On the cost and complexity of the successor function

Valérie Berthé[1], Christiane Frougny[2], Michel Rigo[3], and Jacques Sakarovitch[4]

[1] LIRMM, CNRS-UMR 5506, Univ. Montpellier II, 161 rue Ada,
F-34392 Montpellier Cedex 5, France,
berthe@lirmm.fr
[2] LIAFA, UMR 7089, case 7014, 2, place Jussieu,
F-75251 Paris Cedex 5, France, and University Paris 8,
Christiane.Frougny@liafa.jussieu.fr
[3] Université de Liège, Institut de Mathématiques, Grande Traverse 12 (B 37),
B-4000 Liège, Belgium,
M.Rigo@ulg.ac.be
[4] LTCI, ENST/CNRS, 46, rue Barrault,
F-75634 Paris Cedex 13, France,
sakarovitch@enst.fr

Abstract. For a given numeration system, the successor function maps the representation of an integer n onto the representation of its successor $n+1$. In a general setting, the successor function maps the n-th word of a genealogically ordered language L onto the $(n+1)$-th word of L. We show that, if the ratio of the number of elements of length $n+1$ over the number of elements of length n of the language has a limit $\beta > 1$, then the amortized cost of the successor function is equal to $\beta/(\beta-1)$. From this, we deduce the value of the amortized cost for several classes of numeration systems (integer base systems, canonical numeration systems associated with a Parry number, abstract numeration systems built on a rational language, and rational base numeration systems).

1 Introduction

Deeply linked with numeration systems and the representation of numbers (real numbers or integers), the so-called *odometers* or "adding machines" play a central role in various fields of mathematics and theoretical computer science: in approximation of ergodic systems [20], in symbolic dynamics (see for instance [8] for two-sided dynamical systems) or in fractal geometry (they correspond to tiles exchange for the Rauzy fractal [4, 6]). They have also been studied from a combinatorial and topological point of view (see [13] where the case of linear numeration systems was first investigated).

From a dynamical point of view, it is natural to consider an odometer as a map acting on infinite words over a finite alphabet of digits. By putting an infinite sequence of zeroes in front of any finite word, the set of representations of all the integers is embedded into the set of infinite words onto which the odometer acts.

We focus in the present paper on the action of the odometer on finite words. The function acting on finite words representing integers and which maps the representation of an integer n onto the representation of $n+1$ is usually called the *successor function*. In the context of abstract numeration systems where a language L over a totally ordered alphabet $(A, <)$ is ordered by the radix order, the *successor function* maps the n-th word in L onto the $(n+1)$-th word.

For positional numeration systems, addition of 1 to compute the successor of the representation of a non-negative integer n leads to the apparition of a carry which can propagate to the left (as usual integers are represented most significant digit first). The representation of $n+1$ is obtained when there is no more carry to take into account. The unaffected prefix (if any) of the representation of n is then copied as prefix of the representation of $n+1$. For abstract numeration systems (see Section 9.3), the successor function can simply be seen as a function acting on words and possibly leaving unchanged the prefix of the argument. It is quite natural to consider two kinds of questions about the successor function.

The first one addressed in this paper is concerned with the estimation of the length of the carry propagation when applying the successor map on the first n integers (or more generally on the first n elements in a given language). This leads to the notion of (amortized) cost, i.e., the average carry propagation when applying the successor function. The second question is a computational issue: estimating the number of operations (in classical terms of Turing machines complexity) required to compute the representations of the first n integers from the first one by applying n times the successor function. This leads to the notion of (amortized) complexity, i.e., the average amount of computations required to obtain the successor of an element.

It turns out that in particular cases, namely, when the successor function can be realized by a right sequential letter-to-letter finite transducer (see Section 3 fro definitions), cost and complexity coincide. So realization of the successor by a letter-to-letter finite transducer was part of the motivation for this work and is discussed in Section 7: the successor function on a rational language can be realized by a *right* letter-to-letter finite transducer, which is not sequential in general.

Let us mention that in [3] this question of complexity was considered in the very special case of numeration systems built on a linear recurrent sequence of order 2 and it was shown that the amortized cost was bounded by a constant. It turns out that in general the amortized cost of the successor function can be obtained thanks to a trie traversal (see Section 6). More precisely we show that if the ratio of the number of elements of length $n+1$ over the number of elements of length n of the language has a limit $\beta > 1$, then the amortized cost of the successor function is equal to $\beta/(\beta-1)$ (Theorem 1). From this, we deduce in Section 9 the value of the amortized cost for several classes of numeration systems (integer base systems, canonical numeration systems associated with a Parry number [16, Chapter 7], abstract numeration systems built on a rational language whose minimal automaton has primitive components [15], and rational base numeration systems [1]). We also discuss the differences between cost and complexity based on computational results from [9] where sequential properties of the successor function for positional numeration systems were investigated.

