Keyboards as a New Model of Computation

Yoan GÉRAN
ENS Paris-Saclay, France
yoan.geran@ens-paris-saclay.fr

Bastien LABOUREIX
ENS Paris-Saclay, France
bastien.laboureix@ens-paris-saclay.fr

Corto MASCLE
ENS Paris-Saclay, France
corto.mascle@ens-paris-saclay.fr

Valentin D. RICHARD
ENS Paris-Saclay, France
valentin.richard@ens-paris-saclay.fr

Abstract

We introduce a new formalisation of language computation, called keyboards. We consider a set of atomic operations (writing a letter, erasing a letter, going to the right or to the left) and we define a keyboard as a set of finite sequences of such operations, called keys. The generated language is the set of words obtained by applying some non-empty sequence of those keys. Unlike classical models of computation, every key can be applied anytime. We define various classes of languages based on different sets of atomic operations, and compare their expressive powers. We also compare them to rational, context-free and context-sensitive languages. We obtain a strict hierarchy of classes, whose expressiveness is orthogonal to the one of the aforementioned classical models. We also study closure properties of those classes, as well as fundamental complexity problems on keyboards.

2012 ACM Subject Classification Theory of computation

Keywords and phrases formal languages, models of computation, automata theory

Related Version Here is a French version of the paper: `https://arxiv.org/abs/2102.10182`

Acknowledgements We want to thank Pierre Béaur, Lucas Buéri, Jean-Baptiste Daval, Paul Gastin, Colin Geniet, Valentin Maestracci and Clément Théron for their helpful advice and feedback.

1 Introduction

We present a new formalisation of languages, called keyboards. A keyboard K is a finite set of keys, which are finite sequences of atomic operations, such as writing or erasing a letter, going one position to the right or to the left... The language of K is the set of words obtained by applying a sequence of its keys on an initially empty writing space.

This idea of studying the set generated by a set of algebraic operations is far from new: Many works exist on the sets generated by a subset of elements of an algebraic structure, for instance in the context of semigroup and group theory [2, 7], of matrix monoids [1, 9] or the theory of codes [3]. There is however, to the best of the author's knowledge, no previous work on a model resembling the one presented here.

The atomic operations we use in this paper are the base of other models of computation, such as forgetting automata and erasing automata [4, 5, 10]. The use of those operations was originally to simulate some analysis strategies in linguistics. As a first study of the model, we chose the actions of the operations (backspace and arrows) to behave like an actual keyboard in a text editor.

We can define various classes of languages based on the set of atomic operations we consider, and compare their expressive powers between them, and to well-known classes of languages. We obtain a strict hierarchy of classes, with a wide range of expressiveness and difficulty of comprehension. The expressiveness of keyboards seems to be overall orthogonal to the ones of classical models of computation, which we explain by two key differences with the latter.

First, keyboards are blind and memoryless, in that they do not have states and cannot read the tape at any point. Second, because of this weakness, we can allow operations such as moving in the word or erasing letters without blowing up their expressive power too much.

The main interests of keyboards are: 1. to obtain many deep and complex mathematical questions from a deceptively simple model, and 2. that their expressiveness is very different to the ones of classical models. A language that is simple to express with a keyboard may be more complicated for automata, and vice versa. This paper is meant as a first step in the study and comprehension of keyboards and their languages.

The paper is organised as follows. In Sections 2 and 3 we establish notations and basic definitions. Section 4 and 5 are dedicated to building properties and tools necessary to the study of keyboards. In Section 6 we dive into the specific properties of each keyboard class, and prove the inclusions of some of them in regular, context-free and context-sensitive languages. Then in Section 7, we study the inclusions between those classes, in particular showing that they are all different. Some complexity results are given in Section 8. Finally, in Section 9 we show that keyboard classes are not stable by union or intersection, and that some (but not all) of them are stable by mirror.

All proofs can be found in the appendices.

2 Preliminaries

Given a finite alphabet A, we note A^* the set of finite words over A and A^+ for the set of non-empty ones. Given a word $w = a_1 \cdots a_n \in A^*$, we write $|w|$ for its length and, for all $a \in A$, $|w|_a$ the number of occurrences of a in w. For all $1 \leq i, j \leq n$ we use the notation $w[i]$ for the i^{th} letter of w (i.e. a_i) and $w[i, j]$ for its factor $a_i \cdots a_j$ (and ε if $j < i$). We denote the mirror of w by $\tilde{w} = a_n \cdots a_1$.

We write $\mathrm{Pref}(w)$ for the set of prefixes of w, $\mathrm{Suff}(w)$ for its set of suffixes, $\mathrm{Fact}(w)$ for its set of factors and $\mathrm{Sub}(w)$ for its set of subwords.

We represent a finite automaton A as a tuple $(Q, \Delta, \mathrm{Init}, \mathrm{Fin})$ with Q a finite set of states, $\Delta : Q \times A \to 2^Q$ a transition function, and $\mathrm{Init}, \mathrm{Fin} \subseteq Q$ sets of initial and final states.

We represent a pushdown automaton on A as a tuple $(Q, \Gamma, \bot, \Delta, \mathrm{Init}, \mathrm{Fin})$ with

- Q a finite set of states ;
- Γ a finite stack alphabet ;
- $\bot \in \Gamma$ an initial stack symbol ;
- $\Delta \colon Q \times A \times (\Gamma \cup \{-\})^2 \to 2^Q$ a transition function ;
- Init and Fin sets of initial an final states.

We accept a word on final states with an empty stack. We write transitions as follows:

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

with

- $\mathrm{op}_1 = \uparrow\gamma$ if we unstack $\gamma \in \Gamma$, and $\mathrm{op}_1 = -$ if we do not unstack anything.
- $\mathrm{op}_2 = \downarrow\gamma$ if we add $\gamma \in \Gamma$ on the stack, and $\mathrm{op}_2 = -$ if we do not add anything.

We will use ε-transitions in both finite and pushdown automata to simplify some proofs.

For more details and properties of those models, we refer the reader to [6].

3 Definitions

We fix a finite alphabet A and the following special symbols, taken out of A:

The backspace : $\leftarrow$ The left arrow : $\blacktriangleleft$ The right arrow : $\blacktriangleright$

The set of all symbols is $S \triangleq A \cup \{\leftarrow, \blacktriangleleft, \blacktriangleright\}$. An element of S is called an *atomic operation*.

▶ **Definition 3.1.** *A* configuration *is a pair of words $(u, v) \in A^* \times A^*$. We will use $\mathcal{C}(A)$ to denote the set of configurations over A, and $\langle u|v\rangle$ to denote the configuration (u, v).*

We define the notation $\langle u|v\rangle_i$ as $\langle u|v\rangle_i = \widetilde{u}[-i]$ if $i < 0$ and $v[i]$ if $i > 0$.

▶ **Definition 3.2.** *The* action *of an atomic operation $\sigma \in S$ on a configuration $\langle u|v\rangle$ is written $\langle u|v\rangle \cdot \sigma$ and is defined as follows:*

$$\langle u|v\rangle \cdot a = \langle ua|v\rangle \text{ if } a \in A.$$
$$\langle \varepsilon|v\rangle \cdot \leftarrow = \langle \varepsilon|v\rangle \quad \text{and} \quad \langle u'a|v\rangle \cdot \leftarrow = \langle u'|v\rangle$$
$$\langle \varepsilon|v\rangle \cdot \blacktriangleleft = \langle \varepsilon|v\rangle \quad \text{and} \quad \langle u'a|v\rangle \cdot \blacktriangleleft = \langle u'|av\rangle$$
$$\langle u|\varepsilon\rangle \cdot \blacktriangleright = \langle u|\varepsilon\rangle \quad \text{and} \quad \langle u|av'\rangle \cdot \blacktriangleright = \langle ua|v'\rangle$$

We will sometimes write $\langle u|v\rangle \xrightarrow{\sigma} \langle u'|v'\rangle$ for $\langle u'|v'\rangle = \langle u|v\rangle \cdot \sigma$.

▶ **Example 3.3.** By applying the following sequence of atomic operations $\leftarrow$, a, $\blacktriangleright$, $\blacktriangleright$, b to the configuration $\langle c|d\rangle$, we obtain the following rewriting derivation:

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

▶ **Definition 3.4.** *We define other semantics for atomic operations, called* effective semantics. *The difference with the previous ones is that we forbid application of atomic operations without effect (such as backspace when the left word of the configuration is empty). Formally, given $u, v \in A^*, a \in A$ we have:*

$$\langle u|v\rangle \xrightarrow{a}_e \langle ua|v\rangle \qquad \langle u'a|v\rangle \xrightarrow{\leftarrow}_e \langle u'|v\rangle$$
$$\langle u'a|v\rangle \xrightarrow{\blacktriangleleft}_e \langle u'|av\rangle \qquad \langle u|av'\rangle \xrightarrow{\blacktriangleright}_e \langle ua|v'\rangle$$

We also define the operator $\odot$ by $\langle u|v\rangle \odot \sigma = \langle u'|v'\rangle$ if and only if $\langle u|v\rangle \xrightarrow{\sigma}_e \langle u'|v'\rangle$.

▶ **Definition 3.5.** *A* key *is a sequence of atomic operations, seen as a word on S. We will use $\mathcal{T}(S)$ to denote the set of keys on S (variables $k, t, ...$), or $\mathcal{T}$ if there is no ambiguity.*

▶ **Definition 3.6.** *The action of a key over a configuration is defined inductively as follows:*

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

We extend the notation $\langle u|v\rangle \xrightarrow{t} \langle u'|v'\rangle$ to keys. We define $\langle u|v\rangle \xrightarrow{t}_e \langle u'|v'\rangle$ and $\langle u|v\rangle \odot t$ analogously.

▶ Remark 3.7. We will also consider sequences of keys $\tau = t_1 \dots t_n$. The action of τ is obtained by composing the actions of the t_i, hence applying τ has the same effect as applying sequentially $t_1, \dots, t_n$. Note that τ is seen as a word on $\mathcal{T}(S)$ (and not on S), thus $|\tau|$ is n.

▶ **Definition 3.8.** *The length of a key t, written $|t|$, is its length as a word on S. Further, given $\sigma \in S$, we note $|t|_\sigma$ the number of occurrences of σ in t. The size of a configuration $\langle u|v\rangle$ is defined as $|\langle u|v\rangle| = |u| + |v|$.*

▶ **Definition 3.9.** *Two keys t and t' are equivalent, denoted $t \sim t'$, if for all $u, v \in A^*$, $\langle u|v\rangle \cdot t = \langle u|v\rangle \cdot t'$.*

▶ **Example 3.10.** ε is equivalent to $a\leftarrow$ for all $a \in A$, but not to $\blacktriangleright\blacktriangleleft$, as we have $\langle a|\varepsilon\rangle \cdot \blacktriangleright\blacktriangleleft = \langle \varepsilon|a\rangle$ whereas $\langle a|\varepsilon\rangle \cdot \varepsilon = \langle a|\varepsilon\rangle$.

Example 3.10 illustrates how $\blacktriangleleft$, $\blacktriangleright$ and $\leftarrow$ act differently if one side of the configuration is empty. We will see that these 'edge effects' add some expressiveness compared to the effective semantics, but make proofs more complex.

▶ **Definition 3.11** (Automatic Keyboard)**.** *An automatic keyboard is a finite subset of $\mathcal{T}(S)$.*

▶ **Definition 3.12.** *An* execution *of an automatic keyboard K on a configuration $c_0 \in \mathcal{C}$ is a* ***non-empty*** *finite sequence $\rho = (t_1, c_1), \dots, (t, n+1, c_{n+1}) \in (K \times C)^{n+1}$ $(n \in \mathbb{N})$ such that*

$$\forall i \in [\![1\,; n+1]\!], c_{i-1} \xrightarrow{t_i} c_i.$$

By default, we take as initial configuration $c_0 = \langle \varepsilon|\varepsilon\rangle$. We usually write $c_0 \xrightarrow{\tau} c_{n+1}$ to mean the execution $(\tau[1], c_0 \cdot \tau[1]), ..., (\tau[n+1], c_0 \cdot \tau[1, n+1])$.

▶ **Definition 3.13.** *A word $w \in A^*$ is* recognized *by an automatic keyboard K if there exist $u, v \in A^*$ and an execution $\langle \varepsilon|\varepsilon\rangle \xrightarrow{\tau} \langle u|v\rangle$ such that $w = uv$. The language $\mathcal{L}(K)$ of K is the set of words recognized by K.*

We now define keyboards as automatic keyboards to which we added some final keys 'with entry', which mark the end of the execution.

▶ **Definition 3.14** (Keyboard (with entry))**.** *A keyboard K on S is a pair (T, F) of finite sets $T, F \subset \mathcal{T}(S)$. We call the elements of F the* final keys *of K and the elements of T its* transient *keys.*

▶ **Definition 3.15** (Accepting execution of a keyboard)**.** *Let $K = (T, F)$ be a keyboard and $c_0 = \langle u_0|v_0\rangle$ an initial configuration. An* accepting execution *of K on c_0 is a finite sequence $\rho = (t_1, c_1), \dots, (t_{n+1}, c_{n+1}) \in (T \times C)^n \cdot (F \times C)$ $(n \in \mathbb{N})$ such that*

$$\forall i \in [\![1\,; n+1]\!], c_{i-1} \xrightarrow{t_i} c_i.$$

By default, an accepting execution is on the empty configuration $\langle \varepsilon|\varepsilon\rangle$.

▶ **Definition 3.16.** *A word $w \in A^*$ is* recognized *by a keyboard K if there exist $u, v \in A^*$ and an execution $\langle \varepsilon|\varepsilon\rangle \xrightarrow{\tau} \langle u|v\rangle$ such that $w = uv$. The language $\mathcal{L}(K)$ of K is the set of words recognized by K.*

▶ **Example 3.17.** The keyboard with one transient key `aa` and one final key `a`, recognizes sequences of a of odd length.

▶ Remark 3.18. Let K_a be an automatic keyboard, then $\mathcal{L}(K_a)$ is recognized by the keyboard (K_a, K_a). In all that follows we will see automatic keyboards as a subclass of keyboards.

▶ **Definition 3.19** (Size of a keyboard). *The size of a keyboard $K = (T, F)$ is defined as*

$$\|K\|_\infty = \max\{|t| \mid t \in T \cup F\}.$$

We may also use another measure $|K|$ of the size of K for complexity purposes:

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

▶ **Definition 3.20** (Minimal keyboard). *A minimal keyboard K is a finite subset of A^*. We will note* MK *the class of minimal keyboards.*

▶ Remark 3.21. We construct our keyboard classes through the sets of special operations we allow. Class names are obtained by adding B (for $\leftarrow$), E (for the entry, noted ■), L (for ◀) and A (for ◀ and ▶) to K. We obtain these classes.

MK : $\{\}$	LK : $\{◀\}$	AK : $\{◀, ▶\}$
EK : $\{■\}$	LEK : $\{◀, ■\}$	EAK : $\{◀, ▶, ■\}$
BK : $\{\leftarrow\}$	BLK : $\{◀, \leftarrow\}$	BAK : $\{◀, ▶, \leftarrow\}$
BEK : $\{\leftarrow, ■\}$	BLEK : $\{◀, \leftarrow, ■\}$	BEAK : $\{◀, ▶, \leftarrow, ■\}$

▶ Remark 3.22. We do not consider classes with ▶ without ◀ because, without the ◀ operator, we can only reach configurations of the form $\langle u|\varepsilon\rangle$ and thus ▶ never has any effect.

▶ Remark 3.23. We use the class names above to designate both keyboard classes and language classes. For instance, we will write that L is in AK if there exists a keyboard $K \in$ AK such that $L = \mathcal{L}(K)$.

4 General properties

In this section, we establish some properties on keyboard. Although most of them are quite intuitive, we take the time to be as formal as possible in order to build solid bases for the study of keyboards.

Our first lemma states that applying a key can only affect a bounded part of the word around the cursor.

▶ **Lemma 4.1** (Locality). *Let $t = \sigma_1 \dots \sigma_n$ be a key. If $\langle u|v\rangle \xrightarrow{t} \langle u'|v'\rangle$, then $u[1, |u| - n]$ is a prefix of u' and $v[n + 1, |v|]$ is a suffix of v'.*

Furthermore, $u'[1, |u'| - n]$ is a prefix of u and $v'[n + 1, |v'|]$ is a suffix of v.

Then we formalize the fact that if the cursoris far enough from the ends of the word then we do not have edge effects.

▶ **Lemma 4.2** (Effectiveness far from the edges). *Let $t = \sigma_1 \dots \sigma_n$ be a key, $\langle u|v\rangle$ a configuration and $\langle u_n|v_n\rangle = \langle u|v\rangle \cdot t$. If $n \leq \min(|u|, |v|)$, then $\langle u|v\rangle \xrightarrow{t}_e \langle u_n|v_n\rangle$, meaning that all the arrows and backspaces are applied effectively.*

The two next lemmas bound the variation in length of the configuration when applying a key.

▶ **Lemma 4.3** (Bounds on the lengths). *Let $t = \sigma_1 \dots \sigma_n$ be a key, $\langle u|v\rangle$ a configuration and $\langle u_n|v_n\rangle = \langle u|v\rangle \cdot t$. Then*

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

In particular $\big||u_n v_n| - |uv|\big| \leq n$. Moreover $\big||u_n| - |u|\big| \leq n$ and $\big||v_n| - |v|\big| \leq n$.

▶ **Lemma 4.4** (Length evolution without left edge effects). *Let $t = \sigma_1 \dots \sigma_n$ be a key, $\langle u|v\rangle$ a configuration such that $|u| \geq n$. Let $\langle u_n|v_n\rangle = \langle u|v\rangle \cdot t$, then*

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

Then, we obtain the following lemma that can be used to show that some languages are not recognized by a keyboard.

▶ **Lemma 4.5.** *Let K be a keyboard with language L. Let $(l_n)_{n\in\mathbb{N}}$ be the sequence obtained by sorting the lengths of the words in L by increasing order. Then $(l_{n+1} - l_n)_{n\in\mathbb{N}}$ is bounded by $\|K\|_\infty$.*

▶ **Example 4.6.** The languages $\left\{a^{n^2} \;\middle|\; n \in \mathbb{N}\right\}$ and $\{a^p \mid p \text{ prime}\}$ are not recognized by a keyboard.

The two following lemmas will be useful when studying effective executions.

▶ **Lemma 4.7.** *Let $t = \sigma_1 \dots \sigma_n$ be a key such that $\langle u|v\rangle \xrightarrow{t}_e \langle u_n|v_n\rangle$. Then, for all words x, y, $\langle xu|vy\rangle \xrightarrow{t}_e \langle xu_n|v_n y\rangle$.*

▶ **Lemma 4.8.** *Let $t = \sigma_1 \dots \sigma_n$ be a key, $\langle u|v\rangle$ and $\langle x|y\rangle$ configurations such that $|u| = |x|$ and $|v| = |y|$. Then t acts efficiently from $\langle u|v\rangle$ if and only if it acts efficiently from $\langle x|y\rangle$.*

5 Key behaviour

This section aims at providing tools to describe the behaviour of a key. How can we formally express the intuitive fact that the i^{th} symbol of $c \cdot t$ was written by t or that the i^{th} symbol of c was moved by t? We are going to distinguish letters from t and c in order to keep track of where t writes its letters and how the letters of c were affected.

▶ **Definition 5.1** (Tracking function). *Let $\mathbb{Z}_t$ and $\mathbb{Z}_c$ be two duplicates of $\mathbb{Z}$. We denote by $\overline{k}$ the elements of $\mathbb{Z}_c$ and by $\widehat{k}$ the elements of $\mathbb{Z}_t$.*

We define the tracking functions, one for keys $f_t \colon S^* \to (S \cup \mathbb{Z}_t)^*$, defined as follows: $f_t(\sigma_1 \dots \sigma_n) = \sigma'_1 \dots \sigma'_n$ where

$$\sigma'_i = \begin{cases} \widehat{i} & \text{if } \sigma_i \in A \\ \sigma_i & \text{otherwise} \end{cases}$$

and one for configurations $f_c \colon \mathcal{C}(A) \to \mathbb{Z}_c^* \times \mathbb{Z}_c^*$ defined by

$$f_c(a_1 \dots a_k, b_1 \dots b_j) = \left\langle \overline{-k} \dots \overline{-1} \middle| \overline{1} \dots \overline{j} \right\rangle.$$

By applying $f_t(t)$ to $f_c(c)$, we can keep track of which letters of the configuration and of the key were written, erased, or displaced, and where. We need two copies of $\mathbb{Z}$ to differentiate between the symbols of $f_t(t)$ (added by the key) and $f_c(c)$ (already in the configuration).

