ITHEA

International Journal

INFORMATION THEORIES & APPLICATIONS

2009 Volume 16 Number 1

International Journal
INFORMATION THEORIES & APPLICATIONS
Volume 16 / 2009, Number 1

IJ ITA is official publisher of the scientific papers of the members of the ITHEA® International Scientific Society

IJ ITA welcomes scientific papers connected with any information theory or its application.
IJ ITA rules for preparing the manuscripts are compulsory.
The **rules for the papers** for IJ ITA as well as the **subscription fees** are given on *www.ithea.org* .
The camera-ready copy of the paper should be received by http://ij.ithea.org.
Responsibility for papers published in IJ ITA belongs to authors.
General Sponsor of IJ ITA is the **Consortium FOI Bulgaria** (www.foibg.com).

International Journal "INFORMATION THEORIES & APPLICATIONS" Vol.16, Number 1, 2009

Printed in Bulgaria

Edited by the **Institute of Information Theories and Applications FOI ITHEA®**, Bulgaria,
in collaboration with the V.M.Glushkov Institute of Cybernetics of NAS, Ukraine,
and the Institute of Mathematics and Informatics, BAS, Bulgaria.

Publisher: **ITHEA®**
Sofia, 1000, P.O.B. 775, Bulgaria. www.ithea.org, e-mail: info@foibg.com

ISSN 1310-0513 (printed) **ISSN 1313-0463 (online)** **ISSN 1313-0498 (CD/DVD)**

TRANSLITERATION AND LONGEST MATCH STRATEGY

Dimiter Skordev

***Abstract**: A natural requirement on transliteration systems is that transliteration and its inversion could be easily performed. To make this requirement more precise, we consider a text transduction as easily performable if it can be accomplished by a finite transducing device such that all successful tokenizations of input words are compliant with the left-to-right longest-match strategy. Applied to inversion of transliteration this gives a convenient sufficient condition for reversibility of transliteration.*

***Keywords**: left to right, longest match, transliteration, reversible transliteration, sequential transducer.*

***ACM Classification Keywords:** E.4 Data: Coding and information theory — Formal models of communication*

Introduction

When considering a transliteration system, it is natural to impose on the transliteration and on its inversion the requirement to be easily performable (cf., for instance, [7, Section 7] or [8, Subsection 25.2]). This requirement can be made more precise in different ways. We present one in the following spirit: a text transduction is regarded as easily performable if it can be accomplished by a finite transducing device such that all successful tokenizations of input words are compliant with the left-to-right longest-match strategy (cf. e.g. [1] for some other applications of this strategy). The requirement that both transliteration and its inversion could be easily performed is understood in this sense, but with one device and its inverse used for transliteration and for its inversion, respectively. A convenient sufficient condition for reversibility of transliteration is obtained in this way.

Some definitions and examples

Let Σ and Δ be two alphabets, and let T be a mapping of Σ^* into the set $P(\Delta^*)$ of the subsets of Δ^*, where Σ^* consists, as usually, of all finite strings of symbols from Σ (including the empty string ε), and similarly for Δ^*. We may intuitively regard T as a mathematical description of some transliteration system from Σ to Δ, and for any ω in Σ^* consider the strings belonging to $T(\omega)$ as the admissible transliterations of ω in this system (clearly each actual transliteration system from Σ to Δ can be supplied with a description of this kind having some specific features). For any element ω of Σ^* the elements of $T(\omega)$ will be called *images* of ω under T, and ω will be called a *pre-image* under T of each of them. The mapping T will be called *total* if each element of Σ^* has an image under T, *surjective* if each element of Δ^* has a pre-image under T, *single-valued* if no element of Σ^* has two distinct images under T, and *injective* if no element of Δ^* has two distinct pre-images under T. The *inverse mapping* T^{-1} is the mapping of Δ^* into $P(\Sigma^*)$ defined as follows: for any τ in Δ^* the set $T^{-1}(\tau)$ consists of all pre-images of τ under T. Clearly T is total if and only if T^{-1} is surjective, and T is injective if and only if T^{-1} is single-valued.

The following will be assumed in the three examples below: $\Sigma\setminus\Delta$ consists of the capital and the small Russian letters, $\Delta\setminus\Sigma$ consists of the capital and the small Latin letters, $\Sigma\cap\Delta$ contains the space character, the digits and other characters commonly used both in Russian and in English (for instance punctuation marks).

