The cost of being co-Büchi is nonlinear*

Jerzy Marcinkowski, Jakub Michaliszyn

Institute of Computer Science,
University Of Wrocław,
ul. Joliot-Curie 15, 50-383 Wrocław, Poland

Abstract. It is well known, and easy to see, that not each nondeterministic Büchi automaton on infinite words can be simulated by a nondeterministic co-Büchi automaton. We show that in the cases when such a simulation is possible, the number of states needed for it can grow nonlinearly. More precisely, we show a sequence of - as we believe, simple and elegant - languages which witness the existence of a nondeterministic Büchi automaton with \( n \) states, which can be simulated by a nondeterministic co-Büchi automaton, but cannot be simulated by any nondeterministic co-Büchi automaton with less than \( c \cdot n^{7/6} \) states for some constant \( c \). This improves on the best previously known lower bound of \( 3(n - 1)/2 \).

1 Introduction

1.1 Previous work

In 1962 Büchi was the first to introduce finite automata on infinite words. He needed them to solve some fundamental decision problems in mathematics and logic ([2], [6], [7]). They became a popular area of research due to their elegance and the tight relation between automata on infinite objects and monadic second-order logic. Nowadays, automata are seen as a very useful tool in verification and specification of nonterminating systems. This is why the complexity of problems concerning automata has recently been considered a hot topic (e. g. [5], [1]).

To serve different applications, different types of automata were introduced. In his proof of the decidability of the satisfiability of S1S, Büchi introduced nondeterministic automata on infinite words (NBW), which are a natural tool to model things that happen infinitely often. In a Büchi automaton, some of the states are accepting and a run on an infinite word is accepting if and only if it visits some accepting state infinitely often ([2]). Dually, a run of co-Büchi

* Research supported by Polish Ministry of Science and Higher Education research project N206 022 31/3660, 2006/2009.

1 Shortly before submitting this paper, we learned that a paper Co-ing Büchi: Less Open, Much More Practical by Udi Boker and Orna Kupferman was accepted for LICS 2009, and that it probably contains results that are similar to ours and slightly stronger. However, at the moment of our submission, their paper was not published, and as far as we know, was not available on the Web.
automaton (NCW) is accepting if and only if it visits non-accepting states only finitely often. There are also automata with more complicated accepting conditions – most well known of them are parity automata, Street automata, Rabin automata and Muller automata.

As in the case of finite automata on finite words, four basic types of transition relation can be considered: deterministic, nondeterministic, universal and alternating. In this paper, from now on, we only consider nondeterministic automata.

The problem of comparing the power of different types of automata is well studied and understood. For example it is easy to see that not every language that can be recognized by a Büchi automaton on infinite words (such languages are called ω-regular languages) can be also recognized by a co-Büchi automaton. The most popular example of ω-regular language that cannot be expressed by NCW is the language $L = \{w | w \text{ has infinitely many } 0\text{'s}\}$ over the alphabet $\{0, 1\}$. On the other hand, it is not very hard to see that every language that can be recognized by a co-Büchi automaton is ω-regular.

As we said, the problem of comparing the power of different types of automata is well studied. But we are quite far from knowing everything about the number of states needed to simulate an automaton of one type by an automaton of another type – see for example the survey [3] to learn about the open problems in this area.

In this paper we consider the problem of the cost of simulating a Büchi automaton on infinite words by a co-Büchi automaton (if such NCW exists), left open in [3]: given a number $n \in \mathbb{N}$, for what $f(n)$ can we be sure that every nondeterministic Büchi automaton with no more than $n$ states, which can be simulated by a co-Büchi automaton, can be simulated by a co-Büchi automaton with at most $f(n)$ states?

There is a large gap between the known upper bound and the known lower bound for such a translation. The best currently known translation goes via intermediate deterministic Street automaton, involving exponential blowup of the number of states ([8], [1]). More precisely, for NBW with $n$ states, we get NCW with $2^O(n \log n)$ states. For a long time the best known lower bound for $f$ was nothing more than the trivial bound $n$. In 2007 it was shown that there is an NBW with equivalent NCW such that there is no NCW equivalent to this NBW on the same structure ([1]). The first non-trivial (and the best currently known) lower bound is linear - the result of [1] is that, for each $n \in \mathbb{N}$, there exists a NBW with $n$ states such that there is a NCW which recognizes the same language, but every such NCW has at least $3(n - 1)/2$ states.

