On the concrete complexity of the successor function (long abstract)

V. Berthé, Ch. Frougny[*], M. Rigo[†], J. Sakarovitch[‡]

June 3, 2012

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, in symbolic dynamics or in fractal geometry. They have also been studied from a combinatorial and topological point of view. 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 this talk 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 Succ_L 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 $n+1$.

Recall the following results. Let L be a rational language. The successor function Succ_L is realized by a (left) letter-to-letter finite transducer. It can also be realized by a *right* letter-to-letter finite transducer. Recently, P.-Y. Angrand and J. Sakarovitch have show the following result (for precise definitions, see the corresponding paper).

Theorem 1. *Let L be a rational language. The successor function* Succ_L *is piecewise right sequential.*

It is quite natural to consider two kinds of questions about the successor function. The first one 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

[*]LIAFA, CNRS UMR 7089, Université Paris-Diderot
[†]Université de Liège
[‡]LTCI, ENST/CNRS

given language). This leads to the notion of amortized (or 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.

1 Preliminaries

Let $(A, <)$ be a finite alphabet, totally ordered. The set of all words over A is denoted by A^*. The empty word is denoted by ε. The set of words of length $\leq n$ is denoted by $A^{\leq n}$. The length of a word x of A^* is denoted by $|x|$. If $x = uv$, u is a *prefix* (or a *left-factor*) of x, and v is a *suffix* (or a *right-factor*) of x.

The *radix order* (also called the *genealogical order* or the *short-lex 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 what follows, $L \subseteq A^*$ stands for a language ordered by the radix order. Without loss of generality, we can always assume that A is made of consecutive integers starting with 0 and naturally ordered, $A = \{0, 1, \ldots, r-1\}$. The set of largest words of each length for the radix ordering of L is denoted by $\mathsf{maxlg}(L)$. In particular, the results presented in this paper can also be expressed in terms of abstract numeration systems.

Let i be a non-negative integer; the $(i+1)$-th word of L in the radix order is denoted as $\langle i \rangle_L$. The *successor* of a word x in L is the unique word y of 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 $x = \langle i \rangle_L$ of L onto its successor $y = \langle i+1 \rangle_L$ in L. The *odometer* is an extension to infinite words or to bi-infinite words of the successor function.

A language L is said to be *prefix-closed* if every prefix of a word of L is in L. Moreover, L is said to be *right extendable* if, for all $w \in L$, there exists a non-empty word u such that $wu \in L$. A language L is *right essential* if it is prefix-closed and right extendable.

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

An *automaton over* A, $\mathcal{A} = (Q, A, E, I, T)$, is a directed graph labeled by elements of A. The set of vertices, traditionally called *states*, is denoted by Q, $I \subset Q$ 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 labeled *edges*. If $(p, a, q) \in E$, we write $p \xrightarrow{a} q$. The automaton is *finite* if Q is finite. The automaton $\mathcal{A}$ is *deterministic* if E is the graph of a (partial) function from $Q \times A$ into Q, and if there is a unique initial state. A subset L of A^* is said to be *recognizable by a finite automaton* or *rational* if there exists a finite automaton $\mathcal{A}$ such that L is equal to the set $|\mathcal{A}|$ of labels of paths starting in an initial state and ending in a terminal state. The set of rational languages over A is denoted as $\mathrm{Rat}(A^*)$.

Let L in $\mathrm{Rat}(A^*)$ having $\mathcal{M}$ as a minimal automaton. We suppose that $\mathcal{M}$ is trim, *i.e.*, accessible and co-accessible. It is said to be *primitive* if its adjacency matrix M is primitive, *i.e.*, there exists some k such that $M^k > 0$. If q is a state of $\mathcal{M}$, we denote by $\mathbf{u}_q(n)$ (resp. $\mathbf{v}_q(n)$) the number of words of length n (resp. at most n) accepted from state q. In particular, if q_0 stands for the initial state of $\mathcal{M}$, we write either $\mathbf{u}_{q_0}(n)$ or $\mathbf{u}_L(n)$ to stand for $\#(L \cap A^n)$. In the same way, we use both $\mathbf{v}_{q_0}(n)$ or $\mathbf{v}_L(n)$ denoting $\#(L \cap A^{\le n})$. It is well-known that all the sequences $(\mathbf{u}_q(n))_{n\ge 0}$ satisfy the same recurrence relation whose characteristic polynomial is the one of M. In particular, if $\gamma_1, \ldots, \gamma_t$ are the roots of this polynomial, then using a standard result about the general form of linear recurrence sequences, there exist polynomials $P_{q,1}, \ldots, P_{q,t}$ such that

$$\mathbf{u}_q(n) = P_{q,1}(n)\, \gamma_1^n + \cdots + P_{q,t}(n)\, \gamma_t^n. \tag{1}$$

2 Carry propagation

Let L be a language ordered by the radix order. Given two words x and y, we set

$$\Delta(x,y) = \begin{cases} \max(|x|,|y|) & \text{if } |x| \ne |y|, \\ \min\{|v| \mid \exists u, w, x = uv, y = uw\} & \text{if } |x| = |y|. \end{cases}$$

With this notation, we define the carry propagation when computing the successor as follows:

Definition 1. The *carry propagation* in the computation of $\mathrm{Succ}_L(\langle i \rangle_L)$ is $\Delta(\langle i \rangle_L, \langle i+1 \rangle_L)$. If the language L is clear from the context, we simply write $\Delta(x)$ as a shorthand and for $\Delta(x, \mathrm{Succ}_L(x))$, for all $x \in L$.

