Theoretical Informatics and Applications Will be set by the publisher
Informatique Théorique et Applications

RADIX ENUMERATION OF RATIONAL LANGUAGES

Pierre-Yves Angrand[1] and Jacques Sakarovitch[2]

Abstract. We prove that the function that maps a word of a rational language onto its successor for the radix order in this language is a finite union of co-sequential functions.

1991 Mathematics Subject Classification. 68Q45,68Q70.

Introduction

The purpose of this paper is to prove the following property of rational languages (and of the radix order):

Theorem 1. *The radix enumeration of a rational language is a finite union of co-sequential functions.*

The radix order (whose formal definition will be recalled below, along with the one of every notion involved in this statement) is a well-order on A^* and the elements of any subset L of A^* may thus be ordered:

$$L = \{f_0 < f_1 < f_2 < \ldots\}$$

the word f_0 being the smallest of L, f_1 the smallest of $L \setminus \{f_0\}$, f_2 the smallest of $L \setminus \{f_0, f_1\}$, *etc.* The (radix) enumeration of L is the function Succ_L of A^* into itself whose domain is L and which maps each f_i to f_{i+1}.

The motivation for establishing Theorem 1 comes from the study of numeration systems, more precisely from the study of the *concrete complexity* of the successor function in *non standard* numeration systems. If L is the set of representations of the integers in a numeration system, then the radix enumeration is the successor

Keywords and phrases: finite automata, rational functions of words, sequential transducers

[1] LTCI (UMR 5141), Telecom ParisTech, 46 rue Barrault, 75634 Paris Cedex 13, France
e-mail: angrand@enst.fr
[2] LTCI (UMR 5141), CNRS / Telecom ParisTech, 46 rue Barrault, 75634 Paris Cedex 13, France
e-mail: sakarovitch@enst.fr

function lifted at the level of representations (hence the notation). Conversely, any language L ordered by the radix order can *ipso facto* be considered as a set of representations of the integers: f_i being the representation of the integer i; L is then called an *abstract number system*, that is, a set of representations of the integers without any reference to a method for computing them (*cf.* [8]). By concrete complexity, we mean the number of operations necessary to compute the function under scrutiny, here the successor function. The evaluation of this quantity supposes that a precise computational model has been chosen. In [3, 4] it is shown that a reasonable such model is, roughly speaking, that of a *cascade of sequential (right) transducers*, that is, a first transducer reads the input and produces an output which is then taken as the input of second transducer *which depends on the final state in the computation of the first one*, and so on.

It turns out — as we show in [1] — that the functions realized by cascades of sequential (right) transducers are exactly those that are a finite union of (co-)sequential functions — which we propose to call *piecewise (co-)sequential functions*. These functions were already considered (long ago!) by Choffrut and Schützenberger, who called them "plurisubsequential" (*cf.* [6]).[1] This family of functions, in particular its decidability within the family of rational functions — already established in [6] — is further investigated in [1].

The proof of Theorem 1, brings into play a number of properties of, and constructions on, finite automata. We first reduce the problem of the *successor function* to the one of *uniform-length successor function* ULSucc_L, that is, the restriction of the successor function $g = \mathsf{Succ}_L(f)$ to the case where $|f| = |g|$ (and is not defined otherwise). Then, the function ULSucc_L applied to a word f is so to speak split into two parts: a first part κ, which acts on a prefix h of f and which is the identity, and a second part τ, which acts on the corresponding suffix k, $f = h\,k$, and which relates k to another word k'. These k and k' — both of same length — are taken from rational sets which contain at most one word per length. From the special form of the automata which recognize that sort of rational languages, one builds a transducer which maps k to k' and which is a sequential right transducer provided an adequate shift is performed on the output. The last part of the proof consists then in establishing that it is possible to put together, that is, to concatenate, the first part κ with the second parts τ_i and to perform adequate transformations in such way that we get a finite union of sequential right transducers.

1. Preliminaries

We basically follow the definitions and notations of [7] (in particular for the post-fixed notation of functions and relations), and of [2] (and also of [12]). In the sequel, A is a finite alphabet, A^* the free monoid generated by A, 1_{A^*} the empty word, identity of A^*. For a word u in A^*, $|u|$ is the length of u.

[1] Since we changed anyway the terminology from "subsequential" to "sequential" (see [12] for the explanation on this change of the terminology), we shall neither use "plurisubsequential".

Let A be an alphabet *totally ordered* by $<$. The *radix order* $\prec$ on A^* is defined by: $u \prec v$ if either $|u| < |v|$ or $|u| = |v|$ and there exist w, u', v' words and $a < b$ letters such that $u = w\,a\,u'$ and $v = w\,b\,v'$. The radix order is a *well-order*, that is, every non empty subset of A^* has a smallest element for $\prec$.

For every language L of A^*, the *successor function* of L, denoted by Succ_L, is the function that maps every word u in L onto the word v which is the smallest of all words of L greater than u. Radix enumeration is another name, easily understandable, for the successor function. The successor function of a rational language can be easily understood on the tree representing the language: words are ordered from top to bottom and left to right.

Example 1. In our running examples, we restrain ourself to the alphabet A with two letters $A = \{a, b\}$. Let us consider the language L_1 of words with an even number of occurrences of b: $L_1 = \{u \in A^* \mid |u|_b \equiv 0\ [2]\}$. Figure 1 shows the tree representing L_1 and it can be seen that: $\mathsf{Succ}_{L_1}(aa) = bb$, $\mathsf{Succ}_{L_1}(abb) = bab$ and $\mathsf{Succ}_{L_1}(bb) = aaa$.

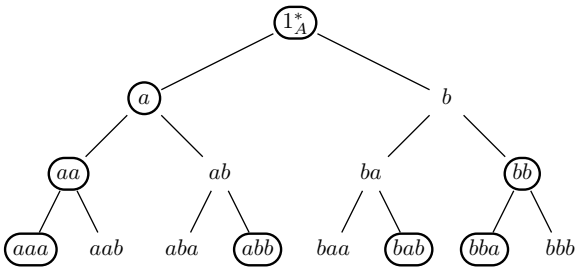


FIGURE 1. The tree representation of L_1

For the general definitions of automata, rational relations and finite transducers, the reader is referred to the references quoted above. In this paper, we shall content ourselves with a restricted type of transducers, realtime transducers. First, and in order to simplify definitions and notation, all word functions or relations map words of A^* into A^*. Since we are never interested in the functions or relations being total or surjective, this is indeed not a restrictive hypothesis.

