# Learning Concatenations of Locally Testable Languages from Positive Data

Satoshi Kobayashi Takashi Yokomori
Department of Computer Science and Information Mathematics
The University of Electro-Communications

1-5-1, Chofugaoka, Chofu, Tokyo 182, Japan
e-mail:{satoshi,yokomori}@cs.uec.ac.jp

**Abstract**

This paper introduces the class of concatenations of locally testable languages and its subclasses, and presents some results on the learnability of the classes from positive data. We first establish several relationships among the language classes introduced, and give a sufficient condition for a concatenation operation to preserve finite elasticity of a language class $\mathcal{C}$. Then we show that, for each $k$, the class $CLT^{\leq k}$, a subclass of concatenations of locally testable languages, is identifiable in the limit from positive data. Further, we introduce a notion of *local parsability*, and define a class $(k,l)$-$CLTS$, which is a subclass of the class of concatenations of strictly locally testable languages. Then, for each $k, l \geq 1$, $(k,l)$-$CLTS$ is proved to be identifiable in the limit from positive data using reversible automata with the conjectures updated in polynomial time. Some possible applications of this result are also briefly discussed.

## 1 Introduction

Inductive inference is a process of acquiring a concept from its examples. This process was formulated by Gold as a process of identifying a target concept in the limit, which is called Gold's idetification in the limit [Gol67]. He also showed that *a superfinite class*, i.e., a class which contains all finite concepts and at least one infinite concept, is not identifiable in the limit from positive data only, which was shocking to us because it leads us to the negative result on the learnability of the class of regular languages from positive data.

On the other hand, we have also known some interesting classes of languages, $k$-reversible languages [Ang82], pattern languages [Ang80], etc., which are identifiable in the limit from positive data. However, as is mentioned in [Ang82], further research on the learnability from positive data for subclasses of regular languages remains open to be studied. In particular, [Ang82] refers to a possibility of close relationships between *noncounting languages* and *reversible languages*, and suggests that a certain synthetic approach to learning these two language classes might give some useful results for analyzing subclasses of regular languages learnable from positive data. Recently, in [Yok90], Yokomori has shown results on the learnability of the class of strictly locally testable languages from positive data and presents an interesting relationship between strictly $k$-testable languages and $(k+1)$-reversible languages.

This paper introduces some subclasses of noncounting languages, and investigates relationships among those classes and the class of reversible languages. Further, we present some learnability results on the classes. In section 2, we introduce the class of concatenations of locally testable languages $CLTS$ and its subclasses, and then we compare them with the class of reversible languages in section 3. Section 4 presents theoretical results on the learnability of $CLTS$ and its subclasses. Especially we show that the class $(k,l)$-$CLTS$, which is a subclass of $CLTS$ with *local parsability*, is identifiable in the limit using $(k+2l)$-reversible automata with the conjectures updated in polynomial time. Some possible applications of the results are also briefly discussed.

## 2 Concatenations of Locally Testable Languages

Let $\Sigma$ be a finite alphabet and $\Sigma^*$ be the set of all finite length strings over $\Sigma$. Let $\Sigma^k$ be the set of all strings over $\Sigma$ of length $k$. We denote the null string by $\lambda$. $\Sigma^+$ is defined as $\Sigma^* - \lambda$. The length of a string $w \in \Sigma^*$ is denoted by $\mid w \mid$. Please do not confuse it with the notation $\mid S \mid$ for a set $S$, which represents the cardinality of $S$. *A language* is a subset of $\Sigma^*$. In this section, we consider only non-null languages. Therefore, in this section, we assume that a language over $\Sigma$ is a subset of $\Sigma^+$. A concatenation of languages, $L_1 \cdot L_2$, is defined as a set of strings $\{w_1w_2 \mid w_1 \in L_1, w_2 \in L_2\}$. $L_k(w)$ and $R_k(w)$ are defined as the $k$-length prefix and $k$-length suffix of $w$, respectively. These notations are defined only when $w$ has length $k$ or more. Further, let $I_k(w)$ be the set of all interior substrings of length k. Note that, for any string $w$ with $\mid w \mid \leq k+1$, it holds that $I_k(w) = \emptyset$, where $\emptyset$ denotes an empty set.