▶ **Definition 5.2.** *Let $\langle u|v\rangle$ be a configuration and t a key. We note $\langle u'|v'\rangle = \langle u|v\rangle \cdot t$ and $\langle x|y\rangle = f_c(u, v) \cdot f_t(t)$. We say that t writes its k^{th} symbol at position i from $\langle u|v\rangle$ if $\langle x|y\rangle_i = \widehat{k}$.*

▶ Remark 5.3. Let $t = \sigma_1 \dots \sigma_n$ be a key, $\langle u|v\rangle$ a configuration and $1 \leq j < k \leq n$ integers. Then t writes its k^{th} symbol at position i from $\langle u|v\rangle$ if and only if $\sigma_{j+1} \dots \sigma_n$ writes its $(k-j)^{\text{th}}$ symbol at position i from $\langle u|v\rangle \cdot \sigma_1 \dots \sigma_j$. In particular, t writes its k^{th} symbol at position i from $\langle u|v\rangle$ if and only if $\sigma_k \dots \sigma_n$ writes its 1^{st} symbol from $\langle u|v\rangle \cdot \sigma_1 \dots \sigma_{k-1}$.

We defined an intuitive notion of writing the k^{th} symbol of t. In particular, if t writes its k^{th} symbol in i^{th} position from $\langle u|v\rangle$, then $\langle u'|v'\rangle_i = t_k$, as stated below.

▶ **Proposition 5.4.** *Let $t = \sigma_1 \dots \sigma_n$ be a key, $\langle u|v\rangle$ a configuration. We note*

$$\langle u_n|v_n\rangle = \langle u|v\rangle \cdot t \qquad \langle x'_n|y'_n\rangle = f_c(u,v) \cdot t$$
$$\langle u'_n|v'_n\rangle = \langle u|v\rangle \cdot f_t(t) \qquad \langle x_n|y_n\rangle = f_c(u,v) \cdot f_t(t)$$

Then $|u_n| = |x_n| = |u'_n| = |x'_n|$ and $|v_n| = |y_n| = |v'_n| = |y'_n|$. And for all $a \in A$,

$$\begin{aligned}
\langle u_n|v_n\rangle_j = a \text{ iff } & \langle x_n|y_n\rangle_j = \overline{k} \text{ and } \langle u|v\rangle_k = a && \text{or } \langle x_n|y_n\rangle_j = \widehat{k} \text{ and } t_k = a \\
\text{iff } & \langle u'_n|v'_n\rangle_j = a && \text{or } \langle u'_n|v'_n\rangle_j = \widehat{k} \text{ and } t_k = a \\
\text{iff } & \langle x'_n|y'_n\rangle_j = \overline{k} \text{ and } \langle u|v\rangle_k = a && \text{or } \langle x'_n|y'_n\rangle_j = a. \\
& \text{(iff } a \text{ already in configuration} && \text{or } a \text{ added by } t.)
\end{aligned}$$

Tracking functions are a convenient formalism to show some results on keyboards. Besides, they permit to take multiples points of view.

▶ **Corollary 5.5.** *Let t be a key and $\langle u|v\rangle$ a configuration. Then t writes its k^{th} symbol at position i from $\langle u|v\rangle$ if and only if $(\langle u|v\rangle \cdot f_t(t))_i = \widehat{k}$.*

▶ **Definition 5.6.** *Let t be a key, $\langle u|v\rangle$ a configuration. We say that t writes an a in i^{th} position from $\langle u|v\rangle$ if there exists k such that $t_k = a$ and t writes its k^{th} symbol in position i from $\langle u|v\rangle$. We say that t writes an a from $\langle u|v\rangle$ if t writes an a in some position from $\langle u|v\rangle$.*

Then, we obtain some results, which are direct consequences of Proposition 5.4.

▶ **Proposition 5.7.** *If t writes its k^{th} symbol in i^{th} position from $\langle u|v\rangle$ then $(\langle u|v\rangle \cdot t)_i = t_k$. In particular, if t writes an a in i^{th} position from $\langle u|v\rangle$ then $(\langle u|v\rangle \cdot t)_i = a$ and if t writes an a from $\langle u|v\rangle$ then $\langle u|v\rangle \cdot t$ contains an a.*

▶ **Proposition 5.8.** *Let t be a key and $\langle u|v\rangle$, $\langle u'|v'\rangle$ two configurations such that $|u| = |u'|$ and $|v| = |v'|$. Then t writes its k^{th} symbol in i^{th} position from $\langle u|v\rangle$ if and only if t writes its k^{th} symbol in i^{th} position from $\langle u'|v'\rangle$. In particular, t writes an a in j^{th} position from $\langle u|v\rangle$ if and only if t writes an a in j^{th} position from $\langle x|y\rangle$, and t writes an a from $\langle u|v\rangle$ if and only if t writes an a from $\langle x|y\rangle$.*

This proposition makes explicit the fact that keys cannot read the content of a configuration. This leads to the following characterization.

▶ **Proposition 5.9.** *Let t be a key, $a \in A$ and $\langle u|v\rangle$ a configuration containing no a. Let $\langle u'|v'\rangle = \langle u|v\rangle \cdot t$. t writes an a in position i from $\langle u|v\rangle$ if and only if $\langle u'|v'\rangle_i = a$. In particular, t writes an a from $\langle u|v\rangle$ if and only if $\langle u'|v'\rangle$ contains an a.*

Clearly, if the number of a's in a configuration increases after applying a key then this key writes an a.

▶ **Proposition 5.10.** *Let t be a key and $\langle u|v\rangle$ a configuration. If $|\langle u|v\rangle|_a < |\langle u|v\rangle \cdot t|_a$, then t writes an a from $\langle u|v\rangle$.*

Note that if a key behaves differently from two configurations, then there have to be some edge effects. In what follows we focus on effective executions.

▶ **Proposition 5.11.** *Let t be a key and $\langle u|v\rangle$ and $\langle x|y\rangle$ two configurations on which t acts effectively. Then t writes its k^{th} symbol in i^{th} position from $\langle u|v\rangle$ if and only if t writes its k^{th} symbol in i^{th} position from $\langle x|y\rangle$. Therefore, t writes an a in position i from $\langle u|v\rangle$ if and only if t writes an a in position i from $\langle x|y\rangle$.*

In other words, a key always behaves the same way far from the edges of the configuration.

▶ **Definition 5.12.** *Let t be a key. We say that t ensures an a in position i far from the edges if there exists a configuration $\langle u|v\rangle$ such that t acts effectively on $\langle u|v\rangle$ and t writes an a in position i from $\langle u|v\rangle$.*

Then, we immediately obtain the following propositions.

▶ **Proposition 5.13.** *Let t be a key and $u, v \in A^*$ such that $|u| \geq |t|$ and $|v| > |t|$. Then t ensures an a far from the edges if and only if t writes an a from $\langle u|v\rangle$.*

▶ **Proposition 5.14.** *Let t be a key and $u, v \in A^*$ such that $|u| \geq |t|$ and $|v| \geq |t|$. If $|\langle u|v\rangle|_a < |\langle u|v\rangle \cdot t|_a$, then t ensures an a far from the edges.*

▶ **Proposition 5.15.** *Let t be a key which ensures an a far from the edges and $\langle u|v\rangle$ such that $|u| \geq |t|$ and $|v| \geq |t|$. Then $\langle u|v\rangle \cdot t$ contains an a.*

6 Characterisation of the classes

6.1 Languages of BEK (without the arrows)

To begin, we study keyboards that do not contains any arrows.

6.1.1 MK and EK

MK and EK are quite easy to understand. Indeed, since a key of a minimal keyboard K is just a word on A, $K \subset A^*$.

▶ Remark 6.1. Let $K = \{w_1, \ldots, w_n\}$ be a minimal keyboard. Then $\mathcal{L}(K) = (w_1 + \cdots + w_n)^+$.

EK languages are rather similar.

▶ **Lemma 6.2.** *Let $K = (T, F)$ be a* EK *keyboard. Then $\mathcal{L}(K) = T^*F$ and this regular expression can be computed in $O(|K|)$.*

Thus, we can build in linear time a regular expression that recognizes $\mathcal{L}(K)$.

6.1.2 BK and BEK

As a BEK keyboard has no ◀ operation, the right component of a configuration in an execution of $K \in$ BEK (starting from $\langle\varepsilon|\varepsilon\rangle$) is always empty. Thus, in this part we will sometimes denote u for the configuration $\langle u|\varepsilon\rangle$.

Some of the expressiveness of BEK comes from edge effects. For instance, finite languages are recognized by BK keyboards.

▶ **Example 6.3.** Let L be a finite language and $M = \max\{|w| \mid w \in L\}$. Then L is recognized by $K = \left\{ {}^{\leftarrow^M}w \;\middle|\; w \in L \right\}$.

These edge effects could make BEK languages quite complex.

▶ **Lemma 6.4** (Normal form). *Let $t \in S^*$ be a key from* BEK*. Then there exist $m \in \mathbb{N}$ and $w \in A^*$ such that $t \sim \leftarrow^m w$. Further, m and w can be computed from t in polynomial time.*

Using this normal form, we understand that the action of a BEK key always consists in deleting a bounded number of letters at the end of the word, then adding a bounded number of letters. This reminds us of stacks. Following this intuition, we can easily encode the behavior of a BEK keyboard into a pushdown automaton.

However, BEK is even more narrow since all languages of BEK are regular.

▶ **Theorem 6.5.** *Let K be a* BEK *keyboard. Then, $\mathcal{L}(K)$ is regular and we can build an NFA $\mathcal{A}(K)$ recognizing $\mathcal{L}(K)$ in polynomial time.*

▶ Remark 6.6. There are several ways to prove this result. One of them is to apply the pushdown automaton construction from the proof of Theorem 6.11 (presented later in the paper) in the particular case of BEK. The language of the BEK keyboard is then essentially the stack language of this automaton. A slight adaptation of the classical proof that the stack language of a pushdown automaton is rational then yields the result.

We choose to include another proof in this work, as it is elementary, not much longer than the one aforementioned, and seems more elegant to the authors.

6.2 Languages of BLEK (without the right arrow)

In this section, we allow the use of ◀. With this symbol, we have the possibility to move into the word, and then erase or write letters. It opens a new complexity level.

Moreover, we provide a non-regular language of BLEK, hence showing that it is more expressive than BEK (see Theorem 6.5).

▶ **Example 6.7.** Let $K = \{aa◀, bb◀\}$. Then, $\mathcal{L}(K) = \{u\widetilde{u} \mid u \in (a+b)^+\}$, that is, K recognizes the non-empty palindromes of even length.

Thus, we can represent context-free non-regular languages with BLEK (one can observe that the keyboard of Example 6.7 is actually even in LK).

However, a basic observation helps us to understand the behaviour of a key of BLEK: as we do not have the symbol ▶, we cannot go back to the right and all the letters to the right of the cursor are written forever. The following lemma can be proven easily by induction.

▶ **Lemma 6.8.** *Let $t = \sigma_1 \dots \sigma_n$ be a sequence of atomic operations, and $\langle u|v\rangle$ a configuration. Then, $\langle u|v\rangle \cdot t$ is of the form $\langle u'|v'v\rangle$.*

Then, we can make some assertions about a key observing its result over a configuration.

▶ **Lemma 6.9** (Independence from position). *Let t be a key of* BLEK *and $\langle u|v\rangle$ a configuration. If t writes an a from $\langle u|v\rangle$, then for all configurations $\langle u'|v'\rangle$, t writes an a from $\langle u'|v'\rangle$.*

Moreover, we can refine Lemma 6.8.

▶ **Theorem 6.10** (BLEK fundamental). *Let $t = \sigma_1 \dots \sigma_n$ be a sequence of atomic operations, and $\langle u|v\rangle$ a configuration. We set $\langle x_n|y_n\rangle = \langle\varepsilon|\varepsilon\rangle \cdot t$. Then $\langle u|v\rangle \cdot t$ is of the form $\langle u_n x_n|v_n v\rangle$ with y_n a subword of v_n and u_n a prefix of u.*

These observations help us to better understand BLEK. A key observation is that we can see the left part of the configuration as a stack, which can be modified, and the left part as the fixed one, just as the prefix of a word that has been read by an automaton. We can then recognize (the mirror of) a BLEK language with a pushdown automaton which guesses a sequence of keys, maintains the left part of the configuration in the stack and reads the right part of the configuration.

▶ **Theorem 6.11.** *Let K be a* BLEK *keyboard. Then, $\mathcal{L}(K)$ is context-free and we can build a non-deterministic pushdown automaton $\mathcal{A}(K)$ recognizing $\mathcal{L}(K)$ in polynomial time.*

6.3 Languages of EAK (without backspace)

A third interesting class is EAK, where the backspace is not allowed. Thus, the size of a configuration does not decrease along an execution. The execution of such a keyboard can therefore be easily simulated on a linear bounded automaton, as stated in Theorem 6.14.

In all the proofs of this section we will say that t *writes an* a when t contains an a (as we have no $\leftarrow$ if t contains an a then this a will not be erased when applying t).

▶ **Lemma 6.12.** *Let $K = (T, F)$ be a* EAK *keyboard. Let $u, v \in A^*$, let $\tau \in (T \cup F)^*$ and let $\langle u'|v'\rangle = \langle u|v\rangle \cdot \tau$. Then uv is a subword of $u'v'$. In particular $|uv| \leq |u'v'|$.*

▶ **Lemma 6.13.** *Let $K = (T, F)$ be a* EAK *keyboard, let $w \in \mathcal{L}(K)$. There exists an execution $\tau = t_1 \cdots t_n \in T^*F$ such that $\langle \varepsilon|\varepsilon\rangle \cdot \tau = \langle u|v\rangle$ with $uv = w$ and $n \leq |w|^2 + 1$.*

▶ **Theorem 6.14.** *For all keyboards $K = (T, F)$ of* EAK *we can construct a linear bounded automaton $\mathcal{A}(K)$ of polynomial size recognizing $\mathcal{L}(K)$.*

7 Comparison of the keyboard classes

The characterisations that we provide give us some information about each class independently. We now compare the subclasses of BEAK in order to find out which inclusions hold between them. One of these inclusions, between BAK and BEAK, is especially interesting since it shows that keyboards with entry are strictly more powerful than automatic ones.

We decompose our results into the following propositions. Those establish that a class is included in another if and only if that same inclusion holds between their sets of operators, except possibly for the inclusion of EK and BEK in BAK, which we do not prove or disprove.

To start with, we show that a class containing the left arrow cannot be included in a class lacking it. This is a direct consequence of Example 6.7 and Theorem 6.5 as we have a language of LK which is not rational, and thus not in BEK.

▶ **Proposition 7.1.** LK $\not\subseteq$ BEK.

We continue with the two next propositions, showing that a class containing the entry cannot be included in a class excluding it, except possibly for BAK.

▶ **Proposition 7.2** (EK $\not\subseteq$ BLK)**.** EK *is not included in* BLK.

▶ **Proposition 7.3** (EK $\not\subseteq$ AK)**.** EK *is not included in* AK.

Then we prove that a class with $\leftarrow$ cannot be included in a class without $\leftarrow$.

▶ **Proposition 7.4** (BK $\not\subseteq$ EAK)**.** BK *is not included in* EAK.

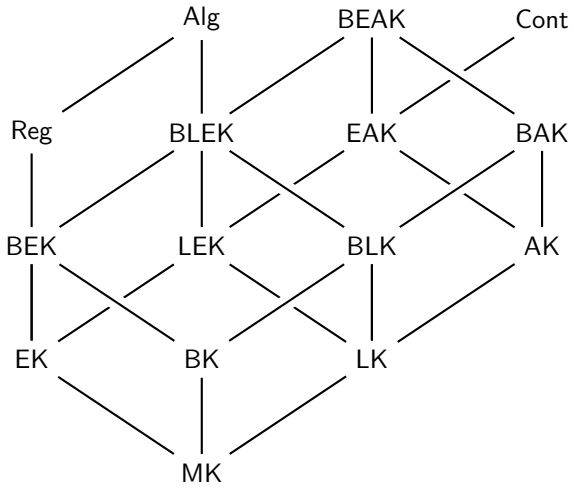

Figure 1 Hierarchy of language classes

The next proposition states that a class containing ▶ cannot be included in a class lacking it.

▶ **Proposition 7.5** (AK $\not\subseteq$ BLEK)**.** AK *is not included in* BLEK*.*

And finally we show that, except possibly for EK and BEK, a class with entry cannot be included in a class without entry.

▶ **Proposition 7.6** (LEK $\not\subseteq$ BAK)**.** LEK *is not included in* BAK*.*

The inclusion Hasse diagram of all subclasses of BEAK and traditional language classes is displayed in Figure 1.

8 Complexity results

In this section we establish some complexity upper bounds on the membership and universality problems for various keyboard classes. The following three propositions are direct consequences of the known complexity bounds of the models which we translated keyboards into (in Remark 6.1, Lemma 6.2, Theorem 6.5 and Theorem 6.11).

▶ **Proposition 8.1.** *The membership problem on* MK *and* EK *is in PTIME. The universality problem is in PTIME on* MK *and PSPACE on* EK*.*

The problem for EK seems simple: it amounts to deciding, given two finite sets of words T and F, if T^*F is universal. This problem may relate to the factor universality problem, proven to be PSPACE-complete [8].

▶ **Proposition 8.2.** *The membership problem over* BEK *is in PTIME, and the universality problem over* BEK *in PSPACE.*

▶ **Proposition 8.3.** *The membership problem over* BLEK *is in PTIME.*

For BK keyboards, we prove that there exists a word not accepted by a given BK keyboard if and only if there exists one of polynomial length.

▶ **Proposition 8.4.** *The universality problem for* BK *keyboards is in* coNP*.*

For EAK keyboards, we know by Lemma 6.13 that every word w recognized by a EAK keyboard can be written with an execution of length polynomial in $|w|$, hence this proposition:

▶ **Proposition 8.5.** *The membership problem for* EAK *keyboards is in NP.*

9 Closure properties

In this section we study closure properties of keyboard classes. We selected three operators, the union and the intersection, as they are the most natural closure operators, and the mirror, under which some classes are stable.

▶ **Proposition 9.1** (Mirror)**.** MK*,* AK *and* EAK *are stable by mirror.* EK*,* BK*,* BEK *and* BLK *are not stable by mirror.*

▶ **Proposition 9.2** (Intersection)**.** *None of the keyboard language classes are stable by intersection.*

▶ **Proposition 9.3** (Union)**.** *None of the keyboard language classes are stable by union.*

We end this section with an undecidability result, showing that intersecting keyboards can lead to highly complex languages. This shows another link with context-free languages as the emptiness of the intersection of two context-free languages is undecidable as well.

▶ **Proposition 9.4** (Intersection emptiness problem)**.** *The following problem is undecidable:*
Input: K_1, K_2 *two* LK *keyboards.*
Output: *Is* $\mathcal{L}(K_1) \cap \mathcal{L}(K_2)$ *empty?*

10 Conclusion

A natural question when it comes to models of computation is what we can do without any memory or any information on the current state of the system. We initiated a line of research aiming at studying such 'blind' models. The one we considered here, keyboards, proved to be mathematically complex and interestingly orthogonal in expressiveness to several of the most classical models. We have established a number of properties of keyboards, as well as a vocabulary facilitating their study. We separated almost all classes and compared their expressiveness, thereby uncovering the lattice of their power.

Future work

As keyboards are a completely new model, there are many open problems we are working on or intend to address. We conjecture that EK is not included in BAK, but we do not have a proof. We also conjecture that not all AK languages are algebraic (the language generated by $\{a\blacktriangleright\blacktriangleright, b\blacktriangleleft\blacktriangleleft\}$ is a candidate as a counter-example) and that BEAK does not contain all rational languages ($a^* + b^*$ being a potential counter-example). We plan on extending the set of operations to add, for instance, a right erasing operator, symmetric to $\leftarrow$. It could also be interesting to study the semantics in which we forbid non-effective operations. Finally, we could equip the model with states and transitions labelled with keys. We would then have more control over which keys are applied at which times, thus increasing the expressiveness of the model and facilitating its study.