Definition 1. Let us call *transducer* (from A^* into itself) an automaton $\tau = \langle\, Q, A, A^*, E, I, T \,\rangle$ where:

Q is a finite set of states;

I is a partial function, the *initial function*, from Q into A^*; we rather write I_q for the value $I(q)$, and q is said to be *initial* if I_q is defined;

T is a partial function, the *final function*, from Q into A^*; we rather write T_q for the value $T(q)$, and q is said to be *final* if T_q is defined;

E is a finite set of transitions whose labels are in $A \times A^*$.

A transducer is *letter-to-letter* if the label of transitions are all in $A \times A$.

The way a relation φ from A^* into itself is associated with a transducer τ — we say that τ realizes φ — is classical. A functional relation is rational (*i.e.* its graph is a rational subset of $A^* \times A^*$) if, and only if, it is realized by a transducer (our definition implies that transducers are finite).

The *underlying input automaton* of a transducer $\tau = \langle\, Q, A, A^*, E, I, T \,\rangle$ is the finite automaton obtained from τ by keeping the first component of the label of every transition and by replacing the initial and final functions by their domains.

Definition 2. A transducer is said to be a sequential (resp. *co-sequential*) transducer if its underlying input automaton is deterministic (resp. *co-deterministic*).

A rational function is sequential (resp. co-sequential) if it is realized by a sequential (resp. co-sequential) transducer.

Remark 1. A co-sequential transducer is also often seen as a transducer with deterministic underlying input automaton but which reads and writes words from the right to the left, which is called a *sequential right transducer*.

Sequential functions were introduced by Schützenberger in [13] under the name of *subsequential functions* (cf. also [2]). It will be no surprise that we rather follow here the terminology of [12]. Moreover, and as in this last reference, we reserve the qualifiers 'right' and 'left' *for automata* (and hence for transducers) which model physical machines, and according to as they read words *from right to left* or *from left to right* respectively. Functions are neither left nor right but they are realized by transducers which can be right or left.

Example 2. The sequential right transducer of Figure 2 (a) and the left co-sequential transducer of Figure 2 (b), which is the transpose of the first one, realize the same co-sequential function.

(a) a sequential right transducer (b) a co-sequential left transducer

FIGURE 2. Left and right transducers realizing the same co-sequential function

Sequential functions are characterized within rational functions by a topological criterion in the following way: the *prefix distance* d of two words u and v is defined by $\mathsf{d}(u, v) = |u| + |v| - 2\,|u \wedge v|$, where $u \wedge v$ is the longest common prefix of u and v.

Definition 3. A function φ is said to be k-Lipschitz (for the prefix distance) if:

$$\forall u, v \in \mathsf{Dom}(\varphi) \;\;,\quad \mathsf{d}(u\varphi, v\varphi) \leq k\;\mathsf{d}(u, v) \;\;.$$

The function φ is Lipschitz if there exists a k such that φ is k-Lipschitz.

Theorem 2 ([5]). *A rational function is sequential if, and only if, it is Lipschitz.*

Remark 2. By the left–right duality we define the suffix distance d' on A^*. A rational function is co-sequential if, and only if, it is Lipschitz for the suffix distance.

Here, we just use this characterization in order to show that in some cases the successor function of a rational language is not sequential or not co-sequential.

Example 3 (*Ex. 2 cont.*). Let us consider the language consisting of the whole A^*. The successor function Succ_{A^*} is realized by the transducers of Figure 2, and is thus co-sequential. If we consider the words $u = (b)^k a$ and $v = (b)^k$ it holds:

$$\mathsf{d}(u, v) = 1 \qquad \mathsf{d}(\mathsf{Succ}_{A^*}(u), \mathsf{Succ}_{A^*}(v)) = \mathsf{d}((b)^k b, (a)^{k+1}) = 2k + 2 \ .$$

The function Succ_{A^*} is not Lipschitz and hence not sequential.

Example 4 (*Ex. 1 cont.*). Let $u = (b)^{2k}$ and $v = bba(b)^{2k}$ fo $k \geq 1$. It holds:

$$\mathsf{d}'(u, v) = 3 \qquad \mathsf{d}'(\mathsf{Succ}_{L_1}(u), \mathsf{Succ}_{L_1}(v)) = \mathsf{d}'((a)^{2k+1}, bbb(a)^{2k-1}b) = 4k + 4 \ .$$

The function Succ_{L_1} is not co-sequential.

Since the radix order, as well as the identity on every rational set, are *synchronous rational relations*, standard properties of these synchronous rational relations allow to prove the following result (cf. [3], [12]):

Proposition 1. The successor function of a rational language is realized by a letter-to-letter finite transducer.

Definition 4. A function which is a finite union of sequential (resp. co-sequential) functions is called a *piecewise sequential* (resp. *piecewise co-sequential*) function.

Theorem 1 can now be restated as:

Theorem 1. *The successor function of a rational language is piecewise co-sequential.*

2. The synchronous product of ray automata

Let L and K be two rational languages which have at most one word per length. In this section, we prove that the function which maps every word of L onto the word of K with same length is piecewise sequential and piecewise co-sequential.

We adapt the terminology proposed by Christophe Reutenauer in [10] and call *ray language* a language L of the form $L = uv^*w$. It is folklore (and has been proved many times) that rational languages whose growth function is bounded – and in particular rational languages which have at most one word per length – are finite union of pairwise disjoint ray languages.

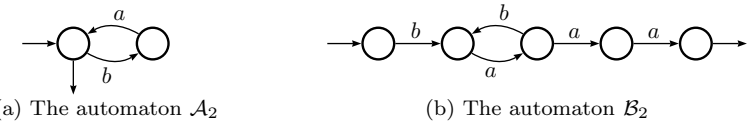
2.1. Ray automata

Definition 5. We call *ray automaton* an automaton which is trim, and has a unique initial state, a unique final state and at most one circuit. Transducers are automata and we call *ray transducer* a transducer with the same properties.

We call *loop-initial* a ray automaton in which the initial state belongs to the circuit and *loop-final* if the final state belongs to the circuit; the same terminology carries over to ray transducers.

The structure of ray automata clearly shows that they recognize ray languages and that any ray language can be recognized by a ray automaton.

Example 5. Figure 3 shows two examples of ray automata, $\mathcal{A}_2$ and $\mathcal{B}_2$. where $\mathcal{A}_2$ recognizes $L_2 = \{(a\,b)^k \mid k \in \mathbb{N}\}$ and $\mathcal{B}_2$ recognizes $K_2 = \{b\,a\,(b\,a)^k a\,a \mid k \in \mathbb{N}\}$. $\mathcal{A}_2$ is both loop-initial and loop-final and $\mathcal{B}_2$ is neither loop-initial nor loop-final.