Then, we define the class of locally testable languages as follows [MP71]. Let $k$ be a positive integer. A language $L$ over $\Sigma$ is *k-testable* iff for all strings, $w_1, w_2$, of length $k$ or more, if $L_k(w_1) = L_k(w_2)$, $R_k(w_1) = R_k(w_2)$ and $I_k(w_1) = I_k(w_2)$, then either $w_1$ and $w_2$ are in $L$ or neither are. A language $L$ is *locally testable* iff $L$ is $k$-testable for some positive integer $k$. The class of $k$-testable languages and the class of locally testable languages are denoted by $LT^{=k}$ and $LT$, respectively. We denote $\cup_{i \leq k} LT^{=i}$ by $LT^{\leq k}$.

The definition of $k$-testable languages says nothing about strings of length less than $k$. So, a $k$-testable language may include any subset of strings of length less than $k$.

For any positive integer $k$, a language $L$ over $\Sigma$ is said to be *strictly k-testable* iff there exist finite sets $A$, $B$, and $C$ such that $A, B, C \subseteq \Sigma^k$, and for any string $w$ with $\mid w \mid \geq k$, $w \in L$ iff $L_k(w) \in A$, $R_k(w) \in B$, and $I_k(w) \subseteq C$. Here,

$< A, B, C >$ is called *a triple for* $L$ and denoted by $triple(L)$. A language $L$ is *strictly locally testable* iff $L$ is strictly $k$-testable for some positive integer $k$. We denote the class of strictly $k$-testable languages and the class of strictly locally testable languages by $LTS^{=k}$ and $LTS$, respectively. The class $\cup_{i \le k} LTS^{=i}$ is denoted by $LTS^{\le k}$.

**Theorem 1** [MP71]

**(1)** The class of locally testable languages (k-testable languages) is closed under the Boolean operations.

**(2)** The class of strictly locally testable languages (strictly k-testable languages) is closed under intersection.

**(3)** The class of locally testable languages (k-testable languages) is the closure of that of strictly locally testable languages (strictly k-testable languages) under the Boolean operations.

**Example 1** Let us consider a strictly 2-testable language $L$ over $\Sigma = \{a, b\}$, for which $< \{aa\}, \{bb\}, \{aa, ab, bb\} >$ is a triple. This language is also denoted by a regualr exspression $aaa^*bbb^*$. Here we can easily show that $L$ is a strictly 3-testable language for which $< \{aaa, aab\}, \{bbb, abb\}, \{aaa, aab, abb, bbb\} >$ is a triple. At first thought, it seems to hold that for any positive integer $k$, $LTS^{=k}$ and $LT^{=k}$ are contained in $LTS^{=k+1}$ and $LT^{=k+1}$, respectively. However, this is not the case. For example, let us consider a strictly $k$-testable language $L' = \{a^k, a^{k+1}\}$ for which $< \{a^k\}, \{a^k\}, \emptyset >$ is a triple. Then, it holds that $L'$ is not in $LT^{=k+1}$ because for $w_1 = a^{k+1} \in L'$ and $w_2 = a^{k+2} \notin L'$, we have $L_{k+1}(w_1) = L_{k+1}(w_2)$, $R_{k+1}(w_1) = R_{k+1}(w_2)$, and $I_{k+1}(w_1) = I_{k+1}(w_2)(= \emptyset)$. Therefore, in general it holds that $LTS^{=k} \not\subseteq LT^{=k+1}$. Further, by Theorem 1, we have $LT^{=k} \not\subseteq LT^{=k+1}$ and $LTS^{=k} \not\subseteq LTS^{=k+1}$. □

Here, let us consider a slightly different definition of locally testable language as follows. In this setting, we say that a language $L$ is $k$-testable iff for all strings, $w_1, w_2$, of length $\underline{k+1}$ or more, if $L_k(w_1) = L_k(w_2)$, $R_k(w_1) = R_k(w_2)$ and $I_k(w_1) = I_k(w_2)$, then either $w_1$ and $w_2$ are in $L$ or neither are. The difference between this definition and the original one is underlined. In this definition, we can prove that $LT^{=k}$ is contained in $LT^{=k+1}$ in the following manner.