References

1. L. Babai and E. Szemeredi. On the complexity of matrix group problems i. In *25th Annual Symposium onFoundations of Computer Science, 1984.*, pages 229–240, 1984. doi:10.1109/SFCS.1984.715919.
2. Martin Beaudry. Membership testing in commutative transformation semigroups. *Information and Computation*, 79(1):84–93, 1988. doi:https://doi.org/10.1016/0890-5401(88)90018-1.
3. Jean Berstel, Dominique Perrin, and Christophe Reutenauer. *Codes and Automata*, volume 129 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2010.
4. Petr Jančar, Frantiek Mráz, and Martin Plátek. Forgetting automata and the chomsky hierarchy. In *Proc. SOFSEM'92*, 1993.
5. Petr Jančar, František Mráz, and Martin Plátek. Characterization of context-free languages by erasing automata. In Ivan M. Havel and Václav Koubek, editors, *Mathematical Foundations of Computer Science 1992*, pages 307–314, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.
6. Rajeev Motwani John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Pearson Education, Limited, 2013.
7. Neil D. Jones and William T. Laaser. Complete problems for deterministic polynomial time. *Theoretical Computer Science*, 3(1):105–117, 1976. doi:https://doi.org/10.1016/0304-3975(76)90068-2.
8. Maksymilian Mika and Marek Szykuła. The frobenius and factor universality problems of the kleene star of a finite set of words. *J. ACM*, 68(3), March 2021. doi:10.1145/3447237.
9. Michael S Paterson. Unsolvability in 3× 3 matrices. *Studies in Applied Mathematics*, 49(1):105–107, 1970.
10. Burchard von Braunmühl and Rutger Verbeek. Finite-change automata. In K. Weihrauch, editor, *Theoretical Computer Science 4th GI Conference*, pages 91–100, Berlin, Heidelberg, 1979. Springer Berlin Heidelberg.

A Proofs of general properties

Proof of Lemma 4.1

▶ **Lemma 4.1** (Locality). *Let $t = \sigma_1 \dots \sigma_n$ be a key. If $\langle u|v\rangle \xrightarrow{t} \langle u'|v'\rangle$, then $u[1, |u| - n]$ is a prefix of u' and $v[n+1, |v|]$ is a suffix of v'.*

Furthermore, $u'[1, |u'| - n]$ is a prefix of u and $v'[n+1, |v'|]$ is a suffix of v.

Proof. Let $t = \sigma_1 \dots \sigma_n$. We set $\langle u_0|v_0\rangle = \langle u|v\rangle$ and $\langle u_i|v_i\rangle = \langle u_0|v_0\rangle \cdot \sigma_1 \cdots \sigma_i$ for all $0 \leq i \leq n$ (note that for all $0 < i \leq n$, $\langle u_i|v_i\rangle = \langle u_{i-1}|v_{i-1}\rangle \cdot \sigma_i$).

Further, for all i we use the notations $w_p(u, i) = u[1, |u| - i]$ and $w_s(v, i) = v[i + 1, |v|]$.

We will now show by induction on i that for all $0 \leq i \leq n$, $w_p(u, i)$ is a prefix of u_i and $w_s(v, i)$ a suffix of v_i. This clearly holds for $i = 0$ ($u_0 = u$ et $v_0 = v$).

Now suppose the property holds for some $i \in [\![0, n-1]\!]$. Then $w_p(u, i)$ is a prefix of u_i and $w_s(v, i)$ a suffix of v_i. Note that $w_p(u, i+1)$ is a prefix of $w_p(u, i)$ (and thus of u_i) and $w_s(v, i+1)$ a suffix of $w_p(v, i)$ (and thus of v_i).

We distinguish cases according to the nature of σ_{i+1}.

- If $\sigma_{i+1} = a \in A$, then $u_{i+1} = u_i a$ and $v_{i+1} = v_i$, hence the property holds.
- If $\sigma_{i+1} = \leftarrow$, we have $v_{i+1} = v_i$. In order to show that $w_p(u, i+1)$ is a prefix of u_{i+1}, we consider two subcases
  - If u_i is empty , then $u_{i+1} = u_i$.
  - Otherwise, u_i is of the form $w_p(u, i)u'c$ with $u \in A^*$ and $c \in A$, and thus $u_{i+1} = w_p(u, i)u'$.

  In both cases, $w_p(u, i+1)$ is indeed a prefix of u_{i+1}.
- If $\sigma_{i+1} = \blacktriangleleft$, we again consider two subcases.
  - If u_i is empty, then $u_{i+1} = u_i$ and $v_{i+1} = v_i$.
  - Otherwise, u_i is of the form $w_p(u, i)u'c$ for some $c \in A$, thus $u_{i+1} = w_p(u, i)u'$ et $v_{i+1} = cv_i$.

  We obtain the result in both cases.
- If $\sigma_{i+1} = \blacktriangleright$, we proceed similarly.
  - If v_i is empty, then $u_{i+1} = u_i$ and $v_{i+1} = v_i$.
  - Otherwise, v_i is of the form $cv'w_s(v, i)$ for some $c \in A$ hence $u_{i+1} = u_i c$ and $v' = v'w_s(v, i)$.

  We obtain the result in both cases.

The first part of the lemma is proven as it corresponds to the case $i = n$.

A nearly identical induction gives the second part of the lemma. ◀

Proof of Lemma 4.2

▶ **Lemma 4.2** (Effectiveness far from the edges). *Let $t = \sigma_1 \dots \sigma_n$ be a key, $\langle u|v\rangle$ a configuration and $\langle u_n|v_n\rangle = \langle u|v\rangle \cdot t$. If $n \leq \min(|u|, |v|)$, then $\langle u|v\rangle \xrightarrow{t}_e \langle u_n|v_n\rangle$, meaning that all the arrows and backspaces are applied effectively.*

Proof. Let us proceed by induction on n. If $n = 0$, then the property trivially holds.

Now suppose the property holds for some n, we prove it for $n + 1$. Let $t = \sigma_1 \dots \sigma_{n+1}$ be a key, and $\langle u|v\rangle$ a configuration such that $n + 1 \leq \min(|u|, |v|)$. Let t' be the key $\sigma_1 \dots \sigma_n$. There exists $\langle u_n|v_n\rangle$ such that $\langle u|v\rangle \xrightarrow{t'}_e \langle u_n|v_n\rangle$. Further, by Lemma Lemma 4.1 we have

that u_n and v_n are non-empty. Thus u_n is of the form $u'a$ and v_n of the form bv' for some $a, b \in A$. We have

$\langle u_n|v_n\rangle \xrightarrow{c}_{\mathrm{e}} \langle u_n c|v_n\rangle$ if $c \in A$.
$\langle u_n|v_n\rangle \xrightarrow{\leftarrow}_{\mathrm{e}} \langle u'|v_n\rangle$.
$\langle u_n|v_n\rangle \xrightarrow{\blacktriangleleft}_{\mathrm{e}} \langle u'|av_n\rangle$.
$\langle u_n|v_n\rangle \xrightarrow{\blacktriangleright}_{\mathrm{e}} \langle u_n b|v'\rangle$.

Thus, no matter the nature of σ_{n+1}, there exists a configuration $\langle u_{n+1}|v_{n+1}\rangle$ such that $\langle u_n|v_n\rangle \xrightarrow{\sigma_{n+1}}_{\mathrm{e}} \langle u_{n+1}|v_{n+1}\rangle$ and therefore $\langle u|v\rangle \xrightarrow{t}_{\mathrm{e}} \langle u_{n+1}|v_{n+1}\rangle$.

The lemma is proven. ◀

Proof of Lemma 4.3

▶ **Lemma 4.3** (Bounds on the lengths). *Let $t = \sigma_1 \dots \sigma_n$ be a key, $\langle u|v\rangle$ a configuration and $\langle u_n|v_n\rangle = \langle u|v\rangle \cdot t$. Then*

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

In particular $\big||u_n v_n| - |uv|\big| \leq n$. Moreover $\big||u_n| - |u|\big| \leq n$ and $\big||v_n| - |v|\big| \leq n$.

Proof. Let us proceed by induction on n. The property clearly holds for $n = 0$.

Suppose the property holds for some n, we show it for $n+1$. Let $t = \sigma_1 \dots \sigma_{n+1}$ be a key, let t' be the key $\sigma_1 \dots \sigma_n$, let $\langle u_n|v_n\rangle = \langle u|v\rangle \cdot t'$ and $\langle u_{n+1}|v_{n+1}\rangle = \langle u|v\rangle \cdot t$. We set

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

We have $M_n \leq |u_n v_n| \leq N_n$ by induction hypothesis. We then consider cases based on the nature of σ_{n+1}

- If $\sigma_{n+1} = a \in A$ then $M_{n+1} = M_n + 1$, $N_{n+1} = N_n + 1$ and $|u_{n+1}v_{n+1}| = |u_n v_n| + 1$.
- If $\sigma_{n+1} \in \{\blacktriangleright, \blacktriangleleft\}$ then $M_{n+1} = M_n$, $N_{n+1} = N_n$ and $|u_{n+1}v_{n+1}| = |u_n v_n|$.
- If $\sigma_{n+1} = \leftarrow$ then $M_{n+1} = M_n - 1$, and $N_{n+1} = N_n$. If u_n is empty then $|u_{n+1}v_{n+1}| = |u_n v_n|$, otherwise $|u_{n+1}v_{n+1}| = |u_n v_n| - 1$.

In all cases we have $M_{n+1} \leq |u_{n+1}v_{n+1}| \leq N_{n+1}$, proving the first part of the lemma. Similar inductions prove the second part, i.e., $\big||u_n| - |u|\big| \leq n$ and $\big||v_n| - |v|\big| \leq n$. ◀

Proof of Lemma 4.4

▶ **Lemma 4.4** (Length evolution without left edge effects). *Let $t = \sigma_1 \dots \sigma_n$ be a key, $\langle u|v\rangle$ a configuration such that $|u| \geq n$. Let $\langle u_n|v_n\rangle = \langle u|v\rangle \cdot t$, then*

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

Proof. We proceed by induction on n. The lemma holds trivially for $n = 0$.

Suppose the property holds for some n. Let $t = \sigma_1 \dots \sigma_{n+1}$ be a key and $\langle u|v\rangle$ a configuration such that and $|v| \geq n+1$. Let $t' = \sigma_1 \dots \sigma_n$, $\langle u_n|v_n\rangle = \langle u|v\rangle \cdot t'$ and $\langle u_{n+1}|v_{n+1}\rangle = \langle u|v\rangle \cdot t$.

By Lemma 4.1, u_n is non-empty. Thus, u_n is of the form $u'a$, with $a, b \in A$. We then have

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

By induction hypothesis applied to t' and $\langle u_n | v_n \rangle$, we have

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

which allows us to obtain the following equality:

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

which proves the lemma. ◀

Proof of Lemma 4.5

▶ **Lemma 4.5.** *Let K be a keyboard with language L. Let $(l_n)_{n \in \mathbb{N}}$ be the sequence obtained by sorting the lengths of the words in L by increasing order. Then $(l_{n+1} - l_n)_{n \in \mathbb{N}}$ is bounded by $\|K\|_\infty$.*

Proof. We start by proving the result for automatic keyboards, and then extend it to the general case.

Let K be an automatic keyboard, L its language, we prove that for all $w \in L$ of size greater than $\|K\|_\infty$, there exists w' such that $w' \neq w$ and

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

proving the proposition.

If for all $w \in L$, $|w| \leq \|K\|_\infty$, then the property is trivially true.

Otherwise, let $w \in L$ be such that $|w| > \|K\|_\infty$. By Lemma 4.3 we know that any execution yielding w is of length superior or equal to 2, as otherwise we would have:

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

Hence there exist $t_0 \dots t_{n+1}$ ($n \in \mathbb{N}$) and $u, v \in A^*$ such that $\langle \varepsilon | \varepsilon \rangle \cdot t_0 \cdots t_{n+1} = \langle u | v \rangle$ and $w = uv$. For all $i \in [\![0, n+1]\!]$, let

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

All w_i are in L (K being automatic) and by Lemma 4.3 we know that for all $i \in [\![0, n]\!]$, $\big||w_{i+1}| - |w_i|\big| \leq \|K\|_\infty$. Further, $0 \leq |w_0| \leq \|K\|_\infty$ and $|w_n| = |w| > \|K\|_\infty$, hence there exists $k \in [\![0, n-1]\!]$ such that $w_k \neq w$ and

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

which demonstrates the lemma for automatic keyboards.

Now let $K = (T, F)$ be a keyboard of language L. We show that for all $w \in L$ of size greater than $5\|K\|_\infty$, there exists $w' \in L$ such that $|w| - 5\|K\|_\infty \leq |w'| < |w|$.

If for all $w \in L$, $|w| \leq 5\|K\|_\infty$, then the result holds trivially.

Otherwise, let w be such that $|w| > 5\|K\|_\infty$. By Lemma 4.3, we cannot obtain w by only executing one key. By seeing T as an automatic keyboard, we have $\|T\|_\infty \leq \|K\|_\infty$ and there exist $w' \in \mathcal{L}(T)$, $f \in F$ and two configurations $\langle u|v\rangle$ and $\langle u'|v'\rangle$ such that

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

By Lemma 4.3, we have $|w'| > 4\|K\|_\infty$. Then the first part of the proof ensures the existence of a word $w'' \in \mathcal{L}(T)$ such that

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

In particular, there exists $\tau \in T^*$ such that $\langle \varepsilon|\varepsilon\rangle \cdot \tau = \langle u''|v''\rangle$ with $u''v'' = w''$. Let $\langle x|y\rangle = \langle u''|v''\rangle \cdot \tau \cdot f$. We have $xy \in L$ and Lemma 4.3 gives us the inequalities

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

and therefore

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

This proves the lemma.

◀

Proof of Lemma 4.7

▶ **Lemma 4.7.** *Let $t = \sigma_1 \dots \sigma_n$ be a key such that $\langle u|v\rangle \xrightarrow{t}_e \langle u_n|v_n\rangle$. Then, for all words x, y, $\langle xu|vy\rangle \xrightarrow{t}_e \langle xu_n|v_n y\rangle$.*

Proof. We use an induction on n. The result is clear for $n = 0$. Let $n > 0$, we have

$$\langle u|v\rangle \xrightarrow{\sigma_1 \dots \sigma_n}_{\mathrm{e}} \langle u_n|v_n\rangle \xrightarrow{\sigma_{n+1}}_{\mathrm{e}} \langle u_{n+1}|v_{n+1}\rangle.$$

By induction hypothesis,

$$\langle xu|vy\rangle \xrightarrow{\sigma_1 \dots \sigma_n}_{\mathrm{e}} \langle xu_n|v_n y\rangle.$$

We separate cases depending on σ_{n+1}.

- If $\sigma_{n+1} = a \in A$, then $\langle u_{n+1}|v_{n+1}\rangle = \langle u_n a|v_n\rangle$ and $\langle xu_n|v_n y\rangle \xrightarrow{a}_{\mathrm{e}} \langle xu_n a|v_n y\rangle$.
- If $\sigma_{n+1} = \leftarrow$, then, $\langle u_{n+1}|v_{n+1}\rangle = \langle u'|v_n\rangle$ and $\langle xu_n|v_n y\rangle \xrightarrow{\leftarrow}_{\mathrm{e}} \langle xu'|v_n y\rangle$.
- If $\sigma_{n+1} = \blacktriangleleft$, then, as the execution is effective, u_n is not empty, hence of the form $u'a$ and thus $\langle u_{n+1}|v_{n+1}\rangle = \langle u'|av_n\rangle$ et $\langle xu_n|v_n y\rangle \xrightarrow{\blacktriangleleft}_{\mathrm{e}} \langle xu'|av_n y\rangle$.
- If $\sigma_{n+1} = \blacktriangleright$, then, as the execution is effective, v_n is not empty, hence of the form av' and thus $\langle u_{n+1}|v_{n+1}\rangle = \langle u_n a|v'\rangle$ et $\langle xu_n|v_n y\rangle \xrightarrow{\blacktriangleright}_{\mathrm{e}} \langle xu_n a|v'y\rangle$.

The induction is proven. ◀

Proof of Lemma 4.8

▶ **Lemma 4.8.** *Let $t = \sigma_1 \dots \sigma_n$ be a key, $\langle u|v\rangle$ and $\langle x|y\rangle$ configurations such that $|u| = |x|$ and $|v| = |y|$. Then t acts efficiently from $\langle u|v\rangle$ if and only if it acts efficiently from $\langle x|y\rangle$.*

Proof. For all $i \in [\![1\,;n]\!]$, we set

$$\begin{cases} \langle u_i|v_i\rangle = \langle u|v\rangle \cdot \sigma_1 \dots \sigma_i \\ \langle x_i|y_i\rangle = \langle x|y\rangle \cdot \sigma_1 \dots \sigma_i \end{cases}$$

An induction on i shows that for all $i \in [\![1\,;n]\!]$, we have $|u_i| = |x_i|$ and $|v_i| = |y_i|$, yielding the result (whether a non effective action happens depends only on the lengths of the x_i and y_i, which are the same as the u_i and v_i). ◀

B Proofs of properties on keys

Proof of Proposition 5.4

▶ **Proposition 5.4.** *Let $t = \sigma_1 \dots \sigma_n$ be a key, $\langle u|v\rangle$ a configuration. We note*

$$\begin{aligned} \langle u_n|v_n\rangle &= \langle u|v\rangle \cdot t & \langle x'_n|y'_n\rangle &= f_c(u,v) \cdot t \\ \langle u'_n|v'_n\rangle &= \langle u|v\rangle \cdot f_t(t) & \langle x_n|y_n\rangle &= f_c(u,v) \cdot f_t(t) \end{aligned}$$

Then $|u_n| = |x_n| = |u'_n| = |x'_n|$ and $|v_n| = |y_n| = |v'_n| = |y'_n|$. And for all $a \in A$,

$$\begin{aligned} \langle u_n|v_n\rangle_j = a \ \textit{iff}\ & \langle x_n|y_n\rangle_j = \overline{k} \ \textit{and}\ \langle u|v\rangle_k = a && \textit{or}\ \langle x_n|y_n\rangle_j = \widehat{k} \ \textit{and}\ t_k = a \\ \textit{iff}\ & \langle u'_n|v'_n\rangle_j = a && \textit{or}\ \langle u'_n|v'_n\rangle_j = \widehat{k} \ \textit{and}\ t_k = a \\ \textit{iff}\ & \langle x'_n|y'_n\rangle_j = \overline{k} \ \textit{and}\ \langle u|v\rangle_k = a && \textit{or}\ \langle x'_n|y'_n\rangle_j = a. \\ & \textit{(iff } a \textit{ already in configuration} && \textit{or } a \textit{ added by } t.) \end{aligned}$$

Proof. For all $i \in [\![1\,;n]\!]$, we set

$$\begin{aligned} \langle u_i|vi\rangle &= \langle u|v\rangle \cdot \sigma_1 \dots \sigma_i & \langle x'_i|y'_n\rangle &= f_c(u,v) \cdot \sigma_1 \dots \sigma_i \\ \langle u'_i|v'_i\rangle &= \langle u|v\rangle \cdot f_t(\sigma_1 \dots \sigma_i) & \langle x_i|y_i\rangle &= f_c(u,v) \cdot f_t(\sigma_1 \dots \sigma_i) \end{aligned}$$

An induction on i then shows that $i \in [\![1\,;n]\!]$,

$$|u_i| = |x_i| = |u'_i| = |x'_i| \quad \text{and} \quad |v_i| = |y_i| = |v'_i| = |y'_i|$$

and for all $a \in A$,

$$\begin{aligned} \langle u_i|v_i\rangle_j = a \text{ iff } & \langle x_i|y_i\rangle_j = \overline{k} \text{ and } \langle u|v\rangle_k = a && \text{or } \langle x_i|y_i\rangle_j = \widehat{k} \text{ and } t_k = a \\ \text{iff } & \langle u'_i|v'_i\rangle_j = a && \text{or } \langle u'_i|v'_i\rangle_j = \widehat{k} \text{ and } t_k = a \\ \text{iff } & \langle x'_i|y'_i\rangle_j = \overline{k} \text{ and } \langle u|v\rangle_k = a && \text{or } \langle x'_i|y'_i\rangle_j = a. \end{aligned}$$

◀

Proof of Proposition 5.9

▶ **Proposition 5.9.** *Let t be a key, $a \in A$ and $\langle u|v\rangle$ a configuration containing no a. Let $\langle u'|v'\rangle = \langle u|v\rangle \cdot t$. t writes an a in position i from $\langle u|v\rangle$ if and only if $\langle u'|v'\rangle_i = a$. In particular, t writes an a from $\langle u|v\rangle$ if and only if $\langle u'|v'\rangle$ contains an a.*

Proof. We set $\langle x|y\rangle = f_c(u,v)\cdot f_t(t)$. By Proposition 5.4, $\langle u'|v'\rangle_i = a$ if and only if $\langle u|v\rangle_i = a$ or $\langle x|y\rangle_i = \overline{k}$ and $t_k = a$.