(a) The automaton $\mathcal{A}_2$ (b) The automaton $\mathcal{B}_2$

FIGURE 3. Two ray automata

2.2. Synchronous product

Definition 6. Let $\mathcal{A} = \langle Q, A, E, I, T\rangle$ and $\mathcal{B} = \langle R, A, F, J, U\rangle$ be two finite automata. The *synchronous product* of $\mathcal{A}$ and $\mathcal{B}$ is the trim component of the transducer:

$$\mathcal{A} \bowtie \mathcal{B} = \langle Q\times R, A, A, G, I\times J, T\times U\rangle$$

where the set of transitions G is defined by:

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

Let $\mathcal{A}$ and $\mathcal{B}$ be two finite automata that recognize respectively the rational languages L and K. The synchronous product[2] $\mathcal{A} \bowtie \mathcal{B}$ realizes the relation $\theta\colon A^* \to A^*$ such that $\theta = \{(u,v) \mid u \in L, v \in K, |u| = |v|\}$. The synchronous product distributes over union, that is:

$$\left[\bigcup_{i\in I} \mathcal{A}_i\right] \bowtie \left[\bigcup_{j\in J} \mathcal{B}_j\right] = \bigcup_{(i,j)\in I\times J} (\mathcal{A}_i \bowtie \mathcal{B}_j) \ .$$

From the definition of synchronous product directly follows:

[2]This synchronous product slightly differs by the final functions from the one considered in [12, Exerc. IV.6.17], which realizes the relation whose graph is $L\times K$.

Proposition 2. The synchronous product of two ray automata is a ray transducer.

If K has at most one word for every length, then the transducer $\mathcal{A} \bowtie \mathcal{B}$ realizes a function. Even if $\mathcal{A}$ is deterministic, it is not likely to be sequential as soon as there is a state in $\mathcal{B}$ which is the source of more than one transitions.

Proposition 3. The synchronous product $\mathcal{A} \bowtie \mathcal{B}$ of two ray automata is sequential (resp. co-sequential) if and only if $\mathcal{A}$ is deterministic (resp. co-deterministic) and $\mathcal{B}$ is loop-final (resp. loop-initial).

Example 6 (*Ex. 5 cont.*)**.** Figure 4 shows the synchronous product $\mathcal{A}_2 \bowtie \mathcal{B}_2$ which is not sequential (nor co-sequential).

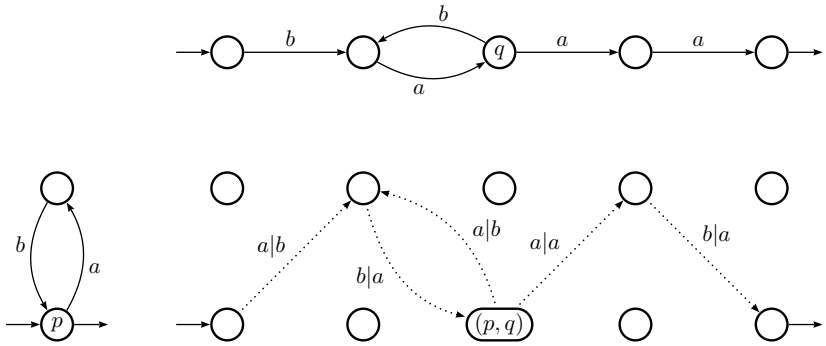


FIGURE 4. The synchronous product of $\mathcal{A}_2$ and $\mathcal{B}_2$

2.3. Making the synchronous product sequential

For the purpose of the construction underlying our proof, we slightly enlarge the family of automata we are considering:

- the transitions of automata are labeled either by a letter a of A or by the empty word 1_{A^*} (transitions labeled by 1_{A^*} are called *spontaneous transitions*),
- the subsets I and T of Q for initial and final states are replaced by (partial) functions from Q into A^*, still denoted by I and T, and their value $I(p)$ and $T(q)$ are rather denoted I_p and T_q.

The *label* $|c|$ of a computation $c : i \xrightarrow[\mathcal{A}]{f} t$ is then $|c| = I_i.f.T_t$. We denote by $\ell(c)$ the number of transitions of the computation c and we call it the *length* of c.

When it is needed, we call *classical automaton* an automaton without spontaneous transitions and with final and initial states. It is well known that the family of languages recognized by (our new class of) automata is not larger than the family of rational languages and that any classical automaton can be seen as

an automaton by setting $T_t = 1_{A^*}$ when t is final and $T_t = \emptyset$ otherwise, $I_i = 1_{A^*}$ if i is initial and $I_i = \emptyset$ otherwise.

The definition of this larger class of automata is taken in view of the following construction.

Proposition 4. Every ray automaton $\mathcal{B}$ can be transformed into a ray automaton $\breve{\mathcal{B}}$ with the following properties:

(i) $\breve{\mathcal{B}}$ is equivalent to $\mathcal{B}$.

(ii) Every state of $\breve{\mathcal{B}}$ is the source of at most one transition, that is, $\breve{\mathcal{B}}$ is loop-final.

(iii) If the computations c in $\mathcal{B}$ and c' in $\breve{\mathcal{B}}$ have same label then $\ell(c) = \ell(c')$.

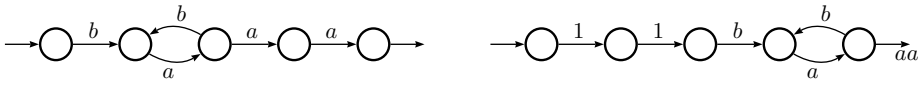
Proof. The transducer $\breve{\mathcal{B}}$ is obtained from $\mathcal{B}$ by the following operations:

(a) replacing the unique path going out of the circuit in $\mathcal{B}$ by a final function on the first state of this path – which thus belongs to the circuit – and which value is the label of this path;

(b) adding at the initial state of $\mathcal{B}$ a path whose label is the empty word and whose length is the length of the path removed in (a).

From (a) $\breve{\mathcal{B}}$ is loop-final and equivalent to $\mathcal{B}$; (iii) follows from (b). □

Such a construction can be seen on Figure 5 for $\mathcal{B}_2$.



