Squaring transducers

An efficient procedure for deciding functionality and sequentiality

Marie-Pierre Béal Olivier Carton[*] Christophe Prieur [†]
Jacques Sakarovitch [‡]

September 14, 2000

DEDICATED TO JEAN BERSTEL
ON THE OCCASION OF HIS 60TH BIRTHDAY

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.

Résumé

Ce papier présente une construction sur les transducteurs qui donne une nouvelle preuve conceptuelle pour deux résultats classiques de décidabilité sur les transducteurs: on peut décider si un transducteur fini réalise une relation fonctionnelle et s'il réalise une relation séquentielle. Il en résulte un algorithme polynomial pour les deux procédures de décision.

[*]Institut Gaspard Monge, Université de Marne-la-Vallée
[†]LIAFA, Université Paris 7 / CNRS
[‡]Laboratoire Traitement et Communication de l'Information (C. N. R. S. URA 820), E. N. S. T., Paris.

In this paper[1], 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 Schützenberger [13] *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 Choffrut [8] *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 [5], 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 ([11, Theorem 2]).

In this paper, 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.

Let us mention that the decidability of Theorem 2 in polynomial time has already been established by Weber and Klemm ([15]) by means of different methods. The complexity obtained in [15] is not explicitely given but seems to be similar to ours.

[1]Journal version of the paper presented at the LATIN 2000 conference under the same title [2]

1 Preliminaries

We basically follow the definitions and notation of [10, 3] 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^*}.

1.1 Automata, as usual.

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.

A *computation* c in $\mathcal{A}$ is a finite sequence of transitions that form a path in the graph and is noted as:

$$c \;:=\; p_0 \xrightarrow[\mathcal{A}]{a_1} p_1 \xrightarrow[\mathcal{A}]{a_2} p_2 \cdots \xrightarrow[\mathcal{A}]{a_n} p_n \qquad \text{or as} \qquad c \;:=\; p_0 \xrightarrow[\mathcal{A}]{a_1\, a_2 \ldots a_n} p_n \ .$$

The *label* of the computation c is the element $a_1\, a_2 \cdots a_n$ of A^*. The computation c is *successful* if $p_0 \in I$ and $p_n \in T$. The *behaviour* of $\mathcal{A}$ is the subset $|\mathcal{A}|$ of A^* consisting of labels of successful computations of $\mathcal{A}$. Kleene's theorem asserts that *a language of* A^* *is rational if and only if it is the behaviour of a finite automaton over* A.

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. The *behaviour* of $\mathcal{A}$ is the subset $|\mathcal{A}|$ of M consisting of the labels of the successful computations of $\mathcal{A}$. In this context, an automaton *over an alphabet* A is indeed an automaton *over the free monoid* A^*. The automaton $\mathcal{A}$ is finite if the set of edges $E \subset Q \times M \times Q$ is finite (and thus Q is finite). *A subset of* M *is rational if and only if it is the behaviour of a finite automaton over* M.

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 $\mathcal{A}$.

It is a slight generalization — that does not increase the generating power — to consider automata $\mathcal{A} = \langle Q, M, E, I, T \rangle$ where I and T are not *subsets* of Q (*i.e.* functions from Q into $\{0, 1\}$) but *functions* from Q into $M \cup \emptyset$ (the classical transducers are those for which the image of a state by I or T is either $\emptyset$ or 1_M). The label of a computation is then defined accordingly, being prefixed by the image of the starting state and sufixed by the image of the ending state of the computation.

1.2 Transducers, as usual?

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 a *transducer* from A^* to B^*. A transition of $\mathcal{T}$ is of the form $(p, (f, u), q)$; the word f is called its *input* and the word u its *output*. The terminology extends to computations of $\mathcal{T}$, denoted as

$$c \;:=\; p \xrightarrow[\mathcal{T}]{g/v} q$$

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.

The generalization quoted above leads us to consider transducers 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 $\mathcal{T}$ is said to be *real-time* if the label of every transition is a pair (a, K) where a is a letter in A and K a *rational* subset of B^* and where I and T are functions from Q into $\operatorname{Rat} B^*$. Using classical algorithms from automata theory, any transducer $\mathcal{T}$ can be transformed into an equivalent transducer that is real-time ([10, Th. IX.5.1], [3, Prop. III.7.1]). In this case, the freeness of B^* does not play any role and B^* may be replaced by any monoid M.

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 $\mathcal{T}$.

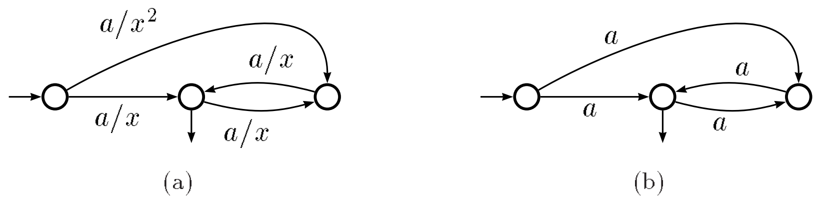


Figure 1: A real-time transducer ... and its underlying input automaton

If the relation α realized by $\mathcal{T}$ is *functional* and if $\mathcal{T}$ is real-time and *trim*, then necessarily the output of any transition is a *single word*, *i.e.* the label of any transition is a pair (a, u) where a is in A and u in B^*, and the image of any state by I or T is either $\emptyset$ or a word in B^*.

