

Les claviers, un modèle de calcul

YOAN GERAN*
CORTO MASCLE[‡]

BASTIEN LABOUREIX[†]
VALENTIN D. RICHARD[§]

École Normale Supérieure Paris-Saclay

Février 2020

Résumé

Nous introduisons ici une nouvelle représentation de langages, les *claviers*. On se munit d'un ensemble d'opérations élémentaires (ajout/effacement d'une lettre, déplacement à droite/gauche,...), et on définit un clavier comme un ensemble de suites finies d'opérations élémentaires, appelées touches. Son langage va être l'ensemble des mots obtenus en appliquant une suite quelconque de touches. Contrairement à des modèles de calcul classiques comme les machines de Turing ou les grammaires, toutes les touches peuvent être appliquées à tout moment.

Nous nous intéressons ici à plusieurs problèmes fondamentaux sur ce modèle de calcul. En premier lieu nous définissons différentes classes de claviers en faisant varier l'ensemble des opérations élémentaires autorisées, et nous comparons l'expressivité des classes de langages obtenues entre elles. Nous comparons également ces classes à des classes de langages connues, notamment les langages réguliers, algébriques et contextuels.

Nous étudions également certains problèmes de décision découlant naturellement de l'étude des claviers. Les deux plus importants sont le *problème du mot*, consistant à tester si un clavier peut engendrer un mot donné, et le problème de l'universalité, consistant à tester si un clavier peut engendrer tout mot sur son alphabet.

Nous obtenons que toutes les classes étudiées sont différentes, et nous caractérisons les classes incluses dans les rationnels et les algébriques. Nous déduisons des inclusions de certaines classes dans des classes de langages connues la décidabilité de certains problèmes de décision évoqués plus haut.

*yoan.geran@ens-paris-saclay.fr

†bastien.laboureix@ens-paris-saclay.fr

‡corto.macle@ens-paris-saclay.fr

§valentin.richard@ens-paris-saclay.fr 

Il semble ainsi que l'expressivité des claviers soit orthogonale à celle de nombreux modèles connus. En effet, les claviers sont limités par leur absence de mémoire, mais cette faiblesse peut être compensée par l'ajout d'opérations élémentaires comme l'effacement ou les déplacements, difficiles à simuler dans des modèles comme les automates sans gagner l'expressivité des machines de Turing.

Table des matières

Introduction	4
1. Définitions et notations	5
1.1. Mots, langages et automates — Rappels	5
1.2. Définitions des claviers	6
1.2.1. Opérations élémentaires	6
1.2.2. Touches	8
1.2.3. Claviers automatiques	9
1.2.4. Claviers manuels	10
1.2.5. Classes de clavier	11
2. Propriétés des claviers	13
2.1. Généralités	13
2.2. Les claviers sans flèches	17
2.3. Les claviers sans flèche droite	18
2.4. Les claviers sans retour	19
3. Les langages de claviers	22
3.1. Les claviers sans flèches	22
3.1.1. MK	22
3.1.2. EK	22
3.1.3. RK	23
3.1.4. REK	24
3.2. Les claviers sans flèche droite	28
3.2.1. GK	28
3.2.2. GEK	28
3.2.3. GRK	29
3.2.4. GREK	29
3.3. Les claviers sans retour	31
3.3.1. FK	31
3.3.2. FEK	31
3.4. Étude de l'automatisme	32
3.4.1. FRK	32
3.4.2. FREK	33

Conclusion	35
A. Propriétés des claviers	37
A.1. Généralités	37
A.2. Les claviers sans flèches	44
A.3. Les claviers sans flèche droite	46
A.4. Les claviers sans retour	47
B. Les langages de claviers	49
B.1. Les langages sans flèches	49
B.1.1. RK	49
B.1.2. REK	49
B.2. Les langages sans flèche droite	62
B.2.1. GK	62
B.2.2. GEK	62
B.2.3. GRK	63
B.2.4. GREK	64
B.3. Les claviers sans retour	66
B.3.1. FK	66
B.3.2. FEK	67
B.4. Étude de l'automatisme	70
B.4.1. FREK	70

Introduction

On s'intéresse ici à la modélisation mathématique d'un clavier défectueux, où les différentes touches ne produisent pas forcément l'effet escompté. Par exemple, la touche `g` peut écrire le mot « rapide » et la touche `h` efface les trois caractères à gauche du curseur. Dans cette situation, lorsque l'utilisateur veut écrire le mot « rap », il lui suffit d'appuyer sur `g` puis sur `h`.

Mais peut-on tout écrire ? Peut-on écrire un mot donné ? Tout dépend de notre clavier ! Nous cherchons donc un algorithme de calcul de la suite de touche à taper pour obtenir un mot donné. Cela paraît relativement simple à première vue lorsque les seules opérations effectuées par les touches sont l'écriture de lettres. Nous avons ainsi ajouté le retour arrière et les flèches directionnelles, qui compliquent significativement le problème.

Nous nous sommes intéressés à la notion de langage de clavier, à savoir l'ensemble des mots qui peuvent être écrits à partir d'un ensemble fini de touches défectueuses. Les différentes classes de clavier (selon que l'on autorise ou non le retour arrière ou les flèches directionnelles) forment une hiérarchie de langages et les ajouts d'opérations augmentent strictement l'expressivité des claviers.

Par exemple, nous montrons dans ce document que les claviers sans flèche directionnelle ne peuvent exprimer que des langages rationnels. Les claviers où la flèche directionnelle droite est interdite (mais la flèche gauche est autorisée) sont quant à eux tous algébriques.

Il semble ainsi que l'expressivité des claviers soit orthogonale à celle de nombreux modèles connus. En effet, les claviers sont limités par leur absence de mémoire et leur incapacité à lire, mais cette faiblesse peut être compensée par l'ajout d'opérations élémentaires comme l'effacement ou les déplacements, difficiles à simuler dans des modèles classiques comme les automates.

Le modèle existant le plus proche à notre connaissance est celui des automates oublieux (*forgetting automata*), ainsi que définis par Jančar, Mráz et Plátek [3]. Ceux-ci autorisent des opérations similaires à celles que nous considérons pour les claviers, mais s'en distinguent car les claviers ne peuvent pas lire et ne possèdent pas d'états.

Nous commencerons par définir la sémantique des claviers via la notion d'opérations élémentaires et de configurations. Nous introduirons également deux types de clavier : manuels et automatiques. Nous donnerons ensuite quelques propriétés fondamentales des claviers, dans le cas général ou pour des classes plus restreintes. Enfin, nous parlerons de la hiérarchie des langages de clavier et montrerons qu'elle est stricte.

1. Définitions et notations

1.1. Mots, langages et automates — Rappels

Dans cette section, nous fixons les notations utilisées dans le document. On fixe un alphabet fini A .

Définition 1.1.1 (Mot)

Un mot $w = a_1 \dots a_n$ est une suite finie sur A . On note w_i ou $w[i]$ la i^e lettre de w . n est la longueur de w , que l'on notera $|w|$.

Définition 1.1.2

On note A^k l'ensemble des mots de longueur k , $A^{\leq k}$ ceux de longueurs inférieures ou égales à k et A^* l'ensemble de tous les mots.

$$A^{\leq k} = \bigcup_{0 \leq i \leq k} A^i \quad \text{et} \quad A^* = \bigcup_{k \in \mathbb{N}} A^k.$$

On fixe A un alphabet fini. Pour tout $k \in \mathbb{N}$, on note A^k l'ensemble des mots de longueur k ,

$$A^{\leq k} = \bigcup_{0 \leq i \leq k} A^i \quad \text{et} \quad A^* = \bigcup_{k \in \mathbb{N}} A^k.$$

Étant donné un mot $w = a_1 \dots a_n$, on notera w_i ou $w[i]$ la i^e lettre de w , à savoir a_i . La longueur de w est notée $|w|$ et le nombre d'occurrences de la lettre a dans w est notée $|w|_a$. On désigne par \tilde{w} le miroir de w . Étant donnés deux indices $1 \leq i \leq j \leq n$, on note $w[i, j]$ le mot $w_i \dots w_j$. Si $i > j$ alors $w[i, j]$ désigne le mot vide.

On note $\text{Pref}(w)$ l'ensemble des préfixes de w , $\text{Suff}(w)$ l'ensemble de ses suffixes et $\text{Fact}(w)$ l'ensemble de ses facteurs. Un sous-mot de w est un mot de la forme $w_{i_1} \dots w_{i_k}$ avec i_1, \dots, i_k une suite strictement croissante d'éléments de $\{1, \dots, |w|\}$, et on note $\text{Sumo}(w)$ l'ensemble des sous-mots de w .

On représentera un automate fini sur A par un tuple $(Q, \Delta, \text{Init}, \text{Fin})$ avec Q un ensemble fini d'états, $\Delta : Q \times A \rightarrow 2^Q$ une fonction de transition, Init et Fin des ensembles d'états initiaux et finaux. On étend Δ à $Q \times A^*$ en posant, pour tout $q \in Q$, $w \in A^*$ et $a \in A$, $\Delta(q, \varepsilon) = \{q\}$ et $\Delta(q, wa) = \Delta(\Delta(q, w), a)$. Dans la suite on pourra autoriser les ε -transitions pour simplifier les preuves.

On représentera un automate à pile sur A par un tuple $(Q, \Gamma, \perp, \Delta, \text{Init}, \text{Fin})$ avec

- Q un ensemble fini d'états ;
- Γ un alphabet de pile fini ;
- $\perp \in \Gamma$ un symbole de fond de pile ;
- $\Delta: Q \times A \times (\Gamma \cup \{-\})^2 \rightarrow 2^Q$ une fonction de transition ;
- Init et Fin des ensembles d'états initiaux et finaux.

L'acceptation se fait par état final et pile vide et les transitions de Δ sont sous la forme

$$s_1 \xrightarrow[\text{op}_1, \text{op}_2]{a} s_2$$

avec

- $\text{op}_1 = \downarrow \gamma$ si on empile un symbole $\gamma \in \Gamma$, et $\text{op}_1 = -$ si on n'empile rien.
- $\text{op}_2 = \uparrow \gamma$ si on dépile un symbole $\gamma \in \Gamma$, et $\text{op}_2 = -$ si on ne dépile rien.

1.2. Définitions des claviers

Notre objectif est de modéliser l'écriture d'un mot à partir d'une liste d'actions écrivant des symboles. Les caractères de A sont les lettres que nous pourrions écrire. Nous considérons également les symboles spéciaux suivants :

- le retour arrière \leftarrow ;
- La flèche gauche \blacktriangleleft ;
- La flèche droite \blacktriangleright .

Nous notons alors S l'ensemble de tous les symboles possibles ;

$$S \triangleq A \cup \{\leftarrow, \blacktriangleleft, \blacktriangleright\}.$$

Définition 1.2.1 (Configuration)

Nous appelons configuration un élément (u, v) de $A^* \times A^*$. Nous notons $C = A^* \times A^*$ l'ensemble des configurations.

Remarque 1.2.2

Intuitivement, être dans la configuration (u, v) correspond à avoir uv écrit sur la zone de texte et avoir le curseur placé entre u et v . Nous noterons les configurations $\langle u|v \rangle$ (où $|$ représente le curseur).

1.2.1. Opérations élémentaires

Définition 1.2.3 (Opération élémentaire)

Nous appelons opération élémentaire sur S un élément de S .

Définition 1.2.4 (Action)

L'action $\langle u|v \rangle \cdot s$ d'une opération élémentaire s sur une configuration $\langle u|v \rangle$ est définie de la manière suivante.

$$\begin{aligned} \langle u|v \rangle \cdot a &= \langle ua|v \rangle \text{ si } a \in A. \\ \langle \varepsilon|v \rangle \cdot \leftarrow &= \langle \varepsilon|v \rangle \quad \text{et} \quad \langle u'a|v \rangle \cdot \leftarrow = \langle u'|v \rangle \\ \langle \varepsilon|v \rangle \cdot \blacktriangleleft &= \langle \varepsilon|v \rangle \quad \text{et} \quad \langle u'a|v \rangle \cdot \blacktriangleleft = \langle u'|av \rangle \\ \langle u|\varepsilon \rangle \cdot \blacktriangleright &= \langle u|\varepsilon \rangle \quad \text{et} \quad \langle u|av' \rangle \cdot \blacktriangleright = \langle ua|v' \rangle \end{aligned}$$

Soit $s \in S$. Nous noterons $\langle u|v \rangle \xrightarrow{s} \langle u'|v' \rangle$ pour $\langle u|v \rangle \cdot s = \langle u'|v' \rangle$.

Remarque 1.2.5

Une opération élémentaire correspond à une action de base. Ainsi, une touche de A écrit une lettre de A , \blacktriangleleft et \blacktriangleright permettent de se déplacer, et \leftarrow permet de supprimer une lettre.

Remarque 1.2.6

Les touches \leftarrow et \blacktriangleleft ne font rien dans une configuration de la forme $\langle \varepsilon|v \rangle$ (quand le curseur est sur le bord gauche du mot). De même, la touche \blacktriangleright ne fait rien dans une configuration de la forme $\langle u|\varepsilon \rangle$ (quand le curseur est sur le bord droit du mot).

Exemple 1.2.7

La suite d'opérations élémentaires $\leftarrow, a, \blacktriangleright, \blacktriangleright, b$, permet de passer de la configuration $\langle c|d \rangle$ à la configuration $\langle adb|\varepsilon \rangle$.

$$\begin{aligned} \langle c|d \rangle &\xrightarrow{\leftarrow} \langle \varepsilon|d \rangle \\ &\xrightarrow{a} \langle a|d \rangle \\ &\xrightarrow{\blacktriangleright} \langle ad|\varepsilon \rangle \\ &\xrightarrow{\blacktriangleright} \langle ad|\varepsilon \rangle \\ &\xrightarrow{b} \langle adb|\varepsilon \rangle. \end{aligned}$$

Cette même suite d'opérations élémentaires permet de passer de $\langle c|de \rangle$ à $\langle adeb|\varepsilon \rangle$, avec cette fois-ci deux lettres entre a et b .

Remarque 1.2.8

Dans la suite, nous confondrons allègrement « action » et « opération élémentaire ».

Définition 1.2.9 (Sémantique efficiente)

Une sémantique alternative des opérations élémentaire, que nous appellerons *sémantique efficiente*, consiste à interdire l'application d'un retour arrière ou d'une flèche gauche sur le bord gauche d'un mot, ou d'une flèche droite sur le bord droit.

Formellement, on définit cette fois les opérations élémentaires comme des relations de

réécriture sur les configurations. Pour tous $u, v \in A^*$, $a \in A$ on a :

$$\begin{aligned} \langle u|v \rangle &\xrightarrow{a}_e \langle ua|v \rangle \\ \langle u'a|v \rangle &\xleftrightarrow{a}_e \langle u'|v \rangle \\ \langle u'a|v \rangle &\xrightarrow{\blacktriangleleft}_e \langle u'|av \rangle \\ \langle u|av' \rangle &\xrightarrow{\blacktriangleright}_e \langle ua|v' \rangle \end{aligned}$$

On définit alors \odot par $\langle u|v \rangle \odot s = \langle u'|v' \rangle$ si et seulement si $\langle u|v \rangle \xrightarrow{s}_e \langle u'|v' \rangle$.

1.2.2. Touches

Définition 1.2.10 (Touche)

Une touche est un élément de S^* ; c'est un mot sur les opérations élémentaires (et donc nous pouvons également voir les touches comme des suites d'actions élémentaires).

Nous notons $\mathcal{T}(S)$ l'ensemble des touches sur S . Lorsque S est donné par le contexte, nous noterons simplement \mathcal{T} l'ensemble des touches.

Définition 1.2.11 (Action d'une touche)

L'action d'une touche de $\mathcal{T}(S)$ sur une configuration est définie inductivement par

$$\begin{cases} \langle u|v \rangle \cdot \varepsilon = \langle u|v \rangle \\ \langle u|v \rangle \cdot st = (\langle u|v \rangle \cdot s) \cdot t \end{cases}$$

Nous étendons également l'écriture $\langle u|v \rangle \xrightarrow{t} \langle u'|v' \rangle$ pour $\langle u|v \rangle \cdot t = \langle u'|v' \rangle$ aux touches.

Exemple 1.2.12

L'action de la touche $\leftarrow mi$ permet de passer de la configuration $\langle u|ne \rangle$ à la configuration $\langle mi|ne \rangle$.

Remarque 1.2.13

Appliquer une touche revient à appliquer l'action de chaque touche élémentaire la composant.

Remarque 1.2.14

Nous définissons de même l'action d'une touche pour la sémantique efficiente par $\langle u|v \rangle \odot \varepsilon = \langle u|v \rangle$ et $\langle u|v \rangle \odot st = \langle u_2|v_2 \rangle$ si et seulement s'il existe $\langle u_1|v_1 \rangle$ telle que

$$\langle u|v \rangle \odot s = \langle u_1|v_1 \rangle \wedge \langle u_1|v_1 \rangle \odot t = \langle u_2|v_2 \rangle.$$

Et bien sûr, nous avons là aussi la notation $\langle u|v \rangle \xrightarrow{t}_e \langle u_2|v_2 \rangle$.

Remarque 1.2.15

Dans la suite, nous considérerons parfois des suites de touches $\tau = t_1 \dots t_n \in \mathcal{T}^*$. L'action

de τ , en sémantique classique ou en sémantique efficiente, doit alors être vue comme l'action de la touche correspond à la concaténation des t_i . Cela correspond à appliquer chaque touche l'une après l'autre.

Définition 1.2.16 (Taille d'une touche et nombre d'occurrences)

Soit t une touche. La taille de t , notée $|t|$ est sa longueur en tant que mot.

De plus, pour $s \in S$, on note $|t|_s$ le nombre d'occurrences du symbole s dans t .

De même, la taille d'une configuration $\langle u|v \rangle$ est définie par $|\langle u|v \rangle| = |u| + |v|$.

1.2.3. Claviers automatiques

Définition 1.2.17 (Clavier automatique)

Soit S un ensemble de symboles. Un clavier automatique sur S est un sous-ensemble fini de $\mathcal{T}(S)$.

Définition 1.2.18 (Exécution d'un clavier automatique)

Soit K un clavier automatique et $c_0 = \langle u_0|v_0 \rangle$ une configuration dite initiale. Une exécution de K sur c_0 est une suite finie $\rho = ((t_1, c_1), \dots, (t_{n+1}, c_{n+1})) \in (K \times C)^{n+1}$ avec $n \in \mathbb{N}$ telle que

$$\forall i \in \llbracket 1; n+1 \rrbracket, c_{i-1} \xrightarrow{t_i} c_i.$$

Si ce n'est pas précisé, une exécution est sur la configuration initiale vide $\langle \varepsilon|\varepsilon \rangle$.

Définition 1.2.19 (Reconnaissance par un clavier automatique)

Soit K un clavier automatique et $w \in A^*$. On dit que w est reconnu par K s'il existe une exécution de A de dernière configuration $\langle u|v \rangle$ telle que $uv = w$.

Exemple 1.2.20

Soit $K_1 = \{aa\}$ et soit $n \in \mathbb{N}^*$. K_1 reconnaît le mot a^{2n} avec l'exécution

$$\langle \varepsilon|\varepsilon \rangle \xrightarrow{aa} \langle a^2|\varepsilon \rangle \xrightarrow{aa} \langle a^4|\varepsilon \rangle \dots \xrightarrow{aa} \langle a^{2n}|\varepsilon \rangle.$$

Remarque 1.2.21

Attention, on demande qu'une exécution soit non vide; il faut donc appuyer sur au moins une touche du clavier. En particulier, ε n'est pas toujours reconnu par un clavier automatique.

Définition 1.2.22 (Langage d'un clavier automatique)

Soit K un clavier automatique. On appelle langage de K , noté $\mathcal{L}(K)$, l'ensemble des mots w qui sont reconnus par K .

Exemple 1.2.23

On a $\mathcal{L}(K_1) = (aa)^+$.

1.2.4. Claviers manuels

À première vue, les claviers automatiques semblent un peu limités. Il semble par exemple impossible d'obtenir un clavier de langage $(aa)^*a$. En effet, nous aimerions écrire deux a (une touche **aa**) et une touche **a** qui serait utilisée à la fin... Mais nous n'avons pas de notion de fin de saisie.

Définition 1.2.24 (Clavier manuel)

Un clavier manuel sur S est un couple d'ensembles finis de touches $(T, F) \subset \mathcal{T}(S)^* \times \mathcal{T}(S)^*$. F est appelé ensemble des touches finales du clavier et T ensemble des touches transientes du clavier.

Définition 1.2.25 (Exécution acceptante d'un clavier manuel)

Soit $K = (T, F)$ un clavier manuel et $c_0 = \langle u_0 | v_0 \rangle$ une configuration dite initiale. Une exécution **acceptante** de K sur c_0 est une suite finie $\rho = ((t_1, c_1), \dots, (t_{n+1}, c_{n+1})) \in (K \times C)^{n+1}$ pour $n \in \mathbb{N}$ telle que

- pour tout $i \in \llbracket 1 ; n+1 \rrbracket$, $c_{i-1} \xrightarrow{t_i} c_i$;
- pour tout $i \in \llbracket 1 ; n \rrbracket$, $t_i \in T$;
- $t_{n+1} \in F$.

Si ce n'est pas précisé, une exécution est sur la configuration initiale vide $\langle \varepsilon | \varepsilon \rangle$.

On parlera d'exécution dans le cas général lorsque $t_{n+1} \in T \cup F$.

Remarque 1.2.26

Avec les claviers manuels, nous avons rajouté la notion de fin de saisie à travers les touches acceptantes qui sont les touches qui mènent à une fin de saisie après avoir effectué leur action.

Définition 1.2.27

Nous définissons la reconnaissance et les langages pour les claviers manuels de la même manière que plus tôt, pour les claviers automatiques.

Exemple 1.2.28

Nous reconnaissons facilement le langage $(aa)^*a$ avec le clavier manuel $K_2 = (\{\mathbf{aa}\}, \{\mathbf{a}\})$.

Théorème 1.2.29 (Simulation)

Soit K_a un clavier automatique. Alors il existe un clavier manuel qui reconnaît le même langage que K_a .

Preuve

Le clavier $K_m = (K_a, K_a)$ reconnaît le même langage que K_a . □

Ce théorème nous permet de définir les claviers généraux en se basant uniquement sur les claviers manuels.

Remarque 1.2.30

Une touche acceptante peut être vue comme une touche se terminant par le symbole

« Entrée » noté \blacksquare . Ceci nous permet de noter les claviers en utilisant un seul ensemble. Par exemple, $\{aa, a\blacksquare\}$ correspond au clavier K_2 vu en [exemple 1.2.28](#).

L'entrée peut alors être vue comme un symbole qui n'apparaît qu'à la fin des touches et qui permet de finir la saisie. On appelle alors les claviers automatiques des claviers « sans entrée », dans le sens où il n'y a pas besoin du symbole \blacksquare pour être accepté.

Définition 1.2.31 (Clavier)

Nous appelons clavier un clavier manuel. Nous notons \mathbf{Clav} l'ensemble des claviers.

Définition 1.2.32 (Taille d'un clavier)

Soit $K = (T, F)$ un clavier. Nous définissons la taille de K notée $|K|$ par

$$|K| = \sum_{t \in T} (|t| + 1) + \sum_{t \in F} (|t| + 1)$$

et nous définissons sa norme infinie par

$$\|K\|_{\infty} = \max_{k \in T \cup F} |k|.$$

1.2.5. Classes de clavier

Nous avons plusieurs symboles spéciaux qui nous permettent d'avoir des claviers avec (potentiellement) plus d'expressivité. Il paraît raisonnable de s'intéresser aux langages que l'on peut reconnaître avec des claviers où certains de ces symboles ne sont pas disponibles.

Définition 1.2.33 (Clavier minimal)

On appelle clavier minimal un clavier ne contenant aucun symbole spécial. Il s'agit donc d'un clavier automatique sans \blacktriangleright , \blacktriangleleft et \leftarrow . Nous notons \mathbf{MK} l'ensemble des claviers minimaux.

Remarque 1.2.34

Nous construisons nos autres classes en partant des claviers minimaux auxquels on rajoute certaines possibilités de symboles spéciaux. Les noms des classes sont obtenus en rajoutant R (pour \leftarrow), E (pour \blacksquare , qu'on peut considérer, rappelons-le, comme un symbole), G (pour \blacktriangleleft) et F (pour \blacktriangleleft et \blacktriangleright) à K. Ainsi, en fonction des symboles spéciaux autorisés, on a ces différentes classes.

MK : $\{\}$	GK : $\{\blacktriangleleft\}$	FK : $\{\blacktriangleleft, \blacktriangleright\}$
EK : $\{\blacksquare\}$	GEK : $\{\blacktriangleleft, \blacksquare\}$	FEK : $\{\blacktriangleleft, \blacktriangleright, \blacksquare\}$
RK : $\{\leftarrow\}$	GRK : $\{\blacktriangleleft, \leftarrow\}$	FRK : $\{\blacktriangleleft, \blacktriangleright, \leftarrow\}$
REK : $\{\leftarrow, \blacksquare\}$	GREK : $\{\blacktriangleleft, \leftarrow, \blacksquare\}$	FREK : $\{\blacktriangleleft, \blacktriangleright, \leftarrow, \blacksquare\}$

Remarque 1.2.35

L'ajout de ► sans ◀ n'apporte pas d'expressivité car, sans ◀, le curseur reste toujours à droite du mot écrit.

Remarque 1.2.36

Nous dirons souvent qu'un langage de clavier est dans une certaine classe pour indiquer qu'il est reconnu par un clavier de cette classe. Ainsi, nous dirons que L est dans \mathbf{FK} s'il existe $K \in \mathbf{FK}$ tel que $L = \mathcal{L}(K)$.

Nous définissons enfin deux problèmes de décision fondamentaux sur les claviers.

Définition 1.2.37

Soit \mathcal{K} une classe de claviers, on définit les deux problèmes de décision suivants.

