Theoretical Computer Science 629 (2016) 64–79

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Recognizing 3-collapsing words over a binary alphabet ☆

Alessandra Cherubini [a], Andrzej Kisielewicz [b,*]

[a] Dipartimento di Matematica, Politecnico di Milano, Italy
[b] Department of Mathematics and Computer Science, University of Wrocław, Poland

ARTICLE INFO

Article history:
Received 27 January 2015
Received in revised form 30 September 2015
Accepted 29 October 2015
Available online 6 November 2015

Keywords:
k-Collapsing words
k-Compressible automata
Permutation conditions

ABSTRACT

A finite deterministic automaton $\mathcal{A} = (Q, \Sigma, \delta)$ is k-compressible if there is a word $w \in \Sigma^+$ such that the image of the state set Q under the natural action of w is reduced by at least k states. In such case w is called a k-compressing word for $\mathcal{A}$. It is known that, for any alphabet Σ and any $k \geq 2$, there exist words that are k-compressing for each k-compressible automaton with the input alphabet Σ. Such words are called k-collapsing. It has been proved that recognizing 2-collapsing words over a 2-element alphabet may be done in polynomial time, while recognizing 2-collapsing words over an alphabet of size $n \geq 3$ is co-NP-complete. The computational complexity of recognizing k-collapsing words with $k \geq 3$ over an n-element alphabet has remained open. In this paper we prove that this remaining problem is co-NP-complete in the simplest case with $k = 3$ and $n = 2$.

1. Introduction

Let $\mathcal{A} = (Q, \Sigma, \delta)$ be a complete deterministic automaton with the set of the states Q, the input alphabet Σ, and the transition function $\delta : Q \times \Sigma \to Q$. For each $\alpha \in \Sigma$, the function δ induces a transformation on Q, which we denote $q\alpha = \delta(q, \alpha)$, for all $q \in Q$. This action extends naturally into the action of words $w \in \Sigma^+$ on Q denoted by $qw = \delta(q, w)$. We are interested in the images $Qw = \{s \in Q : s = qw \text{ for some } q \in Q\}$ for various words w.

If for a word $w \in \Sigma^+$, $|Qw| = 1$, i.e. the image consists of a single state, then the word w is called a *synchronizing* (or *reset*) word for $\mathcal{A}$, and $\mathcal{A}$ itself, in such a case, is called a *synchronizing automaton*. Reset words and synchronizing automata have been intensively studied from the very beginning of the theory of finite automata. In particular, a lot of research has been connected with the Černý conjecture, which states that for each synchronizing automaton $\mathcal{A}$ with n states there exists a reset word of length at most $(n-1)^2$. The conjecture is still open in its generality, and is considered one of the most longstanding open problems in automata theory (see [11] for general motivation and the excellent survey of the field).

One of the directions of research in this area focuses on k-compressing words. Given $k \geq 1$, a word $w \in \Sigma^+$ is called a *k-compressing* word for $\mathcal{A}$, if $|Q \setminus Qw| \geq k$. In such a case, we say also that w *k-compresses* $\mathcal{A}$. Clearly, if $|Q| = n$, then w is synchronizing if and only if it is $(n-1)$-compressing for $\mathcal{A}$. If $\mathcal{A}$ has a k-compressing word, then $\mathcal{A}$ itself is called *k-compressible*. Given an alphabet Σ, a word $w \in \Sigma^+$ that is k-compressing for each k-compressible automaton over the input alphabet Σ is called *k-collapsing* (over Σ). The original motivations for the interest in such words come from algebra and combinatorics (see, e.g., [1]). In [10], using algebraic terminology, it is proved that for each positive integer k, and

☆ The paper is an extended version of presentation [7]. The research of the first author is supported in part by PRIN: "Automi e linguaggi formali: aspetti matematici e applicativi". The research of the second author is supported in part by the Polish MNiSzW grant IP 2012 052272.

* Corresponding author.
E-mail addresses: alessandra.cherubini@polimi.it (A. Cherubini), andrzej.kisielewicz@math.uni.wroc.pl (A. Kisielewicz).

http://dx.doi.org/10.1016/j.tcs.2015.10.045
0304-3975/

each finite alphabet Σ, there exists a k-collapsing word over Σ by exhibiting an effective construction of such words. In automata theory, the interest in such words comes from the fact that they can be seen as universal testers whose action on the set of states of an automaton exposes whether or not the automaton is k-compressible. Recognizing k-collapsing words is, however, not easy (except for trivial case $k = 1$).