We call *subsequential* a transducer that is real-time, functional, and whose underlying input automaton is *deterministic*. A function α from A^* into B^* is *subsequential* if it can be realized by a subsequential transducer.

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 [4, 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 $\mathcal{D}$.*

Proof. By definition, $\mathcal{A}$ is ambiguous if and only if there exist two successful computations c' et c'' that have the same label $f = a_1 a_2 \ldots a_n$:

$$c' \ := \ q'_0 \xrightarrow[\mathcal{A}]{a_1} q'_1 \xrightarrow[\mathcal{A}]{a_2} \cdots \xrightarrow[\mathcal{A}]{a_n} q'_n \qquad \text{and} \qquad c'' \ := \ q''_0 \xrightarrow[\mathcal{A}]{a_1} q''_1 \xrightarrow[\mathcal{A}]{a_2} \cdots \xrightarrow[\mathcal{A}]{a_n} q''_n \ ,$$

that is, if and only if there exists a successful computation c of $\mathcal{A} \times \mathcal{A}$:

$$c \ := \ (q'_0, q''_0) \xrightarrow[\mathcal{A}\times\mathcal{A}]{a_1} (q'_1, q''_1) \xrightarrow[\mathcal{A}\times\mathcal{A}]{a_2} \cdots \xrightarrow[\mathcal{A}\times\mathcal{A}]{a_n} (q'_n, q''_n)$$

in which, for at least one i, $0 \leqslant i \leqslant n$, $q'_i \neq q''_i$ and, thus, if and only if there exists a useful state in $\mathcal{A} \times \mathcal{A}$ which is not in $\mathcal{D}$. ■

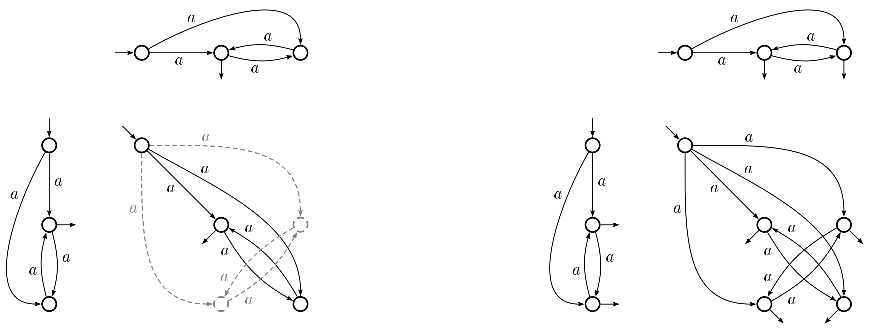
Figure 2 shows the underlying construction to Lemma 1 in the case of an ambiguous automaton and of an unambiguous automaton.

It is clear that Lemma 1 directly implies:

Proposition 2 *It is decidable whether a finite automaton is unambiguous or not.*

Remark that as (un)ambiguity, *determinism* can also be described in terms of Cartesian square, by a simple rewording of the definition:

Lemma 3 *A trim automaton $\mathcal{A}$ is* deterministic *if and only if the* accessible part *of the Cartesian square $\mathcal{A}\times\mathcal{A}$ is equal to $\mathcal{D}$.* ■



(a) The unambiguous case (b) The ambiguous case

Figure 2: The Lemma 1 construction. In dashed gray line, the non co-accessible states and transitions of the square of the automaton.

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

$$\delta \colon S\times M \longrightarrow S$$

which is consistent with the multiplication in M:

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

In order to lighten the notation, we write $s\cdot m$ rather than $(s,m)\delta$ when it causes no ambiguity and (1) becomes

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

Actions as automata. Most often, an action of M on S is equipped with a distinguished element s_0 of S. It may then be seen as an automaton on M (often, without terminal states). More precisely, let δ be an action of M on S with s_0 as distinguished element. The automaton

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

defined by the set of transitions

$$E = \{(s, m, s \cdot m) \mid s \in S\,,\ m \in M\}$$

is such that, for any m in M,

$$s_0 \xrightarrow{m} s = s_0 \cdot m$$

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 the action 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 $\mathcal{A} \times \mathcal{G}_\delta$. It may then 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 subset of $B^* \times B^*$ consisting of those elements (f, g) where at least one of f and g is equal to 1_{B^*}, to which a zero is adjoint:

$$H_B = (B^* \times 1_{B^*}) \cup (1_{B^*} \times B^*) \cup \{\mathbf{0}\}\ .$$

A mapping $\psi\colon B^* \times B^* \to H_B$ is defined by:

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

Intuitively, $(u,v)\psi$ 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, the following holds

$$(u,v)\psi = (1_{B^*}, 1_{B^*}) \quad \Longleftrightarrow \quad u = v \ . \tag{2}$$

And checking the following is easy.

