of B^* — recall that this is a necessary condition for the relation realized by $\mathcal{T}$ to be a function. The transducer $\mathcal{T}$ is not functional if and only if there exist two *distinct* computations:

$$q'_0 \xrightarrow[\mathcal{T}]{a_1/u'_1} q'_1 \cdots \xrightarrow[\mathcal{T}]{a_n/u'_n} q'_n \qquad \text{and} \qquad q''_0 \xrightarrow[\mathcal{T}]{a_1/u''_1} q''_1 \cdots \xrightarrow[\mathcal{T}]{a_n/u''_n} q''_n$$

with $u'_1\,u'_2 \ldots u'_n \neq u''_1\,u''_2 \ldots u''_n$. There exists then at least one i such that $u'_i \neq u''_i$, and thus such that $q'_i \neq q''_i$.

This implies, by projection on the first component, that the underlying input automaton $\mathcal{A}$ of $\mathcal{T}$ is *ambiguous*. But it may be the case that $\mathcal{A}$ is ambiguous and $\mathcal{T}$ still functional, as it is shown for instance with the transducer $\mathcal{Q}_1$ represented on the top of Figure 1 (*cf.* [2]). We shall now carry on the method of Cartesian square of section 2 from automata to transducers.

Cartesian square of a real-time transducer. By definition, the *Cartesian product* of $\mathcal{T}$ by itself is the transducer $\mathcal{T}\times\mathcal{T}$ from A^* into $B^*\times B^*$:

$$\mathcal{T}\times\mathcal{T} = \langle\, Q\times Q, A^*\times(B^*\times B^*), F, I\times I, T\times T\,\rangle$$

whose transitions set F is defined by:

$$F = \{((p,r),(a,(u',u'')),(q,s)) \mid \quad (p,(a,u'),q) \quad \text{and} \quad (r,(a,u''),s)\in E\} \ .$$

The underlying input automaton of $\mathcal{T}\times\mathcal{T}$ is the square of the underlying input automaton $\mathcal{A}$ of $\mathcal{T}$. If $\mathcal{A}$ is unambiguous, then $\mathcal{T}$ is functional, and the trim part of $\mathcal{A}\times\mathcal{A}$ is reduced to its diagonal.

An effective characterization of functionality. The transducer $\mathcal{T}\times\mathcal{T}$ is an automaton on the monoid $M = A^*\times(B^*\times B^*)$. We can consider that the AD action is an action of M on H_B, by forgetting the first component. We can thus make the product of $\mathcal{T}\times\mathcal{T}$, or of any of its subautomata, by the AD action ω_B.

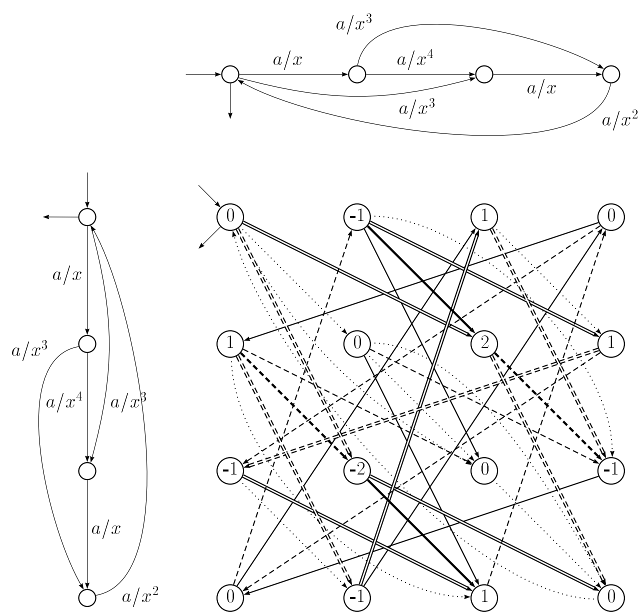


Fig. 1. Cartesian square of $\mathcal{Q}_1$, valued by the product with the action $\omega_{\{x\}}$.
As the output alphabet has only one letter, $H_{\{x\}}$ is identified with $\mathbb{Z}$ and the states are labelled by an integer. Labels of transitions are not shown: the input is always a and is kept implicit; an output of the form (x^n, x^m) is coded by the integer $n-m$ which is itself symbolised by the drawing of the arrow: a dotted arrow for 0, a simple solid arrow for +1, a double one for +2 and a bold one for +3; and the corresponding dashed arrows for the negative values.