(a) The automaton $\mathcal{B}_2$ (b) The automaton $\breve{\mathcal{B}}_2$

FIGURE 5. Transformation into a loop-final ray automaton

The definition of synchronous product goes over this larger class of automata and it now holds:

$$\mathcal{A} \bowtie \mathcal{B} = \langle Q \times R, A, A, G, I \times J, T \times U \rangle$$

where: $[I \times J]_{(p,q)} = (I_p, J_q)$, $[T \times U]_{(p,q)} = (T_p, U_q)$ and:

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

Example 7 (*Ex. 5 cont.*)**.** Figure 6 shows the synchronous product of $\mathcal{A}_2$ and $\breve{\mathcal{B}}_2$.

From the definition of $\breve{\mathcal{B}}$, the two properties follow:

Property 1. Let $\mathcal{A}$ and $\mathcal{B}$ be two ray automata. Then $\mathcal{A} \bowtie \mathcal{B}$ and $\mathcal{A} \bowtie \breve{\mathcal{B}}$ are equivalent.

Property 2. Let $\mathcal{A}$ and $\mathcal{B}$ be two ray automata. If $\mathcal{A}$ is deterministic, then $\mathcal{A} \bowtie \breve{\mathcal{B}}$ is a sequential ray transducer.

We are now able to establish the main result of this section:

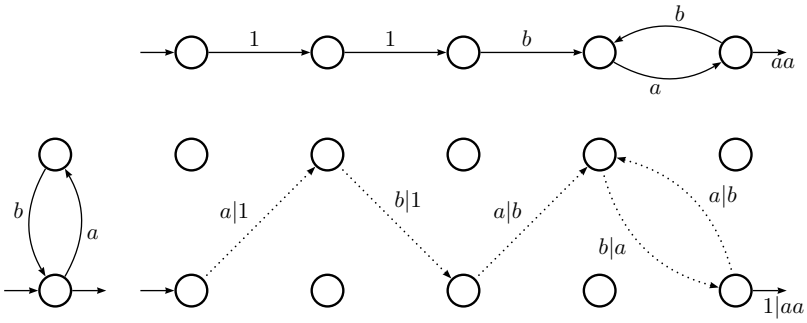


FIGURE 6. The synchronous product of $\mathcal{A}_2$ and $\breve{\mathcal{B}}_2$

Proposition 5. Let L and K be two rational languages with at most one word for each length. The function φ that maps every word of L onto the word of K with the same length, if it exists, is realized by a finite union of sequential ray transducers.

Proof. As we already recalled, a rational language with at most one word for each length is a finite union of pairwise disjoint ray languages and then can be realized by a finite union of deterministic ray automata. Let $\mathcal{A}_1, \ldots, \mathcal{A}_l$ be the deterministic ray automata whose union recognizes L and $\mathcal{B}_1, \ldots, \mathcal{B}_k$ the deterministic ray automata whose union recognizes K. Then K is also recognized by the union of $\breve{\mathcal{B}}_1, \ldots, \breve{\mathcal{B}}_k$ (Proposition 4).

For every i and j, $\mathcal{A}_i \bowtie \breve{\mathcal{B}}_j$ is a sequential transducer and realizes the same function as $\mathcal{A}_i \bowtie \mathcal{B}_j$. The function φ is realized by the union of the $\mathcal{A}_i \bowtie \mathcal{B}_j$, thus by the union of the $\mathcal{A}_i \bowtie \breve{\mathcal{B}}_j$. □

The previous propositions and their proofs are symmetrical when $\mathcal{B}$ is transformed into a ray automaton which is equivalent, loop-initial and for which the computations with same label have same length. The following result holds:

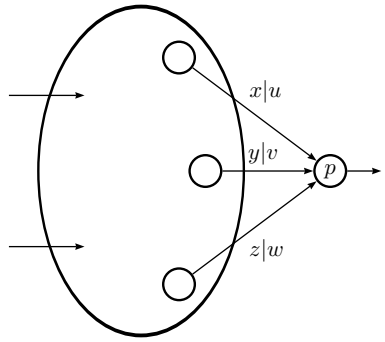
Proposition 6. Let L and K be two rational languages with at most one word for each length. The function φ that maps every word of L onto the word of K with the same length is realized by a finite union of co-sequential ray transducers. □

3. Concatenation of transducers

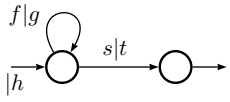
An automaton is said to be *standard* (resp. *co-standard*) if it has a single initial (resp. final) state without any incoming (resp. outgoing) transition. Transducers are automata and the same definitions apply.

In the sequel, we denote relations and their graph by the same letter. The concatenation of two relations φ and φ' is denoted[3] by $\varphi||\varphi'$ and is the relation whose graph is the product (in $A^* \times A^*$) of the graphs of φ and φ': if $(u, v) \in \varphi$ and $(u', v') \in \varphi'$ then $(u\,u', v\,v') \in \varphi||\varphi'$. If φ and φ' are rational functions, $\varphi||\varphi'$ is rational but not necessarily a function.

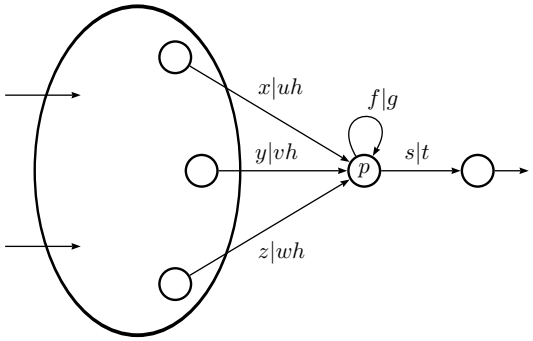
The concatenation of two automata – or two transducers – is a classical operation. In our proof, we consider a co-sequential and co-standard transducer κ and a co-sequential ray transducer τ. In this case, the concatenation $\kappa||\tau$ can be realized by merging the final state of κ with the initial state of τ and concatenating the final function of κ and the initial function of τ to the output of the transitions incoming to the final state of κ.



(a) The transducer κ



(b) The transducer τ



(c) The concatenation $\kappa||\tau$

FIGURE 7. Two transducers and their concatenation

Figure 7 illustrates this construction. The following two properties directly result from this construction.

[3]The concatenation of subsets of a monoid is usually denoted by a simple juxtaposition, but in order to avoid confusion with the composition of functions, which is also denoted in the same way, we prefer to have a different, and explicit, notation.

Property 3. Let κ be a co-deterministic co-standard transducer and τ a co-sequential ray transducer. If one of the following two conditions holds:

(a) τ is not loop-initial,

(b) τ is loop-initial but the input of the incoming transition to the initial state of τ is not the input of any transition incoming to the final state of κ,

then, the concatenation $\kappa||\tau$ is a co-sequential transducer.

