Une Conjecture sur les Mots

Michel Billaud (michel.billaud@laposte.net)

20 octobre 2022

Table des matières

- Les notes sont publiées dans https://www.mbillaud.fr/notes/
- Sources dans https://github.com/MichelBillaud/notes-diverses

Historique

- 2022-07-21 Version initiale

- 2022-10-20 Ajout présentation informelle
- 2023-02-26 Ajout message groupe `comp.theory` de 1993.

1 Introduction

1.1 Il y a très longtemps

Vers 1988, je m'étais posé un petit problème, que je n'ai pas réussi à résoudre. Rien d'étonnant, c'était pas dans mon domaine, et je n'y ai travaillé que mollement. Donc j'ai laissé tomber rapidement, mais après avoir appelé à l'aide sur Usenet, qui ne manque pas de gens plus compétents.

On en retrouve des traces dans le newsgroup `comp.theory` en 1993, https://groups.google.com/g/comp.theory/c/V_xDDtoR9a4/m/zgcM4We0CisJ mais j'avais dû déjà poser la question en 1988, puisque je me rappelle très bien (pour une fois) en avoir discuté avec Gilberto Filè dont je partageais le bureau, avant le STACS 1988 (j'ai une copie de son manuscrit, écrite à la main, quelque part !).

```
Michel BILLAUD
5 févr. 1993, 17:56:34


NOTATIONS. Let A={a,b,c...} be a finite alphabet, A* the set of words over
A, equipped with the concatenation "." (omitted unless necessary) and the
empty word denoted by 1.

Every f: A -> A* defines a morphism (also named f) from A* to A*,
the image of a word w being the concatenation of the images by f
of its letters.
For example, if f(a)=ab and f(b)=aa,
then f(abba)=f(a).f(b).f(b).f(a)=abaaaaab

For any letter x in A, delta_x is the morphism which deletes all
occurrences of the letter x, and keeps all others, that is:
delta_x (y) = 1 if x=y, otherwise y

DEFINITION. A word w is "simple" if for every morphism f
f(w)=w implies f(x)=x for each letter x in w

EXAMPLES of "simple" words : a,aa,aaa, abba, aabb, abcbac ...
non-simple words: ab, abab (take f(a)=ab and f(b)=1)
abcbca ( f(a)=a, f(b)=bc, f(c)=1 )

QUESTION. For every non-empty "simple" word w, does there always
exist a letter x in w such that delta_x(w) is also simple ?

EXAMPLE: for w=abcbac, take x=c, then delta_c(abcbac)=abba is simple.

Couldn't find a proof, or a counter-example... Help appreciated !
```

```
--
Michel BILLAUD : billaud@geocub.greco-prog.fr
Departement d'Informatique : phone W: 56.84.57.92 // 56.84.69.22
IUT "A", Universite Bordeaux I : "Personne n'est exempt de dire des sottises.
33405 Talence (FRANCE) : Le malheur est de les dire curieusement"
```

D'où la conjecture :

> Given a word w, if for each letter x occurring in w, there exists non-trivial morphism f_x such that the word obtained by erasing all the occurrences of x in x is a fixed point of f_x, then there exists a non-trivial morphism f such that w is a fixed point of f.

Et bon, surprise, on est en 2022, mais apparemment c'est toujours pas résolu. Et il y a des gens qui travaillent actuellement dessus.

1.2 Explication informelle

Bon, peut-être que c'est pas très clair, alors j'explique, d'abord informellement.

1. Si vous regardez la chaîne de caractères `s = "abab"`, et que vous substituez *simultanément*

- le caractère `'a'` par la chaîne vide `""`,
- le caractère `'b'` par `"ab"`

vous retrouvez la chaîne `s = "abab"`. La chaîne `s` est un point-fixe de la transformation.

2. Les transformations qu'on considère, c'est le remplacement d'un caractère par une chaîne. Elles correspondent à des morphismes sur les mots, par concaténation : on a $f(u).f(v) = f(u.v)$ pour tous mots u et v.

3. Il y en a de 3 types qui ont `"abab"` commme point fixe : les images de `'a'` et `'b'` sont respectivement

- soit `""` et `"ab"`
- soit inversement `"ab"` et `""`
- soit `"a"` et `"b"` (morphisme identité sur les lettres du mot).