Theorem 3. *A transducer $\mathcal{T}$ from A^* into B^* is functional if and only if the product of* the trim part $\mathcal{U}$ *of the Cartesian square $\mathcal{T} \times \mathcal{T}$ by the AD action ω_B is* a valuation *of $\mathcal{U}$ such that the value of any final state is $(1_{B^*}, 1_{B^*})$.* ■

Figure 1 shows the product of the Cartesian square of a transducer $\mathcal{Q}_1$ by the AD action[1].

Let us note that if α is the relation realized by $\mathcal{T}$, the transducer obtained from $\mathcal{T} \times \mathcal{T}$ by forgeting the first component is a transducer from B^* into itself that realizes the composition product $\alpha \circ \alpha^{-1}$. The conditon expressed may then seen as a condition for $\alpha \circ \alpha^{-1}$ being the identity, which is clearly a condition for the functionality of α.

5 Deciding sequentiality

The original proof of Theorem 2 goes indeed in three steps: first, sequential functions are characterized by a property expressed by means of a *distance function*, then this property (on the function) is proved to be equivalent to a property on the transducer, and finally a pumping-lemma like procedure is given for deciding the latter property (*cf.* [7, 2]). We shall see how the last two steps can be replaced by the computation of the product of the Cartesian square of the transducer by the AD action. We first recall the first step.

5.1 A quasi-topological characterization of sequential functions

If f and g are two words, we denote by $f \wedge g$ the *longuest prefix common* to f and g. The free monoid is then equipped with the *prefix distance*

$$\forall f, g \in A^* \qquad \mathsf{d_p}(f, g) = |f| + |g| - 2|f \wedge g| \ .$$

In other words, if $f = h\, f'$ and $g = h\, g'$ with $h = f \wedge g$, then $\mathsf{d_p}(f, g) = |f'| + |g'|$.

Definition 1. *A function $\alpha\colon A^* \to B^*$, is said to be* uniformly diverging[2] *if for every integer n there exists an integer N which is greater than the prefix distance of the images by α of any two words (in the domain of α) whose prefix distance is smaller than n, i.e.*

$$\forall n \in \mathbb{N}\,,\ \exists N \in \mathbb{N}\,,\ \forall f, g \in \mathsf{Dom}\,\alpha \qquad \mathsf{d_p}(f, g) \leqslant n \quad \Longrightarrow \quad \mathsf{d_p}(f\alpha, g\alpha) \leqslant N \ .$$

Theorem 4. [7, 13] *A rational function is sequential if, and only if it is uniformly diverging.*

Remark 3. The characterization of sequential functions by uniform divergence holds in the larger class of functions whose inverse preserves rationality. This is a generalization of a theorem of Ginsburg and Rose due to Choffrut, a much stronger result, the full strength of which will not be of use here (*cf.* [5, 8]).

---

[1] It turns out that, in this case, the trim part is equal to the whole square.

[2] After [7] and [2], the usual terminology is "function with *bounded variation*". We rather avoid an expression that is already used, with an other meaning, in other parts of mathematics.

5.2 An effective characterization of sequential functions

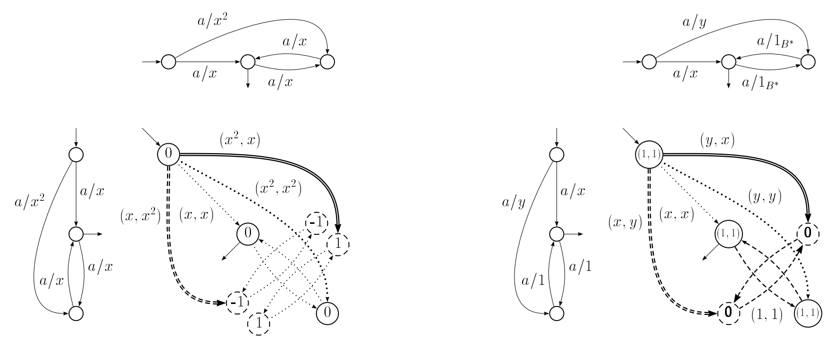
Theorem 5. *A transducer $\mathcal{T}$ realizes a sequential function if, and only if the product of* the accessible part $\mathcal{V}$ *of* $\mathcal{T} \times \mathcal{T}$ *by the AD action* ω_B

i) *is finite;*

ii) *has the property that if a state with value* $\mathbf{0}$ *belongs to a cycle in* $\mathcal{V}$*, then the label of that cycle is* $(1_{B^*}, 1_{B^*})$*.* ■

