Computation of invariant densities for continued fraction algorithms

Paul MERCAT

November 21, 2022

Abstract

We introduce the notion of matrices graph, defining continued fraction algorithms where the past and the future are almost independent, and we give an algorithm to convert more general algorithms to matrices graph. We give an algorithm that computes exact invariant densities of some continued fraction algorithms, including classical ones and some extensions of it. For two coordinates, we provide a more precise algorithm that computes invariant density as soon as it is a rational fraction. And for any finite set of quadratic numbers, we construct a continued fraction algorithm whose invariant density are rational fractions where the quadratic numbers appears.

Contents

1 Settings

1.1 Matrices graphs and win-lose graphs

Usually, an additive continued fraction algorithm is a map from $\mathbb{R}_+^d$ to $\mathbb{R}_+^d$ which is linear by pieces, and usually the pieces are polyhedra. In this paper, we consider continued fraction algorithms given by matrices graph, which is more restrictive since it corresponds to algorithms where the past and the future are almost independant. But we will see that usual algorithms can be converted to matrices graph (see Section 3), and that it has a lot of advantages. See Section 5 for some examples.

A **matrices graph** is a finite oriented graph, labeled by matrices of size $d \times d$, such that from each vertex s, if the leaving edges are labeled by matrices $\mathcal{M}_s = \{m_1, ..., m_k\}$, then we have $\mathbb{R}_+^d = \bigcup_{m \in \mathcal{M}_s} m\mathbb{R}_+^d$, and the union is quasi disjoint. In particular, such graph defines a deterministic automaton if we define some initial state and final states. We denote $i \xrightarrow{m} j$ if it is an edge (also called transition) in the graph.

A matrices graph defines a map $F : \mathbb{R}_+^d \times S \to \mathbb{R}_+^d \times S$ by

$$F(x, i) = (m^{-1}x, j) \text{ if } i \xrightarrow{m} j \text{ such that } x \in m\mathbb{R}_+^d,$$

where S is the set of vertices (also called states) of the graph. We call this map the **continued fraction algorithm** associated to the matrices graph. It is well-defined almost everywhere, and we can also iterate it infinitely often on a set of full Lebesgue measure (everywhere except a countable union of hyperplanes). We see that the only information kept from the past is the element of the set S.

A **win-lose graph** (also called **simplicial system** in [Fougeron]) is another way to describe a continued fraction algorithm. It is an oriented graph, labeled by integers $\{0, ..., d-1\}$, such that from each vertex, there exists at most one edge labeled by each integer.

It defines a map $F : \mathbb{R}_+^d \times S \to \mathbb{R}_+^d \times S$ by

$$F(x,i) = (x',k) \text{ if } i \xrightarrow{j} k \text{ such that } \forall i \xrightarrow{l} k', x_j \leq x_l$$

where x' is the vector x where we subtract x_j to every x_l such that $i \xrightarrow{l} k'$ with $l \neq j$ and $k' \in S$. A win-lose graph can be seen as a matrices graph, where the matrix corresponding to an edge $i \xrightarrow{j} k$ is $I_d + \sum_{i \xrightarrow{l} k', l \neq j} E_{j,l}$, where $E_{j,l}$ is the matrix with a 1 in coordinate j,l and 0 at every other coordinates.

The **continued fraction expansion** (or just **expansion**) of a point $(x,i) \in \mathbb{R}_+^d \times S$, is the sequences of matrices $(m_n)_{n\in\mathbb{N}}$ that appears when iterating the continued fraction algorithm from (x,i). We say that a continued fraction algorithm is **convergent** if for almost every $x \in \mathbb{R}_+^d$, its expansion $(m_n)_{n\in\mathbb{N}}$ satisfies $\bigcap_{n\in\mathbb{N}} m_0...m_n\mathbb{R}_+^d = \mathbb{R}_+ x$.

1.2 Invariant density

In all this subsection, we consider a matrices graph with vertices S.

We say that μ is an **invariant measure** on $\mathbb{R}_+^d \times S$ if for every measurable set E we have $\mu(F^{-1}E) = \mu(E)$, and for every $\alpha \in \mathbb{R}_+$, $\mu(\alpha E) = \mu(E)$, where $\alpha E = \{(\alpha x, i) \in \mathbb{R}_+^d \times S \mid (x,i) \in E\}$.

If an invariant measure is absolutely continuous with respect to the Lebesgue measure on $\mathbb{R}_+^d \times S$, then its density function is called **invariant density**.

Let $f : \mathbb{R}_+^d \times S \to \mathbb{R}_+$ be an invariant density. Then, for every $\alpha \in \mathbb{R}_+$, $x \in \mathbb{R}_+^d$ and $i \in S$, $f(\alpha x, i) = \frac{1}{\alpha^d} f(x,i)$. We call **invariant density at state** $i \in S$ the map $f_i : \mathbb{R}_+^d \to \mathbb{R}_+$ defined by $f_i(x) := f(x,i)$.

An invariant density satisfy for every state $j \in S$ and almost every $x \in \mathbb{R}_+^d$ the relation

$$f_j(x) = \sum_{i \xrightarrow{m} j} |\det(m)|\, f_i(mx).$$

Usually densities are expressed as functions restricted to the standard simplex $\{x \in \mathbb{R}_+^d \mid (1,...,1)x = 1\}$. In that case, there is a Jacobian of the map $x \mapsto mx/\left\|mx\right\|_1$ that appears in such functional relations. But it is enough to take the restriction of functions f_j to the standard simplex to get usual densities, and it avoid making arbitrary choice to parameterize the standard simplex.

We say that a continued fraction algorithm is **ergodic** if there exists an unique ergodic invariant measure absolutely continuous with respect to Lebesgue.

1.3 Rational languages and limit sets

A **automaton** is a finite graph, labeled by letters in a finite alphabet, with some state (or vertex) called **initial state**, and a set of **final states**. In this article, we denote initial states by thick circles, and final states by double circles. The **language recognized by an automaton** is the set of finite words labeling a path from the initial state to a final state. We denote $s_0 \xrightarrow{u_1} ... \xrightarrow{u_n} s_n$ if there is a path in the automaton labelled by a word $u_1...u_n$.

A **rational language** (or **regular language**) is a language recognized by an automaton. Any rational language is recognized by a **deterministic automaton**, that is an automaton such that if $s \xrightarrow{i} s'$ and $s \xrightarrow{i} s''$, then $s' = s''$.

The set of rational languages is stable by many operations: union, intersection, complementary, mirror, image by a morphism, inverse image by a morphism, Kleene star. See for example [Carton] for more details. The set of rational languages is also stable by prefixes: if L is a rational language over an alphabet A, then

$$\mathrm{Pref}(L) = \{u \in A^* \mid \exists v \in A^*,\ uv \in L\}$$

is rational. Indeed, if a pruned automaton recognize L, then the same automaton where every state is final recognize $\mathrm{Pref}(L)$.

Let L be a rational language over an alphabet $A \subset M_d(\mathbb{R}_+)$ of matrices. We define the **limit set** of L by

$$\Lambda_L = \bigcap_{n\in\mathbb{N}} \bigcup_{k\geq n} \bigcup_{u_1\dots u_k \in L} u_1 \dots u_k \mathbb{R}_+^d.$$

In other words, the limit set is the set of vectors x such that $x \in u_1..u_n\mathbb{R}_+^d$ for infinitely many words $u_1...u_n \in L$.

In the particular case where L is stable by prefixes, we have

$$\Lambda_L = \bigcap_{n\in\mathbb{N}} \bigcup_{u_1\dots u_n \in L} u_1 \dots u_n \mathbb{R}_+^d,$$

and Λ_L is a closed set.

If B and C are two languages over alphabets included in $M_d(\mathbb{R}_+)$, then

$$\Lambda_{B\cup C} = \Lambda_B \cup \Lambda_C, \quad \text{and} \quad \bigcup_{w\in B} w\Lambda_C \subseteq \Lambda_{BC} \subseteq \Lambda_B \cup \bigcup_{w\in B} w\Lambda_C.$$

1.4 Natural extension and dual algorithm

In all this subsection, we consider a matrices graph, defining some continued fraction algorithm on $\mathbb{R}_+^d \times S$.

Definition 1.1. *We say that $(D_i)_{i\in S}$ are **domains** of the matrices graph if*

$$\forall i \in S,\ D_i = \biguplus_{j \xrightarrow{m} i} {}^t m D_j,$$

where $\biguplus$ means that the union is disjoint in Lebesgue measure.

Such domains are not necessarily unique, but they always exists (see [AS]).

Let $(D_i)_{i\in S}$ be domains for the matrices graph, and let $D = \bigcup_{i\in S} D_i \times \{i\}$. Then, the **natural extension** of the continued fraction algorithm is the map $\tilde{F} : \mathbb{R}_+^d \times D \to \mathbb{R}_+^d \times D$ defined by

$$\tilde{F}(x,(y,i)) = (m^{-1}x, ({}^t m y, j))$$

where $i \xrightarrow{m} j$ is a transition such that $x \in m\mathbb{R}_+^d$. This maps is well-defined almost everywhere and it preserves the Lebesgue measure of $\mathbb{R}^d \times \mathbb{R}^d \times S$. Moreover, by definition of the set D, this map is almost everywhere one-to-one.

We define the **dual algorithm** as the map $F^* : D \to D$ such that

$$F^*(y, j) = ({}^t m^{-1} y, i),$$

if $i \xrightarrow{m} j$ is such that $y \in {}^t m D_i$. This dual algorithm is well-defined almost everywhere on D. However, the set D could have zero Lebesgue measure. For example, the dual of the fully subtractive algorithm is the Arnoux-Rauzy algorithm, defined on a set D of zero Lebesgue measure called the Rauzy gasket (see [AHS] for more details).

