On the Maximality of Languages with Combined Types of Code Properties

Lila Kari[a], Stavros Konstantinidis[b], Steffen Kopecki[a]

[a] *The University of Western Ontario, London, Ontario, Canada*
[b] *St. Mary's University, Halifax, Nova Scotia, Canada*

Abstract

We consider the decision problem of whether or not a given regular language is maximal with respect to certain *combined* types of code properties. In the recent past, there have been a few formal methods for defining code properties, such as the trajectory-based type of codes and the transducer-based type of codes, that allow one to decide the maximality problem, including the case where maximality is tested with respect to combined properties *within* these formal methods. The property of "deciphering delay 1", also known as decoding delay 1, is not known to be definable with these methods, but it is known that the maximality of this property alone is decidable for regular languages. Here, we consider the problem of deciding maximality of a regular language with respect to deciphering delay 1 *and* a transducer-based property, such as suffix code, overlap-free language, and error-detection properties.

Keywords: code properties, deciphering delay, decoding delay, decidable, maximal codes, syntactic monoids

1. Introduction

The problem of deciding whether or not a given finite, or regular, language L is maximal with respect to a certain code property has been investigated for various fixed properties, such as prefix codes [1], bifix codes [2, 3], infix codes [4], and finite deciphering delay codes [5]. The notion of maximality of a language L with respect to a certain property $\mathcal{P}$ means that L has property $\mathcal{P}$, but no language that strictly contains L has property $\mathcal{P}$. In fact, for most of these properties, the more general embedding (or completion) problem has been solved, that is, the problem of effectively constructing a maximal finite, or regular, language containing L. We note that the property "deciphering delay" is often referred to as "decoding delay" [6], or even "verbal deciphering delay"; in this paper, we consistently call the property "deciphering delay".

Email addresses: lila@csd.uwo.ca (Lila Kari), s.konstantinidis@smu.ca (Stavros Konstantinidis), steffen@csd.uwo.ca (Steffen Kopecki)

In the recent past, there have been a few formal methods for defining code properties in the sense that a code property is described via a certain formal expression [7–9]. Some of these methods allow one to decide the *maximality problem* for regular languages, that is, the problem of whether or not a given regular language L is maximal with respect to a given property $\mathcal{P}$ definable within these formal methods. More specifically in [8] trajectory-based type of code properties were defined: a code property $\mathcal{P}_{\bar{e}}$ is defined via a trajectory-set expression $\bar{e}$, that is a regular expression $\bar{e}$ over $\{0,1\}$ which can be used to effectively provide the property as input to a decision algorithm. The method of [9] considers transducer-based type properties: a code property $\mathcal{P}_{\mathbf{t}}$ is defined via a certain kind of transducer $\mathbf{t}$ which can be used to effectively provide the property as input to a decision algorithm. We note that the transducer-based approach stems from the ideas in [10] where a code property is defined via a language equation involving a word operation.

The formal methods which define types of code properties in [8] and [9] can also decide the maximality problem of two (or more) *combined properties* provided that both of these properties have the same type; that is, they are definable within the same formal method (trajectory-based or transducer-based). In this work, we make a first attempt to decide the maximality problem for combined *types* of properties, where one of the properties is not known to be definable as trajectory-based or transducer-based code. More specifically, we consider the decidability of maximality of a given regular language L with respect to the "deciphering delay 1" property *and* a property $\mathcal{P}_{\mathbf{t}}$, where $\mathbf{t}$ is any given *input-preserving transducer*. Our approach also allows one to decide maximality of these combined code properties within a fixed regular language M that we refer to as *cover language* M. With respect to a certain property, maximality of L within a cover language M is obtained when, among all subsets of M, L is a maximal language. This type of situation has been considered several times in the past [9–13]. We note that ordinary maximality (over an alphabet Σ) is obtained when $M = \Sigma^*$.

We renamed the "maximum language" to "cover language".

An input-preserving transducer $\mathbf{t}$ is a formalism which defines a relation over words such that $u \in \mathbf{t}(u)$ for all words u in the domain of $\mathbf{t}$. The (code) property $\mathcal{P}_{\mathbf{t}}$, defined by $\mathbf{t}$ is satisfied by all languages L which do not contain two distinct words u, v such that $v \in \mathbf{t}(u)$. We note that input-preserving transducers can be used to define many known (code) properties such as prefix, suffix, bifix, outfix, infix, hypercode, overlap-free, and error-detection properties [9]. We also note that the problem of whether or not a given regular language L is maximal with deciphering delay d, for any non-negative integer d, has already been solved in [5] using deep results from the theory of codes; see [1, 14] for a more recent presentation. However, our objective here is to investigate the decidability of maximality of L with respect to the deciphering delay 1 property *and* some input-preserving transducer property $\mathcal{P}_{\mathbf{t}}$. Finally, we note that every trajectory property is also an input-preserving transducer property [9].

Ref [14] added.

A code L with *deciphering delay* d (DD-d) has the property that when decoding a message w and after reading a prefix $u_0u_1 \cdots u_d$ of w with $u_i \in L$, the decoding of w has to start with u_0. The codes satisfying DD-0 are the prefix

codes, a property that can be defined by an input-preserving transducer. The property DD-1, however, is not known to be definable by an input-preserving transducer. Note that every property $\mathcal{P}_\mathbf{t}$ which defines a subclass of the prefix codes, such as infix, bifix, and hypercodes, also defines a subclass of DD-1 codes; furthermore, for such a property, a code is maximal with respect to $\mathcal{P}_\mathbf{t}$ and DD-1 if and only if it is maximal with respect to $\mathcal{P}_\mathbf{t}$. Properties that are of interest in our context include suffix code (SC), overlap-freeness (OF: no two distinct words $ux, xv \in L$ with u, v, x non-empty), or k-substitution error detecting (SUB-k: no two distinct words in L with Hamming distance $\leq k$). Note that OF and SUB-k do not imply the code property.

Example 1. The code $L_1 = 00 + 01^+$ has the combined property DD-1 and SC, but it is not maximal with the combined property DD-1 and SC as the word 0010 can be added to L_1.

Example 2. The code $L_2 = 00 + 11011$ is not maximal with respect to DD-1, as 11 can be added, nor with respect to OF, as 1 can be added. However, L_2 is maximal with respect to the combined property DD-1 and OF: If a word $w \notin L_2$ is added to the language and $|w| \geq 2$, then its first letter overlaps with the last letter of 00 or 11011. If $w = 0$ or $w = 1$, the language $L_2 \cup w$ does not satisfy DD-1 since ww is a prefix of 00 or 11011.

Example 3. The code $L_3 = (00)^+1^+ + 01$ is neither maximal within $M = 0^+1^+ + 1^+0^+$ with respect to DD-1, as 011 can be added, nor with respect to SUB-1, as 10 can be added. However, L_3 is maximal within M with respect to the combined property DD-1 and SUB-1: A word $w = 1^i0^j \notin L_3$ for $i, j \geq 1$ cannot be added to L_3 without violating DD-1 because $(001)(1^i0^j) \in L_3w$ is a prefix of $(001^{i+1})(0^{2j}1) \in L_3^2$. A word $w \in 0^i1^j$ with $i, j \geq 1$ which does not belong to L_3 (i. e., $i = 2k + 1$ for $k \in \mathbb{N}$) cannot be added to L without violating SUB-1. Indeed, if $k \geq 1$, then w has Hamming distance 1 to the word $(00)^k1^{j+1} \in L_3$; otherwise, $i = 1$, $j \geq 2$, and w has Hamming distance 1 to $001^{j-1} \in L_3$.