Example 1. The transliteration system proposed in [8] can be described by a mapping T such that for any ω in Σ* the set $T(\omega)$ has as its only element the string from Δ* obtainable from ω by means of replacements of the following kinds (for being easier distinguishable, all Russian letters will be given in boldface):

А→A, **а**→a, **Б**→B, **б**→b, **В**→V, **в**→v, **Г**→G, **г**→g, **Д**→D, **д**→d, **Е**→E, **е**→e, **Ё**→Yo, **ё**→yo,
Ж→Zh, **ж**→zh, **З**→Z, **з**→z, **И**→I, **и**→i, **Й**→Yj, **й**→yj, **К**→K, **к**→k, **Л**→L, **л**→l, **М**→M. **м**→m,
Н→N, **н**→n, **О**→O, **о**→o, **П**→P, **п**→p, **Р**→R, **р**→r, **С**→S, **с**→s, **Т**→T, **т**→t, **У**→U, **у**→u,
Ф→F, **ф**→f, **Х**→Kh, **х**→kh, **Ц**→C, **ц**→c. **Ч**→Ch, **ч**→ch, **Ш**→Sh, **ш**→sh, **Щ**→Th, **щ**→th,
Ъ→Jh, **ъ**→jh, **Ы**→Ih, **ы**→ih, **Ь**→J, **ь**→j, **Э**→Eh, **э**→eh, **Ю**→Yu, **ю**→yu, **Я**→Ya, **я**→ya

(for instance, if ω_0 is the sentence "**Гармонический ряд расходится**." then $T(\omega_0)$ has as its only element the string "Garmonicheskiyj ryad raskhoditsya."). The mapping T is total and single-valued by its definition, and it is injective, but not surjective because of the fact that no Russian letter is replaced by "Y" or "y", and there is no Russian letter whose replacement string begins with "h".

Example 2. We shall add to the transliteration system considered in Example 1 some replacements used by a system that is suggested in a document accessible from http://www.metodii.com. Let us consider a mapping T such that for any ω in Σ* the set $T(\omega)$ consists of all strings from Δ* obtainable from ω by means of replacements of the kinds considered in Example 1 and of the following additional ones: **Ж**→X, **ж**→x, **Ч**→Q, **ч**→q, **Ш**→W, **ш**→w (now the set $T(\omega_0)$ for the concrete ω_0 from Example 1 will have two elements – the string indicated there and the same string with "Garmoniqeskiyj" instead of "Garmonicheskiyj"). The mapping T is again total, injective and non-surjective, but it is not single-valued.

Example 3. Let us define a single-valued mapping T in the same way as in Example 1, except that we take now **Й**→Yy, **й**→yy, **Х**→Hh and **х**→hh instead of **Й**→Yj, **й**→yj, **Х**→Kh and **х**→kh, respectively (thus the set $T(\omega_0)$ for the concrete ω_0 from that example will have as its only element the string "Garmonicheskiyy ryad rashhoditsya."). Then T is still total, single-valued, injective and not surjective, but its injectiveness is seen in a somewhat more complicated way.

The mappings T from the above examples have the property that $T(\varepsilon)=\{\varepsilon\}$ and for any ω_1 and ω_2 in Σ* the equality $T(\omega_1\omega_2)=T(\omega_1)T(\omega_2)$ holds (its right-hand side denotes the set of all concatenations $\tau_1\tau_2$, where τ_1 belongs to $T(\omega_1)$ and τ_2 belongs to $T(\omega_2)$). Any mapping T with this property will be called *homomorphic*. Each mapping C of Σ into $P(\Delta^*)$ can be extended in a unique way to a homomorphic mapping T of Σ* into $P(\Delta^*)$; the mapping T in question will be said to be *generated by C*. Intuitively, we may regard the elements of $C(\sigma)$ for any σ in Σ as the admissible code strings for σ, and regard the mapping T generated by C as a description of transliteration done by replacing the symbols from Σ with admissible code strings for them. If T is generated by C then T is total if and only if all sets $C(\sigma)$ are non-empty, T is single-valued if and only if each set $C(\sigma)$ has at most one element, and T is injective if and only if the following two conditions are satisfied:

(i) $C(\sigma_1)$ and $C(\sigma_2)$ have no common element, whenever σ_1 and σ_2 are distinct symbols of Σ;

(ii) no string in Δ* can be represented in two different ways as a concatenation of strings belonging to the union of all $C(\sigma)$ corresponding to symbols σ of Σ.

In the practically important case when the above-mentioned union is finite (the mapping T will be called *finitary* in that case) a check for the condition (ii) can be always done by means of a theorem of Sardinas and Patterson [5].

In particular, the injectivity of the mappings T from Examples 1, 2 and 3 can be established also in this way (taking as C the restriction of T to Σ).

The possibility of proving injectivity by means of the Sardinas-Patterson theorem does not make pointless the search for other convenient injectivity criteria, namely such ones that give only sufficient conditions for injectivity but guarantee a better quality of the injectivity. There are at least two reasons for this.

1. For the convenience of a transliteration system not only the injectivity matters, but also the quality of the injectivity. An injective mapping T of Σ^* into $P(\Delta^*)$ could allow to find easily some element of $T(\omega)$ for any ω in Σ^*, but the problem to find ω when an element of $T(\omega)$ is given could be much more difficult. To have an example of such a situation, let us consider the mapping T from Example 3. Suppose that a string τ is given whose first symbol is "s", all other ones being "h", and we look for a string ω in Σ^* such that τ belongs to $T(\omega)$. Such a string ω exists, and its first symbol is "**ш**" if the length of τ is even and "**с**" otherwise. Evidently, it is not possible to determine the first symbol of ω on the base of knowing only some proper prefix of τ, thus in the case of a long τ it would be not easy to find ω by reading τ only once from left to right and writing consecutive symbols of ω.
2. Among the numerous transliteration systems proposed until now there are some whose corresponding mappings T are non-homomorphic due to context-dependent encoding of some letters. Such is the case for example with the second of the two systems proposed by Uspensky in [7] – in that system the Cyrillic letter "**Й**" has the encoding "Jh" when followed by a vowel or by a soft sign and has the encoding "J" otherwise, the letter "**й**" being treated in a similar way (however, any other symbol from the corresponding alphabet Σ has a unique code string not depending on the context).