For each state i, we define the **domain language** $\mathcal{D}_i$ by the set of words $m_1^t ... m_n^t$ such that $m_n ... m_1$ labels a path toward i in the matrices graph. It is a rational language. Notice that languages $\mathcal{D}_i$ are stable by prefixes, thus

$$\Lambda_{\mathcal{D}_i} = \{y \in \mathbb{R}_+^d \mid \forall n \in \mathbb{N},\ \exists i_n \xrightarrow{m_n} ... \xrightarrow{m_1} i_0 = i,\ y \in {}^t m_1 ... {}^t m_n \mathbb{R}_+^d\}.$$

It gives an upper bound of domains.

Lemma 1.2. *If $(D_i)_{i \in S}$ are domains, then $\forall i \in S$, $D_i \subseteq \Lambda_{\mathcal{D}_i}$.*

And in some cases, Λ_{D_i} are domains. It is the case for example with the fully subtractive on two letters, and it is what we use in Section 4 and in Section 6. Domain languages are also useful to find domains that are unions of simplicies, see Section 2.

1.5 Invariant densities from natural extension

In the following, we consider some matrices graph with vertices S, and a set $D = \bigcup_{i \in S} D \times \{i\}$ for some domains $(D_i)_{i \in S}$. Let Γ be the subset of $\mathbb{R}_+^d \times D$ defined by

$$\Gamma = \{(x, (y, i)) \in \mathbb{R}_+^d \times D \mid (y|x) \leq 1\}.$$

Notice that the natural extension $\tilde{F}$ preserves this set Γ. Hence, we have

Lemma 1.3. *The maps $f_i(x) = \lambda(\{y \in D_i \mid (y|x) \leq 1\})$ define an invariant density for the continued fraction algorithm.*

Such map f_i can be explicitly computed if the domain D_i is a simplex, thanks to the following formulae.

Lemma 1.4. *Let m be a square matrix of size d. Then we have*

$$\lambda(\{y \in m\mathbb{R}_+^d \mid (y|x) \leq 1\}) = \frac{|\det(m)|}{d! \prod({}^t m x)},$$

where $\prod({}^t m x)$ is the product of coefficients of the vector ${}^t m x$.

Hence, if the domain D_i is a finite union of quasi-disjoint simplicies, then we can compute the density, and it is a rational fraction.

If the domain D_i is not a finite union of simplicies, we don't have an explicit formulae, but we can write the density as an integral:

Lemma 1.5. *For every measurable cone $E \subseteq \mathbb{R}_+^d$ and every $x \in \mathbb{R}^d$,*

$$\lambda(\{y \in E \mid (y|x) \leq 1\}) = \int_{y \in \mathbb{P}E} \frac{dy}{(y|x)^d},$$

where $y \in \mathbb{P}E$ means that we integrate over any set of unique representatives of $(E \backslash \{0\})/\mathbb{R}_+^*$.

Notice that if $\lambda(D_i) = 0$, then the corresponding density is the null function. Hence, if every domain has zero Lebesgue measure, it doesn't permits to get a non-trivial invariant density.

2 Computation of an invariant density for a matrices graph

In [AL], Arnoux-Labbe compute invariant densities by considering a natural extension, constructed ad hoc for each example, following ideas of [AN]. The advantage to rather consider matrices graphs rather than general continued fraction algorithms is that a natural extension is easier to define (see subsection 1.4). Moreover, the computation is easier and gives a regular function, rather than a piecewise regular map. Another advantage is that it is easier to convert a matrices graph to a win-lose graph, and for a win-lose graph we have an algorithmic criterion to test if the continued fraction algorithm is ergodic (see [Fougeron]).

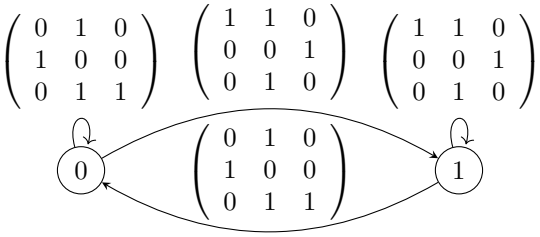
The following of this section is devoted to the computation of a decomposition of domains as finite unions of simplicies, for a matrices graph with set of vertices S.

2.1 Minimized continued fraction algorithm

The first step of the algorithm is to minimize the continued fraction algorithm. This is done by seeing the matrices graph as a deterministic automaton, and computing an automaton recognizing the same language with the minimal number of states, using for example the Hopcroft's minimization algorithm.

If the continued fraction algorithm was an extension of a simpler algorithm, the minimization permits to work with the simpler algorithm.

Example 2.1. *Consider the following matrices graph.*



Then, the minimized matrices graph is the Cassaigne algorithm since the language of both states are all finite words over the alphabet

$$\left\{ \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \right\}.$$

2.2 Computation of convex polyhedra domains

In this subsection, we give an algorithm to compute convex polyhedra that are domains, for a given matrices graphs with set of states S.

We say that a cone $C \subseteq \mathbb{R}_+^d$ is a **convex polyhedron** if there exists a matrix $m \in M_{d,k}(\mathbb{R}_+)$ such that $C = m\mathbb{R}_+^k$. We say that a vector $v \in C$ is an **extremal point** of a cone $C \subseteq \mathbb{R}_+^d$ if $C \backslash \mathbb{R}_+ v$ is convex.

The algorithm to compute convex polyhedra is based on the following.

Lemma 2.2. *Assume that $(D_i)_{i\in S}$ are domains with finitely many extremal points of sum 1. Then, for every $i \in S$ and every extremal point V of D_i, we have*

$$V = m_u V_v$$

where V_v is a positive eigenvector of m_v, u and v are two words such that $uv^* \subseteq \mathcal{D}_i$, with v not the empty word, and m_u denotes the product of matrices of u.

Proof. For every $j \in S$, we have

$$D_j = \bigcup_{i \xrightarrow{m} j} {}^t m D_i.$$

Hence, extremal points of D_j are of the form ${}^t mV$, with V a extremal point of D_i. If we iterate it, we see that if V is an extremal point of D_j, then there exists a sequence $(V_n)_{n\in\mathbb{N}}$ of vectors and an infinite word w, such that for every $n \in \mathbb{N}$, $V = w_1 ... w_n V_n$, $w_1 ... w_n \in \mathcal{D}_j$, and V_n is an extremal point of D_k for some $k \in S$. Then, the word w is ultimately periodic: there exists $u \in A^*$ and $v \in A^+$ such that $w = uv^\omega$, and it concludes the proof. □

If moreover $uv^* \subseteq \mathcal{D}_i$ is such that $\bigcap_{n\in\mathbb{N}} m_v^n \mathbb{R}_+^d = \mathbb{R}_+ V_v$, then we have $m_u V_v \in D_i$, for any domains $(D_i)_{i\in S}$.

The idea is to compute such vectors V_v for many loops of the matrices graph, and to stabilize it by adding vectors for each edge. It gives some convex cones D_i, and the stabilization step guarantees if it terminates that $\bigcup_{i \xrightarrow{m} j} {}^t m D_i \subseteq D_j$. If we considered only vectors that are necessarily in domains as above, then this union is necessarily Lebesgue-disjoint. And if we consider enough loops and that there exists domains with finitely many extremal points, then we get the equality and it gives domains.

More precisely, the algorithm is as follow:

1. For each state i of the matrices graph, compute matrices m of loops starting at i, and for each such matrix m, compute extremal points of sum 1 of $\lim_{n\to\infty} {}^t m^n (\mathbb{R}_+^*)^d$ with the Jordan form. If we are looking only for rational vectors, then we can stop as soon as it remains only loops with strictly positive matrices, otherwise continue this computation and do the following in parallel.

2. For each transition $i \xrightarrow{m} j$ in the matrices graph, and for each vector V computed for state i, we add vector ${}^t mV$ to state j. Then, for each state i, we keep only extremal points: if a vector is a linear span of the other vectors with non-negative coefficients, then we remove it.

3. Re-do Step 2 until the number of vectors doesn't increase, and stop if $\forall j \in S$, $m_j \mathbb{R}_+^{k_j} = \bigcup_{i \xrightarrow{m} j} {}^t m m_i \mathbb{R}_+^{k_i}$.

Example 2.3. *Consider the Cassaigne's continued fraction algorithm. The two only loops giving rational vectors are the two trivial ones, labeled respectively by matrices* $m_0 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix}$ *and* $m_1 = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$. *It gives corresponding left eigenvectors* $(1,1,0)$ *and* $(0,1,1)$.

Then, if we consider images of these vectors by the transposed of the matrices, we get one more vector $(1,1,1)$.

Then, we easily check that the cone $C = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \mathbb{R}_+^d$ *satisfies*

$$ {}^t m_0 C \uplus {}^t m_1 C = C.$$

Thus the algorithm terminates here, and C *is a domain.*

2.3 Computation of domains for an extension

We assume that domains $(D'_i)_{i\in S'}$ have been found for the minimized algorithm, and we want to compute domains $(D_i)_{i\in S}$ for the original algorithm, which is an extension of the minimized one.

The idea is to decompose each domain language $\mathcal{D}_i$ of the original algorithm as a finite union of the languages $W_{i,j}\mathcal{D}'_j$, where $W_{i,j}$ are rational languages,

and $\mathcal{D}'_j$ are domain languages of the minimized algorithm. Such decomposition is quite natural since every language L over an alphabet A can be decomposed as $L = \bigcup_{a\in A} a(a^{-1}L)$, and since $w^{-1}\mathcal{D}_i$ are finite union of $\mathcal{D}_j$'s and $\mathcal{D}'_k$'s are also finite union of $\mathcal{D}_j$'s.