Problème du mot : $\begin{cases} \text{DONNÉE :} & K \in \mathcal{K} \text{ et } w \in A^* \\ \text{QUESTION :} & w \in \mathcal{L}(K) ? \end{cases}$

Problème d'universalité : $\begin{cases} \text{DONNÉE :} & K \in \mathcal{K} \\ \text{QUESTION :} & \mathcal{L}(K) = A^* ? \end{cases}$

2. Propriétés des claviers

Avant de rentrer dans le vif du sujet et de nous attaquer à l'étude des langages de claviers, nous établissons des propriétés sur les différents types de claviers. Elles faciliteront alors nos études ultérieures.

2.1. Généralités

Commençons par exprimer le fait qu'une touche d'un clavier ne peut effectuer que des transformations locales, et ne peut affecter que les lettres à une distance du curseur inférieure à la longueur de cette touche.

Lemme 2.1.1 (Localité)

Soient $\langle u|v \rangle$ une configuration et $t = \sigma_1 \dots \sigma_n$ une touche. En notant $\langle u_n|v_n \rangle = \langle u|v \rangle \cdot t$, on a que $u[1, |u| - n]$ est un préfixe de u_n et que $v[n + 1, |v|]$ est un suffixe de v_n .

Voir preuve en A.1.1

Ainsi, t ne permet pas de modifier les lettres d'une configuration à une distance plus grande que $|t|$ du curseur.

Lorsqu'une touche est appliquée avec un curseur suffisamment loin des extrémités du mot, alors son action ne souffrira pas d'« effets de bord » tels que l'application d'un retour à l'extrémité gauche.

Lemme 2.1.2 (Efficience loin des bords)

Soient $t = \sigma_1 \dots \sigma_n$ une touche et $\langle u|v \rangle$ une configuration. Si $n \leq \min(|u|, |v|)$, alors en notant $\langle u_n|v_n \rangle = \langle u|v \rangle \cdot t$, on a que $\langle u|v \rangle \xrightarrow{t}_e \langle u_n|v_n \rangle$ c'est-à-dire que l'exécution efficace ne bloque pas.

Voir preuve en A.1.2

De plus, une touche ne peut pas ajouter ou retirer un nombre de lettres supérieur à sa longueur.

Lemme 2.1.3 (Encadrement des tailles)

Soient $t = \sigma_1 \cdots \sigma_n$ une touche et $\langle u|v \rangle$ une configuration. On pose $\langle u_n|v_n \rangle = \langle u|v \rangle \cdot t$. Alors

$$|uv| - |t|_{\leftarrow} + \sum_{x \in A} |t|_x \leq |u_nv_n| \leq |uv| + \sum_{x \in A} |t|_x.$$

On en déduit notamment que $||u_nv_n| - |uv|| \leq n$.

Voir preuve en A.1.3

On peut même être plus précis et combiner les lemmes précédents pour obtenir que, si le curseur est loin des extrémités du mot, la différence de longueur due à l'application d'une touche est son nombre de lettres moins son nombre de retours.

Lemme 2.1.4 (Égalité des tailles loin des bords)

Soient $t = \sigma_1 \cdots \sigma_n$ une touche et $\langle u|v \rangle$ une configuration telle que $|u| \geq n$ et $|v| \geq n$. En posant $\langle u_n|v_n \rangle = \langle u|v \rangle \cdot t$, alors

$$|u_nv_n| = |uv| - |t|_{\leftarrow} + \sum_{x \in A} |t|_x.$$

Voir preuve en A.1.4

Ces résultats ont pour conséquence la proposition suivante.

Proposition 2.1.5

Soit K un clavier automatique de langage L . Notons $(l_n)_{n \in \mathbb{N}}$ les longueurs des mots reconnus par L triés par ordre croissant. Alors $(l_{n+1} - l_n)_{n \in \mathbb{N}}$ est bornée par $\|K\|_\infty$.

Voir preuve en A.1.5

Remarque 2.1.6

Ceci nous permet de montrer qu'on ne peut pas reconnaître $\{\mathbf{a}^{n^2} \mid n \in \mathbb{N}\}$ ou encore $\{\mathbf{a}^p \mid p \text{ premier}\}$ avec un clavier automatique.

En fait, cette proposition peut s'étendre à tout clavier.

Proposition 2.1.7

Soit $K = (T, F)$ un clavier de langage L . Notons $(l_n)_{n \in \mathbb{N}}$ les longueurs des mots reconnus par L triés par ordre croissant. Alors $(l_{n+1} - l_n)_{n \in \mathbb{N}}$ est bornée par $5\|K\|_\infty$.

Voir preuve en A.1.6

Définissons maintenant quelques notions utiles. Premièrement, une touche peut tout à fait écrire une lettre puis l'effacer en appliquant un retour. La présence d'une lettre a dans une touche ne signifie donc pas qu'elle apparaîtra toujours lorsque la touche sera appliquée. C'est pourquoi on définit les notions suivantes.

Définition 2.1.8

On dit qu'une touche $t \in T$ écrit un a s'il existe une lettre $b \neq a$, $(b^{\|K\|_\infty}, b^{\|K\|_\infty}) \cdot t$ contient un a .

Nos configurations sont des couples $\langle u|v \rangle$ mais on s'intéresse souvent à la concaténation uv des deux mots. La notion suivante facilite le passage d'un point de vue à l'autre.

Définition 2.1.9

Soient $u = u_n \dots u_1$, $v = v_1 \dots v_k$ et i un entier *non nul* tel que $-n \leq i \leq k$. On pose

$$\langle u|v \rangle_i = \begin{cases} u_{-i} & \text{si } i < 0 \\ v_i & \text{sinon} \end{cases}$$

Notamment, $|\langle u|v \rangle| = n + k$.

À présent on s'intéresse à l'effet d'une touche. Nous allons observer, étant données une lettre a et une touche t , l'ensemble des positions par rapport au curseur auxquelles t écrit des a . Ensuite nous prouverons que cet ensemble est toujours le même à condition d'être suffisamment loin des extrémités du mot.

Définition 2.1.10

Soient $u, v \in A^*$, on définit

$$\begin{cases} I_a(u, v) = \{i \mid \langle u|v \rangle_i = a\} \\ G_a(u, v) = I_a(u, v) \setminus \mathbb{N} \\ D_a(u, v) = I_a(u, v) \cap \mathbb{N} \end{cases}$$

On a donc $I_a(u, v) = G_a(u, v) \cup D_a(u, v)$. I_a correspond aux positions des a , G_a aux positions à gauche du curseur, et D_a à celles à droite du curseur.

Définition 2.1.11 (Distance maximale entre deux a)

On définit $d_a(u)$ comme le nombre maximal de lettres entre deux a dans u . Formellement,

$$d_a(u) = \max\{|w| \mid awa \in \text{Fact}(u)\}.$$

avec la convention $\max(\emptyset) = -\infty$. On écrit abusivement $d_a(\langle u|v \rangle)$ pour $d_a(uv)$.

À présent nous montrons que lorsque le curseur est loin des extrémités du mot, l'action d'une touche place les mêmes lettres aux mêmes distances du curseur.

Lemme 2.1.12

Soient $t = \sigma_1 \dots \sigma_n$ une touche, et $u, v, u', v' \in A^*$ de longueurs supérieures ou égales à n et ne contenant pas de a . En posant $\langle u_n|v_n \rangle = \langle u|v \rangle \cdot t$ et $\langle u'_n|v'_n \rangle = \langle u'|v' \rangle \cdot t$, on a

$$I_a(u_n, v_n) = I_a(u'_n, v'_n).$$

Voir preuve en A.1.7

On en tire immédiatement les deux résultats suivants.

Corollaire 2.1.13

Soient $t \in T$ une touche et $u, v, u', v' \in A^*$ des mots ne contenant pas de a et de longueurs supérieures ou égales à n . Alors

$$d_a(\langle u|v \rangle \cdot t) = d_a(\langle u'|v' \rangle \cdot t).$$

Corollaire 2.1.14 (Écriture hors contexte)

Si $t \in T$ écrit un a alors pour tous $u, v \in A^*$ avec $|u|, |v| \geq \|K\|_\infty$, $\langle u|v \rangle \cdot t$ contient un a .

Lemme 2.1.15

Soient $t = \sigma_1 \dots \sigma_n$ une touche et $u, v \in A^*$ de longueurs supérieures ou égales à n . Si $|\langle u|v \rangle|_a < |\langle u|v \rangle \cdot t|_a$, alors t écrit un a .

Voir preuve en A.1.8

Définition 2.1.16 (Équivalence contextuelle de touches)

Soit $t_1, t_2 \in S^*$ deux touches quelconques. On dit que t_1 et t_2 sont équivalentes (contextuellement), et on note $t_1 \sim t_2$, si pour toute configuration $\langle u|v \rangle$, on a

$$\langle u|v \rangle \cdot t_1 = \langle u|v \rangle \cdot t_2.$$

On dit que t_1 et t_2 sont équivalentes loin des bords, et on note $t_1 \sim_l t_2$, s'il existe $n \in \mathbb{N}$ tel que pour toute configuration $\langle u|v \rangle$ telle que $|u| \geq n$ et $|v| \geq n$ on a

$$\langle u|v \rangle \cdot t_1 = \langle u|v \rangle \cdot t_2.$$

Exemple 2.1.17

On a $a \leftarrow b \sim b$. Mais on n'a pas $\blacktriangleleft \blacktriangleright \sim \varepsilon$, car ces touches ne donnent pas la même configuration à partir de $\langle \varepsilon|a \rangle$. En revanche, on a $\blacktriangleleft \blacktriangleright \sim_l \varepsilon$.

Lemme 2.1.18 (Monotonie)

Soient $u, v, u', v' \in A^*$ et $t \in S^*$ une touche. Notons $\langle x|y \rangle = \langle u|v \rangle \cdot t$ et $\langle x'|y' \rangle = \langle u'|v' \rangle \cdot t$. Si $|u| \leq |u'|$ et $|v| \leq |v'|$, alors $|x| \leq |x'|$ et $|y| \leq |y'|$.

Voir preuve en A.1.9

Corollaire 2.1.19

Soient $t \in T$ et $u, v, u', v' \in A^*$ tels que $|u| \leq |u'|$ et $|v| \leq |v'|$. Si t efface une lettre de $\langle u|v \rangle$, alors t efface une lettre de $\langle u'|v' \rangle$.

Preuve

Cela découle de l'observation suivante : \leftarrow efface une lettre de $\langle u|v \rangle$ si et seulement si $|u| > 0$. \square

2.2. Les claviers sans flèches

Dans cette sous-section, nous considérons des claviers REK, à savoir des claviers où $S = A \cup \{\leftarrow, \blacksquare\}$. En utilisant un tel clavier, la partie droite d'une configuration est toujours vide (ce que montre le lemme qui suit). Ainsi, les configurations obtenues seront de la forme $\langle u|\varepsilon \rangle$ et il nous arrivera d'utiliser la notation u pour une telle configuration.

Lemme 2.2.1 (Caractérisation des configurations REK)

Soient K un clavier de REK, $u \in A^*$ et $t_1 \dots t_n \in K^*$. Alors $\langle u|\varepsilon \rangle \cdot t_1 \dots t_n$ est de la forme $\langle v|\varepsilon \rangle$ avec $v \in A^*$.

Voir preuve en A.2.1

Lemme 2.2.2 (Forme normale d'une touche REK)

Soit $t \in S^*$ une suite d'opérations élémentaires de REK. Alors il existe un unique $k \in \mathbb{N}$ et un unique $w \in A^*$ tels que $t \sim \leftarrow^k w$. De plus, k et w peuvent être trouvées en temps polynomial en la taille de t .

Idée de la preuve

L'idée est de supprimer les séquences $a\leftarrow$ (avec $a \in A$) de la touche tant qu'il y en a (car $a\leftarrow$ est équivalent à la séquence vide).

Voir preuve en A.2.2

Nous donnons ci-dessous un algorithme qui calcule la forme normale REK d'une touche de S .

Algorithme 2.2.3

Nous pouvons mettre une touche sous forme normale REK en temps linéaire.

Algorithme 1 : Mise sous forme normale REK

```
Fonction FormeNormaleREK( $t$ )
  si  $t$  est vide alors
    | retourner  $\square$ ;
  fin
   $t' \leftarrow \square$ ;
  pour  $i$  allant de 2 à  $|t|$  faire
    | si  $t'[0] \in A$  et  $t[i] = \leftarrow$  alors
    | |  $t' \leftarrow \text{queue}(t')$ 
    | sinon
    | |  $t' = t[i] : : t'$ 
    | fin
  fin
  retourner  $\text{inverse}(t')$ 
```

Fin

Corollaire 2.2.4 (Forme normale de REK)

Soit K un clavier de REK. Alors, il existe un clavier K_0 dont les touches sont de la forme $\leftarrow^k w$ avec $w \in A^*$ et tel que $\mathcal{L}(K_0) = \mathcal{L}(K)$. De plus K_0 est constructible en temps polynomial.

Preuve

Soit K un clavier de REK. Le lemme 2.2.2 (Forme normale d'une touche REK) nous donne, pour tout touche t de K , l'existence d'une touche $f(t)$ équivalente à t et de la forme voulue.

Le clavier $K_0 = \{f(t) \mid t \in K\}$ reconnaît donc le même langage que K et a la forme voulue. Et puisque $f(t)$ est constructible en temps polynomial en $|t|$, K_0 est constructible en temps polynomial en $|K|$. \square

Remarque 2.2.5

Pour la suite, nous supposons que les claviers REK considérés sont de cette forme. Notons de plus que la construction du clavier équivalent peut se faire en temps polynomial, ce qui aura son importance lors de résultats de complexité.

2.3. Les claviers sans flèche droite

On considère ici des claviers de GREK, à savoir des claviers n'utilisant pas \blacktriangleright . On se place donc avec $S = A \cup \{\leftarrow, \blacktriangleleft, \blacksquare\}$.

L'une des grandes idées de GREK est que l'on ne peut aller à droite et donc revenir sur un mot à droite du pointeur, ce que formalise le théorème suivant.

Théorème 2.3.1 (fondamental de GREK)

Soient $t = \sigma_1 \cdots \sigma_n$ une suite d'opérations élémentaires et $\langle u|v \rangle$ une configuration. Nous posons $\langle x_n|y_n \rangle = \langle \varepsilon|\varepsilon \rangle \cdot t$. Alors $\langle u|v \rangle \cdot t$ est de la forme $\langle u_n x_n|v_n v \rangle$ avec y_n un sous-mot de v_n et u_n un préfixe de u .

Voir preuve en A.3.1

Parlons désormais d'un lemme fondamental permettant de mieux distinguer la puissance de GREK : l'insensibilité à la position. On prend une touche de GREK qui écrit une lettre a (elle peut faire de nombreuses autres actions mais n'efface pas ce a) depuis une configuration. Alors cette touche écrit ce a sans l'effacer depuis toute configuration.

Pour énoncer ce lemme de manière formelle, on utilise un caractère distingué $\hat{a} \notin A$ que l'on peut ensuite remplacer par $a \in A$ via le morphisme $\hat{a} \mapsto a$.

Lemme 2.3.2 (Insensibilité à la position)

Soient \hat{a} une lettre distinguée, et t une touche de GREK contenant un unique \hat{a} . Soit $\langle u|v \rangle$ une configuration ne contenant pas de \hat{a} . Si $\langle u|v \rangle \cdot t$ contient un \hat{a} , alors pour toute configuration $\langle u'|v' \rangle$, $\langle u'|v' \rangle \cdot t$ contient \hat{a} .

Idée de la preuve

L'idée est de considérer la distance du curseur à la lettre \hat{a} qui vient d'être écrite. Dès lors que l'on passe à gauche de \hat{a} , il devient impossible de l'effacer. Pour effacer \hat{a} , il est donc suffisant et nécessaire, de faire un retour alors qu'on est à droite du \hat{a} , et ce avant d'éventuellement repasser à sa gauche.

On remarque ensuite que cela ne dépend pas du contexte, *i.e.* de la position du curseur par rapport aux bords ce qui se voit plutôt bien : on ne peut pas aller à droite, donc pas d'effets de bord à droite et pour faire des effets de bords à gauche, il faut déjà être passé à gauche du \hat{a} (donc l'avoir effacé ou l'avoir inscrit pour de bon).

Voir preuve en A.3.2

Remarque 2.3.3

L'hypothèse d'unicité de l'occurrence de \hat{a} dans t peut être retirée en utilisant un morphisme $\hat{a} \mapsto a$.

2.4. Les claviers sans retour

Dans cette section, nous allons étudier les langages sans retour FEK. Nous nous plaçons donc avec $S = A \cup \{\blacktriangleleft, \blacktriangleright, \blacksquare\}$.

L'absence du retour arrière permet d'obtenir des propriétés très intéressantes.

Lemme 2.4.1

Soit K un clavier de FEK et $c_0 \xrightarrow{t_1} \dots \xrightarrow{t_n} c_n$ une exécution de K . On a $|c_0| \leq \dots \leq |c_n|$.

Preuve

Par induction immédiate sur $n \in \mathbb{N}$. □

Nous pouvons même être plus précis en nous intéressant à chaque lettre. Puisqu'il n'y a pas de retour, si une touche contient une lettre, alors l'action de cette touche permet effectivement d'écrire la touche.

Lemme 2.4.2 (Non effacement)

Soient t une touche, $\langle u|v \rangle$ une configuration et $a \in A$. En posant $\langle u_t|v_t \rangle = \langle u|v \rangle \cdot t$, on a

$$|u_t v_t|_a = |u v|_a + |t|_a.$$

Preuve

On montre le résultat par induction sur t . Si t est vide, le résultat est vrai. Sinon, $t = t' \sigma$, et l'hypothèse d'induction nous donne que $|u_{t'} v_{t'}|_a = |u v|_a + |t'|_a$. On distingue plusieurs cas.

- Si $\sigma = a \in A$, $u_t = u_{t'} a$ et $v_t = v_{t'}$.
- Si $\sigma \in \{\blacktriangleleft, \blacktriangleright\}$, $u_t v_t = u_{t'} v_{t'}$.

Dans les deux cas, on a

$$|u_tv_t|_a = |u_{t'}v_{t'}|_a + |\sigma|_a = |uv|_a + |t'|_a + |\sigma|_a = |uv|_a + |t|_a$$

ce qui permet de conclure notre induction. \square

Ce résultat nous permet alors de caractériser un peu les langages de claviers sans retour en nous intéressant aux occurrences des lettres.

Lemme 2.4.3 (Itération de lettre)

Soient $L \in \text{FK}$, $w \in L$ et a une lettre apparaissant dans w . Alors il existe une suite $(w_n)_{n \in \mathbb{N}}$ de mots de L telle que $\forall n \in \mathbb{N}, |w_n|_a \geq n$.

Idée de la preuve

Des hypothèses on déduit que le clavier contient une touche qui écrit un a . Il suffit alors d'itérer cette touche pour écrire autant de a que souhaité, car on est assuré de ne rien effacer !

Voir preuve en A.4.1

Cette proposition n'est pas vraie telle quelle dans FEK (la touche écrivant un a pourrait être finale). Cependant, on peut l'adapter de la manière suivante.

Lemme 2.4.4 (Itération de lettre FEK)

Soit $K = (T, F)$ un clavier de FEK. S'il existe $w \in L$ tel que $|w|_a > \|K\|_\infty$, alors pour tout $n \in \mathbb{N}$, il existe $w_n \in L$ tel que $|w_n|_a \geq n$.

Idée de la preuve

S'il y a un mot avec plus de $\|K\|_\infty$ a , alors tous les a de ce mot n'ont pas pu être écrits par la touche acceptante d'une exécution menant à accepter ce mot. Il y a donc une touche transiente qui contient un a et qu'on peut itérer avant d'appuyer sur une touche acceptante.

Voir preuve en A.4.2

Corollaire 2.4.5

Soit L un langage de FEK. Il existe $N \in \mathbb{N}$ tel que s'il existe $w \in L$ vérifiant $|w|_a \geq N$ alors pour tout $k \geq N$, il existe $w_k \in L$ tel que $|w_k|_a \geq k$.

Preuve

Soit K un clavier de FEK reconnaissant L . Le lemme précédent montre que $N = \|K\|_\infty$ convient. \square

Lemme 2.4.6 (Distanciation minimale)

Soit L un langage de FEK et $a \in A$. S'il existe $M \in \mathbb{N}$ tel que pour tout $w \in L$, $|w|_a \leq M$, alors il existe $D \in \mathbb{N}$ tel que pour tout $w \in L$, $d_a(w) \leq D$.

Preuve

On considère K qui reconnaît L et on prend $D = \|K\|_\infty$. Soit t une touche de K qui contient un a . Si t n'est pas acceptante, alors le mot obtenu en appuyant $M + 1$ fois sur t contient au moins $M + 1$ occurrences de a (par le [lemme 2.4.2 \(Non effacement\)](#)), ce qui est absurde.

Donc toute touche contenant un a est acceptante. Si w écrit par K contient des a , ils ont donc tous été écrits par une seule touche t (la dernière), et le [lemme 2.1.1 \(Localité\)](#) nous permet de conclure ;

$$d_a(w) \leq |t| \leq \|K\|_\infty. \quad \square$$

Théorème 2.4.7 (du sous-mot)

Soient K un clavier de FEK, τ une suite de touches de K et $\langle u|v \rangle$ une configuration. En posant $\langle u_\tau|v_\tau \rangle = \langle u|v \rangle \cdot \tau$, on a que uv est un sous-mot de $u_\tau v_\tau$.

Preuve

L'action d'une suite de touches étant l'action de la concaténation des touches de cette suite, il suffit de montrer le résultat pour les touches. Soit donc t une touche, on montre le résultat par induction sur t . Si t est vide, le résultat est vrai. Sinon, $t = t'\sigma$ et l'hypothèse d'induction nous donne que uv est un sous-mot de $u_{t'}v_{t'}$ où $\langle u_{t'}|v_{t'} \rangle = \langle u|v \rangle \cdot t'$. On distingue plusieurs cas.

- Si $\sigma = a \in A$, alors $u_t = u_{t'}a$ et $v_t = v_{t'}$.
- Si $\sigma \in \{\blacktriangleleft, \blacktriangleright\}$, alors $u_t v_t = u_{t'} v_{t'}$.

Dans les deux cas, uv est bien un sous-mot de $u_t v_t$ ce qui permet de conclure. \square

3. Les langages de claviers

3.1. Les claviers sans flèches

3.1.1. MK

Commençons par étudier les claviers minimaux.

Remarque 3.1.1

Soit $K \in \text{MK}$. Les touches de K sont des mots de A^* et on a $\mathcal{L}(K) = K^+$.

Exemple 3.1.2

Le langage du clavier $\{\text{ab}, \text{bc}\}$ est $(ab + bc)^+$.

Lemme 3.1.3

Le problème du mot sur les claviers de MK est décidable en temps polynomial.

Preuve

Si K est un clavier de MK, alors $K = \{w_1, \dots, w_n\} \subseteq A^*$ et $\mathcal{L}(K) = (w_1 + \dots + w_n)^+$. Ainsi, on construit en temps polynomial l'expression rationnelle de $\mathcal{L}(K)$, et le problème du mot sur les expressions rationnelles est polynomial. \square

Lemme 3.1.4

L'universalité est décidable en temps polynomial sur les claviers de MK.

Preuve

Soit $K \subseteq A^*$ un clavier. Le langage de K est K^+ , donc K est universel si et seulement si $K^+ = A^*$. Montrons à présent que $K^+ = A^*$ si et seulement si $A \cup \{\varepsilon\} \subseteq K$.

- Supposons que $A \cup \{\varepsilon\} \subseteq K$. Alors, clairement $A^* = (A \cup \varepsilon)^+ \subseteq K^+$.
- Réciproquement, supposons que $A \cup \{\varepsilon\} \not\subseteq K$. Alors soit $\varepsilon \notin K$, auquel cas $\varepsilon \notin K^+$, soit il existe $a \in A$ tel que $a \notin K$, auquel cas $a \notin K^+$. Dans les deux cas $A^* \neq K^+$.

On peut donc tester l'universalité en regardant si $A \cup \{\varepsilon\} \subseteq K$, ce qui est possible en temps polynomial. \square

3.1.2. EK

Ici, on travaille avec $S = A \cup \{\blacksquare\}$.

Remarque 3.1.5

Soit $K = (T, F)$ un clavier de **EK**. Alors on a $\mathcal{L}(K) = T^*F$. On peut donc construire l'expression rationnelle correspondante en temps polynomial en $|K|$.

Lemme 3.1.6

Le problème du mot sur les claviers de **EK** est décidable en temps polynomial.

Preuve

Soit $K = (T, F) \in \mathbf{EK}$. Par la caractérisation donnée en [remarque 3.1.5](#), le langage de K est reconnu par une expression rationnelle constructible en temps polynomial. Ainsi, on construit l'expression rationnelle correspondant à $\mathcal{L}(K)$ et on peut vérifier en temps polynomial l'appartenance d'un mot à ce langage. \square

Lemme 3.1.7

Le problème de l'universalité sur les claviers de **EK** est décidable en espace polynomial.

Preuve

Soit $K = (\{w_1, \dots, w_n\}, \{v_1, \dots, v_k\})$ un clavier de **EK**. Par la [remarque 3.1.5](#), $\mathcal{L}(K)$ est reconnu par une expression rationnelle constructible en temps polynomial. Le problème de l'universalité d'une expression rationnelle étant décidable en espace polynomial, ce problème l'est aussi. \square

Remarque 3.1.8

MK est strictement inclus dans **EK**. Par exemple, on sait faire $\{a\}$ avec le clavier $\{a\blacksquare\}$, ce que l'on ne sait pas faire avec un clavier minimal.

3.1.3. RK**Exemple 3.1.9 (Les langages finis dans RK)**

Soit L un langage fini. Posons $M = \max_{w \in L} |w|$. On considère des touches qui effacent M caractères puis écrivent un mot de L , c'est-à-dire

$$K = \{\leftarrow^M w \mid w \in L\}.$$

Alors K reconnaît le langage L .

Corollaire 3.1.10

$\mathbf{MK} \subsetneq \mathbf{RK}$.