Our approach to answering the desired decision question, for a given regular language L, a regular cover language M, and transducer property $\mathcal{P}_\mathbf{t}$, reduces to whether or not a certain system of *seven language equations* (corresponding to the deciphering delay 1 aspect of maximality) has a solution that belongs to $R_\mathbf{t} \cap M$ (a regular language corresponding to the $\mathcal{P}_\mathbf{t}$-property aspect of maximality). In effect, this question is equivalent to whether the intersection of nine, not necessarily regular, languages is empty or not:

$$L_1 \cap L_2 \cap L_3 \cap L_4 \cap L_5 \cap L_6 \cap L_7 \cap R_\mathbf{t} \cap M \stackrel{?}{=} \emptyset$$

where each L_i is the solution set of the i-th equation; see Section 3. This question turns out to be non-trivial. Our approach to answering this question is as follows.

1.) We solve explicitly four of the equations with solutions L_1, L_3, L_5, L_6, showing that these are effectively regular languages, and we give a clear characterization of the non-regular language L_7.

2.) We show that $L_3 \cap L_4 = L_3$ and that there is, effectively, a regular language $L_{2.1}$ such that $L_1 \cap L_2 = L_1 \cap L_{2.1}$. Thus, we have that the language

$$K = L_1 \cap L_{2.1} \cap L_3 \cap L_5 \cap L_6$$

is an effectively regular language.

3.) We show that the emptiness problem

$$K \cap L_7 \cap R \stackrel{?}{=} \emptyset$$

where R is an arbitrary regular language, is equivalent to the emptiness of a regular language that is computable when L is effectively given.

We also show that deciding if $K \cap L_7 = \emptyset$ is equivalent to deciding if $K = \emptyset$, which is decidable as K is effectively regular. It was proven first in [5] that it is decidable whether or not a regular language is maximal with deciphering delay 1. Our approach provides an alternate method of deciding maximality of deciphering delay 1 for regular languages.

While the system of equations we consider is quite specific, we believe that the reductions mentioned in 2.) and 3.) could lead to similar future reductions in attempting to decide solution existence of similar systems of equations. The tools we use to establish the above results involve concepts from combinatorics on words and the theory of syntactic monoids.

We note that, contrary to the fact that most natural problems about *fixed* properties of regular languages are decidable, one cannot say the same about decision problems involving *types* of properties as opposed to fixed properties. For example, in [15] the authors show that if properties are described via multiple sets of trajectories, then already the satisfaction problem (given a regular language and such a property, does the language satisfy the property?) is undecidable.

The paper is organized as follows. In the next section, we introduce some basic notation and background about languages, finite monoids, and combinatorics on words. In Section 3, we phrase the maximality decision problem with respect to the deciphering delay 1 property as a solution existence problem for a system of seven language equations, which is then reduced to various emptiness problems of the intersection of certain languages. In Section 4, we address the maximality problem with respect to the combined types of properties mentioned above, by considering the concept of solution existence *with a restriction*, and we obtain the desired decidability result. Finally, Section 5 contains a few concluding remarks.

2. Notation and Preliminaries

We assume the reader to be familiar with the fundamental concepts of language theory; see e. g., [16, 17]. Let Σ be a finite set of *letters*, the *alphabet*; Σ^* be the set of all words over Σ; and ε denote the *empty word*. A subset L of Σ^* is a *language* over Σ. The *complement* L^c of L is the language of all words that are not in L. For a length bound $m \in \mathbb{N}$ we let $\Sigma^{\leq m}$ denote the set of words whose length is at most m, i. e., $\Sigma^{\leq m} = \bigcup_{i \leq m} \Sigma^i$. Analogously, we define $\Sigma^{<m} = \bigcup_{i<m} \Sigma^i$, $\Sigma^{\geq m} = \bigcup_{i \geq m} \Sigma^i$, and $\Sigma^{>m} = \bigcup_{i>m} \Sigma^i$. Throughout this paper, we consider languages over the fixed alphabet Σ only. Because the investigation of codes over unary alphabets is trivial, we presume $|\Sigma| \geq 2$.

Let $w \in \Sigma^*$ be a word. Unless confusion arises, by w we also denote the singleton language $\{w\}$. The length of w is denoted by $|w|$. If $w = xyz$ for some $x, y, z \in \Sigma^*$, then x, y, and z are called *prefix*, *infix* (or *factor*), and *suffix* of w, respectively. If $x \neq w$ (resp., $y \neq w$ or $z \neq w$) it is called a *proper prefix* (resp., *proper infix* or *proper suffix*) of w. By $x \leq_p w$ (resp., $x <_p w$) we denote the prefix relationship (resp., proper prefix relationship). For a language $L \subseteq \Sigma^*$, the set $\mathrm{Pref}(L) = \{x \in \Sigma^* \mid \exists y \in \Sigma^* \colon xy \in L\}$ denotes the language containing all prefixes of words in L. The language L is said to be *prefix-closed* if $L = \mathrm{Pref}(L)$. Analogously, we define the languages $\mathrm{Inf}(L) = \{y \in \Sigma^* \mid \exists x, z \in \Sigma^* \colon xyz \in L\}$ and $\mathrm{Suff}(L) = \{y \in \Sigma^* \mid \exists x \in \Sigma^* \colon xy \in L\}$ which contain the infixes and suffixes of words in L, respectively.

A language property $\mathcal{P}$ is any set of languages. A language L satisfies $\mathcal{P}$, or has $\mathcal{P}$, if L is in $\mathcal{P}$. Here by a *property* $\mathcal{P}$ we mean an independence property in the sense of [6]: there is $n \in \mathbb{N} \cup \{\infty\}$ such that a language L satisfies $\mathcal{P}$, if and only if L' satisfies $\mathcal{P}$ for all subsets L' of L of cardinality strictly less than n. A language L satisfying $\mathcal{P}$ is *maximal* (with respect to $\mathcal{P}$), if for every word $w \in \Sigma^*$ either $w \in L$ or $L \cup w$ does not satisfy L. We note that the definition of independence property ensures that every langauge in $\mathcal{P}$ is a subset of a maximal language in $\mathcal{P}$ [6]. Moreover, to our knowledge all code related properties in the literature, including DNA code properties, are independence properties.

