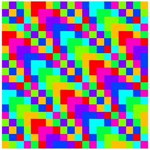
# On the concrete complexity of the successor function

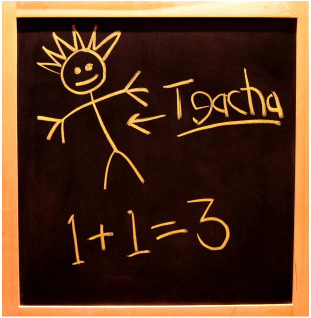
M. Rigo
joint work with V. Berthé, Ch. Frougny, J. Sakarovitch

http://www.discmath.ulg.ac.be/
http://orbi.ulg.ac.be/handle/2268/130094

Université de Liège



Let's start with a quite naïve question.
*Just add one*.




Given the representation of the integer $n$,
*compute* the representation of $n+1$.

# General framework

### Definition

Let $L$ be a language over a finite (totally) ordered alphabet $(A, <)$. We order the words in $L$ by increasing genealogical (or radix) order:

$$w_0 \prec w_1 \prec w_2 \prec \cdots \prec w_n \prec w_{n+1} \prec \cdots$$

The *successor function* on $L$ is

$$\mathrm{Succ}_L : L \to L, w_n \mapsto w_{n+1}.$$

$$\mathrm{Succ}_L(x) = y \Leftrightarrow (x \prec y) \wedge (\forall z \in L)\,((x \prec z) \Rightarrow ((y = z) \vee (y \prec z))).$$

## Connection with abstract numeration systems

An *abstract numeration system* is a triple $\mathcal{S} = (L, A, <)$ where $L$ is an infinite regular language over the ordered alphabet $(A, <)$ [P. Lecomte, M.R., 2001].

## Example (classical)

Take $L = 1\{0, 01\}^* \cup \{\varepsilon\}$.

$$\begin{array}{rcc} \varepsilon & 1000 & \vdots \\ 1 & 1001 & 101001010 \\ 10 & 1010 & 101010000 \\ 100 & \vdots & 101010001 \\ 101 & & \end{array}$$

We get back to the usual Zeckendorf numeration system based on the Fibonacci sequence.

**Theorem [Ch. Frougny (1997)]**

Let L be a regular language.
The successor function $\mathrm{Succ}_L$ is realized by a letter-to-letter transducer.

**Theorem [P.-Y. Angrand, J. Sakarovitch (2010)]**

Let L be a regular language.
The successor function $\mathrm{Succ}_L$ is piecewise right sequential.

sequential right transducer = co-sequential transducer

- transducer with deterministic underlying input automaton,
- reads and writes words from the right to the left.
- A function which is a finite union of (co-)sequential functions with pairwise disjoint domains is called a piecewise (co-)sequential function.

**Theorem [M.-P. Schützenberger (1975)]**

One can decide whether or not a transducer is functional (i.e., is realizing a rational function).

**Theorem [Ch. Choffrut (1977)]**

One can decide whether or not a functional transducer is realizing a sequential function.

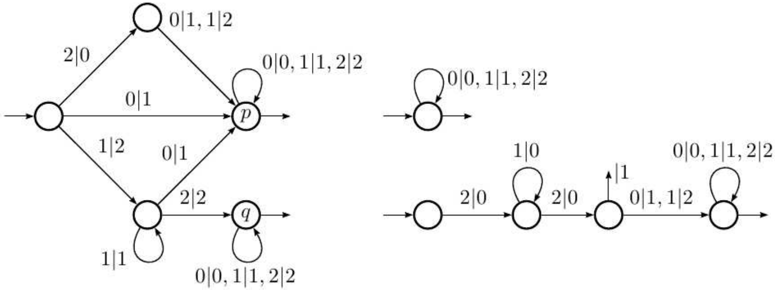
**Theorem [P.-Y. Angrand, J. Sakarovitch (2010)]**

A rational function is piecewise right sequential if and only if it can be realized by a *cascade* of sequential right transducers of height $1$ or $2$.

## Square of the Golden ratio

$X^2 - 3X + 1$, $\beta = \frac{3+\sqrt{5}}{2}$, $d_\beta(1) = 21^\omega$, $(U_n)_{n\geq 0} = 1, 3, 8, 21, \ldots$

$\text{rep}(\mathbb{N}) = \{\varepsilon, 1, 2, 10, 11, 12, 20, 21, \ldots\}$ forbidden factors: $21^*2$;
here $\text{Succ}_L$ is neither (left) sequential nor right sequential.