2 Preliminaries and notation

2.1 Words

Let $(A, <)$ be a finite alphabet, totally ordered. The empty word is denoted ε. The length of a word x of A^* is denoted $|x|$. The *radix order* (also called the *genealogical order*), denoted by $\prec$, is defined as follows. Let x and y be two words in A^*; $x \prec y$ if $|x| < |y|$, or $|x| = |y|$ and $x = pas$, $y = pbt$ with a and b in A, $a < b$ (i.e., for two words of same length, the radix order coincides with the usual lexicographic order).

In all that follows, $L \subseteq A^*$ denotes a language ordered by the radix order. Let i be a non-negative integer; the $(i+1)$-th word of L in the radix order is denoted $\langle i \rangle_L$.

The *successor* of a word x in $L \subseteq A^*$ is the unique word $y \in L$ such that

$$(x \prec y) \wedge (\forall z \in L)\,((x \prec z) \Rightarrow ((y = z) \vee (y \prec z))).$$

The *successor function* on L is the function $\mathrm{Succ}_L : A^* \to A^*$ that maps a word $w = \langle i \rangle_L$ of L onto its successor $\langle i+1 \rangle_L$ in L. The *odometer* is an extension to infinite words [13, 5] or to bi-infinite words [8] of the successor function.

A language L is said to be *prefix-closed* if every prefix of a word of L is in L.

The number of words in L of length n is denoted $\mathbf{u}_L(n)$ and the number of words in L of length $\leq n$ is denoted $\mathbf{v}_L(n)$.

2.2 Tries

The *trie* $\mathcal{T}_L$ of a language $L \subseteq A^*$ is a tree where the edges are labelled by letters from A, and the nodes are labelled by prefixes of words of L. The root is associated with the empty word ε. If

w is a prefix of a word of L and a is a letter, there is an edge with label a going from the node labelled w to the node labelled wa if wa is a prefix of a word of L.

When L is a prefix-closed language ordered by radix order, the first level of the trie contains all words of L of length 1, the second level contains all words of L of length 2, and so on, in lexicographic order from left-to-right. Thus a breadth-first traversal of $\mathcal{T}_L$ gives all the words of L in radix order.

3 Models of computation

3.1 Automata

We assume that the reader has some basic knowledge in automata theory (see for instance [18]). The set of rational languages over A is denoted $\mathrm{Rat}(A^*)$. A deterministic finite automaton (or simply DFA) is denoted (Q, q_0, A, δ, F). Let L in $\mathrm{Rat}(A^*)$ having $\mathcal{M} = (Q, q_0, A, \delta, F)$ as a minimal automaton. We assume that $\mathcal{M}$ is trim, i.e., accessible and co-accessible. In particular, this means that the possible "sink state" has been removed.

For any language $K \subseteq A^*$, $\max(K)$ is the language made up of the largest words of each length in K for the genealogical ordering induced by the ordering of A, that is to say,

$$\max(K) = \{w \in K \mid \forall x \in K,\ |x| = |w| \Rightarrow x \prec w\}.$$

It is folklore that if K is rational, $\max(K)$ is also rational (see for instance [19]).

Let A and B be two alphabets. A *transducer* is an automaton $\mathcal{A} = (Q, I, A^* \times B^*, E, F)$ where the edges of E are labelled by couples in $A^* \times B^*$. It is said to be *finite* if the set Q of states and the set E of edges are finite. I is the set of initial states and F is the set of terminal states. A *letter-to-letter* automaton is a transducer with labels in $A \times B$. We will use transducers with a terminal function ω, instead of terminal states, as follows, see [10]. The terminal function ω is a function from Q to $\mathcal{P}(A^* \times B^*)$. The relation realized by $\mathcal{A} = (Q, I, A^* \times B^*, E, \omega)$ is the set of $(u, v) \in A^* \times B^*$ such that there exist u', u'' in A^*, and v', v'' in B^*, with $u'u'' = u$ and $v'v'' = v$, an initial state p of I, and a path labelled by (u', v') from p to a state q, with $(u'', v'') \in \omega(q)$.

A transducer is said to be (left) *sequential* is there is only one initial state and if the input automaton obtained by taking the projection on A is a *deterministic* automaton.

A relation is said to be *synchronized* if it is realized by a finite letter-to-letter transducer with a terminal function taking its value into $\{(S \times \{\varepsilon\}) \cup (\{\varepsilon\} \times T) \mid S \in \mathrm{Rat}(A^*), T \in \mathrm{Rat}(B^*)\}$.

A relation R is said to have *bounded length differences* if there exists a k such that for any (u, v) in R, the difference between $|u|$ and $|v|$ is less than k.

The automata defined so far work implicitly from left to right. The same notions for a processing from right to left can be defined accordingly; they are called *right* automata.