The finitary homomorphic mappings of Σ^* into $P(\Delta^*)$ are a particular case of mappings accomplished by means of sequential transducers in the sense of [2, Section 3.3]. By the definition accepted there, a *sequential transducer* with input alphabet Σ and output alphabet Δ is any quintuple of the form (K,Σ,Δ,H,s_0), where K is a finite set (the set of the *states*), s_0 is an element of K (the *start state*), H is a finite set of quadruples (called *moves*) with first and last components in K, and second and third components in Σ^* and Δ^*, respectively.[1] The mapping T *accomplished* by such a sequential transducer is defined as follows: for any ω in Σ^* the set $T(\omega)$ consists of the elements τ of Δ^* such that for some non-negative integer k, some $\omega_1, \ldots, \omega_k$ in Σ^*, some $\tau_1, \ldots, \tau_k$ in Δ^* and some $s_1, \ldots, s_k$ in K the quadruples $(s_{i-1},\omega_i,\tau_i,s_i)$, $i=1,\ldots,k$, belong to H, and the equalities $\omega=\omega_1\ldots\omega_k$, $\tau=\tau_1\ldots\tau_k$ hold.

If T is a finitary homomorphic mapping of Σ^* into $P(\Delta^*)$ then T can be accomplished by a sequential transducer (K,Σ,Δ,H,s_0) such that $K=\{s_0\}$, and H consists of all quadruples (s_0,σ,θ,s_0) with σ in Σ and θ in $T(\sigma)$. We shall denote by S_1, S_2 and S_3, respectively, sequential transducers constructed in this way for the mappings T considered in Examples 1, 2 and 3.

To have an example of a transliteration system needing a more complicated sequential transducer for the accomplishment of the corresponding mapping, we shall consider in more detail the already mentioned second transliteration system from [7].

Example 4. Let Σ and Δ be as in the previous examples except that the Russian alphabet is supposed to be without capital hard and soft signs, and $\Delta\setminus\Sigma$ contains also the apostrophe besides the Latin letters. The

[1] This terminology is not universally adopted. For example the same term means something else in [3], and its present meaning is somewhat closer to the notion of finite transducer considered there (cf. Section 1.3.3 of that book, but note that there are certain omissions in the definitions of both notions in that section).

corresponding mapping T can be described for instance as follows: for any ω in Σ* the set $T(\omega)$ has as its only element the string from Δ* obtainable from ω by the application of a normal algorithm (in the sense of [6] and [4]) such that its scheme begins with the substitution formulas $\sigma\sigma_1 \rightarrow \theta\sigma_1$, where either σ="**Й**", θ="Jh", or σ="**й**", θ="jh", and σ_1 is a Cyrillic vowel or a soft sign, and the further substitution formulas do the replacements listed in Example 1 except that

(a) "J" and "j" are used instead of "Y" and "y", respectively, in the strings for the letters "**Ё**", "**ё**", "**Ю**", "**ю**", "**Я**", "**я**";

(b) there are no substitution formulas for "**Ъ**" and "**Ь**";

(c) the substitution formulas for "**Й**", "**й**", "**Щ**", "**щ**", "**ъ**", "**Ы**", "**ы**" and "**ь**" are

Й→J, **й**→j, **Щ**→Xh, **щ**→xh, **ъ**→j', **Ы**→Y, **ы**→y, **ь**→'

(of course such a normal algorithm would be not practically convenient, since its execution would require to read one and the same symbol many times; a more appropriate normal algorithm for the same transliteration system can be indicated whose substitution formulas contain an auxiliary symbol and whose execution actually performs a letter-by-letter transliteration from left to right).The mapping T can be accomplished by a sequential transducer $(\{k_0,k_1\},\Sigma,\Delta,H,k_0)$, where $k_0 \neq k_1$. We shall indicate two such sequential transducers (both of them get into the state k_1 after reading "**Й**" or "**й**" and only in that case). The first one closely corresponds to the brief description we gave of the transliteration system in question. The set H of this sequential transducer consists of all quadruples of the following forms:

(i) $(k_0,\sigma,T(\sigma),k_0)$, where σ is in Σ\{"**Й**","**й**"};

(ii) $(k_i,\sigma,T(\sigma),k_1)$, where σ is "**Й**" or "**й**" (i=0,1);

(iii) $(k_1,\sigma,T(\sigma),k_0)$, where σ is in Σ\{"**Й**","**й**"}, and σ is neither a vowel, nor a soft sign;

(iv) $(k_0,\sigma\sigma_1,T(\sigma)$"h"$T(\sigma_1),k_0)$, where σ is "**Й**" or "**й**", and σ_1 is a Cyrillic vowel or a soft sign.