A *transducer* $\mathbf{t}$ is a non-deterministic finite state automaton with output; see e. g., [18]. Formally, a transducer is a tuple $\mathbf{t} = (Q, \Sigma, E, I, F)$ where Q is a finite set of nodes, E is a set of directed edges between states from Q which are labelled by word pairs u/v from $\Sigma^* \times \Sigma^*$, I is a set of initial states, and F a set of final states. For an edge label u/v the word u is called *input*, while the word v is called *output*. For a word w, the set $\mathbf{t}(w)$ contains all possible outputs of $\mathbf{t}$ on input w; that is $v \in \mathbf{t}(w)$ if and only if there exists a path from an initial state $p \in I$ to a final state $q \in F$ which is labelled w/v. The *domain* of $\mathbf{t}$ is the set of all words w such that $\mathbf{t}(w) \neq \emptyset$. A transducer $\mathbf{t}$ is called *input-preserving* if $w \in \mathbf{t}(w)$ for every word w in the domain of $\mathbf{t}$. The transducer $\mathbf{t}^{-1}$ is the inverse of $\mathbf{t}$; that is, $w \in \mathbf{t}^{-1}(v)$ if and only if $v \in \mathbf{t}(w)$ for all words v, w. Note that $\mathbf{t}^{-1}$ is obtained from $\mathbf{t}$ by simply swapping the input with the output word

on each edge in $\mathbf{t}$. For a language L we naturally extend our notation such that

$$\mathbf{t}(L) = \{v \in \Sigma^* \mid \exists w \in L \colon v \in \mathbf{t}(w)\},$$
$$\mathbf{t}^{-1}(L) = \left\{w \in \Sigma^* \;\middle|\; \exists v \in L \colon w \in \mathbf{t}^{-1}(v)\right\}.$$

The property $\mathcal{P}_\mathbf{t}$ defined by an input-preserving transducer $\mathbf{t}$ is the set of all languages L satisfying

$$\forall w \in L \colon \mathbf{t}(w) \cap (L \setminus w) = \emptyset.$$

For example, any transducer $\mathbf{t}_i$ such that $\mathbf{t}_i(w) = \mathrm{Inf}(w)$ can be used to define the infix code property. Indeed, a language L is an *infix code* if no word of L is an infix of another word of L.

2.1. Regular Languages and Finite Monoids

Let $L \subseteq \Sigma^*$ be a language, let $(\mathcal{M}, \cdot)$ be a monoid, and $h \colon \Sigma^* \to \mathcal{M}$ be a morphism. The morphism h is said to *recognize* the language L if $h^{-1}(h(L)) = L$. We also say the monoid $(\mathcal{M}, \cdot)$ recognizes L if a morphism $h \colon \Sigma^* \to \mathcal{M}$ recognizing L exists. Throughout this paper, we only consider monoids $(\mathcal{M}, \cdot)$ with multiplication operator '$\cdot$' and, for the ease of notation, we call the underlying set $\mathcal{M}$ a monoid. By the inverse morphism h^{-1} every element $X \in \mathcal{M}$ defines an equivalence class on Σ^* such that for words $u, v \in h^{-1}(X)$ and $x, y \in \Sigma^*$ we have $xuy \in L$ if and only if $xvy \in L$. In other words, if $h(u) = h(v)$ and u is a factor of some word w, then u can be replaced by v in w without changing w's membership with respect to L.

A monoid recognizing L which divides every other monoid recognizing L is called *syntactic monoid* of L. It is well-known that L is a regular language if and only if its syntactic monoid is finite. When considering regular languages, the syntactic monoid of L is the smallest monoid recognizing L. Let $R_1, \ldots, R_k$ be regular and let $\mathcal{M}_1, \ldots, \mathcal{M}_k$ be their syntactic monoids; there is a finite monoid $\mathcal{M}$ which recognizes every language R_i for $1 \leq i \leq k$, e. g., the Cartesian product monoid $\mathcal{M} = \mathcal{M}_1 \times \ldots \times \mathcal{M}_k$. A profound introduction on recognizability of languages by monoids (or semigroups) and syntactic equivalency is given in [19–21]. One of the well-known pumping lemmas for regular languages is: Refs [20, 21] added.

Lemma 1 (Pumping). *Let $\mathcal{M}$ be a finite monoid of size $m = |\mathcal{M}|$ and let $h \colon \Sigma^* \to \mathcal{M}$ be a morphism. For a word u with length $|u| \geq m$ there exists a factorization $u = u_1u_2u_3$ such that $u_2 \neq \varepsilon$, $|u_1u_2| \leq m$, and $h(u_1) = h(u_1u_2)$.*

2.2. Preliminaries

Most of the lemmas in this section are folklore or based on well-known techniques in language theory. For words $u, v, w \in \Sigma^*$ such that $w = uv$, we let $u^{-1}w = v$ and $wv^{-1} = u$. For languages K, L the *quotient languages* are defined as

$$K^{-1}L = \{v \in \Sigma^* \mid \exists u \in K \colon uv \in L\}, \quad LK^{-1} = \{u \in \Sigma^* \mid \exists v \in K \colon uv \in L\}.$$

Note that quotients are not associative with catenation, i. e., $(X^{-1}Y)Z \neq X^{-1}(YZ)$ in general; however, $X^{-1}YZ^{-1} = (X^{-1}Y)Z^{-1} = X^{-1}(YZ^{-1})$ is well defined. The following lemma is well-known; see e. g., [16].

Lemma 2. *Let K, L be regular. The quotients $K^{-1}L$ and LK^{-1} are effectively regular.*

Here and in the following, by an *effectively regular* language we mean that if the base languages (here, K and L) are given by their syntactic monoids (or finite automata, regular expressions, ...), a (syntactic) monoid recognizing the language can be effectively constructed.

In order to solve language equations we will use the following lemma.

Lemma 3. *Let L, X, Y, Z be languages.*

$$L \cap XYZ = \emptyset \iff X^{-1}LZ^{-1} \cap Y = \emptyset.$$

Proof. Suppose there is $w \in L \cap XYZ$, then there exist $x \in X$, $y \in Y$, and $z \in Z$ such that $w = xyz$. Therefore, $x^{-1}wz^{-1} = y$ is well-defined and $y \in X^{-1}LZ^{-1} \cap Y$.

Conversely, consider $y \in X^{-1}LZ^{-1} \cap Y$. For some $x \in X$ and $z \in Z$ we have $xyz \in L$. Clearly, $xyz \in L \cap XYZ$. □

The *square-root* of a language $L \subseteq \Sigma^*$ is defined as $\sqrt[2]{L} = \{x \in \Sigma^* \mid xx \in L\}$. Square-roots of regular languages are effectively regular, as proven in [19]. Since this lemma is less known, we will present its short proof.

Lemma 4. *If L is regular, then $\sqrt[2]{L}$ is effectively regular.*

Proof. Let $\mathcal{M}$ be a finite monoid and $h\colon \Sigma^* \to \mathcal{M}$ be a morphism recognizing L. Let $\mathcal{X} = \left\{X \in \mathcal{M} \;\middle|\; h^{-1}(X)h^{-1}(X) \subseteq L\right\}$. We claim that $\sqrt[2]{L} = h^{-1}(\mathcal{X})$.

Firstly, consider $X \in \mathcal{X}$ and $x \in h^{-1}(X)$. By definition of $\mathcal{X}$ we see that $xx \in h^{-1}(X)h^{-1}(X) \subseteq L$ and, therefore, $x \in \sqrt[2]{L}$.

Secondly, let $x \in \sqrt[2]{L}$ and $X = h(x)$ which implies $xx \in L$ and hence $h^{-1}(X)h^{-1}(X) \subseteq L$. We conclude $X \in \mathcal{X}$ and $x \in h^{-1}(\mathcal{X})$. □

We will use some basic facts about combinatorics on words; see e. g., [22]. We start with the well-known Fine and Wilf's Theorem.