4. The uniform-length successor function

Let L be a language of A^*. Let us denote the sets of *minimal words* and of *maximal words* of L (for each length) by $\mathsf{minL}(L)$ and $\mathsf{MaxL}(L)$ respectively:

$$\mathsf{minL}(L) = \{u \in L \mid \forall v \in L, |u| = |v| \Rightarrow u \preccurlyeq v\} \ ,$$
$$\mathsf{MaxL}(L) = \{u \in L \mid \forall v \in L, |u| = |v| \Rightarrow v \preccurlyeq u\} \ .$$

If L is rational, so are $\mathsf{minL}(L)$ and $\mathsf{MaxL}(L)$ (cf. [11, 14] for instance).

The main building block of our construction is the synchronous product, which yields transducers that realize length-preserving functions. As Succ_L is not a length-preserving function, we slightly change both the function and the language that we study: if L is a language of A^*, let ULSucc_L be the *uniform-length successor function*, that is, the restriction of Succ_L to $(A \times A)^*$:

$$\forall u \in L, \quad \mathsf{ULSucc}_L(u) = \mathsf{Succ}_L(u) = v \Leftrightarrow |u| = |\mathsf{Succ}_L(u)| \ ,$$

and if $|\mathsf{Succ}_L(u)| > |u|$ then $\mathsf{ULSucc}_L(u)$ is undefined. It holds then that $\mathsf{ULSucc}_L(u)$ is undefined if and only if u is in $\mathsf{MaxL}(L)$, which is equivalent to $\mathsf{Succ}_L(u)$ being in $\mathsf{minL}(L)$. Moreover if u is in $\mathsf{MaxL}(L)$ and $v = \mathsf{Succ}_L(u)$ there is no word in L of length l, $|u| < l < |v|$.

For an ordered alphabet A, let $A_\$ = A \cup \{\$\}$ where $\$$ is a letter that does not belong to A, and which is by assumption smaller than every letter a in A. We associate with every language L of A^* the language $K = \$^*L$ of $A_\$^*$.

Let u and v in A^* such that $v = \mathsf{Succ}_L(u)$; if $|v| = |u|$ then $\mathsf{ULSucc}_K(u) = v$; if $|v| > |u|$, then let $k = |v| - |u|$ and since $u \in \mathsf{MaxL}(L)$ and $v \in \mathsf{minL}(L)$, it holds that $\mathsf{ULSucc}_K(\$^k u) = v$. Since ULSucc_K is injective, it holds:

$$\mathsf{Succ}_L(u) = v \qquad \Longleftrightarrow \qquad \exists! k \quad \mathsf{ULSucc}_K(\$^k u) = v \ . \tag{1}$$

Example 8 (*Ex. 1 cont.*)**.** Figure 8 shows the tree representing $K_1 = \$^*L_1$. We have seen that $\mathsf{Succ}_{L_1}(bb) = aaa$; it then holds that $\mathsf{ULSucc}_{K_1}(\$bb) = aaa$.

Proposition 7. If ULSucc_K is piecewise co-sequential, then so is Succ_L.

Proof. Let $\pi : A_\$^* \to A^*$ be the projection that erases the $\$$ symbol. With a slight abuse of notation we write:

$$\pi\left[\mathsf{ULSucc}_K \cap (\$^* A^* \times A^*)\right] = \mathsf{Succ}_L \ .$$

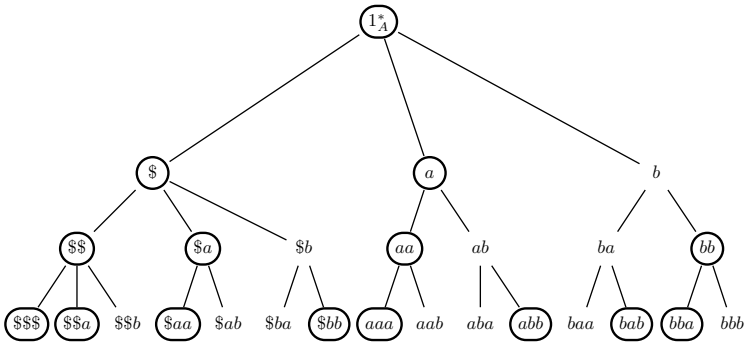


FIGURE 8. Tree representing K_1

Let $\tau_1, \ldots, \tau_k$ be the co-sequential transducers whose union realizes ULSucc_K and for every τ_i let:

$$\tau_i' = \tau_i \cap (\$^* A^* \times A^*) \ .$$

The transducer τ_i' is co-sequential as $(\$^* A^* \times A^*)$ is a recognizable relation (cf. [12, Ex. V.1.5]). In general, a projection applied on the input of a transducer does not preserve co-sequentiality but in our case, since the letters \$ are read only at the beginning of the words, it does, and this is what we prove now.

Let q be a state of τ_i' such that there exists a path from q to the final state of τ_i' whose input u is in A^*. From (1) follows that if $\$^k u$ is in $\mathsf{Dom}(\tau_i')$, then for no $l \neq k$, $\$^l u$ is in $\mathsf{Dom}(\tau_i')$. Then, there exists at most one path – which can be empty – from an initial state of τ_i' to q whose input is in $\$^*$; let v_q – which can be the empty word – be the output of this path.

Let τ_i'' be the transducer built from τ_i' by erasing all transitions whose input is \$ and by setting the initial function to $1|v_q$ for every q, where v_q is the word defined as above. The transducer τ_i' being co-sequential, so is τ_i''. We prove that the union of τ_i'' realizes Succ_L.

Let $v = \mathsf{Succ}_L(u)$, either $|u| = |v|$ and then there is a path from an initial to the final state of τ_i' labelled by $u|v$, this path still exists in τ_i''; or $|u| < |v|$ and then there exists a unique k such that there is a exactly one path – since τ_i' is co-sequential – from an initial state to the final state of τ_i' labelled $\$^k u|v$. Let q be the state reached after reading $\$^k$ in that path, then v_q is the prefix of v of length k and $u|v$ is a pair of words accepted by τ_i''. □

5. Proof of Theorem 1

From Proposition 7, it is sufficient to establish the following:

Theorem 3. *If L is a rational language of A^*, then ULSucc_L is a piecewise co-sequential function.*

Proof. To prove the theorem we build a set of co-sequential transducers whose union realizes the function ULSucc_L. Since L is rational, there exists a deterministic automaton recognizing it. Let $\mathcal{A} = \langle Q, A, \delta, i, T\rangle$ be such a deterministic automaton and which is fixed for the remaining of the proof. Each word of L and its uniform length successor share a longest common prefix. This common prefix leads to a unique state of $\mathcal{A}$. We then build, for every state, a set of concatenations of co-sequential transducers reading and writing the common prefix and a union of co-sequential transducers realizing the adequate function on the suffixes. The main point in the proof is to build these co-sequential transducers in order to make the concatenation co-sequential (Claim 3).