The set H of the second one consists of all quadruples of the forms (i) and (ii) above, as well as of all quadruples (k_1,σ,θ,k_0), where σ is in Σ\{"**Й**","**й**"}, θ is $T(\sigma)$ if σ is not a vowel and not a soft sign, otherwise θ is $T(\sigma)$ preceded by "h". (Note that the second components of all quadruples in this set are one-symbol strings, whereas it is not so for the set H of the first of the considered sequential transducers.). We shall denote the first and the second transducers considered in this example by $S_{4,1}$ and $S_{4,2}$, respectively.

As a further example on the application of sequential transducers to transliteration we shall indicate an extension of Uspensky's transliteration system considered in Example 1. The extension in question can be used for reversible Russian-Latin transliteration of mixed texts — possibly containing not only Russian, but also Latin letters.[1]

Example 5. Let Δ consist of the capital and the small Latin letters and of characters commonly used both in English and in Russian, including the apostrophe, and Σ be obtained from Δ by adding to it all capital and small Russian letters. Let C be the mapping of Σ into Δ* defined as follows: for any Russian letter σ, $C(\sigma)$ is its corresponding string from Example 1, $C(\sigma)=\sigma\sigma$ if σ is an apostrophe, and $C(\sigma)=\sigma$ for all other symbols σ in Σ. Let $D(\sigma)$ be $C(\sigma)$ preceded by an apostrophe. We consider a sequential transducer $S_5=(\{k_0,k_1\},\Sigma,\Delta,H,k_0)$, where $k_0 \neq k_1$ and H consists of the following quadruples:

(i) all quadruples $(k_i,\sigma,C(\sigma),k_i)$, where I = 0 and σ is not a Latin letter, or I =1 and σ is not a Russian letter;

[1] A document accessible from http://www.metodii.com indicates another reversible Russian-Latin transliteration system with some similar features.

(ii) all quadruples $(k_i,\sigma,D(\sigma),k_{1-i})$, where I = 0 and σ is a Latin letter, or I =1 and σ is a Russian letter.

Let T be the mapping accomplished by this transducer. Evidently T is total and single-valued. This mapping will be shown also to be injective, but we prefer to postpone the corresponding proof to the last section.To illustrate the action of this mapping, let us apply it to the string

"**Операционная система** Windows 2000 **создана раньше системы** Windows XP."

The image of this string looks as follows:

"Operacionnaya sistema **'**Windows 2000 **'**sozdana ranjshe sistemih **'**Windows XP."

As a second illustration, we note that the string "**О'Нил** (O'Neill)" has the image "O**''**Nil (**'**O**''**Neill)" under T.

Remark 1. In each of the examples 1, 2 and 3, the mapping T^{-1} is not homomorphic although T is homomorphic and finitary. For instance T^{-1}("sh")={"ш"}, whereas T^{-1}("s")T^{-1}("h") is empty, in any of these examples. However, if a mapping T is accomplished by a sequential transducer S then the corresponding mapping T^{-1} is accomplished by the inverse sequential transducer S^{-1} (cf. Exercise 5 in [2, Section 3.3]). In particular, T^{-1} is accomplished by some sequential transducer in any of the above examples. Therefore the complexity of transliteration and of the inverse transformation can be considered in a uniform way by studying the complexity of using an arbitrary sequential transducer.

Input and Output Tokenizers of a Sequential Transducer

We shall need a notion that is similar to the notion of sequential transducer, but is somewhat simpler. We shall call a *tokenizer* any quadruple (K,Γ,G,s_0), where Γ is an alphabet, K is a finite set (the set of the *states*), s_0 is an element of K (the *start state*), G is a finite set of triples (the *moves*) with second components in Γ* and first and third components in K. For any t_0 in K, we shall call a *path of* (K,Γ,G,s_0) *starting at* t_0 any finite sequence

$$t_0,\psi_1,t_1,\psi_2,t_2,\ldots,t_{m-2},\psi_{m-1},t_{m-1},t_{m-1},\psi_m,t_m \qquad (1)$$

such that the triples (t_0,ψ_1,t_1), (t_1,ψ_2,t_2), …, $(t_{m-2},\psi_{m-1},t_{m-1})$, (t_{m-1},ψ_m,t_m) belong to G (the case of m=0, i.e. of the one-term sequence consisting only of t_0, is also admitted). The string $\psi_1\psi_2\ldots\psi_{m-1}\psi_m$ will be called *the result* of this path (in the case of m=0 the result is empty). A path of (K,Γ,G,s_0) starting at the state s_0 will be called a *tokenization by* (K,Γ,G,s_0) of its result.

To any sequential transducer (K,Σ,Δ,H,s_0) two tokenizers will be made to correspond — its *input tokenizer* (K,Σ,H_1,s_0) and its *output tokenizer* (K,Δ,H_2,s_0), where H_1 and H_2 consist of the triples (k,ω,k') and (k,τ,k'), respectively, corresponding to the quadruples (k,ω,τ,k') in H. The input and the output tokenizers of any sequential transducer S will be denoted by IN S and OUT S, respectively.