3.2 On-line computability

This notion is traditionally defined for a left-to-right processing, but here we will use the right-to-left notion. A function $\varphi : A^* \to B^*$ is said to be *right on-line computable* with delay δ if there exists an integer $\delta \geq 0$ such that when $x = x_k \cdots x_0 \in A^*$ and $y = y_\ell \cdots y_0 \in B^*$, then $y = \varphi(x)$ if and only if for any $j \geq 0$, there exists a function $\Phi_j : A^{j+\delta+1} \to B$ such that $y_j = \Phi_j(x_{j+\delta} \cdots x_0)$ (in this definition we assume that if $j + \delta > k$, then x is extended to the left by zeroes or a special symbol). For instance, the successor function in the Fibonacci numeration system is right on-line computable with delay 1, see Section 5.

4 Cost and complexity

We recall that L is a language ordered by the radix order. Let x be in L. The *cost for computing* $\mathrm{Succ}_L(x)$ is defined as the length of the carry propagation. It is also equal to half the length of the path in the trie $\mathcal{T}_L$ from x to $\mathrm{Succ}_L(x) = y$ if x is not in $\max(L)$, and to $|y|$ else.

Given two words x and y of the same length, we denote by $\mathsf{lcp}(x,y)$ the length of the longuest common prefix of x and y. If x has length k and y has length ℓ, with $\ell \geq k$, we set

$$\Delta(x,y) = \begin{cases} \ell - \mathsf{lcp}(x,y) & \text{if } k = \ell \\ \ell & \text{if } k < \ell. \end{cases}$$

With this notation, the cost for computing $\mathrm{Succ}_L(\langle i\rangle_L)$ is $\Delta(\langle i\rangle_L, \langle i+1\rangle_L)$.

Definition 1. *The (amortized)* cost of Succ_L *is*

$$\mathsf{Cost}(\mathrm{Succ}_L) = \lim_{N\to\infty} \frac{1}{N} \sum_{i=0}^{N-1} \Delta(\langle i\rangle_L, \langle i+1\rangle_L).$$

Suppose that P is a program (i.e., a Turing machine) which, for every $i \geq 0$, computes $\mathrm{Succ}_L(\langle i\rangle_L)$ in $\mathsf{Op}(P, \langle i\rangle_L)$ operations (i.e., elementary operations of tape reading or writing and head moves for computing with a Turing machine $\mathrm{Succ}_L(\langle i\rangle_L)$ from $\langle i\rangle_L$ written on its tape).

Definition 2. *The (amortized)* complexity of P *is*

$$\mathsf{comp}(P) = \lim_{N\to\infty} \frac{1}{N} \sum_{i=0}^{N-1} \mathsf{Op}(P, \langle i\rangle_L).$$

The (amortized) complexity of Succ_L *is*

$$\mathsf{Comp}(\mathrm{Succ}_L) = \inf\{\mathsf{comp}(P) \mid P \text{ computes } \mathrm{Succ}_L\}.$$

5 Example: the Fibonacci numeration system

The Fibonacci numeration system is a positional numeration system defined on the linear recurrent sequence $F = (F_n)_{n\geq 0}$ where $F_0 = 1$, $F_1 = 2$ and $F_{n+2} = F_{n+1} + F_n$ for all $n \geq 0$. Non-negative integers are represented over $\{0,1\}$ in a greedy way as sums of elements of F and these representations are exactly words where the factor 11 is forbidden and starting with a 1, that is to say, $L(F) = 1\{0,1\}^* \setminus \{0,1\}^*11\{0,1\}^* \cup \{\varepsilon\}$ (see for instance [16, Chapter 7]). The greedy representation of 0 is the empty word ε, but it is more usual to denote it 0.

As an example, here are the representations of length ≤ 5 and the corresponding cost for computing Succ.

$$\begin{array}{cccccccccc} 0\ (\mathbf{1}) & 1(\mathbf{2}) & 10 & (\mathbf{3}) & 100 & (\mathbf{1}) & 101 & (\mathbf{4}) & 1000 & (\mathbf{1}) \\ 1001\ (\mathbf{2}) & 1010(\mathbf{5}) & 10000\ (\mathbf{1}) & 10001\ (\mathbf{2}) & 10010\ (\mathbf{3}) & 10100\ (\mathbf{1}) & 10101\ (\mathbf{6}) & & & \end{array}$$

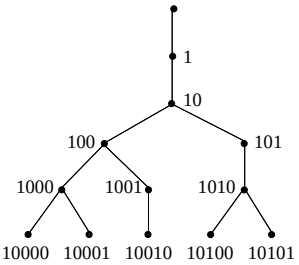


Fig. 1. The trie for the Fibonacci numeration system

The successor function in the Fibonacci numeration system is right sequential and right on-line computable with delay 1 (see [9]), so it can be realized by a right sequential finite transducer with delay 1 (see the proof of Proposition 6). In Fig. 2, the doubly circled loop indicates that the words of the language are simply recopied.

