Keyboards as a new model of computation

Yoan Géran, Bastien Laboureix, *Corto Mascle*, Valentin D. Richard

September 23, 2021

Context

Context

Context

Malfunctioning keyboard

We try to write the word `bip`.

Malfunctioning keyboard

We try to write the word `bip`.

1. We press the `b` key: it writes “bis”.

“bis|”

Malfunctioning keyboard

We try to write the word `bip`.

1. We press the `b` key: it writes "bis".
2. We press the `i` key: it erases two letters, writes "lo" and moves the cursor to the left.

"bl|o"

Malfunctioning keyboard

We try to write the word `bip`.

1. We press the `b` key: it writes “bis”.
2. We press the `i` key: it erases two letters, writes “lo” and moves the cursor to the left.
3. We press the `p` key: it moves the cursor to the right and writes “op”.

“bloop|”

Instead of “bip”, the keyboard wrote “bloop”!

What do we do?

We could try to fix the keyboard...

What do we do?

We could try to fix the keyboard...

...or we could try to see what we can do with it! Can we write any word? If not, which words can we write?

Modelling

Atomic operations

- `a` for $a \in \Sigma$: writes "a" to the left of the cursor.

Modelling

Atomic operations

- `a` for $a \in \Sigma$: writes "a" to the left of the cursor.
- $\leftarrow$: erases the letter to the left of the cursor.

Modelling

Atomic operations

- a for $a \in \Sigma$: writes "a" to the left of the cursor.
- $\leftarrow$: erases the letter to the left of the cursor.
- $\blacktriangleleft$ and $\blacktriangleright$: moves the cursor to the left and to the right respectively.

Modelling

Atomic operations

- $\mathtt{a}$ for $a \in \Sigma$: writes "a" to the left of the cursor.
- $\leftarrow$: erases the letter to the left of the cursor.
- $\blacktriangleleft$ and $\blacktriangleright$: moves the cursor to the left and to the right respectively.

Keyboard

- A key is a sequence of atomic operations.

Modelling

Atomic operations

- $\texttt{a}$ for $a \in \Sigma$: writes "a" to the left of the cursor.
- $\leftarrow$: erases the letter to the left of the cursor.
- $\blacktriangleleft$ and $\blacktriangleright$: moves the cursor to the left and to the right respectively.

Keyboard

- A key is a sequence of atomic operations.
- A keyboard is a finite set of keys.

Atomic operations

- $\mathtt{a}$ for $a \in \Sigma$: writes “a” to the left of the cursor.
- $\leftarrow$: erases the letter to the left of the cursor.
- $\blacktriangleleft$ and $\blacktriangleright$: moves the cursor to the left and to the right respectively.

Keyboard

- A key is a sequence of atomic operations.
- A keyboard is a finite set of keys.

Our broken keyboard

We wrote “bloop” by pressing three keys:

$$\{\mathtt{bis}, \leftarrow\leftarrow\mathtt{lo}\blacktriangleleft, \blacktriangleright\mathtt{op}\}.$$

Modelling

- If the current word is uv with the cursor between u and v, the configuration is denoted $\langle u|v\rangle$.

Modelling

- If the current word is uv with the cursor between u and v, the configuration is denoted $\langle u|v\rangle$.
- Keys induce actions on the configurations.

Modelling

- If the current word is uv with the cursor between u and v, the configuration is denoted $\langle u|v\rangle$.
- Keys induce actions on the configurations.

$$\langle u|v\rangle \cdot a = \langle ua|v\rangle \text{ if } a \text{ is a letter.}$$

$$\langle \varepsilon|v\rangle \cdot \leftarrow = \langle \varepsilon|v\rangle \quad \text{and} \quad \left\langle u'a\middle|v\right\rangle \cdot \leftarrow = \left\langle u'\middle|v\right\rangle$$

$$\langle \varepsilon|v\rangle \cdot \blacktriangleleft = \langle \varepsilon|v\rangle \quad \text{and} \quad \left\langle u'a\middle|v\right\rangle \cdot \blacktriangleleft = \left\langle u'\middle|av\right\rangle$$

$$\langle u|\varepsilon\rangle \cdot \blacktriangleright = \langle u|\varepsilon\rangle \quad \text{and} \quad \left\langle u\middle|av'\right\rangle \cdot \blacktriangleright = \left\langle ua\middle|v'\right\rangle$$

Modelling

Applying a key to a configuration

We apply $t = \leftarrow a \blacktriangleright$ to $\langle c|d \rangle$.

Modelling

Applying a key to a configuration

We apply $t = \leftarrow a \blacktriangleright$ to $\langle c|d\rangle$.

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

Hence $\langle c|d\rangle \xrightarrow{t} \langle ad|\varepsilon\rangle$.