Proposition 3.1.11

Le langage $L = (a^2)^*(b + b^2)$ reconnu par $\{aa, b\blacksquare, bb\blacksquare\}$ n'est pas dans **RK**.

[Voir preuve en B.1.1](#)

Corollaire 3.1.12

$\mathbf{EK} \not\subseteq \mathbf{RK}$ et $\mathbf{RK} \subsetneq \mathbf{REK}$.

Proposition 3.1.13 (RK non inclus dans FEK)

Le langage L_C engendré par le clavier de RK $\{\leftarrow a\#\$, \leftarrow\leftarrow b\#\$\$\}$ n'est pas dans FEK.

Voir preuve en B.3.2

Corollaire 3.1.14

$RK \not\subset EK$.

3.1.4. REK**Théorème 3.1.15 (REK est inclus dans Alg)**

Soit $K \in REK$. Alors $\mathcal{L}(K)$ est algébrique et on peut construire en temps polynomial un automate à pile non déterministe $\mathcal{A}(K)$ reconnaissant $\mathcal{L}(K)$.

Idée de la preuve

On part de deux observations.

1. Par le [lemme 2.2.1 \(Caractérisation des configurations REK\)](#), le mot à droite du curseur est toujours vide.
2. Par le [lemme 2.2.2 \(Forme normale d'une touche REK\)](#), toute touche d'un clavier de REK est équivalente à une touche de la forme $\leftarrow^* A^*$.

Ces deux faits nous indiquent que l'action d'une touche d'un clavier de REK revient toujours à retirer un nombre borné de lettres à la fin du mot puis à en ajouter un nombre borné. Ce comportement étant semblable à celui d'une pile, on encode facilement les actions des touches d'un clavier dans celles d'un automate à pile. On construit un automate non-déterministe qui devine et simule une suite de touches en gardant le mot obtenu dans sa pile, puis lit le contenu de sa pile. À noter que cet automate reconnaît en fait le miroir du langage du clavier correspondant, on a donc besoin d'un dernier argument, la stabilité des langages algébriques par miroir, pour conclure.

Voir preuve en B.1.2

Théorème 3.1.16 (REK est inclus dans Rat)

Soit $K \in REK$. Alors $\mathcal{L}(K)$ est rationnel et on peut construire en temps polynomial un automate $\mathcal{A}(K)$ non déterministe reconnaissant $\mathcal{L}(K)$.

Idée de la preuve

Nous donnons ici une idée de preuve pour RK. La preuve pour REK ne pose pas de difficultés supplémentaires. Soit K un clavier de RK. L'idée est que lorsqu'on exécute une suite de touches d'un clavier de RK, à tout moment on peut diviser le mot obtenu w en deux parties :

- un préfixe u de lettres qui ne seront jamais effacées ;
- un suffixe v de lettres qui seront effacées à une étape ultérieure de l'exécution.

Prenons un exemple : pour écrire le mot abb avec le clavier constitué des touches $\leftarrow^2 abcc, \leftarrow^3 b, \leftarrow^2 cbc$, il nous suffit d'écrire le préfixe ab grâce à t_1 et écrire le b restant grâce à t_2 . Le souci est que si on exécute t_1 puis t_2 , t_2 effacera trop de lettres et on obtiendra ab au lieu de abb . Il nous faut donc une suite de touches qui n'efface pas le ab , mais remplace les deux c par trois lettres quelconques, pour pouvoir appliquer t_2 . Ceci est possible en exécutant t_3 , qui remplace cc par cbc . La suite de touches $t_1 t_3 t_2$ permet donc d'écrire abb .

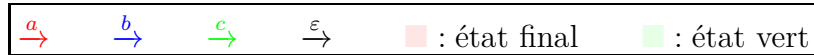
Passons à un exemple plus complexe, et cherchons à écrire abb avec le clavier dont les touches sont $t_1 = \leftarrow^2 abc^5, t_2 = \leftarrow^3 b$. Encore une fois, il nous suffit d'écrire le ab avec t_1 et le b final avec t_2 . Cependant, il nous faut une suite de touches capable de supprimer les cinq c écrits en appliquant t_1 , et de les remplacer par trois lettres quelconques.

Remarquons la chose suivante : t_1 retire deux lettres et en ajoute sept (donc ajoute cinq lettres au total), tandis que t_2 retire trois lettres et en ajoute une (donc retire deux lettres au total). Cinq et deux étant premiers entre eux, par le théorème de BÉZOUT (à peu de choses près), il existe des entiers positifs m_1 et m_2 tels que $5m_1 - 2m_2 = -2$, et donc en appliquant m_1 fois t_1 puis m_2 fois t_2 , on retire en tout deux lettres après le ab . On peut ensuite appliquer t_2 pour obtenir le mot abb .

Les exemples précédents illustrent l'idée de la construction : on construit un automate qui, lorsqu'il lit un mot w , devine une décomposition $w = u_1 \cdots u_k$ et pour chaque u_i une touche t_i de la forme $\leftarrow^{k_i} u_i v_i$. L'automate vérifie ensuite que pour tout i , il existe une suite de touches permettant de remplacer le suffixe v_i par k_{i+1} lettres quelconques, afin de pouvoir appliquer t_{i+1} sans effacer trop ou trop peu de lettres. L'automate n'a pas à deviner cette suite de touches, on aura préalablement déterminé par des critères de divisibilité les couples (k, k') avec $0 \leq k, k' \leq \|K\|_\infty$ tels qu'il existe une suite de touches permettant de passer de k à k' lettres.

Nous omettons dans ce schéma de preuve plusieurs difficultés, par exemple le cas où il nous faut passer de k à k' lettres mais toutes les touches ajoutant des lettres effacent plus de k lettres. Ces cas sont traités dans la preuve complète en annexe.

Enfin, nous présentons en [figure 3.1 \(Automate du clavier \$\{\leftarrow^2 abc, \leftarrow^4 bb\}\$ \)](#) l'automate obtenu à partir du clavier $\{\leftarrow^2 abc, \leftarrow^4 bb\}$ en utilisant la construction présentée dans la preuve de ce lemme. Son langage est $a^*(abc + bb)$.



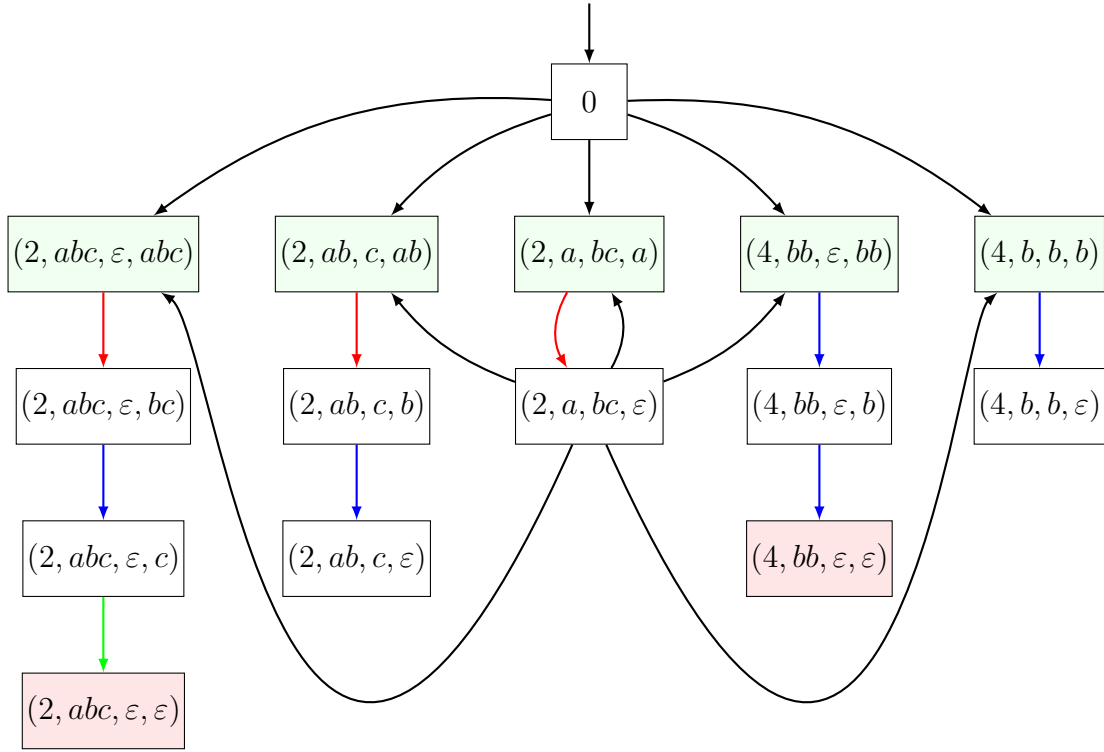


Figure 3.1. – Automate du clavier $\{\leftarrow^2\text{abc}, \leftarrow^4\text{bb}\}$

Voir preuve en B.1.2

Ce résultat nous permet de donner la complexité de certains problèmes de décision.

Théorème 3.1.17

Le problème du mot est décidable en temps polynomial sur les claviers de REK.

Preuve

Soient K un clavier de REK et $u \in A^*$. Le corollaire 2.2.4 (Forme normale de REK) et le théorème 3.1.16 (REK est inclus dans Rat), nous permet de construire un clavier K' en forme normale, équivalent à K et de taille polynomiale en celle de K , puis un automate $\mathcal{A}(K')$ de taille polynomiale en celle de K' (et donc en celle de K), le tout en temps polynomial.

$\mathcal{A}(K')$ reconnaît le même langage que K ; pour tester si $w \in \mathcal{L}(K)$, il nous suffit donc de tester si $w \in \mathcal{L}(\mathcal{A}(K))$, ce qu'on peut faire en espace polynomial. \square

Théorème 3.1.18

L'universalité est décidable en espace polynomial sur les claviers de REK.

Preuve

Soit K un clavier de REK. En appliquant le [corollaire 2.2.4 \(Forme normale de REK\)](#) et le [théorème 3.1.16 \(REK est inclus dans Rat\)](#), nous pouvons construire un clavier K' en forme normale, équivalent à K et de taille polynomiale en celle de K , puis un automate $\mathcal{A}(K')$ de taille polynomiale en celle de K' (et donc en celle de K), le tout en temps polynomial.

$\mathcal{A}(K')$ reconnaît le même langage que K ; pour tester l'universalité de K , il nous suffit donc de tester l'universalité de $\mathcal{A}(K)$, ce qu'on peut faire en temps polynomial. \square

Pour finir cette petite étude de REK, on peut se demander si tous les langages rationnels sont reconnus par clavier sans flèches... Et ce n'est pas le cas.

Proposition 3.1.19

$L = a^*c^* \notin \text{REK}$.

Idée de la preuve

On raisonne par l'absurde. On possède une suite de touches τ qui écrit un a et valide. Mais on possède également une suite de touches τ' qui écrit un nombre aussi grand de c que souhaité, sans valider. On conclut en regardant $\varepsilon \cdot \tau \cdot \tau'$.

[Voir preuve en B.1.2](#)

Corollaire 3.1.20

$\text{Rat} \not\subset \text{REK}$.

Finalement, on obtient la [figure 3.2 \(Hiérarchie des langages sans flèches\)](#) qui résume les inclusions des classes étudiées dans cette section.

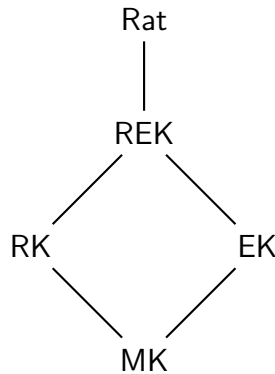


Figure 3.2. – Hiérarchie des langages sans flèches

3.2. Les claviers sans flèche droite

3.2.1. GK

Exemple 3.2.1 (Palindromes pairs)

Le clavier $K = \{aa\blacktriangleleft \mid a \in A\}$ reconnaît le langage L_{pal} des palindromes pairs non vide (on a en effet l'invariant que toute configuration atteinte est de la forme $\langle u|\tilde{u} \rangle$ pour un $u \in A^*$).

Ce langage est donc dans GK.

Corollaire 3.2.2

$\text{MK} \subsetneq \text{GK}$ et $\text{GK} \not\subseteq \text{REK}$ car tous les éléments de REK sont rationnels.

Exemple 3.2.3 (Mot de Dyck)

Le clavier $K = \{()\blacktriangleleft, \blacktriangleleft\}$ reconnaît le langage L des mots bien parenthésés qui est donc dans GK.

Voir preuve en B.2.1

Proposition 3.2.4

Si L est fini et dans GK alors $L \subset \{\varepsilon\}$.

Voir preuve en B.3.1

Corollaire 3.2.5

$\text{EK} \not\subseteq \text{GK}$ et $\text{RK} \not\subseteq \text{GK}$ car les langages finis sont exprimables dans EK et dans RK.

On en déduit également $\text{GK} \subsetneq \text{GEK}$ et $\text{GK} \subsetneq \text{GRK}$ car les langages finis sont dans EK et dans RK.

3.2.2. GEK

Théorème 3.2.6

On considère le langage $L_{\text{pal}@} = \{w@w \mid w \in A^*\}$ où @ est une lettre fraîche. Le problème suivant est indécidable.

$$\text{Palindrome : } \begin{cases} \text{DONNÉE :} & \text{Un langage } L \text{ de GEK} \\ \text{QUESTION :} & L \cap L_{\text{pal}@} = \emptyset ? \end{cases}$$

Idée de la preuve

On réduit le problème de correspondance de Post. Soit $I = \{(u_i, v_i) \mid i \in \llbracket 1, n \rrbracket\}$ une instance de Post. On considère le clavier

$$K = \{u_i \tilde{v}_i \blacktriangleleft^{|v_i|} \mid i \in \llbracket 1, n \rrbracket\} \cup \{@\blacksquare\}.$$

On montre que $\mathcal{L}(K) \cap L_{\text{pal}@} \neq \emptyset$ si et seulement si I est dans PCP.

Voir preuve en B.2.2

Proposition 3.2.7

Le langage L_C engendré par le clavier de RK $\{\leftarrow a\# \$, \leftarrow \leftarrow b\# \$ \$\}$ n'est pas dans GEK.

Voir preuve en B.3.2

Corollaire 3.2.8

On en déduit les résultats suivants.

$$\text{RK} \not\subseteq \text{GEK} \quad \text{GEK} \subsetneq \text{GREK} \quad \text{GRK} \not\subseteq \text{GEK}.$$

3.2.3. GRK

Proposition 3.2.9

Le langage $L \triangleq (a^2)^*(b + b^2)$ n'est pas dans GRK.

Voir preuve en B.2.3

Corollaire 3.2.10

On en déduit les résultats suivants.

$$\text{EK} \not\subseteq \text{GRK} \quad \text{GRK} \subsetneq \text{GREK} \quad \text{GEK} \not\subseteq \text{GRK}.$$

3.2.4. GREK

Proposition 3.2.11

Le langage $L = \{ab^{n+1}a^n \mid n \in \mathbb{N}\}$ n'est pas dans GREK.

Voir preuve en B.2.4

Théorème 3.2.12

Soit L un langage de GREK. Alors L est algébrique et on peut construire en temps polynomial un automate à pile non déterministe reconnaissant L .

Idée de la preuve

La preuve complète est en annexe.

L'idée est d'utiliser le fait que rien de ce qui est écrit à droite du curseur ne peut être affecté par une touche sans flèche droite.

On construit un automate à pile devinant une suite de touches et maintenant l'invariant « Après avoir exécuté une suite de touches $t_1 \cdots t_n$, en posant u le contenu de la pile et v le mot lu par l'automate, on a $\langle \varepsilon | \varepsilon \rangle \cdot t_1 \cdots t_n = \langle u | \tilde{v} \rangle$ ».

Les opérations élémentaires de clavier sont faciles à traduire en transitions de manière à respecter cet invariant. Écrire une lettre revient à l’empiler, effectuer un retour arrière revient à supprimer la tête de pile, et exécuter une flèche gauche revient à dépiler la tête de pile et lire la lettre correspondante.

Après avoir deviné la suite de touches, l’automate dépile tous les éléments de sa pile en les lisant. On a alors lu le miroir du mot obtenu en exécutant cette suite de touches. Il suffit alors de construire un automate à pile reconnaissant le miroir du précédent.

Voir preuve en B.2.4

Théorème 3.2.13

Le problème du mot est décidable en temps polynomial sur les claviers de GREK.

Preuve

Soit $K = (T, F)$ un clavier de GREK, soit $u \in A^*$. Par le Théorème 3.2.12, il suffit de construire l’automate $\mathcal{A}(K)$, et de vérifier que u appartient à son langage. L’automate $\mathcal{A}(K)$ étant constructible en temps polynomial, et l’appartenance d’un mot au langage d’un automate à pile étant de complexité polynomiale, ce problème est également résolvable en temps polynomial. \square

Finalement, on obtient la figure 3.3 (Hiérarchie des langages GREK) qui résume les inclusions des classes étudiées dans cette section.

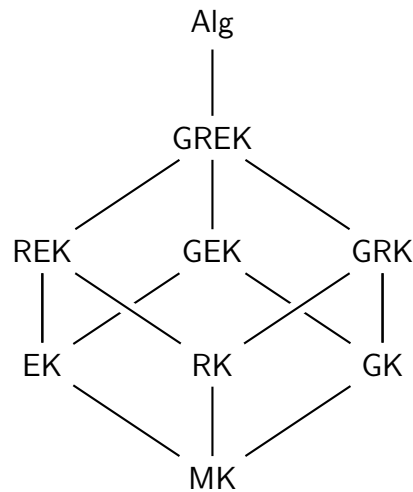


Figure 3.3. – Hiérarchie des langages GREK

3.3. Les claviers sans retour

3.3.1. FK

Théorème 3.3.1

Si L est fini et dans FK alors $L \subset \{\varepsilon\}$.

Voir preuve en B.3.1

Corollaire 3.3.2

$EK \not\subset FK$ et $RK \not\subset FK$ car les langages finis sont exprimables dans EK et dans RK.

On en déduit également $FK \subsetneq FEK$ et $FK \subsetneq FRK$ car les langages finis sont dans EK et dans RK.

Lemme 3.3.3

Le langage $L = \{ab^{n+1}a^n \mid n \in \mathbb{N}\}$ est dans FK mais pas dans GREK.

Preuve

Considérons le clavier de FK à une touche $\{a\blacktriangleleft^2\blacktriangleright b\}$. En l'appliquant sur $\langle\varepsilon|\varepsilon\rangle$ on obtient $\langle ab|\varepsilon\rangle$, et en l'appliquant sur une configuration de la forme $\langle ub|v\rangle$ on obtient $\langle ubb|av\rangle$. Le mot obtenu en l'appliquant n fois sur $\langle\varepsilon|\varepsilon\rangle$ est donc $ab^{n+1}a^n$, ce clavier produit donc bien le langage $\{ab^{n+1}a^n \mid n \in \mathbb{N}\}$.

La proposition 3.2.11 nous donne que $L \notin \text{GREK}$. □

Corollaire 3.3.4

On en déduit $FK \not\subset \text{GREK}$ et $GK \subsetneq FK$.

3.3.2. FEK

Proposition 3.3.5 (RK non inclus dans FEK)

Le langage L_C engendré par le clavier de RK $\{\leftarrow a\#\$, \leftarrow\leftarrow b\#\$\$\}$ n'est pas dans FEK.

Voir preuve en B.3.2

Corollaire 3.3.6

On en déduit $RK \not\subset FEK$ et $FEK \subsetneq FREK$.

Théorème 3.3.7

Tout langage de clavier sans effacement est un langage contextuel.

Voir preuve en B.3.2

Théorème 3.3.8 (Stabilité par miroir)

FEK et FK sont stables par miroir.

Voir preuve en B.3.2

3.4. Étude de l'automatisme

3.4.1. FRK

Une des grandes forces d'un clavier FRK est de permettre, grâce à la flèche et au retour arrière, de simuler des petites conditions en utilisant des « effets de bord ».

Proposition 3.4.1

La touche $\blacktriangleleft \mathbf{a} \blacktriangleright \blacktriangleleft \leftarrow \blacktriangleright$ permet d'écrire un a à partir d'une configuration vide et de ne pas affecter une configuration avec au moins une lettre à gauche.

Preuve

Regardons le comportement de la touche.

$$\langle \varepsilon | \varepsilon \rangle \xrightarrow{\blacktriangleleft} \langle \varepsilon | \varepsilon \rangle \xrightarrow{a} \langle a | \varepsilon \rangle \xrightarrow{\blacktriangleright} \langle a | \varepsilon \rangle \xrightarrow{\blacktriangleleft} \langle \varepsilon | a \rangle \xrightarrow{\leftarrow} \langle \varepsilon | a \rangle \xrightarrow{\blacktriangleright} \langle a | \varepsilon \rangle.$$

À partir d'une configuration $\langle ub|v \rangle$

$$\langle ub|v \rangle \xrightarrow{\blacktriangleleft} \langle u|bv \rangle \xrightarrow{a} \langle ua|bv \rangle \xrightarrow{\blacktriangleright} \langle uab|v \rangle \xrightarrow{\blacktriangleleft} \langle ua|bv \rangle \xrightarrow{\leftarrow} \langle u|bv \rangle \xrightarrow{\blacktriangleright} \langle ub|v \rangle. \quad \square$$

Remarque 3.4.2

Sur une configuration de la forme $\langle \varepsilon | bv \rangle$, cette suite de touche mène à $\langle b|v \rangle$ et n'implémente pas exactement ce qui est souhaité.

La touche $\mathbf{a} \blacktriangleright \blacktriangleleft \leftarrow \blacktriangleright$ corrige ce défaut, mais sur une configuration de la forme $\langle ub|\varepsilon \rangle$, elle retourne $\langle u|\varepsilon \rangle$. En fonction des invariants que nous voulons conserver, l'une ou l'autre de ces touches sera très utile.

La proposition suivante illustre ces mécanismes.

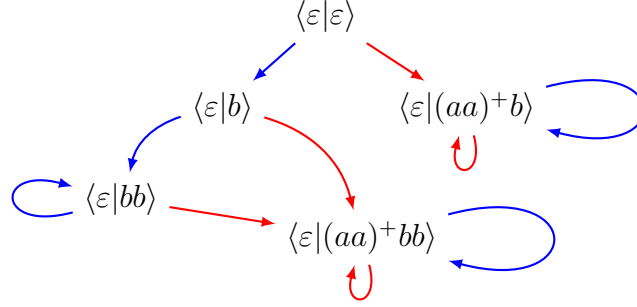
Proposition 3.4.3

Le langage $L_L = (a^2)^*(b + b^2)$, qui est dans EK mais pas dans GRK par la proposition 3.2.9, est reconnu par le clavier $K_L = \{t_1, t_2\}$ de FRK, avec $t_1 = \textcolor{blue}{b} \blacktriangleright \blacktriangleright \blacktriangleleft \blacktriangleleft \leftarrow$ et $t_2 = \textcolor{red}{b} \blacktriangleright \blacktriangleright \blacktriangleleft \blacktriangleleft \leftarrow \textcolor{red}{a} \blacktriangleleft \blacktriangleleft$.

Preuve

Pour montrer l'inclusion de $\mathcal{L}(K_L)$ dans $(a^2)^*(b + b^2)$, on considère les ensembles de configurations représentés sur le schéma suivant. Les transitions bleues désignent l'application de t_1 , les rouges celle de t_2 . La notation $\langle \varepsilon | L \rangle$ désigne l'ensemble de configuration $\{\langle \varepsilon | w \rangle \mid w \in L\}$. On vérifie facilement que depuis une configuration de chacun de ces

langages, en appliquant la touche t_i on obtient une configuration du langage atteint par la transition t_i sur le schéma. Les mots correspondant aux configurations de tous ces langages étant dans $(a^2)^*(b + b^2)$, on obtient l'inclusion.



Pour l'inclusion inverse, il nous suffit d'observer que pour tout $k \in \mathbb{N}$, $\langle \varepsilon | \varepsilon \rangle \xrightarrow{t_2^k} \langle \varepsilon | b^{2k+2} a \rangle$, $\langle \varepsilon | \varepsilon \rangle \xrightarrow{t_1 t_2^k} \langle \varepsilon | b^{2k} aa \rangle$ et $\langle \varepsilon | \varepsilon \rangle \xrightarrow{t_1} \langle \varepsilon | a \rangle \xrightarrow{t_1} \langle \varepsilon | aa \rangle$. \square

3.4.2. FREK

Théorème 3.4.4

Prenons comme alphabet $A = \{a, b, c\}$ Soit $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2$ avec $\mathcal{L}_1 = \{wc\tilde{w} \mid w \in \{a, b\}^*\}$ et $\mathcal{L}_2 = \{wcc\tilde{w} \mid w \in \{a, b\}^*\}$. Alors $\mathcal{L} \in \text{FREK} \setminus \text{FRK}$.

Voir preuve en B.4.1

Corollaire 3.4.5

$\text{FREK} \neq \text{FRK}$, à savoir tout clavier manuel n'est pas simulable par un clavier automatique.

Ce qui conclut cette petite étude des claviers.

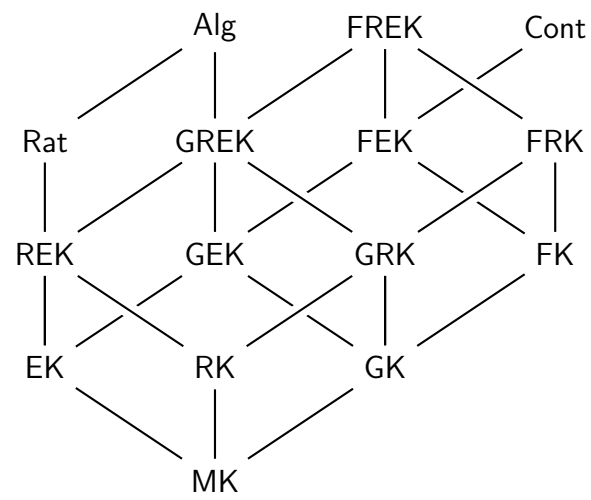


Figure 3.4. – Hiérarchie des langages

Conclusion

Les claviers paraissent à première vue extrêmement erratiques dans leur expressivité : ils reconnaissent aisément les mots de Dyck ou les palindromes mais certains langages rationnels semblent leur résister. Ils renversent notre vision usuelle des machines car ne peuvent ni lire ni avoir d'état, mais peuvent effacer et se déplacer à l'aveugle dans les mots. Une simple paire de touches de quelques caractères peut mener à un langage à première vue abominable et les opérations usuelles sur les langages n'offrent guère de propriétés de clôture.