As $\langle u|v\rangle_i$ does not contain any a, then $\langle x|y\rangle_i = \overline{k}$ and $t_k = a$, i.e., t writes an a from $\langle u|v\rangle$. ◀

Proof of Proposition 5.10

▶ **Proposition 5.10.** *Let t be a key and $\langle u|v\rangle$ a configuration. If $|\langle u|v\rangle|_a < |\langle u|v\rangle\cdot t|_a$, then t writes an a from $\langle u|v\rangle$.*

Proof. We set $\langle u'|v'\rangle = \langle u|v\rangle\cdot t$ and $\langle x|y\rangle = f_c(u,v)\cdot f(t)$. By Proposition 5.4, $\langle u'|v'\rangle_i = a$ if and only if $\langle x|y\rangle_i = \overline{k}$ and $\langle u|v\rangle = a$ or $\langle x|y\rangle_i = k$ and $t_k = a$.

Further, $|\langle u'|v'\rangle|_a > \langle u|v\rangle_a$, thus there exists i such that $\langle x|y\rangle_i = k$ and $t_k = a$, i.e., t writes an a from $\langle u|v\rangle$. ◀

Proof of Proposition 5.11

▶ **Proposition 5.11.** *Let t be a key and $\langle u|v\rangle$ and $\langle x|y\rangle$ two configurations on which t acts effectively. Then t writes its k^{th} symbol in i^{th} position from $\langle u|v\rangle$ if and only if t writes its k^{th} symbol in i^{th} position from $\langle x|y\rangle$. Therefore, t writes an a in position i from $\langle u|v\rangle$ if and only if t writes an a in position i from $\langle x|y\rangle$.*

Proof. Let $n = |u|, m = |v|, p = |x|$ and $q = |y|$ and let $a \in A$. We define

$$\langle u'|v'\rangle = \langle a^n|a^m\rangle\cdot f_t(t) \quad \text{and} \quad \langle x'|y'\rangle = \langle a^p|a^q\rangle\cdot f_t(t).$$

By Corollary 5.5 and Proposition 5.8, t writes its k^{th} symbol at position i from $\langle u|v\rangle$ (resp. from $\langle x|y\rangle$) if and only if $\langle u'|v'\rangle_i = \widehat{k}$ (resp. $\langle x'|y'\rangle_i = \widehat{k}$). Let $N = \max(n,p)$ and $M = \max(m,q)$. By Lemma 4.8, t acts effectively on $\langle a^n|a^m\rangle$ and on $\langle a^p|a^q\rangle$, thus by Lemma 4.7,

$$\left\langle a^N\middle|a^M\right\rangle\cdot f_t(t) = \left\langle a^{N-n}u'\middle|v'a^{M-m}\right\rangle \quad \text{and} \quad \left\langle a^N\middle|a^M\right\rangle\cdot f_t(t) = \left\langle a^{N-p}x'\middle|y'a^{M-q}\right\rangle.$$

In particular, we have that $\left\langle a^N\middle|a^M\right\rangle_i = \widehat{k}$ if and only if $\langle u'|v'\rangle_i = \widehat{k}$ and that $\left\langle a^N\middle|a^M\right\rangle_i = \widehat{k}$ if and only if $\langle x'|y'\rangle_i = \widehat{k}$. The result is proven. ◀

C Proofs of properties of BEK

Proof of Lemma 6.4

▶ **Lemma 6.4** (Normal form). *Let $t \in S^*$ be a key from* BEK*. Then there exist $m \in \mathbb{N}$ and $w \in A^*$ such that $t \sim \leftarrow^m w$. Further, m and w can be computed from t in polynomial time.*

Proof. Suppose k is of the form $k_1 a{\leftarrow}k_2$ with $k_1, k_2 \in S^*$ and $a \in A$. Let u be a configuration, and let $v = u\cdot k_1$. We get

$$\begin{aligned} u\cdot k &= \big((v\cdot a)\cdot\leftarrow\big)\cdot k_2 \\ &= (va\cdot\leftarrow)\cdot k_2 \\ &= v\cdot k_2 \\ &= (u\cdot k_1)\cdot k_2 = u\cdot k_1k_2. \end{aligned}$$

Thus, if k contains a letter followed by a backspace, we can erase both from k to obtain an equivalent key.

By iterating the removal of such patterns as many times as possible, we end up with a key k', equivalent to k and of the form $\leftarrow^m w$ for some $m \in \mathbb{N}$ and $w \in A^*$.

Detecting those patterns can be done in time linear in the size of the key. Further, the size of the key decreases at each step, and thus we only have to iterate at most $|t|$ times. As a result turning a key into normal form requires at most quadratic time. ◀

Proof of Theorem 6.5

▶ **Theorem 6.5.** *Let K be a* BEK *keyboard. Then, $\mathcal{L}(K)$ is regular and we can build an NFA $\mathcal{A}(K)$ recognizing $\mathcal{L}(K)$ in polynomial time.*

We will first show this result for BK. Let K be a keyboard of BK. In order to construct an NFA recognizing $L(K)$, we decompose executions of K into blocks, each block being associated with the writing of a prefix of the final word. Those blocks allow us to see the execution in a "monotonous" way. This decomposition is given by Lemma C.5. The following technical lemmas provide a method to construct an automaton recognizing $L(K)$ in polynomial time. This automaton is defined in Definition C.12.

The following definitions will be useful. We assume all keyboards to be in normal form, i.e., to be subsets of $\leftarrow^* A^*$ (see Lemma 6.4).

▶ **Definition C.1** (Action on natural numbers)**.** *Let K be a keyboard of* BK*. We extend the $\cdot$ operator to natural numbers by setting, for all $n \in \mathbb{N}$ and and $t = \leftarrow^k w \in K$*

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

We extend this definition to K^* by setting, for all $n \in \mathbb{N}$, $t \in K$ and $\tau \in K^*$, $n \cdot \varepsilon = n$ and $n \cdot (\tau t) = (n \cdot \tau) \cdot t$.

▶ Remark C.2. This definition allows us to abstract away the different letters in an execution, to focus on the evolution of the size of the word: For all $u \in A^*$ and $\tau \in K^*$, we have $|u \cdot \tau| = |u| \cdot \tau$.

▶ Remark C.3. We extend the definitions of $\xrightarrow{\tau}$, $\xrightarrow{\tau}_{\mathrm{e}}$ and $\odot$ to natural integers similarly.

Let us start by a quick study of the behaviour of the keys of BKkeyboards. Let $u_0, v \in A^*$, say applying a sequence of keys τ turns u_0 into vu, with u some extra letters. Then the action of τ on u_0 decomposes as follows:

1. At the start u_0 is of the form $v_0 x_0$ with v_0 a common prefix of v and u (and v_0 will not be affected throughout the execution).
2. Then we would like to have a key erasing x_0 and writing letters following v_0 in v, but this key does not necessarily exist. First, some sequence of keys τ_0 modifies x_0 (without affecting v_0), turning it into a word x_0' of different size such that there exists a key $t_0 = \leftarrow^{|x_0'|} v_1 x_1$ erasing $|x_0'|$ and writing $v_1 x_1$ such that $v_0 v_1$ is a prefix of v (and $v_0 v_1$ will not be affected in the rest of the execution).
3. We apply t_1 to obtain $v_0 v_1 x_1$.
4. And so on until we obtain v (plus some extra suffix u)

The following lemma formalizes this idea. Note that in this lemma we only consider effective executions.

▶ **Lemma C.4.** *Let K be a keyboard of* BK *and $u_0, v \in A^*$. Let $\ell \in \mathbb{N}$. There exist $\tau \in K^*$ and $u \in A^*$ such that $|u| = \ell$ and $u_0 \xrightarrow{\tau}_e vu$ if and only if there exist an integer $k > 0$, $v_0, \dots, v_k \in A^+, x_0, \dots, x_k \in A^*$, $0 \leq s_1, \dots, s_k \leq \|K\|_\infty$ and $\tau_0, \dots, \tau_k \in K^*$ such that:*

1. *$u_0 = v_0 x_0$ and $v = v_0 \cdots v_k$.*
2. *For all $1 \leq i \leq k$, $\leftarrow^{s_i} v_i x_i \in K$.*
3. *By setting $s_{k+1} = \ell$, for all $0 \leq i \leq k$, $|x_i| \xrightarrow{\tau_i}_e s_{i+1}$.*

Proof. The $\leftarrow^{s_i} v_i x_i$ are the keys writing parts of v, and the τ_i are the sequences of keys turning x_i into x_i' of length s_{i+1} (allowing us to apply $\leftarrow^{s_{i+1}} v_{i+1} x_{i+1}$).

$\Longleftarrow$ Let $\tau = \tau_0 t_1 \tau_1 \cdots t_k \tau_k$, one easily checks that $u_0 \xrightarrow{\tau}_{\mathrm{e}} vu$ for some u such that $|u| = \ell$.

$\Longrightarrow$ Let $\ell \in \mathbb{N}$, suppose there exists $u \in A^*$ and $\tau \in K^*$ such that $|u| = \ell$ and $u_0 \xrightarrow{\tau}_{\mathrm{e}} vu$. τ is of the form $t_1 \dots t_n$, with $t_i = \leftarrow^{r_i} w_i$.
For all $i \in [\![1\,;n]\!]$, let

$$u_i = u_0 \cdot (t_1 \dots t_i)$$
$$y_i = u_0 \cdot (t_1 \dots t_{i-1}) \leftarrow^{r_i}$$

Thus u_i is the word obtained after applying the i first keys of τ, and y_i is the word obtained after applying the $i-1$ first keys and the erasing part of t_i. In particular, y_i is a prefix of u_{i-1} and $u_i = y_i w_i$.
We also set

$$j = \min\{i \mid \forall i' > i, v \text{ is a prefix of } y_{i'}\}$$

with $j = 0$ if v is a prefix of all y_i. Intuitively, j is the index after which v is written and will no longer be affected. We proceed by induction on v.
If $j = 0$ (which includes the case $v = \varepsilon$), then v is a prefix of all y_i and thus all u_i. By setting $v_0 = v$ and x_0 such that $u_0 = v_0 x_0$, we have $x_0 \xrightarrow{\tau}_{\mathrm{e}} u$. We then obtain the result by setting $k = 0$ and $\tau_0 = \tau$.
If $j > 0$, then $u_j = vz$ (if $j < n$ then v is a prefix of y_{j+1} and thus of u_j, and if $j = n$ then $u_j = vu$) with $z \xrightarrow{t_{j+1} \dots t_n}_{\mathrm{e}} u$. Hence t_j is in fact the last key affecting the v prefix, and $t_{j+1} \dots t_n$ turns $u_j = vz$ into vu without touching v.
There exists u' of length r_j such that $u_{j-1} = y_j u'$. As j is minimal, y_j is not a prefix of v. Further, as they are both prefixes of u_j, y_j is a strict prefix of v.
Then we can apply the induction hypothesis to $v' = y_j$. We have $u_0 \xrightarrow{t_1 \dots t_{j-1}}_{\mathrm{e}} v'u'$, thus there exist $k' \in \mathbb{N}$, $v_0 \in A^*$, $v_1, \dots, v_{k'} \in A^+, x_0, \dots, x_{k'} \in A^*$, $0 \leq s_1, \dots, s_{k'} \leq \|K\|_\infty$ and $\tau_0, \tau_1, \dots, \tau_{k'} \in K^*$ such that:
1. $u_0 = v_0 x_0$ and $v' = v_0 \cdots v_{k'}$.
2. For all $1 \leq i \leq k'$, $\leftarrow^{s_i} v_i x_i \in K$.
3. By setting $s_{k'+1} = |u'| = r_j$, for all $0 \leq i \leq k$, $|w_i| \xrightarrow{\tau_i}_{\mathrm{e}} s_{i+1}$.

We hereby obtain the first part of the wanted decomposition. We then set:

$$k = k' + 1$$
$$\tau_k = t_{j+1} \dots t_n$$
$$v_k \text{ such that } v = v' v_k$$
$$s_k = |u'| = r_j$$
$$s_{k+1} = \ell$$
$$x_{k+1} = z$$

allowing us to satisfy the three conditions of the lemma.

◀

We now slightly refine this lemma. We want to write some word v from ε. The idea is that we can assume that every key of the execution acts effectively except possibly for the first one:

If we write v by applying $t_1 \dots t_n$ and some t_j (with $j > 1$) does not act effectively on $\varepsilon \cdot t_1 \cdots t_{j-1}$, then that means t_j erases all that was written before and applying $t_j \cdots t_n$ instead of $t_1 \cdots t_n$ would yield the same result.

▶ **Lemma C.5** (Decomposition of an BK execution). *Let K be a keyboard of* BK *and let $v \in A^*$. There exists $\tau \in K^+$ such that $\varepsilon \xrightarrow{\tau} v$ if and only if there exists an integer $k > 0$, $v_1, \dots, v_k \in A^*$, $x_1, \dots, x_k \in A^*$, $0 \leq s_1, \dots, s_k \leq \|K\|_\infty$ and $\tau_1, \dots, \tau_k \in K^*$ such that:*

1. $v = v_1 \cdots v_k$.
2. *For all* $1 \leq i \leq k$, $\leftarrow^{s_i} v_i x_i \in K$.
3. *By setting* $s_{k+1} = 0$, *for all* $1 \leq i \leq k$, $|x_i| \xrightarrow{\tau_i}_e s_{i+1}$.

In particular, we have $(\varepsilon \cdot t_1) \xrightarrow{\tau'}_e v$, with $\tau' = t_2 \tau_2 \dots t_n \tau_n$ and $t_1 = \leftarrow^{s_1} v_1 x_1$.

Proof. $\Longleftarrow$ We simply observe that $\tau = \leftarrow^{s_1} v_1 x_1 \tau_1 \cdots \leftarrow^{s_k} v_k x_k \tau_k$ yields v when applied on ε.

$\Longrightarrow$ τ is of the form $t_1 \dots t_n$ with $t_i = \leftarrow^{r_i} w_i$. Let $u_0 = \varepsilon$ and for all $i \in [\![1\,;n]\!]$,

$$u_i = \varepsilon \cdot (t_1 \dots t_i)$$
$$y_i = \varepsilon \cdot (t_1 \dots t_{i-1}) \leftarrow^{r_i}$$

Now let j be the maximal index such that $y_j = \varepsilon$. Then we get $\varepsilon \cdot (t_j \cdots t_n) = v$. Moreover, for all $i > j$, $y_i \neq \varepsilon$, which implies that the action of the t_i for $i > j$ is effective. As a result, $(\varepsilon \cdot t_j) \xrightarrow{t_{j+1} \dots t_n}_{\mathrm{e}} v$.

We finally get the result by applying Lemma C.4 with $\ell = 0$.

◀

The following lemma isolates a part of the previous one for clarity.

▶ **Lemma C.6.** *Let $v \in A^*$, let $K \subseteq \leftarrow^* A^*$ be a keyboard. There exists $\tau \in K^+$ such that $\varepsilon \xrightarrow{\tau} v$ if and only if there exists $t \in K$ and $\tau' \in K^*$ such that $(\varepsilon \cdot t) \xrightarrow{\tau'}_e v$.*

Proof. This is a direct consequence of Lemma C.5. ◀

In order to produce a word v, we have to find a decomposition $v = v_1 \dots v_k$, with some keys $\leftarrow^{r_i} v_i x_i$ and for all i a sequence turning x_i into some x_i' of size r_{i+1}.

This allows us to construct BK, an automaton recognizing its language. The idea behind the construction is to memorize in the states the number of extra letters we have to erase and to have two types of transitions.

- We can go from n_1 to n_2 reading v if there exists a key $\leftarrow^{n_1} vw$ with $|w| = n_2$ (we erase the n_1 extra letters, write v and get a configuration with n_2 letters to erase.
- We can go from n_1 to n_2 with an ε-transition if there exists τ such that $n_1 \xrightarrow{\tau}_{\mathrm{e}} n_2$ (we transform the extra x of size n_1 into an extra x' of size n_2).

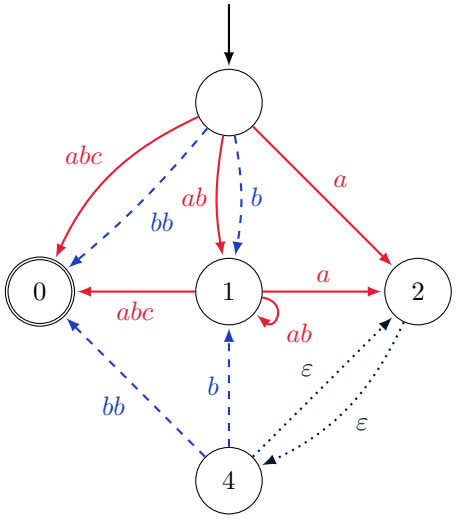

Figure 2 Automaton of $\{\leftarrow\texttt{abc}, \leftarrow^4\texttt{bb}\}$

The automaton then allows us to simulate an execution from its decomposition. If we reach state i after reading some word v, it means there exists a sequence of keys τ leading to configuration vx with x of length i. The only accepting state is then 0. This automaton will be formally defined in Definition C.12.

We gives in Figure 2 the automaton obtained by applying this construction to the keyboard $\{\leftarrow\texttt{abc}, \leftarrow^4\texttt{bb}\}$ (with a little bit simplifications).

Red transitions (solid arrows) denote the action of $\leftarrow\texttt{abc}$ and blue ones (dashed arrows) the action of $\leftarrow^4\texttt{bb}$. A black transition (dotted arrows) from i to j means that there is a sequence of keys turning effectively the i extra letters (that should be erased) into j letters.

However, we want to be able to construct this automaton (efficiently), thus we still have to be able to decide, for all $0 \leq n_1, n_2 \leq \|K\|_\infty$, if there is an execution τ turning n_1 into n_2 effectively (and thus turning the x_i into x'_i). We are now interested in this transformation. We show that we can decide in polynomial time whether there exists a sequence of keys transforming effectively a word of length n_1 into one of length n_2. Note that as the x_i and x'_i from the decomposition are respectively written and erased by a single key, their lengths are at most $\|K\|_\infty$, thus it is enough for us to focus on integers lower or equal to $\|K\|_\infty$.

We start by giving necessary and sufficient conditions for the existence of a sequence of keys turning a word of size n_1 into one of size n_2 effectively. For that we consider several cases.

In our informal explanations we will sometimes confuse the x_i and x'_i with their lengths (as we are interested in their lengths and not their content).

▶ **Lemma C.7.** *Let K be a keyboard of* BK*. We set $E = \{|w|- \mid \leftarrow^r w \in K\}$ and $p = \mathrm{GCD}(E)$. Let x and x' be two integers such that $0 \leq x, x' \leq \|K\|_\infty$. If there exists $\leftarrow^{r_1} w_1, \leftarrow^{r_2} w_2 \in K$ such that*

- *$|w_1| - r_1 < 0 < |w_2| - r_2$;*
- *$r_2 \leq x$ and $|w_1| \leq x'$;*
- *p divides $x' - x$,*

then there exists a sequence of keys $\tau = t_1 \dots t_n \in K^*$ such that $x \xrightarrow{\tau}_e x'$.

Proof. The elements of E are the numbers of letters written by each key, minus the number of letters it erases. We split E between positive integers (corresponding to keys which write

more letters than they erase), and negative ones, (keys which erase more than they write). We are not interested in the keys which write as much as they erase for this proof. We set

$$E_+ = \{i \mid i \in E \text{ and } i > 0\} \text{ and } E_- = \{i \mid i \in E \text{ and } i > 0\}$$

and p^+ and p^- their respective gcds, $p^+ = \mathrm{GCD}(E_+)$ and $p^- = \mathrm{GCD}(E_-)$. We have $\mathrm{GCD}(p^+, p^-) = p$, thus, by Bézout's identity, there exist $\alpha, \beta \in \mathbb{N}$ such that $\alpha p^+ + \beta p^- = p$.

We then consider M a multiple of p^+ and p^- greater than $\|K\|_\infty$ and

$$D = x' + (M^2 + M)(r_1 - |w_1|) - x - M(|w_2| - r_2).$$

D is non-negative ($(M^2 + M)(r_1 - |w_1|) \geq M^2 + M \geq M\|K\| + \|K\| \geq M(|w_2| - r_2) + x$). Further, p divides D, hence there exist $c \in \mathbb{N}$ such that $D = cp$.