Language

The language of a keyboard K is the set of words we can obtain from configuration $\langle \varepsilon | \varepsilon \rangle$ by applying a sequence of keys from K,

$$\mathcal{L}(K) = \left\{ uv \;\middle|\; \exists t_1, \ldots, t_n \in K, \langle \varepsilon | \varepsilon \rangle \xrightarrow{t_1 \ldots t_n} \langle u | v \rangle \right\}.$$

Language

The language of a keyboard K is the set of words we can obtain from configuration $\langle \varepsilon | \varepsilon \rangle$ by applying a sequence of keys from K,

$$\mathcal{L}(K) = \left\{ uv \;\middle|\; \exists t_1, \ldots, t_n \in K, \langle \varepsilon | \varepsilon \rangle \xrightarrow{t_1 \ldots t_n} \langle u | v \rangle \right\}.$$

Let $t_1 = \texttt{bis}, t_2 = \leftarrow\leftarrow\texttt{lo}\blacktriangleleft, t_3 = \blacktriangleright\texttt{op}$ and $K = \{t_1, t_2, t_3\}$.

$$\begin{aligned} \langle \varepsilon | \varepsilon \rangle &\xrightarrow{t_1} \langle bis | \varepsilon \rangle \\ &\xrightarrow{t_2} \langle bl | o \rangle \\ &\xrightarrow{t_3} \langle bloop | \varepsilon \rangle \end{aligned}$$

The word “bloop” is in the language of K.

Some examples

- The language of $K = \{\texttt{ab}, \texttt{a}\}$?

Some examples

- The language of $K = \{\mathtt{ab}, \mathtt{a}\}$?

$$(ab + a)^+.$$

Some examples

- The language of $K = \{\texttt{ab}, \texttt{a}\}$?

$$(ab + a)^+.$$

- The language of $K = \{\texttt{a}, \texttt{b}◀, \varepsilon\}$?

Some examples

- The language of $K = \{\texttt{ab}, \texttt{a}\}$?

$$(ab + a)^+.$$

- The language of $K = \{\texttt{a}, \texttt{b}\blacktriangleleft, \varepsilon\}$?

$$a^*b^*.$$

Some examples

- The language of $K = \{\texttt{ab}, \texttt{a}\}$?

$$(ab + a)^+.$$

- The language of $K = \{\texttt{a}, \texttt{b}\blacktriangleleft, \varepsilon\}$?

$$a^*b^*.$$

- The language of $K = \{(), \blacktriangleleft, \blacktriangleright\}$?

Some examples

- The language of $K = \{\mathtt{ab}, \mathtt{a}\}$?

$$(ab + a)^+.$$

- The language of $K = \{\mathtt{a}, \mathtt{b}\blacktriangleleft, \varepsilon\}$?

$$a^* b^*.$$

- The language of $K = \{(), \blacktriangleleft, \blacktriangleright\}$?

  The Dyck language (correctly nested sequences of brackets)!

Some examples

- A keyboard for $\{w \in \{a,b\}^* \mid |w|_a \equiv 0[3]\}$?

Some examples

- A keyboard for $\{w \in \{a,b\}^* \mid |w|_a \equiv 0[3]\}$?

$$K = \{aaa, \blacktriangleleft, b\}.$$

Some examples

- A keyboard for $\{w \in \{a, b\}^* \mid |w|_a \equiv 0[3]\}$?

$$K = \{aaa, \blacktriangleleft, b\}.$$

- A keyboard for $\{w \in \{a, b\}^* \mid |w|_a = |w|_b\}$?

Some examples

- A keyboard for $\{w \in \{a,b\}^* \mid |w|_a \equiv 0[3]\}$?

$$K = \{aaa, \blacktriangleleft, b\}.$$

- A keyboard for $\{w \in \{a,b\}^* \mid |w|_a = |w|_b\}$?

$$K = \{ab, ba, \blacktriangleleft\}.$$

Some examples

- A keyboard for $\{w \in \{a,b\}^* \mid |w|_a \equiv 0[3]\}$?

$$K = \{aaa, \blacktriangleleft, b\}.$$

- A keyboard for $\{w \in \{a,b\}^* \mid |w|_a = |w|_b\}$?

$$K = \{ab, ba, \blacktriangleleft\}.$$

- A keyboard for ab^+?

Some examples

- A keyboard for $\{w \in \{a,b\}^* \mid |w|_a \equiv 0[3]\}$?

$$K = \{aaa, \blacktriangleleft, b\}.$$

- A keyboard for $\{w \in \{a,b\}^* \mid |w|_a = |w|_b\}$?

$$K = \{ab, ba, \blacktriangleleft\}.$$

- A keyboard for ab^+?

$$K = \{\leftarrow ab \blacktriangleleft\}$$

Keyboards expressivity

Keyboard languages are recursive, but which languages can keyboards represent?

Keyboards expressivity

Keyboard languages are recursive, but which languages can keyboards represent?

Are those keyboard languages?

- Finite languages?
- $\left\{a^{2n+1} \;\middle|\; n \in \mathbb{N}\right\}$?

Keyboards expressivity

Keyboard languages are recursive, but which languages can keyboards represent?

Are those keyboard languages?