There is a good reason why it is hard to show a lower bound for the above problem. The language (or rather, to be more precise, class of languages) used to show such a bound has to be hard enough to be expressed by a co-Büchi automaton, but on the other hand not too hard, because some (actually, most of) ω-regular languages cannot be expressed by NCW at all. The idea given in the proof of $3(n - 1)/2$ lower bound in [1], was to define a language which can be easily split into parts that can by recognized by a NBW but cannot by
recognized by a NCW. The language they used was \( L_k = \{ w \in \{0,1\}^\omega \mid \text{both 0 and 1 appear at least } k \text{ times in } w \} \). Let \( L_k^i = \{ w \in \{0,1\}^\omega \mid i \text{ appears infinitely often in } w \text{ and } (1-i) \text{ appear at least } k \text{ times in } w \} \), then it is easy to see that \( L_k = L_k^0 \cup L_k^1 \), \( L_k^i \) can be recognized by Büchi automata of size \( k \) and \( L_k^i \) cannot by recognized by any co-Büchi automaton. It is still, however, possible to build a NCW that recognizes \( L_k \) with \( 3k + 1 \) states and indeed, as it was proved in [1], every NCW recognizing \( L_k \) has at least \( 3k \) states.

### 1.2 Our contribution – a nonlinear lower bound

In this paper we give a strong improvement of the lower bound from [1]. We show that, for every integer \( k \), there is a language \( L_k \) such that \( L_k \) can be recognized by an NBW with \( \Theta(k^2) \) states, whereas every NCW that recognizes this language has at least \( \Theta(k^{7/3}) \) states. Actually, the smallest NCW we know, which recognizes \( L_k \), has \( \Theta(k^3) \) states, and we believe that this automaton is indeed minimal. In the terms of function \( f \) from the above subsection this means that \( f \) equals at least \( cn^{7/6} \) for some \( c \) (this is since \( n \) is \( \Theta(k^2) \)) and, if our conjecture concerning the size of a minimal automaton for \( L_k \) is true, \( f \) would equal at least \( cn^{3/2} \) for some constant \( c \).

The technical part of of this paper is organized as follows. In subsection 2.1 we give some basic definitions. In subsection 2.2 the definition of the language \( L_k \) is presented. Also in this section we show how this language can be recognized by a Büchi automaton with \( O(k^2) \) states and how \( L_k \) can be recognized by a co-Büchi automaton with \( O(k^3) \) states. The main theorem, saying that every co-Büchi automaton that recognizes \( L_k \) has at least \( \Theta(k^{7/3}) \) states is formulated in the end of subsection 2.2 and the rest of the paper is devoted to its proof.

### 2 Technical Part

#### 2.1 Preliminaries

A non-deterministic \( \omega \)-automaton is a quintuple \( \langle \Sigma, Q, q_0, \delta, \alpha \rangle \), where \( \Sigma \) is an alphabet, \( Q \) is a set of states, \( q_0 \in Q \) is an initial state, \( \delta \subseteq Q \times \Sigma \times Q \) is a transition relation and \( \alpha \subseteq Q \) is an accepting condition.

A run of \( \omega \)-automaton over a word \( w = w_1w_2\ldots \) is a sequence of states \( q_0q_1q_2\ldots \) such that for every \( i \geq 0 \), \( q_i \in Q \) and \( (q_i, w_{i+1}, q_{i+1}) \in \delta \).

Depending on type of the automaton we have different definitions of accepting run. For a Büchi automaton, a run is accepting, if it visits some state from accepting condition \( \alpha \) infinitely often. In the case of a co-Büchi automaton, a run is accepting, if only states from the set \( \alpha \) are visited infinitely often in this run. For a given non-deterministic \( \omega \)-automaton \( \mathcal{A} \) and a given word \( w \), we say that \( \mathcal{A} \) accepts \( w \) if there exists an accepting run of \( \mathcal{A} \) on \( w \). The words accepted by \( \mathcal{A} \) form the language of \( \mathcal{A} \), denoted by \( L(\mathcal{A}) \).

We say that a co-Büchi automaton \( \mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle \) is in the normal form iff for each \( (q, a, q') \in \delta \) if \( q \) is in \( \alpha \), then also \( q' \) is in \( \alpha \). Note that for a
given NCW \( \mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle \) the automaton \( \mathcal{A}' = \langle \Sigma, Q', \langle q_0, 0 \rangle, \delta', \alpha \times \{1\} \rangle \), where \( Q' = Q \times \{0\} \cup \alpha \times \{1\} \) and \( \delta' = \{ \langle (q,i), a, \langle q', j \rangle \rangle \mid \langle q,a,q' \rangle \in \delta \land i \leq j \land \langle q,i \rangle, \langle q', j \rangle \in Q \} \) is in the normal form, recognizes the same language and has at most \( 2|Q| \) states.