We define $A = c\alpha p^+ + M^2$ and $B = c\alpha p^- + M^2$. They are respectively divisible by p^+ and p^- and we further have $A - B = cp = D$.

We have $A \geq M^2 \geq \|K\|_\infty^2 \geq \max(E_+)^2$ and similarly $B \geq \max(E_-)^2$. A result first proven by Schur (and then proven again many times) states that given a finite set S of positive integers, any number greater than $max(S)^2$ and divisible by $\mathrm{GCD}(S)$ can be written as a linear combination combination of elements of S with natural numbers as coefficients (the bound given by Schur is actually better, but we do not need it here). There exist coefficients $(a_i)_{i \in E_+}$ and $(b_i)_{i \in E_-}$ such that

$$\sum_{i \in E_+} i a_i = A \text{ and } \sum_{i \in E_-} i b_i = B.$$

We can now construct the sequence of keys allowing us to go from x to x' effectively.

- We start from x. We apply M times the key $\leftarrow^{r_2} w_2$. The execution is effective ($r_2 < x$ and the key is positive) and leads to $x + M(|w_2| - r_2)$.
- For all $i \in E$, let $t_i = \leftarrow^{r_i} w_i$ be a key such that $i = |w_i| - r_i$. For all $i \in E_+$, we apply a_i times t_i (in an arbitrary order). The execution is effective as we start with more than M letters and we only apply positive keys. We obtain configuration $x + M(|w_2| - r_2) + A$.
- For all $i \in E_-$, we apply b_i times t_i (in an arbitrary order). This leads to configuration

  $$x + M(|w_2| - r_2) + A - B = x + M(|w_2| - r_2) + D = x' + (M^2 + M)(r_1 - |w_1|).$$

  The final configuration is greater than M and we applied only negative keys, thus the execution is again effective.
- Finally, we apply $M + M^2$ times the key $\leftarrow^{r_1} w_1$, yielding the final configuration x'. As $x' \geq r_1$ and as this key is negative, the execution is effective.

We indeed obtain a sequence of keys $\tau \in K^*$ such that $x \xrightarrow{\tau}_{\mathrm{e}} x'$. ◀

The previous lemma presented a necessary condition for the existence of an execution from x to x', the two following ones will give necessary and sufficient conditions in other cases. Another lemma will then use those three to give a necessary and sufficient condition in the general case. To begin with, if every negative key writes more than x' letters, then we can only use positive keys (as we then have no way of getting back to x' from a larger configuration) and in particular $x \leq x'$.

▶ **Lemma C.8.** *Let K be a keyboard of* BK*. Let x and x' be two integers such that $0 \leq x, x' \leq \|K\|_\infty$. Suppose that for all $\leftarrow^r w \in K$ such that $|w| < r$, $|w| > x'$. Then the following conditions are equivalent.*

- *There exists* $\tau \in K^*$ *such that* $x \xrightarrow{\tau}_e x'$.
- *There exists* $\tau = t_1 \dots t_n \in K^*$ *such that* $x \xrightarrow{\tau}_e x'$ *and for all* i, $x \leq x \cdot (t_1 \dots t_i) \leq x'$.

Proof. $\Longleftarrow$ Clear as τ is already constructed.

$\Longrightarrow$ Suppose there exists $\tau = t_1 \dots t_n$ such that $x \xrightarrow{\tau}_e x'$. We proceed by contradiction and assume there exists $i \in [\![1\,;n]\!]$ such that t_i is of the form $\leftarrow^r w$ with $|w| < r$ (and thus $|w| > x'$, by the lemma's hypothesis). Let i be the maximal index satisfying that property.

As $t_{i+1}, \dots, t_n$ write at least as many letters as they erase, we have $x \cdot (t_1 \dots t_i) \leq x' < |w|$. However, $x \cdot (t_1 \dots t_i)$ is the word obtained after applying t_i, thus $x \cdot (t_1 \dots t_i) \geq |w|$, yielding a contradiction.

As a consequence, all keys in τ add at least as many letters as they erase. The sequence $x \cdot (t_1 \dots t_i)$ is therefore nondecreasing, giving us the inequalities:

$$\forall i \in [\![1\,;n]\!], x \leq x \cdot (t_1 \dots t_i) \leq x'.$$

◀

We now present the symmetric case, in which we cannot apply any positive key. The lemma and proof are similar to the previous ones.

▶ **Lemma C.9.** *Let* K *be a keyboard of* BK. *Let* x *and* x' *be two integers such that* $0 \leq x, x' \leq \|K\|_\infty$. *Suppose that for all* $\leftarrow^r w \in K$ *such that* $|w| > r$, $r > x$. *Then the following conditions are equivalent.*

- *There exists* $\tau \in K^*$ *such that* $x \xrightarrow{\tau}_e x'$.
- *There exists* $\tau = t_1 \dots t_n \in K^*$ *such that* $x \xrightarrow{\tau}_e x'$ *and for all* i, $x \geq x \cdot (t_1 \dots t_i) \geq x'$.

Proof. $\Longleftarrow$ Clear as τ is already constructed.

$\Longrightarrow$ Suppose there exists $\tau = t_1 \dots t_n$ such that $x \xrightarrow{\tau}_e x'$. We proceed by contradiction and assume there exists $i \in [\![1\,;n]\!]$ such that t_i is of the form $\leftarrow^r w$ with $|w| > r$ (and thus $r > x$, by the lemma's hypothesis). Let i be the minimal index satisfying that property.

As $t_1, \dots, t_{i-1}$ erase at least as many letters as they write, we have $x \cdot (t_1 \dots t_{i-1}) \leq x < r$. However, t_i acts effectively on $x \cdot (t_1 \dots t_{i-1})$, thus $x \cdot (t_1 \dots t_{i-1}) \geq r$, yielding a contradiction.

As a consequence, all keys in τ erase at least as many letters as they write. The sequence $x \cdot (t_1 \dots t_i)$ is therefore nonincreasing, giving us the inequalities:

$$\forall i \in [\![1\,;n]\!], x \geq x \cdot (t_1 \dots t_i) \geq x'.$$

◀

Equipped with these results, we can now decide in polynomial time if we can go from x to x' effectively.

▶ **Lemma C.10.** *The following problem is decidable in polynomial time.*

$$\begin{cases} \textsc{Input}: & K \text{ a keyboard of } \mathsf{BK} \text{ and } 0 \leq x, x' \leq \|K\|_\infty \\ \textsc{Output}: & \text{Does there exist } \tau = t_1 \dots t_n \in K^* \text{ such that } x \xrightarrow{\tau}_e x' \ ? \end{cases}$$

Proof. Let K be a keyboard of BK and $0 \leq x, x' \leq \|K\|_\infty$. We distinguish four cases:

- If for all $\leftarrow^r w \in K$ such that $|w| - r < 0$ we have $|w| > x'$, then we construct a graph whose vertices are integers from 0 to $\|K\|_\infty$ and such that there is a transition from s to s' if and only if there exists $t \in K$ such that $s \cdot t = s'$. By Lemma C.8, there exists an execution $x \xrightarrow{\tau}_e x'$ if and only if there is a path from x to x' in that graph.

- If for all $\leftarrow^r w \in K$ such that $0 < |w| - r$ we have $r > x$, then we construct the same graph as in the previous case. By Lemma C.9, there exists τ such that $x \xrightarrow{\tau}_{\mathrm{e}} x'$ if and only if there is a path from x to x' in that graph.
- If there exists $\leftarrow^{r_1} w_1, \leftarrow^{r_2} w_2 \in K$ such that $|w_1| - r_1 < 0 < |w_2| - r_2$, $r_2 \leq x$, $|w_1| \leq x'$ and p divides $x' - x$, then, by Lemma C.7, there exists τ such that $x \xrightarrow{\tau}_{\mathrm{e}} x'$.
- Otherwise, p does not divide $x - x'$: a straightforward induction on τ shows that for all $\tau \in K^*$, if τ acts effectively on x, then p divides $(x \cdot \tau) - x$ and thus $x \cdot \tau \neq x'$.

◀

We have all the necessary tools to construct, given a keyboard K of BK, an automaton recognizing the language $\mathcal{L}(K)$.

▶ Remark C.11. In the following definition we will allow transitions to be labelled with words of A^* (of length at most $\|K\|_\infty$) and not only letters. This allows for a clearer presentation and one can easily get from this construction an automaton labelled with letters, by decomposing each transition $n_1 \xrightarrow{a_1 \cdots a_k} n_2$ into a sequence of transitions reading each a_i one by one. As we only use words of size at most $\|K\|_\infty$, the resulting automaton is still of size polynomial in the one of K.

▶ **Definition C.12.** *Let K be a keyboard of* BK*, we define $\mathcal{A}(K)$ the NFA whose states are elements of $\{Init\} \cup [\![0\,;\|K\|_\infty]\!]$.*

The only initial state is Init, the only final state is 0*.*

The set of transitions is $\Delta = \Delta_{Init} \cup \Delta_t \cup \Delta_\tau$ with

$$\Delta_{Init} = \left\{ Init \xrightarrow{v} n \;\middle|\; \exists m, w, \leftarrow^m vw \in K \wedge |w| = n \right\}$$
$$\Delta_t = \left\{ n_1 \xrightarrow{v} n_2 \;\middle|\; \exists w, \leftarrow^{n_1} vw \in K \wedge |w| = n_2 \right\}$$
$$\Delta_\tau = \left\{ n_1 \xrightarrow{\varepsilon} n_2 \;\middle|\; \exists \tau \in K^*, n_1 \xrightarrow{\tau}_e n_2 \right\}$$

▶ **Lemma C.13.** *Let K be a keyboard of* BK*, $\mathcal{A}(K)$ and K recognize the same language.*

Proof. Let $v \in A^*$, suppose $v \in \mathcal{L}(K)$, then by Lemma C.5, there exist $k > 0$, $v_1, \ldots, v_k \in A^*$, $x_1, \ldots, x_k \in A^*$, $0 \leq s_1, \ldots, s_k \leq \|K\|_\infty$ and $\tau_1, \ldots, \tau_k \in K^*$ such that:

1. $v = v_1 \cdots v_k$.
2. For all $1 \leq i \leq k$, $\leftarrow^{s_i} v_i x_i \in K$.
3. By setting $s_{k+1} = 0$, for all $1 \leq i \leq k$, $|x_i| \xrightarrow{\tau_i}_{\mathrm{e}} s_{i+1}$.

One can then easily observe that the execution $\mathrm{Init} \xrightarrow{v_1} |x_1| \xrightarrow{\varepsilon} s_2 \xrightarrow{v_2} |x_2| \cdots \xrightarrow{v_k} |x_k| \xrightarrow{\varepsilon} 0$ accepts v.

Now suppose $v \in \mathcal{L}(\mathcal{A}(K))$, there exists a run on $\mathcal{A}(K)$ accepting v.

First observe that:

- for all state $n \in [\![0\,;\|K\|_\infty]\!]$ there is a transition $n \xrightarrow{\varepsilon} n$;
- for all transitions $n_1 \xrightarrow{\varepsilon} n_2, n_2 \xrightarrow{\varepsilon} n_3$, there is a transition $n_1 \xrightarrow{\varepsilon} n_3$.

As a consequence, we can assume the run of the automaton to be of the form

$$\mathrm{Init} \xrightarrow{v_1} n_1 \xrightarrow{\varepsilon} n_2' \xrightarrow{v_2} n_2 \xrightarrow{\varepsilon} \cdots \xrightarrow{v_k} n_k \xrightarrow{\varepsilon} n_{k+1}' = 0.$$

By definition of $\mathcal{A}(K)$, we can then define:

- for all $1 \leq i \leq k$, $\tau_i \in K^*$ such that $n_i \xrightarrow{\tau_i}_{\mathrm{e}} n_i'$;

- for all $1 \leq i \leq k$, $x_i \in A^*$ and n'_1 such that $\leftarrow^{n'_i} v_i x_i \in K$ and $|x_i| = n_i$;
- $s_i = n'_i$ for all $1 \leq i \leq k+1$.

We obtain that $v \in \mathcal{L}(K)$ by applying Lemma C.5. ◀

▶ **Theorem C.14.** *All languages recognized by a keyboard of* BK *are rational.*

Proof. Let K be a keyboard of BK, by Lemma C.13, its language is the one of $\mathcal{A}(K)$, $\mathcal{L}(K)$ is thus rational. ◀

▶ **Lemma C.15.** *Let K be a keyboard of* BEK*, we can construct $\mathcal{A}(K)$ in polynomial time in the size of K.*

Proof. By Lemma C.10, for all $0 \leq n_1, n_2 \leq \|K\|_\infty$, we can decide in time polynomial in $\|K\|_\infty$ whether there exists $\tau \in K^*$ such that $n_1 \xrightarrow{\tau}_{\mathrm{e}} n_2$.

As a consequence (and by Remark C.11), the automaton $\mathcal{A}(K)$ can be constructed in polynomial time. ◀

We now extend the previous results to BEK. The idea is that an execution of a keyboard (T, F) of BEK, is an execution of T as an BK keyboard followed by the application of a key from F.

▶ **Lemma C.16.** *Let $K = (T, F)$ be a keyboard of* BEK*. We can construct in polynomial time an NFA $\mathcal{A}(K)$ recognizing $\mathcal{L}(K)$.*

Proof. Let $K \triangleq (T, F)$ be a keyboard of BEK. Let $L \triangleq \mathcal{L}(K)$ and $L_T = \mathcal{L}(T)$ where T is seen as a keyboard of BK.

By Lemma C.13 and Lemma C.15, L_T is rational we can construct in polynomial time an NFA recognizing it. Further, note that

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

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

Let f be a final key of the form $\leftarrow^k u$. Then $L_f = (L_T / A^k)u + u$ where

$$L_T / A^k = \left\{ w \;\middle|\; \exists v \in A^k, wv \in L_T \right\}$$

is the right quotient of L_T by A^k and is thus rational.

Union, concatenation and quotient by A^k all correspond to polynomial-time operations on NFAs, proving the result. ◀

A more explicit construction of this automaton is also possible, for instance by transforming the automaton associated with T of BK (see Definition C.12).

- The state 0 is no longer final, and we add a final state Fin.
- For all states i and $t_f = \leftarrow^i w \in F$, we add a transition from i to Fin labeled by w. This simulates the action of, after producing vx (with $|x| = i$) with T, applying t_f.
- For all $t_f = \leftarrow^r w \in F$, we add a transition from Init to Fin labeled by w.

D Proofs of properties of BLEK

Proof of Lemma 6.9

▶ **Lemma 6.9** (Independence from position). *Let t be a key of* BLEK *and $\langle u|v\rangle$ a configuration. If t writes an a from $\langle u|v\rangle$, then for all configurations $\langle u'|v'\rangle$, t writes an a from $\langle u'|v'\rangle$.*

Proof. We set $t = \sigma_1 \cdots \sigma_n$. Let $\langle x|y\rangle$ be a configuration. By Proposition 5.8, we can assume it does not contain any a and by Remark 5.3, we can assume $\sigma_1 = a$ and t writes its first symbol from $\langle u|v\rangle$. We then define for all $1 \leq i \leq n$

$$\langle u_i|v_i\rangle = \langle u|v\rangle \cdot \sigma_1 \dots \sigma_i \qquad \text{and} \qquad \langle x_i|y_i\rangle = \langle x|y\rangle \cdot \sigma_1 \dots \sigma_i.$$

Let i be the smallest index such that u_i or x_i is empty, and $i = n$ if such an index does not exist (note that $i > 1$ as u_1 and x_1 contain the a written by σ_1). As t writes its first symbol from $\langle u|v\rangle$, $\sigma_1 \dots \sigma_i$ writes an a from $\langle u|v\rangle$ in some position j.

Further, $\sigma_1 \dots \sigma_i$ acts efficiently on $\langle u|v\rangle$ and on $\langle x|y\rangle$, thus by Proposition 5.11, t writes an a from $\langle x|y\rangle$ at position j. As either u_i or x_i is empty, we have $j > 0$ and $y_i[j] = v_i[j] = a$. By Lemma 6.8, y_n must then contain an a, while $\langle x|y\rangle$ does not, showing the result by Proposition 5.9. ◀

Proof of Theorem 6.10

▶ **Theorem 6.10** (BLEK fundamental). *Let $t = \sigma_1 \dots \sigma_n$ be a sequence of atomic operations, and $\langle u|v\rangle$ a configuration. We set $\langle x_n|y_n\rangle = \langle \varepsilon|\varepsilon\rangle \cdot t$. Then $\langle u|v\rangle \cdot t$ is of the form $\langle u_n x_n|v_n v\rangle$ with y_n a subword of v_n and u_n a prefix of u.*

Proof. We use an induction on n. If $n = 0$ then the property is clear.

Let $n > 0$, let $\langle u|v\rangle$ be a configuration. We set $\langle x_{n-1}|y_{n-1}\rangle = \langle \varepsilon|\varepsilon\rangle \cdot \sigma_1 \cdots \sigma_{n-1}$ and $c_n = \langle u|v\rangle \cdot \sigma_1 \cdots \sigma_n$.

By induction hypothesis, there exists u_{n-1} and v_{n-1} such that

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

with y_{n-1} a subword of v_{n-1} and u_{n-1} a prefix of u.

We distinguish cases according to σ_n.

- If $\sigma_n = a \in A$ then $x_n = x_{n-1}a$ and $y_n = y_{n-1}$. We set $u_n = u_{n-1}$ et $v_n = v_{n-1}$.
- If $\sigma_n = \leftarrow$ and $x_{n-1} = \varepsilon$, then $x_n = \varepsilon$ and $y_n = y_{n-1}$. We set $v_n = v_{n-1}$ and

$$u_n = \begin{cases} u'_{n-1} & \text{if } u_{n-1} \text{ is of the form } u'_{n-1}a \text{ with } a \in A \\ \varepsilon & \text{otherwise} \end{cases}.$$

- If $\sigma_n = \leftarrow$ and $x_{n-1} = x'_{n-1}a$ with $a \in A$ then $x_n = x'_{n-1}$ and $y_n = y_{n-1}$. We set $u_n = u_{n-1}$ and $v_n = v_{n-1}$.
- If $\sigma_n = \blacktriangleleft$ and $x_{n-1} = \varepsilon$ then $x_n = \varepsilon$ and $y_n = y_{n+1}$. We separate two cases.
  - If $u_{n-1} = \varepsilon$, we set $u_n = \varepsilon$ and $v_n = v_{n-1}$.
  - If $u_{n-1} = u'_{n-1}a$ with $a \in A$, we set $u_n = u'_{n-1}$ and $v_n = av_{n-1}$.
- If $\sigma_n = \blacktriangleleft$ and $x_{n-1} = x'_{n-1}a$ with $a \in A$, then $x_n = x'_{n-1}$ and $y_n = ay_{n-1}$. We set $u_n = u_{n-1}$ and $v_n = av_{n-1}$.

In all cases, we have $c_n = \langle u_n x_n|v_n v\rangle$ with y_n a subword of v_n and u_n a prefix of u.

