Addition in Dumont-Thomas Numeration Systems
in Theory and Practice

Journées Montoises 2024, Nice, France

O. Carton, JM. Couvreur, M. Delacourt, and **N. Ollinger**

IRIF, Université Paris Cité, CNRS et LIFO, Université d'Orléans

nicolas.ollinger@univ-orleans.fr

We want to compute on a word

$$aabaababcaabaababcaabab\cdots$$

We want to compute* on a word

$$aabaababcaabaababcaabab\cdots$$

* Compute **first-order properties** using the **Büchi-Bruyère** framework. In practice, with tools like **Walnut**.

We want to compute on a **word***

$$aabaababcaabaababcaabab\cdots$$

* **fixpoint** $\varphi^\omega(a)$ of $\varphi : a \to aab$, $b \to abc$, $c \to a$

$$\varphi : \begin{aligned} a &\to aab \\ b &\to abc \\ c &\to a \end{aligned}$$

Pick **your** favorite fixpoint.

We want to compute on a word

$$aabaababcaabaababcaabab\cdots$$

fixpoint $\varphi^{\omega}(a)$ of φ : $a \to aab$

$$b \to abc$$

$$c \to a$$

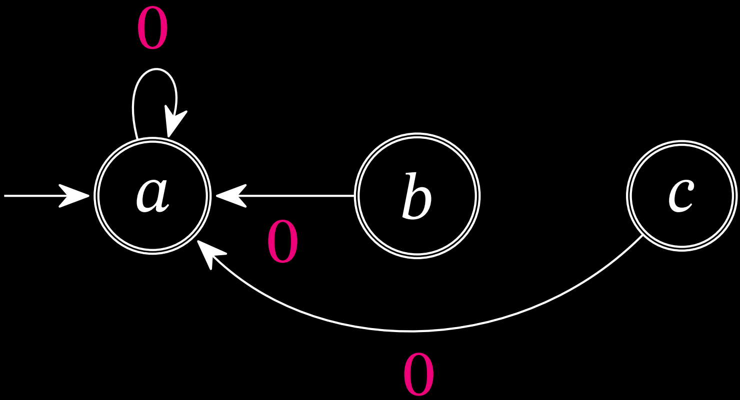
We need a **numeration system**!

Dumont-Thomas numeration system

$$\begin{aligned} \varphi : a &\to aab \\ b &\to abc \\ c &\to a \end{aligned}$$

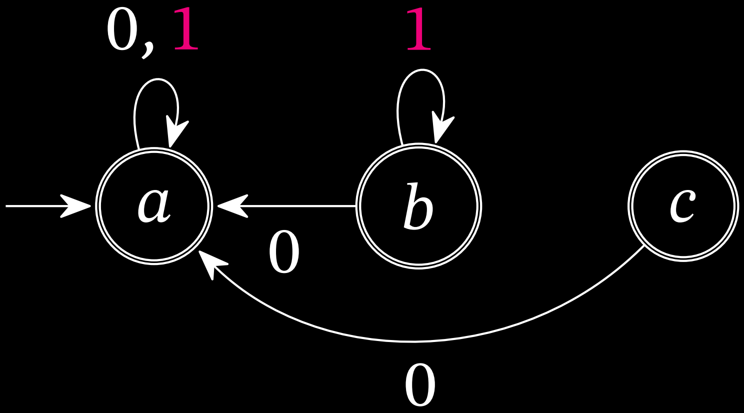
Dumont-Thomas numeration system

$$\begin{aligned} & \qquad\quad\ \ {\scriptstyle 0\ 1\ 2} \\ \varphi : a &\to aab \\ b &\to abc \\ c &\to a \end{aligned}$$



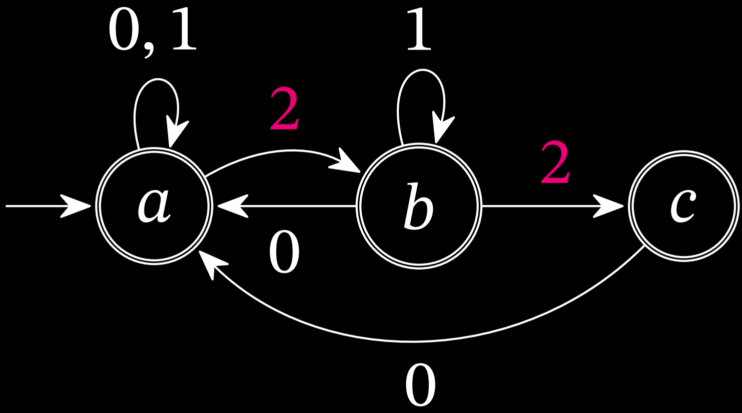
Dumont-Thomas numeration system

$$\begin{aligned} \varphi : a &\to \overset{0}{a}\overset{1}{a}\overset{2}{b} \\ b &\to abc \\ c &\to a \end{aligned}$$