- Finite languages?
- $\{a^{2n+1} \mid n \in \mathbb{N}\}$?

Add an "Entry"!

An "Entry" symbol ■ which validates the word!

Final keys

- Some keys, called final keys, validate the current word. They end with an “Entry” ■.

Final keys

- Some keys, called final keys, validate the current word.
  They end with an "Entry" ■.
- The current word is accepted when the entry is applied.

$$\mathcal{L}(K) = \left\{ uv \;\middle|\; \exists t_1, \ldots, t_n \text{ and } t_f \text{ final such that } \langle \varepsilon | \varepsilon \rangle \xrightarrow{t_1 \ldots t_n t_f} \langle u | v \rangle \right\}$$

Final keys

- Some keys, called final keys, validate the current word. They end with an "Entry" ■.
- The current word is accepted when the entry is applied.

$$\mathcal{L}(K) = \left\{ uv \;\middle|\; \exists t_1, \ldots, t_n \text{ and } t_f \text{ final such that } \langle \varepsilon | \varepsilon \rangle \xrightarrow{t_1 \ldots t_n t_f} \langle u | v \rangle \right\}$$

■ is useful!

The language $\left\{ a^{2n+1} \;\middle|\; n \in \mathbb{N} \right\}$ is recognized by $\{aa, a\blacksquare\}$.

Two types of keyboards

- Keyboards with entry are called **manual**.
- Keyboards without entry are called **automatic**.

Two types of keyboards

- Keyboards with entry are called **manual**.
- Keyboards without entry are called **automatic**.

Theorem (Simulation)

The language of an automatic keyboard K_A is also recognized by the (manual) keyboard

$$K_M = \{t \mid t \in K_A\} \cup \{t\blacksquare \mid t \in K_A\}.$$

Edge effects

The action of a key may differ when the cursor is close to an end of the word!

An automatic keyboard for $\left\{a^{2n+1} \;\middle|\; n \in \mathbb{N}\right\}$

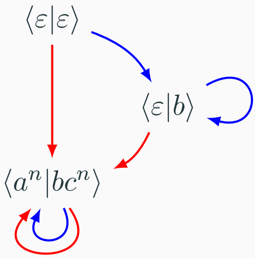
This language is recognized by the keyboard $\{t_1 = \leftarrow \texttt{a}, t_2 = \leftarrow \texttt{aaa}\}$.

$$\langle \varepsilon | \varepsilon \rangle \xrightarrow{t_1} \langle a | \varepsilon \rangle \qquad \left\langle a^{2n+1} \middle| \varepsilon \right\rangle \xrightarrow{t_1} \left\langle a^{2n+1} \middle| \varepsilon \right\rangle$$

$$\langle \varepsilon | \varepsilon \rangle \xrightarrow{t_2} \langle aaa | \varepsilon \rangle \qquad \left\langle a^{2n+1} \middle| \varepsilon \right\rangle \xrightarrow{t_2} \left\langle a^{2n+3} \middle| \varepsilon \right\rangle$$

The language $a^n bc^n$

$L = a^n bc^n$ is recognized by $K = \{$b▶◀←, ▶←abc◀◀$\}$.

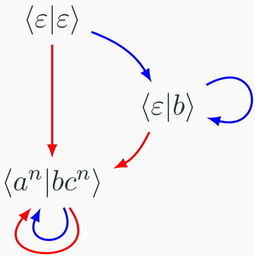


Starting from $\langle \varepsilon | \varepsilon \rangle$, b▶◀← writes a b, and from $\langle a^n | bc^n \rangle$ it doesn't do anything.

Invariant: we are always in a configuration of the form $\langle a^n | bc^n \rangle$.

The language $a^n bc^n$

$L = a^n bc^n$ is recognized by $K = \{$`b▶◀←`, `▶←abc◀◀`$\}$.



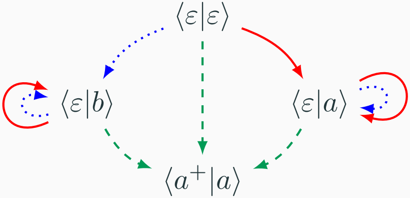
$$
\begin{aligned}
\langle \varepsilon | \varepsilon \rangle &\xrightarrow{t_2} \langle a | bc \rangle \\
&\xrightarrow{t_2} \langle aa | bcc \rangle \\
&\xrightarrow{t_1} \langle aa | bcc \rangle \\
&\xrightarrow{t_2} \langle aaa | bccc \rangle
\end{aligned}
$$

Starting from $\langle \varepsilon | \varepsilon \rangle$, `b▶◀←` writes a b, and from $\langle a^n | bc^n \rangle$ it doesn't do anything.

Invariant: we are always in a configuration of the form $\langle a^n | bc^n \rangle$.

The language $a^* + b$

$L = a^* + b$ is recognized by $K = \{$b▶◀←, a▶◀←, ▶←aa◀$\}$.