For a given word \( w = w_1, w_2, \ldots \), let \( w[i,j] = w_i, w_{i+1}, \ldots, w_j \) and let \( w[i, \infty] = w_i, w_{i+1}, \ldots \).

An accepting run \( q_0, q_1, \ldots \) on a word \( w = w_1, w_2, \ldots \) of a co-Buchi automaton in the normal form is called shortest, if it reaches an accepting state as early as possible, that is if for each accepting run \( p_0, p_1, \ldots \) of this automaton on \( w \) it holds that if \( p_i \in \alpha \) then also \( q_i \in \alpha \).

\section{Languages \( L_k \) and their automata}

Let \( k \geq 64 \) be a fixed natural number and let \( \mathfrak{A}_k = \{1, 2 \ldots k\} \). The set \( \Sigma_k = \{a, \bar{a} \mid a \in \mathfrak{A}_k\} \) will be the alphabet of our language \( L_k \).

Let us begin with the following informal interpretation of words over \( \Sigma_k \).

Each symbol \( j \in \Sigma_k \) should be read as “agent \( j \) makes a promise”. Each symbol \( j \in \Sigma_k \) should be read as “\( j \) fulfills his promise”. The language \( L_k \) consists (roughly speaking) of the words in which there is someone who at least \( 2k \) times fulfilled his promises, but there are also promises which were never fulfilled.

To be more formal:

\begin{definition}
For a word \( w \in \Sigma_k^\omega \), where \( w = w_1 w_2 \ldots \) and \( i \in \mathcal{N} \), define the interpretation \( h_i(w) \) as:

- \( h_i(w) = 0 \) if \( w_i \in \mathfrak{A}_{2k} \) and \( w_i \) occurs in \( w[i+1, \infty] \); (it is the fulfillment that counts, not a promise).
- \( h_i(w) = 0 \) if \( w_i \in \mathfrak{A}_{2k} \) and \( w_i \) does not occur in \( w[i+1, \infty] \); (unfulfilled promises are read as 0).
- Suppose \( w_i = \bar{s} \) for some \( s \in \mathfrak{A}_{2k} \). Then \( h_i(w) = s \) if there is \( j < i \) such that \( w_j = s \) and \( s \) does not occur in the word \( w[j, i-1] \), and \( h_i(w) = \bar{s} \) if there is no such \( j \) (one first needs to make a promise, in order to fulfill it).

The interpretation \( h(w) \) is now defined as the infinite word \( h_1(w)h_2(w) \ldots \).
\end{definition}

Now we are ready to formally define the language \( L_k \):

\begin{definition}
\( L_k \) is the set of such words \( w \in \Sigma_k^\omega \) that:

- either there is at least one 0 in \( h(w) \) and there exists \( s \in \mathfrak{A}_{2k} \) which occurs at least \( 2k \) times in \( h(w) \),
- or there exists \( i \) such that \( h_i(w) = \bar{s} \) for all \( j > i \).
\end{definition}

It is easy to see that each \( w \in \Sigma_k^\omega \) satisfies at least one of the following three conditions: there is \( s \in \mathfrak{A}_{2k} \) such that \( h_i(w) = s \) for infinitely many numbers \( i \), or there are infinitely many occurrences of 0 in \( h(w) \), or there is only a finite number of occurrences of symbols from \( \mathfrak{A}_{2k} \) in \( w \). Using this observation, we can represent \( L_k \) in the following way:
Fig. 1. The $\omega$-automaton recognizing $L_k$ – all differences between NBW version and NCW version are in the body of $A_i$ (fig. 2). Label $\neg i$ stands (for better readability) for alternative of every label except $i$.

Fig. 2. Automaton $A_i$ – the same for NBW and NCW, modulo the body of $B_{i,j}$ (fig. 3 for NBW and fig. 4 for NCW)
\[ L_k = \{ viw \mid v \in \Sigma_k \land \\
( i \in \mathfrak{A}_{2k} \land w \in (\Sigma_k \setminus \{i\})^\omega \land \exists j \in \mathfrak{A}_{2k} \; h_m(viw) = j \text{ for infinitely many numbers } m ) \] (1) \\
\lor ( i \in \mathfrak{A}_{2k} \land w \in (\Sigma_k \setminus \{i\})^\omega \land \exists j \in \mathfrak{A}_{2k} \; h_m(viw) = j \text{ for at least } 2k \text{ numbers } m \text{ such that } m \leq |v| ) \] (2) \\
\lor ( w \in \{1, \ldots, 2k\}^\omega ) \] (3)