If such finite union of languages can be found, then we define

$$D_i := \bigcup_{j\in S'} W_{i,j} D'_j,$$

where we denote $W_{i,j}D'_j := \bigcup_{u_1\dots u_k\in W_{i,j}} u_1\dots u_k D'_j$.

Lemma 2.4. *$(D_i)_{i\in S}$ are domains, and each D'_j is partionned by some D_i's.*

Proof. Since D'_k's are domains, notice that an equality of languages $\bigcup_{k\in S'} A_k\mathcal{D}'_k = \bigcup_{k\in S'} B_k\mathcal{D}'_k$ implies the equality $\bigcup_{k\in S'} A_k D'_k = \bigcup_{k\in S'} B_k D'_k$. Moreover, if W is a language such that $W\mathcal{D}'_k \subseteq \mathcal{D}'_i$, then the union $\bigcup_{w\in W} m_w D'_k$, that we denote WD'_k in the following, is Lebesgue disjoint.

States S' of the minimized algorithm can be seen as sets of states of S: $S' \subseteq \mathcal{P}(S)$. And for every $I \in S'$, we have $\mathcal{D}'_I = \biguplus_{i\in I} \mathcal{D}_i$. So, we have $\mathcal{D}'_I = \bigcup_{i\in I}\bigcup_{K\in S'} W_{i,K}\mathcal{D}'_K$. Thus, we obtain $D'_I = \biguplus_{i\in I}\bigcup_{K\in S'} W_{i,K}D'_K = \biguplus_{i\in I} D_i$.

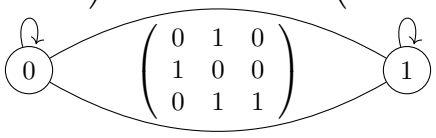
By the same argument, we show that for every $j \in S$, $D_j = \bigcup_{j\xrightarrow{m} i} {}^t m D_i$, and the union is quasi-disjoint since $\forall J \in S'$, $\forall j \in J$, $\mathcal{D}_j \subseteq \mathcal{D}'_J$. □

Such a decomposition of rational languages can be obtained with usual operations on rational languages. Indeed, for each $i \in S$, take a deterministic automaton recognizing $\mathcal{D}_i$. For each $j \in S'$, let S_j be the set of states whose language is equal to $\mathcal{D}'_j$. Then, the language $W_{i,j}$ is recognized by the automaton where we remove every outgoing edges from S_i and we set S_i as the set of final states. Then, we easily check if equalities $\mathcal{D}_i = \bigcup_{j\in S'} W_{i,j}\mathcal{D}'_j$ are satisfied.

Then, if languages $W_{i,j}$ are finite, the formulae of Lemma 1.4 permits to compute the invariant density, by splitting this union as quasi-disjoint simplicies.

Example 2.5. *Consider the following extension of the Cassaigne continued fraction algorithm.*

$$\begin{pmatrix} 0 & 1 & 0\\ 1 & 0 & 0\\ 0 & 1 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 1 & 0\\ 0 & 0 & 1\\ 0 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 1 & 1 & 0\\ 0 & 0 & 1\\ 0 & 1 & 0 \end{pmatrix}$$



$$\begin{pmatrix} 0 & 1 & 0\\ 1 & 0 & 0\\ 0 & 1 & 1 \end{pmatrix}$$

The minimized matrices graph is the Cassaigne algorithm whose a domain is $D'_0 = \begin{pmatrix} 0 & 1 & 1\\ 1 & 1 & 1\\ 1 & 0 & 1 \end{pmatrix} \mathbb{R}^d_+$, *and whose domain language is* $\mathcal{D}'_0 = \{ {}^t m_0, {}^t m_1\}^*$,

where $m_0 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix}$ *and* $m_1 = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$.

We easily see that $\mathcal{D}_0$ *is the language of all finite words over* $\{{}^t m_0, {}^t m_1\}$ *starting by* ${}^t m_0$*, and* $\mathcal{D}_1$ *is the language of all words over the same alphabet starting by* ${}^t m_1$*.*

Thus, we have $\mathcal{D}_0 = {}^t m_0 \mathcal{D}'_0$ *and* $\mathcal{D}_1 = {}^t m_1 \mathcal{D}'_0$*, so we get that* $D_0 = {}^t m_0 D'_0 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 0 & 1 \end{pmatrix} \mathbb{R}^d_+$ *and* $D_1 = {}^t m_1 D'_0 = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 1 & 1 \end{pmatrix} \mathbb{R}^d_+$ *are domains.*

Then, we get an invariant density thanks to the formulae of Lemma 1.4: $f_0(x,y,z) = \frac{1}{(x+y)(x+y+z)(x+2y+z)}$ *and* $f_1(x,y,z) = \frac{1}{(y+z)(x+y+z)(x+2y+z)}$.

3 Converting a continued fraction algorithm to a matrices graph

In this section, we give an algorithm to convert a general continued fraction algorithm defined as a piecewise linear map on pieces that are polyhedra, to a matrices graph (see subsection 1.1 for a definition). The algorithm may not terminate, but it terminates for every usual continued fraction algorithms.

We represent the input continued fraction algorithm as a graph with vertices S and edges $i \xrightarrow{m,D} j$, where $m \in M_d(\mathbb{R}_+)$ and D is a non-negative matrix with d rows such that $m^{-1}D$ is non-negative. We denote $c(D)$ the number of columns of D. And we assume that for every $i \in S$, the union $\bigcup_{i \xrightarrow{m,D} j} D\mathbb{R}_+^{c(D)}$ is quasi-disjoint.

It defines a partial map $F : \mathbb{R}^d_+ \times S \to \mathbb{R}^d_+ \times S$ by $F(x,i) = (m^{-1}x, j)$ if $i \xrightarrow{m,D} j$ such that $x \in D\mathbb{R}_+^{c(D)}$. The map is defined almost everywhere if for every $i \in S$, $\bigcup_{i \xrightarrow{m,D} j} D\mathbb{R}_+^{c(D)} = \mathbb{R}^d_+$, but we don't need this hypothesis in the following.

The output of the algorithm is a graph labeled by square matrices, with states $S \times \mathcal{M}$, where $\mathcal{M}$ is a finite set of square matrices. See Algorithm 1 for the algorithm.

Remark 3.1. *In Algorithm 1 there is choices to make, to partition the cone* $m^{-1}(D\mathbb{R}_+^{c(D)} \cap I\mathbb{R}^d_+)$ *by projective simplicies* $J\mathbb{R}^d_+$*. And we have to normalize such matrices* J *in order that same cones give same matrices.*

The following proposition explain the link between the input continued fraction algorithm and the output matrices graph. It shows that under a small hypothesis on the input graph, the output graph is indeed almost a matrices graph.

Proposition 3.2. *Let* $F : \mathbb{R}^d_+ \times S \to \mathbb{R}^d_+ \times S$ *be the continued fraction algorithm corresponding to the input. Then, the continued fraction algorithm of the output*

Start with an empty result graph;
$d \leftarrow$ number of lines of matrices;
$\mathcal{S} \leftarrow \{(i, I_d) \mid i \in S\}$;
while $\mathcal{S} \neq \emptyset$ **do**
  Pop (i, I) from $\mathcal{S}$;
  Set (i, I) as seen;
  for $i \xrightarrow{m,D} j$ **do**
    if $\dim(D\mathbb{R}_+^{c(D)} \cap I\mathbb{R}_+^d) = d$ **then**
      Compute a set $\mathcal{J}$ of square matrices such that $\biguplus_{J\in\mathcal{J}} J\mathbb{R}_+^d = m^{-1}(D\mathbb{R}_+^{c(D)} \cap I\mathbb{R}_+^d)$;
      for $J \in \mathcal{J}$ **do**
        Add the edge $(i, I) \xrightarrow{I^{-1}mJ} (j, J)$ to the result graph;
        if (j, J) *has not already been seen* **then**
          Add (j, J) to $\mathcal{S}$;
        end
      end
    end
  end
end

Algorithm 1: Algorithm to convert a continued fraction algorithm to a matrices graph

$G : \mathbb{R}_+^d \times S \times \mathcal{M} \to \mathbb{R}_+^d \times S \times \mathcal{M}$ *is well-defined, and we have for every* $n \in \mathbb{N}$,

$$F^n \circ \phi = \phi \circ G^n,$$

where $\phi : \mathbb{R}_+^d \times S \times \mathcal{M} \to \mathbb{R}_+^d \times S$ *is defined by* $\phi((x, i, I)) = (Ix, i)$.

Moreover, if the input graph satisfies

$$\forall i \xrightarrow{m,D} j, \quad m^{-1}D\mathbb{R}_+^{c(D)} \subseteq \bigcup_{j \xrightarrow{m',D'} k} D'\mathbb{R}_+^{c(D')},$$

then the graph obtained by the Algorithm 1 is a matrices graph, up to remove vertices with the identity matrix.