Let $L$ be a language in $LT^{=k}$ and $w_1, w_2$ be strings of length $k+2$ or more such that $L_{k+1}(w_1) = L_{k+1}(w_2)$, $R_{k+1}(w_1) = R_{k+1}(w_2)$, and $I_{k+1}(w_1) = I_{k+1}(w_2)$. It suffices to show $w_1 \in L$ iff $w_2 \in L$.

First, we prove $L_k(w_1) = L_k(w_2)$, $R_k(w_1) = R_k(w_2)$ and $I_k(w_1) = I_k(w_2)$. It is easy to see $L_k(w_1) = L_k(w_2)$ and $R_k(w_1) = R_k(w_2)$. For proving $I_k(w_1) = I_k(w_2)$, we consider the next two cases.

In case $| w_1 | \ge k + 3$, $I_{k+1}(w_1) = I_{k+1}(w_2)$ immediately implies $I_k(w_1) = I_k(w_2)$.

In case $| w_1 | = k + 2$, we have $I_{k+1}(w_2) = I_{k+1}(w_1) = \emptyset$. Therefore, $| w_2 | = k + 2$ holds. Let $aw$ be $L_{k+1}(w_1) (= L_{k+1}(w_2))$, where $a \in \Sigma$ and $w \in \Sigma^*$. Then we have $I_k(w_1) = \{w\} = I_k(w_2)$.

Hence, in any case, we have $I_k(w_1) = I_k(w_2)$.

Therefore, it holds that $w_1 \in L$ iff $w_2 \in L$, since $L \in LT^{=k}$. This implies that $L$ is a $(k+1)$-testable language.

As disscussed above, if we use the new definition, then we have interesting inclusion properties among the classes of $k$-testable languages. However, in the rest of the paper, we restrict the attention to the original definition of locally testable languages.

Let us consider the class of concatenations of locally testable languages. For any class of languages $\mathcal{C}$, we denote by $Con(\mathcal{C})$ the class of languages which is the smallest class of languages that includes $\mathcal{C}$ and is closed under concatenation. Then, by $CLT$, $CLT^{=k}$, $CLT^{\leq k}$, $CLTS$, $CLTS^{=k}$, and $CLTS^{\leq k}$, we denote $Con(LT)$, $Con(LT^{=k})$, $Con(LT^{\leq k})$, $Con(LTS)$, $Con(LTS^{=k})$, and $Con(LTS^{\leq k})$, respectively.

**Example 2** Let us consider languages $L_1$ and $L_2$ over $\Sigma = \{a, b, c\}$, which are denoted by regular expressions $(a+b)(a+b)^*$ and $(b+c)(b+c)^*$, respectively. It is easy to see that $L_1$ and $L_2$ are strictly 1-testable languages such that $triple(L_1) =< \{a,b\}, \{a,b\}, \{a,b\} >$, $triple(L_2) =< \{b,c\}, \{b,c\}, \{b,c\} >$. Let $L_3 = L_1 \cup L_2$. Then, by Theorem 1, we have that $L_3$ is 1-testable, so $L_3 \in LT$. Please note that $L_3$ is $k$-testable for any positive integer $k$ since both $L_1$ and $L_2$ are strictly $k$-testable for any positive integer $k$.

However, we can prove that $L_3 \notin CLTS$. Let us assume $L_3 \in CLTS$. Then there exist some positive integer $n$ and a sequence of strictly locally testable languages $S_1, S_2, ..., S_n$ such that $L_3 = S_1 \cdot S_2 \cdot \cdots \cdot S_n$ and $S_i$ is strictly $k_i$-testable for some positive integer $k_i$. Here we have $n = 1$ since $a \in L_3$. (Recall we consider only non-null languages.) Let $< A, B, C >$ be a triple for $S_1 = L_3$. Then, since $a^{k_1+1}b^{k_1+1} \in L_3$ and $b^{k_1+1}c^{k_1+1} \in L_3$, it holds that $a^{k_1}, b^{k_1} \in A$, $b^{k_1}, c^{k_1} \in B$, and $0 \leq \forall j \leq k_1 (a^{k_1-j}b^j \in C \wedge b^{k_1-j}c^j \in C)$. Therefore, $a^{k_1}b^{k_1}c^{k_1} \in L_3$, which is a contradiction.