En nous intéressant aux différentes opérations élémentaires sur les claviers, nous avons pu voir que chacune apportait une grande expressivité et que la hiérarchie des langages de clavier était stricte et contrôlée par les classes usuelles de langages (rationnels, algébriques, contextuels). Cette étude permet également de mieux comprendre les opérations qu'admettent ces différentes classes : si les automates finis suffisent en l'absence de flèches, il faut une pile pour gérer les déplacements gauche et une grammaire contextuelle pour les déplacements quelconques.

Les claviers n'ayant pas fini de nous livrer leurs secrets, de nombreuses questions restent en suspens :

- A-t-on $EK \subset FRK$? (dernière question de la hiérarchie)
- $FREK$ est-il inclus dans les algébriques ? Nous pensons que non, après regardé le clavier $\{a\blacktriangleright\blacktriangleright, b\blacktriangleleft\blacktriangleleft\}$ qui semble avoir un comportement très erratique.
- $FREK$ est-il inclus dans une classe de langages connus ? Nous pensons que les langages de clavier sont reconnaissables par des machines de TURING en espace linéaire. Il suffirait pour cela de majorer le nombre de lettres à effacer pour obtenir un mot.
- $FREK$ permet-il d'exprimer tous les langages rationnels ? Nous pensons que non et cherchons à montrer que $a^*b^*c^*$ ou $a^* + b^* + c^*$ ne sont pas de clavier.
- Caractériser la décidabilité et la complexité du problème du mot dans les différentes classes.
- Conclure sur les propriétés de clôture des différentes classes de clavier. Nous pensons que ces langages ne sont pas stables par union, intersection, complémentaire, concaténation. Le premier espoir de stabilité est selon nous le miroir mais l'absence de touche de suppression à droite empêche de trivialiser le problème.
- Étudier d'autres extensions, sur lesquelles nous ne nous sommes pas penchés pour le moment : claviers de Büchi (avec une sémantique à définir proprement), claviers avec états, etc.

Bibliographie

- [1] Paul ERDOS et Ronald L GRAHAM. « On a linear diophantine problem of Frobenius ». In : *Acta Arith* 21.1 (1972), p. 399-408.
- [2] Jean GALLIER. « The Frobenius Coin Problem Upper Bounds on The Frobenius Number ». In : (2014).
- [3] Petr JANČAR, Frantiek MRÁZ et Martin PLÁTEK. « Forgetting automata and the Chomsky hierarchy ». In : *in Proc. SOFSEM'92*. Citeseer. 1993.
- [4] Carton OLIVIER. *Langages formels, calculabilité et complexité / Olivier Carton ; préface de Dominique Perrin*. fre. Paris : Vuibert, 2008.

A. Propriétés des claviers

A.1. Généralités

A.1.1. Preuve du lemme 2.1.1 (Localité)

Lemme 2.1.1 (Localité)

Soient $\langle u|v \rangle$ une configuration et $t = \sigma_1 \dots \sigma_n$ une touche. En notant $\langle u_n|v_n \rangle = \langle u|v \rangle \cdot t$, on a que $u[1, |u| - n]$ est un préfixe de u_n et que $v[n + 1, |v|]$ est un suffixe de v_n .

Preuve

Soit $t = \sigma_1 \dots \sigma_n$. On note $\langle u_0|v_0 \rangle = \langle u|v \rangle$ et pour tout entier $0 \leq i \leq n$, on note $\langle u_i|v_i \rangle = \langle u_0|v_0 \rangle \cdot \sigma_1 \dots \sigma_i$ (en particulier, pour $0 < i \leq n$, $\langle u_i|v_i \rangle = \langle u_{i-1}|v_{i-1} \rangle \cdot \sigma_i$).

Notons de plus pour tout entier i , $w_p(u, i) = u[1, |u| - i]$ et $w_s(v, i) = v[i + 1, |v|]$.

Nous allons montrer par récurrence sur i que pour tout $0 \leq i \leq n$, $w_p(u, i)$ est un préfixe de u_i et $w_s(v, i)$ est un suffixe de v_i . Le résultat est vrai pour $i = 0$ ($u_0 = u$ et $v_0 = v$).

Supposons le résultat vrai pour un entier $i \in \llbracket 0, n - 1 \rrbracket$. Alors $w_p(u, i)$ est un préfixe de u_i et $w_s(v, i)$ un suffixe de v_i . Notons que $w_p(u, i + 1)$ est un préfixe de $w_p(u, i)$ et que $w_s(v, i + 1)$ est un suffixe de $w_s(v, i)$.

Considérons l'action de σ_{i+1} sur $\langle u_i|v_i \rangle$.

- Si $\sigma_{i+1} = a \in A$, alors $u_{i+1} = u_i a$ et $v_{i+1} = v_i$, et on a bien le résultat.
- Si $\sigma_{i+1} = \leftarrow$, on a $v_{i+1} = v_i$. Pour montrer que $w_p(u, i + 1)$ est un préfixe de u_{i+1} , on distingue deux cas.
 - Si u_i est vide, alors $w_p(u, i + 1)$ est vide (car $w_p(u, i)$ est un préfixe de u_i).
 - Sinon, u_i est de la forme $w_p(u, i)u'c$ et donc $u_{i+1} = w_p(u, i)u'$.Dans les deux cas, $w_p(u, i + 1)$ est bien un préfixe de u_{i+1} .
- Si $\sigma_{i+1} = \blacktriangleleft$, on distingue deux cas.
 - Si u_i est vide, alors $u_{i+1} = u_i$ et $v_{i+1} = v_i$.
 - Sinon, u_i est de la forme $w_p(u, i)u'c$ donc $u_{i+1} = w_p(u, i)u'$ et $v_{i+1} = cv_i$.Dans les deux cas, on a le résultat.
- Si $\sigma_{i+1} = \blacktriangleright$, on distingue deux cas.
 - Si v_i est vide, alors $u_{i+1} = u_i$ et $v_{i+1} = v_i$.
 - Sinon, v_i est de la forme $cv'w_s(v, i)$ donc $u_{i+1} = u_i c$ et $v' = v'w_s(v, i)$.Dans les deux cas, on a le résultat.

Le résultat s'obtient alors en $i = n$. □

A.1.2. Preuve du lemme 2.1.2 (Efficience loin des bords)

Lemme 2.1.2 (Efficience loin des bords)

Soient $t = \sigma_1 \dots \sigma_n$ une touche et $\langle u|v \rangle$ une configuration. Si $n \leq \min(|u|, |v|)$, alors en notant $\langle u_n|v_n \rangle = \langle u|v \rangle \cdot t$, on a que $\langle u|v \rangle \xrightarrow{t}_e \langle u_n|v_n \rangle$ c'est-à-dire que l'exécution efficace ne bloque pas.

Preuve

Procédons par récurrence sur n . Si $n = 0$, alors la propriété est triviale.

Supposons le résultat vrai pour un entier n . Soient $t = \sigma_1 \dots \sigma_{n+1}$ une suite d'opérations élémentaires $\langle u|v \rangle$ une configuration telle que $n + 1 \leq \min(|u|, |v|)$. On note t' la suite d'opérations élémentaires $\sigma_1 \dots \sigma_n$. Par hypothèse de récurrence, il existe $\langle u_n|v_n \rangle$ telle que $\langle u|v \rangle \xrightarrow{t'}_e \langle u_n|v_n \rangle$.

Le lemme de localité nous donne de plus que u_n et v_n contiennent au moins un caractère. Ainsi u_n est de la forme $u'a$ et v_n est de la forme bv' avec $a, b \in A$. On a

$$\begin{aligned} \langle u_n|v_n \rangle &\xrightarrow{c}_e \langle u_nc|v_n \rangle \text{ si } c \in A. \\ \langle u_n|v_n \rangle &\xleftrightarrow{e} \langle u'|v_n \rangle. \\ \langle u_n|v_n \rangle &\xrightarrow{\blacktriangleleft}_e \langle u'|av_n \rangle. \\ \langle u_n|v_n \rangle &\xrightarrow{\blacktriangleright}_e \langle u_nb|v' \rangle. \end{aligned}$$

Ainsi, peu importe la valeur de σ_{n+1} , il existe une configuration $\langle u_{n+1}|v_{n+1} \rangle$ telle que $\langle u_n|v_n \rangle \xrightarrow{\sigma_{n+1}}_e \langle u_{n+1}|v_{n+1} \rangle$ et donc telle que $\langle u|v \rangle \xrightarrow{t}_e \langle u_{n+1}|v_{n+1} \rangle$.

Et donc on a bien le résultat. \square

A.1.3. Preuve du lemme 2.1.3 (Encadrement des tailles)

Lemme 2.1.3 (Encadrement des tailles)

Soient $t = \sigma_1 \dots \sigma_n$ une touche et $\langle u|v \rangle$ une configuration. On pose $\langle u_n|v_n \rangle = \langle u|v \rangle \cdot t$. Alors

$$|uv| - |t|_{\leftarrow} + \sum_{x \in A} |t|_x \leq |u_nv_n| \leq |uv| + \sum_{x \in A} |t|_x.$$

On en déduit notamment que $||u_nv_n| - |uv|| \leq n$.

Preuve

Procédons par induction sur n . Si $n = 0$, la propriété est vraie.

Supposons la propriété vraie pour un entier n . Soit $t = \sigma_1 \dots \sigma_{n+1}$ une suite d'opérations élémentaires. On note t' la suite $\sigma_1 \dots \sigma_n$, $\langle u_n|v_n \rangle = \langle u|v \rangle \cdot t'$ et $\langle u_{n+1}|v_{n+1} \rangle = \langle u|v \rangle \cdot t$.

On pose

$$\begin{aligned} M_n &= |uv| - |t|_{\leftarrow} + \sum_{x \in A} |t'|_x & N_n &= |uv| + \sum_{x \in A} |t'|_x \\ M_{n+1} &= |uv| - |t|_{\leftarrow} + \sum_{x \in A} |t|_x & N_{n+1} &= |uv| + \sum_{x \in A} |t|_x \end{aligned}$$

Par hypothèse de récurrence, $M_n \leq |u_n v_n| \leq N_n$. On distingue alors ces différents cas.

- Si $\sigma_{n+1} = a \in A$ alors $M_{n+1} = M_n + 1$, $N_{n+1} = N_n + 1$ et $|u_{n+1} v_{n+1}| = |u_n v_n| + 1$.
- Si $\sigma_{n+1} \in \{\blacktriangleright, \blacktriangleleft\}$ alors $M_{n+1} = M_n$, $N_{n+1} = N_n$ et $|u_{n+1} v_{n+1}| = |u_n v_n|$.
- Si $\sigma_{n+1} = \leftarrow$ alors $M_{n+1} = M_n - 1$, $N_{n+1} = N_n$ et $u_{n+1} v_{n+1}$ est égal à $|u_n v_n|$ si u_n est vide et à $|u_n v_n| - 1$ sinon.

Dans tous les cas, on a bien $M_{n+1} \leq |u_{n+1} v_{n+1}| \leq N_{n+1}$. \square

A.1.4. Preuve du lemme 2.1.4 (Égalité des tailles loin des bords)

Lemme 2.1.4 (Égalité des tailles loin des bords)

Soient $t = \sigma_1 \dots \sigma_n$ une touche et $\langle u|v \rangle$ une configuration telle que $|u| \geq n$ et $|v| \geq n$. En posant $\langle u_n|v_n \rangle = \langle u|v \rangle \cdot t$, alors

$$|u_n v_n| = |uv| - |t|_{\leftarrow} + \sum_{x \in A} |t|_x.$$

Preuve

Procédons par induction sur n . Si $n = 0$ alors la propriété est triviale.

Supposons la propriété vraie pour un entier n . Soient $t = \sigma_1 \dots \sigma_{n+1}$ une suite d'opérations élémentaires et $\langle u|v \rangle$ une configuration telle que $|u| \geq n + 1$ et $|v| \geq n + 1$. On note t' la suite $\sigma_1 \dots \sigma_n$, $\langle u_n|v_n \rangle = \langle u|v \rangle \cdot t'$ et $\langle u_{n+1}|v_{n+1} \rangle = \langle u|v \rangle \cdot t$.

Le lemme de localité nous donne que u_n et v_n contiennent au moins un caractère. Ainsi u_n est de la forme $u'a$ et v_n est de la forme bv' , avec $a, b \in A$. On a

$$\begin{cases} u_{n+1} = u_n c & v_{n+1} = v_n & \text{si } \sigma_{n+1} = c \in A \\ u_{n+1} = u' & v_{n+1} = v_n & \text{si } \sigma_{n+1} = \leftarrow \\ u_{n+1} = u' & v_{n+1} = a v_n & \text{si } \sigma_{n+1} = \blacktriangleleft \\ u_{n+1} = u_n b & v_{n+1} = v' & \text{si } \sigma_{n+1} = \blacktriangleright \end{cases}$$

L'hypothèse d'induction appliquée à t' et à $\langle u_n|v_n \rangle$, à savoir

$$|u_n v_n| = |uv| - |t'|_{\leftarrow} + \sum_{x \in A} |t'|_x$$

nous permet de conclure et d'obtenir le résultat :

$$|u_{n+1} v_{n+1}| = |uv| - |t|_{\leftarrow} + \sum_{x \in A} |t|_x.$$

Ceci conclut notre récurrence et nous donne le résultat. \square

A.1.5. Preuve de la proposition 2.1.5

Proposition 2.1.5

Soit K un clavier automatique de langage L . Notons $(l_n)_{n \in \mathbb{N}}$ les longueurs des mots reconnus par L triés par ordre croissant. Alors $(l_{n+1} - l_n)_{n \in \mathbb{N}}$ est bornée par $\|K\|_\infty$.

Preuve

Si pour tout mot $w \in L$, $|w| \leq \|K\|_\infty$, la propriété est trivialement vraie.

Sinon, on considère $w \in L$ tel que $|w| > \|K\|_\infty$. Le lemme 2.1.3 (Encadrement des tailles) nous assure qu'une exécution menant à w est au moins de taille 2. En effet, si on avait w en une seule touche t , on aurait

$$|w| \leq |\varepsilon| + \sum_{a \in A} |t|_a \leq \|K\|_\infty.$$

Ainsi, il existe $t_0 \dots t_{n+1}$ ($n \in \mathbb{N}$) telle que $\langle \varepsilon | \varepsilon \rangle \cdot t_0 \dots t_{n+1} = \langle u | v \rangle$ et $w = uv$. Pour tout $i \in \llbracket 0, n+1 \rrbracket$, on pose

$$\begin{cases} \langle u_i | v_i \rangle = \langle \varepsilon | \varepsilon \rangle \cdot t_1 \dots t_i \\ w_i = u_i v_i \end{cases}.$$

Les w_i sont dans L (car K est automatique) et le lemme 2.1.3 (Encadrement des tailles) nous assure que pour tout $i \in \llbracket 0, n \rrbracket$, $||w_{i+1}| - |w_i|| \leq \|K\|_\infty$. De plus, $0 \leq |w_0| \leq \|K\|_\infty$ et $|w_n| = |w| > \|K\|_\infty$, donc il existe $k \in \llbracket 0, n-1 \rrbracket$ tel que $w_k \neq w$ et

$$|w| - \|K\|_\infty \leq |w_k| \leq |w|.$$

Ainsi, pour tout $w \in L$ de taille strictement supérieure à $\|K\|_\infty$, il existe $w' \in L$ tel que $w' \neq w$ et

$$|w| - \|K\|_\infty \leq |w'| \leq |w|,$$

ce qui démontre la proposition. □

A.1.6. Preuve de la proposition 2.1.7

Proposition 2.1.7

Soit $K = (T, F)$ un clavier de langage L . Notons $(l_n)_{n \in \mathbb{N}}$ les longueurs des mots reconnus par L triés par ordre croissant. Alors $(l_{n+1} - l_n)_{n \in \mathbb{N}}$ est bornée par $5\|K\|_\infty$.

Preuve

Si pour tout $w \in L$, $|w| \leq 5\|K\|_\infty$, alors le résultat est vrai.

Sinon, on considère w tel que $|w| > 5\|K\|_\infty$. Le lemme 2.1.3 (Encadrement des tailles) assure qu'on ne peut pas obtenir w en appuyant sur une seule touche. Ainsi, en voyant T comme un clavier automatique tel que $\|T\|_\infty \leq \|K\|_\infty$, on a qu'il existe $w' \in \mathcal{L}(L')$, $f \in F$ et deux configuration $\langle u | v \rangle$ et $\langle u' | v' \rangle$ tels que

$$w = uv, w' = u'v' \text{ et } \langle u' | v' \rangle \cdot f = \langle u | v \rangle.$$

Par le [lemme 2.1.3 \(Encadrement des tailles\)](#), on a $|w'| > 4\|K\|_\infty$. La [proposition 2.1.5](#) nous donne alors qu'il existe $w'' \in \mathcal{L}(T)$ tel que

$$|w'| - 3\|K\|_\infty \leq |w''| < |w'| - 2\|K\|_\infty.$$

En particulier, il existe $\tau \in T^*$ tel que $\langle \varepsilon | \varepsilon \rangle \cdot \tau = \langle u'' | v'' \rangle$ avec $u''v'' = w''$. Posons alors $\langle x | y \rangle = \langle u'' | v'' \rangle \cdot \tau \cdot f$. On a $xy \in L$ et le [lemme 2.1.3 \(Encadrement des tailles\)](#) nous assure

$$|w'| - 4\|K\|_\infty \leq |xy| < |w'| - \|K\|_\infty$$

et donc

$$|w| - 5\|K\|_\infty \leq |xy| < |w|.$$

Ainsi, pour tout $w \in L$ de taille strictement supérieure à $5\|K\|_\infty$, il existe $w' \in L$ tel que $|w| - 5\|K\|_\infty \leq |w'| < |w|$, ce qui démontre la proposition. \square

A.1.7. Preuve du [lemme 2.1.12](#)

Lemme 2.1.12

Soient $t = \sigma_1 \dots \sigma_n$ une touche, et $u, v, u', v' \in A^*$ de longueurs supérieures ou égales à n et ne contenant pas de a . En posant $\langle u_n | v_n \rangle = \langle u | v \rangle \cdot t$ et $\langle u'_n | v'_n \rangle = \langle u' | v' \rangle \cdot t$, on a

$$I_a(u_n, v_n) = I_a(u'_n, v'_n).$$

Preuve

Soient $t = \sigma_1 \dots \sigma_n$ une suite d'opérations élémentaires, et $u, v, u', v' \in A^*$ de longueurs supérieures ou égales à n et ne contenant pas de a . Pour tout $i \in \llbracket 0, n \rrbracket$, on pose

$$\begin{aligned} \langle u_i | v_i \rangle &= \langle u | v \rangle \cdot t_1 \dots t_i & \langle u'_i | v'_i \rangle &= \langle u' | v' \rangle \cdot t_1 \dots t_i \\ G(i) &= G_a(u_i, v_i) & G'(i) &= G_a(u'_i, v'_i) \\ D(i) &= D_a(u_i, v_i) & D'(i) &= D_a(u'_i, v'_i) \\ I(i) &= I_a(u_i, v_i) & I'(i) &= I_a(u'_i, v'_i) \end{aligned}$$

Si X est un sous-ensemble de \mathbb{Z} , on note $X_{+k} = \{x + k \mid x \in X\}$ pour $k \in \mathbb{Z}$.

Nous allons montrer par récurrence sur i que pour tout $i \in [0, n]$, $I(i) = I'(i)$. Le résultat est vrai pour $i = 0$: $u_0 = u$, $v_0 = v$, $u'_0 = u'$, $v'_0 = v'$ et aucun d'eux ne contient de a .

Soit $i \in \llbracket 0, n-1 \rrbracket$. On suppose le résultat vrai pour i , c'est-à-dire $I(i) = I'(i)$, et donc $G(i) = G'(i)$ et $D(i) = D'(i)$.

Puisque les tailles de u, v, u' et v' sont supérieures à n , le [lemme 2.1.1 \(Localité\)](#) nous assure que u_i, v_i, u'_i et v'_i sont non vides. Notons b_u la dernière lettre de u_i et b_v la première lettre de v_i . Nous distinguons alors plusieurs cas.

— Si $\sigma_{i+1} = b \in A \setminus \{a\}$, alors

$$\begin{cases} G(i+1) = G(i)_{-1} = G'(i)_{-1} = G'(i+1) \\ D(i+1) = D(i) = D'(i) = D'(i+1) \end{cases}$$

— Si $\sigma_{n+1} = a$, alors

$$\begin{cases} G(i+1) = G(i)_{-1} \cup \{-1\} = G'(i)_{-1} \cup \{-1\} = G'(i+1) \\ D(i+1) = D(i) = D'(i) = D'(i+1) \end{cases}$$

— Si $\sigma_{n+1} = \blacktriangleleft$ et $b_u = a$, alors

$$\begin{cases} G(i+1) = G(i)_{+1} \setminus \{0\} = G'(i)_{+1} \setminus \{0\} = G'(i+1) \\ D(i+1) = D(i)_{+1} \cup \{1\} = D'(i)_{+1} \cup \{1\} = D'(i+1) \end{cases}$$

— Si $\sigma_{n+1} = \blacktriangleleft$ et $b_u \neq a$, alors

$$\begin{cases} G(i+1) = G(i)_{+1} = G'(i)_{+1} = G'(i+1) \\ D(i+1) = D(i)_{+1} = D'(i)_{+1} = D'(i+1) \end{cases}$$

— Si $\sigma_{n+1} = \blacktriangleright$ et $b_u = a$, alors

$$\begin{cases} G(i+1) = G(i)_{-1} \cup \{-1\} = G'(i)_{-1} \cup \{-1\} = G'(i+1) \\ D(i+1) = D(i)_{-1} \setminus \{0\} = D'(i)_{-1} \setminus \{0\} = D'(i+1) \end{cases}$$

— Si $\sigma_{n+1} = \blacktriangleright$ et $b_u \neq a$, alors

$$\begin{cases} G(i+1) = G(i)_{-1} = G'(i)_{-1} = G'(i+1) \\ D(i+1) = D(i)_{-1} = D'(i)_{-1} = D'(i+1) \end{cases}$$

— Si $\sigma_{n+1} = \blacktriangleleft$ et $b_u = a$, alors

$$\begin{cases} G(i+1) = G(i)_{+1} \setminus \{0\} = G'(i)_{+1} \setminus \{0\} = G'(i+1) \\ D(i+1) = D(i) = D'(i) = D'(i+1) \end{cases}$$

— Si $\sigma_{n+1} = \blacktriangleleft$ et $b_u \neq a$, alors

$$\begin{cases} G(i+1) = G(i)_{+1} = G'(i)_{+1} = G'(i+1) \\ D(i+1) = D(i) = D'(i) = D'(i+1) \end{cases}$$

Dans tous les cas, on a bien $I(i) = I'(i)$ ce qui permet de conclure notre récurrence.

Le lemme s'obtient alors en $i = n$. □

A.1.8. Preuve du **lemme 2.1.15**

Lemme 2.1.15

Soient $t = \sigma_1 \dots \sigma_n$ une touche et $u, v \in A^*$ de longueurs supérieures ou égales à n . Si $|\langle u|v \rangle|_a < |\langle u|v \rangle \cdot t|_a$, alors t écrit un a .

Le but ici est de prouver le plus formellement possible cette proposition qui est très intuitive.

Preuve

Soit $\langle u'|v' \rangle = \langle u|v \rangle \cdot t$. On pose $t = \sigma_1 \dots \sigma_n$.

Soit $t' = \sigma'_1 \dots \sigma'_n$ la touche où les occurrences de a sont remplacées par une lettre distinguée $\hat{a} \notin A$, et λ le morphisme de mots $[\hat{a} \mapsto a]$.

On va construire récursivement les ensembles $G_a(n)$ et $D_a(n)$ des a ajoutées à gauche et à droite par t .

Pour $i = 0$, on pose $G_a(0) = D_a(0) = \emptyset$.

Si $i > 0$,

- Si $\sigma_i = b \neq a$, on pose $G_a(i) = G_a(i-1)_{-1}$ et $D_a(i) = D_a(i-1)$
- Si $\sigma_i = a$, on pose $G_a(i) = G_a(i-1)_{-1} \cup \{-1\}$ et $D_a(i) = D_a(i-1)$
- Si $\sigma_i = \blacktriangleleft$ et $-1 \notin G_a(i-1)$, on pose $G_a(i) = G_a(i-1)_{+1}$ et $D_a(i) = D_a(i-1)_{+1}$
- Si $\sigma_i = \blacktriangleleft$ et $-1 \in G_a(i-1)$, on pose $G_a(i) = G_a(i-1)_{+1} \setminus \{0\}$ et $D_a(i) = D_a(i-1)_{+1} \cup \{1\}$
- Si $\sigma_i = \blacktriangleright$ et $1 \notin D_a(i-1)$, on pose $G_a(i) = G_a(i-1)_{-1}$ et $D_a(i) = D_a(i-1)_{-1}$
- Si $\sigma_i = \blacktriangleright$ et $1 \in D_a(i-1)$, on pose $G_a(i) = G_a(i-1)_{-1} \cup \{-1\}$ et $D_a(i) = D_a(i-1)_{-1} \setminus \{0\}$
- Si $\sigma_i = \blackleftarrow$ et $-1 \notin G_a(i-1)$, on pose $G_a(i) = G_a(i-1)_{+1}$ et $D_a(i) = D_a(i-1)$
- Si $\sigma_i = \blackleftarrow$ et $-1 \in G_a(i-1)$, on pose $G_a(i) = G_a(i-1)_{+1} \setminus \{0\}$ et $D_a(i) = D_a(i-1)_{+1}$

Et on note de même $G_{\hat{a}}(i)$ et $D_{\hat{a}}(i)$ pour t' .