From a configuration of the form $\langle a^n | a \rangle$, b▶◀← and a▶◀← have no effect, but ▶←aa◀ adds an a and leads to $\langle a^{n+1} | a \rangle$.

Classes of languages

- B: with $\leftarrow$
- E: with $\blacksquare$
- L: with $\blacktriangleleft$
- A: with $\blacktriangleright$ and $\blacktriangleleft$

$\mathsf{MK} : \{\}$	$\mathsf{LK} : \{\blacktriangleleft\}$	$\mathsf{AK} : \{\blacktriangleleft, \blacktriangleright\}$
$\mathsf{EK} : \{\blacksquare\}$	$\mathsf{LEK} : \{\blacktriangleleft, \blacksquare\}$	$\mathsf{AEK} : \{\blacktriangleleft, \blacktriangleright, \blacksquare\}$
$\mathsf{BK} : \{\leftarrow\}$	$\mathsf{BLK} : \{\blacktriangleleft, \leftarrow\}$	$\mathsf{BAK} : \{\blacktriangleleft, \blacktriangleright, \leftarrow\}$
$\mathsf{BEK} : \{\leftarrow, \blacksquare\}$	$\mathsf{BLEK} : \{\blacktriangleleft, \leftarrow, \blacksquare\}$	$\mathsf{BAEK} : \{\blacktriangleleft, \blacktriangleright, \leftarrow, \blacksquare\}$

Visiting the zoo

BEK{$\leftarrow$, ■}: A gentle animal

Lemma

For all $c \in A$, $c\leftarrow$ *is equivalent to* ε.

BEK{←, ■}: A gentle animal

Lemma

For all $c \in A$, $c\leftarrow$ is equivalent to ε.

Simplification

$$
\begin{aligned}
\leftarrow abb\leftarrow ba\leftarrow^3 &\iff \leftarrow abb\leftarrow ba\leftarrow^3 \\
&\iff \leftarrow abba\leftarrow\leftarrow^2 \\
&\iff \leftarrow abb\leftarrow\leftarrow \\
&\iff \leftarrow ab\leftarrow \\
&\iff \leftarrow a
\end{aligned}
$$

BEK$\{\leftarrow, \blacksquare\}$: A gentle animal

Lemma (BEK normal form)

Every key of BEK is equivalent to a key of the form $\leftarrow^* A^*$.

Further, as we start on the empty configuration and never apply any ◀, the cursor is always on the right end of the word.

Lemma

Applying a sequence of keys of BEK from a configuration $\langle w|\varepsilon\rangle$ *yields a configuration of the form* $\langle w'|\varepsilon\rangle$.

BEK$\{\leftarrow, \blacksquare\}$: A gentle animal

Applying a key of BEK comes down to erasing a few letters at the end of the word, then writing a few others.

BEK$\{\leftarrow, \blacksquare\}$: A gentle animal

Applying a key of BEK comes down to erasing a few letters at the end of the word, then writing a few others.

Theorem

For all keyboard K of BEK *there exists a pushdown automaton recognising $\mathcal{L}(K)$.*

BEK$\{\leftarrow, \blacksquare\}$: A gentle animal

Applying a key of BEK comes down to erasing a few letters at the end of the word, then writing a few others.

Theorem

For all keyboard K of BEK *there exists a pushdown automaton recognising $\mathcal{L}(K)$.*

Theorem

For all keyboard K of BEK *there exists an NFA of polynomial size recognising $\mathcal{L}(K)$.*

Theorem

For all keyboard K of BEK *there exists an NFA of polynomial size recognising $\mathcal{L}(K)$.*

We can see a run of a BEK keyboard like this:

Each key erases some letters,

BEK$\{\leftarrow, \blacksquare\}$: Proof of rationality

Theorem

For all keyboard K of BEK *there exists an NFA of polynomial size recognising $\mathcal{L}(K)$.*

We can see a run of a BEK keyboard like this:

Each key erases some letters, writes some letters that will never be erased,

BEK$\{\leftarrow, \blacksquare\}$: Proof of rationality

Theorem

For all keyboard K of BEK *there exists an NFA of polynomial size recognising $\mathcal{L}(K)$.*

We can see a run of a BEK keyboard like this:

Each key erases some letters, writes some letters that will never be erased, then some letters that will eventually be erased .

$$\leftarrow^m a_1 \cdots a_n b_1 \cdots b_p$$

BEK$\{\leftarrow, \blacksquare\}$: **Proof of rationality**

Say we want to write abc with the keyboard
$K = \{\leftarrow^2 aba, \leftarrow^4 bc\}$:

Say we want to write abc with the keyboard $K = \{\leftarrow^2 aba, \leftarrow^4 bc\}$:

$$\underbrace{\varepsilon \xrightarrow{\leftarrow^2 aba} aba}_{\text{writing}} \underbrace{\xrightarrow{\leftarrow^2 aba} aaba \xrightarrow{\leftarrow^2 aba} aaaba}_{\text{adjusting extra letters}} \underbrace{\xrightarrow{\leftarrow^4 bc} abc}_{\text{writing}}$$

BEK$\{\leftarrow, \blacksquare\}$: Proof of rationality

Say we want to write abc with the keyboard $K = \{\leftarrow^2 aba, \leftarrow^4 bc\}$:

$$\underbrace{\varepsilon \xrightarrow{\leftarrow^2 aba} aba}_{\text{writing}} \underbrace{\xrightarrow{\leftarrow^2 aba} aaba \xrightarrow{\leftarrow^2 aba} aaaba}_{\text{adjusting extra letters}} \underbrace{\xrightarrow{\leftarrow^4 bc} abc}_{\text{writing}}$$

→ We only care about the **number** of letters that will be erased, not about the word they form!