Example 6. The strings "**тайна**", "**рай**" and "**район**" have, respectively, the following tokenizations by the tokenizer IN $S_{4,1}$:

k_0,"**т**",k_0,"**а**",k_0,"**й**",k_1,"**н**",k_0,"**а**",k_0,

k_0,"**р**",k_0,"**а**",k_0,"**й**",k_1,

k_0,"**р**",k_0,"**а**",k_0,"**йо**", k_0,"**н**",k_0.

Example 7. T he string “O’’Nil (’O’’Neill)” has the following tokenization by OUT S_5:

k_0,“O”,k_0,“’”,k_0,“N”,k_0,“i”,k_0,“l”,k_0,“ ”,k_0,“(”,k_0,“’O”,“’’”,k_1,“N”,k_1,“e”,k_1,“i”,k_1,“l”,k_1,“l”,k_1,“)”,k_1.

Next statement is obvious, and it indicates a way for applying the input and output tokenizers to the problems of single-valuedness and injectivity of the mapping accomplished by a sequential transducer.

Main sufficient conditions for single-valuedness and for injectivity. Let T be the mapping accomplished by a sequential transducer (K,Σ,Δ,H,s_0). Then T is surely single-valued if the following two conditions are satisfied:

(i) no string in Σ^* has two different tokenizations by IN(K,Σ,Δ,H,s_0);

(ii) for any s, s' in K and any ω in Σ^* there is at most one τ in Δ^* such that (s,ω,τ,s') belongs to H.

The mapping T is surely injective if the following two conditions are satisfied:

(iii) no string in Δ^* has two different tokenizations by OUT(K,Σ,Δ,H,s_0);

(iv) for any s, s' in K and any τ in Δ^* there is at most one ω in Σ^* such that (s,ω,τ,s') belongs to H.

The condition (i) is clearly satisfied in the case when (K,Σ,Δ,H,s_0) is the one-state sequential transducer that corresponds to a finitary homomorphic mapping of Σ^* into $P(\Delta^*)$, and the condition (ii) is equivalent in this case to the non-existence of σ in Σ with more than one element in $T(\sigma)$. In particular, both conditions are satisfied for the sequential transducers S_1 and S_3. These conditions are satisfied also for the sequential transducers $S_{4,1}$, $S_{4,2}$ and S_5, but the verification of (i) needs some care for the first of them. The conditions (iii) and (iv) are satisfied for all considered concrete sequential transducers S_1, S_2, S_3, $S_{4,1}$, $S_{4,2}$, S_5, however the verification of (iii) is somewhat cumbersome in all these cases.

The considerations below can make all above-mentioned verifications easier, except the one for the sequential transducer S_3 (but it corresponds just to the example of transliteration with a more difficult inverse transformation).

We shall introduce the left-to-right longest-match strategy (LRLMS) as an algorithm for transforming strings into tokenizations of them. Let a tokenizer (K,Γ,G,s_0) and a string θ from Γ^* be given. We shall consider a partial operation on the tokenizations by (K,Γ,G,s_0) of proper prefixes of θ, namely if

$$s_0,\varphi_1,s_1,\varphi_2,s_2,\ldots,s_{k-2},\varphi_{k-1},s_{k-1},\varphi_k,s_k \qquad (2)$$

is such a tokenization then we look for a triple (s_k,φ,s) from G such that $\varphi_1\varphi_2\ldots\varphi_{k-1}\varphi_k\varphi$ is a prefix of θ with the maximal possible length, and if there is exactly one such triple then we append its components φ and s to the sequence (2). The following algorithm will be called *the LRLMS-algorithm*: given a string θ from Γ^*, we start with the one-term sequence consisting only of s_0, and we apply the above-mentioned partial operation until a tokenization of θ is obtained or no further application of the operation is possible. A termination of this process is considered as successful if a tokenization of θ is obtained.

A tokenization (2) by the tokenizer (K,Γ,G,s_0) will be said to be *compliant with the left-to-right longest-match strategy* (*LRLMS-compliant*, for short) if this tokenization can be obtained by applying to its result the LRLMS-algorithm. The condition is trivially satisfied if $k=0$, and for $k\neq0$ it is equivalent to the following requirement: φ_k is non-empty, and there are no I in $\{1,\ldots,k\}$ and no triple (s_{i-1},φ,t) from G distinct from (s_{i-1},φ_i,s_i) such that φ is a prefix of $\varphi_i\varphi_{i+1}\ldots\varphi_{k-1}\varphi_k$, and φ_i is a prefix of φ.

Example 8. The two sequences below are tokenizations by OUT S_3 (of the strings "suhhoyy" and "ishhod", respectively), but the first one is LRLMS-compliant, whereas the second one is not (due to the prefix "sh" in the string "shhod"):

$$s_0,\text{“s”},s_0,\text{“u”},s_0,\text{“hh”},s_0,\text{“o”},s_0,\text{“yy”},s_0,$$
$$s_0,\text{“i”},s_0,\text{“s”},s_0,\text{“hh”},s_0,\text{“o”},s_0,\text{“d”},s_0.$$

Remark 2. If there are no triples in G with empty second components, and we apply the LRLMS-algorithm to some string θ from Γ* that has no LRLMS-compliant tokenization, then the application of the LRLMS-algorithm terminates unsuccessfully. For instance, if (K,Δ,G,s_0) is OUT S_3,.then the application of the algorithm to the string "ishhod" terminates unsuccessfully at the sequence $s_0,\text{“i”},s_0,\text{“sh”},s_0$.