Lemma 4 *The mapping ω_B from $H_B \times (B^* \times B^*)$ into H_B defined by:*

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

is an action of $(B^* \times B^*)$ on H_B. ■

This action ω_B will be called the "*Advance or Delay*" (or "*AD* ") action (relative to the alphabet 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

$$\begin{aligned} c' \ &:= \ q'_0 \xrightarrow[\mathcal{T}]{a_1/u'_1} q'_1 \xrightarrow[\mathcal{T}]{a_2/u'_2} \cdots \xrightarrow[\mathcal{T}]{a_n/u'_n} q'_n \\ c'' \ &:= \ q''_0 \xrightarrow[\mathcal{T}]{a_1/u''_1} q''_1 \xrightarrow[\mathcal{T}]{a_2/u''_2} \cdots \xrightarrow[\mathcal{T}]{a_n/u''_n} q''_n \end{aligned} \qquad \text{and}$$

with the same input label $a_1\,a_2 \ldots a_n$ and two distinct output labels:

$$u'_1\,u'_2 \ldots u'_n \neq u''_1\,u''_2 \ldots u''_n \ .$$

There exists then at least one index 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 shown at Figure 3 (*cf.* [3]).

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 transition set F is defined by:

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

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.

In order to decide whether $\mathcal{T}$ is functional when $\mathcal{A}$ is ambiguous, it is necessary to describe conditions under which two words such as $u'_1\,u'_2\ldots u'_n$ and $u''_1\,u''_2\ldots u''_n$ are equal or not, or, more precisely, which "information" has to be kept at every step i of the computation (c',c'') in order to be able to conclude at the final step n. This is what the AD action will be used for.

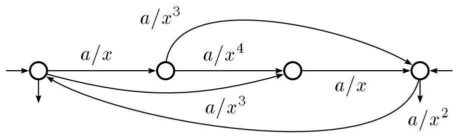


Figure 3: A functional transducer $\mathcal{Q}_1$ with ambiguous underlying input automaton.

4.1 A 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 build the product of $\mathcal{T}\times\mathcal{T}$, or of any of its subautomata, by the AD action ω_B.

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 4 shows the product of the Cartesian square of a transducer $\mathcal{Q}_1$ by the AD action[2] and one can read there that it is indeed functional.

Remark 3 If $\mathcal{T}$ is a *real-time* transducer from A^* into B^* and if α denotes 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 condition expressed in Theorem 3 may then seen as a condition for $\alpha\circ\alpha^{-1}$ to be the identity, which is clearly a condition for the functionality of α.

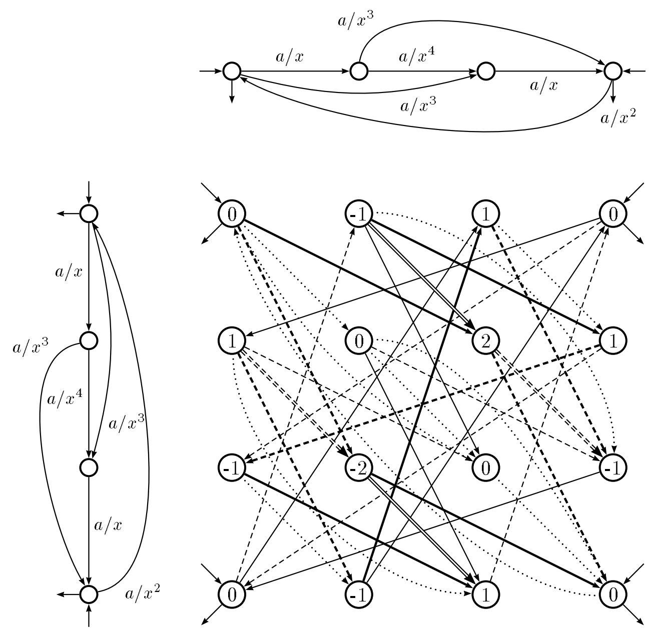


Figure 4: Cartesian square of $\mathcal{Q}_1$,
valued by the product with the action ω_B (with $B = \{x\}$).

This figure is not as complicated as it may look; it illustrates every aspect of the algorithm. Since the output alphabet B has only one letter, H_B 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 symbolized by the drawing of the arrow: a dotted arrow for 0, a simple solid arrow for $+1$, a bold one for $+2$ and a double one for $+3$; and the corresponding dashed arrows for the opposite values.

Proof of Theorem 3. i) The condition is sufficient. Let us denote by ν the valuation defined by the product of $\mathcal{U}$ by ω_B and let, as above, c' and c'' be two distinct successful computations of $\mathcal{T}$:

$$\begin{aligned} c' &:= q'_0 \xrightarrow[\mathcal{T}]{a_1/u'_1} q'_1 \xrightarrow[\mathcal{T}]{a_2/u'_2} \cdots \xrightarrow[\mathcal{T}]{a_n/u'_n} q'_n \\ c'' &:= q''_0 \xrightarrow[\mathcal{T}]{a_1/u''_1} q''_1 \xrightarrow[\mathcal{T}]{a_2/u''_2} \cdots \xrightarrow[\mathcal{T}]{a_n/u''_n} q''_n \end{aligned} \qquad \text{and}$$

It comes that $(q'_0, q''_0)\nu = (1_{B^*}, 1_{B^*})$ and $(q'_0, q''_0)\nu \cdot (u'_1 \cdots u'_i, u''_1 \cdots u''_i) = (q'_i, q''_i)\nu$ for every i and thus $(q'_0, q''_0)\nu \cdot (u'_1 \cdots u'_n, u''_1 \cdots u''_n) = (q'_n, q''_n)\nu = (1_{B^*}, 1_{B^*})$ as (q'_n, q''_n) is a final state of $\mathcal{T} \times \mathcal{T}$. Hence, by (2), $u'_1 \cdots u'_n = u''_1 \cdots u''_n$ and $\mathcal{T}$ is functional.