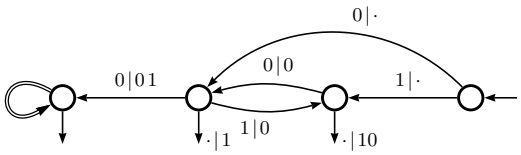


Fig. 2. The successor function for the Fibonacci numeration system

6 Traversal of a trie

We first begin with a technical result.

Lemma 1. *Let $x(n)$, $n \geq 0$, be an increasing sequence of positive integers. Let $y(n) = \sum_{i=0}^{n} x(i)$. Then the following are equivalent:*
(i) there exists a number $\beta > 1$ such that $lim_{n\to\infty} x(n+1)/x(n) = \beta$
(ii) there exists a number $\beta > 1$ such that $lim_{n\to\infty} y(n+1)/y(n) = \beta$
(iii) there exists a number $\beta > 1$ such that $lim_{n\to\infty} y(n)/x(n) = \beta/(\beta-1)$.

Proof. (i) implies (ii): from (i) one has $x(n+1) \sim \beta x(n)$, hence $\sum_{i=0}^{n} x(i+1) \sim \beta \sum_{i=0}^{n} x(i)$, and (ii) follows.
(ii) implies (i): from (ii) and $y(n+1) = y(n) + x(n+1)$, one has $x(n+1) \sim (\beta-1)y(n)$, and (i) follows.
(ii) is equivalent to (iii) by noticing that $y(n+1)/y(n) = 1 + x(n+1)/y(n) = 1 + (x(n+1)/x(n)) \cdot (x(n)/y(n))$, and thus $y(n)/x(n) = \frac{x(n+1)/x(n)}{(y(n+1)/y(n))-1}$. ∎

The breadth-first traversal of the trie $\mathcal{T}_L$ of the prefix-closed language L gives all words of L in radix order. Let us fix n. The n-th level contains the $\mathbf{u}_L(n)$ elements of L of length n. We denote these elements by $w_i \prec \cdots \prec w_{i+k}$ (i and k depend on n) and the first element of length $n+1$ by w_{i+k+1}. Let r be the highest branching node (i.e., the highest node having at least two sons) in the trie, and let k_0 be its level. Wet set $\pi(n)$ as the length of the smallest path going from the smallest element w_i of length n to the smallest element of length $n+1$ and passing consecutively through $w_{i+1}, \ldots, w_{i+k}$. We assume that all branches in $\mathcal{T}_L$ are infinite.

Then

$$\pi(n) = 2(\mathbf{u}_L(n) + \mathbf{u}_L(n-1) + \cdots + \mathbf{u}_L(k_0+1)) + 1.$$

Thus the cost for computing the successor of all the words of L of length n, denoted by $C_L(n)$, is equal to half the length of the path from the smallest element of length n to the greatest element of length n plus $(n+1)$. So

$$C_L(n) = \mathbf{u}_L(n) + \mathbf{u}_L(n-1) + \cdots + \mathbf{u}_L(k_0+1) + k_0 + 1. \tag{1}$$

Theorem 1. *Let L be a radix order language that is prefix-closed and such that all branches in $\mathcal{T}_L$ are infinite. Suppose that $\lim_{n\to\infty} \mathbf{u}_L(n+1)/\mathbf{u}_L(n)$, or $\lim_{n\to\infty} \mathbf{v}_L(n+1)/\mathbf{v}_L(n)$, exists and is equal to some $\beta > 1$. Then*

$$\mathsf{Cost}(\mathrm{Succ}_L) = \frac{\beta}{\beta-1}.$$

Proof. From Equation (1) we get that $C_L(n) = \mathbf{v}_L(n) - c_0$, where c_0 is a positive constant. Thus

$$\mathsf{Cost}(\mathrm{Succ}_L) = \lim_{n\to\infty} \frac{1}{\mathbf{v}_L(n)} \sum_{i=0}^{n} C_L(n) = \frac{\beta}{\beta-1}$$

by Lemma 1. ∎

7 Successor function on a rational language

The aim of this section is to show that when the language is rational, the successor function can be realized by a *right* letter-to-letter finite transducer.

Before proving the result we need some lemmas, see [18].

Lemma 2. *The radix order relation $\rho : A^* \to A^*$ which maps a word u onto $\{v \in A^* \mid u \prec v\}$ is a synchronized relation.*

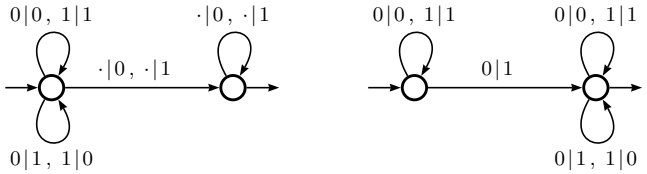


Fig. 3. A synchronized transducer for the radix order relation on $A = \{0, 1\}$

Lemma 3. *Let L be a rational language. The function $\iota_L : A^* \to A^*$ which is the identity on L, and is undefined outside, is a synchronized function.*

Recall two results from [10].