A tokenizer will be called *compliant with the left-to-right longest-match strategy* (*LRLMS-compliant*, for short) if all tokenizations by this tokenizer are LRLMS-compliant. Of course this implies the non-existence of two distinct tokenizations of one and the same string. It is easy to check that all considered concrete sequential transducers S_1, S_2, S_3, $S_{4,1}$, $S_{4,2}$, S_5 have input tokenizers that are LRLMS-compliant (the situation is not completely obvious only in the case of the sequential transducer $S_{4,1}$). Example 8 shows that OUT S_3 is not LRLMS-compliant. However, the output tokenizers of all other sequential transducers in question are LRLMS-compliant. This will be verified in the last section by means of corollaries of the necessary and sufficient condition below, where a state of a tokenizer is called *accessible* if it is the last term of some tokenization by this tokenizer.[1]

Necessary and sufficient condition for LRLMS-compliance of a tokenizer. A tokenizer (K,Γ,G,s_0) is LRLMS-compliant if and only if it has the following properties:

(i) there is no triple in G with accessible first component and empty second one;

(ii) no two triples in G with accessible first component exist that differ from one another only in their third components;

(iii) for any (t_0,φ,t) in G with accessible t_0 there is no path (1) in (K,Γ,G,s_0) with $m>1$ such that $\psi_1\psi_2\ldots\psi_{m-1}$ is a proper prefix of φ and φ is a prefix of $\psi_1\psi_2\ldots\psi_{m-1}\psi_m$.

Proof. Let (K,Γ,G,s_0) be an arbitrary tokenizer. For the proof of the necessity, suppose (K,Γ,G,s_0) is LRLMS-compliant. Let t_0 be an accessible element of K, and (2) be a tokenization by (K,Γ,G,s_0) such that $s_k=t_0$. There is no triple (t_0,φ,t_1) in G with empty φ – otherwise

$$s_0,\varphi_1,s_1,\varphi_2,s_2,\ldots,s_{k-2},\varphi_{k-1},s_{k-1},\varphi_k,t_0,\varphi,t_1 \qquad (3)$$

would be a tokenization that is not LRLMS-compliant. There are also no φ in Γ* and distinct t and t_1 in K such that both (t_0,φ,t) and (t_0,φ,t_1) belong to G – otherwise again (3) would be a tokenization that is not LRLMS-compliant. Finally, it is not possible that there are (t_0,φ,t) in G and a path (1) in (K,Γ,G,s_0) with $m>1$ such that $\psi_1\psi_2\ldots\psi_{m-1}$ is a proper prefix of φ and φ is a prefix of $\psi_1\psi_2\ldots\psi_{m-1}\psi_m$ – this would contradict the LRLMS-compliance of the tokenization

$$s_0,\varphi_1,s_1,\varphi_2,s_2,\ldots,s_{k-2},\varphi_{k-1},s_{k-1},\varphi_k,t_0,\psi_1,t_1,\psi_2,t_2,\ldots,t_{m-2},\psi_{m-1},t_{m-1},\psi_m,t_m$$

[1] All concrete tokenizers we mentioned have only accessible states. On the other hand, from an arbitrary tokenizer we can get one having only accessible states by elimination of the states that are not accessible and of the moves that contain such states as first or third components. It is easy to see that the reduction in question will not affect the set of the tokenizations by the tokenizer.

(since ψ_1 is a proper prefix of φ). For proving the sufficiency, suppose (K,Γ,G,s_0) is not LRLMS-compliant. Then some tokenization (2) by (K,Γ,G,s_0) is not LRLMS-compliant, hence $k\neq 0$ and either φ_k is empty or there are some l in {1, …, k} and some triple (s_{i-1},φ,t) from G distinct from (s_{i-1},φ_i,s_i) such that φ_i is a prefix of φ and φ is a prefix of $\varphi_i\varphi_{l+1}\ldots\varphi_{k-1}\varphi_k$. In the first case the triple (s_{k-1},φ_k,s_k) violates (i). In the second case, either $\varphi=\varphi_i$, and then the pair of triples (s_{i-1},φ,t) and (s_{i-1},φ_i,s_i) violates (ii), or φ_i is a proper prefix of φ. If φ_i is a proper prefix of φ, then $k>i$ and there is some j in {l+1, …,k−1,k} such that φ is a prefix of $\varphi_i\varphi_{l+1}\ldots\varphi_{j-1}\varphi_j$, whereas $\varphi_i\varphi_{l+1}\ldots\varphi_{j-1}$ is a proper prefix of φ. In this case we can violate (iii) by setting $m=j-l+1$, $t_0=s_{i-1}$ and $t_r=s_{l-1+r}$, $\psi_r=\varphi_{l-1+r}$ for r=1,…,m. ∎

Corollary 1. Let (K,Γ,G,s_0) be a tokenizer such that no triple from G has an empty second component and there are no two triples in G that differ from one another only in their third components. Let no triples (t_0,φ,t), (t_0,ψ_1,t_1) and (t_1,ψ_2,t_2) exist in G such that ψ_1 is a proper prefix of φ and some of the strings $\psi_1\psi_2$ and φ is a prefix of the other one. Then (K,Γ,G,s_0) is LRLMS-compliant.