ii) The condition is necessary. Two cases possibly occur.

a) The product of $\mathcal{U}$ with ω_B yields a valuation but there exists a final sate (r', r') of $\mathcal{U}$ whose value is different from $(1_{B^*}, 1_{B^*})$. This means that there exists a successful

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

computation

$$(i', i'') \xrightarrow[\mathcal{T}\times\mathcal{T}]{f/(u',u'')} (r', r'')$$

and it holds: $(1_{B^*}, 1_{B^*}) \cdot (u', u'') \neq (1_{B^*}, 1_{B^*})$. Hence, by (2) again, $u' \neq u''$ and $\mathcal{T}$ is not functional.

b) The product of $\mathcal{U}$ with ω_B does not yield a valuation. There exist then two successful computations:

$$\begin{array}{l} (i', i'') \xrightarrow[\mathcal{T}\times\mathcal{T}]{f_1/(u'_1,u''_1)} (p', p'') \xrightarrow[\mathcal{T}\times\mathcal{T}]{f_2/(u'_2,u''_2)} (r', r'') \\ (j', j'') \xrightarrow[\mathcal{T}\times\mathcal{T}]{g_1/(v'_1,v''_1)} (p', p'') \xrightarrow[\mathcal{T}\times\mathcal{T}]{f_2/(u'_2,u''_2)} (r', r'') \end{array} \quad \text{and}$$

with $(1_{B^*}, 1_{B^*})\cdot(u'_1, u''_1) \neq (1_{B^*}, 1_{B^*})\cdot(v'_1, v''_1)$. The two equalities $u'_1\, u'_2 = u''_1\, u''_2$ and $v'_1\, u'_2 = u''_1\, v''_2$ cannot both hold and $\mathcal{T}$ is not functional. ■

4.2 Making the characterization effective.

We now show that Theorem 3 gives an *effective* characterization of functional transducers, hence is a proof of Theorem 1.

The algorithm for deciding whether $\mathcal{U}\times\omega_B$ is a valuation of $\mathcal{U}$ is elementary. The initial states are first given the value $(1_{B^*}, 1_{B^*})$. Every transition of $\mathcal{U}$ is then considered once, in any order that meet the condition that a transition is considered only if its origin has already been given a value. This is possible as $\mathcal{U}$ is trim. When considering a transition $((p', p''), (u, v), (q', q''))$, where (p', p'') has value (f, g), three cases may occur:

i) if (q', q'') has not been visited yet, then (q', q'') is given the value $(f, g) \cdot (u, v)$;

ii) if (q', q'') has already been visited and its value is not equal to $(f, g) \cdot (u, v)$, then the algorithm stops and $\mathcal{U}\times\omega_B$ is not a valuation of $\mathcal{U}$;

iii) if (q', q'') has already been visited and its value is equal to $(f, g) \cdot (u, v)$, then the algorithm goes on and the next transition is considered. If all transitions have been considered, then the algorithm stops and $\mathcal{U}\times\omega_B$ is a valuation of $\mathcal{U}$.

The transducer $\mathcal{T}$ is functional if and only if the value of every final state of $\mathcal{U}$ is $(1_{B^*}, 1_{B^*})$.

In order to evaluate the complexity of this algorithm, we have to define first the size of the data.

The "*size*" of an automaton $\mathcal{A}$ (on a free monoid A^*) is measured by the number n of states and 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 number n of states, the number m of transitions and the maximal size K of a transition, where the size of a transition $(p, (u, v), q)$ is the length $|uv|$. The sum of the sizes of the transitions is denoted by $\lceil \mathcal{T} \rceil$ and is bounded by $K\, m$.

The number of transitions of $\mathcal{T} \times \mathcal{T}$ is m^2 and the complexity to build it is proportional to m^2. The complexity of determining the trim part $\mathcal{U}$ is also in $O(m^2)$. The size $\lceil \mathcal{U} \rceil$ is bounded by $2\, K\, m^2$.

The computation of the value of one state in the product $\mathcal{U} \times \omega_B$ is at most of complexity $O(\lceil \mathcal{U} \rceil)$. Since for every transition of $\mathcal{U}$ one performs one computation of a value, the overall complexity $O(m^2 \lceil \mathcal{U} \rceil)$.

The same complexity is also established in [7] in the context of transducers for infinite words.

5 Deciding subsequentiality

The original proof of Theorem 2 goes indeed in three steps: first, subsequential 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.* [8, 3]).