Proof. Let $(i, I) \in S \times \mathcal{M}$ be a state of the output graph. By construction, for every $i \xrightarrow{m,D} j$, there exists a set $\mathcal{J}_j$ such that $\bigcup_{J\in\mathcal{J}_j} J\mathbb{R}_+^d = m^{-1}(I\mathbb{R}_+^d \cap$

$D\mathbb{R}_+^{c(D)}$), thus we have

$$\begin{aligned}\bigcup_{(i,I)\xrightarrow{M}(j,J)} M\mathbb{R}_+^d &= \bigcup_{i\xrightarrow{m,D}j}\bigcup_{J\in\mathcal{J}_j} I^{-1}mJ\mathbb{R}_+^d \\ &= \bigcup_{i\xrightarrow{m,D}j} I^{-1}mm^{-1}(I\mathbb{R}_+^d\cap D\mathbb{R}_+^{c(D)}) \\ &= \mathbb{R}_+^d\cap I^{-1}\bigcup_{i\xrightarrow{m,D}j} D\mathbb{R}_+^{c(D)}.\end{aligned}$$

Now, let us prove that this union is quasi-disjoint. Let $(i,I)\xrightarrow{M}(j,J)$ and $(i,I)\xrightarrow{M'}(k,K)$ be two distinct edges. Then, there exists $i\xrightarrow{m,D}j$ and $i\xrightarrow{m',D'}k$ such that $M=I^{-1}mJ$ and $M'=I^{-1}m'K$. If $j=k$, then $m=m'$, and $J\mathbb{R}_+^d$ and $K\mathbb{R}_+^d$ are quasi-disjoint by construction, thus $M\mathbb{R}_+^d$ and $M'\mathbb{R}_+^d$ are quasi-disjoint. Otherwise, by hypothesis $D\mathbb{R}_+^{c(D)}$ and $D'\mathbb{R}_+^{c(D')}$ are quasi-disjoint, and we have $J\mathbb{R}_+^d\subseteq m^{-1}D\mathbb{R}_+^{c(D)}$ and $K\mathbb{R}_+^d\subseteq m'^{-1}D'\mathbb{R}_+^{c(D')}$ by construction, thus $M\mathbb{R}_+^d$ and $M'\mathbb{R}_+^d$ are quasi-disjoint.

Hence, the map G is well-defined, but not necessary on the whole positive cone. Let us show that we have $\phi(G(I^{-1}x,i,I))=F(x,i)$ for all $(i,I)\in S\times\mathcal{M}$ and for almost every $x\in I\mathbb{R}_+^d$. Indeed, we have a transition $(i,I)\xrightarrow{M}(j,J)$ in the matrices graph only if we have a transition $i\xrightarrow{m,D}j$ in the input graph, with $M=I^{-1}mJ$ and $J\mathbb{R}_+^d\subseteq m^{-1}(I\mathbb{R}_+^d\cap D\mathbb{R}_+^{c(D)})$. Thus, if $G(I^{-1}x,i,I)=(M^{-1}I^{-1}x,j,J)$, then we have $I^{-1}x\in M\mathbb{R}_+^d$, so $x\in mJ\mathbb{R}_+^d\subseteq D\mathbb{R}_+^{c(D)}$. Hence, $F(x,i)=(m^{-1}x,j)=\phi((I^{-1}m^{-1}x,j,J))=\phi(G(I^{-1}x,i,I))$. Then, we get the equality for every $n\in\mathbb{N}$ by iterating.

If moreover we assume the additional hypothesis on the input graph and that $I\neq I_d$, then $\mathbb{R}_+^d\cap I^{-1}\bigcup_{i\xrightarrow{m,D}j} D\mathbb{R}_+^{c(D)}=\mathbb{R}_+^d$ since by construction $I\mathbb{R}_+^d$ is included in $m'^{-1}(K\mathbb{R}_+^d\cap D'\mathbb{R}_+^{c(D')})\subseteq\bigcup_{i\xrightarrow{m,D}j} D\mathbb{R}_+^{c(D)}$ for some transition $k\xrightarrow{m',D'}i$ in the input graph, and some K. Thus, the output graph is a matrices graph up to remove states with the identity matrix. □

We can recover the invariant density of the original continued fraction algorithm from the invariant density of the computed matrices graph thank to the following formulae.

Lemma 3.3. *Let a continued fraction algorithm defined by a graph with set of vertices S, and let a corresponding matrices graph with vertices $S\times\mathcal{M}$, obtained from Algorithm 1. Let $(f_{i,I})_{(i,I)\in S\times\mathcal{M}}$ be an invariant density for the matrices graph. Then, for every $i\in S$ the maps*

$$f_i(x)=\sum_{I\in\mathcal{M},\ x\in I\mathbb{R}_+^d}\left|\det(I^{-1})\right| f_{i,I}(I^{-1}x)$$

define an invariant density for the input continued fraction algorithm.

Proof. Let μ be the measure on $\mathbb{R}_+^d \times S$ with density $(f_i)_{i\in S}$ with respect to Lebesgue, and let ν be the measure on $\mathbb{R}_+^d \times S \times \mathcal{M}$ with density $(f_{i,I})_{(i,I)\in S\times\mathcal{M}}$ with respect to Lebesgue. We easily check that $\mu = \phi_*\nu$, where $\phi : \mathbb{R}_+^d \times S \times \mathcal{M} \to \mathbb{R}_+^d \times S$ is defined by $\phi(x, i, I) = (Ix, i)$. Then, we check that μ is an invariant density, using the fact that ν is an invariant density and using the Proposition 3.2. □

Notice that the invariant density for a general continued fraction algorithm is not continuous in general, since the condition $x \in I\mathbb{R}_+^d$ is not continuous.

4 Win-lose graph on two letters

In Section 2, we gave an algorithm to compute the invariant density for some matrices graphs. But the algorithm may fail to find the invariant density, although it is rational fractions. In this section, we give an algorithm to compute the exact invariant density for every continued fraction algorithm given by a win-lose graph on two letters, as soon as it is rational fractions. The algorithm decides if the invariant density is rational fractions and if it is the case it computes it. It is done by describing domains, by computing their boundaries.

4.1 Computation of boundaries

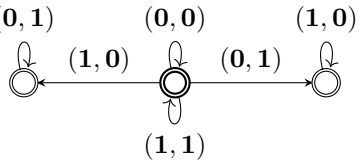
In this subsection, we give a way to compute the boundary of the limit set of some rational language L stable by prefixes, over the alphabet $\{\mathbf{0}, \mathbf{1}\}$, where we denote $\mathbf{0} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ and $\mathbf{1} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$ to lighten the notations.

In order to describe the boundary of the limit set, we need to understand which words correspond to neighboring cones. It is given by the following.

Lemma 4.1. *For every $n \in \mathbb{N}$ and every $(u, v) \in (\{\mathbf{0}, \mathbf{1}\}^2)^n$, we have*

$$u\mathbb{R}_+^d \cap v\mathbb{R}_+^d \neq \emptyset$$

if and only if (u, v) is recognized by the automaton



where the central state is initial, and every state is final.

Proof. Easy verification. □

We call the automaton of this lemma the **relations automaton**, and its language is denoted by L^{rel}. It permits to compute the boundary:

Proposition 4.2. *Let $L \subseteq \{\mathbf{0},\mathbf{1}\}^*$ be a rational language stable by prefixes. The boundary of the limit set of L is the limit set of the rational language*

$$\partial L := \mathrm{prune}^\infty(p_1(L \times \mathrm{Pref}(L^c) \cap L^{\mathrm{rel}})) \cup L^{min} \cup L^{max},$$

where $L^c = \{\mathbf{0},\mathbf{1}\}^* \backslash L$ denotes the complementary of L, L^{min} and L^{max} are respectively the languages recognizing the smallest and the greatest words of L in lexicographical order, $p_1 : (\{\mathbf{0},\mathbf{1}\}^2)^* \to \{\mathbf{0},\mathbf{1}\}$ is the projection on the first coordinate, and $\mathrm{prune}^\infty(L)$ *remove all words of L that cannot be extended to an arbitrarily longer word of L.*

Proof. Language L^{min} (respectively L^{max}) is rational and is easily computed by following the minimal (respectively maximal) path in the automaton of L up to a loop. Then, the language ∂L is rational since it is obtained by usual operations on rational languages (complementary, prefixes, intersection, union, image by the morphism p_1). The prune^∞ also preserves the fact to be rational: indeed, if a pruned automaton recognize a language L', then $\mathrm{prune}^\infty(L')$ is recognized by the same automaton where we remove every state from which we cannot reach a loop. The operation prune^∞ doesn't change the limit set of the language, but it permits to get a simpler language.

Remark that ∂L is stable by prefixes, since this property is stable by product, intersection, union, projection on the first coordinate, and prune^∞.

Let us show that $\partial\Lambda_L = \Lambda_{\partial L}$. The boundary $\partial\Lambda_L$ is equal to $(\overline{\mathbb{R}_+^d \backslash \Lambda_L} \cap \Lambda_L) \cup \Lambda_{L^{min}} \cup \Lambda_{L^{max}}$, since the lexicographical order corresponds to the order on $\mathbb{R}_+^2$ defined by $(x,y) \leq (x',y')$ if and only $xy' \leq x'y$.

Let $x \in \overline{\mathbb{R}_+^d \backslash \Lambda_L} \cap \Lambda_L$. Then, there exists a sequence of elements $x_n \in \mathbb{R}_+^d \backslash \Lambda_L$, such that $\lim_{n\to\infty} x_n = x$. Let $n \in \mathbb{N}$. As $x \in \Lambda_L$, there exists $u_1...u_n \in L$ such that $x \in u_1...u_n\mathbb{R}_+^d$. Let $N \in \mathbb{N}$ be large enough such that $x_N \in v_1...v_n\mathbb{R}_+^d$, for some word $v_1...v_n \in \{\mathbf{0},\mathbf{1}\}^n$ such that $(u_1,v_1)...(u_n,v_n) \in L^{\mathrm{rel}}$. As $x_N \notin \Lambda_L$, there exists an extension of the word $v_1...v_n$ that belongs to L^c. Thus, we have $u \in p_1(L \times \mathrm{Pref}(L^c) \cap L^{\mathrm{rel}})$, and we conclude that $x \in \Lambda_{\partial L}$.