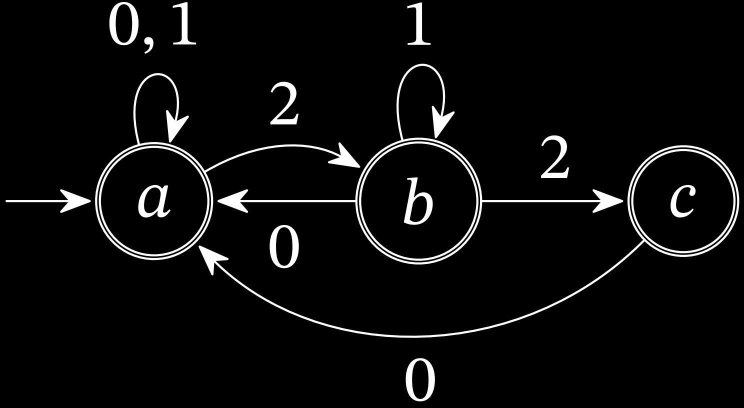
Dumont-Thomas numeration system

$$\varphi : \begin{aligned} a &\to \overset{0}{a}\overset{1}{a}\overset{2}{b} \\ b &\to abc \\ c &\to a \end{aligned}$$



Dumont-Thomas numeration system

$$\varphi : \begin{array}{l} a \to a\overset{0}{}a\overset{1}{}\overset{2}{b} \\ b \to abc \\ c \to a \end{array}$$



Dumont-Thomas numeration system

$$\varphi : \begin{array}{l} a \to \overset{0}{a}\overset{1}{a}\overset{2}{b} \\ b \to abc \\ c \to a \end{array}$$

As a DFA this automaton gives an **Abstract Numeration System** for radix order.

As a DFAO the same automaton gives an **automatic representation** for the fixpoint.

We still need a **regular addition** to compute.

ANS and rational series

Theorem (Lecomte-Rigo) The **valuation series** of an **abstract numeration system** (ANS) is **$\mathbb{N}$-rational**.

$$\nu_{\mathcal{N}} : u \mapsto \begin{cases} \mathrm{val}_{\mathcal{N}}(u) & \text{if defined} \\ 0 & \text{if not} \end{cases}$$

Can we use this to compute the addition?

First we need to define **synchronized addition** on series.

Notation The canonical **isomorphism** between $(\Sigma \times \Gamma)^*$ and $\bigcup_{n \geqslant 0} (\Sigma^n \times \Gamma^n)$ is denoted as $\langle ., . \rangle$ for every alphabets Σ, Γ:

$$\langle u, v \rangle = (u_1, v_1) \cdots (u_m, v_m) \quad \forall u, v \in \Sigma^m \times \Gamma^m$$

Synchronized addition

Definition The **synchronized addition** $f \oplus g$ between two series $f : \Sigma^* \to \mathbb{K}$ and $g : \Gamma^* \to \mathbb{K}$ is defined as

$$(f \oplus g)(\langle u, v \rangle) = f(u) + g(v) \quad \forall (u, v) \in \Sigma^m \times \Gamma^m$$

The **support** $\mathrm{supp}(f)$ of a series f is the language $\Sigma^* \setminus f^{-1}(0)$.

Proposition An **ANS** $\mathcal{N}$ has **regular addition** if and only if $\mathrm{supp}(\nu_\mathcal{N} \oplus \nu_\mathcal{N} \oplus -\nu_\mathcal{N})$ is **regular**.

Unfortunately, $\nu_\mathcal{N} \oplus \nu_\mathcal{N} \oplus -\nu_\mathcal{N}$ is a **rational $\mathbb{Z}$-series** and testing if the support of a $\mathbb{Z}$-series is regular is generally **undecidable**...