Fig. 3. Automaton \( B_{i,j} \) in the NBW case

Keeping in mind the above representation, it is easy to build a small NBW recognizing \( L_k \) (see Figures 1, 2 and 3). The accepting state on the bottom left of Figure 1 checks if condition (3) is satisfied, and the other states (except of the states in the boxes \( A_i \) and of the initial state) check if the condition (2) is satisfied. Reading the input \( y \) the automaton first guesses the number \( j \) from condition (2) and makes sure that \( j \) occurs at least \( 2k \) times in \( h(y) \). Then it guesses \( i \) from condition (2), accepts, and remains in the accepting state forever, unless it spots \( \bar{i} \). This part of the automaton works also correctly for the co-Büchi case.

The most interesting condition is (1). It is checked in the following way. At first, the automaton waits in the initial state until it spots \( i \) from condition (1). Then, it goes to \( A_i \), guesses \( j \) and goes to the module \( B_{i,j} \), which checks if \( i \) does not occur any more, and if both \( j \) and \( \bar{j} \) occur infinitely often. This can be summarized as:

**Theorem 1.** Language \( L_k \) can be recognized with a nondeterministic Büchi automaton with \( \Theta(k^2) \) states.

Condition (1) cannot be checked by any NCW. However, it can be replaced by the condition
\[ i \in \mathfrak{A}_{2k} \land w \in (\Sigma_k \setminus \{i\})^\omega \land \exists j \in \mathfrak{A}_{2k} \; h_m(viw) = j \text{ for at least } 2k \text{ numbers } m \]
which leads us to a NCW as on Figures 1, 2 and 4. In this case, automaton \( B_{i,j} \) needs to count to \( 2k \), so it needs \( \Theta(k) \) states. Therefore, the whole NCW
automaton has \( \Theta(k^3) \) states. Actually, we believe that every NCW recognizing \( L_k \) indeed needs \( \Theta(k^3) \) states.

Now we are ready to state our main theorem:

**Theorem 2.** Every NCW recognizing \( L_k \) has at least \( k \cdot \frac{k^{4/3}}{4} \) states.

The rest of this paper is devoted to the proof of this theorem. In subsection 2.3 we will define, for each co-Büchi automaton in the normal form, recognizing \( L_k \) a family of \( k \) disjoint sets of states, and in subsection 2.5 we will show that each such set has at least \( \frac{k^{4/3}}{4} \) states. As we have seen in subsection 2.1, for a given NCW with \( n \) states we can always build a NCW in the normal form with at most \( 2n \) states, which finally leads to \( \frac{1}{2} \cdot k \cdot \frac{k^{4/3}}{4} \) lower bound.

### 2.3 The \( k \) disjoint sets of states

Let \( A = \langle \Sigma, Q, q_0, \delta, \alpha \rangle \) be an NCW in the normal form with \( N \) states that recognizes \( L_k \).

Let \( w_{i,j} = i(j^N, 2, \ldots, i-1, i+1, \ldots, 2k)^\omega \). For every \( i \neq j \) let \( q_0, q_{i,j}^1, q_{i,j}^2, q_{i,j}^3, \ldots \) be a fixed shortest accepting run of \( A \) on \( w_{i,j} \).

Words \( w_{i,j} \) will be the main tool in our attempt to fool the automaton if it has too few states so let us comment on their structure. First notice, that the \( i \), the very first symbol of \( w_{i,j} \), will turn into the only 0 in \( h(w) \) – this is, among other reasons, since for all \( m \neq i \) the symbol \( m \) occurs infinitely many times in \( w \). See also that if we replaced the blocks \( j^N \) in the definition of \( w_{i,j} \) by just a single \( j \), then the word would still be in \( L_k \) – since we do not count promises but fulfillments, the remaining \( j \)'s are almost redundant. It is only in the proof Lemma 2(ii) that we will need them. In the rest of the proof we will only be interested in one state of \( A \) per each such block of symbols \( j \). For this reason we define \( block(l) = N + 1 + l(N + 2k - 1) \) as the function that points to the index of the state in run \( q_0, q_{i,j}^1, q_{i,j}^2, q_{i,j}^3, \ldots \) just after reading the \( l \)-th block \( j^N \).

Let \( Q_{i,j} = \{ q_{i,j}^{block(c)} | c \in \mathbb{N} \} \).