Let us note $q = p \cdot w$ if $q = \delta(p, w)$, $L_p = \{w \in A^* \mid p \cdot w \in T\}$ and $L'_p = \{w \in A^* \mid i \cdot w = p\}$. The L_p and L'_p are rational and the L'_p, for all p in Q, are pairwise disjoint as $\mathcal{A}$ is deterministic.
Let u in L and $v = \mathsf{ULSucc}_L(u)$. Let $w = u \wedge v$ be the longest common prefix of u and v: $u = w\,a\,u'$ and $v = w\,b\,v'$ with $a < b$ (this holds since $|u| = |v|$). Let $p = i \cdot w$ ($w \in L'_p$), this state is unique since $\mathcal{A}$ is deterministic. Furthermore both $a\,u'$ and $b\,v'$ belong to L_p.

Let $K_{p,a}$ be the set of maximal words of L_p which begin with an a:

$$K_{p,a} = \mathsf{MaxL}(a(a^{-1}L_p))$$

and let $H_{p,a}$ be the set of minimal words of L_p which begin with a letter c greater than a:

$$H_{p,a} = \mathsf{minL}(\bigcup_{c>a} c(c^{-1}L_p)) \ .$$

The sets $K_{p,a}$ and $H_{p,a}$ are rational sets and both have at most one word for every length. We prove in the following two Claims that, for every u and its successor v, the function that maps au' to bv' is the function that maps any word of $K_{p,a}$ to the word of $H_{p,a}$ of same length.

Claim 1: $a\,u' \in K_{p,a}$ and $b\,v' \in H_{p,a}$.

Proof of Claim 1. If $w \in L'_p$ and $w\,a\,u'$ is recognized by $\mathcal{A}$ then $a\,u' \in a(a^{-1}L_p)$. If $au' \notin K_{p,a}$ then there exists $a\,u'' \in a(a^{-1}L_p)$ with $|u'| = |u''|$ and such that $a\,u' \prec a\,u''$. Thus $u = w\,a\,u' \prec w\,a\,u'' \prec w\,b\,v' = v$, contradiction with $v = \mathsf{ULSucc}_L(u) = \mathsf{Succ}_L(u)$. We prove that $b\,v' \in H_{p,a}$ symmetrically. □

Claim 2: If $w' \in K_{p,a}$, $w'' \in H_{p,a}$, and $|w'| = |w''|$, then, for all w in L'_p, we have: $w\,w'' = \mathsf{ULSucc}_L(w\,w')$.

Proof of Claim 2. From the definition of $K_{p,a}$ and $H_{p,a}$, it holds that $w\,w'$ and $w\,w''$ are in L, $|w\,w'| = |w\,w''|$ and $w\,w' \prec w\,w''$ and hence w is a prefix of the longest common prefix of $w\,w'$ and of its successor. Since w' is in $K_{p,a}$ (the set of maximal

words of L_p beginning with a) the longest common prefix of $w\,w'$ and its successor is a prefix of w. Hence the longest common prefix of $w\,w'$ and its successor is w. From Claim 1 and since $K_{p,a}$ and $H_{p,a}$ have at most one word for every length, $w\,w'' = \mathsf{ULSucc}_L(w\,w')$. □

By Proposition 5 and Proposition 6 the synchronous product of the ray automata that recognizes them $K_{p,a}$ and $H_{p,a}$ is either a finite union of sequential transducers, or a finite union of co-sequential transducers $\tau_{p,1}, \ldots, \tau_{p,k}$. Let us choose the co-sequential option. There exists exactly one i such that $a\,u'$ belongs to $\mathsf{Dom}(\tau_{p,i})$ and then $b\,v' = (a\,u')\tau_{p,i}$.

To realize the concatenation with the prefix we need to build a union of concatenation with co-sequential and co-standard transducers that read and write every word of L'_p. For this, let $\mathcal{K}'_p$ be a *co-deterministic* and *co-standard* automaton recognizing L'_p and let κ'_p be the letter-to-letter transducer that realizes the identity on L'_p deduced from $\mathcal{K}'_p$, and thus co-sequential and co-standard.

The concatenations of κ'_p and $\tau_{p,i}$ realize the uniform length successor function: let $\theta_{p,i} = \kappa'_p||\tau_{p,i}$, we have $v = u\theta_{p,i}$. In the following Claim we prove that every $\theta_{p,i}$ is a piecewise co-sequential function by transforming the transducers κ'_p and $\tau_{p,i}$ to make the concatenation co-sequential. The idea is that the concatenation of the transducers is not co-sequential if $\tau_{p,i}$ is loop-initial and the incoming transition to the initial state has same label as one transition incoming to the final state of κ'_p. We somehow transfer the transition of κ'_p to $\tau_{p,i}$ and do this transformation again if needed. We finally prove that we only have to make a finite time this transformation to ensure that the concatenation is co-sequential.

Claim 3: $\theta_{p,i}$ realizes a piecewise co-sequential function.

Proof of Claim 3. We transform the transducer $\theta_{p,i}$ into a finite union of co-sequential transducers which realizes the same function.

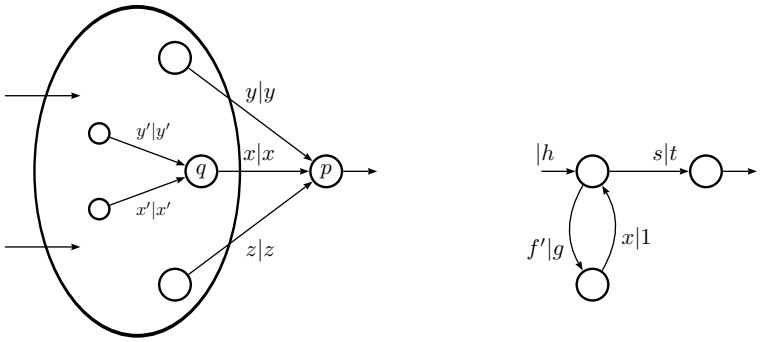
As $\tau_{p,i}$ is a ray transducer, two cases are possible:

(i) Either $\tau_{p,i}$ is not loop-initial or it is loop-initial but the input of the transition arriving at its initial state is different from the input of every transition arriving at the final state of κ'_p; from Property 3, $\theta_{p,i}$ is a co-sequential transducer.