Proposition 1. *The composition of synchronized relations is a synchronized relation.*

Proposition 2. *The family of synchronized relations is an effective Boolean algebra.*

We can now prove the following result.

Proposition 3. *Let L be a rational language. The successor function* Succ_L *is realized by a (left) letter-to-letter finite transducer.*

Proof. Let $S \subseteq A^*$, then $\min(S) = S \setminus \rho(S)$. Denote $\Gamma_L : A^* \to A^*$ the relation that maps a word u of L onto the set $\{v \in L \mid u \prec v\}$. Then

$$\begin{aligned}\mathrm{Succ}_L(u) &= \min(\Gamma_L(u)) \\ &= \min(\iota_L \circ \rho \circ \iota_L(u)) \\ &= \iota_L \circ \rho \circ \iota_L(u) \setminus \rho \circ \iota_L \circ \rho \circ \iota_L(u)\end{aligned}$$

and the result follows by Propositions 1 and 2. ∎

Corollary 1. *On a rational language L the successor function* Succ_L *can be realized by a* right *letter-to-letter finite transducer.*

Proof. Since every synchronized relation with finite image has bounded-length differences [10], and since a function has finite image, it follows that the successor function Succ_L has bounded length differences when L is rational, so it is possible to build a *right* letter-to-letter finite transducer realizing the function. ∎

8 Cost and complexity for rational successor functions

A finite automaton can be seen as a program, thus the following results can be deduced from Corollary 1.

Proposition 4. *If L is a rational language then* $\mathsf{Comp}(\mathrm{Succ}_L) \geq \mathsf{Cost}(\mathrm{Succ}_L)$.

Proof. The number of elementary operations to compute the successor $\mathrm{Succ}_L\langle i\rangle_L$ from $\langle i\rangle_L$ is at least equal to the length of the carry propagation. Indeed, with $\langle i\rangle_L$ on its tape, one has at least to consider the number of head moves corresponding to propagation carry. ∎

More can be said when the successor function is *right sequential*:

Proposition 5. *If* Succ_L *is realized by a right sequential letter-to-letter finite transducer, then*

$$\mathsf{Comp}(\mathrm{Succ}_L) = \mathsf{Cost}(\mathrm{Succ}_L).$$

Note that this result does not hold anymore when the successor function is *left* sequential, and not *right* sequential (see Example 1 below).

More generally we have:

Proposition 6. *If* Succ_L *is a function which is right on-line computable with delay δ, and is realized by a right finite transducer, then*

$$\mathsf{Comp}(\mathrm{Succ}_L) = \mathsf{Cost}(\mathrm{Succ}_L) + \delta.$$

Proof. If Succ_L is a function which is right on-line computable with delay δ, this means that to output a letter of the result it is enough to look δ positions ahead on the left of the current position. Moreover, under these two hypothesis Succ_L is realizable by a right sequential finite transducer with delay δ and of a special form, see [9, 11] (see also Section 5 for the Fibonacci case). So the result follows. ∎

9 Applications

9.1 Integer base p

The language of the p-expansions of the non-negative integers is $L_p = \{1, \ldots, p-1\}\{0, \ldots, p-1\}^* \cup \{0\}$. The number of words of length $\leq n$ is $\mathbf{v}_{L_p}(n) = p^n$. It is also well known that Succ_{L_p} is realized by a right sequential letter-to-letter finite transducer. Thus from Proposition 5 follows

Proposition 7. *In integer base p*

$$\mathsf{Cost}(\mathrm{Succ}_{L_p}) = \mathsf{Comp}(\mathrm{Succ}_{L_p}) = \frac{p}{p-1}.$$

9.2 Beta-numeration

Let $V = (v_n)_{n\geq 0}$ be a strictly increasing sequence of integers with $v_0 = 1$. By a greedy algorithm [7], every non-negative integer N is given a V-expansion of the form $a_k \cdots a_0$, with a_i non-negative digit $< v_{i+1}/v_i$, such that $N = \sum_{i=0}^{k} a_i v_i$. By definition of the greedy expansions, the number of words of length $\leq n$, $\mathbf{v}_L(n)$, is equal to v_n. On the set $L(V)$ of the greedy V-expansions of the non-negative integers the successor function is defined as above.