**Lemma 1.** For every \( i, j, m, l \in \mathbb{N} \) such that \( m \neq i \neq j \neq l \) and \( m \neq j \), the sets \( Q_{i,m} \) and \( Q_{j,l} \) are disjoint.
Lemma 2. For each huge $i$, $m$, $l \in \mathcal{A}_{2k}$, and $s, t \in \mathcal{N}$ such that $m \neq i \neq j \neq l$, $m \neq j$ and $q_{i,m}^t = q_{j,l}^t$. Let $v = w_{i,m}[0, block(s)], w_{j,l}[block(t) + 1, \infty]$. This word is accepted by $A$, because there exists an accepting run $q_{00}, q_{i,m}, \ldots, q_{l,j}, q_{j,l}, q_{j,l}^t$ of $A$.

The only letters without the overline in $v$ are $i$, $m$, and $l$. However, the only overlined letter that does not occur infinitely often in $v$ is $j$. This letter is different from $i$, $m$, and $l$ because of the assumptions we made. Therefore $0$ does not occur in $h(v)$ and $v \notin L_k$. \hfill \Box

We say that $l$ is huge if $l > k$ and that $l$ is small otherwise.

For every $i$ let $Q_i = \bigcup \{Q_{i,j} | j \text{ is small} \}$. A simple conclusion from Lemma 1 is that for each huge $i$ $j$ such that $i \neq j$ the sets $Q_i$ and $Q_j$ are disjoint. This implies, that Theorem 2 will be proved, once we prove the following lemma:

Lemma 2. For each huge $i \in \mathcal{A}_{2k}$ the size of the set $Q_i$ is greater than $\frac{k^{4/3}}{4}$.

2.4 Combinatorial lemma

The $n \times m$ state matrix is a two-dimensional matrix with $n$ rows and $m$ columns. We say that $n \times m$ state matrix is $l$-painted if each of its cells is labeled with one of $l$ colors and the minimal distance between two cells in the same row and of the same color is at least $m$.

For a painted $n \times m$ state matrix, we say that an $M_{i,j}$ is a cell on the left border if $j = 1$, and is on the right border if $j = m$. We say that $M_{i,j}$ is a successor of $M_{i',j'}$ if $i = i'$ and $j = j' + 1$.

The path $w$ through a painted $n \times m$ state matrix $M$ is a sequence of cells $c_1, c_2, \ldots, c_z$ such that $c_1$ is on the left border, $c_z$ is on the right border, and for each $s < z$ either $c_{s+1}$ is a successor of $c_s$ (we say that “there is a right move from $c_s$ to $c_{s+1}$”) or $c_s$ and $c_{s+1}$ are of the same color (we say that “there is a jump from $c_s$ to $c_{s+1}$”)

We say that a path $w$ is good, if there are no consecutive $k$ right moves in $w$, and no jump leads to (a cell in) a row that was already visited by this path. Notice that in particular a good path visits at most $k$ cells in any row.

Our main combinatorial tool will be:

Lemma 3. Let $M$ be an $[\frac{k^{4/3}}{4}]$-painted $k \times [\frac{k^{4/3}}{4}]$ state matrix. Then there exists a good path on $M$.

The proof of this lemma is left to subsection 2.6.

2.5 From automaton to state matrix

We are now going to prove Lemma 2. Let a huge $i \in \mathcal{A}_{2k}$ be fixed in this subsection and assume that $|Q_i| < \frac{k^{4/3}}{4}$. We will show that there exists a word $w$ such that $A$ accepts $w$ and no agent fulfills its promises at least $2k$ times in $w$.

Let $j$ be an small number from $\mathcal{A}_{2k}$. Let us begin from some basic facts about $Q_{i,j}$:
Lemma 4. (i) There exists a number \( l \) such that for every \( s < l \) the state \( q_{i,j}^{\text{block}(s)} \) is not in \( \alpha \) and for every \( s \geq l \) the state \( q_{i,j}^{\text{block}(s)} \) is in \( \alpha \). Define \( \text{acc}(i,j) = l \).

(ii) No accepting state from \( Q \), can be reached on any run of \( A \) before some agent fulfilled its provisos \( 2k - 1 \) times. It also implies that \( \text{acc}(i,j) \geq 2k - 1 \).

(iii) The states \( q_{i,j}^{\text{block}(0)}, q_{i,j}^{\text{block}(1)}, \ldots, q_{i,j}^{\text{block}(\text{acc}(i,j))} \) are pairwise different.

Proof. (i) This is since \( A \) is in the normal form.