(ii) The ray transducer $\tau_{p,i}$ is loop-initial and the input x of the transition arriving at the initial state of $\tau_{p,i}$ is equal to the input of one of the transitions arriving at the final state p of κ'_p. Such κ'_p and $\tau_{p,i}$ are represented in Figure 9. The loop on the initial state is labeled $f|g$ where $f = f'x$. For the figure be correct, we need to have f' not being the empty word, however the same reasoning apply when the loop is labeled by a single letter.

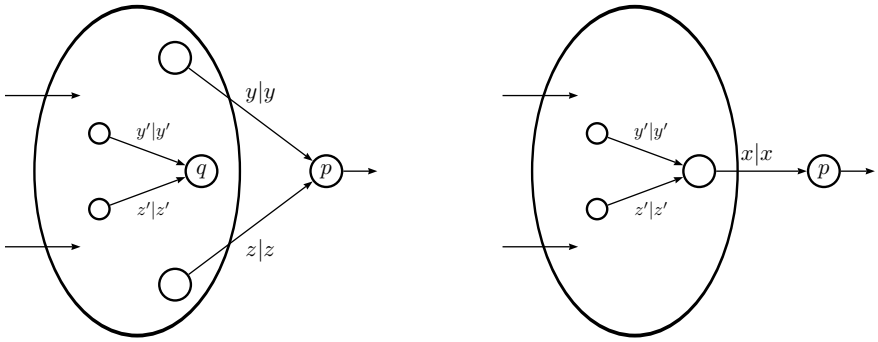
We then duplicate κ'_p; in the first copy $\kappa'_{p,1}$, we remove the transition $q \xrightarrow{x|x} p$. In the second copy we remove all the other transitions arriving at p (see Figure 10).

Let us prove by way of contradiction that the transducers $\kappa'_{p,1}||\tau_{p,i}$ and $\kappa'_{p,2}||\tau_{p,i}$ have disjoint domains. Let us suppose there is a word in both $\kappa'_{p,1}||\tau_{p,i}$ and $\kappa'_{p,2}||\tau_{p,i}$, it is then respectively of the form $ux(f'x)^k s$, with $ux \in L'_p$, and of



(a) κ'_p (b) $\tau_{p,i}$

FIGURE 9. κ'_p and $\tau_{p,i}$



(a) $\kappa'_{p,1}$ (b) $\kappa'_{p,2}$

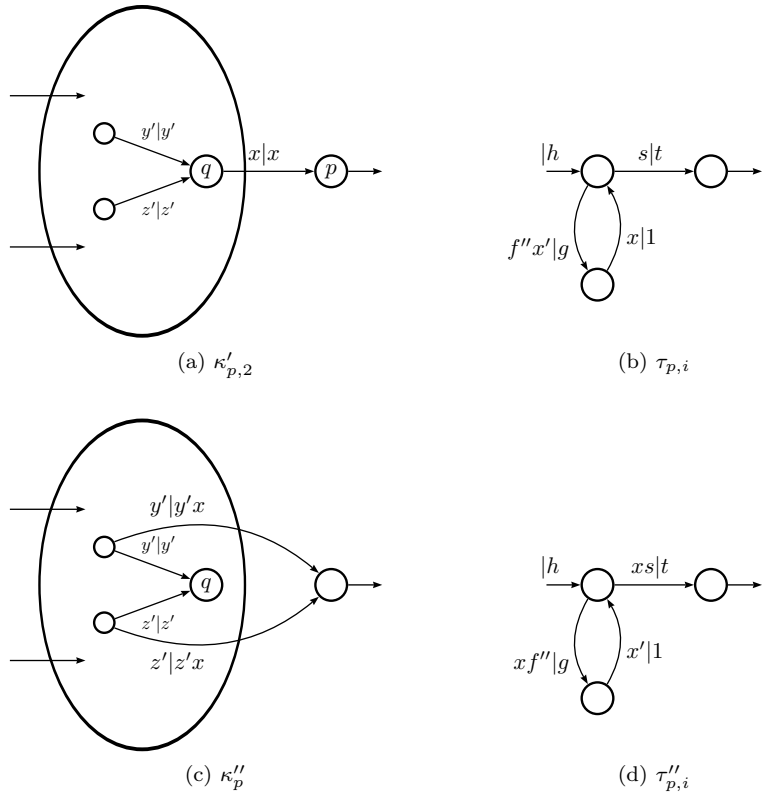
FIGURE 10. Splitting κ'_p into two pieces

the form $u'y(f'x)^{k'}s$, with $u'y \in L'_p$. Since $x \neq y$, we have $k' > k$ and $ux = u'y(f'x)^{k'-k}$. Since $\mathcal{A}$ is deterministic then ux and $u'y$ in L'_p implies that there is a circuit around p labeled by $(f'x)^{k'-k}$. Since $H_{p,a} \subset L_p$, the word $(f'x)^{k'-k}hgt$ is in L_p, and so is $(f'x)^{2(k'-k)}t$. These both words have same length, begin with same letter and $(f'x)^{k'-k}hgt > (f'x)^{2(k'-k)}t$. This is in contradiction with the definition of $\tau_{p,i}$ and hence $\kappa'_{p,1}||\tau_{p,i}$ and $\kappa'_{p,2}||\tau_{p,i}$ have disjoint domains.

The transducer $\kappa'_{p,1}||\tau_{p,i}$ falls into the preceding case (i) and is co-sequential, it remains to deal with $\kappa'_{p,2}$ and $\tau_{p,i}$ represented in Figure 11 (a) and (b).

Let us suppose that $f' = f''x'$. We replace $\kappa'_{p,2}$ by κ''_p and $\tau_{p,i}$ by $\tau''_{p,i}$ as represented in Figure 11 (c) and (d).

We have on one hand $\mathsf{Dom}(\kappa''_p) = \left[\mathsf{Dom}(\kappa'_{p,2})\right] x^{-1}$ and $u\kappa''_p = ux$ and, on the other hand, $\mathsf{Dom}(\tau''_{p,i}) = x\mathsf{Dom}(\tau_{p,i})$ and $(xv)\tau''_{p,i} = v\tau_{p,i}$.