An interesting case is the following one. Let $\beta > 1$ be a real number. Any real number $z \in [0, 1]$ can be represented by a greedy algorithm as $z = \sum_{i=1}^{+\infty} z_i \beta^{-i}$ with $z_i \in \{0, \ldots, \lceil \beta \rceil - 1\}$ for all $i \geq 1$. The greedy sequence $(z_i)_{i\geq 1}$ corresponding to a given real number z is the greatest in the lexicographical order, and is said to be the *β-expansion* of z, see [17]. If the β-expansion of 1 is finite or eventually periodic then β is said to be a *Parry number*. If the β-expansion of 1

is finite β is said to be a *simple* Parry number. When β is a Parry number a linear recurrent sequence $V_\beta = (v_n)_{n\geq 0}$ of integers can be canonically associated with β, see [2], [16, Chapter 7] for details. For instance, the canonical linear numeration system associated with the golden mean is the Fibonacci numeration system, see Section 5.

When β is a Parry number, the language $L_\beta = L(V_\beta)$ of the greedy expansions of all the non-negative integers is rational and the automaton accepting L_β has a very special form (see for instance [12]). Note that the language L_β is prefix-closed.

It is well known that $\lim_{n\to\infty} v_{n+1}/v_n = \beta$. Note that if $L' = 0^*L_\beta$ then $\mathbf{v}_{L_\beta}(n) = \mathbf{u}_{L'}(n)$.

Then from Theorem 1 follows.

Proposition 8. *If $\beta > 1$ is a Parry number, the cost of the successor function for the canonical linear numeration system associated with β is*

$$\mathsf{Cost}(\mathrm{Succ}_{L_\beta}) = \frac{\beta}{\beta - 1}.$$

The conditions under which the successor function is right sequential are completely known [9].

Proposition 9. *The successor function for the canonical linear numeration system associated with β is right sequential if and only if β is a simple Parry number. If the β-expansion of 1 is of length m, then the successor function is right on-line computable with delay $m-1$.*

So from Proposition 6 follows

Proposition 10. *If $\beta > 1$ is a simple Parry number and the β-expansion of 1 is of length m, the complexity of the successor function for the canonical linear numeration system associated with β is*

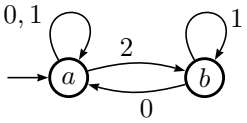
$$\mathsf{Comp}(\mathrm{Succ}_{L_\beta}) = \frac{\beta}{\beta - 1} + (m-1).$$

We now give two examples where the successor function is not right sequential.

Example 1. Take $\tau = (3+\sqrt{5})/2$, i.e., the square of the golden ratio. Then the τ-expansion of 1 is equal to 21^ω, so τ is a non-simple Parry number. From Proposition 8 we know that

$$\mathsf{Cost}(\mathrm{Succ}_{L_\tau}) = \frac{\tau}{\tau - 1} \simeq 1.61803.$$

The automaton below recognizes the language $L' = 0^*L_\tau$.



Since the successor function on L_τ is not realizable by a right sequential finite transducer, the complexity is not equal to the cost.

Denote the conjugate of τ as $\tau' = \frac{3-\sqrt{5}}{2}$. The number of paths of length n going from state a to a in the automaton is

$$\mathbf{p}(n) = \frac{5+\sqrt{5}}{10}\tau^n + \frac{5-\sqrt{5}}{10}\tau'^n.$$

The number of words of length n accepted by the automaton is

$$\mathbf{u}(n) = \frac{5+3\sqrt{5}}{10}\tau^n + \frac{5-3\sqrt{5}}{10}\tau'^n.$$

Two kinds of words of length n need a special attention: those of the form $u21^{i-1}$, with $|u| = n-i$, $i = 1, \ldots, n$, and those of the form $u01^{i-1}$ with $|u| = n-i$, $i = 2, \ldots, n$. Indeed, when a Turing machine has such a word on its tape with the head on the rightmost position, the machine has to go on the left to read the first 0 or 2 before being able to decide what is the rightmost digit to output and act accordingly (at least going back to the rightmost position). Let us say that the number of operations required is $2i$ for each of these "special" words then, since all the other words need a computation requiring one unit. The word 1^n occurs once amongst the words of length n, but a single word will not change the asymptotic computation so we do not take it into account in what follows. So we get the following

$$\begin{aligned}\mathsf{Comp}(\mathrm{Succ}_{L_\tau}) &= \lim_{n\to+\infty}\left(\sum_{i=1}^{n}\frac{\mathbf{p}(n-i)\,2i}{\mathbf{u}(n)}+\sum_{i=2}^{n}\frac{\mathbf{p}(n-i)\,2i}{\mathbf{u}(n)}+\frac{\mathbf{u}(n)-\sum_{i=2}^{n}2\mathbf{p}(n-i)-\mathbf{p}(n-1)}{\mathbf{u}(n)}\right)\\ &= 2\sqrt{5}-2 \simeq 2.47214.\end{aligned}$$

Example 2. Take the sequence $V = (v_n)_{n\geq 0}$ with $v_n = 2^{n+1}-1$. Then the successor function on $L(V)$ is realizable by a left sequential finite transducer, but it is impossible to do it with a right sequential one (see [9]). So, the complexity is still greater than the cost, which is equal to 2 since $\lim_{n\to\infty} v_{n+1}/v_n = 2$.