Theorem 5 (Fine and Wilf's [23]). *Let v, w be words. Suppose v^k and w^ℓ, for some $k, \ell \in \mathbb{N}$, have a common prefix of length $|v| + |w| - \gcd(|v|, |w|)$. Then there exists a word u of length $\gcd(|v|, |w|)$ such that $v, w \in u^*$.*

Moreover, $|v| + |w| - \gcd(|v|, |w|)$ is the smallest value that makes the theorem true.

A word u is called *primitive* if there is no word v and $i \geq 2$ such that $u = v^i$. The *primitive root* (not to be confused with the square root) of a word $w \neq \varepsilon$ is

the unique primitive word u such that $w = u^i$ for some $i \geq 1$. For primitive u, if $xuy \in u^+$, then x and y belong to u^*.

Two words u, v are *conjugates* of each other if u can be factorized $u = xy$ such that $v = x^{-1}ux = yx$; this implies $u = y^{-1}vy$. As a direct consequence of Fine and Wilf's Theorem, we obtain:

Corollary 6. *Let v, w be words. If $v^{|w|}$ is an infix of the word $w^{|v|+1}$, then the primitive roots of v and w are conjugates of each other.*

We need one more preliminary observation:

Lemma 7. *Let v, w be words. If vw is an infix of a word in v^+, then w is a prefix of a word in v^+.*

Proof. This follows immediately from the well known property that for a primitive word u, we have $\Sigma^+ u \Sigma^+ \cap uu = \emptyset$.

3. Equations for Deciphering Delay 1

Informally, when a language L has deciphering delay d, for some nonnegative integer d, one can identify correctly the first codeword u in a given message $w \in L^+$ by reading a prefix $uu_1 \cdots u_d$ of w, where $u, u_1, \ldots, u_d \in L$. Then, the deciphering delay d of L ensures that $w \in uL^*$ and that u is the only word in L with that property. Formally, L has *deciphering delay* d, if

$$\forall u \in L \colon uL^d\Sigma^* \cap (L \setminus u)L^d\Sigma^* = \emptyset.$$

The deciphering delay d property is an independence property and any language L with deciphering delay d is a code. Moreover, L has deciphering delay 0 if and only if it is a prefix code; therefore, deciphering delay 0 is a transducer-based property. We note that, in [24], a language L has verbal deciphering delay d if and only if $L^d\Sigma^* \cap L^{-1}L^* \subseteq L^*$. This definition does not imply that the language L is a code. On the other hand, the definition of deciphering delay used in this paper implies that the language involved is indeed a code.

In this paper, we focus on languages with deciphering delay 1. This case is already important, as for any positive integer d, a language L has deciphering delay d if and only if the language L^d has deciphering delay 1. In this context, a language L with deciphering delay 1 is *not* maximal (with deciphering delay 1) if there exists a word $w \in L^c$ such that

$$\forall u \in (L \cup w) \colon u(L \cup w)\Sigma^* \cap ((L \cup w) \setminus u)(L \cup w)\Sigma^* = \emptyset. \qquad (\star)$$

It is interesting to recall here that maximality of a regular code L is equivalent to completeness of the code L, that is, $\mathrm{Inf}(L^*) = \Sigma^*$, [1], a fact that leads to the decidability of maximality of deciphering delay d [5].

Equation ($\star$) is equivalent to the conjunction of the following five equations.

$$wL\Sigma^* \cap LL\Sigma^* = \emptyset, \tag{1}$$
$$wL\Sigma^* \cap Lw\Sigma^* = \emptyset, \tag{2}$$
$$ww\Sigma^* \cap LL\Sigma^* = \emptyset, \tag{3}$$
$$ww\Sigma^* \cap Lw\Sigma^* = \emptyset, \tag{4}$$
$$\forall u, v \in L \text{ with } u \neq v\colon u(L \cup w)\Sigma^* \cap v(L \cup w)\Sigma^* = \emptyset. \tag{$\dagger$}$$

In Equation ($\dagger$), if u is not a prefix of v and v is not a prefix of u, then the equation is satisfied for all w. Due to symmetry, we are only interested in the case when u is a proper prefix of v:

$$\begin{aligned}&\forall u, v \in L\colon (L \cup w)\Sigma^* \cap (u^{-1}v \cap \Sigma^+)(L \cup w)\Sigma^* = \emptyset\\ \iff\ &(L \cup w)\Sigma^* \cap (L^{-1}L \cap \Sigma^+)(L \cup w)\Sigma^* = \emptyset.\end{aligned}$$

This equation is equivalent to the conjunction of

$$L\Sigma^* \cap (L^{-1}L \cap \Sigma^+)w\Sigma^* = \emptyset, \tag{5}$$
$$w\Sigma^* \cap (L^{-1}L \cap \Sigma^+)L\Sigma^* = \emptyset, \tag{6}$$
$$w\Sigma^* \cap (L^{-1}L \cap \Sigma^+)w\Sigma^* = \emptyset. \tag{7}$$

Note that the equation $L\Sigma^* \cap (L^{-1}L \cap \Sigma^+)L\Sigma^* = \emptyset$ is satisfied because L has deciphering delay 1. Let L_i be the set of words w satisfying Equation (i) for $1 \leq i \leq 7$. The language L is maximal with deciphering delay 1 if and only if

$$L^c \cap L_1 \cap L_2 \cap L_3 \cap L_4 \cap L_5 \cap L_6 \cap L_7 = \emptyset. \tag{$\ddagger$}$$

Remark 1. If w satisfies Equation (1) and hence $w \in L_1$, then $w \notin L$. Therefore, we can omit L^c in Intersection ($\ddagger$).

We will solve the equations independently. Let us begin with the cases where w or w^2 only occurs once in the equation.

Lemma 8. *L_1, L_3, L_5, L_6 are given by the effectively regular languages*

$$\begin{aligned}L_1 &= \left((LL\Sigma^*)(L\Sigma^*)^{-1}\right)^c,\\ L_3 &= \sqrt[2]{((LL\Sigma^*)(\Sigma^*)^{-1})^c},\\ L_5 &= \left((L^{-1}L \cap \Sigma^+)^{-1}(L\Sigma^*)(\Sigma^*)^{-1}\right)^c,\\ L_6 &= \left(\left((L^{-1}L \cap \Sigma^+)L\Sigma^*\right)(\Sigma^*)^{-1}\right)^c.\end{aligned}$$

Proof. By definition $w \in L_1$ if and only if $wL\Sigma^* \cap LL\Sigma^* = \emptyset$ which is equivalent to $w \cap (LL\Sigma^*)(L\Sigma^*)^{-1} = \emptyset$, by Lemma 3. We conclude that $w \in L_1$ if and only if $w \notin (LL\Sigma^*)(L\Sigma^*)^{-1}$. Therefore,

$$L_1 = \left((LL\Sigma^*)(L\Sigma^*)^{-1}\right)^c.$$

We can obtain

$$L_5 = \left((L^{-1}L \cap \Sigma^+)^{-1}(L\Sigma^*)(\Sigma^*)^{-1}\right)^c,$$
$$L_6 = \left(\left((L^{-1}L \cap \Sigma^+)L\Sigma^*\right)(\Sigma^*)^{-1}\right)^c$$

from (5) and (6), respectively, by using the same arguments. Furthermore, from (3) we deduce that $w \in L_3$ if and only if $ww \notin LL\Sigma^*(\Sigma^*)^{-1}$. Clearly, this leads to

$$L_3 = \sqrt[2]{((LL\Sigma^*)(\Sigma^*)^{-1})^c}.$$

By Lemmas 2 and 4, L_1, L_3, L_5, and L_6 are effectively regular. □

The language L_2 is not necessarily regular. However, we can split the language L_2 up into a regular part $L_{2.1}$ and a non-regular part $L_{2.2}$ such that $L_2 = L_{2.1} \cap L_{2.2}$. Moreover, the non-regular part can be omitted in Intersection (‡) as stated in the following lemma.