Reciprocally, let $x \in \Lambda_{p_1(L\times\mathrm{Pref}(L^c)\cap L^{\mathrm{rel}})}$. Then, for every $n \in \mathbb{N}$, there exists $u_1...u_n \in L$ and $v_1...v_n \in \mathrm{Pref}(L^c)$ such that $(u_1,v_1)...(u_n,v_n) \in L^{\mathrm{rel}}$ and $x \in u_1...u_n\mathbb{R}_+^d$. In particular, $x \in \Lambda_L$. And for every $v_1...v_N \in L^c$, the interior of $v_1...v_N\mathbb{R}_+^d$ is disjoint of Λ_L, and the distance between $v_1...v_N\mathbb{R}_+^d$ and x is less than $2\left\|x\right\|_1/n$. Thus, there exists a sequence $(x_n)_{n\in\mathbb{N}} \in (\mathbb{R}_+^d \backslash \Lambda_L)^{\mathbb{N}}$ such that $x = \lim_{n\to\infty} x_n$, so $x \in \partial\Lambda_L$. □

4.2 Non-rational density

In this subsection, we prove the following proposition. With the computation of boundary done in the previous subsection, it permits to algorithmically decide if the invariant density is a rational fraction.

Proposition 4.3. *Let a win-lose graph on two letters with vertices S. Then, the unique densities $(f_i)_{i\in S}$ are rational fractions if and only if the unique domains $(D_i)_{i\in S}$ are finite union of intervals up to a set of zero Lebesgue measure.*

The remaining of this subsection is devoted to the proof of this proposition.

First of all, remark that the fully subtractive algorithm for $d = 2$ is convergent and auto-dual, thus it implies that domains D_i for any extension are unique.

Domains D_i are limit sets of rational languages $\mathcal{D}_i$, thus they are a countable union intervals $\bigcup_{w\in A} m_w\mathbb{R}_+^2$, up to a set of zero Lebesgue measure $\Lambda_A \cup \Lambda_B$ thank to the following.

Lemma 4.4. *Let L be a rational language over the alphabet $\{\mathbf{0},\mathbf{1}\}$. Then, there exists two rational languages A and B such that $L = A\{\mathbf{0},\mathbf{1}\}^* \cup B$, with $\lambda(\Lambda_B) = 0$ and $\lambda(\Lambda_A) = 0$. Moreover, A and B are computable.*

Proof. Consider the minimal automaton recognizing the language L. In this automaton, there is at most one state whose language is $\{\mathbf{0},\mathbf{1}\}^*$. If such state doesn't exists, then take $A = \emptyset$. Otherwise, remove every outgoing edge from such state, and set it as the unique final state. Then, the language recognized by this new automaton is A. We then obtain $B = \{\mathbf{0},\mathbf{1}\}^*\backslash A$. Obviously we have $L = A\{\mathbf{0},\mathbf{1}\}^* \cup B$, and A and B are rational. Then, if we consider the minimal automaton with sink state recognizing the language B, it gives a win-lose graph on two letters satisfying the Fougeron's criterion (see [Fougeron]), and the sink state is reachable from every other state, thus $\lambda(\Lambda_B) = 0$. The same argument shows that $\lambda(\Lambda_A) = 0$. □

Let $\mathcal{M}_i \subset M_2(\mathbb{R}_+)$ be a countable set such that $D_i = E \cup \bigcup_{m\in\mathcal{M}_i} m\mathbb{R}_+^2$, where $\lambda(E) = 0$. We can moreover assume that such projective intervals $m\mathbb{R}_+^2$, $m \in \mathcal{M}_i$, are pairwise disjoint (they intersect only at $(0, 0)$).

Furthermore, the win-lose graph on two letters is ergodic thanks to Fougeron's criterion, and the unique invariant density is

$$f_i(x) = \sum_{m\in\mathcal{M}_i} \frac{|\det(m)|}{\prod({}^t mx)}, \quad \forall i \in S.$$

For $z \in \mathbb{C}$, let $\varphi_i(z) = \sum_{m\in\mathcal{M}_i} \frac{|\det(m)|}{\prod({}^t m(1/2-z,1/2+z))}$. It is the unique holomorphic extension of the map $z \mapsto f_i(1/2 - z, 1/2 + z)$ to the whole $\mathbb{C}$ but a countable number of points (one for each end point of the projective intervals).

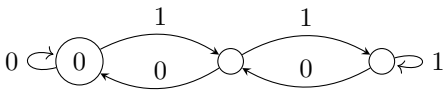
Then, the map f_i is a rational fraction if and only if φ_i has a finite number of poles, if and only if $\mathcal{M}_i$ is finite.

4.3 The algorithm for two letters win-lose graphs

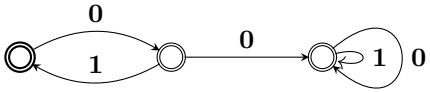
In this subsection, we present the algorithm to test if a win-lose graph on two letters has invariant densities that are rational fractions, and to compute it if it is the case. The algorithm is as follow:

- Compute domain languages $\mathcal{D}_i$. For each $\mathcal{D}_i$, do the following.
- Decompose $\mathcal{D}_i$ as $\mathcal{D}_i = A\{\mathbf{0},\mathbf{1}\}^* \cup B$, as in Lemma 4.4.
- Compute the language ∂L, defined in Proposition 4.2, describing the boundary of the limit set of the language $L = A\{0,1\}^*$.
- Compute non-trivial strongly connected components of the minimal automaton of ∂L.
- If there exists a component which is not a loop or which is not terminal, then we know by Proposition 4.3 that the invariant density f_i is not a rational fraction since D_i is an infinite union of disjoint intervals,
- Otherwise, we can decompose $\partial L = \mathrm{Pref}(u_1 v_1^* \cup ... \cup u_{2N} v_{2N}^*)$, where v_1, ..., v_{2N} are labels of loops, and $\partial\Lambda_L$ is the set of quadratic half lines $m_u V_v \mathbb{R}_+$, where m_u is the product of matrices of u, and where V_v is such that $\mathbb{R}_+ V_v = \lim_{n\to\infty} m_v^n \mathbb{R}_+^2$. Then, we order these vectors for the relation $(x,y) \leq (a,b) \Leftrightarrow xb \leq ya$. We get a finite increasing sequence of vectors V_1, ..., V_{2N}. For every $k \in \{1,...,N\}$, let m_k be the matrix with columns V_{2k-1} and V_{2k}. Then, the domain D_i is equal to the union $m_1\mathbb{R}_+^2 \cup ... \cup m_N \mathbb{R}_+^2$ up to a set of Lebesgue measure zero. Thus, we deduce that the density at state i is $f_i(x) = \sum_{k=1}^{N} \frac{|\det(m_k)|}{(V_{2k-1}|x)(V_{2k}|x)}$.

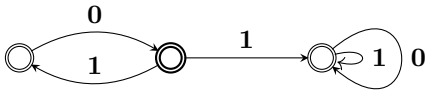
Example 4.5. *Consider the win-lose graph*



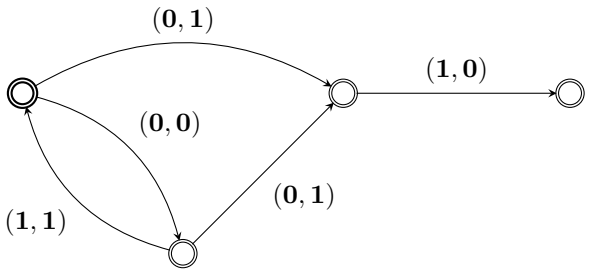
The domain of the state 0 *is the limit set of the language* $\mathcal{D}_0$ *of the automaton*



The decomposition of Lemma 4.4 gives $B = \emptyset$*, thus we compute the language* ∂L *for* $L = \mathcal{D}_0$*. The language* $\mathrm{Pref}(L^c)$ *is recognized by the automaton*



Then, the language $L \times \mathrm{Pref}(L^c) \cap L^{\mathrm{rel}}$ *is recognized by*



If we project on first coordinate then we get the language $\mathrm{Pref}((\mathbf{01})^*\mathbf{001})$. *Then, after* prune^∞ *we get the language* $\mathrm{Pref}((\mathbf{01})^*)$. *The languages* L^{min} *and* L^{max} *are respectively* $\mathbf{0}^*$ *and* $\mathrm{Pref}((\mathbf{01})^*)$, *thus* $\partial L = \mathbf{0}^* \cup \mathrm{Pref}((\mathbf{01})^*)$. *We deduce that the boundary of* Λ_L *is* $\mathbb{R}_+\{(1,0),(\varphi,1)\}$, *where* φ *is the golden number. Indeed, we have* $\mathbf{0}^n\mathbb{R}_+^2 \xrightarrow{n\to\infty} \mathbb{R}_+(1,0)$ *and* $(\mathbf{01})^n\mathbb{R}_+^2 \xrightarrow{n\to\infty} \mathbb{R}_+(\varphi,1)$. *We obtain that the domain of state* 0 *of the win-lose graph is the projective interval* $\begin{pmatrix} 1 & \varphi \\ 0 & 1 \end{pmatrix} \mathbb{R}_+^2$. *Thus, the invariant density of state* 0 *is* $f_0(x,y) = \frac{1}{x(\varphi x+y)}$.