Let $L_4 = L_1 \cdot L_2 \cdot L_1$. Then, $L_4$ is in $CLTS$ by its definition. Please note that $L_4$ is in $CLTS^{=k}$ for any positive integer $k$. However, we can prove $L_4 \notin LT$ as follows.

Let us assume that $L_4 \in LT$. Then there exists some positive integer $k$ such that $L_4$ is $k$-testable. Here we have $w_1 = a^{k+1}ca^k \in L_4$, $w_2 = a^kca^kca^k \notin L_4$. It is easy to see that $L_k(w_1) = L_k(w_2)$, $R_k(w_1) = R_k(w_2)$, and $I_k(w_1) = I_k(w_2)$ hold. This is a contradiction. □

From the discussion above, we have the next lemma.

**Lemma 1** **(1)** There exists a language $L$ such that, for any positive integer $k$, $L \in LT^{=k}$ and $L \notin CLTS$.

**(2)** There exists a language $L$ such that, for any positive integer $k$, $L \in CLTS^{=k}$ and $L \notin LT$.

Then, we have the followings.

**Theorem 2** **(1)** $CLTS$, $CLTS^{=k}$, and $CLTS^{\leq k}$ are incomparable to $LT$.

(2) $CLTS(CLTS^{=k}, CLTS^{\leq k})$ properly includes $LTS(LTS^{=k}, LTS^{\leq k}$, respectively).

(3) $LT(LT^{=k}, LT^{\leq k})$ properly includes $LTS(LTS^{=k}, LTS^{\leq k}$, respectively).

(4) $CLT(CLT^{=k}, CLT^{\leq k})$ properly includes $LT(LT^{=k}, LT^{\leq k}$, respectively).

(5) $CLT(CLT^{=k}, CLT^{\leq k})$ properly includes $CLTS(CLTS^{=k}, CLTS^{\leq k}$, respectively).

In this paper, we define noncounting languages by using the notion of locally testable languages. The class $NC$ of *noncounting languages* is defined as the smallest class of languages that contains $LT$ and is closed under the Boolean operations and concatenation. Therefore, all of the language classes introduced in this section are subclasses of $NC$.

# 3 Comparison with Reversible Languages

In this section, we compare the classes of languages introduced in section 2, with the class of reversible languages which is identifiable in the limit from positive data [Ang82].

Here we give the definition of reversible languages based on the language-theoretic characterization [Ang82]. Let $k$ be a non-negative integer. A language $L$ is *k-reversible* iff whenever $u_1vw$ and $u_2vw$ are in $L$ and $\mid v \mid = k$, it holds that for any $x \in \Sigma^*$, $u_1vx \in L$ iff $u_2vx \in L$. (In case $k = 0$, we say $L$ is *zero-reversible* rather than *0-reversible*.) A language $L$ is said to be *reversible* iff $L$ is $k$-reversible for some non-negative integer $k$. The class of $k$-reversible languages and the class of reversible languages are denoted by $Rev(k)$ and $Rev$, respectively.

Then we have the following.

**Lemma 2** [Ang82] For any non-negative integer $k$, $Rev(k)$ is properly contained in $Rev(k+1)$.

**Example 3** The language denoted by a regular expression $(bb)^+$ is zero-reversible. However, this language is not contained in $NC$. (cf. [MP71], p.6)

Let $L_1$ and $L_2$ be strictly 1-testable languages such that $triple(L_1) = \,< \{a\}, \{a\}, \{a\} >$ and $triple(L_2) = \,< \{c\}, \{c\}, \{a\} >$. Then, by the definition, $L_3 = L_1 \cdot L_2 \in CLTS$ holds. Please note that $L_3$ is in $CLTS^{=k}$ for any positive integer $k$. However, we can prove that $L_3 \notin Rev$ as follows.