4. On s'intéresse aux mots qio ne sont des points fixes **seulement par l'identité** identité. Exemples `"aaaa"`, `"abba"`, `"abcacb"`, … qu'on va appeler, pour faire court, mots “primitifs”.

5. Regardez `"abcacb"`, construit sur 3 lettres :

- Si j'efface le `c`, j'obtiens `abab` qui n'est pas primitif
- Si j'efface le `b`, j'obtiens `acac` qui n'est pas primitif
- mais si j'efface le `a`, j'ai `bccb` qui est primitif.

1.3 Voila la question

- si j'ai un mot primitif (construit sur au moins deux lettres différentes), est-ce qu'il contient toujours au moins une lettre que je peux effacer pour trouver un autre mot primitif ?

ou inversement

- tout mot primitif construit sur n lettres distinctes peut être obtenu en insérant (judicieusement) des occurrences d'une lettre supplémentaire dans un mot primitif à n-1 lettres.

1.4 Petits programmes pour vérifier

Ça me paraissait amusant, alors j'ai écrit des petits programmes vite fait pour

- décider si un mot est primitif ou non,
- trouver tous les mots primitifs d'une certaine taille (à un renommage des lettres près),
- chercher un contre-exemple à la propriété voulue.

1.5 Apparition de la conjecture

Et comme je n'en trouvais pas, je me suis dit que la question pouvait être une conjecture intéressants à démontrer, mais je n'ai pas su faire (programmer c'est cool, mais la combinatoire des mots n'est pas mon truc).

Donc j'ai demandé à des gens.

2 Quelques définitions

Reprenons proprement :

Je suppose que vous avez quand même entendu parler de mots construits sur un alphabet. Sinon, c'est pas compliqué

2.1 Mots

- un **alphabet** c'est un ensemble, donc les éléments sont appelés **lettres**. Exemple $A = \{a, b, c, d\}$.
- un **mot**, c'est une séquence de lettres, exemple $abcbad$.
- on appelle traditionnellement **langage** un ensemble de mots (sans restriction particulière).
- le mot vide, c'est la séquence vide, qu'on note souvent ϵ
- la **concaténation** de deux mots, c'est quand on les met bout à bout. On la note souvent par un point. Par exemple, $ab . cbad = abcbad".
- par commodité, on confond la notation d'une lettre avec le mot qui ne contient que cette lettre.

2.2 Morphisme

En général, un **morphisme** est une application d'un ensemble vers un autre qui préserve une opération.

Ici on considére les morphismes entre deux ensembles de mots (construits sur des alphabets différents ou pas) qui préservent la concaténation.

C'est-à-dire que si f est un morphisme, pour tous mots u, v on a $f(u.v) = f(u).f(v)$.

Une conséquence est qu'un morphisme est déterminé par la connaissance des images des lettres de l'alphabet. C'est évident : si on connait $f(a)$, $f(b)$ etc. alors on peut calculer l'image des mots formés à partir des lettres $a, b, \ldots$ etc.

La notion s'étend à des langages (ensembles de mots) : l'image d'un ensemble de mots par un morphisme, c'est l'ensemble des images par le morphisme.

2.3 Points-fixes d'un morphisme

On regarde maintenant les morphismes dans le cas où les ensembles de départ et d'arrivée sont les mêmes. Automorphismes ?

Un **point fixe du morphisme** f, c'est un mot w qui est sa propre image par f (c-a-d $f(w) = w$).

Exemple ab est un point fixe pour $f(a) = ab$ et $f(b) = \epsilon$.

Un morphisme n'admet pas forcément de point fixe (autre que le mot vide, évidemment), exemple évident : le morphisme qui double chaque lettre : $f(a) = aa, f(b) = bb, ; \ldots$.

Mais quand il en admet un, il en admet une infinité : si $f(w) = w$, alors $f(w^n) = w^n$.

3 Morphismes qui ont un mot comme point-fixe

On aurait pu s'amuser avec les propriétés de l'ensemble des points fixes d'un morphisme, mais on part dans la direction inverse :

- on prend un mot w au départ
- quels morphismes l'admettent comme point fixe ?