(ii) While reading a block of \( N \) symbols \( j \), the automaton is in \( N + 1 \) states, so there is a state visited at least twice. If this state was accepting, then a pumping argument would be possible – we could simply replace the suffix of the word after this block with the word \( j^\omega \) and the new word would still be accepted, despite the fact that it is not in \( L_k \).

(iii) Suppose \( q_{i,j}^{\text{block}(s)} \) and \( q_{i,j}^{\text{block}(t)} \) are equal and non-accepting. For every \( s < t \leq \text{acc}(i,j) \), the words \( w_{i,j}[\text{block}(s) + 1, \infty] \) and \( w_{i,j}[\text{block}(t) + 1, \infty] \) are identical. Then a pumping argument works again – we can find a shorter accepting run by pumping out the states \( q_{i,j}^{\text{block}(s)}, \ldots, q_{i,j}^{\text{block}(t)} \). But this contradicts the assumption that our run is shortest.

\( \square \)

We want to show that \( |Q| \geq \frac{k^{4/3}}{4} \). If for any small \( j \) there is \( \text{acc}(i,j) \geq \frac{k^{4/3}}{4} \), then, thanks to Lemma 4(iii) we are done. So, for the rest of this subsection, we assume that \( \text{acc}(i,j) < \frac{2k}{3} - 1 \) for each small \( j \).

We will now construct a \( \lfloor \frac{k^{4/3}}{4} \rfloor \) - painted \( k \times \lfloor \frac{k^{4/3}}{4} \rfloor \) state matrix \( M \) in such a way, that its \( m \)th row will, in a sense, represent the accepting run on the word \( w_{i,m} \). More precisely, take a \( k \times \lfloor \frac{k^{4/3}}{4} \rfloor \) matrix \( M \) and call the cells \( M_{m,j} \) of \( M \), where \( j \leq \text{acc}(i,m) \), \( \text{real cells} \) and call the cells \( M_{m,j} \) of \( M \) with \( j > \text{acc}(i,m) \) \( \text{ghosts} \). For a ghost cell \( M_{m,j} \) and the smallest natural number \( l \) such that \( j - lk \leq \text{acc}(i,m) \) call the real cell \( M_{m,j - lk} \) the host of \( M_{m,j} \). Notice that each ghost has its host, since, by Lemma 4(ii), \( \text{acc}(i,m) \geq 2k - 1 \), which means that there are at least \( k \) real cells in each row.

If \( M_{m,j} \) is real then define its color as \( q_{i,m}^{\text{block}(j-1)} \). If \( M_{m,j} \) is a ghost then define its color as the color of its host. Now see that \( M \) is indeed a \( \lfloor \frac{k^{4/3}}{4} \rfloor \) - painted \( k \times \lfloor \frac{k^{4/3}}{4} \rfloor \) state matrix – the condition concerning the shortest distance between cells of the same color in the same row of \( M \) is now satisfied by Lemma 4(iii) and the condition concerning the number of colors is satisfied, since we assume that \( |Q| \leq \frac{k^{4/3}}{4} \).

By Lemma 4 we know that there is a good path in \( M \). This means that Lemma 2 will be proved once we show:

Lemma 5. If there exists a good path in \( M \), then there exists a word \( w \notin L_k \) such that \( w \) is accepted by \( A \).

Proof. Suppose \( r \) is a good path in \( M \) and \( c \) is the first ghost cell on \( r \). Let \( c' \) be the direct predecessor of \( c \) on \( r \). If the move from \( c' \) to \( c \) was a right move then define a new path \( p \) as the prefix of \( r \) ending with \( c \). If the move from \( c' \)
to \( c \) was a jump, then suppose \( c'' \) is the host of \( c \), and define \( p \) as the following path: first take the prefix of \( r \) ending with \( c' \). Then jump to \( c'' \) (it is possible, since the color of a ghost is the color of its host). Then make at most \( k - 1 \) right moves to the last real cell in this row.

It is easy to see that \( p \) satisfies all the conditions defining a good path, except that it does not reach the right border of \( M \).

Let \( p \) be a concatenation of words \( p_1, p_2, \ldots, p_z \), such that each move between \( p_x \) and \( p_{x+1} \) is a jump but there are no jumps inside any of \( p_x \). This means that each \( p_x \) is contained in some row of \( M \), let \( \beta(x) \) be a number of this row. This also means, since \( p \) is (almost) a good path, that \( |p_x| \leq k \) for each \( x \).