Il est rapide de vérifier par une récurrence sur i qu'on a les propriétés suivantes :

- pour tout $i \in \llbracket 1; n \rrbracket$, $G_a(i) = G_{\hat{a}}(i)$ et $D_a(i) = D_{\hat{a}}(i)$
- pour tout $i \in \llbracket 1; n \rrbracket$, $G_{\hat{a}}(i) = G_{\hat{a}}(\langle u|v \rangle \cdot (\sigma'_1 \dots \sigma'_i))$ et $D_{\hat{a}}(i) = D_{\hat{a}}(\langle u|v \rangle \cdot (\sigma'_1 \dots \sigma'_i))$
- pour tout $i \in \llbracket 1; n \rrbracket$, $|G_a(i) \cup D_a(i)| + |uv|_a \geq |\langle u|v \rangle \cdot (\sigma_1 \dots \sigma_i)|_a$

On en déduit que

$$|G_{\hat{a}}(i) \cup D_{\hat{a}}(i)| \geq |\langle u|v \rangle \cdot t|_a - |uv|_a > 0$$

□

Or $I_{\hat{a}}(\langle u|v \rangle \cdot t') = G_{\hat{a}}(i) \cup D_{\hat{a}}(i)$ car l'exécution est efficace, donc $I_{\hat{a}}(\langle u|v \rangle \cdot t') \neq \emptyset$.

On peut donc appliquer le [lemme 2.1.12](#) sur la configuration $\langle b^{\|K\|_\infty} | b^{\|K\|_\infty} \rangle$. On a $I_{\hat{a}}(\langle b^{\|K\|_\infty} | b^{\|K\|_\infty} \rangle \cdot t') \neq \emptyset$, d'où t' écrit un \hat{a} , et donc t écrit un a .

A.1.9. Preuve du [lemme 2.1.18 \(Monotonie\)](#)

Lemme 2.1.18 (Monotonie)

Soient $u, v, u', v' \in A^*$ et $t \in S^*$ une touche. Notons $\langle x|y \rangle = \langle u|v \rangle \cdot t$ et $\langle x'|y' \rangle = \langle u'|v' \rangle \cdot t$. Si $|u| \leq |u'|$ et $|v| \leq |v'|$, alors $|x| \leq |x'|$ et $|y| \leq |y'|$.

Preuve

On procède par induction sur la touche t . Si $t = \varepsilon$, le résultat est vrai.

Sinon, $t = t'\sigma$, alors en notant $\langle \alpha|\beta \rangle = \langle u|v \rangle \cdot t'$ et $\langle \alpha'|\beta' \rangle = \langle u'|v' \rangle \cdot t'$, on a, par l'hypothèse d'induction,

$$|\alpha| \leq |\alpha'| \text{ et } |\beta| \leq |\beta'|.$$

Nous analysons les différents cas possibles.

- Si $\sigma = a \in A$ alors $x = \alpha a$, $x' = \alpha' a$, $y = \beta$ et $y' = \beta'$ donc les inégalités sont bien préservées.
- Si $\sigma = \leftarrow$, on a $y = \beta$ et $y' = \beta'$. Pour le côté gauche, on distingue trois cas.
 - Si $\alpha' = \varepsilon$, alors $\alpha = \varepsilon$ et donc $x' = x = \varepsilon$.
 - Si $\alpha = \varepsilon$ et $\alpha' \neq \varepsilon$, alors $|x| = 0 \leq |x'|$.
 - Si $\alpha \neq \varepsilon$ et $\alpha' \neq \varepsilon$, alors $|x| = |\alpha| - 1$ et $|x'| = |\alpha'| - 1$.Les inégalités sont bien préservées.
- Si $\sigma = \blacktriangleleft$, on distingue trois cas.
 - Si $\alpha' = \varepsilon$, alors $\alpha = \varepsilon$ et donc $x' = x = \varepsilon$, $y = \beta$ et $y' = \beta'$.
 - Si $\alpha = \varepsilon$ et $\alpha' \neq \varepsilon$, alors on a $|x| = 0 \leq |x'|$ du côté droit et $|y| = |\beta|$ et $|y'| = |\beta'| + 1$ du côté gauche.
 - Si $\alpha \neq \varepsilon$ et $\alpha' \neq \varepsilon$, alors on a $|x| = |\alpha| - 1$ et $|x'| = |\alpha'| - 1$ du côté droit et $|y| = |\beta| + 1$ et $|y'| = |\beta'| + 1$ du côté gauche.Les inégalités sont bien préservées.
- Si $\sigma = \blacktriangleright$, on distingue trois cas.
 - Si $\beta' = \varepsilon$, alors $\beta = \varepsilon$ et donc $y' = y = \varepsilon$, $x = \alpha$ et $x' = \alpha'$.
 - Si $\beta = \varepsilon$ et $\beta' \neq \varepsilon$, alors on a $|y| = 0 \leq |y'|$ du côté droit et $|x| = |\alpha|$ et $|x'| = |\alpha'| + 1$ du côté gauche.
 - Si $\beta \neq \varepsilon$ et $\beta' \neq \varepsilon$, alors on a $|y| = |\beta| - 1$ et $|y'| = |\beta'| - 1$ du côté droit et $|x| = |\alpha| + 1$ et $|x'| = |\alpha'| + 1$ du côté gauche.Et les inégalités sont bien préservées.

Ceci permet de conclure notre induction et donc notre preuve. \square

A.2. Les claviers sans flèches

A.2.1. Preuve du

lemme 2.2.1 (Caractérisation des configurations REK)

Lemme 2.2.1 (Caractérisation des configurations REK)

Soient K un clavier de REK, $u \in A^*$ et $t_1 \dots t_n \in K^*$. Alors $\langle u|\varepsilon \rangle \cdot t_1 \dots t_n$ est de la forme $\langle v|\varepsilon \rangle$ avec $v \in A^*$.

Preuve

On montre le résultat pour les actions élémentaires, et on l'obtient sur les touches par une induction immédiate sur la longueur de la touche. L'action d'une suite de touches τ étant équivalent à celle de la concaténation des touches de τ , on aura le résultat.

Les actions élémentaires qui sont faisables ici sont le retour arrière et l'écriture d'une lettre a et on a

$$\begin{aligned}\langle u|\varepsilon\rangle \cdot a &= \langle ua|\varepsilon\rangle \\ \langle u|\varepsilon\rangle \cdot \leftarrow &= \begin{cases} \langle u|\varepsilon\rangle & \text{si } u = \varepsilon \\ \langle v|\varepsilon\rangle & \text{si } u = vb \end{cases}\end{aligned}$$

ce qui montre le résultat pour les actions élémentaires. \square

A.2.2. Preuve du lemme 2.2.2 (Forme normale d'une touche REK)

Lemme 2.2.2 (Forme normale d'une touche REK)

Soit $t \in S^*$ une suite d'opérations élémentaires de REK. Alors il existe un unique $k \in \mathbb{N}$ et un unique $w \in A^*$ tels que $t \sim \leftarrow^k w$. De plus, k et w peuvent être trouvées en temps polynomial en la taille de t .

Preuve

Supposons que t est de la forme $t_1 a \leftarrow t_2$ avec $t_1, t_2 \in S^*$ et $a \in A$. Soit u une configuration. Notons $v = u \cdot t_1$. On a

$$\begin{aligned}u \cdot t &= ((v \cdot a) \cdot \leftarrow) \cdot t_2 \\ &= (va \cdot \leftarrow) \cdot t_2 \\ &= v \cdot t_2 \\ &= (u \cdot t_1) \cdot t_2 = u \cdot t_1 t_2.\end{aligned}$$

Ainsi, si t contient une lettre suivie d'un retour, il existe une touche équivalente, strictement plus courte, sans cette séquence d'opérations. Il existe donc une touche t' de même action que t sans aucune séquence d'opérations de la forme $a \leftarrow$, c'est-à-dire de la forme $\leftarrow^k w$.

Nous pouvons bien trouver la touche correspondante en temps polynomial. En effet, nous parcourons la touche t jusqu'à trouver une lettre suivie d'un retour. S'il n'y en a pas, on retourne la touche, sinon on supprime cette séquence et on réitère l'opération.

À chaque étape, la taille de la touche considérée diminue d'au moins 1, et donc il y a au plus $|t|$ étapes avec, à l'étape k , le parcours d'une touche de taille inférieure ou égale à $|t| - k$. On majore donc notre nombre d'opérations par

$$\sum_{k=0}^{|t|} |t| - k = O(|t|^2).$$

\square

A.3. Les claviers sans flèche droite

A.3.1. Preuve du **théorème 2.3.1 (fondamental de GREK)**

Théorème 2.3.1 (fondamental de GREK)

Soient $t = \sigma_1 \cdots \sigma_n$ une suite d'opérations élémentaires et $\langle u|v \rangle$ une configuration. Nous posons $\langle x_n|y_n \rangle = \langle \varepsilon|\varepsilon \rangle \cdot t$. Alors $\langle u|v \rangle \cdot t$ est de la forme $\langle u_n x_n|v_n v \rangle$ avec y_n un sous-mot de v_n et u_n un préfixe de u .

Preuve

Procédons par induction sur n . Si $n = 0$ alors la propriété est trivialement vraie.

Si $n > 0$, posons $\langle x_{n-1}|y_{n-1} \rangle = \langle \varepsilon|\varepsilon \rangle \cdot \sigma_1 \cdots \sigma_{n-1}$. Soit $\langle u|v \rangle$ une configuration. On pose $c_n = \langle u|v \rangle \cdot \sigma_1 \cdots \sigma_n$. Par hypothèse d'induction, il existe u_{n-1} et v_{n-1} tels que

$$c_{n-1} = \langle u_{n-1} x_{n-1} | v_{n-1} v \rangle$$

avec y_{n-1} un sous-mot de v_{n-1} et u_{n-1} un préfixe de u .

On distingue alors plusieurs cas.

- Si $\sigma_n = a \in A$, alors $x_n = x_{n-1}a$ et $y_n = y_{n-1}$. On pose $u_n = u_{n-1}$ et $v_n = v_{n-1}$.
- Si $\sigma_n = \leftarrow$ et $x_{n-1} = \varepsilon$, alors $x_n = \varepsilon$ et $y_n = y_{n-1}$. On pose $v_n = v_{n-1}$ et

$$u_n = \begin{cases} u'_{n-1} & \text{si } u_{n-1} = u'_{n-1}a \ (a \in A) \\ \varepsilon & \text{sinon} \end{cases}.$$

- Si $\sigma_n = \leftarrow$ et $x_{n-1} = x'_{n-1}a$ avec $a \in A$, alors $x_n = x'_{n-1}$ et $y_n = y_{n-1}$. On pose $u_n = u_{n-1}$ et $v_n = v_{n-1}$.
- Si $\sigma_n = \blacktriangleleft$ et $x_{n-1} = \varepsilon$, alors $x_n = \varepsilon$ et $y_n = y_{n+1}$. On distingue deux cas.
 - Si $u_{n-1} = \varepsilon$, on pose $u_n = \varepsilon$ et $v_n = v_{n-1}$.
 - Si $u_{n-1} = u'_{n-1}a$ avec $a \in A$, on pose $u_n = u'_{n-1}$ et $v_n = av_{n-1}$.
- Si $\sigma_n = \blacktriangleleft$ et $x_{n-1} = x'_{n-1}a$ avec $a \in A$, alors $x_n = x'_{n-1}$ et $y_n = ay_{n-1}$. On pose $u_n = u_{n-1}$ et $v_n = av_{n-1}$.

Dans tous les cas, on a bien $c_n = \langle u_n x_n | v_n v \rangle$ avec y_n sous-mot de v_n et u_n préfixe de u .

L'induction est démontrée. \square

A.3.2. Preuve du **lemme 2.3.2 (Insensibilité à la position)**

Lemme 2.3.2 (Insensibilité à la position)

Soient \hat{a} une lettre distinguée, et t une touche de GREK contenant un unique \hat{a} . Soit $\langle u|v \rangle$ une configuration ne contenant pas de \hat{a} . Si $\langle u|v \rangle \cdot t$ contient un \hat{a} , alors pour toute configuration $\langle u'|v' \rangle$, $\langle u'|v' \rangle \cdot t$ contient \hat{a} .

Preuve

Considérons t' , le suffixe de t commençant par \hat{a} et créons P un tableau indicé de 1 à $|t'|$. Le but est d'indiquer dans la case i de P la position du curseur par rapport à \hat{a} après l'exécution des i premières opérations élémentaires de t' (notre origine est juste à droite du \hat{a}). Nous posons alors $P[1] = 0$ et pour $0 \leq i < |t|$

$$P[i+1] = \begin{cases} P[i] + 1 & \text{si } t'_i = a \in A \\ P[i] - 1 & \text{si } t'_i \in \{\blacktriangleleft, \leftarrow\} \end{cases}$$

Une récurrence immédiate sur i montre que tant que $P[i]$ ne devient pas négatif, $P[i]$ correspond bien à la position du curseur par rapport au \hat{a} . Posons alors k l'entier minimal tel que $P[k+1] < 0$ ou -1 si un tel entier n'existe pas.

Nous allons montrer que le \hat{a} est effacé si et seulement si $P[k] = 0$ et $t'_{k+1} = \leftarrow$.

\Leftarrow : si $P[k] = 0$ et $t'_{k+1} = \leftarrow$, alors on est juste à droite du \hat{a} et on efface la lettre à gauche du curseur ; on efface bien le \hat{a} .

\Rightarrow : supposons que le \hat{a} est effacé. Si $k = -1$, alors tous les éléments de P sont positifs, ce qui est impossible (on obtient un -1 après avoir effacé le \hat{a}).

Donc $k \geq 0$ et par définition de k , $P[k] \geq 0$ et $P[k+1] < 0$, d'où $t_{k+1} \in \{\blacktriangleleft, \leftarrow\}$.

Mais si $t_{k+1} = \blacktriangleleft$, alors le curseur passe à gauche du a sans l'avoir effacé ; le [théorème 2.3.1 \(fondamental de GREK\)](#) permet d'exclure cette possibilité.

Il ne reste qu'une possibilité, $t_{k+1} = \leftarrow$. On obtient $P[k] = 0$ grâce à la définition de P (l'écart entre $P[k]$ et $P[k+1]$ est de 1, $P[k+1] < 0$ et $P[k] \geq 0$).

Et on a l'équivalence.

De plus, P ne dépend que de la touche t et pas de la configuration $\langle u|v \rangle$ considérée. Ainsi, si \hat{a} appartient à $\langle u|v \rangle$, alors pour tout u', v' , \hat{a} appartient à $\langle u'|v' \rangle$. \square

A.4. Les claviers sans retour

A.4.1. Preuve du [lemme 2.4.3 \(Itération de lettre\)](#)

Lemme 2.4.3 (Itération de lettre)

Soient $L \in \text{FK}$, $w \in L$ et a une lettre apparaissant dans w . Alors il existe une suite $(w_n)_{n \in \mathbb{N}}$ de mots de L telle que $\forall n \in \mathbb{N}, |w_n|_a \geq n$.

Preuve

Soit K un clavier de FK qui reconnaît L . Il existe une suite de touches $\tau = t_1 \dots t_k$ telle que $\langle \varepsilon | \varepsilon \rangle \cdot \tau = \langle u | v \rangle$ avec $uv = w$.

Puisque a est une lettre de w , alors il existe $i \in \llbracket 1, k \rrbracket$ telle que $|t_i|_a \geq 1$. On pose alors $u_0 = v_0 = \varepsilon$, et pour tout $n \in \mathbb{N}$, $\langle u_{n+1} | v_{n+1} \rangle = \langle u_n | v_n \rangle \cdot t_i$ et $w_n = u_n v_n$.

Une récurrence sur n montre que $|w_n|_a \geq n$. Le résultat est vrai pour $n = 0$ et s'il est vrai pour $n \in \mathbb{N}$, alors le [lemme 2.4.2 \(Non effacement\)](#) et l'hypothèse de récurrence permettent de conclure :

$$|w_{n+1}|_a = |w_n|_a + |t_i|_a \geq |w_n|_a + 1 \geq n + 1.$$

Et on a le résultat. \square

A.4.2. Preuve du [lemme A.4.1 \(Itération de lettre FEK\)](#)

Lemme A.4.1 (Itération de lettre FEK)

Soit $K = (T, F)$ un clavier de FEK. S'il existe $w \in L$ tel que $|w|_a > \|K\|_\infty$, alors pour tout $n \in \mathbb{N}$, il existe $w_n \in L$ tel que $|w_n|_a \geq n$.

Preuve

Soit w tel que $|w|_a > \|K\|_\infty$. Le [lemme 2.1.3 \(Encadrement des tailles\)](#) nous donne qu'une exécution acceptant w est au moins de longueur 2 (car $|w| > \|K\|_\infty$).

Soit $\tau = t_0 \dots t_{n+1}$ une suite de touches menant à accepter w . On note w' le mot obtenu en appuyant sur $\tau' = t_0 \dots t_n$. Puisque $|w|_a > \|K\|_\infty$, alors, par le [lemme 2.4.2 \(Non effacement\)](#),

$$|w'|_a = |w|_a - |t_{n+1}|_a \geq |w|_a - |t_{n+1}| \geq |w|_a - \|K\|_\infty > 0.$$

On en déduit qu'il existe t transiente tel que $t_a > 0$. Pour tout $k \in \mathbb{N}$, on pose alors $w_k = u_k v_k$ où

$$\langle u_k | v_k \rangle = \langle \varepsilon | \varepsilon \rangle \cdot \underbrace{t \dots t}_{k \text{ fois}} \cdot t_{n+1}.$$

Une récurrence immédiate, à l'aide du [lemme 2.4.2 \(Non effacement\)](#), montre que $|w_k|_a \geq k$. \square

B. Les langages de claviers

B.1. Les langages sans flèches

B.1.1. RK

Preuve de la [proposition 3.1.11](#)

Proposition 3.1.11

Le langage $L = (a^2)^*(b + b^2)$ reconnu par $\{aa, b\blacksquare, bb\blacksquare\}$ n'est pas dans RK.

Preuve

Raisonnons par l'absurde et supposons qu'il existe un clavier K de RK qui reconnaît L . Le [corollaire 2.2.4 \(Forme normale de REK\)](#) nous permet de supposer qu'il est sous forme normale.

Puisque $b^2 \in L$, alors il existe une suite de touches τ telle que $\varepsilon \cdot \tau = b^2$. La suite τ est alors, par mise sous forme normale, équivalent à $\leftarrow^k b^2$ avec $k \in \mathbb{N}$. Nous raisonnons alors suivant la valeur de k .

- Si $k = 0$, alors $\tau \sim b^2$; on a alors $\varepsilon \cdot \tau \cdot \tau = b^2 \cdot \tau = b^4 \in L$.
- Si $k = 1$, alors $\tau \sim \leftarrow b^2$; on a alors $\varepsilon \cdot \tau \cdot \tau = b^2 \cdot \tau = b^3 \in L$.
- Si $k > 1$ et k pair; on a $a^{2k}b \in L$, et donc $a^{2k}b \cdot \tau = a^{k+1}b^2 \in L$.
- Si $k > 1$ et k impair; on a $a^{2k}b^2 \in L$, et donc $a^{2k}b^2 \cdot \tau = a^{k+2}b^2 \in L$.

Dans tous les cas, on aboutit à une contradiction, ce qui permet de conclure la preuve. \square

B.1.2. REK

Preuve de la [proposition 3.1.19](#)

Proposition 3.1.19

$L = a^*c^* \notin \text{REK}$.

Preuve

Raisonnons par l'absurde et supposons qu'il existe un clavier K de REK reconnaissant L . Nous posons $M = \|K\|_\infty$.

Puisque $a \in L$, il existe une suite de touches transientes $t_1 \dots t_i$ et une touche finale t_f telles que $\varepsilon \cdot t_1 \dots t_i t_f = a$. La suite $t_1 \dots t_i t_f$ est alors, par mise sous forme normale, équivalente à $\tau = \leftarrow^k a$ ■.

De même, puisque $c^{k+M+1} \in L$, il existe une suite de touches transientes $t'_1 \dots t'_i$ et une touche finale t'_f telles que $\varepsilon \cdot t'_1 \dots t'_i t'_f = c^{k+M+1}$. Puisque $|t_f| \leq M$, le [lemme 2.1.1 \(Localité\)](#) nous permet de dire que $u = \varepsilon \cdot \varepsilon \cdot t'_1 \dots t'_i$ admet c^{k+1} comme préfixe.

Mais on a alors $w \triangleq \varepsilon \cdot (t'_1 \dots t'_j) \cdot \tau \in L$, c'est-à-dire que $u[1, |u| - k]a \in L$.

Puisque u admet c^{k+1} comme préfixe, $u[1, |u| - k]$ admet k comme préfixe et donc w est de la forme cva (avec $v \in A^*$) ce qui est impossible. \square

Preuve du [théorème 3.1.15 \(REK est inclus dans Alg\)](#)

Théorème 3.1.15 (REK est inclus dans Alg)

Soit $K \in \text{REK}$. Alors $\mathcal{L}(K)$ est algébrique et on peut construire en temps polynomial un automate à pile non déterministe $\mathcal{A}(K)$ reconnaissant $\mathcal{L}(K)$.

Preuve

Soit $K \triangleq (T, F)$ un clavier de REK, soit $\mathcal{A}(K)$ l'automate à pile non-déterministe tel que :

- Son ensemble d'états est $\text{Pref}(T \cup F) \cup \{\text{Fin}\}$.
- Son alphabet d'entrée est A , son alphabet de pile est $A \cup \{\perp\}$, \perp étant le symbole de fond de pile.
- ε est le seul état initial
- Fin est le seul état final. L'automate accepte uniquement dans l'état Fin avec une pile vide.
- L'ensemble des transitions est $\Delta = \Delta_A \cup \Delta_{\leftarrow} \cup \Delta_{\text{boucle}} \cup \Delta_{\text{Fin}}$ avec

$$\begin{aligned} \Delta_A &= \left\{ t \xrightarrow[-, \uparrow a]{\varepsilon} ta \mid ta \in \text{Pref}(T \cup F), a \in A \right\} \\ \Delta_{\leftarrow} &= \left\{ t \xrightarrow[\downarrow a, -]{\varepsilon} t\leftarrow \mid t\leftarrow \in \text{Pref}(T \cup F), a \in A \right\} \cup \\ &\quad \left\{ t \xrightarrow[\downarrow \perp, \uparrow \perp]{\varepsilon} t\leftarrow \mid t\leftarrow \in \text{Pref}(T \cup F) \right\} \\ \Delta_{\text{boucle}} &= \left\{ t \xrightarrow[-, -]{\varepsilon} \varepsilon \mid t \in T \right\} \\ \Delta_{\text{Fin}} &= \left\{ t \xrightarrow[-, -]{\varepsilon} \text{Fin} \mid t \in F \right\} \cup \\ &\quad \left\{ \text{Fin} \xrightarrow[\downarrow a, -]{a} \text{Fin} \mid a \in A \right\} \cup \\ &\quad \left\{ \text{Fin} \xrightarrow[\downarrow \perp, -]{\varepsilon} \text{Fin} \right\} \end{aligned}$$

On observe qu'un mot w est accepté par $\mathcal{A}(K)$ si et seulement si il existe une exécution de l'automate menant à l'état Fin avec $w\perp$ comme contenu de pile, si et seulement si il existe $t \in F$ et $w' \in A^*$ tels qu'il existe une exécution de $\mathcal{A}(K)$ menant à ε avec $w'\perp$ comme contenu de pile, et $w' \cdot t = w$.

De plus une induction sur n nous montre que pour toute exécution de $\mathcal{A}(K)$ de longueur n menant à un état $p \in \text{Pref}(T)$, il existe un mot w et une suite de touches $t_1 \cdots t_k \in T^*$ tels que $\varepsilon \cdot t_1 \cdots t_k p = \tilde{w}$ et le contenu de pile soit $w\perp$ à la fin de l'exécution.

Une autre induction sur n nous montre que pour toute suite de $\sigma_1 \cdots \sigma_n$ d'opérations formant un élément de T^*p avec $p \in \text{Pref}(T)$, il existe une exécution de l'automate menant à l'état p avec comme contenu de pile $\varepsilon \cdot \widetilde{\sigma_1 \cdots \sigma_n} \perp$.

Ces deux inductions nous montrent en particulier que les mots w tels qu'il existe une exécution de $\mathcal{A}(K)$ menant à ε avec $w\perp$ dans la pile sont exactement les miroirs des mots w obtenus en appliquant une suite de touches de T à ε .

En conclusion, $\mathcal{A}(K)$ reconnaît exactement le miroir de $\mathcal{L}(K)$. Pour tout automate à pile on peut construire en temps polynomial un automate à pile reconnaissant le langage miroir du sien, on obtient donc le résultat. \square

Preuve du **théorème 3.1.16 (REK est inclus dans Rat)**

Théorème 3.1.16 (REK est inclus dans Rat)

Soit $K \in \text{REK}$. Alors $\mathcal{L}(K)$ est rationnel et on peut construire en temps polynomial un automate $\mathcal{A}(K)$ non déterministe reconnaissant $\mathcal{L}(K)$.

On va commencer par montrer ce résultat pour les langages de RK. Pour construire un automate fini non-déterministe qui reconnaît un langage de RK, on décompose une exécution sur un clavier en blocs qui permettent de lire les lettres de manière monotone. Le **lemme B.1.7 (Décomposition monotone d'une exécution de RK)** établit cette décomposition. Les lemmes techniques suivants apportent des conditions nécessaires et suffisantes pour construire les transitions de l'automate en temps polynomial, défini en définition **B.1.15**. Les définitions qui suivent nous seront utiles.

Définition B.1.1 (Profondeur)

Soient K un clavier de RK et $\tau = t_1 \dots t_n \in K^*$. Pour tout $i \in \llbracket 1, n \rrbracket$, il existe un unique couple (k_i, w_i) tel que $t_i \sim \leftarrow^{k_i} w_i$. On définit la *profondeur* de τ par

$$\text{Prof}(\tau) = \max \left\{ k_i + \sum_{j=1}^{i-1} (k_j - |w_j|) \mid i \in \llbracket 1, n \rrbracket \right\}.$$

si $n \geq 1$, et $\text{Prof}(\tau) = 0$ si $\tau = \varepsilon$.