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 subsequential functions

If f and g are two words, we denote by $f \wedge g$ the *longuest common prefix* of f and g: if $h = f \wedge g$, then $f = h$ and $g = f\, g'$, or $g = h$ and $f = g\, f'$, or $f = h\, a\, f''$ and $g = h\, b\, g''$ and a and b are two distinct letters. 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 $\mathsf{d_p}(f, g) = |f'| + |g'|$. This function $\mathsf{d_p}$ is indeed a "distance" but the topology it defines on A^* is a *discrete* topology, as the distance between two distinct words is greater than or equal to 1. Indeed, the prefix distance is not so much used to express *how close* two words are — which is usualy the purpose of a topology — but rather to describe *how far they are apart.*

Definition 1 *A function $\alpha \colon A^* \to B^*$, is said to be* uniformly divergent[3] *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.*

$$\forall n \in \mathbb{N}, \ \exists N \in \mathbb{N}, \ \forall f, g \in \mathsf{Dom}\, \alpha \qquad \mathsf{d_p}(f, g) \leqslant n \quad \Longrightarrow \quad \mathsf{d_p}(f\alpha, g\alpha) \leqslant N \ . \qquad (3)$$

[3]After [8] and [3], 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.

Equation (3) is to be put in parallel with the one that defines *uniform continuity* of functions, *i.e.* the ratio between the distance between the points and their images is bounded in the domain of the function. But the point is not that this ratio keeps bounded when the distance tends toward 0 — which is not possible with the prefix distance — but when this distance becomes arbitrarily large, hence the chosen denomination: "*uniform divergence*".

The following characterization is due[4] to Ch. Choffrut ([8, Proposition 3.4]).

Theorem 4
A rational function is subsequential if and only if it is uniformly divergent. ■

Remark 4 The characterization of subsequential 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 as well, a much stronger result, the full strength of which will not be of use here (*cf.* [6, 9]).

5.2 A characterization of subsequential functions on their transducers

Theorem 5 *A (real-time trim) transducer* $\mathcal{T} = \langle Q, A^* \times B^*, E, I, T \rangle$ *realizes a subsequential 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 *has the following two properties:*

i) *it is finite;*

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

Figure 5 shows two cases where the function is subsequential: 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 $(1_{B^*}, 1_{B^*})$.

Figure 6 shows two cases where the function is not subsequential: 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 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 subsequential 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.

[4]It is indeed in [14, Propriété 2] as well, but without the explicit definition of uniformly divergent functions.

One can also observe that Figure 4 is another example of the construction described in Theorem 5: the function realized by $\mathcal{Q}_1$ is sequential.

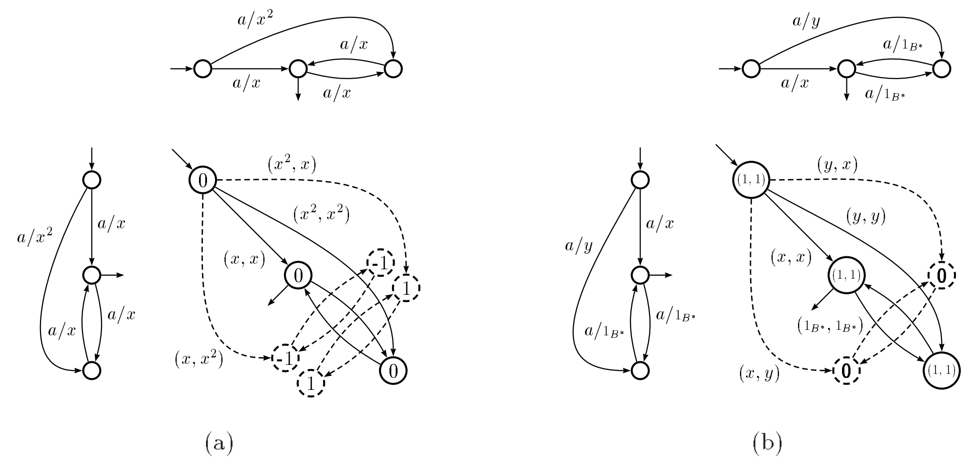


Figure 5: Two transducers that realize subsequential functions.

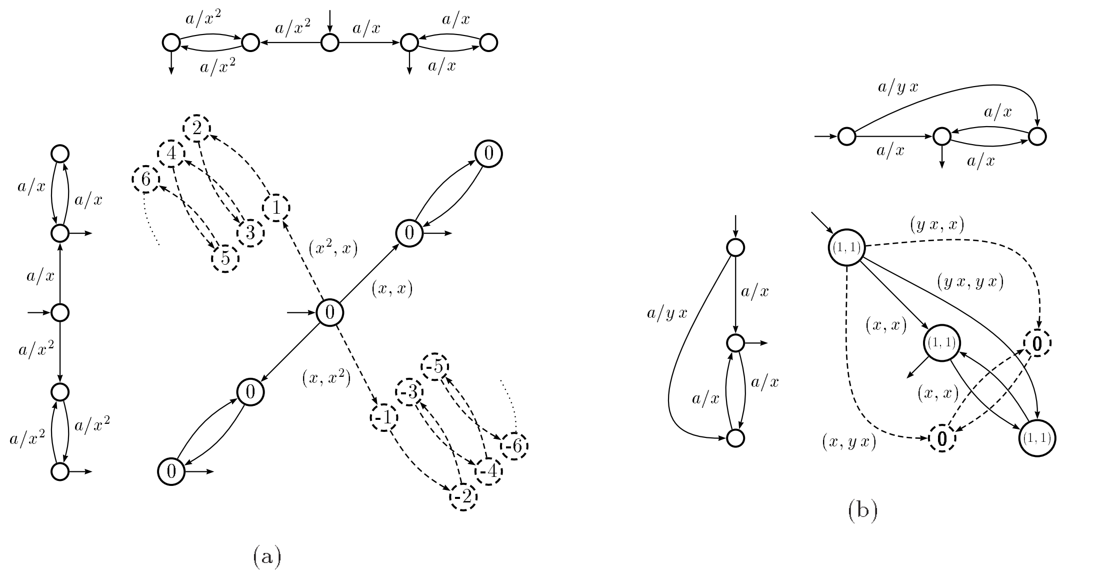


Figure 6: Two transducers that realize functions that are not subsequential.