The theorem is proven. ◀

Proof of Theorem 6.11

▶ **Theorem 6.11.** *Let K be a* BLEK *keyboard. Then, $\mathcal{L}(K)$ is context-free and we can build a non-deterministic pushdown automaton $\mathcal{A}(K)$ recognizing $\mathcal{L}(K)$ in polynomial time.*

▶ **Definition D.1.** *Let $K = (T, F)$ be a keyboard of* BLEK*. We define the pushdown automaton $\mathcal{M}(K)$ as follows.*

- *Its set of states is* $\mathrm{Pref}(T \cup F) \cup \{Fin\}$.
- *Its input alphabet is A, its stack alphabet is $A \cup \{\bot\}$, $\bot$ being its initial stack symbol.*
- *ε is the only initial state.*
- *Fin is the only final state. The automaton accepts on an empty stack in state Fin.*
- *The set of transitions is $\Delta = \Delta_A \cup \Delta_\blacktriangleleft \cup \Delta_\leftarrow \cup \Delta_{loop} \cup \Delta_{Fin}$ avec*

$$\Delta_A = \left\{ t \xrightarrow[-,\uparrow a]{\varepsilon} ta \mid ta \in \mathrm{Pref}(T \cup F), a \in A \right\}$$

$$\Delta_\blacktriangleleft = \left\{ t \xrightarrow[\downarrow a,-]{a} t\blacktriangleleft \mid t\blacktriangleleft \in \mathrm{Pref}(T \cup F), a \in A \right\} \cup \left\{ t \xrightarrow[\downarrow\bot,\uparrow\bot]{\varepsilon} t\blacktriangleleft \mid t\blacktriangleleft \in \mathrm{Pref}(T \cup F) \right\}$$

$$\Delta_\leftarrow = \left\{ t \xrightarrow[\downarrow a,-]{\varepsilon} t\leftarrow \mid t\leftarrow \in \mathrm{Pref}(T \cup F), a \in A \right\} \cup \left\{ t \xrightarrow[\downarrow\bot,\uparrow\bot]{\varepsilon} t\leftarrow \mid t\leftarrow \in \mathrm{Pref}(T \cup F) \right\}$$

$$\Delta_{loop} = \left\{ t \xrightarrow[-,-]{\varepsilon} \varepsilon \mid t \in T \right\}$$

$$\Delta_{Fin} = \left\{ t \xrightarrow[-,-]{\varepsilon} Fin \mid t \in F \right\} \cup \left\{ Fin \xrightarrow[\downarrow a,-]{a} Fin \mid a \in A \right\} \cup \left\{ Fin \xrightarrow[\downarrow\bot,-]{\varepsilon} Fin \right\}$$

▶ **Lemma D.2.** *Let $K = (T, F)$ be a keyboard of* BLEK*, then $\mathcal{L}(K) = \mathcal{L}(\widetilde{\mathcal{M}(K)})$.*

Proof. Let $t = \sigma_1 \cdots \sigma_k \in T$.

We use the notation s^u for the configuration of the automaton $\mathcal{M}(K)$ in state s with u as stack content (from bottom to top).

First we observe that for all configuration s^u of $\mathcal{M}(K)$, for all state s' there is at most one transition from s^u to a configuration of the form $s'^{u'}$.

We also observe that the only accepting runs from ε to Fin are of the form:

$$\varepsilon \xrightarrow{t_1} t_1 \xrightarrow[-,-]{\varepsilon} \varepsilon \xrightarrow{t_2} t_2 \xrightarrow[-,-]{\varepsilon} \varepsilon \cdots t_k \xrightarrow[-,-]{\varepsilon} \varepsilon \xrightarrow{t'} t_f \xrightarrow[-,-]{\varepsilon} \mathrm{Fin} \xrightarrow{read} \mathrm{Fin}$$

with $t_1, \ldots, t_k \in T, t_f \in F$, and where $\xrightarrow{t}$ stands for the run of the form

$$\varepsilon \xrightarrow[op_1,op'_1]{x_1} \sigma_1 \xrightarrow[op_2,op'_2]{x_2} \sigma_1\sigma_2 \cdots \xrightarrow[op_k,op'_k]{x_k} \sigma_1 \cdots \sigma_k$$

and $\mathrm{Fin} \xrightarrow{read} \mathrm{Fin}$ stands for the run

$$\mathrm{Fin} \xrightarrow[-,\uparrow a_1]{a_1} \mathrm{Fin} \xrightarrow[-,\uparrow a_2]{a_2} \cdots \xrightarrow[-,\uparrow a_m]{a_m} \mathrm{Fin} \xrightarrow[-,\uparrow\perp]{\varepsilon} .$$

A straightforward case distinction on σ_i shows that for all $1 \leq i \leq k$, for all $u, v, u' \in A^*$, $x \in A \cup \{\varepsilon\}$ there is a transition reading x from $\sigma_1 \cdots \sigma_{i-1}^{\perp u}$ to $\sigma_1 \cdots \sigma_i^{\perp u'}$ if and only if $\langle u|v\rangle \cdot \sigma_i = \langle u'|vx\rangle$.

We easily infer that for all $t_1, \ldots, t_k \in T, t_f \in F$, the run $\varepsilon \xrightarrow{t_1} t_1 \xrightarrow[-,-]{\varepsilon} \varepsilon \xrightarrow{t_2} t_2 \xrightarrow[-,-]{\varepsilon} \varepsilon \cdots t_k \xrightarrow[-,-]{\varepsilon} \varepsilon \xrightarrow{t'} t_f \xrightarrow[-,-]{\varepsilon}$ Fin reads $\widetilde{v}$ and ends in configuration $\mathrm{Fin}^{\perp u}$, with $\langle u|v\rangle = \langle \varepsilon|\varepsilon\rangle \cdot (t_1 \cdots t_k t_f)$.

As a result, a word w is accepted if and only if there exist $u, v \in A^*$ and $t_1, \ldots t_n \in T, t_f \in F$ such that $w = \widetilde{v}\widetilde{u}$ and $\langle u|v\rangle = \langle \varepsilon|\varepsilon\rangle \cdot t_1 \cdots t_n t_f$.

In other words, a word w is accepted by $\mathcal{M}(K)$ if and only it is in $\mathcal{L}(K)$.

We have shown $\mathcal{L}(\mathcal{M}(K)) = \widetilde{\mathcal{L}(K)}$. ◀

Given a pushdown automaton, one can construct in polynomial time a pushdown automaton recognizing its mirror language. Thus, as $\mathcal{M}(K)$ can clearly be constructed in polynomial time, we obtain the result.

Note that we use ε-transitions in the constructions, however these can be eliminated in polynomial time as well.

E Proofs of EAK results

Proof of Lemma 6.12

▶ **Lemma 6.12.** *Let $K = (T, F)$ be a* EAK *keyboard. Let $u, v \in A^*$, let $\tau \in (T \cup F)^*$ and let $\langle u'|v'\rangle = \langle u|v\rangle \cdot \tau$. Then uv is a subword of $u'v'$. In particular $|uv| \leq |u'v'|$.*

Proof. We simply have to prove the lemma for $\tau = \sigma \in S$ an atomic operation, and the general case follows by a straightforward induction.

If $\sigma \in \{\blacktriangleleft, \blacktriangleright\}$ then $u'v' = uv$ and thus uv is clearly a subword of $u'v'$.

If $\sigma = a \in A$ then $u'v' = uav$ and again uv is clearly a subword of $u'v'$.

This concludes our proof. ◀

Proof of Lemma 6.13

▶ **Lemma 6.13.** *Let $K = (T, F)$ be a* EAK *keyboard, let $w \in \mathcal{L}(K)$. There exists an execution $\tau = t_1 \cdots t_n \in T^*F$ such that $\langle \varepsilon|\varepsilon\rangle \cdot \tau = \langle u|v\rangle$ with $uv = w$ and $n \leq |w|^2 + 1$.*

Proof. Let $w \in \mathcal{L}(K)$, and let $u, v \in A^*$, $\tau = t_1 \cdots t_n \in T^*F$ be such that $w = uv$ and $\langle \varepsilon|\varepsilon\rangle \cdot \tau = \langle u|v\rangle$.

We show that if $n > |w|^2 + 1$ then there exists a shorter execution writing $\langle u|v\rangle$.

Suppose $n > |w|^2 + 1$, and for all $0 \leq i \leq n$ let $k_i = |\langle \varepsilon|\varepsilon\rangle \cdot t_1 \cdots t_i|$ and $\langle u_i|v_i\rangle = \langle \varepsilon|\varepsilon\rangle \cdot t_1 \cdots t_i$. By Lemma 6.12, the sequence (k_i) is nondecreasing. As $n > |w|^2 + 1$ and $k_n = |w|$, there exists $0 \leq i \leq n - |w|$ such that $k_i = k_{i+1} = \cdots = k_{i+|w|}$. Again by Lemma 6.12, we have $u_i v_i = u_{i+1} v_{i+1} = \cdots = u_{i+|w|} v_{i+|w|}$. If $u_i v_i = w$ then the execution $t_1 \cdots t_i$ writes w.

If $u_i v_i \neq w$ then $k_i < |w|$. There are $|w| > k_i$ configurations $\langle u'|v'\rangle$ such that $u'v' = u_i v_i$, thus there exist $i \leq j_1 < j_2 \leq i + |w|$ such that $\langle u_{j_1}|v_{j_1}\rangle = \langle u_{j_2}|v_{j_2}\rangle$.

As a result, the execution $t_1 \cdots t_{j_1} t_{j_2+1} \dots t_n$ writes w.

The lemma is proven. ◀

Proof of Theorem 6.14

▶ **Theorem 6.14.** *For all keyboards $K = (T, F)$ of* EAK *we can construct a linear bounded automaton $\mathcal{A}(K)$ of polynomial size recognizing $\mathcal{L}(K)$.*

Proof. We construct $\mathcal{A}(K)$ the linear bounded automaton which, given an input w, proceeds as follows:

- It divides the tape in three parts: one to memorize the input (of linear size), one to simulate an execution of K (of linear size as well by Lemma 6.12) and one containing a counter (of logarithmic size).
- It guesses a sequence of keys of T followed by a key of F and computes their effect on the fly. After the application of each key the counter is incremented.
- If the counter goes beyond $|w|^2 + 1$ then the automaton rejects.
- If not, the automaton compares the obtained word to w, accepts if they are equal, and rejects otherwise.

This machine guesses a sequence of at most $|w|^2 + 1$ keys and accepts if the word obtained by their actions is the input.

Clearly if a word is accepted by $\mathcal{A}(K)$ then it is in $\mathcal{L}(K)$. Conversely, if a word w is in $\mathcal{L}(K)$ then by Lemma 6.13 there exists an execution of length at most $|w|^2 + 1$ accepting it, thus $\mathcal{A}(K)$ can guess this execution and accept w.

As a result, $\mathcal{L}(K) = \mathcal{L}(\mathcal{A}(K))$. ◀

F Comparisons between classes

▶ **Proposition 7.2** (EK $\not\subseteq$ BLK)**.** EK *is not included in* BLK*.*

Proof. Consider the language $L = (a^2)^*(b + b^2)$, which is clearly in EK via the keyboard $(\{a^2\}, \{b, b^2\})$. We show that there is no BLK keyboard recognizing it.

Suppose there exists a keyboard K of BLK recognizing L.

As $b^2 \in L$, some sequence of keys $\tau_{b^2} \in K^+$ leads to some $\langle x|y\rangle$ with $xy = b^2$ from $\langle\varepsilon|\varepsilon\rangle$.

We first prove that for all $\tau \in K^*$, $\langle\varepsilon|\varepsilon\rangle \cdot \tau$ is of the form $\langle u|\varepsilon\rangle$.

Let $\tau \in K^*$, let $\langle u|v\rangle = \langle\varepsilon|\varepsilon\rangle \cdot \tau$. By Theorem 6.10, $\langle\varepsilon|\varepsilon\rangle \cdot \tau\tau_{b^2}$ is of the form $\langle u'x|v'v\rangle$ with y a subword of v'. As $uv \in L$, uv contains a b, and thus $u'xv'v$ contains at least three, contradicting $u'xv'v \in L$.

Thus the right component of every configuration obtained with K is empty.

We infer that left arrows ◀ are only ever applied on the configuration $\langle\varepsilon|\varepsilon\rangle$, as otherwise we would obtain a nonempty right component which would then stay nonempty throughout the execution by Theorem 6.10, contradicting the previous statement.

We can thus delete every ◀ from the keys of K to obtain an equivalent keyboard of BK.

In what follows we will assume K to be an BK keyboard.

By Lemma 6.4, we can assume it to be in normal form.

As $b^2 \in L$, there exists $\tau \in K^*$ such that $\varepsilon \cdot \tau = b^2$. Again by Lemma 6.4 we have a sequence of atomic operations $\leftarrow^k b^2$ avec $k \in \mathbb{N}$ equivalent to τ.

We distinguish cases depending on k.

- If $k = 0$, then $\tau \sim b^2$; we then have $\varepsilon \cdot \tau \cdot \tau = b^2 \cdot \tau = b^4 \notin L$.

- If $k = 1$, then $\tau \sim \leftarrow b^2$; then $\varepsilon \cdot \tau \cdot \tau = b^2 \cdot \tau = b^3 \notin L$.
- If $k > 1$ and **k is even** ; as $a^{2k}b \in L$, $a^{2k}b$ is a reachable configuration for K. Then we have $a^{2k}b \cdot \tau = a^{k+1}b^2 \notin L$.
- If $k > 1$ and **k is odd** ; as $a^{2k}b^2 \in L$, $a^{2k}b$ is a reachable configuration for K. Then we have $a^{2k}b^2 \cdot \tau = a^k b^2 \notin L$.

In all cases, K recognizes a word outside of L, yielding a contradiction. ◀

▶ **Proposition 7.3** (EK $\not\subseteq$ AK)**.** EK *is not included in* AK*.*

Proof. Consider the language $\{a\}$. It is recognized by the keyboard $(\emptyset, \{a\})$ of EK.

Suppose there exists K a keyboard of AK recognizing $\{a\}$. Then there exists $t \in K$ containing an a. As K does not contain any $\leftarrow$, applying t twice from $\langle\varepsilon|\varepsilon\rangle$ yields a configuration with two a, and thus a word outside of $\{a\}$. We obtain a contradiction. ◀

Proof of Proposition 7.4

In all that follows we will use the notations $t_a = \leftarrow a \lozenge \blacklozenge$ and $t_b = \leftarrow\leftarrow b \lozenge \blacklozenge\blacklozenge$.

We define $L_{\lozenge\blacklozenge}$ as the language recognized by the BK keyboard $\{t_a, t_b\}$.

▶ **Lemma F.1.** *Let $x = x_1 \cdots x_n \in \{a, b\}^+$, we have*

$$\langle\varepsilon|\varepsilon\rangle \cdot t_{x_1} \cdots t_{x_n} = x_1 w_{x_1x_2} x_2 w_{x_2x_3} x_3 \cdots w_{x_{n-1}x_n} x_n v_{x_n}$$

with $w_{aa} = w_{bb} = \lozenge$, $w_{ab} = \varepsilon$, $w_{ba} = \lozenge\blacklozenge$, $v_a = \lozenge\blacklozenge$ and $v_b = \lozenge\blacklozenge\blacklozenge$.
In particular, for all $u_1, u_2, u_3 \in A^*$, if $u_1 b u_2 a u_3 \in L_{\lozenge\blacklozenge}$ then u_2 contains a $\blacklozenge$.

Proof. We proceed by induction on n. For $n = 1$ the property is clear.

Let $n > 1$, let $x = x_1 \cdots x_n \in \{a, b\}^+$, we define $x' = x_1 \cdots x_{n-1}$. Suppose $\langle\varepsilon|\varepsilon\rangle \cdot t_{x_1} \cdots t_{x_{n-1}} = x_1 w_{x_1x_2} \cdots x_{n-1} v_{x_{n-1}}$, we separate four cases:

- $x_{n-1} = x_n = a$: then $v_{x_{n-1}} = \lozenge\blacklozenge$ and thus
$$\langle\varepsilon|\varepsilon\rangle \cdot t_{x_1} \cdots t_{x_n} = x_1 w_{x_1x_2} \cdots x_{n-1} \lozenge a \lozenge\blacklozenge.$$
- $x_{n-1} = x_n = b$: then $v_{x_{n-1}} = \lozenge\blacklozenge\blacklozenge$ and thus
$$\langle\varepsilon|\varepsilon\rangle \cdot t_{x_1} \cdots t_{x_n} = x_1 w_{x_1x_2} \cdots x_{n-1} \lozenge b \lozenge\blacklozenge\blacklozenge.$$
- $x_{n-1} = a$ and $x_n = b$: then $v_{x_{n-1}} = \lozenge\blacklozenge$ and thus
$$\langle\varepsilon|\varepsilon\rangle \cdot t_{x_1} \cdots t_{x_n} = x_1 w_{x_1x_2} \cdots x_{n-1} b \lozenge\blacklozenge\blacklozenge.$$
- $x_{n-1} = b$ and $x_n = a$: then $v_{x_{n-1}} = \lozenge\blacklozenge\blacklozenge$ and thus
$$\langle\varepsilon|\varepsilon\rangle \cdot t_{x_1} \cdots t_{x_n} = x_1 w_{x_1x_2} \cdots x_{n-1} \lozenge\blacklozenge a \lozenge\blacklozenge.$$

The property is verified in all cases. ◀

▶ **Proposition 7.4** (BK $\not\subseteq$ EAK)**.** BK *is not included in* EAK*.*

Proof. Suppose there exists a keyboard $K = (T, F)$ of EAK recognizing $L_{\Diamond\blacklozenge}$. In this proof we will use the letter k to denote keys of K, in order to avoid confusion with the keys t_a and t_b.

As $\Diamond\blacklozenge \in K$, there exists $\tau_f \in T^*F$ yielding that word.

Moreover, $w = (a\Diamond)^{3\|K\|_\infty} a\Diamond\blacklozenge \in L$ (obtained by applying t_a $3\|K\|_\infty$ times). Hence there exists an execution of K writing that word. By Lemma 6.12, as $|k|_a \leq \|K\|_\infty$ for all $k \in T \cup F$, this execution contains at least three keys containing an a, including at least two in T (as the execution only has one final key).

As w contains a single $\blacklozenge$, there exists a key k_a of T writing an a, but writing neither $\blacklozenge$ nor b. By applying k_a then τ_f, we obtain a word of L containing a single $\blacklozenge$ and no b, thus of the form $(a\Diamond)^i a\Diamond\blacklozenge$. We infer that k_a contains as many a and $\Diamond$. Let n be its number of a.

We can similarly show using the word $(b\Diamond)^{3\|K\|_\infty} b\Diamond\blacklozenge\blacklozenge$ that there exists $k_b \in T$ containing as many b and $\Diamond$ but neither $\blacklozenge$ nor a. Let m be its number of b (and of $\Diamond$).

Let $\tau = k_a k_b$. We write

$$\langle u|v\rangle = \langle\varepsilon|\varepsilon\rangle \cdot \tau \quad \text{and} \quad \langle u'|v'\rangle = \langle u|v\rangle \cdot \tau_f$$

We have that $u'v'$ contains a single $\blacklozenge$. As $u'v' \in L$, $u'v'$ has to be of the form $wa\Diamond\blacklozenge$ with w not containing any $\blacklozenge$, $|w|_a = n$, $|w|_b = m$ and $|w|_\Diamond = n + m$.

As wa contains at least a b, wa it is of the form $u_1 b u_2 a$, thus by Lemma F.1, u_2 contains a $\blacklozenge$, yielding a contradiction. As a conclusion, $L \notin \mathsf{EAK}$. ◀

Proof of Proposition 7.5

Consider the keyboard $K = \{\mathtt{a}\blacktriangleleft^2\blacktriangleright\mathtt{b}\}$.

By applying $a\blacktriangleleft^2\blacktriangleright b$ on $\langle\varepsilon|\varepsilon\rangle$ we obtain $\langle ab|\varepsilon\rangle$, and by applying it on a configuration of the form $\langle ub|v\rangle$ we obtain $\langle ubb|av\rangle$. Hence after applying it n times on $\langle\varepsilon|\varepsilon\rangle$ we get $ab^{n+1}a^n$. The language of K is therefore $L = \left\{ab^{n+1}a^n \;\middle|\; n \in \mathbb{N}\right\}$.

▶ **Lemma F.2.** *Let K be a keyboard of* BLEK*. If K recognizes L then for all $\tau \in T^*$, $\langle\varepsilon|\varepsilon\rangle \cdot \tau$ is of the form $\left\langle u\middle|a^k\right\rangle$.*

Proof. As $abba \in L$, there exists $\tau \in T^*$ and $t_f \in F$ such that $\langle\varepsilon|\varepsilon\rangle \cdot \tau t_f = \langle x|y\rangle$ with $xy = abba$.

Let $\langle u|v\rangle$ be a configuration reachable by a sequence of keys of τ with v containing a b. By Theorem 6.10, $\langle u|v\rangle \cdot \tau t_f$ is of the form $\langle u'x|v'v\rangle$ with y a subword of v'. As a consequence, $abba$ is a subword of $u'xv'v$, which contradicts the fact that $u'xv'v \in L$. ◀

▶ **Proposition 7.5** ($\mathsf{AK} \not\subseteq \mathsf{BLEK}$)**.** AK *is not included in* BLEK.

Proof. Let $K = (T, F)$ be a keyboard of BLEK, suppose it recognizes L. Let $\tau \in T^*$ and $t_f \in F$. We set:

$$\langle x|y\rangle = \langle\varepsilon|\varepsilon\rangle \cdot t_f \quad \text{and} \quad \langle u|v\rangle = \langle\varepsilon|\varepsilon\rangle \cdot \tau.$$

There exists $n \in N$ such that $xy = ab^{n+1}a^n$. By Theorem 6.10, $\langle\varepsilon|\varepsilon\rangle \cdot \tau t_f = \langle u|v\rangle \cdot t_f$ is of the form $\langle u'x|v'v\rangle$ with y a subword of v'.