$\text{Prof}(\tau)$ correspond à la taille maximale d'un suffixe du mot initial effacé lors de l'exécution de la séquence de touches τ , comme le montre le lemme suivant.

Lemme B.1.2

Soit K un clavier de RK, $\tau = t_1 \dots t_n \in K^*$ et $\# \notin A$. On pose $u = \#^{n\|K\|_\infty}$. Alors

$$\text{Prof}(\tau) = |u| - |u \cdot \tau|_{\#}.$$

Preuve

On pose $m = |u|$,

$$S_i = \begin{cases} 0 & \text{si } i = 0 \\ k_i + \sum_{j=1}^{i-1} (k_j - |w_j|) & \text{si } i \in \llbracket 1; n \rrbracket \end{cases}$$

$$M_i = \begin{cases} 0 & \text{si } i = 0 \\ \max_{j \in \llbracket 1; n \rrbracket} S_j & \text{sinon} \end{cases}$$

Montrons par récurrence sur n que $u \cdot (t_1 \dots t_n) = \#^{m-M_n} v$ avec $v \in A^*$ tel que $|v| = M_n - S_n + |w_n|$.

Si $n = 0$, $u \cdot \varepsilon = u = \#^{m-0} \varepsilon$ convient car $S_0 = M_0 = 0$.

Si $n > 0$, on pose $\tau' = t_1 \dots t_{n-1}$. Par hypothèse de récurrence, on a $u \cdot \tau' = \#^{m-M_{n-1}} v'$ où $v' \in A^*$ tel que $|v'| = M_{n-1} - S_{n-1} + |w_{n-1}|$.

— Cas 1 : $k_n \leq M_{n-1} - S_{n-1} + |w_{n-1}|$, alors

$$\begin{aligned} u \cdot \tau &= \#^{m-M_{n-1}} v' [1, |v'| - k_n] w_n \\ &= \#^{m-M_{n-1}} v \end{aligned}$$

où $v = v' [1, |v'| - k_n] w_n \in A^*$ tel que

$$\begin{aligned} |v| &= |v'| - k_n + |w_n| \\ &= M_{n-1} - S_{n-1} + |w_{n-1}| - k_n + |w_n| \\ &= M_{n-1} - S_n + |w_n| \end{aligned}$$

De plus, par hypothèse, $M_{n-1} \geq S_{n-1} + k_n - |w_{n-1}| = S_n$, donc $M_n = M_{n-1}$. D'où la décomposition souhaitée.

— Cas 2 : $k_n > M_{n-1} - S_{n-1} + |w_{n-1}|$, alors,

$$u \cdot \tau = \#^{m-M_{n-1}-(k_n-|v|)} w_n$$

De plus, comme $M_{n-1} < S_{n-1} + k_n - |w_{n-1}| = S_n$, on a

$$\begin{aligned} M_n &= S_n \\ &= S_{n-1} + k_n - |w_{n-1}| \\ &= -|v'| + M_{n-1} + k_n \end{aligned}$$

D'où la décomposition souhaitée $u \cdot \tau = \#^{m-M_n} w_n$ avec $|w_n| = M_n - S_n + |w_n|$. Cela termine la récurrence. On a donc $|u| - |u \cdot \tau|_{\#} = m - (m - M_n) = M_n = \text{Prof}(\tau)$. \square

Définition B.1.3 (Action sur les entiers)

Soient K un clavier de RK et $t = \leftarrow^k w \in K$. On étend l'opération \cdot aux entiers naturels en posant pour tout $n \in \mathbb{N}$,

$$n \cdot t = \max(0, n - k) + |w|.$$

On étend cette définition à K^* en posant $n \cdot \varepsilon = n$ et $n \cdot (\tau t) = (n \cdot \tau) \cdot t$.

Remarque B.1.4

L'intérêt de cette définition réside dans le fait que pour tous $u \in A^*$ et $\tau \in K^*$, on a $|u \cdot \tau| = |u| \cdot \tau$.

Remarque B.1.5

Les notations $\xrightarrow{\tau}$, $\xrightarrow{\tau}_e$ et \odot sont introduites de manière analogue sur les entiers naturels.

À la suite de ce résultat, dans tous les lemmes suivants on supposera que les claviers considérés sont inclus dans $\leftarrow^* A^*$.

Lemme B.1.6

Soit $K \subseteq \leftarrow^* A^*$ un clavier. Soient $w_0, v \in A^*$ avec $|w_0| \leq \|K\|_\infty$. Soit $0 \leq s \leq \|K\|_\infty$. Il existe $u \in A^s$ et $\tau \in K^*$ tel que $w_0 \xrightarrow{\tau}_e vu$ si et seulement s'il existe $k \in \mathbb{N}$ et $v_1, \dots, v_k \in A^+$, $w_1, \dots, w_k \in A^*$, $0 \leq s_1, \dots, s_k \leq \|K\|_\infty$ et $\tau_0, \tau_1, \dots, \tau_k \in K^*$ tels que :

1. $v = v_1 \cdots v_k$
2. Pour tout $1 \leq i \leq k$, $\leftarrow^{s_i} v_i w_i \in K$
3. En posant $s_{k+1} = s$, pour tout $0 \leq i \leq k$, $|w_i| \xrightarrow{\tau_i}_e s_{i+1}$.

Preuve

\Leftarrow : on pose $\tau = \tau_0(\leftarrow^{s_1} v_1 w_1) \tau_1(\leftarrow^{s_2} v_2 w_2) \cdots \tau_k$. On peut facilement vérifier que $w_0 \xrightarrow{\tau}_e vu$ avec $|u| = s_{k+1} = s$.

\Rightarrow : nous allons procéder par induction sur v .

Si $v = \varepsilon$, alors les deux premières conditions sont trivialement validées. Pour la troisième, on pose $s_1 = |u|$, et on obtient immédiatement que, comme $w_0 \xrightarrow{\tau}_e u$, $|w_0| \xrightarrow{\tau}_e s_1$.

Si $v \neq \varepsilon$, alors soient $t_1, \dots, t_n \in K$ tels que $t_1 \cdots t_n = \tau$, posons

$j = \min\{i \mid \text{Prof}(t_{i+1} \cdots t_n) \leq |w_0 \cdot t_1 \cdots t_i| - |v|\}$, c'est-à-dire que j est l'indice de la dernière touche qui affecte une des $|v|$ premières lettres du mot w_0 . Notamment, on a $w_0 \cdot t_1 \cdots t_j = \alpha z$ avec $|\alpha| = |v|$ et $|z| \geq \text{Prof}(t_{j+1} \cdots t_n)$. Par le lemme B.1.2, on a que la réécriture $\alpha z \xrightarrow{t_{j+1} \cdots t_n} vu$ n'affecte pas α , donc $\alpha = v$.

De plus, par minimalité de j , on a aussi $w_0 \cdot t_1 \cdots t_{j-1} = v' u'$ avec $v = v' y$ (par le même argument de conservation du préfixe) pour un $y \in A^+$ tel que $u' \xrightarrow{t_j}_e yz$. En posant $\leftarrow^m x = t_j$, nécessairement $|u'| = m$ et $yz = x$.

Par hypothèse d'induction sur $w_0 \xrightarrow{t_1 \cdots t_{j-1}}_e v' u'$, on a qu'il existe $k' \in \mathbb{N}$ et $v_1, \dots, v_{k'} \in A^+$, $w_1, \dots, w_{k'} \in A^*$, $0 \leq s_1, \dots, s_{k'} \leq \|K\|_\infty$ et $\tau_0, \tau_1, \dots, \tau_{k'} \in K^*$ tels que :

1. $v' = v_1 \cdots v_{k'}$
2. Pour tout $1 \leq i \leq k'$, $\leftarrow^{s_i} v_i w_i \in K$
3. En posant $s_{k'+1} = |u'| = m$, pour tout $0 \leq i \leq k$, $|w_i| \xrightarrow{\tau_i}_e s_{i+1}$, et $\text{Prof}(\tau_i) = |w_i|$.

On pose $\tau_k = t_{j+1} \cdots t_n$, $k = k' + 1$, $v_k = y$, $s_k = |u'| = m$ et $w_k = z$. Les trois conditions du lemme se vérifient facilement. \square

Lemme B.1.7 (Décomposition monotone d'une exécution de RK)

Soit $K \subseteq \leftarrow^* A^*$ un clavier et soit $v \in A^*$. Il existe $\tau \in K^*$ tel que $\varepsilon \xrightarrow{\tau} v$ si et seulement s'il existe $k \in \mathbb{N}$ et $v_1, \dots, v_k \in A^+$, $w_1, \dots, w_k \in A^*$, $0 \leq s_1, \dots, s_k \leq \|K\|_\infty$ et $\tau_0, \tau_1, \dots, \tau_k \in K^*$ tels que :

1. $v = v_1 \cdots v_k$
2. Pour tout $1 \leq i \leq k$, $\leftarrow^{s_i} v_i w_i \in K$
3. En posant $s_{k+1} = 0$, pour tout $1 \leq i \leq k$, $|w_i| \xrightarrow{\tau_i}_e s_{i+1}$.
4. $0 \xrightarrow{\tau_0} s'_1$ avec $s'_1 \leq s_1$

Preuve

\Leftarrow : est immédiate en posant $\tau = \tau_0 \leftarrow^{s_1} v_1 w_1 \tau_1 \cdots \leftarrow^{s_k} v_k w_k \tau_k$.

\Rightarrow : posons $\tau = t_1 \cdots t_n$. Soit i l'indice minimal tel que $\text{Prof}(t_{i+1} \cdots t_n) < |\varepsilon \cdot t_1 \cdots t_i|$. On pose $pr = \text{Prof}(t_{i+1} \cdots t_n)$, et v_1, v' tels que $v = v_1 v'$ et $|v_1| = |v| - pr$.

On obtient que $\varepsilon \xrightarrow{t_1 \cdots t_i} v_1 w$ avec $w \xrightarrow{t_{i+1} \cdots t_n}_e v'$. Par minimalité de i , on a que t_i est de la forme $\leftarrow^{s_1} v_1 w$ avec $s_1 \geq |\varepsilon \cdot t_1 \cdots t_{i-1}|$. On pose $s'_1 = |\varepsilon \cdot t_1 \cdots t_{i-1}|$ et $\tau_0 = t_1 \cdots t_{i-1}$, ce qui nous permet de vérifier la quatrième condition, et on applique le Lemme B.1.6 à $w \xrightarrow{t_{i+1} \cdots t_n}_e v'$ pour obtenir les autres conditions. \square

Lemme B.1.8

Soit $K \subseteq \leftarrow^* A^*$ un clavier, soit $s_v \leq \|K\|_\infty$. Il existe $\tau \in K^*$ tel que $0 \xrightarrow{\tau} s_v$ si et seulement si il existe $t'_1, \dots, t'_m \in K$ avec $m \leq \|K\|_\infty$, $s_1, \dots, s_m, s'_1, \dots, s'_m \leq \|K\|_\infty$ et $\tau_0, \dots, \tau_m \in K^*$ tels que :

- Pour tout $1 \leq i \leq m$, $s_i \cdot t'_i = s'_i$
- En posant $s'_0 = 0$ et $s_{m+1} = s_v$, pour tout $0 \leq i \leq m$, $s'_i \xrightarrow{\tau_i}_e s_{i+1}$.

Preuve

\Leftarrow : il suffit de prendre $\tau = \tau_0 t'_1 \tau_1 \cdots t'_m \tau_m$.

\Rightarrow : posons $\tau = t_1 \cdots t_n$ de taille minimale tel que $0 \xrightarrow{\tau} s_v$. Par l'absurde, supposons qu'il existe M indices $i_1 < \dots < i_M$ avec $M > \|K\|_\infty$ tels que pour tout j , t_{i_j} soit de la forme $\leftarrow^{k_j} x_j$ avec $k_j > 0 \cdot t_1 \cdots t_{i_j-1}$. Alors, comme tous les x_j sont de taille inférieure à $\|K\|_\infty$, il existe $a < b$ tels que $|x_a| = |x_b|$. On en conclut qu'en posant $\tau' = t_1 \cdots t_{i_a} t_{i_b+1} \cdots t_n$, on obtient $0 \xrightarrow{\tau'} s_v$, contredisant la minimalité de τ .

Donc, en posant $t_i = \leftarrow^{k_i} x_i$ pour tout i , il existe $i_1 < \dots < i_m$ tels que $m \leq \|K\|_\infty$, pour tout j , $k_{i_j} > 0 \cdot t_1 \cdots t_{i_j-1}$, et pour tout $i \in \{1, \dots, n\} \setminus \{i_1, \dots, i_m\}$, $k_i \leq 0 \cdot t_1 \cdots t_{i-1}$.

On obtient donc bien les conditions du lemme en posant $\tau_j = t_{i_j+1} \cdots t_{i_{j+1}-1}$, $s_j = 0 \cdot (t_1 \cdots t_{i_j-1})$, $s'_j = 0 \cdot (t_1 \cdots t_{i_j})$ et $t'_j = t_{i_j}$, pour tout j .

Lemme B.1.9

Soit $K \subseteq \leftarrow^* A^*$ un clavier, soit p le PGCD de $N = \{|x| - k \mid \leftarrow^k x \in K\}$. Soient $0 \leq s_u, s_v \leq \|K\|_\infty$, supposons qu'il existe $\leftarrow^{k_1} x_1, \leftarrow^{k_2} x_2 \in K$ tels que $|x_1| - k_1 < 0 < |x_2| - k_2$, $|x_1| \leq s_v$ et $k_2 \leq s_u$, et supposons que p divise $s_v - s_u$.

Alors il existe une suite d'opérations $\tau = t_1 \cdots t_n \in K^*$ telle que $s_u \xrightarrow{\tau}_e s_v$.

Preuve

L'idée de la preuve est de construire un mot (grand, mais de taille bornée par une puissance de $\|K\|_\infty$) accepté par K et de longueur s , tel qu'on a $s_u \rightarrow_e s \rightarrow_e s_v$. On utilise notamment des lemmes d'existence de décomposition en combinaison linéaire entière basés sur le PGCD pour obtenir s .

On pose $M = \|K\|_\infty$. En particulier, on a $M \geq \max\{|x| - k \mid \leftarrow^k x \in K\}$.

Considérons d'abord $s'_u = s_u \cdot (\leftarrow^{k_2} x_2)^M$. On a $s_u \xrightarrow{(\leftarrow^{k_2} x_2)^M}_e s'_u$ car $k_2 \leq s_u$ et $0 < |x_2| - k_2$. De plus, $s'_u = s_u + M(|x_2| - k_2) \geq M$, et $s_v - s'_u = s_v - s_u - M(|x_2| - k_2)$. Comme p divise $s_v - s_u$ et $|x_2| - k_2$, p divise $s_v - s'_u$.

On considère également la constante $s'_v = s_v + M(k_1 - |x_1|)$. $s_u - s_v$ et $M(k_1 - |x_1|)$ étant multiples de p , $s_u - s'_v$ l'est aussi.

Maintenant essayons de construire $s'_u \rightarrow_e s \rightarrow_e s'_v$.

Soient $N_+ = N \cap (\mathbb{N} \setminus \{0\})$ et $N_- = N \setminus \mathbb{N}$, on a $|x_1| - k_1 \in N_-$ et $|x_2| - k_2 \in N_+$, donc ces ensembles sont non vides.

Soient p_+ le PGCD de N_+ , p_- celui de N_- , leur PGCD est p . De plus p_+ et p_- sont positifs et donc il existe $0 \leq a_+, a_- \leq \max\{p_+, p_-\}$ tels que $a_+ p_+ - a_- p_- = p$ par le théorème de Bézout. En particulier on a $a_+, a_- \leq M$.

Les ensembles $\{\frac{n}{p_+} \mid n \in N_+\}$ et $\{\frac{n}{p_-} \mid n \in N_-\}$ sont tous deux des ensembles d'entiers naturels de PGCD 1. Par le théorème de Schur, il existe $B_+ \in \mathbb{N}$ (resp. B_-) tel que pour tout $m \geq B_+$ (resp. B_-), il existe $(i_n)_{n \in N_+}$ (resp. $(i_n)_{n \in N_-}$) une famille d'entiers naturels telle que $\sum_{n \in N_+} \frac{n}{p_+} i_n = m$ (resp. $\sum_{n \in N_-} \frac{n}{p_-} i_n = m$).

Nous allons à présent utiliser un résultat d'Erdős and Graham selon lequel, pour tous entiers naturels $a_1 < \cdots < a_k$ avec $k > 1$ tel que $\text{PGCD}(\{a_1, \dots, a_k\}) = 1$, pour tout entier $n \geq 2a_{k-1} \lfloor \frac{a_k}{k} \rfloor - a_k$, il existe $i_1, \dots, i_k \in \mathbb{N}$ tels que $\sum_{j=1}^k a_j i_j = n$ [1]. On surapproxime la borne $2a_{k-1} \lfloor \frac{a_k}{k} \rfloor - a_k$ par a_k^2 . La borne d'Erdős et Graham est l'une de plusieurs bornes nous permettant d'obtenir la majoration par a_k^2 . Nous renvoyons le lecteur vers un document de Gallier contenant une liste de ces résultats [2]. On peut donc supposer que $B_+, B_- \leq M^2$.

Soit $B \geq \max\{p_+B_+, p_-B_-\} + M(k_1 - |x_1|) \leq 2M^2$ tel que p_+ divise B , p_+ étant divisible par p , B l'est aussi. Soit $r \in \{0, \dots, p_- - 1\}$ tel qu'il existe $q \in \mathbb{N}$ tel que $B + s'_u - s'_v = qp_- + r$ par division euclidienne. B et p_- étant divisible par p , r l'est aussi, ainsi que $p_- - r$. Il existe donc $r' \in \{0, \dots, p_- - 1\}$ tel que $p_- - r = r'p = r'a_+p_+ - r'a_-p_-$. On a donc que $B + s'_u - s'_v + r'a_+p_+ = B + s'_u - s'_v + p_- - r + r'a_-p_-$. B étant divisible par p_+ , $B + r'a_+p_+$ aussi, et $B + r'a_+p_+ \geq B \geq p_+B_+$, donc il existe $(i_n)_{n \in N_+}$ une famille d'entiers naturels telle que $\sum_{n \in N_+} ni_n = B + r'a_+p_+$. Pour tout $n \in N$, soit $t_n \in K$ de la forme $\leftarrow^k x$ avec $|x| - k = n$. Posons $\theta = \prod_{n \in N_+} (t_n)^{i_n}$ (les $(t_n)^{i_n}$ étant dans un ordre arbitraire), on obtient donc $s'_u \cdot \theta = s$ avec

$$s \triangleq s_u + B + r'a_+p_+ + M(|x_2| - k_2)$$

À tout pas du calcul le nombre obtenu est plus grand que $M(|x_2| - k_2) \geq M$, donc aucune touche ne peut appliquer de retour sur le mot vide. En conséquence, on a $s_u \xrightarrow{(\leftarrow^{k_2} x_2)^M \theta} s$.

On procède de manière similaire pour construire $s \rightarrow_e s'_v$.

$$\begin{aligned} s - s'_v &= B + r'a_+p_+ + M(|x_2| - k_2) + (s_u - s'_v) \\ &= B + (p_- - r + r'a_-p_-) + s'_u - s'_v \\ &= qp_- + r + (p_- - r + r'a_-p_-) \\ &= (q + 1 + r'a_-)p_- \end{aligned}$$

Comme $B \geq p_-B_- + M(k_1 - |x_1|)$ et $s_v \leq M \leq M(|x_2| - k_2)$, on a

$$\begin{aligned} s - s'_v &= B + r'a_+p_+ + M(|x_2| - k_2) + (s_u - s'_v) \\ &= B + r'a_+p_+ + M(|x_2| - k_2) + M(|x_1| - k_1) + (s_u - s_v) \\ &\geq p_-B_- + r'a_+p_+ + s_u \\ &\geq p_-B_- \end{aligned}$$

Il existe donc $(i_n)_{n \in N_-}$ une famille d'entiers naturels telle que $\sum_{n \in N_-} ni_n = s - s'_v$.

Pour tout $n \in N_-$, on pose t_n tel que $t_n = \leftarrow^k x$ avec $|x| - k = n \leq 0$. On pose $\nu = \prod_{n \in N_-} (t_n)^{i_n}$ (les $(t_n)^{i_n}$ étant dans un ordre arbitraire). On obtient que $s \cdot \nu = s'_v$. De plus à tout pas la taille du mot obtenu est supérieure à $s'_v \geq M(k_1 - |x_1|) \geq M$, donc aucune touche ne peut appliquer de retour arrière sur le mot vide. On obtient donc $s \xrightarrow{\theta} s'_v$.

Enfin, $s'_v \cdot (\leftarrow^{k_1} x_1)^M = s'_v - M(k_1 - |x_1|) = s_v$. De plus comme $k_1 - |x_1| \leq s_v$, on a $s'_v \xrightarrow{(\leftarrow^{k_1} x_1)^M} s_v$.

En conclusion, $u \xrightarrow{(\leftarrow^{k_2} x_2)^M \theta \nu (\leftarrow^{k_1} x_1)^M} s_v$. De plus, l'entier le plus grand obtenu lors de cette exécution est $s = s_u + B + r'a_+p_+ + M(|x_2| - k_2) \leq \|K\|_\infty + 2M^2 + M^3 + M^2 \leq 5\|K\|_\infty^3$. \square

Lemme B.1.10

Soit $K \subseteq \leftarrow^* A^*$ un clavier. Soient $0 \leq s_u, s_v \leq \|K\|_\infty$, supposons que pour tout $\leftarrow^k x \in K$ tel que $|x| - k < 0$, $|x| > s_v$.

Alors les propositions suivantes sont équivalentes :

- Il existe $\tau \in K^*$ tel que $s_u \xrightarrow{\tau}_e s_v$.
- Il existe $\tau = t_1 \dots t_n \in K^*$ tel que $s_u \xrightarrow{\tau}_e s_v$ et, pour tout préfixe $t_1 \dots t_i$ de τ , $s_u \leq s_u \cdot t_1 \dots t_i \leq s_v$ et $s_v - (s_u \cdot t_1 \dots t_i)$ est strictement décroissante en i .

Remarque B.1.11

La condition pour tout $\leftarrow^k x \in K$ tel que $|x| - k < 0$, $|x| > s_v$ signifie que toutes les touches “négatives” (c’est-à-dire appartenant à N_- avec les notations du lemme B.1.9) écrivent strictement plus que s_v . Ainsi, on est obligé de se restreindre à des touches “positives”, et donc $s_u \leq s_v$. C’est ce que montre ce lemme.

Preuve

Le fait que la deuxième proposition implique la première est immédiat.

Supposons qu’il existe $\tau \in K^*$ tel que $s_u \xrightarrow{\tau}_e s_v$.

Soit $\tau \in K^*$ tel que $s_u \xrightarrow{\tau}_e s_v$, avec τ de longueur minimale pour cette propriété.

Posons $\tau = t_1 \dots t_p$ avec $t_1, \dots, t_p \in K$, et pour tout $1 \leq i \leq p$ posons $t_i = \leftarrow^{k_i} x_i$. Supposons qu’il existe $1 \leq i \leq p$ tel que $|x_i| - k_i < 0$, alors par hypothèse on a $|x_i| > s_v$. Prenons i maximal pour cette propriété, alors on a $|x_j| - k_j \geq 0$ pour tout $j > i$, donc $|s_u \cdot t_1 \dots t_i| \leq |s_u \cdot t_1 \dots t_p| = s_v$. Cependant, l’exécution de τ sur s_u étant efficace, on a $|s_u \cdot t_1 \dots t_{i-1}| \geq k_i$, et donc $|s_u \cdot t_1 \dots t_i| \geq |x_i| > s_v$. On a donc $s_v = |s_u \cdot t_1 \dots t_p| \geq |s_u \cdot t_1 \dots t_i| > s_v$, d’où contradiction. En conséquence, on a $|x_i| - k_i \geq 0$ pour tout i .

De plus on a supposé τ minimal. Supposons qu’il existe i tel que t_i est de la forme $\leftarrow^k x$ avec $|x| = k$. Alors on peut poser $\theta = t_1 \dots t_{i-1} t_{i+1} \dots t_n$, et facilement vérifier que $s_u \xrightarrow{\theta}_e s_v$, contredisant la minimalité de τ . Donc à chaque étape $s_v - s_u \cdot t_1 \dots t_i$ décroît strictement, cette valeur étant bornée par $\|K\|_\infty$ au départ.

On a donc bien $p \leq \|K\|_\infty$.

Lemme B.1.12

Soit $K \subseteq \leftarrow^* A^*$ un clavier. Soient $0 \leq s_u, s_v \leq \|K\|_\infty$, supposons que pour tout $\leftarrow^k x \in K$ tel que $|x| - k > 0$, $k > s_u$.

Alors les propositions suivantes sont équivalentes :

- Il existe $\tau \in K^*$ tel que $s_u \xrightarrow{\tau}_e s_v$.
- Il existe $\tau = t_1 \dots t_p \in K^*$ tel que $s_u \xrightarrow{\tau}_e s_v$ et, pour tout préfixe $t_1 \dots t_i$ de τ , $s_v \leq s_u \cdot t_1 \dots t_i \leq s_u$ et $s_u \cdot t_1 \dots t_i - s_v$ est strictement décroissante en i .

Le deuxième point implique notamment $p \leq \|K\|_\infty$.

Preuve

Le fait que la deuxième proposition implique la première est immédiat.

Supposons qu'il existe $\tau \in K^*$ tel que $s_u \xrightarrow{\tau}_e s_v$.

Soit $\tau \in K^*$ tel que $s_u \xrightarrow{\tau}_e s_v$, avec τ de longueur minimale pour cette propriété.