BEK$\{\leftarrow, \blacksquare\}$: **Proof of rationality**

To each key we can associate its numerical *trace*, which is the number of letters it writes minus its number of $\leftarrow$.

$$\leftarrow^2 aba \longmapsto +1$$
$$\leftarrow^4 bc \longmapsto -2$$

BEK$\{\leftarrow, \blacksquare\}$: **Proof of rationality**

To each key we can associate its numerical *trace*, which is the number of letters it writes minus its number of $\leftarrow$.

$$\begin{aligned} \leftarrow^2 aba &\longmapsto +1 \\ \leftarrow^4 bc &\longmapsto -2 \end{aligned}$$

Let p be the gcd of all the traces of keys of K.

To each key we can associate its numerical *trace*, which is the number of letters it writes minus its number of $\leftarrow$.

$$\leftarrow^2 aba \longmapsto +1$$
$$\leftarrow^4 bc \longmapsto -2$$

Let p be the gcd of all the traces of keys of K.

Proposition

We can turn i extra letters into j extra letters if and only if p divides $|i-j|$ (up to some minor conditions).

BEK$\{\leftarrow, \blacksquare\}$: Proof of rationality

We construct an NFA with states $\{0, \ldots, n\}$, n being the maximal length of a key of K.

BEK$\{\leftarrow, \blacksquare\}$: Proof of rationality

We construct an NFA with states $\{0, \ldots, n\}$, n being the maximal length of a key of K.

States count the number of extra letters.

$\mathsf{BEK}\{\leftarrow, \blacksquare\}$: Proof of rationality

We construct an NFA with states $\{0, \ldots, n\}$, n being the maximal length of a key of K.

States count the number of extra letters.

It has two types of transitions:

- $i \xrightarrow{u} j$ simulates the application of a key of the form $\leftarrow^{i} uv$ with $|v| = j$.

BEK$\{\leftarrow, \blacksquare\}$: **Proof of rationality**

We construct an NFA with states $\{0, \ldots, n\}$, n being the maximal length of a key of K.

States count the number of extra letters.

It has two types of transitions:

- $i \xrightarrow{u} j$ simulates the application of a key of the form $\leftarrow^{i} u v$ with $|v| = j$.
- $i \xrightarrow{\varepsilon} j$ simulates the application of a series of keys not affecting the permanent letters but switching the number of extra letters from i to j.

BEK$\{\leftarrow, \blacksquare\}$: **Proof of rationality**

Theorem

For all keyboard K of BEK *there exists an NFA of polynomial size recognising $\mathcal{L}(K)$.*

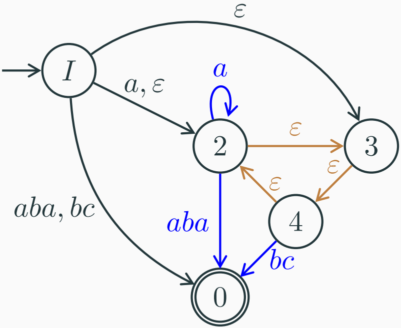
For the keyboard $K = \{\leftarrow^2 aba, \leftarrow^4 bc\}$, we get:

BEK$\{\leftarrow, \blacksquare\}$: Proof of rationality

Theorem

For all keyboard K of BEK *there exists an NFA of polynomial size recognising $\mathcal{L}(K)$.*

For the keyboard $K = \{\leftarrow^2 aba, \leftarrow^4 bc\}$, we get:



$$L(K) = a^*(aba + bc)$$

BLEK$\{\blacktriangleleft, \leftarrow, \blacksquare\}$: A ferocious creature?

The problem with BLEK

The left arrow allows for modifications anywhere in the word!

For instance, $\blacktriangleleft^3\leftarrow$ allows one to erase letters inside the word.

BLEK{◀, ←, ■}: A tamed creature

The problem with BLEK

The left arrow allows for modifications anywhere in the word!

For instance, $◀^3←$ allows one to erase letters inside the word.

Not so fast!