Let \( v_1 = \overline{1}, \overline{2}, \ldots, \overline{1-T}, \overline{T+1}, \ldots, \overline{2k} \). Now define an infinite word \( w \) as follows:

\[
w = i\beta(1)^N(v_{i\beta(1)^N})^{p_{i-1}}(v_{i\beta(2)^N})^{p_{i-1}} \ldots (v_{i\beta(z)^N})^{p_{i-1}}\beta(z)^\omega
\]

To see that \( w \not\in L_k \) notice, that a symbol \( s \in \mathbb{A}_k \) occurs in \( h(w) \) only if \( s = \beta(x) \) for some \( x \in \{1, 2, \ldots, z\} \) and that it occurs at most \( |p_x| + 1 \leq k \) times in \( w \). The fact that \( A \) accepts \( w \) follows from the construction of path \( p \) and from Lemma 3(ii).

\[\square\]

### 2.6 Proof of the combinatorial lemma

Let \( n = \lfloor \frac{k^{4/3}}{4} \rfloor \) and \( M \) be an \( n \)-painted \( k \times n \) state matrix. We split the matrix \( M \) into matrices \( M^0, M^1, \ldots, M^{\lceil \frac{2n}{k} \rceil - 1} \), each of them of \( k \) rows and each of them (possibly except of the last one) of \( \frac{n}{k} \) columns, such that \( M^i \) contains columns \( i\frac{k}{2} + 1, i\frac{k}{2} + 2 \ldots, \min(i\frac{k}{2} + \frac{k}{2}, n) \). The matrices \( M^0, M^1, \ldots, M^{\lceil \frac{2n}{k} \rceil - 1} \) will be called *multicolumns*.

We are going to build a path \( w = c_1c_2 \ldots c_z \) through \( M \) satisfying the following:

- if \( w \) has a jump from \( c_j \) to \( c_{j+1} \) then both \( c_j \) and \( c_{j+1} \) belong to the same multicolumn;
- \( w \) has exactly \( \lceil \frac{2n}{k} \rceil - 1 \) jumps, one in each multicolumn;
- no jump on \( w \) leads to a previously visited row of \( M \).

Clearly, such a path will be a good path. This is since the width of each multicolumn is \( \frac{k}{2} \), and each sequence of consecutive right moves on \( w \) will be contained in two adjacent multicolumns (except of the last such sequence, which is contained in the last multicolumn and \( M^{\lceil \frac{2n}{k} \rceil - 1} \)).

Let \( s = \frac{k^{4/3}}{2} \). Since \( [s] = \lceil \frac{2k^{4/3}}{k} \rceil \geq \lceil \frac{2n}{k} \rceil \), the number \( [s] - 1 \) is not smaller than the number of jumps we want to make.

Now we concentrate on a single multicolumn \( M^i \), which is a matrix with \( k \) rows and with \( \frac{k}{2} \) columns. We will call two rows of such a multicolumn *brothers* if at least one cell of one of those rows is of the same color as at least one cell of another (i.e. two rows are brothers if a path through \( M^i \) can make a jump between them).
Suppose some of the rows of the multicolumn $M_i$ belong to some set $D_i$ of dirty rows. The rows which are not dirty will be called clean. A color will be called clean if it occurs in some of the clean rows. A row will be called poor if it has less than $\lceil s \rceil$ clean brothers. One needs to take care here in the following procedure, while more rows will get dirty, more rows will also get poor:

**Procedure** (Contaminate a single multicolumn($D_i,M_i$))

while there are clean poor rows (with respect to the current set $D_i$ of dirty rows) in $M_i$, select any clean poor row and all his brothers, and make them dirty (changing $D_i$ accordingly).

end of procedure

We would like to know how many new dirty rows can be produced as a result of an execution of the above procedure.

Each execution of the body of the while loop makes dirty at most $\lceil s \rceil$ rows and decreases the number of clean colors by at least $\frac{k}{2}$ – none of the colors of the selected clean poor row remains clean after the body of the while loop is executed. Since there are at most $n$ colors in the multicolumn (as $M$ is $n$-colored), the body of the while loop can be executed at most $\frac{nk}{2} = \lceil s \rceil$ times, which means that at most $\lceil s \rceil^2$ new dirty rows can be produced.

Notice that after an execution of the procedure, none of the clean rows is poor.

Now we are ready for the next step:

**Procedure** (Contaminate all multicolumns)

Let $D_{\lceil \frac{n}{2} \rceil - 1} = \emptyset$;

for $i = \lceil \frac{n}{2} \rceil - 2$ down to 0

Let $D_i = D_i+1$;

Contaminate a single multicolumn($D_i,M_i$);

end of procedure

We used a convention here, that a set $D_i$ of rows is identified with the set of numbers of these rows. Thanks to that we could write the first line of the above procedure, saying “consider the dirty rows of $M^{i+1}$ to be also dirty in $M^i$”.