Posons $\tau = t_1 \cdots t_p$ avec $t_1, \dots, t_p \in K$, et pour tout $1 \leq i \leq p$ posons $t_i = \leftarrow^{k_i} x_i$. Supposons qu'il existe $1 \leq i \leq p$ tel que $|x_i| - k_i > 0$, alors par hypothèse on a $k_i > s_u$. Prenons i minimal pour cette propriété, alors on a $|x_j| - k_j \leq 0$ pour tout $j < i$, donc $s_u \geq |s_u \cdot t_1 \cdots t_{i-1}|$. Cependant, l'exécution de τ sur s_u étant efficace, on a $|s_u \cdot t_1 \cdots t_{i-1}| \geq k_i$. On a donc $s_u \geq |s_u \cdot t_1 \cdots t_{i-1}| \geq k_i > s_u$, d'où contradiction. En conséquence, on a $|x_i| - k_i \leq 0$ pour tout i .

De plus on a supposé τ minimal. Supposons qu'il existe i tel que t_i est de la forme $\leftarrow^k x$ avec $|x| = k$. Alors on peut poser $\theta = t_1 \cdots t_{i-1} t_{i+1} \cdots t_n$, et facilement vérifier que $s_u \xrightarrow{\theta}_e s_v$, contredisant la minimalité de τ . Donc à chaque étape $s_u \cdot t_1 \cdots t_i - s_v$ décroît strictement, cette valeur étant toujours positive (car décroissante et égale à 0 pour $i = p$) et bornée par $\|K\|_\infty$ au départ.

On a donc bien $p \leq \|K\|_\infty$. □

Lemme B.1.13

Le problème suivant est décidable en temps polynomial :

Entrée : $K \subseteq \leftarrow^* A^*$ un clavier, $0 \leq s_u, s_v \leq \|K\|_\infty$.

Sortie : Existe-t-il $\tau = t_1 \cdots t_n \in K^*$ tel que $s_u \xrightarrow{\tau}_e s_v$?

Preuve

Soient $K \subseteq \leftarrow^* A^*$ un clavier, soit p le PGCD de $N = \{|x| - k| \mid \leftarrow^k x \in K\}$. Soient $0 \leq s_u, s_v \leq \|K\|_\infty$.

On procède par cas :

Cas 1 : Pour tout $\leftarrow^k x \in K$ tel que $|x| - k < 0$, $|x| > s_v$. Alors on peut définir un algorithme de programmation dynamique basé sur le Lemme B.1.10 utilisant un tableau associant à chaque $0 \leq s'_u \leq s_v$ un booléen, valant **vrai** si et seulement si s_v est accessible par une suite de touches depuis s'_u . On s'arrête quand on atteint s_u , qui est nécessairement plus petit que s_v d'après le lemme B.1.10.

Cas 2 : Pour tout $\leftarrow^k x \in K$ tel que $0 < |x| - k$, $k > s_u$. Similaire au Cas 1, cette fois on utilise le Lemme B.1.12 pour un algorithme sur $0 \leq s'_v \leq s_u$.

Cas 3 : Il existe $\leftarrow^{k_1} x_1, \leftarrow^{k_2} x_2 \in K$ tels que $|x_1| - k_1 < 0$, $0 < |x_2| - k_2$, $|x_1| \leq s_v$ et $k_2 \leq s_u$, et p divise $s_v - s_u$. Alors, par le Lemme B.1.9, il existe τ tel que $s_u \xrightarrow{\tau}_e s_v$.

Cas 4 : p ne divise pas $s_v - s_u$. Alors on peut montrer par une récurrence immédiate sur τ que pour tout $\tau \in K^*$, $s_u \cdot \tau \neq s_v$.

Lemme B.1.14

Le problème suivant est décidable en temps polynomial :

Entrée : $K \subseteq \leftarrow^* A^*$ un clavier, $0 \leq s_v \leq \|K\|_\infty$.

Sortie : Existe-t-il $\tau = t_1 \cdots t_n \in K^*$ tel que $\varepsilon \xrightarrow{\tau} s_v$?

Preuve

D'après le Lemme B.1.8, il nous suffit de trouver un chemin de 0 à s_v dans le graphe dont les sommets sont les éléments de $\{0, \dots, \|K\|_\infty\}$ et les arêtes sont les éléments de $\{(m, n) \mid \exists \leftarrow^k x \in K, k = m, |x| = n\} \cup \{(m, n) \mid \exists \tau \in K^*, m \xrightarrow{\tau_e} n\}$.

Ce graphe étant calculable en temps polynomial d'après le Lemme B.1.8, il suffit de le construire puis de faire un test d'accessibilité en temps polynomial. \square

Définition B.1.15

Soit $K \subseteq \leftarrow^* A^*$ un clavier, on définit $\mathcal{A}(K)$ l'automate fini non-déterministe dont les états sont les éléments de $\{\text{Init}\} \cup \{(m, v, w, v') \in \mathbb{N} \times (A^*)^3 \mid \leftarrow^m vw \in K, v' \in \text{Suff}(v)\}$.

L'unique état initial est Init, les états finaux sont les éléments de

$$\{(m, v, w, \varepsilon) \mid \leftarrow^m vw \in K, \exists \tau \in K^*, |w| \xrightarrow{\tau_e} 0\},$$

plus Init si et seulement si $\varepsilon \in \mathcal{L}(K)$.

L'ensemble des transitions est $\Delta = \Delta_{\text{Init}} \cup \Delta_{\text{lit}} \cup \Delta_{\text{touche}}$ avec

$$\begin{aligned} \Delta_{\text{Init}} &= \{\text{Init} \xrightarrow{\varepsilon} (m, v, w, v) \mid \exists \tau, \leftarrow^m vw \in K, 0 \xrightarrow{\tau} m\} \\ \Delta_{\text{lit}} &= \{(m, v, w, av') \xrightarrow{a} (m, v, w, v') \mid \leftarrow^m vw \in K, a \in A, av' \in \text{Suff}(v)\} \end{aligned}$$

Pour tous états $s_1 = (m_1, v_1, w_1, \varepsilon)$ et $s_2 = (m_2, v_2, w_2, v_2)$ tel qu'il existe une suite de touches τ tel que $|w_1| \xrightarrow{\tau_e} m_2$ on a une transition $s_1 \xrightarrow{\varepsilon} s_2$. Δ_{touche} est l'union de ces transitions.

Lemme B.1.16

Soit $K \subseteq \leftarrow^* A^*$ un clavier, $\mathcal{A}(K)$ et K reconnaissent le même langage.

Preuve

Par définition de $\mathcal{A}(K)$, K reconnaît le mot vide si et seulement si $\mathcal{A}(K)$ le reconnaît.

Soit $v \in A^+$. Une induction sur v montre que v est accepté par $\mathcal{A}(K)$ si et seulement si il existe $v_1, \dots, v_k \in A^*$, $s_1, \dots, s_k \leq \|K\|_\infty$, $w_1, \dots, w_k \in A^*$ et $\tau_0, \dots, \tau_k \in K^*$ tels que :

- $v = v_1 \cdots v_k$
- Pour tout i , $\leftarrow^{s_i} v_i w_i \in K$
- En posant $s_{k+1} = 0$, pour tout i , $|w_i| \xrightarrow{\tau_i_e} s_{i+1}$
- $0 \xrightarrow{\tau_0} s_1$

Donc d'après le Lemme B.1.7, $v \in \mathcal{L}(K)$ si et seulement si $v \in \mathcal{L}(\mathcal{A}(K))$. \square

Théorème B.1.17

Tout langage reconnu par un clavier de RK est rationnel.

Preuve

Soit $K \subseteq \leftarrow^* A^*$ un clavier, par le Lemme B.1.16, son langage est exactement celui de $\mathcal{A}(K)$, $\mathcal{L}(K)$ est donc rationnel. \square

Lemme B.1.18

Soit $K \subseteq \leftarrow^* A^*$ un clavier, on peut construire $\mathcal{A}(K)$ en temps polynomial en la taille de K .

Preuve

Le calcul de l'ensemble des transitions de $\mathcal{A}(K)$ revient à déterminer, d'une part pour chaque $\leftarrow^m u \in K$ s'il existe $\tau \in K^*$ tel que $0 \xrightarrow{\tau} m$, et d'autre part pour chaque $\leftarrow^k x, \leftarrow^\ell y \in K$, pour chaque suffixe w de x , si il existe $\tau \in K^*$ tel que $|w| \xrightarrow{\tau}_e \ell$. Ces opérations peuvent être faites en temps polynomial d'après les Lemmes B.1.14 et B.1.13, donc l'automate peut être construit en temps polynomial. \square

Nous allons à présent étendre les résultats de la partie précédente à REK.

Lemme B.1.19

Soit $K = (T, F)$ un clavier de REK. On peut construire en temps polynomial un automate $\mathcal{A}(K)$ reconnaissant $\mathcal{L}(K)$.

Preuve

Soit $K \triangleq (T, F)$ un clavier de REK. On pose $L \triangleq \mathcal{L}(K)$ et $L_T = \mathcal{L}(T)$ où T est vu comme un clavier de RK.

Par les théorèmes B.1.16 et B.1.18, L_T est rationnel et on peut construire en temps polynomial un automate le reconnaissant.

On remarque alors que

$$L = \cup_{f \in F} L_f$$

où $L_f \triangleq \{w \cdot f \mid w \in L_T\}$.

Soit f une touche finale de la forme $\leftarrow^k u \blacksquare$. Alors $L_f = (L_T/A^k)u + \delta_u$ où

$$\begin{aligned} L_T/A^k &= \{w \mid \exists v \in A^k, wv \in L_T\} \\ \delta_u &\triangleq u \text{ si } L_T \cap A^{\leq k} \neq \emptyset \text{ et } \delta_u = \emptyset \text{ sinon.} \end{aligned}$$

Comme la vacuité de $L_T \cap A^{\leq k}$ est décidable en temps polynomial, que l'union, la concaténation et le quotient par A^k d'automates sont calculables en temps polynomial, on peut construire un automate pour L en temps polynomial. \square

Nous présentons ci-dessous une preuve alternative, sous la forme d'une construction explicite d'un automate reconnaissant $\mathcal{L}(K)$.

Preuve (Plus constructiviste)

Nous avons

$$\mathcal{L}(K) = \cup_{f \in F} \{w \cdot f \mid w \in \mathcal{L}(T)\},$$

avec T vu comme un clavier de RK. On a donc qu'un mot u appartient à $\mathcal{L}(K)$ si et seulement si l'une des deux conditions suivantes est vraie.

1. Il existe $k \in \mathbb{N}$ tel que $\leftarrow^k u \in F$ et il existe $x \in \mathcal{L}(T)$ tel que $|x| \leq k$.
2. il existe $k \in \mathbb{N}$, $v, w, x \in A^*$ tels que $u = vw$, $\leftarrow^k w \in F$, $vx \in \mathcal{L}(K)$ et $|x| = k$.

Par les théorèmes B.1.16 et B.1.18, on peut construire en temps polynomial un automate $\mathcal{A}(T) = (Q_T, \Delta_T, \text{Init}_T, \text{Fin}_T)$ reconnaissant $\mathcal{L}(T)$. Pour tout $k \leq \|K\|_\infty$, soit

$$Q_{-k} = \{q \in Q_T \mid \exists y \in A^k, \text{Fin}_T \cap \Delta_T(q, y) \neq \emptyset\}$$

l'ensemble des états pouvant atteindre un état final en lisant k lettres. Tous les Q_{-k} peuvent être calculés en temps polynomial. On pose également $\ell = \min\{|w| \mid w \in \mathcal{L}(T)\}$, qui peut être calculé en temps polynomial.

Considérons à présent l'automate $\mathcal{A}(K) = (Q_T \cup Q_F, \Delta_T \cup \Delta_F, \text{Init}_F, \{\varepsilon\})$, avec

- $Q_F = A^* \cap \text{Suff}(F)$ est l'ensemble des suffixes de mots écrits par des touches de F .
- $\Delta_F = \{q \xrightarrow{\varepsilon} w \mid \exists k \in \mathbb{N}, \leftarrow^k w \in F \wedge q \in Q_{-k}\} \cup \{av \xrightarrow{a} v \mid av \in Q_F\}$.
- $\text{Init}_F = \text{Init}_T \cup \{w \in A^* \mid \exists k \geq \ell, \leftarrow^k w \in F\}$.

Remarquons d'abord que pour tout état $w \in Q_F$, le seul mot permettant d'atteindre un état acceptant depuis w est w . Remarquons également que toute exécution acceptante commençant dans Init_T peut se décomposer en une exécution dans Q_T utilisant des transitions de Δ_T jusqu'à un état de Q_{-k} , une ε -transition jusqu'à un état $w \in Q_F$, puis un chemin lisant w .

Un mot u a une exécution acceptante dans cet automate si et seulement s'il satisfait une des deux conditions suivantes :

1. Il existe $w \in A^*$ et $k \geq \ell$ tels que u est accepté depuis w , c'est-à-dire que $u = w$. Autrement dit, par définition de ℓ , il existe $k \in \mathbb{N}$ tel que $\leftarrow^k u \in F$ et il existe $x \in \mathcal{L}(T)$ tel que $|x| \leq k$.
2. Il existe $k \in \mathbb{N}$, $w \in A^*$, $v \in A^k$ et $s \in Q_T$ tels que $u = vw$, $\leftarrow^k w \in F$ et $s \in \Delta_T(\text{Init}_T, v) \cap Q_{-k}$, i.e., il existe $x \in A^k$ tel que $vx \in \mathcal{L}(T)$.

On peut voir à présent que la condition 1 (resp. 2) ci-dessus est équivalente à la condition 1 (resp. 2) du début de la preuve. En conclusion, un mot est accepté par $\mathcal{A}(K)$ si et seulement s'il est dans $\mathcal{L}(K)$.

Comme cet automate peut être construit en temps polynomial, le théorème est prouvé. \square

B.2. Les langages sans flèche droite

B.2.1. GK

Preuve de l'exemple 3.2.3 (Mot de Dyck)

Exemple 3.2.3 (Mot de Dyck)

Le clavier $K = \{() \blacktriangleleft, \blacktriangleleft\}$ reconnaît le langage L des mots bien parenthésés qui est donc dans GK.

Preuve

Les mots engendrés sont clairement dans L , donc $\mathcal{L}(K) \subset L$.

On montre la réciproque par récurrence sur la longueur de $u \in L$. Montrons ainsi qu'il existe une suite de touche τ telle que $\langle \varepsilon | \varepsilon \rangle \cdot \tau = \langle \varepsilon | u \rangle$.

- Pour $u = \varepsilon$, on prend $\tau = \blacktriangleleft$.
- Soit u bien parenthésé de taille $2(n+1)$. Il existe v et w de tailles respectives $2k$ et $2(n-k)$ tel que $u = (v)w$. Par hypothèse de récurrence, il existe τ_v et τ_w tel que $\langle \varepsilon | \varepsilon \rangle \cdot \tau_v = \langle \varepsilon | v \rangle$ et $\langle \varepsilon | \varepsilon \rangle \cdot \tau_w = \langle \varepsilon | w \rangle$. En posant $\tau = \tau_w() \blacktriangleleft \tau_v \blacktriangleleft$, on obtient bien $\langle \varepsilon | \varepsilon \rangle \cdot \tau = \langle \varepsilon | u \rangle$. \square

B.2.2. GEK

Preuve du théorème 3.2.6

On considère le langage $L_{\text{pal}@} = \{w@w \mid w \in A^*\}$ où $@$ est une lettre fraîche. Le problème suivant est indécidable.

$$\text{Palindrome : } \begin{cases} \text{DONNÉE :} & \text{Un langage } L \text{ de GEK} \\ \text{QUESTION :} & L \cap L_{\text{pal}@} = \emptyset ? \end{cases}$$

Preuve

On réduit le problème de correspondance de Post. Soit $(u_i, v_i)_{i \in \llbracket 1, n \rrbracket}$ une instance de Post. On considère le clavier

$$K = \{u_i \widetilde{v_i} \blacktriangleleft^{|v_i|} \mid i \in \llbracket 1, n \rrbracket\} \cup \{ @ \blacksquare \}.$$

Montrons que $\mathcal{L}(K) \cap L_{\text{pal}@} \neq \emptyset$ si et seulement si $(u_i, v_i)_{i \in \llbracket 1, n \rrbracket} \in \text{PCP}$. En appelant t_j la touche $u_j \widetilde{v_j} \blacktriangleleft^{|v_j|}$, on montre aisément par récurrence sur $k \in \mathbb{N}$ que pour toute suite d'indices $i_1, \dots, i_k \in \llbracket 1, n \rrbracket$,

$$\langle \varepsilon | \varepsilon \rangle \cdot (t_{i_1} \dots t_{i_k}) = \langle u_{i_1} \dots u_{i_k} | \widetilde{v_{i_k}} \dots \widetilde{v_{i_1}} \rangle.$$

On note $t_{@}$ la touche $@ \blacksquare$.

Montrons maintenant l'équivalence.

\implies : Soient $i_1, \dots, i_k \in \llbracket 1, n \rrbracket$ tels que $u_{i_1} \dots u_{i_k} = v_{i_1} \dots v_{i_k}$. En appliquant la suite de touches $t_{i_1} \dots t_{i_k}$, on obtient la configuration

$$\langle u_{i_1} \dots u_{i_k} | \widetilde{v_{i_k}}, \dots, \widetilde{v_{i_1}} \rangle.$$

En appliquant ensuite la touche finale $t_{@}$, on obtient la configuration

$$\langle u_{i_1} \dots u_{i_k} @ | \widetilde{v_{i_k}}, \dots, \widetilde{v_{i_1}} \rangle.$$

Donc, en posant $u = u_{i_1} \dots u_{i_k}$, on obtient $u @ \widetilde{u} \in \mathcal{L}(K)$. Donc $\mathcal{L}(K) \cap L_{\text{pal}@} \neq \emptyset$.

\Leftarrow : Soit $w \in \mathcal{L}(K) \cap L_{\text{pal}@}$. Comme $w \in \mathcal{L}(K)$, w est obtenu à partir d'une suite de touches $t_{i_1} \dots t_{i_k} t_{@}$ (car seule $t_{@}$ est finale). La configuration obtenue à partir de cette suite est

$$\langle u_{i_1} \dots u_{i_k} @ | \widetilde{v_{i_k}}, \dots, \widetilde{v_{i_1}} \rangle.$$

Donc $w = u @ \widetilde{v}$ où $u = u_{i_1} \dots u_{i_k}$ et $v = v_{i_1} \dots v_{i_k}$.

De plus, comme $w \in L_{\text{pal}@}$, w est aussi de la forme $w_0 @ \widetilde{w_0}$ pour un $w_0 \in A^*$. Comme $@$ n'est pas une lettre de A , cela impose $u = w_0 = v$. Donc $(u_i, v_i)_{i \in \llbracket 1, n \rrbracket} \in \text{PCP}$. \square

B.2.3. GRK

Preuve de la [proposition 3.2.9](#)

Proposition 3.2.9

Le langage $L \triangleq (a^2)^*(b + b^2)$ n'est pas dans GRK.

Preuve

Par l'absurde, supposons qu'il existe un clavier K de GRK reconnaissant L .

Puisque $b^2 \in L$, il existe une suite de touches τ telle que b^2 est reconnu par τ . Comme on peut écrire b^2 sans l'effacer à partir de ε via la suite de touches τ , on peut écrire b^2 sans l'effacer depuis n'importe quelle configuration, par le lemme d'insensibilité.

Montrons désormais que si $\langle u | v \rangle$ est une configuration atteignable depuis ε alors $v = \varepsilon$. En effet, en posant $\langle u' | v' \rangle \triangleq \langle u | v \rangle \cdot \tau \in L$, alors $u'v'$ est de la forme wv par le théorème fondamental de GREK. De plus, par insensibilité, w contient b^2 comme sous-mot. Comme $wv \in L$, cela impose $v = \varepsilon$.

Le pointeur reste donc toujours à droite du mot, ce qui signifie que notre clavier est équivalent à un clavier de RK. On sait pourtant, par la [proposition 3.1.11](#) que $L \notin \text{RK}$, ce qui prouve que $L \notin \text{GRK}$. \square

B.2.4. GREK

Preuve de la proposition 3.2.11

Proposition 3.2.11

Le langage $L = \{ab^{n+1}a^n \mid n \in \mathbb{N}\}$ n'est pas dans GREK.

Preuve

Supposons par l'absurde qu'il existe $K = (T, F)$ dans GREK reconnaissant L . Pour toute suite de touches $\tau \in T^*$, pouvons que $\langle \varepsilon | \varepsilon \rangle \cdot \tau$ est de la forme $\langle u | a^k \rangle$.

Depuis une configuration $\langle u | v \rangle$ avec v contenant un b , on peut exécuter une suite de touches écrivant $abba$ (et ne l'effaçant pas par insensibilité) et acceptant, car $abba \in L$. Par le théorème 2.3.1 (fondamental de GREK), on obtient un mot de la forme wv avec $abba$ un sous-mot de w , donc $abbab$ est un sous-mot de wv , ce qui est impossible car $wv \in L$.

Il existe une suite de touches $\tau \in T^*$ et une touche $f \in F$ telles que $\langle \varepsilon | \varepsilon \rangle \cdot \tau f = \langle u | a^k \rangle$ avec $ua^k = ab^{\|K\|_\infty+1}a^{\|K\|_\infty}$ car $|t_f| \leq \|K\|_\infty$. On a donc $\langle \varepsilon | \varepsilon \rangle \cdot \tau = \langle abw | a^{k'} \rangle$, avec $\langle w | a^{k'} \rangle \cdot f = b^{\|K\|_\infty}a^{\|K\|_\infty}$. De plus f étant finale, $\langle \varepsilon | \varepsilon \rangle \cdot f = \langle x | y \rangle$ avec xy de la forme $ab^{n+1}a^n$. Par le théorème 2.3.1 (fondamental de GREK), $ab^{n+1}a^n$ est donc un sous-mot de $b^{\|K\|_\infty}a^{\|K\|_\infty}$, ce qui est faux. On en conclut que L n'est pas dans GREK \square

Preuve du théorème 3.2.12

Théorème 3.2.12

Soit L un langage de GREK. Alors L est algébrique et on peut construire en temps polynomial un automate à pile non déterministe reconnaissant L .

Définition B.2.1

Soit $K = (T, F)$ un clavier de GREK. On définit l'automate à pile $\mathcal{A}(K)$ comme suit :

- Son ensemble d'états est $\text{Pref}(T \cup F) \cup \{\text{Fin}\}$.
- Son alphabet d'entrée est A , son alphabet de pile est $A \cup \{\perp\}$, \perp étant le symbole de fond de pile.
- ε est le seul état initial.
- Fin est le seul état final. L'automate accepte uniquement dans l'état Fin avec une pile vide.

— L'ensemble des transitions est $\Delta = \Delta_A \cup \Delta_{\blacktriangleleft} \cup \Delta_{\leftarrow} \cup \Delta_{\text{boucle}} \cup \Delta_{\text{Fin}}$ avec

$$\begin{aligned}\Delta_A &= \left\{ t \xrightarrow[-,\uparrow a]{\varepsilon} ta \mid ta \in \text{Pref}(T \cup F), a \in A \right\} \\ \Delta_{\blacktriangleleft} &= \left\{ t \xrightarrow[\downarrow a, -]{\varepsilon} t\blacktriangleleft \mid t\blacktriangleleft \in \text{Pref}(T \cup F), a \in A \right\} \cup \\ &\quad \left\{ t \xrightarrow[\downarrow\perp, \uparrow\perp]{\varepsilon} t\blacktriangleleft \mid t\blacktriangleleft \in \text{Pref}(T \cup F) \right\} \\ \Delta_{\leftarrow} &= \left\{ t \xrightarrow[\downarrow a, -]{\varepsilon} t\leftarrow \mid t\leftarrow \in \text{Pref}(T \cup F), a \in A \right\} \cup \\ &\quad \left\{ t \xrightarrow[\downarrow\perp, \uparrow\perp]{\varepsilon} t\leftarrow \mid t\leftarrow \in \text{Pref}(T \cup F) \right\} \\ \Delta_{\text{boucle}} &= \left\{ t \xrightarrow[-,-]{\varepsilon} \varepsilon \mid t \in T \right\} \\ \Delta_{\text{Fin}} &= \left\{ t \xrightarrow[-,-]{\varepsilon} \text{Fin} \mid t \in F \right\} \cup \\ &\quad \left\{ \text{Fin} \xrightarrow[\downarrow a, -]{a} \text{Fin} \mid a \in A \right\} \cup \\ &\quad \left\{ \text{Fin} \xrightarrow[\downarrow\perp, -]{\varepsilon} \text{Fin} \right\}\end{aligned}$$

Lemme B.2.2

Pour tout clavier $K = (T, F)$ de GREK, $\mathcal{L}(K) = \mathcal{L}(\widetilde{\mathcal{A}(K)})$.

Preuve

Soient $u, v \in A^*$, montrons que pour toute suite d'opérations élémentaires $\sigma_1 \cdots \sigma_k \in \text{Pref}(T \cup F)$, si on applique depuis l'état ε avec $\perp u$ comme contenu de pile la suite de transitions le long du chemin $(\sigma_1)(\sigma_1\sigma_2) \cdots (\sigma_1 \cdots \sigma_k)$, on lit un mot v' et on obtient un contenu de pile $\perp u'$ tels que $\langle u|v \rangle \cdot \sigma_1 \cdots \sigma_k = \langle u|\widetilde{v'}v \rangle$.

On procède par induction sur k . Si $k = 0$, alors la propriété est trivialement vraie. Si $k > 0$, alors par hypothèse d'induction en appliquant les transitions jusqu'à $\sigma_1 \cdots \sigma_{k-1}$ on lit un mot v'' et on obtient un contenu de pile u'' tels que $\langle u|v \rangle \cdot \sigma_1 \cdots \sigma_{k-1} = \langle u''|\widetilde{v''}v \rangle$.