Positional Numeration Systems

Here, the substitution $\varphi : a \to aab,\ b \to abc,\ c \to a$ is **Pisot**.

$$M_\varphi = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix} \qquad P_\varphi(X) = X^3 - 3X^2 + X - 1$$

We could use the **Bertrand** positional NS associated to its **Pisot root**.

$$\psi : a \to aab,\ b \to ac,\ c \to aac \qquad U = (1, 3, 8, \cdots)$$

From **Bruyère-Hansel** and **Frougny-Solomyak**, we know that this PNS has **regular addition** and $\varphi^\omega(a)$ is **U-automatic**.

In this talk

We introduce **Sequence Automata** as a generic way to manipulate similar constructions of $\mathbb{Z}$-series related to **linear recurrence sequences**.

We lift the **Bruyère-Hansel** and **Frougny-Solomyak** proofs to obtain **regular support** for the associated series under **Pisot** conditions.

It comes with an implementation and **practical tools** for Walnut users.

```
[1]: %DT nice "a->aab, b->abc, c->a"

Studying substitution a->aab, b->abc, c->a
aabaababcaabaababcaababcaaabaababcaabaababcaababcaaabaababcaababcaaaba...
Substitution polynomial: X^3-3X^2+X-1
Minimized substitution a->aab, b->abc, c->a
Substitution polynomial: X^3-3X^2+X-1
Addition polynomial: X^3-3X^2+X-1 (n=3)
θ=2.7692923542386314
Substitution automaton: 3 states, 3 final states, 7 transitions.
>>> Writing /Users/nopid/soft/Walnut/Word Automata Library/Nice.txt in format Walnut

>>> Writing /Users/nopid/soft/Walnut/Word Automata Library/NiceParent.txt in format Walnut
```

Dumont-Thomas prefix factoring

Dumont and Thomas proved that **every prefix** p of $\varphi^\omega(a)$ can be represented using a **unique sequence** $(p_i, a_i)_{i=0}^{k} \in (\Sigma^* \times \Sigma)^*$ as

$$p = \prod_{i=0}^{k} \varphi^{k-i}(p_i) \quad \text{where } p_i a_i \text{ is a prefix of } \varphi(a_{i-1}) \quad \forall i \leqslant k \quad (\text{with } a_{-1} = a)$$

Thus every natural number can be decomposed as a sum of elements of **prefix length sequences**:

$$|p| = \sum_{i=0}^{k} \left|\varphi^{k-i}(p_i)\right|$$

Linear Recurrence Sequences

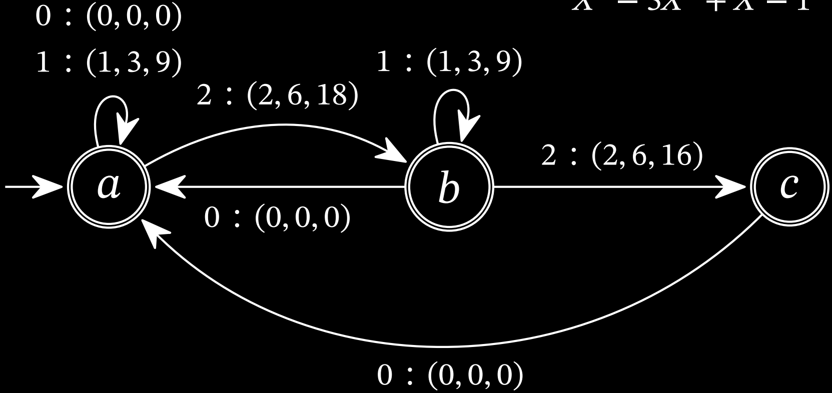
$$\varphi : \begin{aligned} a &\to aab \\ b &\to abc \\ c &\to a \end{aligned} \qquad M_\varphi = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

By **Cayley-Hamilton theorem**, all the sequences $(|\varphi^n(u)|)$ are **linear recurrence sequences** (LRS) satisfying the recurrence relation given by the **monic characteristic polynomial** of M_φ, here $X^3 - 3X^2 + X - 1$.

$$\begin{aligned} |\varphi^n(a)| &= 1,\quad 3,\quad 9,\quad 25,\quad \ldots \\ |\varphi^n(b)| &= 1,\quad 3,\quad 7,\quad 19,\quad \ldots \\ |\varphi^n(c)| &= 1,\quad 1,\quad 3,\quad 9,\quad \ldots \end{aligned}$$

Prefix length sequences are **linear combinations** of these sequences: they satisfy the same recurrence relation.