Lemma 9. *There exists an effectively regular language $L_{2.1}$ such that $L_1 \cap L_2 = L_1 \cap L_{2.1}$.*

Proof. Equation (2) is satisfied if and only if both of the following equations are satisfied

$$wL\Sigma^* \cap (L \cap \Sigma^{\geq |w|})w\Sigma^* = \emptyset, \tag{2.1}$$
$$wL\Sigma^* \cap (L \cap \Sigma^{< |w|})w\Sigma^* = \emptyset. \tag{2.2}$$

Let $L_{2.1}$ and $L_{2.2}$ be the sets of words w satisfying (2.1) and (2.2), respectively. Clearly, $L_2 = L_{2.1} \cap L_{2.2}$.

We will first prove that $L_1 \subseteq L_{2.2}$, respectively $L_{2.2}^c \subseteq L_1^c$, whence $L_1 \cap L_2 = L_1 \cap L_{2.1}$. Consider $w \notin L_{2.2}$. There exist $u_1, u_2 \in L$ and $v_1, v_2 \in \Sigma^*$ such that $wu_1v_1 = u_2wv_2$ and $u_2 <_p w$. Next, we see that $u_2u_2 \leq_p u_2wv_2$ and, therefore, $wu_1v_1 = u_2wv_2 \in wL\Sigma^* \cap LL\Sigma^*$ and $w \notin L_1$.

Next, let us construct the language $L_{2.1}$. By Lemma 3, Equation (2.1) is satisfied if and only if

$$L\Sigma^* \cap (w^{-1}L)w\Sigma^* = \emptyset \iff (w^{-1}L)^{-1}(L\Sigma^*)(\Sigma^*)^{-1} \cap w = \emptyset$$

is satisfied. Let $\mathcal{M}$ be a finite monoid, let $h\colon \Sigma^* \to \mathcal{M}$ be a surjective morphism recognizing the language L (i. e., $h^{-1}(h(L)) = L$). For $W \in \mathcal{M}$ we define the set $\mathcal{X}_W = \{X \in \mathcal{M} \mid W \cdot X \in h(L)\}$; observe that $h^{-1}(\mathcal{X}_W) = w^{-1}L$ for every $w \in h^{-1}(W)$. Furthermore, we define the regular language

$$L_W = h^{-1}(W) \setminus (h^{-1}(\mathcal{X}_W))^{-1}(L\Sigma^*)(\Sigma^*)^{-1}.$$

Clearly, if a word w belongs to a language L_W, then $h(w) = W$. We claim that

$$L_{2.1} = \bigcup_{W \in \mathcal{W}} L_W.$$

Indeed, $w \in L_{2.1}$ if and only if $w \notin (w^{-1}L)^{-1}(L\Sigma^*)(\Sigma^*)^{-1}$. For $W = h(w)$ we substitute $w^{-1}L$ by $h^{-1}(\mathcal{X}_W)$ and we see that $w \in L_{2.1}$ if and only if $w \in L_W$. □

Next, we prove that L_4 can be ignored in Intersection (‡).

Lemma 10. *We have* $L_3 \cap L_4 = L_3$.

Proof. This is equivalent to $L_3 \subseteq L_4$, respectively $L_4^c \subseteq L_3^c$. Consider $w \notin L_4$. There exist $u \in L$ and $v_1, v_2 \in \Sigma^*$ such that

$$wwv_1 = uwv_2.$$

We distinguish between three cases:

1.) $u \leq_p w$: we have that $uu \leq_p uwv_2$ whence $wwv_1 = uwv_2 \in ww\Sigma^* \cap LL\Sigma^*$ and $w \notin L_3$.

2.) $w <_p u \leq_p ww$: we have that $ww \leq_p uw \leq_p uu$ whence $uu \in ww\Sigma^* \cap LL\Sigma^*$ and $w \notin L_3$.

3.) $ww <_p u$: clearly, $uu \in ww\Sigma^* \cap LL\Sigma^*$ and $w \notin L_3$. □

From Remark 1 and Lemmas 8, 9, and 10 it follows that:

Corollary 11. *The following language is effectively regular:*

$$K = L^c \cap L_1 \cap L_2 \cap L_3 \cap L_4 \cap L_5 \cap L_6 = L_1 \cap L_{2.1} \cap L_3 \cap L_5 \cap L_6.$$

Note that L_7 is not included in the intersection in Corollary 11. As we will see next, L_7 may not be regular.

Lemma 12. *The complement of* L_7 *is*

$$L_7^c = \bigcup_{u \in L^{-1}L} \mathrm{Pref}(u^+).$$

Proof. Equation (7) is equivalent to

$$w\Sigma^* \cap (L^{-1}L \cap \Sigma^+)w = \emptyset.$$

For a word w Equation (7) is *not* satisfied ($w \notin L_7$) if and only if

$$w \in \mathrm{Pref}\left((L^{-1}L \cap \Sigma^+)w\right).$$

Consider $w \in \mathrm{Pref}(u^+)$ for some word u; clearly, $w \leq_p uw$. If $u \in L^{-1}L \cap \Sigma^+$, then $w \in \mathrm{Pref}\left((L^{-1}L \cap \Sigma^+)w\right)$ and $w \notin L_7$. Conversely, let $w \notin L_7$, i. e., $w \leq_p uw$ for some $u \in L^{-1}L \cap \Sigma^+$. We have that $w \leq_p uw \leq_p uuw \leq_p \cdots \leq_p u^{|w|}w$. Thus, $w \in \mathrm{Pref}(u^+)$ as desired. Finally, note that $\bigcup_{u \in L^{-1}L \cap \Sigma^+} \mathrm{Pref}(u^+) = \bigcup_{u \in L^{-1}L} \mathrm{Pref}(u^+)$. □

Even though L_7 is not regular in general, we can decide whether $K \cap L_7$ is empty or not. We will show that $K \cap L_7 = \emptyset$ if and only if $K = \emptyset$ (though, in general $K \cap L_7 \neq K$). The case when $K = \emptyset$ is trivial, thus, we assume $v \in K$. Furthermore, we assume that $v \notin L_7$; otherwise, we are done. Lemma 14 shows that for this choice of v we find $v\Sigma^* \subseteq K$ and that there exists $w \in v\Sigma^*$ such that $w \in L_7$. Lemma 13 states a prerequisite property.