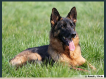
The letters to the right of the word are “fixed”.

$$\langle u | v \rangle \xrightarrow{a} \langle ua | v \rangle$$

$$\langle ua | v \rangle \xrightarrow{◀} \langle u | av \rangle$$

$$\langle ua | v \rangle \xrightarrow{\leftarrow} \langle u | v \rangle$$

BLEK$\{\blacktriangleleft, \leftarrow, \blacksquare\}$: A tamed creature

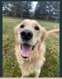


Lemma (A property of BLEK)

Any sequence of keys of BLEK *applied from a configuration* $\langle u|v\rangle$ *leads to a configuration of the following form:* $\langle u'|wv\rangle$.

The left arrow can be interpreted as a way to record the letter to the left of the cursor.

BLEK$\{\blacktriangleleft, \leftarrow, \blacksquare\}$: A tamed creature



Lemma (A property of BLEK)

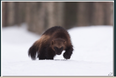
Any sequence of keys of BLEK *applied from a configuration* $\langle u|v\rangle$ *leads to a configuration of the following form:* $\langle u'|wv\rangle$.

The left arrow can be interpreted as a way to record the letter to the left of the cursor.

Theorem

For all keyboard K *of* BLEK *there exists a pushdown automaton of polynomial size recognising* $\mathcal{L}(K)$.

AEK{◄, ►, ■}: A wild being

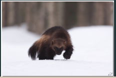


No more erasing, we only add letters!

Lemma (Monotony)

Applying any sequence of keys of AEK *to a configuration* $\langle u|v\rangle$ *yields a configuration* $\langle u'|v'\rangle$ *with* $|u'| + |v'| \geq |u| + |v|$.

AEK$\{\blacktriangleleft, \blacktriangleright, \blacksquare\}$: A wild being



No more erasing, we only add letters!

Lemma (Monotony)

Applying any sequence of keys of AEK *to a configuration* $\langle u|v\rangle$ *yields a configuration* $\langle u'|v'\rangle$ *with* $|u'| + |v'| \geq |u| + |v|$.

Theorem

For all keyboard K *of* AEK *there exists a linear bounded automaton of polynomial size recognising* $\mathcal{L}(K)$.

BAEK$\{\blacktriangleleft, \blacktriangleright, \leftarrow, \blacksquare\}$: The monster

BAEK does not have any of the previous properties.

BAEK$\{\blacktriangleleft, \blacktriangleright, \leftarrow, \blacksquare\}$: The monster

BAEK does not have any of the previous properties.

Proposition

Since a key can only modify the size of a configuration in a bounded way, if w is accepted, then some slightly smaller or longer word is also accepted.

Application

$\left\{ \mathsf{a}^{n^2} \,\middle|\, n \in \mathbb{N} \right\}$ and $\{\mathsf{a}^p \mid p \text{ prime}\}$ are not recognised by any keyboard.

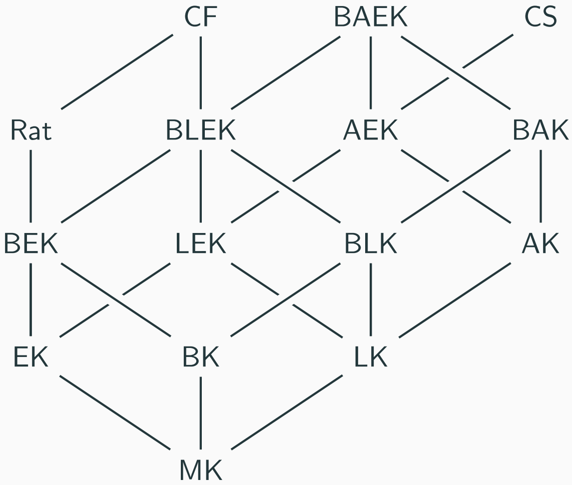
The keyboard hierarchy

Strict hierarchy theorem

Theorem

- *All 12 keyboard language classes we considered are distinct.*
  In particular, not all keyboards are automatic!
- *The only inclusions between classes are trivial ones*
  (except possibly for the inclusions of EK *and* BEK *in* BAK*).*

A strict hierarchy



	Membership	Universality
MK	P	P
EK	P	P
BK	P	coNP
BEK	P	PSPACE
LK	P	?
LEK	P	?
BLK	P	?
BLEK	P	?
AK	NP	?
AEK	NP	?
BAK	?	?
BAEK	?	?

	Complement	Concatenation	Intersection
MK	a^{2n}	a^*c^*	$(ab+bb+ba)^* \cap (ba+b)^*$
EK	a^{2n+3}	a^*c^*	$(ab+bb+ba)^* \cap (ba+b)^*$
BK	$(a+b)^*$ with $\|A\| = 3$	a^*c^*	$\mathcal{L}(K_1) \cap \mathcal{L}(K_2)$
BEK	$(a+b)^*$ with $\|A\| = 3$	a^*c^*	$\mathcal{L}(K_1) \cap \mathcal{L}(K_2)$
LK	a^{2n}	$a^n b^n c^m d^m$	$a^n b^n c^n$
LEK	a^{2n+3}	$a^n c a^n a^m c a^m$	$a^n b^n c^n$
BLK	$\{w \mid \|w\|_a \leq 1\}$	$(aa)^*(b+b^2)$	$a^n b^n c^n$
BLEK	$\{w \mid \|w\|_a \leq 1\}$	$a^n c a^n a^m c a^m$	$a^n b^n c^n$
AK	a^{2n}	$a^n b^n c^m d^m$	$a^n b^n c^n$
AEK	a^{2n+3}	$a^n c a^n a^m c a^m$	$a^n b^n c^n$
BAK	?	$a^n c a^n a^m c a^m$	$a^n b^n c^n$
BAEK	?	$a^n c a^n a^m c a^m$	$a^n b^n c^n$