9.3 Abstract numeration system

An *abstract numeration system* [15] is a triple $S = (L, A, <)$ where L is a infinite rational language over a totally alphabet $(A, <)$. The representation of $i \geq 0$ in S denoted by $\langle i\rangle_L$ is the $(i+1)$-th word in the radix ordered language L. As already said above, the successor function on a rational language L is simply the function mapping $\langle i\rangle_L$ onto $\langle i+1\rangle_L$.

Various aspects of abstract numeration systems have been studied from [15] and in this framework odometers acting accordingly on infinite words were presented in [5].

Proposition 11. *Let $S = (L, A, <)$ be an abstract numeration system built on a rational language whose trim minimal automaton $\mathcal{M}$ is primitive. If β is the dominating eigenvalue of $\mathcal{M}$, then the cost of the successor function for S is*

$$\mathsf{Cost}(\mathrm{Succ}_L) = \frac{\beta}{\beta-1}.$$

Proof. From Perron's Theorem, the number of words of L of length n satisfies $\lim_{n\to\infty}\mathbf{u}_L(n+1)/\mathbf{u}_L(n) = \beta$. ∎

This result can also be extended to a more general situation.

Proposition 12. *Let L be a rational language having a trim minimal automaton $\mathcal{M}$ whose each strongly connected component of is primitive. Let $S = (L, A, <)$ be an abstract numeration system built on L. If β is the maximal value amongst the dominating eigenvalue of the strongly connected component of $\mathcal{M}$, then the cost of the successor function for S is*

$$\mathsf{Cost}(\mathrm{Succ}_L) = \frac{\beta}{\beta-1}.$$

Proof. Since each strongly connected component $\mathcal{C}$ of $\mathcal{M}$ is primitive, one can thanks to Perron's Theorem associate with $\mathcal{C}$ a unique dominating eigenvalue $\beta_{\mathcal{C}}$. Let $\beta = \max\beta_{\mathcal{C}}$ where the maximum is taken over all the strongly connected components of $\mathcal{M}$. If β occurs exactly in t components of $\mathcal{M}$, then the number of words of length n in L is such that $\mathbf{u}_L(n) \asymp n^{t-1}\beta^n$ (see for instance [14]). Therefore $\lim_{n\to+\infty}\mathbf{u}_L(n+1)/\mathbf{u}_L(n) = \beta$ hence the conclusion follows. ∎

9.4 Rational base numeration systems

This part of the paper is devoted to the study of the successor function in a quite different type of numeration, where the base is a rational number $\frac{p}{q}$, where $p > q \geq 1$ are two co-prime integers [1].

Let N be any positive integer; let us write $N_0 = N$ and, for $i \geq 0$, write

$$q\,N_i = p\,N_{i+1} + a_i \tag{2}$$

where a_i is the remainder of the division of $q\,N_i$ by p, and thus belongs to $A = \{0, \ldots, p-1\}$. Since N_{i+1} is strictly smaller than N_i, the division (2) can be repeated only a finite number of times, until eventually $N_{k+1} = 0$ for some k. This algorithm produces the digits a_0, a_1, …, a_k, and it holds:

$$N = \sum_{i=0}^{k} \frac{a_i}{q} \left(\frac{p}{q}\right)^i .$$

We will say that the word $a_k \cdots a_0$, computed from N from right to left, that is to say, *least significant digit first*, is a $\frac{p}{q}$-*expansion* of N. It is known that this representation is indeed unique. When $q = 1$, we recover the usual p-ary number system.

It is to be stressed that this definition *is not* the classical one —corresponding to β-expansions — for the numeration system in base $\frac{p}{q}$: the digits *are not* the integers smaller than $\frac{p}{q}$ but rather the integers *whose quotient by* q is smaller than $\frac{p}{q}$.

We recall some results from [1].

Proposition 13. *The set $L_{\frac{p}{q}}$ of the $\frac{p}{q}$-expansions of elements of $\mathbb{N}$ is* not *a rational language.*

Note even that two distinct subtrees of the trie of $L_{\frac{p}{q}}$ are never isomorphic.