P.-Y. Angrand [Ph.D. thesis (2012), p. 128]

$$2102 \longrightarrow 2110(p) \qquad 2111 \longrightarrow 2112(q) \longrightarrow 10000$$

One of the motivations stems from combinatorial, metrical, topological, dynamics, sequential properties of odometers.

$\cdots 010010100\textcolor{blue}{01010}$


$\cdots 010010100\textcolor{red}{10000}$


- A. M. Vershik, A theorem on the Markov periodical approximation in ergodic theory, *J. Sov. Math.* **28** (1985) 667–674.
- P. G. Grabner, P. Liardet, R. F. Tichy, Odometers and systems of numeration, *Acta Arith.* **70** (1995) 103–123.
- G. Barat, T. Downarowicz, P. Liardet, Dynamiques associées à une échelle de numération, *Acta Arith.* **103** (2002), 41–78.
- Ch. Frougny,On-line odometers for two-sided symbolic dynamical systems, Proc. *Lect. Notes in Comput. Sci.* **2450** (2002) 405–416.
- V. Berthé, M. Rigo, Odometers on regular languages, *Theory Comput. Syst.* **40** (2007) 1–31.

We can compute, but how do we compute?

## ORIGINAL PROBLEM (WORDS 2005)

E. Barcucci, R. Pinzani, M. Poneti, *Exhaustive generation of some regular languages by using numeration systems*.
For numeration systems built on some linear recurrent sequences of order 2, the "amortized cost" for computing $\mathrm{rep}(n+1)$ from $\mathrm{rep}(n)$ is bounded by a constant (CAT).

- Estimate the length of the carry propagation when applying the successor map on the first $n$ words in $L$.

  $\longrightarrow$ *amortized (or average) carry propagation*.

- A computational issue: estimate the number of operations (in 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.

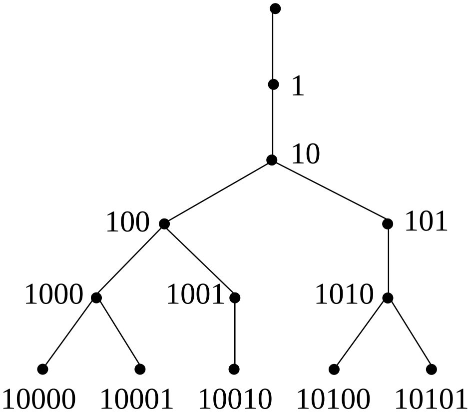
  $\longrightarrow$ *(amortized) complexity*, i.e., the average amount of computations required to obtain the successor of an element.

# Framework

We assume that $L$ is *right essential*:

- $L$ is *prefix-closed*;
- $L$ is *right extendable*, $\forall w \in L$, $\exists u \neq \varepsilon : wu \in L$.

Example: Trie for the Zeckendorf system



# Part I — carry propagation

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

The *carry propagation* in the computation of $\mathrm{Succ}_L(\mathrm{rep}(i))$ is

$$\Delta(\mathrm{rep}(i), \mathrm{rep}(i+1)).$$

## Example (Zeckendorf system)

| | $\Delta$ |
|---|---|
| $\varepsilon$ | $-$ |
| 1 | 1 |
| 10 | 2 |
| 100 | 3 |
| 101 | 1 |

| | $\Delta$ |
|---|---|
| 1000 | 4 |
| 1001 | 1 |
| 1010 | 2 |
| 10000 | 5 |
| 10001 | 1 |

| | $\Delta$ |
|---|---|
| 10010 | 2 |
| 10100 | 3 |
| 10101 | 1 |
| 100000 | 6 |
| 100001 | 1 |

## Definition

The (amortized) *carry propagation of* $\mathrm{Succ}_L$ is defined as the following limit if it exists

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

- The limit might be infinite (e.g., language with polynomial growth, simple case: $a^*$).
- The limit might not exist even for a right essential language.

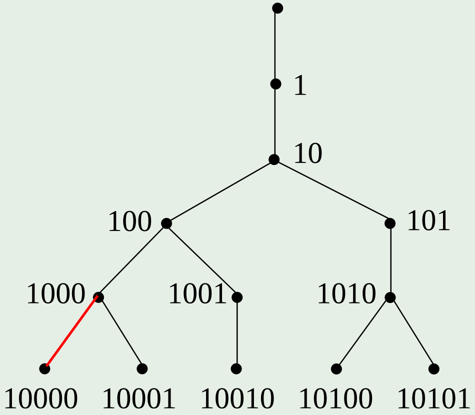
$$\boxed{\mathbf{u}_L(n) = \#(L \cap A^n) \qquad \mathbf{v}_L(n) = \#(L \cap A^{\leq n})}$$

**Proposition**

Let $L$ be a right essential language. The carry propagation for computing $\mathrm{Succ}_L$ for all words of $L$ of length $n \geq 0$ is $\mathbf{v}_L(n)$.