Definition 2. The (amortized) *carry propagation of* Succ_L is defined as the following limit if it exists

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

Note that the amortized carry propagation might be infinite (for instance, for a language having a polynomial growth, the simplest example being $L = a^*$). Furthermore, the limit might not exist even for a right essential language.

Proposition 2. *Let L be a right essential language. The carry propagation for computing the successor of all the words of L of length $n \ge 0$ is given by $\mathbf{v}_L(n)$.*

Theorem 3. *Let L be a right essential language. 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 $\gamma_L > 1$. Suppose moreover that the limit $\lim_{N\to\infty} \frac{1}{N}\sum_{i=0}^{N-1} \Delta(\langle i \rangle_L, \langle i+1 \rangle_L)$ exists. Then*

$$\mathsf{CP}(\mathrm{Succ}_L) = \frac{\gamma_L}{\gamma_L - 1}.$$

Lemma 1. *Let L be a rational right essential language. Suppose that the adjacency matrix M of the the trim minimal automaton $\mathcal{M}$ of L has a unique dominating eigenvalue $\gamma_L > 1$,* i.e., *for any other eigenvalue* $\gamma_2, \ldots, \gamma_t$*, we have* $|\gamma_j| < \gamma_L < 1$*. Then, for all states* q *of* $\mathcal{M}$*, there exist polynomials* $P_{q,1}, P_{q,2}, \ldots, P_{q,t}$ *such that*

$$\mathbf{u}_q(n) = P_{q,1}(n)\, \gamma_L^n + \sum_{j=2}^{t} P_{q,t}(n)\, \gamma_j^n, \tag{2}$$

$$P_{q_0,1} \neq 0 \text{ and } \deg P_{q_0,1} \geq \deg P_{q,1}, \text{ for all states } q, \tag{3}$$

$$\text{if } P_{q,1} \neq 0, \text{ then } \lim_{n\to\infty} \frac{\mathbf{v}_q(n)}{P_{q,1}(n)\, \gamma_L^n} = \frac{\gamma_L}{\gamma_L - 1}; \text{ otherwise } \lim_{n\to\infty} \frac{\mathbf{v}_q(n)}{\gamma_L^n} = 0. \tag{4}$$

In that case, the limit $\lim_{N\to\infty} \frac{1}{N} \sum_{i=0}^{N-1} \Delta(\langle i \rangle_L, \langle i+1 \rangle_L)$ *exists.*

Proposition 4. *Let* L *be a rational right essential language. Suppose that the adjacency matrix* M *of the the trim minimal automaton* $\mathcal{M}$ *of* L *has a unique dominating eigenvalue* $\gamma_L > 1$, i.e., *for any other eigenvalue* $\gamma_2, \ldots, \gamma_t$*, we have* $|\gamma_j| < \gamma_L < 1$*. Then, the limit* $\lim_{N\to\infty} \frac{1}{N} \sum_{i=0}^{N-1} \Delta(\langle i \rangle_L, \langle i+1 \rangle_L)$ *exists.*

3 Concrete complexity of the successor function

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.

Definition 3. 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

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

A finite transducer can be seen as a program, but it is not sequential in general.

Proposition 5. *If* L *is a rational language then* $\mathsf{Comp}(\mathrm{Succ}_L) \geq \mathsf{CP}(\mathrm{Succ}_L)$*.*

To compute precisely the complexity, we must take into account what is the needed information to take a decision of move or writing. If x is in L, we denote by $CP(x)$ the carry propagation and $Comp(x)$ the total number of operations needed to compute $\mathrm{Succ}_L(x)$. Note that we do not consider the prefix of x which is invariant under the successor function. The *surcharge* $SC(x)$ is the difference $SC(x) = Comp(x) - CP(x)$.

The (amortized) *surcharge* for computing Succ_L is

$$\mathsf{SC}(\mathrm{Succ}_L) = \lim_{N\to\infty} \frac{1}{N} \sum_{i=0}^{N-1} SC(\langle i \rangle_L) = \mathsf{Comp}(\mathrm{Succ}_L) - \mathsf{CP}(\mathrm{Succ}_L).$$

When the successor function is realized by a right sequential letter-to-letter finite transducer, each move in the transducer is determined only by the input letter and produces an output letter, so there is no surcharge.