5 Examples

In this section, we apply our algorithms to classical continued fraction algorithms, and extensions of it.

5.1 Cassaigne

The Cassaigne continued fraction algorithm is described by a matrices graph with a single state and with matrices

$$\left\{ \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \right\}.$$

We found that $\begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \mathbb{R}_+^3$ is a domain, thus an invariant density is

$$\frac{1}{(x_0+x_1+x_2)(x_0+x_1)(x_1+x_2)}.$$

The Cassaigne continued fraction algorithm can be slowed down to the win-lose graph of figure 1, with initial state 1. The invariant densities for this

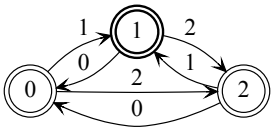


Figure 1: Win-lose graph for the Cassaigne algorithm

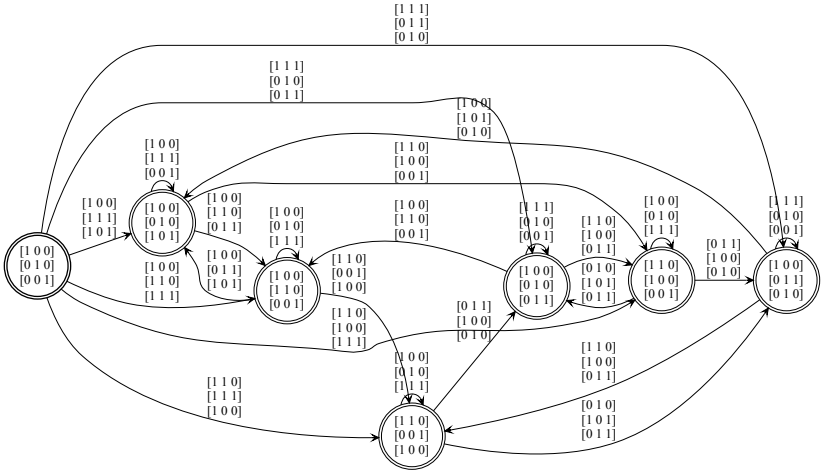


Figure 2: Matrices graph describing the Brun algorithm for $d = 3$

win-lose graph are

$$f_0(x_0, x_1, x_2) = \frac{1}{(x_0 + x_1 + x_2)(x_0 + x_1)(x_0 + x_2)},$$
$$f_1(x_0, x_1, x_2) = \frac{1}{(x_0 + x_1 + x_2)(x_0 + x_1)(x_1 + x_2)},$$
$$f_2(x_0, x_1, x_2) = \frac{1}{(x_0 + x_1 + x_2)(x_0 + x_2)(x_1 + x_2)}.$$

Thanks to Fougeron's criterion (see [Fougeron]), we can check that this algorithm is ergodic. This algorithm is almost auto-dual: in restriction to the domain, the dual is the Cassaigne's algorithm, up to permutation.

5.2 Brun

The Brun continued fraction algorithm subtract the second greatest coordinate to the greatest one. This is not directly described by a matrices graph but we can convert it to a matrices graph thanks to the algorithm described in section 3. For $d = 3$, we obtain the matrices graph of figure 2.

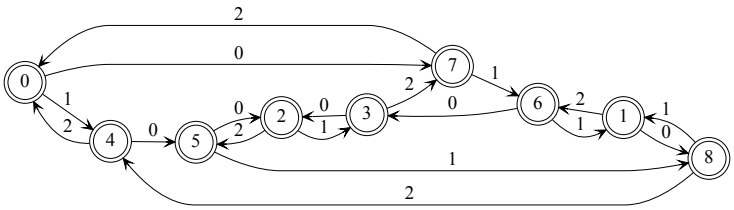


Figure 3: Win-lose graph of the Brun algorithm for $d = 3$

Then, we can compute the domains. For $d = 3$, we get

$$\emptyset, \begin{pmatrix} 1 & 1 & 2 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \mathbb{R}_+^3, \begin{pmatrix} 1 & 1 & 2 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \mathbb{R}_+^3, \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 1 & 1 \end{pmatrix} \mathbb{R}_+^3, \begin{pmatrix} 1 & 1 & 2 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \mathbb{R}_+^3,$$
$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 1 & 1 \end{pmatrix} \mathbb{R}_+^3, \begin{pmatrix} 1 & 1 & 2 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \mathbb{R}_+^3,$$

thus, we get the densities for the matrices graph

$$0, \frac{1}{(2\,x_0 + x_1 + x_2)(x_0 + x_1 + x_2)(x_0 + x_1)}, \frac{1}{(2\,x_0 + x_1 + x_2)(x_0 + x_1 + x_2)(x_0 + x_1)},$$
$$\frac{1}{(x_0 + 2\,x_1 + x_2)(x_0 + x_1 + x_2)(x_1 + x_2)}, \frac{1}{(2\,x_0 + x_1 + x_2)(x_0 + x_1 + x_2)(x_0 + x_1)},$$
$$\frac{1}{(x_0 + 2\,x_1 + x_2)(x_0 + x_1 + x_2)(x_1 + x_2)}, \frac{1}{(2\,x_0 + x_1 + x_2)(x_0 + x_1 + x_2)(x_0 + x_2)}.$$

We deduce the invariant density for the original algorithm:

$$f(x) = \sum_{m \in \mathcal{M},\ x \in m\mathbb{R}_+^d} f_m(m^{-1}x),$$

where $\mathcal{M} \subset M_d(\mathbb{R})$ is the set of states of the matrices graph. For $d = 3$, and for $x_0 < x_1 < x_2$, we get

$$f(x_0, x_1, x_2) = \frac{1}{(x_0 + x_2)x_1x_2}.$$

The matrices graph can be decomposed to a win-lose graph. The figure 3 shows the strongly connected component of this graph for $d = 3$. Thanks to Fougeron's criterion, we can check that this algorithm is ergodic for every d (see [Fougeron]).

5.3 Poincare

The Poincaré algorithm subtract the second greatest coordinate to the greatest, the third greatest to the second, etc... For $d = 3$, it is defined by the matrices

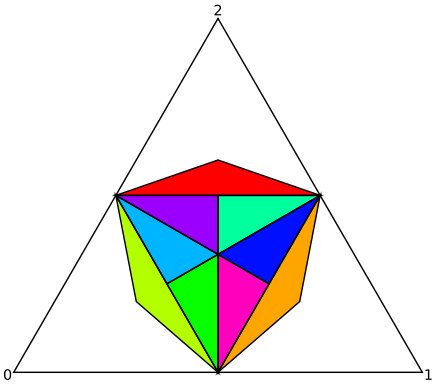


Figure 4: Domains of the win-lose graph of the Brun algorithm for $d = 3$

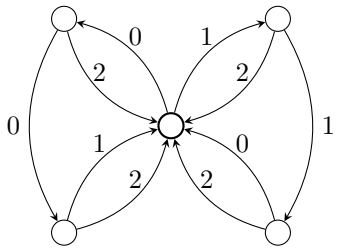


Figure 5: Poincaré algorithm as a win-lose graph for $d = 3$

graph with one state and with matrices

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

Since the set of matrices is stable by transposition, the algorithm is auto-dual, and the full positive cone $\mathbb{R}_+^d$ is a domain, thus $\frac{1}{\prod x}$ is an invariant density.

For $d = 3$, the algorithm can be described by the win-lose graph of Figure 5, and we can check that it is not ergodic and not convergent (see [Nogueira]). For $d = 4$, it can be described by a win-lose graph with 20 states, but it doesn't satisfy the Fougeron's criterion, and it is an open question to determine if it is ergodic.

5.4 Reverse

The reverse algorithm is defined as the Arnoux-Rauzy's one if one coordinate is greater than the sum of the others, and it sends the remaining center in the whole positive cone. It is given by the matrices graph with one state and

matrices

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}.$$

A domain is $\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \mathbb{R}^3_+$, thus an invariant density is

$$f(x_0, x_1, x_2) = \frac{2}{(x_0 + x_1)(x_0 + x_2)(x_1 + x_2)}.$$

This algorithm is almost auto-dual: in restriction to the domain, the dual is the reverse algorithm.

5.5 Fully subtractive

The fully subtractive algorithm subtract the smallest coordinate to every other ones. It is described by the win-lose graph with one state and d letters.

The unique domain is the Rauzy gasket for $d = 3$ and generalization of it if $d \geq 4$. It has zero-Lebesgue measure (see [AHS] for more details for $d = 3$), thus we cannot find an invariant density by this method. For $d \geq 3$, this algorithm is not ergodic nor convergent. The dual of this algorithm is the Arnoux-Rauzy's one, that subtract to the greatest coordinate the sum of the others.

5.6 Jacobi-Perron

The Jacobi-Perron continued fraction algorithm subtract as many times as possible the smallest coordinate to the other ones.

For $d = 3$, we can describe a slowed down version of this algorithm by a matrices graph with 4 states whose main strongly connected component is depicted in figure 6. And we can decompose this strongly connected component to the win-lose graph of Figure 7. Thanks to Fougeron's criterion, we can check that this algorithm is ergodic for $d = 3$.

The invariant density for this algorithm is unknown. The domain is fractal, and we don't know if it has zero Lebesgue measure. See Figure 8 for an approximation of the domains of the win-lose graph.

5.7 Arnoux-Rauzy-Poincaré

The Arnoux-Rauzy-Poincaré continued fraction algorithm is a combination of Arnoux-Rauzy and Poincaré's one. We apply the Arnoux-Rauzy algorithm if possible (i.e. we subtract to the greatest coordinate the sum of the others), otherwise we apply the Poincaré's one (i.e. we subtract to the second greatest coordinate the smallest, and to the greatest the second greatest).