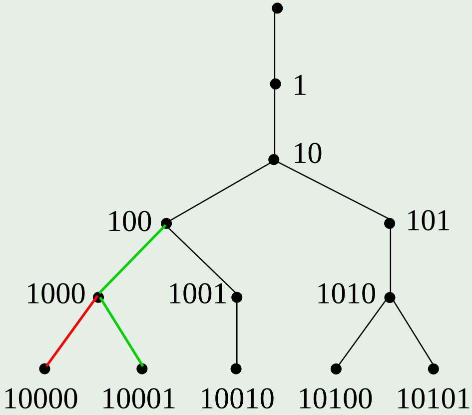
Let's have a look at the trie, $\mathbf{v}_L(5) = 13$.

## Fibonacci words of length 5



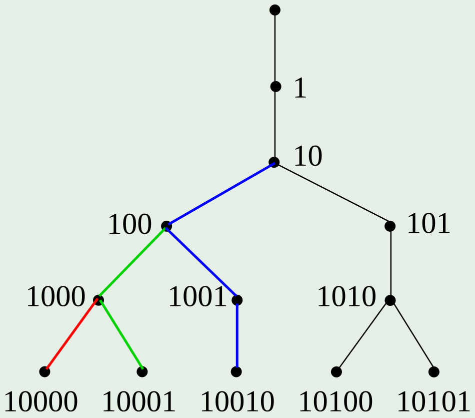
Let's have a look at the trie, $\mathbf{v}_L(5) = 13$.

## Fibonacci words of length 5



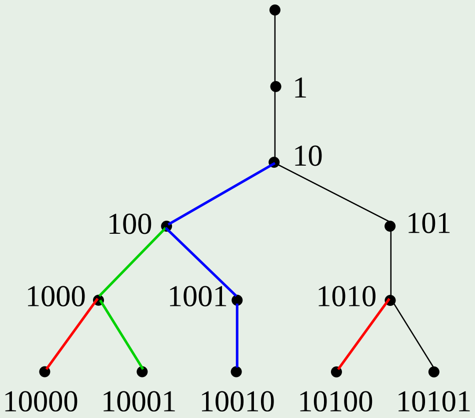
Let's have a look at the trie, $\mathbf{v}_L(5) = 13$.

## Fibonacci words of length 5



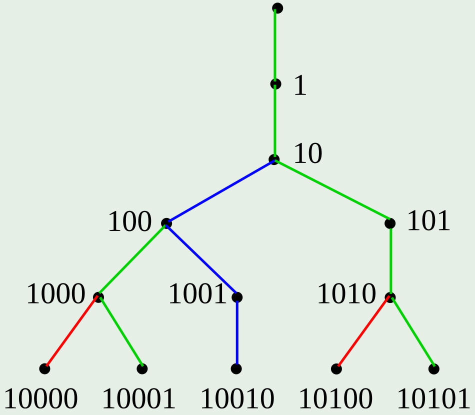
Let's have a look at the trie, $\mathbf{v}_L(5) = 13$.

## Fibonacci words of length 5



Let's have a look at the trie, $\mathbf{v}_L(5) = 13$.

### FIBONACCI WORDS OF LENGTH 5



## Theorem

Let $L$ be a right essential language.
Suppose that

$$\lim_{n\to\infty} \mathbf{u}_L(n+1)/\mathbf{u}_L(n), \text{ or } \lim_{n\to\infty} \mathbf{v}_L(n+1)/\mathbf{v}_L(n),$$

exists and equals some $\gamma_L > 1$.
Suppose that $\lim_{N\to\infty} \frac{1}{N}\sum_{i=0}^{N-1} \Delta(\mathrm{rep}(i), \mathrm{rep}(i+1))$ exists.
Then

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

Can be applied to many classical numeration systems, for instance:

- trim minimal automaton $\mathcal{M}$ of $L$ with
  a unique dominating eigenvalue $\gamma_L > 1$,
- primitiveness of the trim minimal automaton,
- beta-numeration.

Suppose that $P$ is a program (i.e., a Turing machine) which, for every $i \geq 0$, computes $\mathrm{Succ}_L(\mathrm{rep}(i))$ in $\mathsf{Op}(P, \mathrm{rep}(i))$ operations. The (amortized) *complexity of* $P$ is

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

The (amortized) *complexity of* $\mathrm{Succ}_L$ is

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

## REMARK

If $L$ is a regular language, then $\mathsf{Comp}(\mathrm{Succ}_L) \geq \mathsf{CP}(\mathrm{Succ}_L)$.