As $ab^{n+1}a^n$ is a subword of xv' and $u'xv'v \in L$, u' is necessarily empty (otherwise u' would contain an a and as ab is a subword of xv', aab would be a subword of $u'xv'v$, contradicting $u'xv'v \in L$).

By Lemma 4.3, $|v'v| - |v| \leq |t| \leq \|K\|_\infty$, i.e., $|v'| \leq \|K\|_\infty$. Furthermore, again by Lemma 4.3, $|x| \leq \|K\|_\infty$.

By Lemma F.2, v is of the form a^k. Hence all b in $u'xv'v$ are in xv', thus $u'xv'v$ contains at most $2\|K\|_\infty$ b.

We have shown that all words in $\mathcal{L}(K)$ contain at most $2\|K\|_\infty$ b, contradicting $\mathcal{L}(K) = L$ as the number of b of words of L is unbounded. ◀

Proof of Proposition 7.6

Consider the following language over $A = \{a, b, c\}$:

$L = L_1 \cup L_2$ with $L_1 = \left\{wc\widetilde{w} \;\middle|\; w \in \{a,b\}^*\right\}$ and $L_1 = \left\{wcc\widetilde{w} \;\middle|\; w \in \{a,b\}^*\right\}$.

We are going to prove that it is in LEK but not in BAK.

▶ **Lemma F.3.** *Let $K = (\{\texttt{aa}\blacktriangleleft, \texttt{bb}\blacktriangleleft\}, \{\texttt{c}, \texttt{cc}\})$. Then $\mathcal{L}(K) = L$.*

Proof. We use the notations $t_a = \texttt{aa}\blacktriangleleft$ and $t_b = \texttt{bb}\blacktriangleleft$. A straightforward induction shows that $\mathcal{L}((t_a, t_b))$ is the language of even palindromes over $\{a, b\}$ and that the configurations reached by application of those keys are of the form $\langle w|\widetilde{w}\rangle$.

We then easily infer that the language of K is L. ◀

We now show that $L \notin$ BAK. The intuition is as follows:

Suppose we have a keyboard K of BAK recognizing L. As we want to recognize palindromes, we need to stay close to the center in order to always modify both halves of the word.

If the word is large enough, this means that we have to get far from the edges. In particular, there is a key t writing an a far from the edges (even two a, as we have to stay in the language).

We study the behaviour of this key on b^ncb^n and b^nccb^n with large n. The cursor being far from the edges, t behaves the same way in both cases. In particular, we make the following remarks:

- The maximal distance d between two a will be the same in both words. Thus the resulting words have either both two c or both one c in the center.
- In both cases the key adds a number of letters δ, and thus if b^ncb^n is turned into a word of even length, then b^nccb^n is turned into a word of odd length, and vice-versa.
  As a result, they have different numbers of c in the center.

As those facts are contradictory, we conclude that there cannot exist such a keyboard.

Now for the formal proof, we start by showing that the maximal distance between two a is the same when we apply the same key to two configurations without a effectively.

▶ **Definition F.4.** *Let $\langle u|v\rangle$ be a configuration, let $a \in A$. We define $\mathrm{d}_a(\langle u|v\rangle)$ as*

$$\begin{cases} +\infty \text{ if } \langle u|v\rangle \text{ contains zero or one } a. \\ \max(\{|w| \mid awa \in \mathrm{Fact}(uv)\}) \text{ otherwise.} \end{cases}$$

▶ **Lemma F.5.** *Let $t \in T$ be a key and $u, v, u', v' \in A^*$ words with no a and of length at least $|t|$. Then*

$$\mathrm{d}_a\big(\langle u|v\rangle \cdot t\big) = \mathrm{d}_a\big(\langle u'|v'\rangle \cdot t\big).$$

Proof. As u, v, u', v' are all of length at least $|t|$, t acts effectively on both configurations (by Lemma 4.2). By Proposition 5.11, t writes an a in position i from $\langle u'|v'\rangle$ if and only if t writes an a in position i from $\langle u|v\rangle$.

As $\langle u|v\rangle$ does not contain an a, by Proposition 5.4, $(\langle u|v\rangle \cdot t)_i = a$ if and only if t writes an a in position i from $\langle u|v\rangle$. Similarly, $(\langle u'|v'\rangle \cdot t)_i = a$ if and only if t writes an a in position i from $\langle u'|v'\rangle$.

Thus $(\langle u|v\rangle \cdot t)_i = a$ if and only if $(\langle u'|v'\rangle \cdot t)_i = a$, proving the result. ◀

We can now prove our result. Suppose we have K an BAK keyboard recognizing L.

We first show that in an execution of K the cursor must stay close to the center of the word.

▶ **Lemma F.6.** *Let $\langle u|v\rangle$ a reachable configuration of K and $t \in K$ such that, by setting $\langle u'|v'\rangle = \langle u|v\rangle \cdot t$, we have $uv \neq u'v'$. Then $\big||u| - |v|\big| \leq 6\|K\|_\infty$ and $\big||u'| - |v'|\big| \leq 8\|K\|_\infty$.*

Proof. Suppose $\big||u| - |v|\big| > 6\|K\|_\infty \geq 6|t|$. Then, either $|u| - |v| > 6|t|$, or $|v| - |u| > 6|t|$. We assume to be in the first case, the other one being similar. We thus have

$$|u| = \frac{|u|}{2} + \frac{|u|}{2} > \frac{|u|}{2} + \frac{|v| + 6|t|}{2}$$

Hence $|u| \geq \frac{|uv|}{2} + 3|t|$.

As $uv \in L$, uv is of the form either $wc\widetilde{w}$ or $wcc\widetilde{w}$ for some $w \in \{a, b\}^*$.

By Lemma 4.1, there exists a common prefix u_p to u and u' such that

$$|u_p| \geq |u| - |t| \geq \frac{|uv|}{2} + 2|t| \geq \frac{|uv|}{2} + 2,$$

The last inequality holds as t is non-empty (as $uv' \neq uv$).

As a consequence u_p is of the form either wcy (and $uv = wc\widetilde{w}$) or $wccy$ (and $uv = wcc\widetilde{w}$) with $y \in \{a, b\}^+$. As $u'v'$ is in L, in both cases $uv = u'v'$, contradicting our hypothesis.

As a conclusion, we have $\big||u| - |v|\big| \leq 6\|K\|_\infty$.

By Lemma 4.3, we have $\big||u| - |u'|\big| \leq |t|$ and $\big||v| - |v'|\big| \leq |t|$. Consequently,

$$\Big||u'| - |v'|\Big| \leq \Big||u| - |v|\Big| + \Big||u| - |u'|\Big| + \Big||v| - |v'|\Big| \leq 6\|K\|_\infty + 2|t| \leq 8\|K\|_\infty.$$

◀

▶ **Lemma F.7.** *If $K \in$ BAK recognizes L, then K contains a key ensuring an a far from the edges.*

Proof. There exists $t_1 \cdots t_n \in T^*$ such that $\langle\varepsilon|\varepsilon\rangle \cdot t_1 \cdots t_n = \langle u|v\rangle$ with $uv = a^{6\|K\|_\infty} cca^{6\|K\|_\infty}$. Let i be the smallest index such that the number of a in $\langle u_i|v_i\rangle = \langle\varepsilon|\varepsilon\rangle \cdot t_1 \cdots t_i$ is maximized. We set

$$\langle u_{i-1}|v_{i-1}\rangle = \langle\varepsilon|\varepsilon\rangle \cdot t_1 \cdots t_{i-1}.$$

In particular $\langle u_i|v_i\rangle$ contains at least $12\|K\|_\infty$ a. By Lemma 4.3, we have

$$|u_{i-1}v_{i-1}| \geq |u_iv_i| - |t_i| \geq 12\|K\|_\infty - \|K\|_\infty \geq 11\|K\|_\infty. \tag{1}$$

Further, by Lemma F.6, $\big||u_{i-1}| - |v_{i-1}|\big| \leq 6\|K\|_\infty$, thus by triangle inequality,

$$|u_{i-1}v_{i-1}| \leq |u_{i-1}| + \big(|u_{i-1}| + 6\|K\|_\infty\big).$$

Suppose, $|u_{i-1}| < \|K\|_\infty$, we then have $|u_{i-1}v_{i-1}| \leq 8\|K\|_\infty$, contradicting 1. Thus $|u_{i-1}| \geq \|K\|_\infty$.

We prove similarly that $|v_{i-1}| \geq \|K\|_\infty$. As a consequence, u_{i-1} and v_{i-1} are of length at least $\|K\|_\infty$, and by Proposition 5.14 t_i ensures an a far from the edges, as by minimality of i, $\langle u_{i-1}|v_{i-1}\rangle$ contains less a than $\langle u_i|v_i\rangle$. ◀

▶ **Lemma F.8.** *The language L is not recognized by an* BAK *keyboard.*

Proof. Suppose there exists K an BAK keyboard recognizing L. Then, there exist τ et τ' of minimal length leading respectively to configurations $\langle u|v\rangle$ and $\langle u'|v'\rangle$ with $uv = b^{5\|K\|_\infty}cb^{5\|K\|_\infty}$ and $u'v' = b^{5\|K\|_\infty}ccb^{5\|K\|_\infty}$.

Let $\langle x|y\rangle$ be the penultimate configuration in the execution of τ. By minimality of τ, $xy \neq uv$ and by Lemma F.6 we then have $\big||u| - |v|\big| \leq 8\|K\|_\infty$.

We show similarly that $\big||u'| - |v'|\big| \leq 8\|K\|_\infty$.

In particular, as $|uv|, |u'v'| > 10\|K\|_\infty$ we have $|u|, |v|, |u'|, |v'| \geq \|K\|_\infty$.

By Lemma F.7, there exists $t \in T$ such that t ensures an a far from the edges. We then set

$$\langle u_a|v_a\rangle = \langle\varepsilon|\varepsilon\rangle \cdot \tau t \quad \text{and} \quad \langle u'_a|v'_a\rangle = \langle\varepsilon|\varepsilon\rangle \cdot \tau' t.$$

By Lemma F.5, we have $\mathrm{d}_a(u_av_a) = \mathrm{d}_a(u'_av'_a)$. We set

$$d = \mathrm{d}_a(u_av_a) \quad \text{and} \quad \delta = \sum_{x\in A} |t|_x - |t|_{\leftarrow}.$$

We have $|u| \geq |t|$, $|v| \geq |t|$ and t ensures an a far from the edges, hence u_av_a contains at least an a and thus at least two as $u_av_a \in L$. We therefore have $d \in \mathbb{N}$.

Further, as u_av_a and $u'_av'_a$ are in L, then if d is even they must both be in L_2 and otherwise both in L_1.

However, if δ is even then by Lemma 4.4,

$$\begin{cases} |u_av_a| = |uv| + \delta = 10\|K\|_\infty + 2 + \delta & \text{is even} \\ |u'_av'_a| = |u'v'| + \delta = 10\|K\|_\infty + 1 + \delta & \text{is odd} \end{cases}$$

thus $u_av_a \in L_2$ et $u'_av'_a \in L_1$, contradicting the fact that they must be in the same L_j.

Similarly, if δ is odd then $u_av_a \in L_1$ and $u'_av'_a \in L_2$, again contradicting the fact that they must be in the same L_j.

We obtained a contradiction. As a result, L is not recognized by an BAK keyboard. ◀

▶ **Proposition 7.6** (LEK $\not\subseteq$ BAK)**.** LEK *is not included in* BAK.

Proof. We have shown in Lemma F.3 that L is in LEK and in Lemma F.8 that it is not in BAK, hence the result. ◀

G Complexity

Proof of Proposition 8.1

▶ **Proposition 8.1.** *The membership problem on* MK *and* EK *is in PTIME. The universality problem is in PTIME on* MK *and PSPACE on* EK*.*

Proof. The complexities of the membership problem on MK and the universality problem on EK arise directly from the complexities of the membership and universality problems on rational expressions.

Let $K \subset A^*$ be a minimal keyboard. Since $\mathcal{L}(K) = K^+$, K is universal if and only if $K^+ = A^*$. Let us show that $K^+ = A^*$ if and only if $A \cup \{\varepsilon\} \subseteq K$.

- If $A \cup \{\varepsilon\} \subseteq K$, then $A^* = (A \cup \{\varepsilon\})^+ \subset K^+$.
- If $A \cup \{\varepsilon\} \not\subseteq K$, then either $\varepsilon \notin K$ (and $\varepsilon \notin K^+$) or there exists $a \in A$ such that $a \notin K$ (and $a \notin K^+$). In both cases, $A^* \neq K^+$.

To check the universality of a minimal keyboard, we just have to check (in polynomial time) if $A \cup \{\varepsilon\} \subset K$. ◀

Proof of Proposition 8.4

▶ **Proposition 8.4.** *The universality problem for* BK *keyboards is in* coNP.

Proof. We prove that if an BK keyboard K is not universal then there is a word of length at most $\|K\|_\infty + 1$ it does not recognize.

Let K be a keyboard of BK, suppose there exists $w \in A^*$ not recognized by K. We take w of minimal length. If $|w| \leq \|K\|_\infty + 1$ then the property holds.

If $|w| > \|K\|_\infty + 1$ then there exist $a \in A, v \in A^+$ such that $av = w$. As we assumed w to be of minimal length, v is recognized by K. By Lemma C.6, there exist $t \in K, \tau \in K^*$ such that $(\varepsilon \cdot t) \xrightarrow{\tau}_{\mathrm{e}} v$. Let $u = \varepsilon \cdot t$, $k \in \mathbb{N}$ be such that $t = \leftarrow^k u$.

As $\left|a^{k+1}\right| = k + 1 \leq \|K\|_\infty + 1 < |w|$, a^{k+1} is recognized by K. Let τ' be such that $\varepsilon \cdot \tau' = a^{k+1}$, then we have $\varepsilon \xrightarrow{\tau'} a^{k+1} \xrightarrow{t} au \xrightarrow{\tau}_{\mathrm{e}} av = w$.

This contradicts the fact that w is not recognized by K. The property is proven. ◀

H Closure properties

Proof of Proposition 9.1

▶ **Proposition 9.1** (Mirror). MK, AK *and* EAK *are stable by mirror.* EK, BK, BEK *and* BLK *are not stable by mirror.*

▶ **Lemma H.1.** MK, EAK *and* AK *are stable under mirror.*

Proof. For MK we simply observe that given K an MK keyboard, we have $\widetilde{K}^* = \widetilde{K^*}$. As a consequence, the MK keyboard $\widetilde{K}$ recognizes the mirror of $\mathcal{L}(K)$.

For EAK and AK we define $\overline{\cdot}$ the word morphism generated by

$$\overline{\blacktriangleright} = \blacktriangleleft \qquad \overline{\blacktriangleleft} = \blacktriangleright \qquad \overline{a} = a\blacktriangleleft \text{ if } a \in A$$

If $\langle u|v\rangle$ is a configuration, we write $\widetilde{\langle u|v\rangle} = \langle \widetilde{v}|\widetilde{u}\rangle$. We can show by a straightforward induction on n that for all sequence of elementary operations $\sigma_1 \dots \sigma_n$,

$$c_0 \xrightarrow{\sigma_1 \dots \sigma_n} c_n \iff \widetilde{c_0} \xrightarrow{\overline{\sigma_1} \dots \overline{\sigma_n}} \widetilde{c_n}.$$

Let L be a language of EAK and $K = (T, F)$ a keyboard recognizing it. Let $\overline{K} = (\overline{T}, \overline{F})$ where the morphism is applied on all keys. For all $w \in A^*$, the execution $\langle \varepsilon|\varepsilon\rangle \xrightarrow{t_1 \dots t_n} w$ can be reversed into an execution $\langle \varepsilon|\varepsilon\rangle \xrightarrow{\overline{t_1} \dots \overline{t_n}} \widetilde{w}$ of $\overline{K}$ and vice versa. As a result, $w \in L$ if and only if $\widetilde{w} \in \mathcal{L}(\overline{K})$, where $\mathcal{L}(\overline{K}) = \widetilde{L}$. Further, $\overline{K}$ is an EAK keyboard, and is automatic if and only if K is automatic as well, showing the result for both AK and EAK. ◀

▶ **Lemma H.2.** BEK, BK *and* EK *are not stable under mirror.*

Proof. Let $L = ab^*$, we have $\widetilde{L} = b^*a$. We have $b^*a \in$ EK (with the keyboard $(\{\mathtt{b}\}, \{\mathtt{a}\})$) and $b^*a \in$ BK, with the keyboard $\{\leftarrow\mathtt{a}, \leftarrow\mathtt{ba}\}$.

We now show that $\widetilde{L} \notin$ BEK. Suppose there exists $K = (T, F)$ an BEK keyboard recognizing $\widetilde{L}$. Since $w = ab^{\|K\|_\infty} \in \widetilde{L}$, there exists $\tau \in T^*, t_f \in F$ yielding w. In particular, t_f only wrote some b and is thus equivalent to $\leftarrow^k b^n$ for some $k, n \in \mathbb{N}$.

By applying t_f from ε, we obtain b^n, which is impossible as $b^n \notin \widetilde{L}$. As a result, $\widetilde{L} \notin$ BEK.

We then have L in EK and BK, and $\widetilde{L} \notin$ BEK, proving the lemma. ◀

▶ **Lemma H.3.** BLK *is not stable under mirror.*

Proof. Let $L = a^*(b + b^2)$. We showed in the proof of Proposition 7.2 that $L \notin \mathsf{BLK}$. We then simply observe that $\widetilde{L} = (b + b^2)a^*$ is recognized by the following BLK keyboard:

$$K = \left\{\leftarrow^2 \mathtt{a} \blacktriangleleft \mathtt{b}, \leftarrow^2 \mathtt{a} \blacktriangleleft \mathtt{bb}, \leftarrow^2 \mathtt{b}, \leftarrow^2 \mathtt{bb}\right\}.$$

◀

Proof of Proposition 9.2

We give, for each keyboard language class, two languages of $\mathcal{C}$ whose intersection is not in $\mathcal{C}$.

▶ **Lemma H.4.** *The language* $L = \{bw \mid w \in (a + b)^*, |w| \text{ odd}, aa \notin \mathrm{Fact}(w)\}$ *is not in* EK.

Proof. Suppose there exists $K = (T, F)$ an EK keyboard recognizing L. Then $L = T^*F$. As all words in L are of even length, so are all keys in F.

Further, $(ba)^{\|K\|_\infty}$ is in L, thus accepted by K, and thus of the form $t_1 \cdots t_n f$ with $t_i \in T$ for all i, $f \in F$ and $n > 0$ (as its length is greater than $\|K\|_\infty$). As $|f|$ is even, t_n ends with an a (we can assume T does not contain any empty keys).

Similarly, $bb(ab)^{\|K\|_\infty}$ is in L, thus of the form $t'_1 \cdots t'_m f'$ with $t_i \in T$ for all i, $f \in F$ and $m > 0$. As $|f'|$ is an even suffix of $(ab)^{\|K\|_\infty}$, f' starts with an a. As a result, $t_n f'$ contains aa as a factor, and is thus not in L, yielding a contradiction. ◀

▶ **Lemma H.5.** MK *and* EK *are not stable by intersection.*

Proof. The language L from Lemma H.4 is the intersection of $(ab + ba + bb)^*$ and $(ba + b)^*$, which are both in MK. By Lemma H.4, their intersection is not in EK. ◀

▶ **Lemma H.6.** BK *and* BEK *are not stable by intersection.*

Proof. We consider again the language $L_{\Diamond\blacklozenge}$ described in the proof of Proposition 7.4. Recall that $L_{\Diamond\blacklozenge}$ is defined as the language of $\{t_a, t_b\}$ with $t_a = \leftarrow a \Diamond \blacklozenge$ and $t_b = \leftarrow\leftarrow b \Diamond \blacklozenge\blacklozenge$.

We also define L' the language $(a + b + \Diamond)^+\blacklozenge^2$, recognized by the BK keyboard

$$\left\{\leftarrow^2 a\blacklozenge^2, \leftarrow^2 b\blacklozenge^2, \leftarrow^2 \Diamond\blacklozenge^2\right\}.$$

By Lemma F.1, the intersection of $L_{\Diamond\blacklozenge}$ and L' is the language $[(a\Diamond)^*a + \varepsilon](b\Diamond)^+\blacklozenge^2$. We now prove that this language is not in BEK.