Let us assume that $L_3$ is $k$-reversible for some non-negative integer $k$. Then, $aa^kc \in L_3$, $aca^kc \in L_3$, and $aa^kcac \in L_3$ hold. Therefore, by the definition of $k$-reversible language, we have that $aca^kcac \in L_3$, which is a contradiction.

Let $L_4$ and $L_5$ be strictly 1-testable languages such that $triple(L_4) = \,< \{a\}, \{a, b\}, \{a\} >$ and $triple(L_5) = \,< \{c\}, \{a\}, \{a\} >$. Then $L_6 = L_4 \cup L_5$ is in $LT$. Please note that $L_6$ is $k$-testable for any positive integer $k$. We can prove that $L_6$ is not in $Rev$.

Let us assume that $L_6$ is $k$-reversible for some non-negative integer $k$. Then, $a(a)^k a \in L_6$, $c(a)^k a \in L_6$, and $a(a)^k b \in L_6$ hold. Therefore, by the definition of $k$-reversible language, we have that $c(a)^k b \in L_6$, which is a contradiction. □

Using the discussion above, we have the followings.

**Lemma 3** **(1)** There exists a language $L$ such that, for any positive integer $k$, $L \in CLTS^{=k}$, and $L \notin Rev$.

**(2)** There exists a language $L$ such that, for any positive integer $k$, $L \in LT^{=k}$, and $L \notin Rev$.

Further, the next fact is proved.

**Lemma 4** [Yok90] $LTS^{=k}$ is properly contained in $Rev(k{+}1)$.

Therefore, we have the followings.

**Theorem 3** **(1)** The following classes are incomparable to $Rev$.
$NC$, $LT$, $LT^{=k}$, $LT^{\leq k}$, $CLTS$, $CLTS^{=k}$, $CLTS^{\leq k}$, $CLT$, $CLT^{=k}$, and $CLT^{\leq k}$

**(2)** $LTS$ is properly contained in $Rev$.

A part of relationships among the classes of languages introduced in this paper is summarized in Figure 1.

# 4 Learnability Results

## 4.1 Definitions

Here we briefly introduce some fundamental definitions. For more details, please refer to [Gol67], [BB75], [Ang80], and [LZ93].

Let $\mathcal{C}$ be a class of non-empty languages over a fixed alphabet $\Sigma$. Then, we consider *a class of representations* $\mathcal{R}$ for $\mathcal{C}$ with the following properties.

**1.** $\mathcal{R}$ is a recursively enumerable language (over some fixed alphabet).

**2.** For all $L \in C$, there exists $r \in \mathcal{R}$ such that $r$ represents $L$ (denoted by $L(r) = L$).

**3.** There exists a recursive function $f$ such that for all $r \in \mathcal{R}$ and $w \in \Sigma^*$,

$$f(r, w) = \begin{cases} 1 & if\ w \in L(r) \\ 0 & otherwise \end{cases}$$

We say that a class of representation $\mathcal{R}$ is *class preserving with respect to* $\mathcal{C}$ iff $\mathcal{C} = \{L(r) \mid r \in \mathcal{R}\}$ holds. A class of representation $\mathcal{R}$ is said to be *class comprising with respect to* $\mathcal{C}$ iff $\mathcal{C} \subseteq \{L(r) \mid r \in \mathcal{R}\}$ holds.(cf.[LZ93]))

For a given $L \in \mathcal{C}$, *a positive presentation of* $L$ is any infinite sequence $w_1, w_2, w_3, ...$ of strings such that $\forall w \in L\ \exists i(w = w_i)$ and $\forall i(w_i \in L)$. Let $L$ be a given language. We say that an algorithm $A$ *identifies* $L$ *in the limit from positive data using* $\mathcal{R}$ iff for any positive presentation of $L$, the infinite sequence, $r_1, r_2, r_3, ...$, of representations in $\mathcal{R}$ produced by $A$ converges to a representation $r$ such that $L = L(r)$. A class $\mathcal{C}$ of languages is said to be *identifiable in the limit from positive data using* $\mathcal{R}$ iff there exists an algorithm $A$ such that $A$ identifies every language in $\mathcal{C}$ in the limit from positive data using $\mathcal{R}$.