$X^3 - 3X^2 + X - 1$



Sequence Automaton

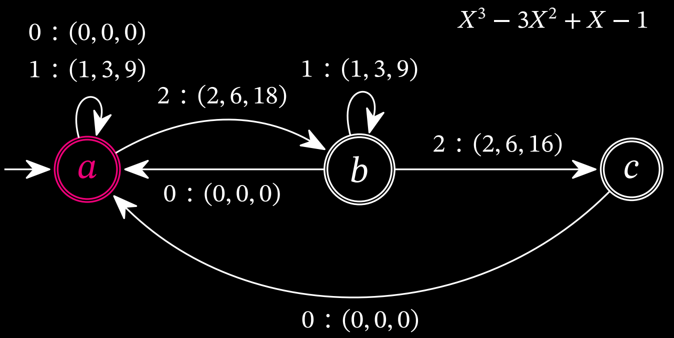
Sequence Automata

Definition A **sequence automaton** is a **partial DFA** $(Q, \Sigma, \delta, q_0, F)$ equipped with a **partial vector map** $\pi : Q \times \Sigma \to \mathbb{Z}^{\mathbb{N}}$ sharing a common domain with the **partial transition map** $\delta : Q \times \Sigma \to Q$.

Let (0) denote the zero sequence and σ denote the **shift map** on sequences: $\sigma(u)[n] = u[n+1]$ for all $u \in \mathbb{Z}^{\mathbb{N}}$ and $n \geqslant 0$.

The **transition map** and **vector map** are inductively extended from symbols to words as follows, for all $q \in Q$, $u \in \Sigma^*$ and $a \in \Sigma$:

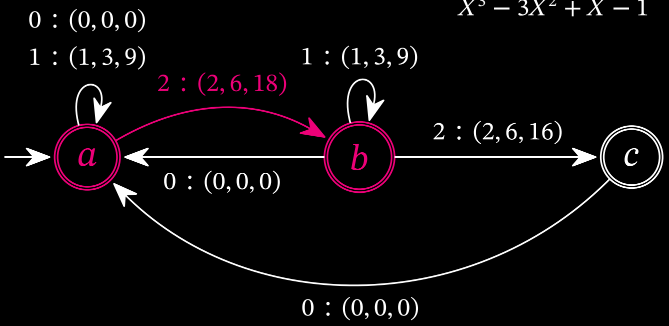
$$\delta(q, \varepsilon) = q \qquad \pi(q, \varepsilon) = (0)$$
$$\delta(q, ua) = \delta(\delta(q, u), a) \qquad \pi(q, ua) = \sigma\pi(q, u) + \pi(\delta(q, u), a)$$



$$(0, 0, 0)$$

$$\pi(a, {}_{\blacktriangle}2102) = \ldots$$

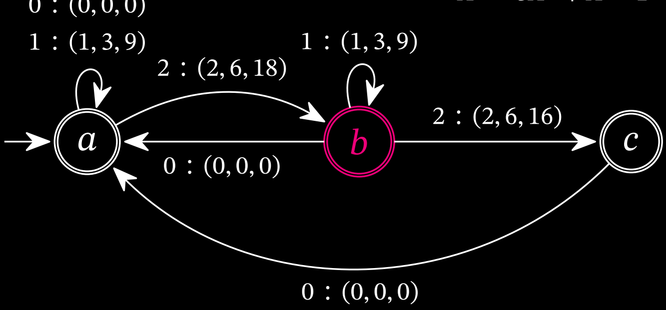
$X^3 - 3X^2 + X - 1$



$$\sigma(0,0,0) + (2,6,18) = (0+2, 0+6, 0+18)$$

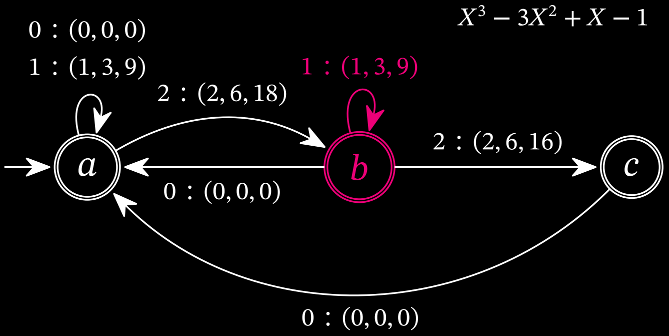
$$\pi(a, 2102) = \ldots$$

$X^3 - 3X^2 + X - 1$



$$(2, 6, 18)$$

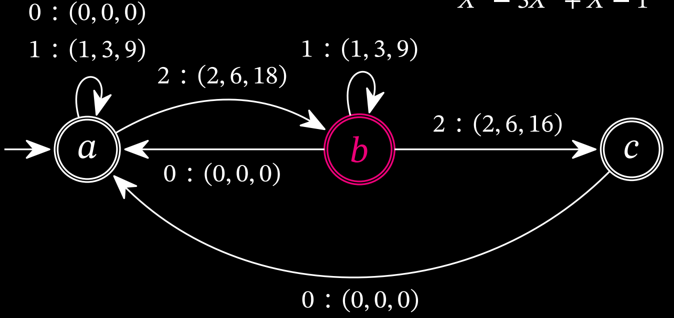
$$\pi(a, 2_{\blacktriangle}102) = \ldots$$