On distingue quatre cas :

- Si $\sigma_k = a \in A$, alors $\langle u|v \rangle \cdot \sigma_1 \cdots \sigma_k = \langle u''a|v \rangle$. De plus en appliquant la transition de $\sigma_1 \cdots \sigma_{k-1}$ à $\sigma_1 \cdots \sigma_k$ on obtient un contenu de pile $u''a$ et on ne lit pas de lettre.
- Si $\sigma_k = \blacktriangleleft$ ou \leftarrow et $u'' = \varepsilon$, alors $\langle u|v \rangle \cdot \sigma_1 \cdots \sigma_k = \langle u''|v \rangle$. De plus en appliquant la transition de $\sigma_1 \cdots \sigma_{k-1}$ à $\sigma_1 \cdots \sigma_k$ on obtient un contenu de pile u'' et on ne lit pas de lettre.
- Si $\sigma_k = \blacktriangleleft$ et $u'' = wa$, $w \in A^*$, $a \in A$ alors $\langle u|v \rangle \cdot \sigma_1 \cdots \sigma_k = \langle w|av \rangle$. De plus en appliquant la transition de $\sigma_1 \cdots \sigma_{k-1}$ à $\sigma_1 \cdots \sigma_k$ on obtient un contenu de pile w et on lit la lettre a .
- Si $\sigma_k = \leftarrow$ et $u'' = wa$, $w \in A^*$, $a \in A$, alors $\langle u|v \rangle \cdot \sigma_1 \cdots \sigma_k = \langle w|v \rangle$. De plus en appliquant la transition de $\sigma_1 \cdots \sigma_{k-1}$ à $\sigma_1 \cdots \sigma_k$ on obtient un contenu de pile w et on ne lit pas de lettre.

L'induction est prouvée.

De plus toutes les transitions d'un $t \in T$ vers ε sont des ε -transitions sans effet sur la pile.

De part la structure de l'automate, les seules suites d'états menant de ε à Fin sont de la forme $\overline{t_1} \cdots \overline{t_n t'}$ avec $t_1, \dots, t_n \in T$ et $t' \in F$, et où pour tout $t \in T$, \overline{t} désigne la suite de tous les préfixes de t ordonnés par ordre croissant.

On obtient que pour tous $u, v \in A^*$, il existe une exécution de l'automate lisant \tilde{v} depuis ε avec une pile \perp vers un état $t' \in F$ avec une pile $\perp u$ si et seulement si $\langle u|v \rangle = \langle \varepsilon|\varepsilon \rangle \cdot t_1 \cdots t_n t'$ avec $t_1, \dots, t_n \in T$. De plus il y a une seule transition depuis t' , n'affectant pas la pile ni le mot lu, vers *fin*, et dans l'état Fin la pile est vidée en lisant son contenu au fur et à mesure (donc lisant \tilde{u} si le contenu de la pile est $\perp u$).

Donc un mot w est accepté si et seulement si il est de la forme $\tilde{v}\tilde{u}$ avec u, v tels qu'il existe $t' \in F$ et une exécution de ε avec une pile \perp vers t' avec une pile $\perp u$ lisant \tilde{v} , c.à.d. si et seulement s'il existe $u, v \in A^*$ et $t_1, \dots, t_n \in T, t' \in F$ tels que $\tilde{w} = uv$ et $(u, v) = (\varepsilon, \varepsilon) \cdot t_1 \cdots t_n t'$, donc si et seulement si $\tilde{w} \in \mathcal{L}(K)$.

On a bien $\mathcal{L}(\mathcal{A}(K)) = \widetilde{\mathcal{L}(K)}$

□

B.3. Les claviers sans retour

B.3.1. FK

Preuve du théorème 3.3.1 pour FK (de la proposition 3.2.4 pour GK)

Théorème 3.3.1

Si L est fini et dans FK alors $L \subset \{\varepsilon\}$.

Preuve

Nous raisonnons par contraposée. S'il existe w non vide dans L , alors w contient une lettre a . Le lemme 2.4.3 (Itération de lettre) nous donne l'existence de $(w_n)_{n \in \mathbb{N}}$ telle que $\forall n \in \mathbb{N}, |w_n|_a \geq n$.

Donc L est infini.

□

B.3.2. FEK

Preuve du proposition 3.3.5 (RK non inclus dans FEK) pour FEK (de la proposition 3.2.7 pour GEK)

Proposition 3.3.5 (RK non inclus dans FEK)

Le langage L_C engendré par le clavier de RK $\{\leftarrow a\#\$, \leftarrow\leftarrow b\#\$\$\}$ n'est pas dans FEK.

Preuve

Notons $t_a = \leftarrow a\#\$$ et $t_b = \leftarrow\leftarrow b\#\$\$$ le clavier K_0 . Appliquer la touche t_a n fois nous donne le mot $(a\#)^n\$,$ et appliquer la touche t_b n fois nous donne le mot $(b\#)^n\$\$.$

De manière évidente, si $w \in L_C$, alors w termine par un $\$,$ puisque t_a et t_b terminent par un $\$,$ et que le curseur reste toujours à droite dans RK.

Montrons d'abord que si $ubwav \in L_C$ avec $u, w, v \in A^*$ et uw ne contient pas de a , alors w contient un $\$.$

Puisque t_a et t_b sont sous forme normale, effacent au plus deux lettres et écrivent au moins trois lettres, par une récurrence facile on peut montrer que pour toute exécution de K_0 $u_0 \xrightarrow{t_1} u_1 \dots \xrightarrow{t_n} u_n$, pour tout $0 \leq i < n$, u_{i+1} est de la forme $u_i w$, w contenant un a si et seulement si $t_i = t_a$ un b si et seulement si $t_i = t_b$. De plus aucun a ou b n'est jamais effacé.

Si $ubwav \in L$ avec $u, w, v \in A^*$ et pas de a dans uw , une exécution donnant ce mot à partir de ε est de la forme $\tau_b t_a \tau'$, où t_a introduit le a en question. Par la remarque précédente, nécessairement $\tau_b = t_b^k$ pour $k > 0$. On a donc $\varepsilon \cdot \tau_b = xb\#\$\$$ pour $x \in A^*$, d'où $\varepsilon \cdot (\tau_b t_a) = xb\#\$a\#\$.$ Ce a n'étant jamais effacé, le $\$$ qui le précède non plus, et on a donc nécessairement que w contient un $\$.$ On obtient la propriété suivante :

$$\text{Pour tout } ubwav \in L, w \text{ contient un } \$. \quad (\text{B.1})$$

Par l'absurde, supposons qu'il existe $K = (T, F)$ un clavier de FEK reconnaissant ce langage. Comme le langage contient les mots $(a\#)^n\$$ et $(b\#)^n\$\$$ pour n arbitrairement grand, K ne pouvant pas effacer, il existe des touches $t'_a, t'_b \in T$ telles que t'_a écrit des a , éventuellement des $\#$, mais ni b ni $\$,$ et t'_b écrit des b , éventuellement des $\#$, mais ni a ni $\$.$

En exécutant t'_b n fois depuis $\langle \varepsilon | \varepsilon \rangle$, on obtient un mot contenant au moins n b mais aucun $\$.$ Donc si on est dans une configuration $\langle u | v \rangle$ avec $|v| > \|K\|_\infty$, appuyer sur une touche entrée nous fera accepter un mot sans $\$$ à la fin, ce qui est impossible car tous les mots du langage en ont un.

On applique t'_b $4\|K\|_\infty^2$ fois à partir de $\langle \varepsilon | \varepsilon \rangle$, on obtient une configuration $\langle u | v \rangle$ sans $\$$ avec $|u| \geq 3\|K\|_\infty^2$ (car $|v| \leq \|K\|_\infty \leq \|K\|_\infty^2$). On applique ensuite t'_a $2\|K\|_\infty + 1$ fois pour obtenir $\langle x | y \rangle$ toujours sans $\$.$ Si on avait $|y| > \|K\|_\infty$, alors en exécutant une touche acceptante, on ne pourrait obtenir de mot terminant par $\$.$ Ainsi $|y| \leq \|K\|_\infty.$

On pose x_1, x_2 tels que $x = x_1ax_2$, x_1 ne contenant pas de a . Rappelons que l'on a exécuté la touche t'_a $2\|K\|_\infty + 1$ fois depuis $\langle u|v \rangle$, avec $|u| \geq 3\|K\|_\infty^2$ et uv ne contenant pas de a .

On en déduit que x_2 est de taille au plus $(2\|K\|_\infty + 1)\|K\|_\infty$ et donc que $|x_1| = |xy| - |y| - |x_2| - 1 \geq 4\|K\|_\infty^2 + 2\|K\|_\infty + 1 - \|K\|_\infty - (2\|K\|_\infty + 1)\|K\|_\infty - 1 = 2\|K\|_\infty^2$.

Le mot x_1ax_2y contient $2\|K\|_\infty + 1$ a , donc comme $|y| \leq \|K\|_\infty$ et x_1 ne contient pas de a , x_2 est donc de taille au moins $\|K\|_\infty$.

On exécute ensuite une touche acceptante $f \in F$, pour obtenir une configuration acceptée de la forme $\langle x_1ax'_2|y' \rangle$ avec x_1 contenant au moins un b mais ni a ni $\$$. Donc $x_1ax'_2y' \in L_C$ contredit la propriété B.1. \square

Preuve du théorème 3.3.7

Théorème 3.3.7

Tout langage de clavier sans effacement est un langage contextuel.

Preuve

Rappelons que les langages contextuels sont équivalents aux langages de grammaires croissantes (avec mot vide autorisé) [4], c'est-à-dire des grammaire où pour toute production $\alpha \rightarrow \beta$ on a $\alpha \leq \beta$. La preuve consiste à construire une grammaire croissante \mathcal{G} qui reconnaît $\#L\#$, pour L un langage de clavier sans effacement et $\# \notin A$.

Soit L un langage de clavier sans effacement. Il existe $K = (T, F)$ un clavier sans effacement sur l'alphabet A qui reconnaît L . On définit sur l'alphabet $A \cup \{\#\}$ où $\# \notin A$,

$$\mathcal{G} = (\{\mid\}, A, \{S \rightarrow \#|\#\} \cup P \cup P_\varepsilon \cup P_F, S)$$

une grammaire croissante. Les dièses représentent le début et la fin du mot. Dans cette construction, ils ne sont pas enlevés à la toute fin.

On prend $P_\varepsilon = \{S \rightarrow \#\#\}$ s'il existe une touche acceptante $t_F \in F$ qui ne contient pas de symbole de A , et $P_\varepsilon = \emptyset$ sinon. Il reste à déterminer P et P_F .

L'idée est de définir, pour chaque touche t , une règle qui déplace le curseur $|$ et ajoute des symboles de A , en définissant l'action de chaque lettre de t . Pour construire l'action du mot entier, on unifie les actions de ses lettres.

On se munit de variables de mot $\alpha, \beta, \dots \in \mathcal{V}$ (infini dénombrable). Soit U l'union des ensembles de couples suivants :

$$\begin{aligned} U_a &= \{(\alpha|\beta, \alpha a|\beta)\} \text{ pour tout } a \in A \\ U_\blacktriangleleft &= \{(\#|\beta, \#|\beta)\} \cup \{\alpha a|\beta, \alpha|a\beta\} \mid a \in A \\ U_\blacktriangleright &= \{(\alpha|\#, \alpha|\#)\} \cup \{\alpha|a\beta, \alpha a|\beta\} \mid a \in A \end{aligned} \tag{B.2}$$

Pour deux couples $q_1 = (u_1, v_1)$ et $q_2 = (u_2, v_2) \in U$ (alpha-renommés différemment), on définit l'unification de $q_1 \sqcup q_2$ par **echec** si l'unification de v_1 et u_2 est **echec**, et sinon par $(u_1\sigma, v_2\sigma)$, où σ est la substitution telle que $v_1\sigma = u_1\sigma$.

On pose $U_\varepsilon = \emptyset$ pour toute touche $t = s_1 \dots s_n$, $U_t = (\dots (U_{s_1} \star U_{s_2}) \star \dots) \star U_{s_n}$, où la composition¹ \star est définie par

$$U_s \star U_{s'} = \{q \sqcup q' \neq \text{echec} \mid q \in U_s, q' \in U_{s'}\} \quad (\text{B.3})$$

À un ensemble X de couples, on peut maintenant associer l'ensemble des productions $P(X) = \{u\sigma_\varepsilon \rightarrow v\sigma_\varepsilon \mid (u, v) \in X\}$, où σ_ε est la substitution qui à toute variable de mot associe ε . On définit alors $P = \bigcup_{t \in T} P(U_t)$.

Les couples des U_s pour $s \in A \cup \{\blacktriangleleft, \blacktriangleright\}$ sont croissants. De plus, il est facile de voir que l'unification conserve cette croissance, puisque les substitutions la conserve (car chaque variable est présente dans la composante de gauche ssi elle est présente dans la composante de droite du couple). Ainsi, les productions de P sont bien toutes croissantes.

Pour P_F , créé à partir des touches acceptantes, il faut s'assurer de ne pas avoir de production où effacer le curseur $|$ vient faire décroître la production. L'idée est donc de concaténer la production avec des productions strictement croissantes, d'où $P_F = \bigcup_{t_F \in F} P(U'_{t_F})$ où

$$U'_{t_F} = \bigcup_{\substack{t \in T \\ \exists a \in A, a \in t}} U_t \star (U_{t_F} \star \{(\alpha|\beta, \alpha\beta)\}) \quad (\text{B.4}) \quad \square$$

Notons que la taille de P est bornée par $(|T| + |F|)|A|^{2\|K\|_\infty} + 1$.

On voit bien que $L(\mathcal{G}) = \#L\#$.

Preuve du **théorème 3.3.8 (Stabilité par miroir)**

Théorème 3.3.8 (Stabilité par miroir)

FEK et FK sont stables par miroir.

Preuve

On pose $\bar{\cdot}$ le morphisme de mots généré par :

$$\overline{\blacktriangleright} = \blacktriangleleft \quad \overline{\blacktriangleleft} = \blacktriangleright \quad \overline{a} = a\blacktriangleleft \quad \text{si } a \in A$$

Si $\langle u|v \rangle$ est une configuration, on note $\widetilde{\langle u|v \rangle} = \langle \widetilde{v}|\widetilde{u} \rangle$.

1. En fait, \star est associative.

Montrons par récurrence sur une suite d'opérations élémentaires $\sigma_1 \dots \sigma_n$ que $c_0 \xrightarrow{\sigma_1 \dots \sigma_n} c_n$ ssi $\widetilde{c}_0 \xrightarrow{\overline{\sigma_1 \dots \sigma_n}} \widetilde{c}_n$.

Si $n = 0$, la propriété est trivialement vraie.

Si $n > 0$, par hypothèse de récurrence, on a $c_0 \xrightarrow{\sigma_1 \dots \sigma_{n-1}} c_{n-1}$ ssi $\widetilde{c}_0 \xrightarrow{\overline{\sigma_1 \dots \sigma_{n-1}}} \widetilde{c}_{n-1}$. Soit $\langle u|v \rangle = c_{n-1}$ et $c' = \widetilde{c}_{n-1} \cdot \overline{\sigma_n}$.

Cas 1 : $\sigma_n = a$. D'une part $c_n = \langle ua|v \rangle$. D'autre part

$$c' = \langle \widetilde{v}|\widetilde{u} \rangle \cdot (a \blacktriangleleft) = \langle \widetilde{v}a|\widetilde{u} \rangle \cdot \blacktriangleleft = \langle \widetilde{v}|a\widetilde{u} \rangle = \widetilde{c}_n$$

Cas 2 : $\sigma_n = \blacktriangleleft$ et $u = \varepsilon$. D'une part $c_n = \langle \varepsilon|v \rangle$. D'autre part $c' = \langle \widetilde{v}|\varepsilon \rangle \cdot \blacktriangleright = \widetilde{c}_n$.

Cas 3 : $\sigma_n = \blacktriangleleft$ et $u = u'a$ pour $a \in A$. D'une part $c_n = \langle u'|av \rangle$. D'autre part $c' = \langle \widetilde{v}|a\widetilde{u}' \rangle \cdot \blacktriangleright = \widetilde{c}_n$.

Les cas avec $\sigma_n = \blacktriangleleft$ sont symétriques.

La récurrence est terminée, et se généralise immédiatement aux suites de touches.

Soit L in langage de FEK et $K = (T, F)$ un clavier le reconnaissant. Soit $\overline{K} = (\overline{T}, \overline{F})$ où le morphisme est appliqué respectivement sur chacune des touches. Pour tout mot w , l'exécution $\langle \varepsilon|\varepsilon \rangle \xrightarrow{t_1 \dots t_n} w$ peut se renverser en une exécution $\langle \varepsilon|\varepsilon \rangle \xrightarrow{\overline{t_1} \dots \overline{t_n}} \widetilde{w}$ de \overline{K} et vice-versa. Donc $w \in L$ ssi $\widetilde{w} \in \mathcal{L}(\overline{K})$, d'où $\mathcal{L}(\overline{K}) = \widetilde{L}$. De plus, \overline{K} est un clavier de FEK, et est automatique ssi K l'est aussi. \square

B.4. Étude de l'automatisme

B.4.1. FREK

Preuve du théorème 3.4.4

Théorème 3.4.4

Prenons comme alphabet $A = \{a, b, c\}$ Soit $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2$ avec $\mathcal{L}_1 = \{wc\widetilde{w} \mid w \in \{a, b\}^*\}$ et $\mathcal{L}_2 = \{wcc\widetilde{w} \mid w \in \{a, b\}^*\}$. Alors $\mathcal{L} \in \text{FREK} \setminus \text{FRK}$.

On a $\mathcal{L} \in \text{FREK}$ grâce au clavier $\{aa\blacktriangleleft, bb\blacktriangleleft, c\blacksquare, cc\blacksquare\}$.

Nous allons montrer que $\mathcal{L} \notin \text{FRK}$.

On commence par prouver que le curseur reste proche du centre du mot.

Lemme B.4.1

Soient $u, v, u', v' \in A^*$ et $t \in T$ une touche quelconque. On suppose que $\langle u|v \rangle \cdot t = \langle u'|v' \rangle$, que $uv, u'v' \in \mathcal{L}$ et que $uv \neq u'v'$. Alors $\|u\| - \|v\| \leq 2|t| + 3$ et $\|u'\| - \|v'\| \leq 4|t| + 3$.

Preuve

Supposons $||u| - |v|| > 2|t| + 3$. Alors soit $|u| - |v| > 2|t| + 3$ soit $|v| - |u| > 2|t| + 3$. On suppose

$$|u| - |v| > 2|t| + 3$$

car l'autre cas est similaire. On a donc

$$|u| = \frac{|u|}{2} + \frac{|u|}{2} > \frac{|u|}{2} + \frac{|v| + 2|t| + 3}{2}$$

D'où $|u| \geq \frac{|uv|}{2} + |t| + 2$ puisqu'il est entier.

Comme $uv \in \mathcal{L}$, uv est de la forme $wc\tilde{w}$ ou $wcc\tilde{w}$ pour un certain $w \in \{a, b\}^*$, donc u est de la forme wcx ou $wccx$ avec $xv = \tilde{w}$ et $|x| \geq |t| + 1$. On obtient du [lemme 2.1.1 \(Localité\)](#) qu'il existe un préfixe commun w_p à u et u' tel que $|w_p| \geq |u| - |t| \geq \frac{|uv|}{2} + 2$. Donc w_p est de la forme wcy (et $uv = wc\tilde{w}$) ou $wccy$ (et $uv = wcc\tilde{w}$) avec $y \in \{a, b\}^+$. Comme $u'v'$ est dans \mathcal{L} , dans les deux cas $uv = u'v'$, ce qui contredit nos hypothèses.

En conclusion, on a bien $||u| - |v|| \leq 2|t| + 3$.

Une induction immédiate sur n nous donne que pour toute suite d'opérations élémentaires $\sigma_1 \cdots \sigma_n$, en posant $\langle u'' | v'' \rangle = \langle u | v \rangle \cdot \sigma_1 \cdots \sigma_n$, $-n \leq |u| - |u''| \leq n$ et $-n \leq |v| - |v''| \leq n$. De ce fait, $||u'| - |v'|| \leq ||u| - |v|| + ||u| - |u''| + ||v| - |v''| \leq 4|t| + 3$. \square

Lemme B.4.2

Si $K \in \text{FRK}$ reconnaît \mathcal{L} , alors K contient une touche écrivant a .

Preuve

Il existe $t_1 \cdots t_n \in T^*$ tel que $\langle \varepsilon | \varepsilon \rangle \cdot t_1 \cdots t_n = \langle u | v \rangle$ avec $uv = a^{3\|K\|_\infty + 1} cca^{3\|K\|_\infty + 1}$. Soit i minimal tel que le nombre de a dans $\langle u_i | v_i \rangle = \langle \varepsilon | \varepsilon \rangle \cdot t_1 \cdots t_i$ soit maximal, on pose $\langle u_{i-1} | v_{i-1} \rangle = \langle \varepsilon | \varepsilon \rangle \cdot t_1 \cdots t_{i-1}$. En particulier $\langle u_i | v_i \rangle$ contient au moins $6\|K\|_\infty + 2$ a , et $\langle u_{i-1} | v_{i-1} \rangle$ contient strictement moins de a que $\langle u_i | v_i \rangle$ par minimalité de i .

Par le [lemme B.4.1](#), $||u_{i-1}| - |v_{i-1}|| \leq 2|t_i| + 3$. Or par le [lemme 2.1.3 \(Encadrement des tailles\)](#), on a $|u_{i-1}v_{i-1}| \geq |u_i v_i| - |t_i| \geq 5\|K\|_\infty + 2$. Par l'absurde, si on avait $|u_{i-1}| < \|K\|_\infty$ (et donc nécessairement $|u_{i-1}| \leq |v_{i-1}|$), on aurait

$$|u_{i-1}v_{i-1}| \leq |u_{i-1}| + (|u_{i-1}| + 2|t_i| + 3) \leq 2(\|K\|_\infty - 1) + (2\|K\|_\infty + 3) < 4\|K\|_\infty + 2$$

\square

Contradiction. Ce raisonnement fonctionne de même avec v_{i-1} . Ainsi u_{i-1} et v_{i-1} sont de longueurs supérieures ou égales à $\|K\|_\infty$.

Ainsi on peut appliquer le [lemme 2.1.15](#), d'où t_i écrit un a .

Lemme B.4.3

Le langage \mathcal{L} n'est pas reconnu par un clavier de FRK.

Preuve

Supposons qu'il existe un clavier K de FRK reconnaissant \mathcal{L} . Alors, il existe des suites de touches minimales τ_a , τ_{ac} , τ_b et τ_{bc} telles que

$$\begin{aligned}\langle \varepsilon | \varepsilon \rangle \cdot \tau_a &= a^{3\|K\|_\infty+1} cca^{3\|K\|_\infty+1} \\ \langle \varepsilon | \varepsilon \rangle \cdot \tau_{ac} &= a^{3\|K\|_\infty+1} ca^{3\|K\|_\infty+1} \\ \langle \varepsilon | \varepsilon \rangle \cdot \tau_b &= b^{3\|K\|_\infty+1} cc b^{3\|K\|_\infty+1} \\ \langle \varepsilon | \varepsilon \rangle \cdot \tau_{bc} &= b^{3\|K\|_\infty+1} cb^{3\|K\|_\infty+1}.\end{aligned}\quad \square$$

Pour tout $x \in \{a, b, ac, bc\}$, posons $\langle u_x | v_x \rangle = \langle \varepsilon | \varepsilon \rangle \cdot \tau_x$. En posant t'_x la dernière touche de τ_x , et (u'_x, v'_x) la configuration précédant l'application de t'_x , par minimalité de τ_x , $\langle u_x | v_x \rangle \neq (u'_x, v'_x)$, donc par le [lemme B.4.1](#), $\|u_x\| - \|v_x\| \leq 2\|K\|_\infty + 3$. On en déduit, comme $\|u_x v_x\| \geq 6\|K\|_\infty + 3$, que $\|u_x\|, \|v_x\| \geq \|K\|_\infty$.

Par le [lemme B.4.2](#), il existe $t_a \in T$ telle que t_a écrit un a . Par le [corollaire 2.1.13](#), en posant $\langle u_b^a | v_b^a \rangle = \langle \varepsilon | \varepsilon \rangle \cdot \tau_b t_a$ et $\langle u_{bc}^a | v_{bc}^a \rangle = \langle \varepsilon | \varepsilon \rangle \cdot \tau_{bc} t_a$, on a $d_a(u_{bc}^a v_{bc}^a) = d_a(u_b^a v_b^a)$. On pose $d = d_a(u_b^a v_b^a)$ et $\delta = \sum_{x \in A} |t_a|_x - |t_a|_{\leftarrow}$. On a $d \in \mathbb{N}$ car $u_b^a v_b^a$ et $u_{bc}^a v_{bc}^a$ contiennent tous deux au moins un a par le [corollaire 2.1.14 \(Écriture hors contexte\)](#), et donc au moins deux a car ils appartiennent à \mathcal{L} .

On utilise à présent le fait que, comme $u_b^a v_b^a, u_{bc}^a v_{bc}^a \in \mathcal{L}$, si d est pair alors ils sont tous deux dans \mathcal{L}_2 , et si d est impair ils sont tous deux dans \mathcal{L}_1 .

De plus si δ est pair alors par le [lemme 2.1.4 \(Égalité des tailles loin des bords\)](#), $|u_b^a v_b^a| = |u_b v_b| + \delta = 6\|K\|_\infty + 4 + \delta$ est pair et $|u_{bc}^a v_{bc}^a| = |u_{bc} v_{bc}| + \delta = 6\|K\|_\infty + 3 + \delta$ est impair, donc $u_b^a v_b^a \in \mathcal{L}_2$ et $u_{bc}^a v_{bc}^a \in \mathcal{L}_1$, ce qui contredit le fait qu'ils sont tous deux dans le même \mathcal{L}_i .

De même si δ est impair alors $u_b^a v_b^a \in \mathcal{L}_1$ et $u_{bc}^a v_{bc}^a \in \mathcal{L}_2$, ce qui contredit le fait qu'ils sont tous deux dans le même \mathcal{L}_i .

On obtient une contradiction, donc \mathcal{L} n'est pas reconnu par un clavier de FRK.