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

Lemma 5 *Let* $(1_{B^*}, z)$ *be in* $H_B \setminus \mathbf{0}$ *and* (u, v) *in* $B^* \times B^* \setminus (1_{B^*}, 1_{B^*})$. *Then the set* $X = \{(1_{B^*}, z) \cdot (u, v)^n \mid n \in \mathbb{N}\}$ *is finite and does not contain* $\mathbf{0}$ *if and only if* u *and* v *are congugate by a word* t, *i.e.* $u\,t = t\,v$, *and* z *is equal to* $u^k t$ *for a certain* k. *If this condition holds then the set* X *is indeed a singleton.*

Proof. If the condition holds, it then comes:

$$(1_{B^*}, z) \cdot (u, v) = (1_{B^*}, u^{-1}(z\, v)) = (1_{B^*}, u^{-1}(u^k t\, v)) = \\ (1_{B^*}, u^{k-1} t\, v) = (1_{B^*}, u^{k-1} u\, t) = (1_{B^*}, z)$$

Conversely, if X does not contains $\mathbf{0}$ then necessarily one of the following conditions holds

i) either $u = 1_{B^*}$;

ii) either $v = 1_{B^*}$ and z is a prefix of a power of u;

iii) or z is a prefix of a power of u, *i.e.* $z = u^k t$ where t is a prefix of u, and there exist two integers h and l such that u^k is conjugated to v^l by t.

It is then clear that X is finite if and only if iii) holds, with $h = l$. ■

Remark 5 The original proof of Theorem 2 by Ch. Choffrut goes by the definition of the so-called *twinning property* (*cf.* [3, 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 happens thus that the conditions expressed in [8, Proposition 3.2] (or in [3, Proposition IV.6.4]) and in Theorem 5 are the same. It is the formulation that differs: the technicalities of the twinning property are hidden in Lemma 5.

Proof of Theorem 5. By Theorem 4, it is sufficient to show that the conditions stated in the theorem hold if and only if the function realized by $\mathcal{T}$ is uniformly divergent.

i) The conditions are sufficient. Let K be a bound for the lengths of the output of $\mathcal{T}$ and L a bound for the lengths of the values of states in the product $\mathcal{V} \times \omega_B$.

Let f and g in $\mathsf{Dom}\,\alpha$; we write $h = f \wedge g$, $f = h\, f'$ and $g = h\, g'$. There exist in $\mathcal{T}$ two successful computations

$$i \xrightarrow{h/u} p \xrightarrow{f'/u'} t \qquad \text{and} \qquad j \xrightarrow{h/v} q \xrightarrow{g'/v'} s \ , \quad \text{hence}$$

$$(i, j) \xrightarrow{h/(u,v)} (p, q) \tag{4}$$

is a computation in $\mathcal{V}$.

Case 1: $(1_{B^*}, 1_{B^*}) \cdot (u, v) \neq \mathbf{0}$, then

$$\begin{aligned} \mathsf{d_p}(f\alpha, g\alpha) &= \mathsf{d_p}(u\, u', v\, v') &\leqslant\quad & L + |u'| + |v'| \\ &\leqslant L + K\,(|f'| + |g'|) &=\quad & L + K\, \mathsf{d_p}(f, g) \ . \end{aligned}$$

Case 2: $(1_{B^*}, 1_{B^*}) \cdot (u,v) = \mathbf{0}$, then h is factorized as $h = h_1\, a\, h_2\, h_3$, with a in A, h_1, h_2 and h_3 in A^* (and possibly equal to 1_{A^*}), in such a way that the computation (4) factorizes into

$$(i,j) \xrightarrow{h_1/(u_1,v_1)} (p_1,q_1) \xrightarrow{a/(x,y)} (p_2,q_2) \xrightarrow{h_2/(1_{B^*},1_{B^*})} (p_3,q_3) \xrightarrow{h_3/(u_2,v_2)} (p,q)$$

where the value of (p_1,q_1) is different from $\mathbf{0}$, the one of (p_2,q_2) is equal to $\mathbf{0}$ and (u_2,v_2) is different from $(1_{B^*},1_{B^*})$ if h_3 is different from 1_{A^*}. As every state that follows (p_2,q_2) in the computation has value $\mathbf{0}$, the computation

$$(p_3,q_3) \xrightarrow{h_3/(u_2,v_2)} (p,q)$$

may not contain any cycle and its length is bounded by $|Q|^2 = n^2$. It then comes that

$$\begin{aligned} \mathsf{d_p}(f\alpha, g\alpha) &= \mathsf{d_p}(u_1\, x\, u_3\, u', v_1\, y\, v_3\, v') \\ &\leqslant L + K\,(n^2+1) + K\,(|f'|+|g'|) = L + K\,(n^2+1) + K\ \mathsf{d_p}(f,g) \ . \end{aligned}$$