The Arnoux-Rauzy-Poincaré algorithm can be represented by a matrices graph, thanks to the algorithm of section 3. We get the graph of figure 9.

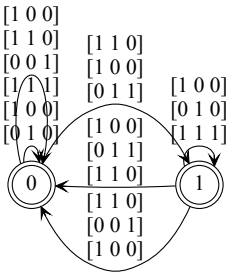


Figure 6: A matrices graph for the Jacobi-Perron algorithm for $d = 3$

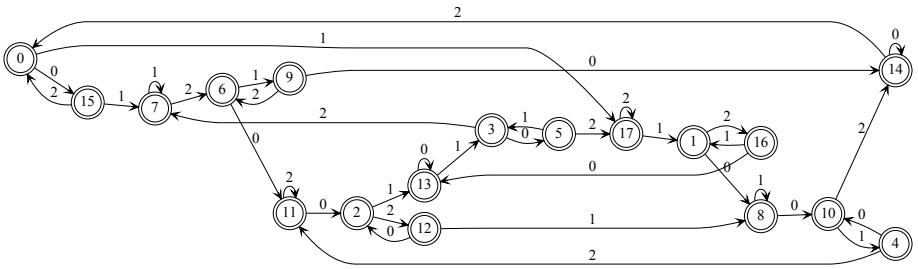


Figure 7: Win-lose graph for the Jacobi-Perron algorithm for $d = 3$

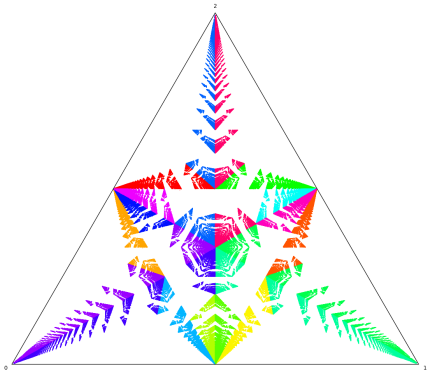


Figure 8: Approximation of domains of the win-lose graph for the Jacobi-Perron algorithm for $d = 3$

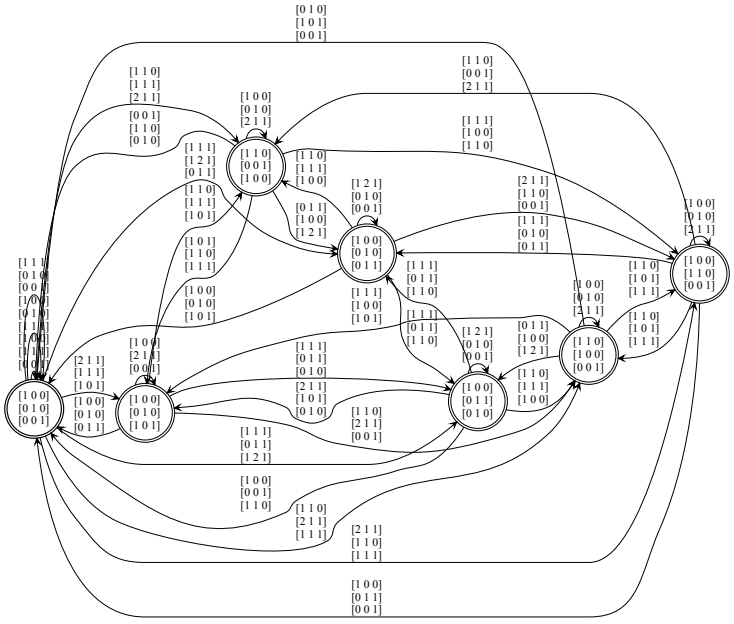
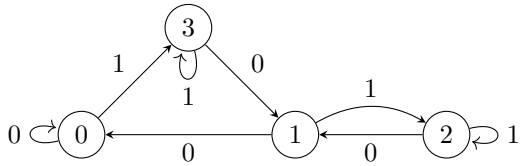


Figure 9: Matrices graph for the Arnoux-Rauzy-Poincaré algorithm

The invariant density is unknown for this algorithm. The domains are fractal, and we don't know if they have non-zero Lebesgue measure.

5.8 Two letters win-lose graph with non-rational density

The win-lose graph



have domain languages $\mathcal{D}_0 = \mathbf{0}^2\{\mathbf{0},\mathbf{1}\}^*$, $\mathcal{D}_1 = \mathbf{01}\{\mathbf{0},\mathbf{1}\}^*$, $\mathcal{D}_2 = \mathbf{1}^+\mathbf{01}\{\mathbf{0},\mathbf{1}\}^*$ and $\mathcal{D}_3 = \mathbf{1}^+\mathbf{0}^2\{\mathbf{0},\mathbf{1}\}^*$. Thus, domains are $D_0 = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix} \mathbb{R}_+^2$, $D_1 = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} \mathbb{R}_+^2$, $D_2 = \bigcup_{n\geq 1} \begin{pmatrix} 2 & 1 \\ 2n+1 & n+1 \end{pmatrix} \mathbb{R}_+^2$ and $D_3 = \bigcup_{n\geq 1} \begin{pmatrix} 1 & 2 \\ n & 2n+1 \end{pmatrix} \mathbb{R}_+^2$. And all

these interval are pairwise quasi-disjoint. We get that an invariant density is

$$\begin{aligned} f_0(x,y) &= \frac{1}{x(2x+y)}, \quad f_1(x,y) = \frac{1}{(2x+y)(x+y)}, \\ f_2(x,y) &= \sum_{n\geq 1} \frac{1}{(2x+(2n+1)y)(x+(n+1)y)}, \\ f_3(x,y) &= \sum_{n\geq 1} \frac{1}{(x+ny)(2x+(2n+1)y)}. \end{aligned}$$

By Proposition 4.3, f_2 and f_3 are not rational fractions.

5.9 Other examples

More examples can be found here: `http://www.i2m.univ-amu.fr/perso/paul.mercat/ComputeInvariantDensities.html`.

And many other examples can be easily tested since algorithms described in this article are implemented in a package for the Sage maths software (see `https://www.sagemath.org/`). The package is freely available here: `https://gitlab.com/mercatp/badic`. It can be installed with the command line

```
$ sage -pip install git+https://gitlab.com/mercatp/badic
```

6 Construction of extensions from sets of quadratic numbers

In this section, we present an algorithm that inputs a finite set of quadratics numbers, and that outputs a win-lose graph on two letters with an invariant density where the quadratic numbers appear. We denote $\mathbf{0} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ and $\mathbf{1} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$ to lighten the notations.

The algorithm is as follow:

- If the number of elements of the set is odd, add or remove the number 0 to the set. Change signs in order to have non negative numbers.
- Compute the continued fraction expansion of $(x, 1)$ for each quadratic number x, for the fully subtractive algorithm on two letters. Each expansion is of the form uv^ω, where u and v are two finite words over the alphabet $\{\mathbf{0}, \mathbf{1}\}$.
- For each quadratic number x with expansion uv^ω, compute the rational language L_x of finite words less than uv^ω in lexicographical order. A deterministic automaton recognizing this language is easily computed by considering the minimal automaton recognizing the language uv^*, and

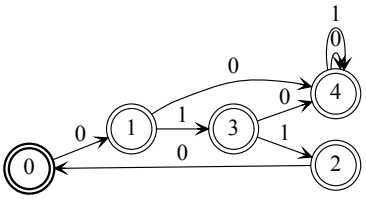


Figure 10: Automaton recognizing $L_{\sqrt{2}}$

then adding adding a new state s with edges $s \xrightarrow{\mathbf{0}} s$ and $s \xrightarrow{\mathbf{1}} s$, and adding edges $t \xrightarrow{\mathbf{0}} s$ for each state t that have no outgoing edge labeled by $\mathbf{0}$.

- If $x_0 < x_1 < ... < x_{2n+1}$ are the ordered quadratic numbers, compute the language
$$L = \bigcup_{i \in \{0,...,n\}} L_{x_{2i}} \cap L^c_{x_{2i+1}}.$$
- Take the mirror L^{mirror} (i.e. the language of words of L in reverse order).
- Compute a deterministic automaton recognizing this mirror L^{mirror}. This deterministic automaton gives a win-lose graph on two letters.

Example 6.1. *Consider the set $\{0, \sqrt{2}\}$. The expansion of $(0,1)$ is $\mathbf{1}^\omega$ and the expansion of $(\sqrt{2},1)$ is $(\mathbf{0110})^\omega$. Then, the rational language L_0 is $\{\mathbf{0},\mathbf{1}\}^*$, and the rational language $L_{\sqrt{2}}$ is recognized by the automaton of Figure 10. The mirror of the language $L = L_0 \cap L^c_{\sqrt{2}}$ is recognized by the deterministic automaton of Figure 11. It is a win-lose graph whose domains are projective intervals between points*

$$\{(0:1), (\sqrt{2}-1:1), (\sqrt{2}-1:2-\sqrt{2}), (\sqrt{2}:1), (\sqrt{2}-1:3-2\sqrt{2}), (1:0)\}.$$

Then, we easily deduce the invariant density, where $\sqrt{2}$ appears.