(a) $\kappa'_{p,2}$ (b) $\tau_{p,i}$

(c) κ''_p (d) $\tau''_{p,i}$

FIGURE 11. Transformation of $\kappa_{p,2}$ and $\tau_{p,i}$ into κ''_p and $\tau''_{p,i}$

It holds then:

$$\begin{aligned}(uxv)\left[\kappa''_p||\tau''_{p,i}\right] &= (u\kappa''_p)\left((xv)\tau''_{p,i}\right)\\ &= \left((ux)\kappa'_{p,2}\right)(v\tau_{p,i}) = (uxv)\left[\kappa'_{p,2}||\tau_{p,i}\right]\end{aligned}$$

We write (as in a program) $\kappa'_p = \kappa''_p$, $\tau_{p,i} = \tau''_{p,i}$ and $\theta_{p,i} = \kappa'_p||\tau_{p,i}$, and we are back to the beginning of the proof of the claim: either $\theta_{p,i}$ is co-sequential and we are done, or we are in case (ii) and we perform the transformation again. This kind of transformation could go for ever and it is not difficult to build examples of κ and τ for which it really does (see remark below). However, in our case, the transformation has to stop at some point, and this is what we establish now.

Let n be the number of states of $\mathcal{A}$ we started from and let $l = |f|$. Suppose that we have done nl times the previous transformation, and matched every time the case (ii). Then f^n is a suffix of L'_p. There is then a state r of $\mathcal{A}$ such that there exist integers k_1 and k_2 such that $r \cdot f^{k_1} = r$ and $r \cdot f^{k_2} = p$. The first

equality implies that there exists a circuit labelled by f^{k_1} around r and, since $\mathcal{A}$ is deterministic, p belongs to that circuit and $p \cdot f^{k_1} = p$.

As we have already noted before, we prove that the circuit around p labeled k_1 is in contradiction with the definition of $\tau_{p,i}$. Since $\tau_{p,i}$ realizes a restriction of $H_{p,a} \bowtie K_{p,a}$, it holds that $f^{k_1}s$ and $hg^{k_1}t$ are words of L_p of same length and it also holds that a is the first letter of f and, if b is the first letter of hg, then $a < b$.

Using once the circuit around p, we have that both $f^{2k_1}s$ and $f^{k_1}hg^{k_1}t$ belong to L_p and these two words have same length. Since hg begins with b it holds: $f^{2k_1}s \prec f^{k_1}hg^{k_1}t$; hence $f^{2k_1}s$ is not a maximal word beginning with a and should not be in $\mathsf{Dom}(\tau_{p,i})$. This is in contradiction with the definition of $\tau_{p,i}$; it is then impossible to have to perform nl successive transformations: $\theta_{p,i}$ realizes a piecewise co-sequential function. □

Now we prove that the $\theta'_{p,i}$ realize exactly the uniform length successor function. Let $\theta'_{p,i}$ be the finite union of co-sequential transducers with pairwise disjoint domains built as above from $\theta_{p,i}$ and let r and s be two words of A^* such that:

$$r = s\theta'_{p,i} = s\theta_{p,i} \ .$$

First, s belongs to $\mathsf{Dom}(\theta_{p,i}) = \mathsf{Dom}(\kappa_p)\,\mathsf{Dom}(\tau_{p,i})$ and necessarily $s = w\,w'$ with $w \in L'_p$ and $w' \in K_{p,a} \subset L_p$. The word s thus belongs to L. It follows then that $r = w\,w''$ with w'' in $H_{p,a}$ and $|w''| = |w'|$. According to Claim 2 we have $w\,w'' = \mathsf{ULSucc}_L(w\,w')$.

As ULSucc_L is a function, the domains of the $\theta_{p,i}$ are pairwise disjoint, and this completes the proof of Theorem 3 and thus of Theorem 1. □

Remark 3. A pair of transducers κ and τ, such that κ is co-standard and co-sequential and τ is a co-sequential ray-transducer, for which the transformation performed in the proof of Claim 3 goes for ever is represented in Figure 12.

(a) A transducer κ (b) A transducer τ

FIGURE 12. Transducers for which the construction goes for ever

Remark 4. Every construction involved in the proof of Theorem 3 is symmetrical in the sense that the automata can be chosen to be deterministic or co-deterministic and the transducers sequential or co-sequential (for instance, the 'τ_i') but for one point: the automata $\mathcal{K}'_p$ can be chosen to be co-deterministic and co-standard but it is not possible to assume, in general, that they can be chosen *deterministic* and *co-standard* (which would be necessary to complete the symmetrical construction).

References

[1] Angrand, P.-Y., Sakarovitch, J. and de Souza, R. Sequential transducer cascades. *In preparation*.

[2] Berstel, J. *Transductions and Context-Free Languages*. Teubner, 1979.

[3] Berthé, V., Frougny, Ch., Rigo, M., and Sakarovitch, J. On the cost and complexity of the successor function. In *Proc. WORDS 2007* (P. Arnoux, N. Bédaride, and J. Cassaigne, eds.), Tech. Rep., Institut de mathématiques de Luminy (Marseille), (2007) 43–56.

[4] Berthé, V., Frougny, Ch., Rigo, M., and Sakarovitch, J. On the concrete complexity of the successor function. *In preparation*.

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

[6] Choffrut, Ch. and Schützenberger, M. P. Décomposition de fonctions rationnelles, *Proc. STACS'86* (B. Monien, G. Vidal-Naquet, eds.) LNCS 210 (1986), 213–226.

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

[8] Lecomte, P. and Rigo, M. Numeration systems on a regular language, *Theory Comput. Syst. 34* (2001) 27–44.

[9] Perrin, D. Finite automata, in: *Handbook of Theoretical Computer Science* (J. van Leeuwen, ed.) Elsevier (1990) vol. B, 1–53.

[10] Reutenauer Ch. Une caractérisation de la finitude de l'ensemble des coefficients d'une série rationnelle en plusieurs variables non commutatives, *C. R. Acad. Sci. Paris 284* Série A (1977) 1159–1162.

[11] Sakarovitch, J. Deux remarques sur un théorème de S. Eilenberg. *RAIRO Theor. Informatics and Appl. 17* (1983), 23–48.

[12] Sakarovitch, J. *Eléments de théorie des automates*. Vuibert, 2003. English corrected edition: *Elements of Automata Theory*, Cambridge University Press, 2009.

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

[14] Shallit, J. Numeration systems, linear recurrences, and regular sets. *Inform. and Comput. 113* (1994), 331–347.

Communicated by (The editor will be set by the publisher).