In both cases, α is a uniformly divergent (rational) function.

ii) The conditions are necessary. Case 1: in $\mathcal{V}\times\omega_B$, there exists a cycle whose every state has value $\mathbf{0}$ and whose label is not equal to $(1_{B^*},1_{B^*})$. In $\mathcal{V}$, a computation

$$(i,j) \xrightarrow{h_1/(u_1,v_1)} (p,q) \xrightarrow{h_2/(u_2,v_2)} (p,q)$$

is found such that $(1_{B^*},1_{B^*}) \cdot (u_1,v_1) = \mathbf{0}$. This implies that the distance

$$\begin{aligned} \mathsf{d_p}((h_1\, h_2^r\, f')\alpha, (h_1\, h_2^r\, g')\alpha) &= \mathsf{d_p}(u_1\, u_2^r\, u', v_1\, v_2^r\, v') \\ &\geqslant r\,(|u_2|+|v_2|) + |u'| + |v'| \end{aligned}$$

can be made arbitrarily large with r.

Case 2: the product $\mathcal{V}\times\omega_B$ is infinite. There exists then in $\mathcal{V}$ at least one computation

$$(i,j) \xrightarrow{h_1/(u_1,v_1)} (p,q) \xrightarrow{h_2/(u_2,v_2)} (p,q)$$

which is lifted in $\mathcal{V}\times\omega_B$ as an infinite graph. Hence

$$(1_{B^*},1_{B^*}) \cdot (u_1,v_1) = (x,y) \neq \mathbf{0} \qquad \text{and} \qquad \forall r \in \mathbb{N} \qquad (x,y)\cdot(u_2,v_2)^r \neq \mathbf{0} \ .$$

From Lemma 5, it follows first that $|u_2| \neq |v_2|$ and then that there exists an n_0 such that

$$|(x,y)\cdot(u_2,v_2)^r| \geqslant (r-n_0)\ |(|u_2|-|v_2|)| \ \ ,$$

and thus

$$\begin{aligned} \mathsf{d_p}((h_1\, h_2^r\, f')\alpha, (h_1\, h_2^r\, g')\alpha) &= \mathsf{d_p}(u_1\, u_2^r\, u', v_1\, v_2^r\, v') \\ &\geqslant [(r-n_0)\ |(|u_2|-|v_2|)|] - \big|(|u'|-|v'|)\big| \end{aligned}$$

can be made arbitrarily large.

In both cases,

$$\mathsf{d_p}(h_1\, h_2^r\, f', h_1\, h_2^r\, g') \ \leqslant |f'| + |g'|$$

is fixed, and α is not uniformly divergent. ■

5.3 Making the characterization effective

We now show that the conditions of Theorem 5 may be effectively tested by means of an algorithm of polynomial time complexity.

Let $\mathcal{T} = \langle Q, A^* \times B^*, E, I, T \rangle$ be a transducer with n states, m transitions, and maximal size of transitions K. As in section 4.2, the accessible part $\mathcal{V}$ of $\mathcal{T} \times \mathcal{T}$ is computed in $O(m^2)$. And we have to build the product $\mathcal{W}$ of $\mathcal{V}$ by the AD action ω_B. As the "size" of a value of a state in $\mathcal{W}$ is linear in K, a too rough estimation for the size of $\mathcal{W}$ would be exponential in the size of $\mathcal{T}$. The overall idea of the algorithm is that on the states of $\mathcal{V}$ that really "matter" for the decision procedure, the non-zero values (that are elements of H_B and thus almost words of B^*) have to be prefix of eachother. There are a linear number of them and we show that they can be computed in polynomial time.

Let $\mathcal{V}'$ be the subautomaton of $\mathcal{V}$ consisting of those states that are *co-accessible to a cycle whose output label is distinct from* $(1_{B^*}, 1_{B^*})$. The computation of $\mathcal{V}'$ is done in a time bounded by the number of transitions of $\mathcal{V}$, at most m^2. And let $\mathcal{W}'$ be the product of $\mathcal{V}'$ by ω_B. It is clear that the conditions of Theorem 5 are fulfilled on $\mathcal{W}$ if and only if they are fulfilled on $\mathcal{W}'$ and from now on we will only consider the transducer $\mathcal{V}'$ and its product $\mathcal{W}'$. We shall say that two words w and w' of B^* are *comparable* if one is the prefix of the other.

Lemma 6 *If $((p, q), (1_{B^*}, w))$ and $((p, q), (1_{B^*}, w'))$ are both states of $\mathcal{W}'$, then w and w' are comparable or condition (ii) of Theorem 5 is not fulfilled.*

Proof. Since (p, q) is in $\mathcal{V}'$, it is co-accessible to a state (r, s) that belongs a cycle labelled by $(u, v) \neq (1_{B^*}, 1_{B^*})$, *i.e.* there exists a path $(p, q) \xrightarrow{f_3/(x,y)} (r, s)$ in $\mathcal{V}'$. If w and w' are not comparable then at least one of the sets $X = \{(1_{B^*}, w) \cdot (x, y) \cdot (u, v)^n \mid n \in \mathbb{N}\}$ or $X' = \{(1_{B^*}, w') \cdot (x, y) \cdot (u, v)^n \mid n \in \mathbb{N}\}$ contains $\mathbf{0}$ and the state $((r, s), \mathbf{0})$ is in $\mathcal{W}'$. ∎