Proposition 6. *If* Succ_L *is realized by a right sequential letter-to-letter finite transducer, then*

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

Note that this result does not hold anymore when the successor function is *left* sequential, but not *right* sequential.

4 Results for beta-numeration

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

We now recall some definitions and results on the so-called *beta-numeration*. 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 $z_i \in A_\beta = \{0, \ldots, \lceil \beta \rceil - 1\}$ 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. It is denoted by $d_\beta(z) = (z_i)_{i\geq 1}$. When the expansion ends in infinitely many 0's, it is said *finite*, and the 0's are omitted.

Let $d_\beta(1) = (t_n)_{n\geq 1}$ be the β-expansion of 1. If $d_\beta(1)$ is finite, of the form $d_\beta(1) = t_1 \cdots t_m$, $t_m \neq 0$, let $d^*_\beta(1) = (t_1 \cdots t_{m-1}(t_m - 1))^\omega$ be the *quasi-greedy expansion* of 1. If the β-expansion of 1 is infinite, set $d^*_\beta(1) = d_\beta(1)$.

Definition 4. Let $\beta > 1$ be a real number.
If the β-expansion of 1 is finite, of the form $d_\beta(1) = t_1 \cdots t_m$, then set $v_0 = 1$, $v_n = t_1 v_{n-1} + \cdots + t_n v_0 + 1$ for $1 \leq n \leq m-1$, and $v_n = t_1 v_{n-1} + \cdots + t_m v_{n-m}$ for $n \geq m$.
If the β-expansion of 1 is infinite, $d_\beta(1) = (t_n)_{n\geq 1}$, then set $v_0 = 1$, and $v_n = t_1 v_{n-1} + \cdots + t_n v_0 + 1$ for $n \geq 1$.
The sequence $V_\beta = (v_n)_{n\geq 0}$ of integers with the alphabet A_β forms the *canonical numeration system associated with β.*

It is well known that $\lim_{n\to\infty} v_{n+1}/v_n = \beta$. Then from Theorem 3 follows.

Theorem 7. *If $\beta > 1$, the carry propagation of the successor function for the canonical numeration system V_β associated with β is*

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

We will explain for the cases of simple and non-simple Parry number, how to obtain a specific transducer to compute the successor function for beta-numeration. In particular, **an important part of this talk is to show that our construction permits to compute and estimate precisely the (amortized) surcharge and thus, the complexity** of the successor function for the canonical numeration system V_β associated with β.

References

[1] S. Akiyama, Ch. Frougny, J. Sakarovitch, Powers of rationals modulo 1 and rational base number systems, *Israël J. Math.* **168** (2008) 53–91.

[2] P.-Y. Angrand, J. Sakarovitch, Radix enumeration of rational languages, *RAIRO - Theoret. Informatics and Appl.* **44** (2010) 19–36.

[3] 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.

[4] E. Barcucci, R. Pinzani, M. Poneti, Exhaustive generation of some regular languages by using numeration systems, Proc. Words 2005 (Montréal), *Monographies du LaCIM* **36**, UQaM, (2005) 119–127.

[5] V. Berthé, Ch. Frougny, M. Rigo, J. Sakarovitch, On the cost and complexity of the successor function, in: P. Arnoux, N. Bédaride, and J. Cassaigne (Eds.), Proc. Words 2007, Tech. Rep., Institut de mathématiques de Luminy (Marseille) (2007) 43–56.

[6] V. Berthé, M. Rigo, Odometers on regular languages, *Theory Comput. Syst.* **40** (2007) 1–31.

[7] Ch. Frougny, On the sequentiality of the successor function, *Inform. and Comput.* **139** (1997) 17–38.

[8] Ch. Frougny, On-line odometers for two-sided symbolic dynamical systems, Proc. DLT 2002, *Lect. Notes in Comput. Sci.* **2450** (2002) 405–416.

[9] Ch. Frougny, J. Sakarovitch, Synchronized relations of finite and infinite words, *Theoret. Computer Sci.* **18** (1993) 45–82.

[10] Ch. Frougny, J. Sakarovitch, Number representation and finite automata, Chapter 2 in *Combinatorics, Automata and Number Theory*, V. Berthé, M. Rigo (Eds), Encyclopedia of Mathematics and its Applications 135, Cambridge University Press (2010).

[11] 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.

[12] P. J. Grabner, P. Liardet, and R. F. Tichy, Odometers and systems of numeration, *Acta Arith.* **70** (1995) 103–123.

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

[14] P. Lecomte, M. Rigo, Abstract numeration systems, Chapter 3 in *Combinatorics, Automata and Number Theory*, V. Berthé, M. Rigo (Eds), Encyclopedia of Mathematics and its Applications 135, Cambridge University Press (2010).

[15] M. Lothaire, *Algebraic combinatorics on words*, Encyclopedia of Mathematics and its Applications, no. 90, Cambridge University Press, Cambridge, 2002.

[16] W. Parry, On the β-expansions of real numbers, *Acta Math. Acad. Sci. Hung.* **11** (1960) 401–416.

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