Remark 6.2. *More examples and an implementation of this algorithm can be found here: `http://www.i2m.univ-amu.fr/perso/paul.mercat/AlgoQuadraticSet.html` It is implemented in the Sage mathematical software using the badic package, see Subsection 5.9 for more details.*

Proposition 6.3. *The algorithm above give a win-lose graph whose domains are finite unions of projective intervals and $(x,1)$ is in the boundary, for every quadratic number x of the input.*

Proof. One way of computing the main component a minimal deterministic automaton recognizing the mirror of the language L is with the Algorithm 2. This is the part of the minimal automaton which is strongly connected and for which

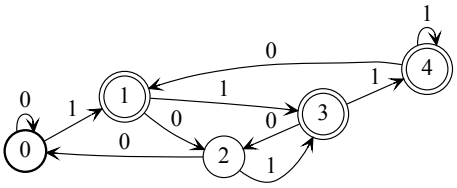


Figure 11: Automaton recognizing the language L^{mirror} of Example 6.1

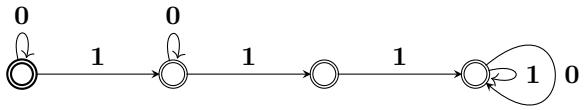
Data: A rational language L, over an alphabet A.
Result: A automaton without initial state.
$\mathcal{P}' \leftarrow \{L, L^c\}$;
do
  $R \leftarrow \emptyset$;
  $\mathcal{P} \leftarrow \mathcal{P}'$;
  $\mathcal{P}' \leftarrow \emptyset$;
  for $L \in \mathcal{P}$ **do**
    for $a \in A$ **do**
      Compute $a^{-1}L = \{u \in A^* \mid au \in L\}$;
      for $L' \in \mathcal{P}$ **do**
        $L'' \leftarrow L' \cap a^{-1}L$;
        Add L'' to $\mathcal{P}'$;
        Add $L'' \xrightarrow{a} L$ to R;
      end
    end
  end
while $\mathcal{P} \neq \mathcal{P}'$;
Return the automaton with set of edges R and final states $\{L' \in \mathcal{P} \mid L' \subseteq L\}$, after removing every state that are not co-reachable from every state;

Algorithm 2: Algorithm that computes the strongly connected component of the minimal deterministic automaton of the mirror

every state has a domain with non-empty interior (or in other words, states with non-zero invariant densities). Indeed, by construction, the automaton given by this Algorithm 2 before the last step of pruning has states $(L_i)_{i\in S}$, where L_i are rational languages whose disjoint union is $\{\mathbf{0},\mathbf{1}\}^*$, and such that $\forall u \in \{\mathbf{0},\mathbf{1}\}^*$, $\forall i,j,\ m_u L_i \subseteq L_j$ if and only if u is the label of a path from L_i to L_j. Thus we have $\forall i \in S,\ L_j = \biguplus_{L_i \xrightarrow{a} L_j} aL_i$. It gives that $\Lambda_{L_j} = \biguplus_{L_i \xrightarrow{a} L_j} a\Lambda_{L_i}$, thus $(\Lambda_{L_i})_{i\in S}$ are domains for the win-lose graph obtained by replacing $\mathbf{0}$ by 0 and $\mathbf{1}$ by 1. It is easily seen that if Λ_{L_i} has non-empty interior, then the state L_i is co-reachable from every other state, and it cannot reach states L_j such that Λ_{L_j} has empty interior. Thus, the pruning keep exactly states having non-zero invariant density. And we see that the given quadratic numbers appears in the density, since by construction if $\mathcal{F}$ is the set of final states returned by the algorithm, then $\bigcup_{L'\in\mathcal{F}} L' = L$. Thus, we have $\bigcup_{L'\in\mathcal{F}} D_{L'} = \Lambda_L$. And by construction Λ_L is an union of disjoints projective intervals whose end points are $\mathbb{R}_+(x,1)$, where x is the given set of quadratic numbers. Indeed for every $z \in \mathbb{R}$ the language L_z describe the projective interval $\{(x,y) \in \mathbb{R}_+^2 \mid x \geq zy\}$ between $\mathbb{R}_+(z,1)$ and $\mathbb{R}_+(1,0)$.

Now let us show that domains are finite union of projective intervals. The Algorithm 2 consists in refining the partition $L \cup L^c$ of $\{\mathbf{0},\mathbf{1}\}^*$, up to have a partition $\bigcup_{i\in S} L_i$ such that for every $i \in S$ and every letter a, $a^{-1}L_i$ is an union of $L'_k s$. It corresponds to refine the partition $\Lambda_L \cup \Lambda_{L^c} = \mathbb{R}_+^2$ up to have a quasi-partition $\bigcup_{i\in S} \Lambda_{L_i} = \mathbb{R}_+^2$ such that $F(\Lambda_{L_i})$ is an union of Λ_{L_k}'s. This refinement will give only finitely many intervals since quadratic numbers are ultimately periodic. □

Remark 6.4. *The construction proposed here works more generally for any rational language L over $\{\mathbf{0},\mathbf{1}\}$ such that Λ_L has non-empty interior. If we consider a deterministic automaton recognizing the mirror of L, it gives a win-lose graph such that the limit set of L is an union of domains. If for example we consider the automaton*



it has a limit set Λ_L with infinitely many accumulation points. The mirror of L give the win-lose graph of Figure 12, and we have $D_0 \cup D_1 \cup D_2 \cup D_3 = \Lambda_L$.

7 Greetings

I thank Charles Fougeron and Vincent Delecroix for interesting discussions. Without them, this article wouldn't exists.

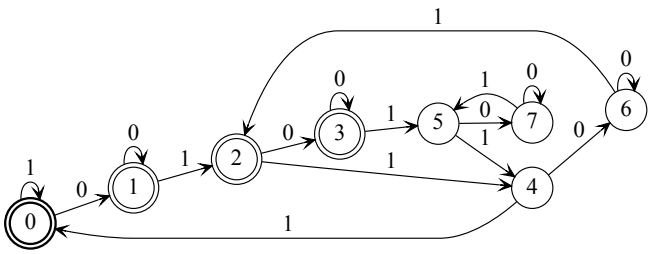


Figure 12: Win-lose graph with domains that have infinitely many accumulation points

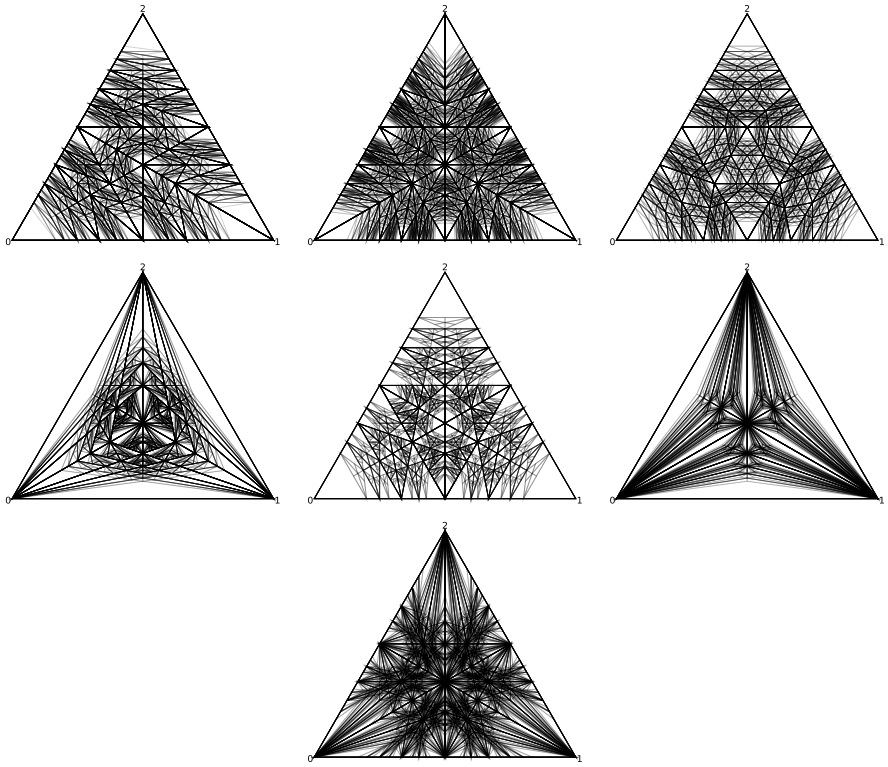


Figure 13: Some steps of Cassaigne, Brun, reverse, Jacobi-Perron, Arnoux-Rauzy-Poincaré, fully subtractive, and Poincaré's algorithms

References

[AHS] A. Avila, P. Hubert, A. Skripchenko *On the Hausdorff dimension of the Rauzy gasket*, Bull. SMF, 3 **154**, pp. 539-568, 2016.
`https://arxiv.org/abs/1311.5361`

[AL] P. Arnoux, S. Labbé *On some symmetric multidimensional continued fraction algorithms*, E.T.D.S, **38**, pp. 1601-1626, 2018.
`https://arxiv.org/abs/1508.07814`

[AN] P. Arnoux, A. Nogueira *Mesures de Gauss pour des algorithmes de fractions continues multidimensionnelles*, Ann. É.N.S., 4 **26**, no. 6, pp. 645-664, 1993.
`http://www.numdam.org/item/?id=ASENS_1993_4_26_6_645_0`

[AS] P. Arnoux, T. Schmidt *Natural extensions and Gauss measures for piecewise homographic continued fractions*, Bull. Soc. Math. France 147, no. 3, pp. 515–544, 2019.

[Carton] O. Carton *Langages formels, calculabilité et complexité*, isbn 978-2-7117-2077-4, Vuibert, 2008

[Fougeron] C. Fougeron *Dynamical properties of simplicial systems and continued fraction algorithms*, preprint, 2020.
`https://arxiv.org/abs/2001.01367`

[Nogueira] A. Nogueira *The three-dimensional Poincaré continued fraction algorithm*, Israël Journal of Math., 1995.