Lemma 7 *If $((p, q), (1_{B^*}, w))$ is a state of $\mathcal{W}'$ then $|w| \leqslant K\, n^2$ or the conditions of Theorem 5 are not fulfilled.*

Proof. Let us show that the shortest path c in $\mathcal{W}'$ from an initial state $((i, j), (1_{B^*}, 1_{B^*}))$ to $((p, q), (1_{B^*}, w))$ has a length smaller than n^2. If not, c has a decomposition:

$$((i, j), (1_{B^*}, 1_{B^*})) \xrightarrow{f_1/(u_1,v_1)} ((r, s), h_1) \xrightarrow{f_2/(u_2,v_2)} ((r, s), h_2) \xrightarrow{f_3/(u_3,v_3)} ((p, q), (1_{B^*}, w))$$

By Lemma 5, the set $X = \{h_1 \cdot (u_2, v_2)^n \mid n \in \mathbb{N}\}$ has to be a singleton and thus $h_1 = h_2$, which yields a shorter path.

Hence the length of w is bounded by $K\, n^2$. ∎

In order to build $\mathcal{W}'$, we maintain two arrays of words T_1 and T_2 indexed by $Q \times Q$ that are initialized to 1_{B^*}. The states of $\mathcal{W}'$ are computed one after the other. For every state $((p,q),h)$ of $\mathcal{W}'$ and every transition $(p,q) \xrightarrow{(a/(u,v))} (p',q')$ of $\mathcal{V}'$, we compute $h' = h \cdot (u,v)$.

a) If h' is $\mathbf{0}$, condition (ii) of Theorem 5 is not satisfied and the algorithm stops.

b) If h' is an element $(w, 1_{B^*})$ (resp. $(1_{B^*}, w)$), then one checks whether w and $T_1[p',q']$ (resp. $T_2[p',q']$) are comparable. There are two possibilities:

  b.1 If they are not comparable, then, by Lemma 6, condition (ii) of Theorem 5 is not satisfied.

  b.2 If they are comparable, one updates $T_1[p',q']$ (resp. $T_2[p',q']$) with the longer of the two words.

Now, by Lemma 7, the words computed in the arrays T_1 and T_2 have a length bounded by $K\,n^2$. Therefore the algorithm always stops: either because the condition (ii) of Theorem 5 is not satified or, if this condition is satisfied, because no new states of $\mathcal{W}'$ are computed. In the latter case, $\mathcal{W}'$ is finite and the condition (i) is satisfied.

The number of states of $\mathcal{W}'$ that we have constructed is at most $2K\,n^4$. The number of transitions of $\mathcal{W}'$ is at most $m^2 \times (2K\,n^2) = 2K\,n^2 m^2$. Indeed, for each transition leaving a state (p,q) in $\mathcal{V}'$, one constructs at most one transition leaving each state $((p,q),h)$ in $\mathcal{W}'$. The time complexity of the construction is at most $2K\,n^2 m^2 \times K\,n^2 = 2K^2 n^4 m^2$ since the time taken to check whether two words are comparable is at most $K\,n^2$.

In [1], two of the authors show directly that the twinning property is decidable in polynomial time. Let us mention also that in [15], it has also been shown, by a different algorithm, 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] Marie-Pierre Béal, Olivier Carton, Christophe Prieur and Jacques Sakarovitch, Squaring transducers, *in Proc. LATIN 2000*, (G. Gonnet, D. Panario and A. Viola, eds.), L. N.C.S. 1776 (2000), 397–406.

[3] J. Berstel: *Transductions and context-free languages*, Teubner, 1979.

[4] J. Berstel and D. Perrin: *Theory of codes*, Academic Press, 1985.

[5] M. Blattner and T. Head: Single valued a-transducers, *J. Computer System Sci.* **7** (1977), 310–327.

[6] V. Bruyère and Ch. Reutenauer: A proof of Choffrut's theorem on subsequential functions, *Theoret. Comput. Sci.* **215** (1999), 329–335.

[7] O. Carton, Ch. Choffrut and Ch. Prieur: How to decide functionality and continuity of rational relations on infinite words, *to appear.*

[8] Ch. Choffrut: Une caractérisation des fonctions séquentielles et des fonctions sous-séquentielles en tant que relations rationnelles, *Theoret. Comput. Sci.* **5** (1977), 325–337.

[9] 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.

[10] S. Eilenberg: *Automata, Languages and Machines* vol. A, Academic Press, 1974.

[11] E. M. Gurari and O. H. Ibarra: Finite-valued and finitely ambiguous transducers, *Math. Systems Theory* **16** (1983), 61-66.

[12] Ch. Reutenauer: Subsequential functions: characterizations, minimization, examples, *Lecture Notes in Comput. Sci.* **464** (1990), 62–79.

[13] 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.

[14] M. P. Schützenberger: Sur une variante des fonctions séquentielles, *Theoret. Comput. Sci.* **4** (1977), 47–57.

[15] A. Weber and R. Klemm: Economy of Description for Single-Valued Transducers, *Information and Computation* **118** (1995), 327–340.