Suppose there is a keyboard $K = (T, F)$ of BEK recognizing $L = L_{\Diamond\blacklozenge} \cap L'$. We assume all keys of K to be in normal form (see Lemma 6.4). As L contains $(a\Diamond)^{2\|K\|_\infty+1} b \Diamond \blacklozenge^2$, there is a sequence of keys $\tau_a \in T^*$ writing $(a\Diamond)^{\|K\|_\infty+1} w$ for some w.

Similarly as L contains $(b\Diamond)^{(|\tau|+1)\|K\|_\infty+1}\blacklozenge^2$, there is a sequence of keys $\tau_b \in T^*$ writing $(b\Diamond)^{|\tau|\|K\|_\infty+1} w'$ for some w'.

By applying $\tau'\tau$ we get, by Lemma 4.1, a word starting with b and containing $\|K\|_\infty + 1$ a. Thus by then applying some $f \in F$, again by *Lemma* 4.1, we get a word accepted by K starting with b and containing an a, which is impossible as there is no such word in L. ◀

▶ **Lemma H.7.** *No class containing* LK *is stable by intersection.*

Proof. We consider the following languages:

- $(a + b)^*c^*$, recognized by $\{a, b, c\blacktriangleleft\}$
- $a^*(b + c)^*$, recognized by $\{a, b\blacktriangleleft, c\blacktriangleleft\}$
- $\{w \in (a + b + c)^* \mid |w|_a = |w|_b\}$, recognized by $\{ab, ba, c, \blacktriangleleft\}$

- $\{w \in (a+b+c)^* \mid |w|_b = |w|_c\}$, recognized by $\{bc, cb, a, \blacktriangleleft\}$

Those languages are all in LK. Their intersection is $L = \{a^n b^n c^n \mid n \in \mathbb{N}\}$. We now show that L is not in BEAK.

Suppose there exists $K = (T, F)$ in BEAK recognizing L. Let $\tau f \in T^*F$ be the shortest execution of K accepting $a^{6\|K\|_\infty+1} b^{6\|K\|_\infty+1} c^{6\|K\|_\infty+1}$.

We have $\langle \varepsilon|\varepsilon\rangle \cdot \tau f = \langle u|v\rangle$ with $uv = a^{6\|K\|_\infty+1} b^{6\|K\|_\infty+1} c^{6\|K\|_\infty+1}$. Clearly as $|uv| > \|K\|_\infty$, τ is non-empty, thus there exist $\tau' \in T^*$ and $t \in T$ such that $\tau = \tau' t$.

We have either $a^{6\|K\|_\infty} b^{3\|K\|_\infty+1}$ as a prefix of u or $b^{3\|K\|_\infty+1} c^{6\|K\|_\infty+1}$ as a suffix of v. We assume that we are in the first case, as the other one is similar. Let $\langle u'|v'\rangle = \langle \varepsilon|\varepsilon\rangle \cdot \tau'$ and $\langle x|y\rangle = \langle \varepsilon|\varepsilon\rangle \cdot \tau' f$

By Lemma 4.1, $a^{6\|K\|_\infty+1} b^{\|K\|_\infty+1}$ is a prefix of u', and again by Lemma 4.1, $a^{6\|K\|_\infty+1} b$ is a prefix of x. As xy is accepted by K, $xy \in L$, hence $xy = a^{6\|K\|_\infty+1} b^{6\|K\|_\infty+1} c^{6\|K\|_\infty+1} = uv$. This contradicts the minimality of τf.

As a result, L is not in BEAK. ◀

▶ **Proposition 9.2** (Intersection). *None of the keyboard language classes are stable by intersection.*

Proof. We proved in Lemma H.5 that MK and EK are not stable by intersection, then in Lemma H.6 that neither are BK and BEK. We finally proved it in Lemma H.7 for the rest of the classes. ◀

Proof of Proposition 9.3

▶ **Proposition 9.3** (Union). *None of the keyboard language classes are stable by union.*

We decompose the proof into several parts.

To start with we consider the languages a^* and b^*, both in MK, and prove that their union is neither in BLEK nor in EAK.

▶ **Lemma H.8.** *The language $L = a^* + b^*$ is not in* BLEK.

Proof. Suppose there exists a keyboard $K = (T, F)$ of BLEK recognizing L. Then there exists $\tau_a \in T^*$, $f_a \in F$ such that $\langle \varepsilon|\varepsilon\rangle \cdot \tau_a f_a = \langle u_a|v_a\rangle$ with $u_a v_a = a$.

There also exists $\tau_b \in T^*$, $f_b \in F$ such that $\langle \varepsilon|\varepsilon\rangle \cdot \tau_b f_b = \langle u_b|v_b\rangle$ with $u_b v_b = b^{\|K\|_\infty(|\tau_a|+2)+1}$.

By Lemma 4.1 applying τ_b to $\langle \varepsilon|\varepsilon\rangle$ yields a configuration with at least $1 + \|K\|_\infty(|\tau_a|+1)$ b. We apply $\tau_b \tau_a f_a$ to $\langle \varepsilon|\varepsilon\rangle$, by Theorem 6.10 the resulting configuration contains an a, and as $\tau_a f_a$ can only erase at most $|\tau_a| + 1$ it contains a b. This is impossible as the resulting word should be in L. ◀

▶ **Lemma H.9.** *The language $L = a^* + b^*$ is not in* EAK.

Proof. Suppose there exists $K = (T, F)$ a keyboard of EAK recognizing L. As $a^{\|K\|_\infty+1}$ and $b^{\|K\|_\infty+1}$ are both in L, there exist $t_a, t_b \in T$ such that t_a writes an a and t_b a b (and those letters are never erased as we do not have $\leftarrow$). Let $f \in F$, $t_a t_b f$ writes a word containing both a and b, thus not in L. ◀

Proof that BAK and BEAK are not stable under union.

We define the language $L = L_a \cup L_b$ with $L_a = \{a^n c a^n \mid n \in \mathbb{N}\}$ and $L_b = \{b^n c b^n \mid n \in \mathbb{N}\}$ and prove that it is not in BEAK. Note that L_a and L_b are in BAK as they are recognized by $\{\leftarrow c, \leftarrow aca\blacktriangleleft\}$ and $\{\leftarrow c, \leftarrow bcb\blacktriangleleft\}$, respectively.

Suppose we have a keyboard $K = (T, F)$ of BEAK recognizing L.

We start by proving that there is an execution of K leading to a configuration with the cursor far from the edges.

▶ **Lemma H.10.** *Let $M \in \mathbb{N}$, there is a sequence of keys $\tau_a \in T^*$ such that $\langle\varepsilon|\varepsilon\rangle \cdot \tau_a = \langle u_a|v_a\rangle$ with $|u_a| > M$ and u_a starts with $\|K\|_\infty + 1$ a.*

There is also a sequence of keys $\tau_b \in T^*$ such that $\langle\varepsilon|\varepsilon\rangle \cdot \tau_b = \langle u_b|v_b\rangle$ with $|u_b| > M$ and u_b starts with $\|K\|_\infty + 1$ b.

Proof. As $w = a^{M+4\|K\|_\infty} c a^{M+4\|K\|_\infty}$ is in the language, there exists $\tau \in T^*$ and $f \in F$ such that $\langle\varepsilon|\varepsilon\rangle \cdot \tau f = \langle x|y\rangle$ with $w = xy$. Let $\langle u|v\rangle = \langle\varepsilon|\varepsilon\rangle \cdot \tau$. We can assume τ to be of minimal length. Clearly τ is not empty, thus let $\tau' \in T^*, t \in T$ be such that $\tau = \tau' t$. Let $\langle u'|v'\rangle = \langle\varepsilon|\varepsilon\rangle \cdot \tau'$.

Suppose $|u| \leq M$, then $|x| \leq M + \|K\|_\infty$ by Lemma 4.3, hence $|y| \geq M + 7\|K\|_\infty + 1$. Then $a^{3\|K\|_\infty} c a^{M+4\|K\|_\infty}$ is a suffix of y, and by Lemma 4.1, $a^{2\|K\|_\infty} c a^{M+4\|K\|_\infty}$ is a suffix of v'. Again by Lemma 4.1, $c a^{M+4\|K\|_\infty}$ is a suffix of the word given by $\langle\varepsilon|\varepsilon\rangle \cdot \tau' f$. As this word is in L, it has to be w, contradicting the minimality hypothesis on τ.

As a result, $|u| \geq 3(\|K\|_\infty + 1)^2$. Another application of Lemma 4.1 gives that u starts with $\|K\|_\infty + 1$ a. We thus set $u_a = u$, $v_a = v$ and $\tau_a = \tau$.

A similar proof shows the second part of the lemma. ◀

The next lemma states that the keys of K erase more than they write, and thus the number of letters of the configuration can only increase when the cursor is close to the left end of the word.

▶ **Lemma H.11.** *Every key $t \in T$ contains at least as many $\leftarrow$ as letters, i.e., $|t|_\leftarrow \geq |t|_a + |t|_b + |t|_c$.*

Proof. We consider τ_a, τ_b, u_a, u_b and v_a, v_b as in Lemma H.10, with $M = 3(\|K\|_\infty + 4)^2$. Suppose there exists $t \in T$ such that $|t|_\leftarrow < |t|_a + |t|_b + |t|_c$. Then by applying t $3(\|K\|_\infty + 4)$ times from $\langle u_a|v_a\rangle$ we write either $\|K\|_\infty + 2$ a, $\|K\|_\infty + 2$ b or $\|K\|_\infty + 2$ c.

If applying t $3(\|K\|_\infty + 4)$ times effectively writes $\|K\|_\infty + 2$ b or $\|K\|_\infty + 2$ c, then from $\langle u_a|v_a\rangle$ we get a configuration starting with $\|K\|_\infty + 1$ a (by Lemma 4.1) and containing at least $\|K\|_\infty + 2$ b or c, which after applying some final key $f \in F$ still contains an a and two b or c. This is impossible as the resulting word should be in L.

Similarly, if applying t $3(\|K\|_\infty + 4)$ times effectively writes $\|K\|_\infty + 2$ a, then from $\langle u_b|v_b\rangle$ we get a configuration starting with $\|K\|_\infty + 1$ b (by Lemma 4.1) and containing at least $\|K\|_\infty + 2$ a, which after applying some final key $f \in F$ still contains a b and two a. This is impossible as the resulting word should be in L.

Thus there is no such key. ◀

We now show that the number of letters in the configuration cannot increase too much while staying far from the center.

▶ **Lemma H.12.** *Let $\tau_0 \in T^*, t_1, \ldots, t_n \in T$. For all $0 \leq i \leq n$ we define $\langle u_i|v_i\rangle = \langle\varepsilon|\varepsilon\rangle \cdot \tau_0 t_1 \cdots t_i$.*

If $|u_n| + |v_n| \geq |u_0| + |v_0| + 4\|K\|_\infty + 1$ then there exists $0 \leq i \leq n$ such that $||u_i| - |v_i|| \leq 8\|K\|_\infty + 2$.

Proof. Suppose $|u_n| + |v_n| \geq |u_0| + |v_0| + 4\|K\|_\infty + 1$ and for all $0 \leq i \leq n$, $||u_i| - |v_i|| > 8\|K\|_\infty + 2$.

Then as $||u_{i+1}| - |u_i||$ and $||v_{i+1}| - |v_i||$ are both at most $\|K\|_\infty$, and thus $|u_i| - |v_i|$ always has the same sign. We assume it to be positive, the other case is similar.

Let $f \in F$, for all i let $\left\langle u_i^f \middle| v_i^f \right\rangle = \langle u_i | v_i \rangle \cdot f$. We have by Lemma 4.3 $\left|u_i^f\right| - \left|v_i^f\right| \geq 6\|K\|_\infty + 2$ As $u_i^f v_i^f \in L$, we thus have u_i^f of the form $a^m c a^p$ or $b^m c b^p$, with $p \geq 3\|K\|_\infty + 1$. We assume the first case as the other one is similar.

As a consequence, by Lemma 4.1, $a^m c a^{2\|K\|_\infty+1}$ is a prefix of u_i, and $a^m c a^{\|K\|_\infty+1}$ a prefix of u_{i+1}, and $a^m c a$ a prefix of u_{i+1}^f. Therefore we must have $u_{i+1}^f v_{i+1}^f = a^m c a^m$.

As a result, we have $u_n^f v_n^f = u_0^f v_0^f$, and therefore by Lemma 4.3, $||u_n v_n| - |u_0 v_0|| \leq 4\|K\|_\infty$, contradicting the hypothesis.

◀

We showed that the number of letters could only increase with the cursor close to the left end of the word, and that we can only increase the number of a by a bounded number while staying far from the center.

Thus in order to write a very long word $a^M c a^M$, we need to go back and forth between the left edge and the center. We isolate a moment in the execution at which the cursor is far from the center and the left edge, goes to the left edge, and goes back while increasing the number of letters.

We end up with two configurations with different numbers of letters. As we are far from the edge in both, a final key f has the same behaviour (adds and erases as many letters) in both. Thus the word obtained afterwards cannot be equal. However we are far from the center and we have a same suffix ca^M, thus the two words have to be equal, hence a contradiction.

▶ **Lemma H.13.** *No keyboard of* BEAK *recognizes* L.

Proof. Let $M = 30\|K\|_\infty$. Let $\tau = t_1 \cdots t_n t_f \in T^*F$ be an execution of K recognizing $a^M c a^M$. For all $0 \leq i \leq n$ we define $\langle u_i | v_i \rangle = \langle \varepsilon | \varepsilon \rangle \cdot t_1 \cdots t_i$. Let $f \in F$, we define for all $0 \leq i \leq n$ $\left\langle u_i^f \middle| v_i^f \right\rangle = \langle u_i | v_i \rangle \cdot f$.

Let i be the minimal index such that $|u_j v_j| \geq 10\|K\|_\infty$ for all $j \geq i$. In particular we have $|u_i v_i| \leq 11\|K\|_\infty$.

By Lemma H.11, $|u_{j+1} v_{j+1}| - |u_j v_j|$ is negative or null whenever $|u_j| \geq \|K\|_\infty$ (as t_j contains at least as many $\leftarrow$ as letters and every $\leftarrow$ is applied effectively).

Let I be the set of indices $j > i$ such that $|u_j| > \|K\|_\infty$ and $|v_j| - |u_j| > 2\|K\|_\infty$.

The size of the configuration increases by at least $9\|K\|_\infty$ between the applications of t_i and t_f. By Lemma H.11, the size of the configuration can only increase when the left part of the configuration has length at most $\|K\|_\infty$.

As a consequence, by Lemma H.12, there exist $j_1 < j_2$ such that $|v_{j_1}| - |u_{j_1}|, |v_{j_2}| - |u_{j_2}| \leq 8\|K\|_\infty + 2$ and $|u_{j_1} v_{j_1}| < |u_{j_2} v_{j_2}|$.

We take j_1 maximal and j_2 minimal for this property. We obtain that for all $j_1 < j < j_3$, $|v_{j_3}| - |u_{j_3}| > 8\|K\|_\infty + 2 > 2\|K\|_\infty$.

Note that as $|u_j v_j| > 10\|K\|_\infty$ for all $j > i$, we have that there exists $i_1, i_2 \in I$ such that $j_1 < i_1 < i_2 < j_3$. We choose i_1 minimal and i_2 maximal for that property. We cannot have any $j_1 < i' < i_1$ such that $|u_{i'}| \leq \|K\|_\infty$ as otherwise we would necessarily have some index of I between j_1 and i', contradicting the minimality of i_1. Similarly, we cannot have $i_2 < i' < j_2$ such that $|u_{i'}| \leq \|K\|_\infty$.

We therefore have $|u_{i_1}v_{i_1}| \leq |u_{j_1}v_{j_1}| < |u_{j_2}v_{j_2}| \leq |u_{i_2}v_{i_2}|$ (the second inequality holds by construction of j_1 and j_2).

We obtain that:

- $\|K\|_\infty \leq |u_{i_1}|, |u_{i_2}|$
- $|u_{i_1}v_{i_1}| \neq |u_{i_2}v_{i_2}|$
- For all $i_1 < j < i_2$, $|u_j| < |v_j| - 2\|K\|_\infty$ (by definition of j_1 and j_2.

Let $f \in F$, by an argument similar to the proof of Lemma H.12, we have that $u_{i_1}^f v_{i_1}^f = u_{i_2}^f v_{i_2}^f$ as we stay away from the center and thus the part of the word of the form ca^m is unchanged.

However, in both cases we applied the final key f far from the edges, thus by applying Lemma 4.4 we get $|u_{i_1}v_{i_1}| = |u_{i_2}v_{i_2}|$, yielding a contradiction.

As a result there is no BEAK keyboard recognizing L. ◀

Proof of Proposition 9.4

▶ **Proposition 9.4** (Intersection emptiness problem). *The following problem is undecidable:*
Input: *K_1, K_2 two* LK *keyboards.*
Output: *Is $\mathcal{L}(K_1) \cap \mathcal{L}(K_2)$ empty?*

Proof. We reduce the Post Correspondence Problem. Let $(u_i, v_i)_{i\in[\![1,n]\!]}$ be a PCP instance. For all $i \in [\![1, n]\!]$, let $u_i^\blacklozenge$ and $v_i^\blacklozenge$ be u_i and v_i where we added a $\blacklozenge$ at the right of every letter, i.e., if $u_i = a_1a_2\cdots a_n$ then $u_i^\blacklozenge = a_1\blacklozenge a_2\blacklozenge\cdots a_n\blacklozenge$.

We set for all $i \in [\![1, n]\!]$ $t_i = u_i^\blacklozenge \widetilde{v_i^\blacklozenge} \blacktriangleleft^{|v_i|}$. Let

$$K_{pal} = \{aa\blacktriangleleft \mid a \in A \cup \{\blacklozenge\}\} \cup \{\varepsilon\}$$

K_{pal} recognizes the language of even palindromes over $A \cup \{\blacklozenge\}$.
Now let

$$K = \{t_i \mid i \in [\![1, n]\!]\}.$$

We show that $\mathcal{L}(K) \cap \mathcal{L}(K_{pal}) \neq \emptyset$ if and only if $(u_i, v_i)_{i\in[\![1,n]\!]} \in$ PCP. A straightforward induction on $k \in \mathbb{N}$ shows that for all $i_1, \ldots, i_k \in [\![1, n]\!]$,

$$\langle\varepsilon|\varepsilon\rangle \cdot (t_{i_1} \ldots t_{i_k}) = \left\langle u_{i_1}^\blacklozenge \cdots u_{i_k}^\blacklozenge \middle| \widetilde{v_{i_k}^\blacklozenge} \cdots \widetilde{v_{i_1}^\blacklozenge} \right\rangle.$$

Note that if $k > 0$, $u_{i_1}^\blacklozenge \cdots u_{i_k}^\blacklozenge \widetilde{v_{i_k}^\blacklozenge} \cdots \widetilde{v_{i_1}^\blacklozenge}$ has exactly one $\blacklozenge\blacklozenge$ factor, with the last letter of $u_{i_k}^\blacklozenge$ and the first one of $\widetilde{v_{i_k}^\blacklozenge}$. As a consequence, it is an even palindrome if and only if $u_{i_1}^\blacklozenge \cdots u_{i_k}^\blacklozenge = \widetilde{v_{i_k}^\blacklozenge \cdots v_{i_1}^\blacklozenge} = v_{i_1}^\blacklozenge \cdots v_{i_k}^\blacklozenge$.

All that is left to show is that $u_{i_1}^\blacklozenge \cdots u_{i_k}^\blacklozenge = v_{i_1}^\blacklozenge \cdots v_{i_k}^\blacklozenge$ if and only if $u_{i_1} \cdots u_{i_k} = v_{i_1} \cdots v_{i_k}$. The left to right direction is shown by projecting the words on A, the right to left one by observing that $u_{i_1}^\blacklozenge \cdots u_{i_k}^\blacklozenge$ and $v_{i_1}^\blacklozenge \cdots v_{i_k}^\blacklozenge$ are the images of $u_{i_1} \cdots u_{i_k}$ and $v_{i_1} \cdots v_{i_k}$ under the morphism associating $a\blacklozenge$ to each letter $a \in A$. As a result, $\mathcal{L}(K)$ contains an even palindrome if and only if there exists $k > 0$ and $i_1, \ldots, i_k \in [\![1, n]\!]$ such that $u_{i_1} \cdots u_{i_k} = v_{i_1} \cdots v_{i_k}$ ◀