Proposition 14. *The successor function on the set $L_{\frac{p}{q}}$ of the $\frac{p}{q}$-expansions of elements of $\mathbb{N}$ is realizable by a right sequential letter-to-letter finite transducer.*

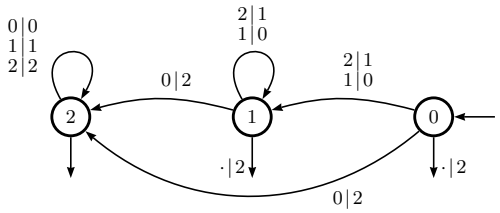


Fig. 4. The successor function for the $\frac{3}{2}$ number system

The number $\mathbf{v}_{L_{\frac{p}{q}}}(n)$ of elements of $L_{\frac{p}{q}}$ of length $\leq n$ is equal to the number G_n where the sequence $(G_n)_{n\in\mathbb{N}}$ is defined by

$$G_0 = 1 \qquad \text{and} \qquad \forall n \in \mathbb{N},\ G_{n+1} = \left\lceil \frac{p}{q}\, G_n \right\rceil .$$

Thus from Proposition 5 follows

Proposition 15. *The cost and the complexity of the successor function on $L_{\frac{p}{q}}$ are equal to*

$$\mathsf{Cost}(\mathrm{Succ}_{L_{\frac{p}{q}}}) = \mathsf{Comp}(\mathrm{Succ}_{L_{\frac{p}{q}}}) = \frac{\frac{p}{q}}{\frac{p}{q} - 1}.$$

Proof. Comes from the fact that $\lim_{n\to\infty} G_{n+1}/G_n = p/q$. ∎

Acknowledgments

Part of this work was initiated when the third author was visiting LIAFA in Paris thanks to a *ESF AuthoMathA* grant. The other authors acknowledge the ACI Interfaces des Mathématiques, Project Numération, contract 04 312.

References

1. S. Akiyama, Ch. Frougny, J. Sakarovitch, Powers of rationals modulo 1 and rational base number systems, to appear in *Israël J. Math.* Available at `www.liafa.jussieu.fr/~cf/publications`.
2. A. Bertrand-Mathis, Comment écrire les nombres entiers dans une base qui n'est pas entière, *Acta Math. Acad. Sci. Hungar.* **54** (1989), 237–241.
3. E. Barcucci, R. Pinzani, M. Poneti, Exhaustive generation of some regular languages by using numeration systems, Proc. of Words 2005 (Montréal), *Monographies du LaCIM* **36**, UQaM, (2005), 119–127.
4. V. Berthé, M. Rigo, Abstract Numeration Systems and Tilings, Proceedings of the 30th International Symposium: Mathematical Foundations of Computer Science (Gdańsk 2005), *Lect. Notes in Comput. Sci.* **3618** (2005), 131–143.
5. V. Berthé, M. Rigo, Odometers on regular languages, *Theory Comput. Syst.* **40** (2007), 1–31.
6. V. Berthé, A. Siegel, Tilings associated with beta-numeration and substitutions, *Integers* **5** (2005), no. 3, A2, 46 pages.
7. A. S. Fraenkel, Systems of numeration, *Amer. Math. Monthly* **92** (1985), 105–114.
8. Ch. Frougny, On-line odometers for two-sided symbolic dynamical systems, *Proceedings of DLT 2002, Lect. Notes in Comput. Sci.* **2450** (2002), 405–416.
9. Ch. Frougny, On the sequentiality of the successor function, *Inform. and Comput.* **139** (1997), 17–38.
10. Ch. Frougny, J. Sakarovitch, Synchronized relations of finite and infinite words, *Theoret. Computer Sci.* **18** (1993), 45–82.
11. Ch. Frougny, J. Sakarovitch, Synchronisation déterministe des automates à délai borné, *Theoret. Comput. Sci.* **182** (1998), 61–77.
12. Ch. Frougny, B. Solomyak, On the representation of integers in linear numeration systems, in Ergodic theory of Z_d actions (Warwick, 1993–1994), 345–368, *London Math. Soc. Lecture Note Ser.* **228**, Cambridge Univ. Press, Cambridge, 1996.
13. P. J. Grabner, P. Liardet, and R. F. Tichy, Odometers and systems of numeration, *Acta Arith.* **70** (1995), 103–123.
14. P. J. Grabner, M. Rigo, Distribution of additive functions with respect to numeration systems on regular languages, *Theory Comput. Syst.* **40** (2007), 205–223.
15. P. Lecomte and M. Rigo, Numeration systems on a regular language, *Theory Comput. Syst.* **34** (2001), 27–44.
16. M. Lothaire, *Algebraic combinatorics on words*, Encyclopedia of Mathematics and its Applications, no. 90, Cambridge University Press, Cambridge, 2002.
17. A. Rényi, Representations for real numbers and their ergodic properties, *Acta Math. Acad. Sci. Hungar.* **8** (1957) 477–493.
18. J. Sakarovitch, *Eléments de théorie des automates*, Vuibert (2003). English translation: *Elements of Automata Theory*, Cambridge University Press, to appear.
19. J. Shallit, Numeration systems, linear recurrences, and regular sets, *Inform. and Comput.*, **113** (1994), 331–347.
20. A. M. Vershik, A theorem on the Markov periodical approximation in ergodic theory, *J. Sov. Math.* **28** (1985), 667–674.