*At least the number of head moves corresponding to carry propagation.*

# Existence of a surcharge...

Consider again the "square of the Golden ratio": forbid $21^*2$

$$\mathrm{Succ}(21111111111) = 100000000000 \quad CP = Comp = 12$$

$$\mathrm{Succ}(11111111111) = 11111111112 \quad CP = 1 \;\; Comp \geq 11$$

$\longrightarrow$ *The needed information to take a decision of move or writing*.
Carry propagation is not the only one that matters!

- $CP(x)$ is the carry propagation,
- $Comp(x)$ is the total number of operations needed to compute $\mathrm{Succ}_L(x)$,
- The *surcharge* is the difference $SC(x) = Comp(x) - CP(x)$.

The (amortized) *surcharge* for computing $\mathrm{Succ}_L$ is

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

### Proposition

If $\mathrm{Succ}_L$ is realized by a right sequential letter-to-letter finite transducer, then

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

each move in the transducer is determined only by the input letter and produces an output letter, so there is no surcharge.

# Case of beta-numeration

$\beta > 1$ real number
Let $d_\beta(1) = (t_n)_{n\geq 1}$ be the (greedy) $\beta$-expansion of $1$.

$$1 = \sum_{i=1}^{\infty} t_i\, \beta^{-i}.$$

- If the $\beta$-expansion of $1$ is *finite*, $d_\beta(1) = t_1 \cdots t_m$, 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 $\beta$-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 $V_\beta = (v_n)_{n\geq 0}$ with $A_\beta = \{0, \ldots, \lceil\beta\rceil - 1\}$ is the *canonical numeration system associated with* $\beta$.
Note that $\lim_{n\to\infty} v_{n+1}/v_n = \beta$.

- If $d_\beta(1)$ is finite, $\beta$ is a *simple Parry number*, or
- if $d_\beta(1)$ is ult. periodic, $\beta$ is a (non-simple) *Parry number*

then

- $V_\beta = (v_n)_{n\geq 0}$ is a linear recurrent sequence;
- the language $L(V)$ of the greedy expansions of all the non-negative integers is regular.

**Proposition [Ch. Frougny (1997)]**

Let $V$ be a linear recurrent sequence with dominant root $\beta$ such that $L(V)$ is regular. Then $\mathrm{Succ}_{L(V)}$ is right sequential IFF
(i) the $\beta$-expansion of $1$ is finite, of the form $d_\beta(1) = t_1 \cdots t_m$,
(ii) $V$ is defined by $v_n = t_1 v_{n-1} + \cdots + t_m v_{n-m}$ for $n \geq n_0 \geq m$ and $1 = v_0 < v_1 < \cdots < v_{n_0-1}$.

$\mathrm{Succ}_{L(V)}$ is right sequential, but cannot be realized by a *letter-to-letter* right sequential transducer whenever $\beta \notin \mathbb{N}$.

# Sketch

In the case of a simple Parry number, we can (algorithmically) build a *specific* right sequential transducer that computes $\mathrm{Succ}_{L(V)}$ in such a way that we can derive a formula of the kind

$$\mathsf{SC}(\mathrm{Succ}_{L_\beta}) = \lim_{n\to\infty} \frac{1}{v_n} \sum_{e\in J} \sum_{i=0}^{n-1} \mathcal{W}_i(src(e))\, \delta(src(e))\, \mathcal{V}_{n-i-1}(trg(e)).$$

where the quantities $\mathcal{W}_\ell$ and $\mathcal{V}_\ell$ refer to the number of particular paths of length $\ell$ in the transducer and $\delta(e)$ is a known weight associated with particular edges in the construction.

Take the Pisot number $\beta > 1$ being dominating root of the polynomial $X^4 - 3X^3 - 3X^2 - 2X - 2$; $d^*_\beta(1) = (3321)^\omega$. $M = \{\varepsilon, 3, 33, 332, 3321, 33213, 332133, \dots\}$ is the set of finite prefixes of $d^*_\beta(1) =$ maximal representations for each length.