The parallel between automata and transducers is now to be emphasized. Unambiguous (resp. deterministic) automata are characterized by a condition on the trim (resp. accessible) part of the Cartesian square of the automaton whereas functional transducers (resp. transducers that realize sequential functions) are characterized by a condition on the product by ω_B of the trim (resp. accessible) part of the Cartesian square of the transducer.

Figure 2 shows two cases where the function is sequential: in (a) since the accessible part of the product is finite and no state has value $\mathbf{0}$; in (b) since the accessible part of the product is finite as well and the states whose value is $\mathbf{0}$ all belong to a cycle every transition of which is labelled by $(1_{B^*}, 1_{B^*})$.



(a) (b)

Fig. 2. Two transducers that realize sequential functions.

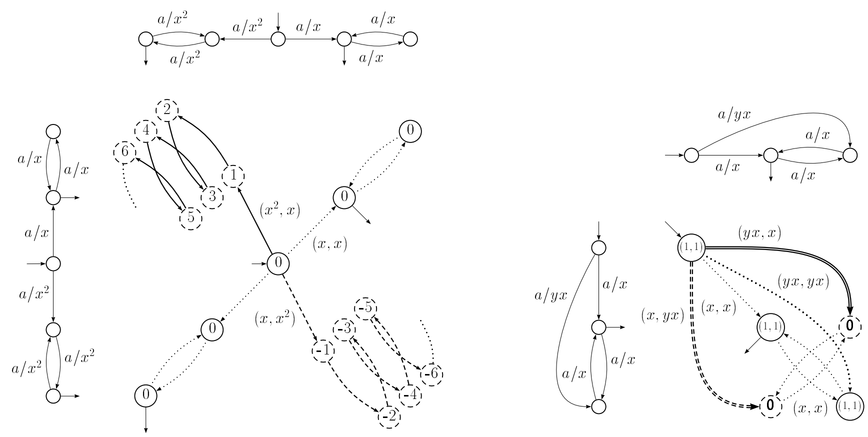
Figure 3 shows two cases where the function is not sequential: in (a) since the accessible part of the product is infinite; in (b) since although the accessible part of the product is finite some states whose value is $\mathbf{0}$ belong to a cycle whose label is different from $(1_{B^*}, 1_{B^*})$.

The following lemma is the key to the proof of Theorem 5 as well as to its effectivity.

Lemma 3. *Let* $w = (1_{B^*}, z)$ *be in* $H_B \setminus \mathbf{0}$ *and* (u, v) *in* $B^* \times B^* \setminus (1_{B^*}, 1_{B^*})$*. Then the set* $\{w \cdot (u, v)^n \mid n \in \mathbb{N}\}$ *is finite and does not contain* $\mathbf{0}$ *if, and only if,* u *and* v *are congugate words and* z *is a prefix of a power of* u*.* ■

Remark 4. The original proof of Theorem 2 by Ch. Choffrut goes by the definition of the so-called *twinning property* (*cf.* [2, p. 128]). It is not difficult to check that two states p and q of a real-time transducer $\mathcal{T}$ are (non trivially) *twinned* when: i) (p, q) is accessible in $\mathcal{T}\times\mathcal{T}$; ii) (p, q) belongs to a cycle in $\mathcal{V}$ every transition of which is not labelled by $(1_{B^*}, 1_{B^*})$; iii) (p, q) has not the value $\mathbf{0}$ in the product of $\mathcal{V}$ by ω_B.

It is then shown that a transducer realizes a sequential function if, and only if, every pair of its states has the twinning property.



(a) (b)

Fig. 3. Two transducers that realize non sequential functions.