	Mirror	Morphism	Union
MK	✓	✓	$a^* + b^*$
EK	b^*a	✓	$a^* + b^*$
BK	b^*a	$(a^2)^*(b+c)$	$a^* + b^*$
BEK	b^*a	?	$a^* + b^*$
LK	$b^n c(ca)^{n-1} a$	?	$a^* + b^*$
LEK	$c + cb(ba)^* a$	?	$a^* + b^*$
BLK	$(b + b^2)a^*$	$(a^2)^*(b+c)$	$a^* + b^*$
BLEK	$c + cb(ba)^* a$	?	$a^* + b^*$
AK	✓	?	$a^* + b^*$
AEK	✓	?	$a^* + b^*$
BAK	?	$w(c+d)\widetilde{w}$	$a^n ca^n \cup b^n cb^n$
BAEK	?	?	$a^n ca^n \cup b^n cb^n$

EK $\not\subset$ BK

Lemma

$L = (a^2)^*(b + b^2)$ *is recognized by* $\{$`aa`, `b`$\blacksquare$, `bb`$\blacksquare\}$ *and is not in* BK.

Proof.

If $\mathcal{L}(K) = L$, there exists τ (of normal form $\leftarrow^k b^2$) writing b^2. We distinguish cases according to the value of k.

EK $\not\subset$ BK

Lemma

$L = (a^2)^*(b + b^2)$ is recognized by $\{\texttt{aa}, \texttt{b}\blacksquare, \texttt{bb}\blacksquare\}$ and is not in BK.

Proof.

If $\mathcal{L}(K) = L$, there exists τ (of normal form $\leftarrow^k b^2$) writing b^2. We distinguish cases according to the value of k.

If $k = 0$, then $\tau \sim b^2$: we then have

$$\varepsilon \cdot \tau \cdot \tau = b^2 \cdot \tau = b^4 \in L.$$

Contradiction

$\square$

EK $\not\subset$ BK

Lemma

$L = (a^2)^*(b + b^2)$ is recognized by $\{\texttt{aa}, \texttt{b}\blacksquare, \texttt{bb}\blacksquare\}$ and is not in BK.

Proof.

If $\mathcal{L}(K) = L$, there exists τ (of normal form $\leftarrow^k b^2$) writing b^2. We distinguish cases according to the value of k.

If $k = 1$, then $\tau \sim \leftarrow b^2$: we then have

$$\varepsilon \cdot \tau \cdot \tau = b^2 \cdot \tau = b^3 \in L.$$

Contradiction

$\square$

EK $\not\subset$ BK

Lemma

$L = (a^2)^*(b + b^2)$ *is recognized by* $\{\texttt{aa}, \texttt{b}\blacksquare, \texttt{bb}\blacksquare\}$ *and is not in* BK.

Proof.

If $\mathcal{L}(K) = L$, there exists τ (of normal form $\leftarrow^k b^2$) writing b^2. We distinguish cases according to the value of k.

If $k > 1$ and ***k* even**: from $a^{2k}b \in L$ we obtain

$$a^{2k}b \cdot \tau = a^{k+1}b^2 \in L.$$

Contradiction

$\square$

EK $\not\subset$ BK

Lemma

$L = (a^2)^*(b + b^2)$ *is recognized by* $\{$`aa`, `b■`, `bb■`$\}$ *and is not in* BK.

Proof.

If $\mathcal{L}(K) = L$, there exists τ (of normal form $\leftarrow^k b^2$) writing b^2. We distinguish cases according to the value of k.

If $k > 1$ and ***k* odd**: from $a^{2k}b^2 \in L$ we obtain

$$a^{2k}b^2 \cdot \tau = a^{k+2}b^2 \in L.$$

Contradiction

□

Research goes on

Decision problems

The membership problem

$$\text{Membership} : \begin{cases} \textsc{Input} : & K \in \mathsf{BAEK}, w \in A^* \\ \textsc{Output} : & w \in \mathcal{L}(K)? \end{cases}$$

- BEK: $\in$ PTIME.
- BLEK: $\in$ PTIME.
- AEK: $\in$ NP.
- BAEK?

Can we do better?

Decision problems

Universality problem

$$\text{Universality} : \begin{cases} \textsc{Input}: & K \in \mathsf{BAEK} \\ \textsc{Output}: & \mathcal{L}(K) = A^*? \end{cases}$$

- BEK: $\in$ PSPACE
- BLEK?
- AEK?
- BAEK?