A learning algorithm $A$ is said to be *responsive on* $\mathcal{C}$ iff for any $L \in \mathcal{C}$ and any positive presentation of $L$, $A$ always outputs some conjecture between any consecutive input requests from $A$. An algorithm $A$ *consistently identifies* $\mathcal{C}$ *in the limit from positive data using* $\mathcal{R}$ iff $A$ identifies $\mathcal{C}$ in the limit from positive data using $\mathcal{R}$ and for any representation $r_i$ produced by $A$, the given set of strings $\{w_1, w_2, ..., w_i\}$ is contained in $L(r_i)$. An algorithm $A$ *conservatively identifies* $\mathcal{C}$ *in the limit from positive data using* $\mathcal{R}$ iff $A$ identifies $\mathcal{C}$ in the limit from positive data using $\mathcal{R}$ and for any output $r_i$ $(i \geq 2)$ of $A$, it holds that, if $L(r_{i-1})$ contains the set of given strings $\{w_1, ..., w_i\}$, then $r_i = r_{i-1}$. A class $\mathcal{C}$ of languages is said to be *identifiable in the limit from positive data using* $\mathcal{R}$ *with the conjectures updated in polynomial time* iff there exists some algorithm $A$ which is *responsive* on $\mathcal{C}$ and *consistently* and *conservatively* identifies $\mathcal{C}$ in the limit from positive data using $\mathcal{R}$ with the property that the time used by $A$ for updating conjectures is bounded by some polynomial with respect to the size of given examples up to that point, i.e. $\mid w_1 \mid + \cdots + \mid w_i \mid$.

It is often the case that a class of representations $\mathcal{R}$ with class preserving property may be encoded by the set of positive integers so that each integer $i$ corresponds to the $i$th representation in $\mathcal{R}$. In this case, by $L_i$, we denote the language which is represented by an integer $i$, and an infinite sequence $L_1, L_2, L_3, ...$ is called *an indexed family of recursive languages*, or *an indexed family* for short.[Ang80]

**Note :** *In the sequel, if a representation class* $\mathcal{R}$ *is not specified, we always assume that some appropriate enumerable class of representations with class preserving property is attached to a target concept class.*

## 4.2 Learnability of $CLT^{\leq k}$ from Positive Data

Let $\mathcal{C}$ be an indexed family. We say $\mathcal{C}$ has *finite thickness* iff for any string $w \in \Sigma^*$, the number of languages in $\mathcal{C}$ which contain $w$ is finite. Angluin showed that finite thickness is a sufficient condition for learnability from positive data [Ang80]. Wright introduced another sufficient condition, called *finite elasticity*, for learnability from positive data, originaly in [Wri89], and correctly in [MSW90].

An indexed family $\mathcal{C}$ of languages has *infinite elasticity* iff there exist an infinite sequence $w_0, w_1, w_2, ...$ of strings and an infinite sequence $L_1, L_2, ...$ of languages in $\mathcal{C}$ such that, for any $k \geq 1$, $\{w_0, w_1, ..., w_{k-1}\} \subseteq L_k$ and $w_k \notin L_k$ hold. A class $\mathcal{C}$ has *finite elasticity* iff $\mathcal{C}$ does not have infinite elasticity.

As proved in [Wri89], it holds that, if a class $\mathcal{C}$ has finite thickness, then $\mathcal{C}$ has finite elasticity.

**Theorem 4** [Wri89] An indexed family $\mathcal{C}$ is identifiable in the limit from positive data if $\mathcal{C}$ has finite elasticity.

Since the number of $k$-testable languages on a *fixed* finite alphabet is finite, we immediately obtain the following.

**Lemma 5** $LT^{\leq k}$ has finite thickness, and therefore, finite elasticity.

By Theorem 4 and Lemma 5, we have the following.

**Theorem 5** $LT^{\leq k}$ is identifiable in the limit from positive data.

Now, for an indexed family $\mathcal{C}$, let us consider the learnability of $Con(\mathcal{C})$ from positive data. The next result is useful for proving the learnability of $Con(\mathcal{C})$ from positive data, when $\mathcal{C}$ has finite elasticity.