Ce qui amène quelques observations :

- il y en a toujours au moins un : le morphisme identité.
- si on en a un, tous les morphismes obtenus en faisant varier les images des lettres qui n'apparaissent pas dans le mot font aussi l'affaire.

Exemple : si on part du mot $w = aa$, on a $f(w) = w$ si et seulement si $f(a) = a$. Les valeurs de $f(b)$, $f(c)$ etc. sont indifférentes.

On va donc se concentrer sur les morphismes restreints aux lettres qui apparaissent dans le mot.

3.1 Mots primitifs : qui ne sont points fixes que par l'identité

Ce qui nous intéresse, c'est les mots qui ne sont leur propre image **que d'une seule et unique façon** : par l'identité.

Exemples : a, aa, a^n, $abba$, $ababa$, $(ab)^n a$, etc.

Contre-exemple : $abab$, en prenant $f(a) = ab$ et $f(b) = \epsilon$.

Il est bien évident qu'il y en a une infinité (cf. les exemples a^n).

On va les appeler “mots primitifs”.

3.2 La conjecture

Informellement, la question c'est

> tout mot primitif sur un alphabet à n lettres peut-il être obtenu à partir d'un mot primitif sur un alphabet à $n-1$ lettres, en y insérant judicieusement des occurrences de la lettre supplémentaire ?

Exemple

- *abba* : à partir de *aa* ou *bb*,
- *abcacb* : à partir de *bccb*, mais pas de *abab* ou *acac* qui ne sont pas primitifs.

Bref, posé dans l'autre sens :

> pour tout mot primitif w, existe-t-il au moins une lettre x dans w qu'on pourrait effacer pour retomber sur un mot primitif ?

Les énumérations par programme n'ont jusqu'ici pas trouvé de contre-exemple, et on conjecture que oui.

Mais la preuve reste à faire.

4 Bibliographie

Dans un ordre aléatoire, en cours de construction.

Disclaimer : je ne les ai pas tous lus, il en manque, et ils n'ont pas forcément tous un rapport direct. Mais c'est intéressant de voir pourquoi l'existence d'un morphisme entre deux mots est un problème NP-complet, ou d'apprendre qu'il existe un algo linéaire pour vois si un mot est “primitif” ou pas.

4.1 Richomme G, Levé F (2005)

Gwenaël Richomme, Florence Levé. On a conjecture about finite fixed points of morphisms. Theoretical Computer Science, Elsevier, 2005, 339, pp.103-128. (hal-00599741)

- Hal https://hal.archives-ouvertes.fr/hal-00599741
- https://www.sciencedirect.com/science/article/pii/S0304397505000277

Abstract : A conjecture of M. Billaud is : Given a word w, if, for each letter x occurring in w, the word obtained by erasing all the occurrences of x in w is a fixed point of a nontrivial morphism f_x, then w is also a fixed point of a nontrivial morphism. We prove that this conjecture is equivalent to a similar one on sets of words. Using this equivalence, we solve these conjectures in the particular case where each morphism f_x has only one expansive letter.

4.2 Łopaciuk, S., Reidenbach, D. (2022).

Łopaciuk, S., Reidenbach, D. (2022). The Billaud Conjecture for $|\Sigma| = 4$, and Beyond. In : Diekert, V., Volkov, M. (eds) Developments in Language Theory.

DLT 2022. Lecture Notes in Computer Science, vol 13257. Springer, Cham. https://doi.org/10.1007/978-3-031-05578-2_17

Abstract. The Billaud Conjecture, first stated in 1993, is a fundamental problem on finite words and their heirs, i.e., the words obtained by a projection deleting a single letter. The conjecture states that every morphically primitive word, i.e., a word which is not a fixed point of any non-identity morphism, has at least one morphically primitive heir. In this paper we give the proof of the Conjecture for alphabet size 4, and discuss the potential for generalising our reasoning to larger alphabets. We briefly discuss how other language-theoretic tools relate to the Conjecture, and their suitability for potential generalisations.

4.3 Łopaciuk, S., Reidenbach, D. (2021).