Other questions?

- Do we have $\mathsf{BEK} \subset \mathsf{BAK}$? $\mathsf{EK} \subset \mathsf{BAK}$?

Other questions?

- Do we have BEK $\subset$ BAK? EK $\subset$ BAK?
- Are all rational languages in BAEK?

$a^* + b^*$ seems to not be in BAEK!

Other questions?

- Do we have $\mathsf{BEK} \subset \mathsf{BAK}$? $\mathsf{EK} \subset \mathsf{BAK}$?
- Are all rational languages in BAEK?
- Is BAEK included in context-sensitive languages? Context-free ones?

Study the keyboard $\{a\blacktriangleright\blacktriangleright, b\blacktriangleleft\blacktriangleleft\}$.

Other questions?

- Do we have $\mathsf{BEK} \subset \mathsf{BAK}$? $\mathsf{EK} \subset \mathsf{BAK}$?
- Are all rational languages in BAEK?
- Is BAEK included in context-sensitive languages? Context-free ones?
- Relations to other known models?

Thanks for your attention!

Thanks for your attention!

Questions?

An example

$$K_C = \{\leftarrow\texttt{a}\lozenge\blacklozenge, \leftarrow\leftarrow\texttt{b}\lozenge\blacklozenge\blacklozenge\}.$$

An example

Some as and bs separated by $\lozenge$ and $\blacklozenge$.

- Between two a: $\lozenge$.
- Between two b: $\lozenge$.
- Between an a and a b: nothing.
- Between a b and an a: $\lozenge\blacklozenge$.

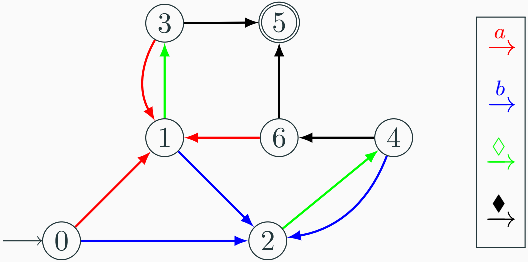
$$K_C = \{\leftarrow\mathtt{a}\lozenge\blacklozenge, \leftarrow\leftarrow\mathtt{b}\lozenge\blacklozenge\blacklozenge\}.$$

An example

Some as and bs separated by $\lozenge$ and $\blacklozenge$.

- Between two a: $\lozenge$.
- Between two b: $\lozenge$.
- Between an a and a b: nothing.
- Between a b and an a: $\lozenge\blacklozenge$.

$$K_C = \{\leftarrow\texttt{a}\lozenge\blacklozenge, \leftarrow\leftarrow\texttt{b}\lozenge\blacklozenge\blacklozenge\}.$$



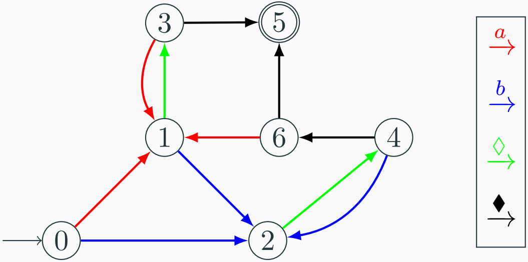
An example

Some as and bs separated by $\lozenge$ and $\blacklozenge$.

- Between two a: $\lozenge$.
- Between two b: $\lozenge$.
- Between an a and a b: nothing.
- Between a b and an a: $\lozenge\blacklozenge$.

$$K_C = \{\leftarrow\mathtt{a}\lozenge\blacklozenge, \leftarrow\leftarrow\mathtt{b}\lozenge\blacklozenge\blacklozenge\}.$$

$$(b(\lozenge b)^*\lozenge\blacklozenge + (a + b(\lozenge b)^*\lozenge\blacklozenge a)((\lozenge + b(\lozenge b)^*\lozenge\blacklozenge)a)^*(\lozenge + b(\lozenge b)^*\lozenge\blacklozenge))\blacklozenge$$



Class inclusions

Lemma ($\mathsf{LK} \not\subset \mathsf{BEK}$)

The language of even palindromes is in LK *via* $\{aa\blacktriangleleft, bb\blacktriangleleft\}$*, and is not rational.*

Lemma ($\mathsf{BK} \not\subset \mathsf{AK}$ and $\mathsf{EK} \not\subset \mathsf{AK}$)

Finite languages are in EK *and* BK*, but not* AK.

BAK $\not\subseteq$ AEK

Lemma

$L = a^* + b^* \notin$ AEK.

Proof.

- There is a (non-final) key writing an a.
- There is a (non-final) key writing a b.

We can write a word with a and b! $\square$

Rat $\not\subset$ BEK

Lemma

$a^*b^* \notin$ BEK

Proof.

- There exists τ writing a and applying entry (τ is of the form $\leftarrow^k a\blacksquare$).
- There exists τ' writing arbitrarily many b without entry (for instance $k+1$).

$\tau'\tau$ writes ba and ends the execution. $\square$