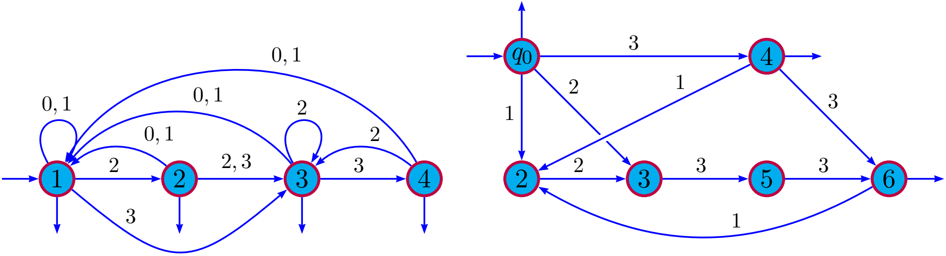
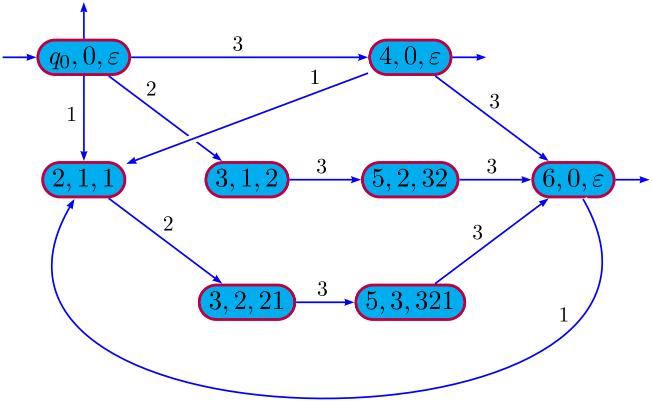



FIGURE: Right automata $\mathcal{L}$ and $\mathcal{M}$.

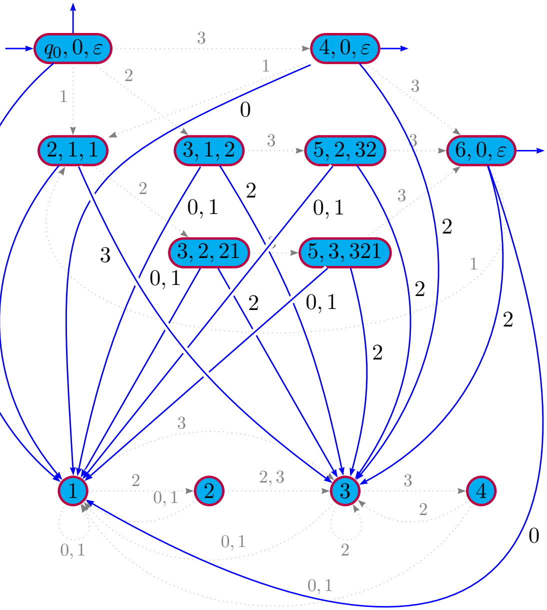
Strategy : *determine when the right-factor just being read is no more a maximal one.*

$\widehat{\mathcal{M}}$: in-splitting of $\mathcal{M}$, $(p, \delta(p), \lambda(p))$ = (state,valuation,label)
$\mathrm{Succ}_L(\cdots 02) = 03$, $\mathrm{Succ}_L(\cdots 021) = 022$ but both reach state 3




Valuations give some information on the length of the right-factor read before detecting that a word does not belong to $M$.

The "final" transducer (needs outputs, edges of $J$ in blue)



# Outputs on the edges

*The right-factor processed so far seems maximal*:

(A1) in $\widehat{\mathcal{M}}$, for each edge $p \xrightarrow{a} q$ with $p \neq s, q \neq s$ where $q$ is a terminal state, the corresponding edge in $\mathcal{T}$ is $p \xrightarrow{a|0^{\delta(p)+1}} q$;

(A2) in $\widehat{\mathcal{M}}$, for each edge $p \xrightarrow{a} q$ where $q$ is not a terminal state, the corresponding edge in $\mathcal{T}$ is $p \xrightarrow{a|\varepsilon} q$.

*We have just discovered that the right-factor is no more maximal*:

(A3) for each edge $p \xrightarrow{a} t$ in $J$ such that $p$ is a terminal state of $\widehat{\mathcal{M}}$, there is an edge $p \xrightarrow{a|a+1} t$ in $\mathcal{T}$;

(A4) for each edge $p \xrightarrow{a} t$ in $J$ such that $p$ is not a terminal state of $\widehat{\mathcal{M}}$, there is an edge $p \xrightarrow{a|a\,\mathrm{Succ}(\lambda(p))} t$ in $\mathcal{T}$.

*Nothing has to be done any more*:

(A5) for each edge $r \xrightarrow{a} t \in \mathcal{L}$ the output is just the copy of the input, namely $r \xrightarrow{a|a} t \in \mathcal{T}$.

For non-simple Parry number, we have developed a similar strategy with similar results about the amortized surcharge.