6 The complexity issue

The "*size*" of an automaton $\mathcal{A}$ (on a free monoid A^*) is measured by the number m of transitions. (The size $|A| = k$ of the (input) alphabet is seen as a constant.) The size of a transducer $\mathcal{T}$ will be measured by *the sum of the sizes* of its transitions where the size of a transition $(p, (u, v), q)$ is the length $|u\,v|$. It is denoted by $|\mathcal{T}|$.

The size of the transducer $\mathcal{T}\times\mathcal{T}$ is $|\mathcal{T}|^2$ and the complexity to build it is proportional to that size. The complexity of determining the trim part as well as the accessible part is linear in the size of the transducer.

Deciding whether the product of the *trim part* $\mathcal{U}$ of $\mathcal{T}\times\mathcal{T}$ by the AD action ω_B is a valuation of $\mathcal{U}$ (and if the value of any final state is $(1_{B^*}, 1_{B^*})$) is again linear in the size of $\mathcal{U}$. Hence deciding whether a transducer $\mathcal{T}$ is functional is quadratic in the size of the transducer. Note that the same complexity is also established in [6].

The complexity of a decision procedure for the sequentiality of a function, based on Theorem 5, is polynomial. However, this is less straightforward to establish than functionality, for the size of the product $\mathcal{V}\times\omega_B$ may be exponential.

One first checks whether the label of every cycle in $\mathcal{V}$ is of the form (u,v) with $|u| = |v|$. It suffices to check it on a base of simple cycles and this can be done by a deep-first search in $\mathcal{V}$. Let us call *true cycle* a cycle which is not labelled by $(1_{B^*}, 1_{B^*})$ and let $\mathcal{W}$ be the subautomaton of $\mathcal{V}$ consisting of states from which a true cycle is accessible. By Theorem 5, if suffices to consider the product $\mathcal{W}\times\omega_B$. This product may still be of exponential size. However one does not construct it entirely. For every state of $\mathcal{W}$, the number of values which are to be considered in $\mathcal{W}\times\omega_B$ may be bounded by the size of $\mathcal{T}$. This yields an algorithm of polynomial complexity in order to decide the sequentiality of the function realized by $\mathcal{T}$.

In [1], it is shown directly that the twinning property is decidable in polynomial time.

References

1. M.-P. Béal and O. Carton: Determinization of transducers over finite and infinite words, *to appear.*
2. J. Berstel: *Transductions and context-free languages*, Teubner, 1979.
3. J. Berstel and D. Perrin: *Theory of codes*, Academic Press, 1985.
4. M. Blattner and T. Head: Single valued a-transducers, *J. Computer System Sci.* **7** (1977), 310–327.
5. V. Bruyère and Ch. Reutenauer: A proof of Choffrut's theorem on subsequential functions, *Theoret. Comput. Sci.* **215** (1999), 329–335.
6. O. Carton, Ch. Choffrut and Ch. Prieur: How to decide functionality of rational relations on infinite words, *to appear.*
7. Ch. Choffrut: Une caractérisation des fonctions séquentielles et des fonctions sous-séquentielles en tant que relations rationnelles, *Theoret. Comput. Sci.* **5** (1977), 325–337.
8. Ch. Choffrut: A generalization of Ginsburg and Rose's characterization of g-s-m mappings, *in Proc. of ICALP'79* (H. Maurer, Ed.), *Lecture Notes in Comput. Sci.* **71** (1979), 88–103.
9. S. Eilenberg: *Automata, Languages and Machines* vol. A, Academic Press, 1974.
10. E. M. Gurari and O. H. Ibarra: Finite-valued and finitely ambiguous transducers, *Math. Systems Theory* **16** (1983), 61-66.
11. Ch. Reutenauer: Subsequential functions: characterizations, minimization, examples, *Lecture Notes in Comput. Sci.* **464** (1990), 62–79.
12. M. P. Schützenberger: Sur les relations rationnelles, *in Automata Theory and Formal Languages* (H. Brackhage, Ed.), *Lecture Notes in Comput. Sci.* **33** (1975), 209–213.
13. M. P. Schützenberger: Sur une variante des fonctions séquentielles, *Theoret. Comput. Sci.* **4** (1977), 47–57.