Proof. Suppose there are a triple (t_0,φ,t) in G and a path (1) in (K,Γ,G,s_0) with $m>1$ such that $\psi_1\psi_2\ldots\psi_{m-1}$ is a proper prefix of φ and φ is a prefix of $\psi_1\psi_2\ldots\psi_{m-1}\psi_m$. Then ψ_1 is also a proper prefix of φ, and some of the strings $\psi_1\psi_2$ and φ is a prefix of the other one. Since (t_0,ψ_1,t_1) and (t_1,ψ_2,t_2) belong to G, this is a contradiction. ∎

Next corollary is actually a particular instance of Corollary 1.

Corollary 2. Let $(\{s_0\},\Gamma,G,s_0)$ be a (one-state) tokenizer, and let W be the set of the second components of the triples from G. Suppose that all strings from W are non-empty, and there are no φ, ψ_1 and ψ_2 in W such that ψ_1 is a proper prefix of φ and some of the strings $\psi_1\psi_2$ and φ is a prefix of the other one. Then $(\{s_0\},\Gamma,G,s_0)$ is LRLMS-compliant.

Some concrete applications

Suppose two alphabets Σ and Δ are given, and T is the mapping of Σ* into P(Δ*) describing a given transliteration system. We shall call this transliteration system *easily usable* if T is an injective mapping that can be accomplished by some sequential transducer with LRLMS-compliant input and output tokenizers. The transliteration systems mentioned in Examples 1, 2, 4 and 5 are easily usable in the above sense, and this will be shown by verifying that any of the sequential transducers S_1, S_2, $S_{4,1}$, $S_{4,2}$, S_5 has LRLMS-compliant input and output tokenizers and satisfies item (iv) from the main sufficient conditions for single-valuedness and for injectivity (of course it would be enough to do this for one of the sequential transducers $S_{4,1}$ and $S_{4,2}$ instead of doing it for both of them).

The verification of (iv) for each of the above-mentioned sequential transducers is almost immediate (even in the case of S_5, although S_5 has moves with distinct second components and one and the same third one – for instance the quadruples (k_0,"**Д**","D",k_0) and (k_1,"D","D",k_1) or the quadruples (k_0,"D","'D",k_1) and (k_1,"**Д**","'D",k_0)).

By their construction, the sequential transducers in question have no moves with empty second or third components, hence the corresponding input and output tokenizers have no moves with empty second components.

The LRLMS-compliance of the input tokenizers of the considered sequential tokenizers was already characterized as more or less obvious. However, there is no problem to verify it also by means of Corollary 1 or (for the case of S_1 and S_2) Corollary 2. For S_1, S_2, $S_{4,2}$ and S_5 the verification is trivial thanks to the fact that all

second components of their moves have length 1. Now let us suppose that (t_0,φ,t), (t_0,ψ_1,t_1) and (t_1,ψ_2,t_2) are moves of the tokenizer IN $S_{4,1}$ such that ψ_1 is a proper prefix of φ and some of the strings $\psi_1\psi_2$ and φ is a prefix of the other one. Then (t_0,φ,t) must have the form $(k_0,\sigma\sigma_1,k_0)$, where σ is "**Й**" or "**й**", and σ_1 is a Cyrillic vowel or a soft sign, hence ψ_1 is "**Й**" or "**й**", t_1 is k_1, and ψ_2 begins with a Cyrillic vowel or a soft sign. From the fact that t_1 is k_1 the conclusion follows that ψ_2 is a symbol of Σ which is neither a Cyrillic vowel nor a soft sign, and this is a contradiction.

The LRLMS-compliance of OUT S_1 and OUT S_2 can be shown again by means of Corollary 2. Since all moves of S_1 are also moves of S_2, it would be sufficient to check the assumption of Corollary 2 only for OUT S_2. Suppose the set W for this tokenizer has elements φ, ψ_1 and ψ_2 such that ψ_1 is a proper prefix of φ and some of the strings $\psi_1\psi_2$ and φ is a prefix of the other one. The assumptions that φ and ψ_1 belong to W and ψ_1 is a proper prefix of φ imply the equality $\varphi = \psi_1$"h". From here, taking into account also the assumptions that ψ_2 belongs to W and some of the strings $\psi_1\psi_2$ and φ is a prefix of the other one, we get a contradiction by firstly concluding that ψ_2 begins with "h".