Lemma 13. *The languages L_5^c and L_6^c are prefix-closed. In particular, for a word $v \in L_5 \cap L_6$, we have $v\Sigma^* \subseteq L_5 \cap L_6$.*

Proof. Let $w \notin L_5$. Because w does not satisfy Equation (5), there exists a word

$$x \in L\Sigma^* \cap (L^{-1}L \cap \Sigma^+)w\Sigma^*.$$

For all prefixes v of w we see that

$$x \in L\Sigma^* \cap (L^{-1}L \cap \Sigma^+)v\Sigma^*.$$

Therefore, $v \notin L_5$. The arguments for the prefix closure of L_6^c are analogue. □

Lemma 14. *For $v \in K \setminus L_7$, we have*

i.) $v\Sigma^* \subseteq K$ and

ii.) $v\Sigma^* \cap L_7 \neq \emptyset$.

Proof. By Lemma 12, we see that $v \in \mathrm{Pref}(u^+)$ for some $u \in L^{-1}L \cap \Sigma^+$. Since $v \in L_6$, it satisfies

$$v\Sigma^* \cap (L^{-1}L \cap \Sigma^+)L\Sigma^* = \emptyset,$$

which implies that $v \notin \mathrm{Pref}(L^{-1}L)$ and $u <_p v$. Furthermore, from (6) we obtain

$$v\Sigma^* \cap uL\Sigma^* = \emptyset \implies u^{-1}v\Sigma^* \cap L\Sigma^* = \emptyset \implies v\Sigma^* \cap L\Sigma^* = \emptyset.$$

Now, let $w \in v\Sigma^*$. Observe that w satisfies Equations (1) through (4); indeed, if any of the four equations were not satisfied, there existed a word x such that v and a word from L would be prefixes of x and, therefore, $x \in v\Sigma^* \cap L\Sigma^*$. Recall from Lemma 13 that $w \in L_5 \cap L_6$, too, whence Statement *i.)* of the claim holds.

For all $w \in v\Sigma^*$ with $w \notin L_7$ there exists a word $y \in L^{-1}L \cap \Sigma^+$ such that $w = \mathrm{Pref}(y^+)$, by Lemma 12. Because $v \notin \mathrm{Pref}(L^{-1}L)$, we see that y has to be a proper prefix of v and $w \in \bigcup_{y <_p v} \mathrm{Pref}(y^+) =: Y$. Let $a, b \in \Sigma$ be two distinct letters. If $v \in a^+$, then we chose $z = vb$; otherwise, we chose $z = va^{|v|}$. In both cases, $z \notin Y$ and we conclude that $z \in v\Sigma^* \cap L_7$. □

By Lemma 14 and Corollary 11, we conclude:

Corollary 15. *$K \cap L_7 = \emptyset$ if and only if $K = \emptyset$.*

Hence, we provide an alternative proof for the following, well known result which was first proven in [5].

Corollary 16. *It is decidable whether or not a given regular code L with deciphering delay 1 is maximal with deciphering delay 1.*

4. Deciphering Delay 1 and a Transducer Property

In this section, we consider the combined property "deciphering delay 1 and $\mathcal{P}_{\mathbf{t}}$" where $\mathcal{P}_{\mathbf{t}}$ is the property defined by a given input-preserving transducer $\mathbf{t}$. In [9], it is shown that if a language L satisfies $\mathcal{P}_{\mathbf{t}}$, then L is maximal if and only if the language $R_{\mathbf{t}} \cap L^c = \emptyset$ where

$$R_{\mathbf{t}} = (\mathbf{t}(L) \cup \mathbf{t}^{-1}(L))^c.$$

Recall that $v \in \mathbf{t}^{-1}(L)$ if and only if $\mathbf{t}(v) \cap L \neq \emptyset$. Moreover, $R_{\mathbf{t}}$ is effectively regular.

Following a reasoning similar to that in the beginning of Section 3, one can verify that, if L has deciphering delay 1 and satisfies $\mathcal{P}_{\mathbf{t}}$, then L is *not* maximal with respect to these combined properties if and only if there exists a word $w \in L^c$ such that $w \in R_{\mathbf{t}}$ and Equations (1)–(7) are satisfied. Then by Corollary 11, L is maximal if and only if

$$K \cap L_7 \cap R_{\mathbf{t}} = \emptyset.$$

Recall from the introductory section that M is a fixed (but arbitrary) regular language, which we refer to as cover language. Maximality *within the cover language* M is formally defined as follows: L is maximal with property $\mathcal{P}$ within the cover language M, if L satisfies $\mathcal{P}$, $L \subseteq M$, and for all L' with $L \subsetneq L' \subseteq M$ we have that L' does not satisfy $\mathcal{P}$. In order to decide maximality of L with respect to "deciphering delay 1" and $\mathcal{P}_{\mathbf{t}}$ within a fixed, *regular* cover language M, we have to decide whether or not $w \in M \setminus L$ exists such that Equations (1)–(7) are satisfied and $L \cup w$ satisfies $\mathcal{P}_{\mathbf{t}}$. Then, L is maximal within the cover language M and with respect to these combined properties if and only if

$$K \cap L_7 \cap R_{\mathbf{t}} \cap M = \emptyset.$$

The following theorem states that both of the above emptiness problems are decidable.

We say that *a language equation system has a solution under restriction* R, where R is a language, if it has a solution that belongs to R.

Theorem 17. *It is decidable, for given regular languages L and R, whether or not the system of Equations (1)–(7) has a solution under restriction R.*

By the discussion in the beginning of this section the following holds.

Corollary 18. *It is decidable, for given regular languages L, M and an input-preserving transducer $\mathbf{t}$, whether or not L is maximal within M with respect to the combined property "deciphering delay 1 and $\mathcal{P}_{\mathbf{t}}$".*

Next, we present the proof of Theorem 17.

Proof. The system of Equations (1)–(7) has no solution under restriction R if and only if $K \cap L_7 \cap R = \emptyset$. Recall, from Lemma 12, that $L_7^c = \bigcup_{u \in L^{-1}L} \mathrm{Pref}(u^+)$. If there exists a word $w \in K \cap L_7 \cap R$, we call w a *witness*. We will show that in the case when L is maximal, any potential witness $w \in (K \cap R) \setminus \mathrm{Pref}(L^{-1}L)$ can be factorized $w = uv$ such that $u \in L^{-1}L$, $v \in \mathrm{Pref}(u^+)$, and $|u| < m$ or $|v| < m$ for a bound m. The test language T containing all those words uv is regular wherefore it is decidable whether or not all potential witnesses are included in T — thus, proving that it is decidable whether or not the system of Equations (1)–(7) has a solution under restriction R.