$$\sigma(2,6,18) + (1,3,9) = (6+1, 18+3, 50+9)$$

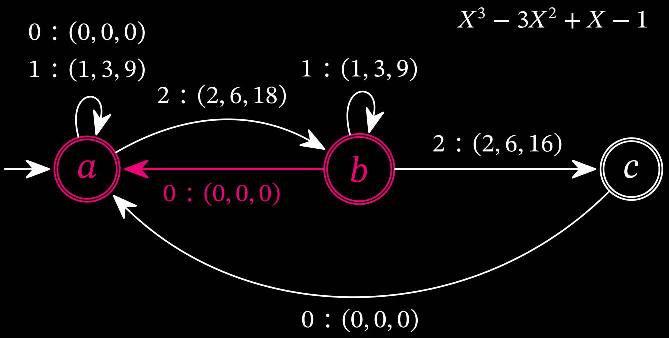
$$\pi(a, 2102) = \ldots$$

$X^3 - 3X^2 + X - 1$



$$(7, 21, 59)$$

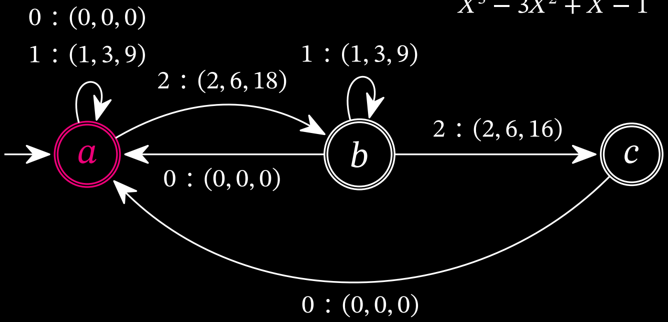
$$\pi(a, 21_{\blacktriangle}02) = \ldots$$



$$\sigma(7, 21, 59) + (0, 0, 0) = (21 + 0, 59 + 0, 163 + 0)$$

$$\pi(a, 2102) = \ldots$$

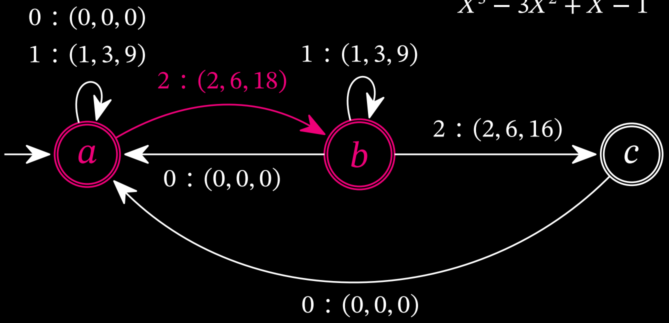
$X^3 - 3X^2 + X - 1$



$(21, 59, 163)$

$\pi(a, 210{\blacktriangle}2) = \ldots$

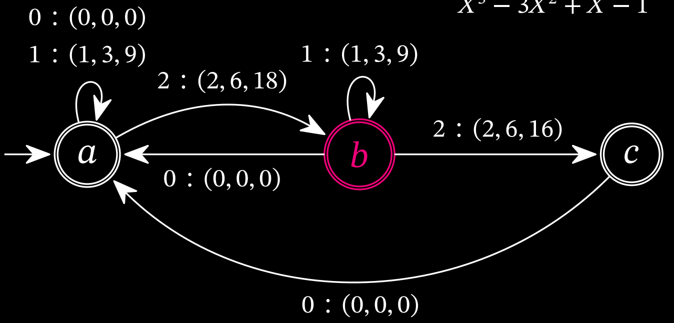
$$X^3 - 3X^2 + X - 1$$



$$\sigma(21, 59, 163) + (2, 6, 18) = (59 + 2, 163 + 6, 451 + 18)$$

$$\pi(a, 2102) = \ldots$$

$X^3 - 3X^2 + X - 1$



$$(61, 169, 469)$$

$$\pi(a, 2102) = (61, 169, 469)$$

Series and Linear Combinations

Definition The **series** $s_{\mathcal{A}}$ of a **sequence automaton** $\mathcal{A}$ maps every word $u \in \Sigma^*$ to the first element of its vector $\pi(q_0, u)[0]$.