As to the LRLMS-compliance of the output tokenizers of $S_{4,1}$, $S_{4,2}$ and S_5, it can be shown by means of Corollary 1. It is straightforward (although somewhat tedious) to see that no of these tokenizers has two moves differing from one another only in their third components. Now suppose for some of this tokenizers the existence of moves (t_0,φ,t), (t_0,ψ_1,t_1) and (t_1,ψ_2,t_2) such that ψ_1 is a proper prefix of φ and some of the strings $\psi_1\psi_2$ and φ is a prefix of the other one. We shall consider the cases of $S_{4,1}$, $S_{4,2}$ and S_5 one by one. By reasoning in two steps, in any of these three cases we shall get a contradiction to the assumption that (t_1,ψ_2,t_2) is a move of the corresponding output tokenizer – in the first step we shall make some conclusions from the assumptions that (t_0,φ,t), (t_0,ψ_1,t_1) are moves of the tokenizer in question and ψ_1 is a proper prefix of φ, whereas in the second step we shall take into account also that some of the strings $\psi_1\psi_2$ and φ is a prefix of the other one.

In the case of $S_{4,1}$ we conclude in the first step that either the first symbol in φ after its prefix ψ_1 is "h" or we have the equalities $t_1 = k_1$, $\varphi=\psi_1\sigma$, where σ is some of the letters "a", "o", "u" or an apostrophe. Making use of this conclusion, we get a contradiction in the second step by inferring that either the string ψ_2 begins with "h" or in the presence of the equality $t_1 = k_1$ this string begins with some of the letters "a", "o", "u" or with an apostrophe.

The first step in the case of $S_{4,2}$ is to conclude that we have either the equalities $t_1=k_0$, $\varphi=\psi_1$"h" or the equalities $t_1 = k_1$, $\varphi=\psi_1\sigma$, where σ is some of the letters "a", "o", "u". We get a contradiction in the second step by inferring that ψ_2 begins with "h" in the presence of the equality $t_1=k_0$ or with some of the letters "a", "o", "u" in the presence of the equality $t_1 = k_1$.

In the case of S_5 we firstly conclude that $t_1=k_0$, $\varphi=\psi_1$"h", and then we get a contradiction by inferring that ψ_2 begins with "h".

Remark 3. Instead by reasoning as above, each of the considered conditions could be verified by straightforward inspection of all finitely many possible cases. Of course, it would be better to do this by using some appropriate computer program.

Remark 4. As we observed (by using Example 8), the tokenizer OUT S_3 is not LRLMS-compliant. This statement can be strengthened as follows: the mapping accomplished by S_3 cannot be accomplished at all by a sequential transducer with a LRLMS-compliant output tokenizer (hence the transliteration indicated in Example 3 is not easily usable in our sense). In fact, if we suppose that such other sequential transducer can be constructed then we can get a contradiction by considering the application of the LRLMS-algorithm for the corresponding output tokenizer to strings consisting of one "s" followed by arbitrarily many "h". Namely, we can show then the existence

of a non-empty string in Σ* whose image will be a prefix of all sufficiently long strings of the above-mentioned form, and obviously such a string cannot exist.

Acknowledgments

The author thanks V. A. Uspensky and M. R. Pentus for some very useful discussions and very helpful information.

Bibliography

[1] Gerdemann, D., van Noord, G. Transducers from rewrite rules with backreferences. In: Ninth Conference of the European Chapter of the Association for Computational Linguistics (8-12 June 1999, Univ. of Bergen, Bergen, Norway). San Francisco, Morgan Kaufmann Publishers, 1999, 126-133. http://acl.ldc.upenn.edu/E/E99/

[2] Ginsburg, S. The Mathematical Theory of Context-Free Languages. McGraw-Hill, 1966.

[3] Lothaire, M. Algebraic Combinatorics on Words, Cambridge University Press, 2002.

[4] Markov, A. A., Nagorny, N. M. The Theory of Algorithms. Dordrecht etc., Kluwer Academic Publishers, 1988.

[5] Sardinas, A., Patterson, C. A necessary and sufficient condition for the unique decomposition of coded messages. IRE Intern. Conv. Record, 8:104-108, 1953.

[6] Марков, А. А. Теория алгорифмов. Труды Математ. инст. им. В. А. Стеклова, 42, Москва-Ленинград, Изд. АН СССР, 1954.

[7] Успенский, В. А. К проблеме транслитерации русских текстов латинскими буквами. In: Научно-техническая информация, серия 2, Информационные процессы и системы. 1967, № 7, 12-20 (also in [9], 390-412).

[8] Успенский, В. А. Невто́н-Ньюто́н-Нью́тон, или Сколько сторон имеет языковой знак? In: Русистика. Славистика. Индоевропеистика. Сборник к 60-летию Андрея Анатольевича Зализняка. Москва, "Индрик", 1996, 598-659 (also in [9], 483-561).

[9] Успенский, В. А. Труды по нематематике. Москва, ОГИ, 2002. http://www.mccme.ru/free-books/usp.htm

Author's Information

Dimiter Skordev – Sofia University, Faculty of Mathematics and Informatics, blvd. J. Bourchier 5, Sofia 1164, Bulgaria; e-mail: skordev@fmi.uni-sofia.bg