Let $\mathcal{M}$ be a finite monoid and $h\colon \Sigma^* \to \mathcal{M}$ be a surjective morphism recognizing the languages K, R, and $L^{-1}L$. The size of $\mathcal{M}$ is denoted by m and is the bound mentioned above. Note that $\mathcal{M}$ also recognizes the language $\mathrm{Pref}(L^{-1}L)$ as

The notation $\mathcal{L}_Z$ was removed from this proof and replaced by $h(Z)$.

$$\mathcal{P}_{L^{-1}L} = h(\mathrm{Pref}(L^{-1}L)) = \left\{ P \in \mathcal{M} \;\middle|\; \exists X \in \mathcal{M}\colon P \cdot X \in h(L^{-1}L) \right\}$$

satisfies $h^{-1}(\mathcal{P}_{L^{-1}L}) = \mathrm{Pref}(L^{-1}L)$. We define the set of possible witnesses $\mathcal{W} = (h(K) \cap h(R)) \setminus h(\mathrm{Pref}(L^{-1}L))$. Surely, if a witness $w \in K \cap L_7 \cap R$ exists, we find w in $(K \cap R) \setminus \mathrm{Pref}(L^{-1}L) = h^{-1}(\mathcal{W})$. Let the set $\mathcal{U}$ contain all pairs $(U, V) \in \mathcal{M}^2$ where $U \in h(L^{-1}L)$ and $U \cdot V \in \mathcal{W}$. For $(U, V) \in \mathcal{U}$, $u \in h^{-1}(U) \cap \Sigma^+$, and $v \in h^{-1}(V)$ with $v \leq_p uv$ (i. e., $v \in \mathrm{Pref}(u^+)$), we have $uv \notin L_7$. We define the test language

$$T = \bigcup_{u \in L^{-1}L \cap \Sigma^{<m}} \mathrm{Pref}(u^+) \;\cup \bigcup_{\substack{(U,V) \in \mathcal{U} \\ v \in h^{-1}(V) \cap \Sigma^{<m}}} v\Sigma^+ \cap h^{-1}(U)v.$$

Due to the length restrictions of u and v in the formula, T is the finite union of regular languages and, hence, regular itself. We claim that the system of Equations (1)–(7) has no solution under restriction R if and only if $h^{-1}(\mathcal{W}) \subseteq T$. Observe that every language in the union of T is a subset of L_7^c wherefore $T \subseteq L_7^c$ as well. This implies that if $h^{-1}(\mathcal{W}) \subseteq T$, then $K \cap L_7 \cap R = h^{-1}(\mathcal{W}) \cap L_7 = \emptyset$ — thus, proving the if-part of the claim.

As for the only-if-part, suppose that $K \cap L_7 \cap R = \emptyset$. Let $W \in \mathcal{W}$ and $w \in h^{-1}(W)$; we have to prove that $w \in T$. Because $w \in K \cap R$, it cannot belong to L_7. There is a factorization $w = uv$ such that $u \in h^{-1}(U) \cap \Sigma^+$, $v \in h^{-1}(V)$, and $v <_p uv$ for some $(U, V) \in \mathcal{U}$ with $U \cdot V = W$. If $|u| < m$ or $|v| < m$, then $w \in T$ and we are done. Henceforth, we suppose that $|u| \geq m$ and $|v| \geq m$. We distinguish between the two cases whether there exists $x \neq \varepsilon$ such that $W \cdot h(x) = W$ or not.

Firstly, suppose $W \cdot h(x) = W$ for $x \neq \varepsilon$. By the Pumping Lemma 1, factorize $u = u_1u_2u_3$ such that $u_2 \neq \varepsilon$, $|u_1u_2| \leq m$, and $h(u_1) = h(u_1u_2)$. Consider the word $w' = u_1u_2^k u_3 v x^\ell$ where $k = |x|$ and ℓ is sufficiently large (e. g., $\ell = 2k \cdot |w|$); the factorization of w' is illustrated in Fig. 1(a). Because $w' \in h^{-1}(W)$ but $w' \notin L_7$, there is at least one factorization $w' = u'v'$ such that $u' \in h^{-1}(U') \cap \Sigma^+$, $v' \in h^{-1}(V')$, and $v' <_p u'v'$ for some $(U', V') \in \mathcal{U}$ with $U' \cdot V' = W$. Among all those factorizations we chose the one where $|u'|$

Fig. 1 added.

is minimal. The word u' has to be a proper prefix of $u_1u_2^ku_3v \notin \mathrm{Pref}(L^{-1}L)$ Therefore, u_2^k and u'^k are infixes of x^ℓ; see Fig. 1(a). By Corollary 6, the primitive roots of u_2, u', and x are conjugates of each other. This has two implications:

1.) u' is a proper prefix of u_1u_2. Otherwise, u' could be factorized $u' = u_1u_2y$, and hence, $|u'|$ would not be chosen minimal because $w' \in \mathrm{Pref}((u_1y)^+)$ and $u_1y \in h^{-1}(U')$.

2.) We can pump down u_2^k to u_2 and still have that $u_1u_2u_3vx^\ell \in \mathrm{Pref}(u'^+)$; see Fig. 1(b).

We conclude $u' \in L^{-1}L$ is a prefix of w, $|u'| \leq m$, and $w \in \mathrm{Pref}(u'^+) \subseteq T$ as desired.

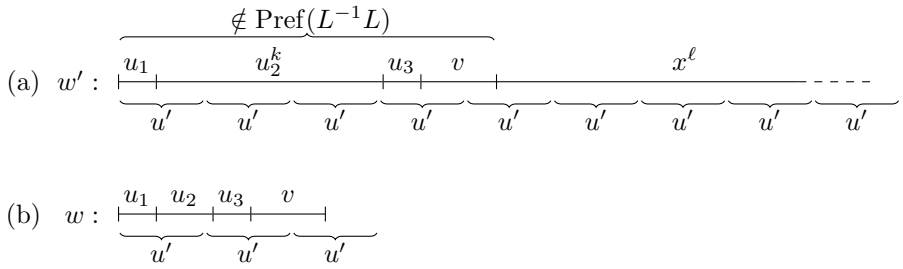

Figure 1: (a) The word $w' \in h^{-1}(W)$ is obtained from w by pumping the factor u_2 and concatenating x^ℓ. There exists a proper prefix $u' \in L^{-1}L$ of $u_1u_2^ku_3v$ such that $w' \in \mathrm{Pref}(u'^+)$, therefore, u_2^k appears as an infix of x^ℓ. (b) By implication 1.), u' is a proper prefix of u_1u_2 and, as stated in implication 2.), after pumping down u_2^k to u_2, we obtain w which also belongs to $\mathrm{Pref}(u'^+)$.

Now, consider the case when $W \cdot h(x) \neq W$ unless $x = \varepsilon$. This means that for two distinct words $w_1, w_2 \in h^{-1}(W)$ the word w_1 cannot be a prefix of w_2; otherwise, $W \cdot h(w_1^{-1}w_2) = W$. In particular, for any $(U', V') \in \mathcal{U}$ such that $U' \cdot V' = W$ no word $v_1 \in h^{-1}(V')$ can be a prefix of another word $v_2 \in h^{-1}(V')$.