Using **product DFA** and linear combination of vector maps, one can combine **sequence automata** to produce **sequence automata**:

- $\mathcal{A} + \mathcal{B}$ of series $s_{\mathcal{A}} \oplus s_{\mathcal{B}}$;
- $\alpha\mathcal{A}$ of series $\alpha s_{\mathcal{A}}$ for every scalar α.

Proposition The **valuation series** $\nu_{\mathcal{N}}$ of an ANS is the **series of a sequence automaton**. As a consequence $\nu_{\mathcal{N}} \oplus \nu_{\mathcal{N}} \oplus -\nu_{\mathcal{N}}$ is also the **series of a sequence automaton**.

Linear Recurrence Sequence Automata

Definition When all sequences in the **vector map** of a sequence automaton are **LRS**, the **recurrence polynomial** of the automaton is the **minimal polynomial** of all these sequences.

Remark When all LRS come from **incidence matrices**, the **recurrence polynomial** is **monic**.

Theorem The **support** of the series of a sequence automaton with **(ultimately) Pisot recurrence polynomial** is **regular**.

The proof lift bounds and techniques from **Bruyère-Hansel** and **Frougny-Solomyak** into sequence automata series.

Case 1. starting from a Pisot substitution

When starting from a (ultimately) **Pisot substitution**, **sequence automata** let us compute numerous DFA an DFAO:

- **addition DFA** for **Dumont-Thomas numeration systems** of fixpoints;

Case 1. starting from a Pisot substitution

When starting from a (ultimately) **Pisot substitution**, **sequence automata** let us compute numerous DFA an DFAO:

- **addition DFA** for **Dumont-Thomas numeration systems** of fixpoints;
- **conversion between numeration systems** sharing the same **recurrence polynomial**, for example the associated **Bertrand** numeration system;

Case 1. starting from a Pisot substitution

When starting from a (ultimately) **Pisot substitution**, **sequence automata** let us compute numerous DFA an DFAO:

- **addition DFA** for **Dumont-Thomas numeration systems** of fixpoints;
- **conversion between numeration systems** sharing the same **recurrence polynomial**, for example the associated **Bertrand** numeration system;
- **abelian complexity DFAO** **[work in progress with J. Shallit]** for Pisot substitutions (including Parikh-colinear morphisms);

Case 1. starting from a Pisot substitution

When starting from a (ultimately) **Pisot substitution**, **sequence automata** let us compute numerous DFA an DFAO:

- **addition DFA** for **Dumont-Thomas numeration systems** of fixpoints;
- **conversion between numeration systems** sharing the same **recurrence polynomial**, for example the associated **Bertrand** numeration system;
- **abelian complexity DFAO** **[work in progress with J. Shallit]** for Pisot substitutions (including Parikh-colinear morphisms);
- **addition DFA** for **Abstract Numeration Systems** built as morphic images of fixpoints;
- ...

Case 2. the Non Pisot substitutions

In general, we can still **construct the sequence automata**.

On a **case-by-case** basis for each application:

- sometimes the **recurrence polynomial** of the automaton is **Pisot**;

  $$a \to aba, \; b \to bab$$

- sometimes it is not but a **manual bound** let us derive a valid DFA;

  $$a \to aba, \; b \to b$$

- sometimes a **pumping argument** let us prove there is no DFA.

  $$a \to ab, \; b \to bc, \; c \to c$$

In Practice

We maintain a collection of **Python tools** to interact with **Walnut**:

- **licofage** — sequence automata library generating Walnut files:
  - basic usage: give a substitution, get the addition DFA and word DFAO;
  - advanced usage: generic sequence automata manipulation in Python;
- **ratser** — missing library for rational power series and semigroup trick;
- **walnut-kernel** — a Jupyter kernel to combine Walnut and all this (hello Walnut Notebooks!);
- **jupywalnut** — a Docker container for a simplified use of everything.

Everything is available on **GitHub**, install Python libs from **Pypi**.

```
$ docker pull nopid/walnut
$ docker run --rm -it -p 8888:8888 nopid/walnut
```