Łopaciuk, S., Reidenbach, D. (2021). On Billaud Words and Their Companions. In : Lecroq, T.,< Puzynina, S. (eds) Combinatorics on Words. WORDS 2021. Lecture Notes in Computer Science(), vol 12847. Springer, Cham. https://doi.org/10.1007/978-3-030-85088-3_11

The Billaud Conjecture, which has been open since 1993, is a fundamental problem on finite words w and their heirs, i.e., the words obtained by deleting every occurrence of a given letter from w. It posits that every morphically primitive word, i.e. a word which is a fixed point of the identity morphism only, has at least one morphically primitive heir. In this paper, we introduce and investigate the related class of so-called Billaud words, i.e. words whose all heirs are morphically imprimitive. We provide a characterisation of morphically imprimitive Billaud words, using a new concept. We show that there are two phenomena through which words can have morphically imprimitive heirs, and we highlight that only one of those occurs in morphically primitive words. Finally, we examine our concept further, use it to rephrase the Billaud Conjecture and study its difficulty.

- https://repository.lboro.ac.uk/articles/conference_contribution/On_Billaud_words_and_their_companions/14872188

4.4 Reidenbach D., Schneider J.C. (2009)

Daniel Reidenbach and Johannes C. Schneider. Morphically primitive words, Theor. Comp. Sci. 410 (2009) 2148-2161

Abstract. In the present paper, we introduce an alternative notion of the primitivity of words, that -unlike the standard understanding of this term -is not based on the power (and, hence, the concatenation) of words, but on morphisms. For any alphabet Σ, we call a word $w \in \Sigma^*$ *morphically imprimitive* provided that there are a shorter word v and morphisms $h, h' : \Sigma^* \to \Sigma^*$ satisfying $h(v) = w$ and $h'(w) = v$, and we say that w is *morphically primitive* otherwise. We explain why this is a well-chosen terminology, we demonstrate that morphic (im-) primitivity of words is a vital attribute in many combinatorial domains based on finite words and morphisms, and we study a number of fundamental properties of the concepts under consideration.

- https://www.sciencedirect.com/science/article/pii/S030439750900084X?via%3Dihub

4.5 Holub, Š. (2009)

Štěpán Holub, Polynomial algorithm for fixed points of nontrivial morphisms, Discrete Mathematics 309 (2009), 5069-5076

Abstract. A word w is a fixed point of a nontrivial morphism h if $w = h(w)$ and h is not the identity on the alphabet of w. The paper presents the first polynomial algorithm deciding whether a given finite word is such a fixed point. The algorithm also constructs the corresponding morphism, which has the smallest possible number of non-erased letters.

- https://www.sciencedirect.com/science/article/pii/S0012365X09001484?via%3Dihub

4.6 T. Kociumaka, J. Radoszewski, W. Rytter, T. Waleń, (2015)

T. Kociumaka, J. Radoszewski, W. Rytter, T. Waleń, Linear-time version of Holub's algorithm for morphic imprimitivity testing, Theoretical Computer Science, Volume 602, 2015, Pages 7-21, ISSN 0304-3975, https://doi.org/10.1016/j.tcs.2015.07.055

- https://www.sciencedirect.com/science/article/pii/S0304397515007112

Abstract : Stepan Holub (Discr. Math., 2009) gave the first polynomial-time algorithm deciding whether a given word is a nontrivial fixed point of a morphism. His algorithm works in quadratic time for large alphabets. We improve the algorithm to work in linear time. Our improvement starts with a careful choice of a subset of rules used in Holub's algorithm that is necessary to grant correctness of the algorithm. Afterwards we show how to choose the order of applying the rules that allows to avoid unnecessary operations on sets. We obtain linear time using efficient data structures for implementation of the rules. Holub's algorithm maintains connected components of a graph corresponding to specially marked positions in a word. This graph is of quadratic size for large alphabet. In our algorithm only a linear number of edges of this conceptual graph is processed. A preliminary version of this paper appeared at LATA 2013 conference.

4.7 Filè, G. (1989)

Filè, G. : The relation of two patterns with comparable languages patterns. RAIRO - Theoretical Informatics and Applications - Informatique Théorique et Applications 23(1), 45–57 (1989)

Gilberto Filé : The Relation of Two Patterns with Comparable Languages. STACS 1988 : 184-192