**Lemma 6** Let us consider an indexed family $\mathcal{C}$ with the following properties.

**(C1)** For any language $L$ in the class, if $\lambda \in L$, then $L = \{\lambda\}$.

**(C2)** The class has finite elasticity.

Then, $Con(\mathcal{C})$ also satisfies the conditions (C1) and (C2).

**Proof**
Let us consider a language $L_1 \cdots L_n$ in $Con(\mathcal{C})$ which contains a null-string. Then, each $L_i$ must contain a null-string. From the condition (C1) of $\mathcal{C}$, since each $L_i = \{\lambda\}$, we have $L_1 \cdots L_n = \{\lambda\}$. Therefore, $Con(\mathcal{C})$ satisfies the condition (C1).

The proof for the claim that $Con(\mathcal{C})$ satisfies the condition (C2) is as follows.

Let us assume that $Con(\mathcal{C})$ has infinite elasticity. Then there exist infinite sequences, $w_0, w_1, w_2 ...$, of strings and $L_1, L_2, ...$ of languages in $Con(\mathcal{C})$ such that $\{w_0, w_1, ..., w_{k-1}\} \subseteq L_k$ and $w_k \notin L_k$ for any positive integer $k$. For any $L_k$ ($k \geq 1$), there is a sequence $L_{k,1}, L_{k,2}, ..., L_{k,l(k)}$ of languages in $\mathcal{C}$ such that $L_k = L_{k,1} \cdot L_{k,2} \cdots L_{k,l(k)}$. Here we may assume that $\forall k \geq 1,\ 1 \leq \forall i \leq l(k)$ ($\lambda \notin L_{k,i}$), since, otherwise, $L_{k,i} = \{\lambda\}$ by the condition (C1) of $\mathcal{C}$, and therefore, $L_{k,i}$ can be removed from the sequence.

Then we construct the following three infinite sequences: $k_0, k_1, k_2, ...$ of non-negative integers, $N_0, N_1, N_2, ...$ of sets of non-negative integers, and $t_{k_0}, t_{k_1}, t_{k_2}, ...$ of tuples of strings, where by $l_i$ we denote the length of $t_{k_i}$. The construction is based on the recursive procedure bellow.

**Initialization :** $i=0, k_0=0, N_{-1}=$the set of all positive integers.

**Stage i :**

For each tuple of strings $t=(s_1,s_2,...,s_p)$ such that $w_{k_i}=s_1s_2...s_p$ and $1\le \forall j\le p\ (s_j\in\Sigma^+)$,
let $A_t=\{k\in N_{i-1}\ \mid\ l(k)=p\wedge 1\le\forall j\le p\ (s_j\in L_{k,j})\}$.

Find $t_{k_i}=(w_{k_i,1},w_{k_i,2},...,w_{k_i,l_i})$ such that $\mid A_{t_{k_i}}\mid=\infty$.

Let $N_i=A_{t_{k_i}}$ and $k_{i+1}=min\{j\in N_i\}$.

Initiallize each $A_t$ to $\emptyset$.

Goto stage $i+1$.

For each tuple $t=(s_1,s_2,...,s_p)$ of strings, $A_t$ represents the set of all indices $i$ of languages $L_i\ (=L_{i,1}\cdot L_{i,2}\cdots L_{i,p})$ such that $s_1\in L_{i,1},s_2\in L_{i,2},...,s_p\in L_{i,p}$.

Here we can prove, for each stage, that there exists some $A_{t_{k_i}}$ such that $\mid A_{t_{k_i}}\mid=\infty$, and it holds that $k_{i+1}>k_i$, $N_i\subseteq N_{i-1}$ and $\mid N_i\mid=\infty$.

The claim is proved by the induction on $i$.