As $|v| \geq m$ and in accordance to the Pumping Lemma 1, we factorize $v = v_1v_2v_3$ such that $v_2 \neq \varepsilon$, $|v_1v_2| \leq m$, and $h(v_1) = h(v_1v_2)$. Since v is a prefix of $uv = w$, we can write $w = v_1v_2v_3x$ where, due to length restrictions, v_1v_2 is a prefix of u and v is a suffix of v_3x; see Fig. 2(a). Consider the word $w' = v_1v_2^kv_3x \in h^{-1}(W)$ where k is sufficiently large (e. g., $k = 2\,|w|^2$). Once more, we factorize $w' = u'v'$ such that $u' \in h^{-1}(U') \cap \Sigma^+$, $v' \in h^{-1}(V')$, and $v' <_p u'v'$ for some $(U', V') \in \mathcal{U}$ with $U' \cdot V' = W$. We distinguish between three cases, of which the first two yield a contradiction and the third case implies that $w \in T$. Fig. 2 added

1.) If $|u'| \leq |v_3x|$, the primitive roots of u' and v_2 were conjugates of each other, by Corollary 6; see Fig. 2(b). In this case v_3 were in $\mathrm{Pref}(v_2^+)$ as $v_2v_3 \in \mathrm{Inf}(u'^+) = \mathrm{Inf}(v_2^+)$, by Lemma 7. This would make $v_1v_3 \in h^{-1}(V)$ a prefix of $v_1v_2v_3 \in h^{-1}(V)$ which contradicted the fact that no word in $h^{-1}(V)$ is a prefix of another word in $h^{-1}(V)$. Hence, we have $|u'| > |v_3x|$ and $v' <_p v_1v_2^k$.

2.) Suppose $v' = v_1 z$ where $z \in \mathrm{Pref}(v_2^k)$; see Fig. 2(c). Then, $v' \in h^{-1}(V')$ were a prefix of $v_1 v_2 z \in h^{-1}(V')$ — again, a contradiction.

3.) Due to cases 1.) and 2.), v' is a prefix of v_1, as such has a length of less than m and is a suffix of v. There is factorization $w = yv'$ such that $v' \leq_p yv'$ and $y \in h^{-1}(U') \cap \Sigma^+$; therefore, $w \in T$.

This concludes the proof that $h^{-1}(\mathcal{W}) \subseteq T$ if and only if $K \cap L_7 \cap R = \emptyset$. □

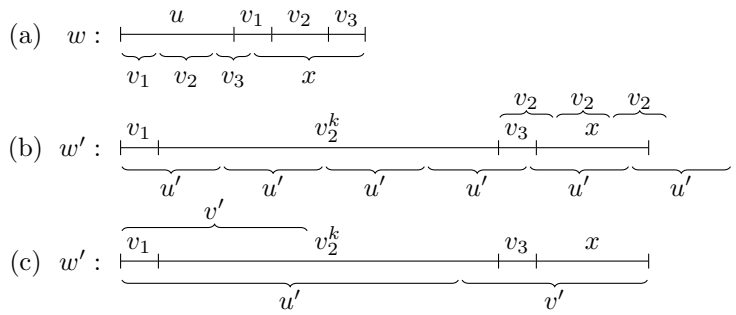

Figure 2: (a) Factorization of the word $w = uv_1v_2v_3 = v_1v_2v_3x \in h^{-1}(W)$; (b) the word $w' \in h^{-1}(W)$ in case 1.) where u' is relatively short; (c) the word $w' \in h^{-1}(W)$ in cases 2.) and 3.) where v' is relatively short.

5. Concluding Remarks

We have considered the maximality decision problem for regular languages with respect to the combined properties deciphering delay 1 and any input-preserving transducer property. This question was phrased conveniently as the solution existence problem for a system of equations with a restriction. We then reduced the solution existence problem to the emptiness problem of the intersection of several languages that are not necessarily regular, and we showed how to further reduce this emptiness problem to an inclusion problem of regular languages which is decidable.

While we have focused on a specific language equation system, we hope that our approach can be used to decide solution existence of similar language equation systems that might correspond to the maximality of other combined types of code properties. For example, consider the *comma-freeness* property for a language L, that is, whether

$$LL \cap \Sigma^+ L \Sigma^+ = \emptyset.$$

It is known that maximality for regular comma-free languages is decidable [25], but again it is not known whether maximality of "comma-freeness" *and* a transducer property $\mathcal{P}_\mathbf{t}$ combined is decidable. One can construct a new system of equations similar to equations (1)–(7) in this paper, so that the given problem is equivalent to the existence of a solution to the new system of equations

References

[1] J. Berstel, D. Perrin, C. Reutenauer, Codes and Automata, Cambridge University Press, 2009.

[2] D. Perrin, Completing biprefix codes, in: Proceedings of 9th Intern. Colloquium on Automata, Languages and Programming, Vol. 140 of LNCS, 1982, pp. 397–406.

[3] L. Zhang, Z. Shen, Completion of recognizable bifix codes, Theoretical Computer Science 145 (1995) 345–355.

[4] N. H. Lam, Finite maximal infix codes, Semigroup Forum 61 (2000) 346–356.

[5] V. Bruyère, Maximal codes with bounded deciphering delay, Theoretical Computer Science 84 (1991) 53–76.

[6] H. Jürgensen, S. Konstantinidis, Codes, in: G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, Vol. 1, Springer-Verlag, Berlin, 1997, pp. 511–607.

[7] H. Jürgensen, Syntactic monoids of codes, Acta Cybernetica 14 (1999) 117–133.

[8] M. Domaratzki, Trajectory-based codes, Acta Informatica 40 (2004) 491–527.

[9] K. Dudzinski, S. Konstantinidis, Formal descriptions of code properties: decidability, complexity, implementation, International Journal of Foundations of Computer Science 23:1 (2012) 67–85.

[10] L. Kari, S. Konstantinidis, Language equations, maximality and error-detection, Journal of Computer and System Sciences 70 (2005) 157–178.

[11] A. Restivo, Codes and local constraints, Theoretical computer science 72 (1) (1990) 55–64.

[12] J. Néraud, On the completion of codes in submonoids with finite rank, Fundamenta Informaticae 74 (4) (2006) 549–562.

[13] J. Néraud, Completing prefix codes in submonoids, Theoretical computer science 356 (1) (2006) 245–254.

[14] M. Lothaire, Algebraic combinatorics on words, Vol. 90 of Encyclopedia of Mathematics and its Applications, Cambridge university press, 2002.

[15] M. Domaratzki, K. Salomaa, Codes defined by multiple sets of trajectories, Theoretical Computer Science 366 (2006) 182–193.

[16] J. E. Hopcroft, J. D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, 1979.

[17] G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, Vol. I, Springer-Verlag, Berlin, 1997.

[18] S. Yu, Regular languages, Handbook of Formal Languages 1 (1997) 41–110.

[19] J. Pin, Syntactic semigroups, Handbook of Language Theory, Vol. I (1997) 679–746.

[20] S. Eilenberg, Automata, languages, and machines, Vol. B, Academic Press, 1976.

[21] G. Lallement, Semigroups and combinatorial applications, Pure and applied mathematics, John Wiley & Sons, Inc., 1979.

[22] M. Lothaire, Combinatorics on words, Cambridge University Press, 1997.

[23] N. J. Fine, H. S. Wilf, Uniqueness theorems for periodic functions, Proceedings of the American Mathematical Society (1965) 109–114.

[24] J. Berstel, D. Perrin, J.-F. Perrot, A. Restivo, Sur le théorème du défaut, Journal of algebra 60 (1) (1979) 169–180.

[25] N. H. Lam, Completing comma-free codes, Theoretical Computer Science 301 (2003) 399–415.