Suppose $D_0, D_1, \ldots, D_{\lceil \frac{n}{2} \rceil - 2}$ are sets of dirty rows in multicolumns $M_0, M_1, \ldots, M_{\lceil \frac{n}{2} \rceil - 2}$ resulting from an execution of the procedure Contaminate all multicolumns. Notice, that for each $0 \leq i \leq \lceil \frac{2n}{3} \rceil - 2$ the inclusion $D^{i+1} \subseteq D^i$ holds. In other words, if a row is clean in $M^i$, then it is also clean in $M^{i+1}$.

The following lemma explains why clean rows are of interest for us:

**Lemma 6.** Suppose $w = c_1c_2 \ldots c_z$ is a path through the matrix consisting of the first $i$ multicolumns of $M$ (or, in other words, of the first $\frac{ni}{2}$ columns of $M$). Suppose (i) $w$ has exactly one jump in each multicolumn, and each jump leads to a row which was not visited before, (ii) if there is a jump from $c_j$ to $c_{j+1}$ then both $c_j$ and $c_{j+1}$ belong to the same multicolumn. Suppose finally, that (iii) the cell where $w$ reaches the right border of the matrix, belongs to a clean row $r$. Then $w$ can be extended to a path through the matrix consisting of the first $i+1$ multicolumns of $M$, in such a way that this extended path will also satisfy conditions (i)-(iii).
Proof. The only thing that needs to be proved is that one can jump, in multi-
column $M^i$, from row $r$ to some clean row which was not visited before. Since,
by assumption, $r$ was clean in $M^{i-1}$, it is also clean in $M^i$. Since there are no
clean poor rows in $M^i$, we know that $r$ has at least $[s]$ clean brothers. At most
$i$ of them were visited so far by the path, where of course $i \leq [s] - 1$.

Now, starting from an empty path and a clean row in $M^0$ and using the above
lemma $[2^{n+1}] - 2$ times we can construct a path $w$ as described in the beginning
of this subsection and finish the proof of Lemma. The only lemma we still need
for that is:

\begin{lemma}
|D^0| < k. In other words, there are clean rows in $M^0$.
\end{lemma}
\begin{proof}
Let $l = [s] - 2$ be the index of the last multicolumn. The number of dirty
rows in $D^{l-i}$ can be bounded by $(i + 1) \cdot [s]^2$ because of observations about
defined procedures. For $i = l$, we have $([s] - 1) \cdot [s]^2$, what is not greater then
$s(s + 1)^2 = \frac{k^{1/3}}{2} (\frac{k^{1/3}}{2} + 1)^2$ which is, finally, less then $k$, because $k \geq 8$.
\end{proof}

References
1. B. Aminof and O. Kupferman and O. Lev. On the Relative Succinctness of Non-
deterministic Büchi and co-Büchi Word Automata. In In Proc. of the 15th Int.
Conf. on Logic for Programming, Artificial Intelligence, and Reasoning, LNCS
5330, pages 183–197. Springer, 2008.
2. J.R. Büchi. On a decision method in restricted second order arithmetic. In Proc.
Int. Congress on Logic, Method, and Philosophy of Science. 1960, pages 1–12.
Stanford University Press, 1962.
3. O. Kupferman. Tightening the exchange rate between automata. In Proc. 16th An-
nual Conf. of the European Association for Computer Science Logic, LNCS 4646,
pages 7–22, 2007.
4. O. Kupferman, G. Morgenstern, and A. Murano. Typeness for $\omega$-regular automata.
In 2nd Int. Symp. on Automated Technology for Verification and Analysis, LNCS
3299, pages 324–338. Springer, 2004.
5. O. Kupferman, M. Vardi. Weak Alternating Automata Are Not That Weak. In Proceedings of the Fifth Israel Symposium on the theory of Computing Systems
(ISTCS ’97) (June 17 - 19, 1997). ISTCS. IEEE Computer Society, Washington,
D C, 147.
6. R. McNaughton. Testing and generating infinite sequences by a finite automaton.
Information and Control, 9:521–530, 1966.
7. M.O. Rabin. Decidability of second order theories and automata on infinite trees.
Transactions of the AMS, 141:1–35, 1969.
8. S. Safra. On the complexity of $\omega$-automata. In Proc. 29th IEEE Symp. on Foun-
dations of Computer Science, pages 319–327, 1988.