In [8], it is proved that the membership of a given word $w \in \Sigma^+$ in the language of k-collapsing words is decidable, for any fixed $k \geq 1$. The decision procedure is in the class co-NP and requires linear space, which shows that the language of k-collapsing words is context-sensitive. In [9], it is shown that it is not context-free even in the simplest nontrivial case of the language of 2-collapsing words over a 2-letter alphabet. Most results so far concern 2-collapsing words. In particular, 2-collapsing words (over an alphabet of arbitrary size) have been characterized in [1], and in [5]. From the first characterization in [1] (that has a group theoretical flavor), a non-deterministic polynomial time algorithm to recognize whether a word $w \in \Sigma^+$ is 2-collapsing was derived in [2]. A refinement of this algorithm was used in [3] to give the list of the shortest 2-collapsing words on a 3-letter alphabet. The second characterization in [5] (which is in terms of solving systems of permutation conditions) is used in [6] to show that the recognizing 2-collapsing words over an alphabet of size ≥ 3 is co-NP-complete. On the other hand, recognizing 2-collapsing words over a 2-element alphabet may be done in polynomial time [5]. In view of these two results a question arises whether for $k \geq 3$, k-collapsing words over a 2-element alphabet can also be recognized in polynomial time. This natural question has remained open so far. In this paper we show that the answer is negative (assuming $NP \neq P$). We prove that the problem of deciding whether a word $w \in \{\alpha, \beta\}^+$ is 3-collapsing is co-NP-complete.

The difficulty in this case is that we have no characterization of 3-collapsing words similar to those for the case of 2-collapsing words. Yet, we have a natural classification of the 3-compressible automata on two input letters [4], and we can see that the problem looks different in different classes. For some classes it is easy to decide whether a word w 3-compresses all the automata in the class. There is however at least one class, where this problem leads to solving a system of transformation conditions, similar to permutation conditions in [6], and it seems that similar methods could be used to prove its computational hardness. So, our idea is to restrict the latter problem concerning solving transformation conditions, so that on the one hand, we could prove its NP-hardness, and on the other hand, we could still transform it back to our original problem.

This restriction is done in a few steps in Section 2. As a result we formulate a certain technical computational problem (*) concerning solving transformation conditions, and demonstrate that it reduces in polynomial time to the problem of deciding whether a given word w over a 2-element alphabet is 3-collapsing. The second part of the paper, Section 3, is devoted to prove that the problem (*) is NP-hard. Here we try to apply the method of the proof in [6]. Yet, since now we consider systems of transformation conditions, rather then permutation conditions, we need substantial modifications and new ideas to achieve the aim.

We note that our result does not settle the case of k-collapsing words over n-element alphabets for any pair with $k, n > 2$. Of course, our results allow to conjecture that in each such case the problem is co-NP-hard all the more, but we notice that our proof does not admit any simple extension to the case of larger k or n.

2. Three-compressible automata

In this section we establish facts on 3-compressible automata allowing us to reduce our task to proving NP-completeness of a certain problem concerning solving a system of transformation conditions.

This reduction is done in a few steps. First, in Subsection 2.1, we introduce some special terminology and provide a simple classification of 3-compressible automata over a 2-element alphabet. In Subsection 2.2, we distinguish one of the classes, and give a characterization of words 3-compressing all automata in this class in terms of solving a system of transformation conditions. Next, in Subsection 2.3, we restrict our consideration to the smallest class $\mathcal{D}$, where the characterization obtained in the previous subsection has a relatively simple formulation. Then, using methods and results of [4], we find a single word $w_{\mathcal{D}}$ that 3-compresses all 3-compressible automata (over 2 letters) not in the class $\mathcal{D}$, which is used to transform the problem to decide whether a word 3-compresses all automata in $\mathcal{D}$ to the problem to decide whether a word is 3-collapsing. In Subsection 2.4 we formulate the technical computational problem (*) concerning solving transformation conditions, and prove that it reduces in polynomial time to the problem of deciding whether a given word over a 2-element alphabet is not 3-collapsing.

2.1. Preliminaries

We deal with automata over a 2-element alphabet $\Sigma = \{\alpha, \beta\}$. The letters α and β are identified with transformations they induce on the set Q of the states. The image of $q \in Q$ by a letter (transformation) α is denoted $q\alpha$, and $q\alpha^{-1}$ denotes the inverse image of q by α. Each word $w = \alpha_1\alpha_2 \dots \alpha_t$ over Σ is identified with the transformation it induces.

We use a special notation for concrete transformations α (similar to well-known permutation notation), where round brackets $(x_1x_2 \dots x_t)$ denote a cycle: $x_1\alpha = x_2, \dots, x_{t-1}\alpha = x_t, x_t\alpha = x_1$, and square brackets $[x_1x_2 \dots x_t]$ denote a path: $x_1\alpha = x_2, \dots, x_{t-1}\alpha = x_t$. This notation is not unique: for example $[123][42](357) = [12][423](357)$. Yet, we always write full cycles, and refer to them as the *cycles* of α, and usually omit all the fixed points, i.e. the cycles of length 1. If x is an element of a cycle in α, then we write $Cyc_\alpha(x)$ to denote the cycle containing x, or the set of elements in this cycle, and