Let us consider the case $i=0$. In this case, we have for all positive integers $k$, $w_{k_0}=w_0\in L_k$. Therefore, for any $k\ge 1$, there exists a tuple of strings $t_k=(s_{k,1},s_{k,2},...,s_{k,l(k)})$ such that $w_{k_0}=w_0=s_{k,1}s_{k,2}\cdots s_{k,l(k)}$ and $1\le\forall j\le l(k)\ (s_{k,j}\in L_{k,j})$. Here we have $1\le\forall j\le l(k)\ (s_{k,j}\ne\lambda)$, since $\lambda\notin L_{k,j}$. Therefore, for each $k\ (\ge 1)$, there exists a tuple $t$ such that $k\in A_t$. The number of tuples $t=(s_1,s_2,...,s_p)$ such that $w_0=s_1s_2...s_p$ and $1\le\forall j\le p\ (s_j\in\Sigma^+)$, is finite. Hence, there exists a tuple $t_{k_0}$ such that $\mid A_{t_{k_0}}\mid=\infty$. Therefore, we have $\mid N_0\mid=\infty$. It is easy to see $N_0\subseteq N_{-1}$ by the definition of $N_0$ and $A_{t_{k_0}}$. From $\forall j\in N_{-1}\ (j\ge 1)$, we have $k_1\ge 1>0=k_0$. Therefore, the claim holds in case $i=0$.

Let us assume the claim holds in case $i=n-1$ and consider the case $i=n$. In a similar manner as in the case $i=0$, we can prove that there exists a tuple of strings $t_{k_n}=(w_{k_n,1},...,w_{k_n,l_n})$ such that $\mid A_{t_{k_n}}\mid=\infty$. Therefore, we have $\mid N_n\mid=\infty$. It is sufficient to show $k_{n+1}>k_n$ and $N_n\subseteq N_{n-1}$. It is easy to see $N_n\subseteq N_{n-1}$ by the definition of $N_n$ and $A_{t_{k_n}}$.

From $w_{k_n}\notin L_{k_n}$, we have that there is no tuples $t=(s_1,...,s_p)$ such that $w_{k_n}=s_1s_2\cdots s_p$, $l(k_n)=p$ and $1\le\forall j\le l(k_n)\ (s_j\in L_{k_n,j})$. Therefore, for any $t=(s_1,...,s_p)$ such that $w_{k_n}=s_1\cdots s_p$ and $1\le\forall j\le p\ (s_j\in\Sigma^+)$, $k_n\notin A_t$ holds. Hence, we have $k_n\notin A_{t_{k_n}}=N_n$. This implies that $k_{n+1}>k_n$. Therefore the claim holds.

Next, for some infinitely many integers $k_i\ge 0$, we select a string $w_{k_i,j^*}$ and a language $L_{k_i,j^*}$ from $t_{k_i}=(w_{k_i,1},w_{k_i,2},...,w_{k_i,l_i})$ and $L_{k_i,1},...,L_{k_i,l(k_i)}$, respectively, and construct an infinite sequence of strings and languages satisfying the condition of infinite elasticity. The construction is as follows.

For $1\le\forall j\le\mid w_0\mid$, let $P_j=\{k_n\ \mid\ j\le l(k_n)\wedge w_{k_n,j}\notin L_{k_n,j}\}$.

Here, we have that $\forall k_n,\ 1\le\exists j\le l(k_n)\ (w_{k_n,j}\notin L_{k_n,j})$, since $w_{k_n}\notin L_{k_n}$. It also holds that, for any positive integer $k$, $l(k)\le\mid w_0\mid$, since $\forall k\ge 1\ (w_0\in L_k)$.

These facts imply that, for each $k_n$, there exists some integer $j$ such that $1 \leq j \leq\mid w_0 \mid$ and $k_n \in P_j$. Therefore, there exists some $j^*$ such that $\mid P_{j^*} \mid= \infty$.

By $k_{p_i}$, we denote the $i$th smallest element in $P_{j^*}$. Let us consider an infinite sequene of strings, $w_{k_{p_1},j^*}, w_{k_{p_2},j^*}, ...$ and an infinite sequence of languages, $L_{k_{p_1},j^*}, L_{k_{p_2},j^*}, ....$ By the relation $N_{p_1} \supseteq N_{p_2} \supseteq N_{p_3} \supseteq \cdots \supseteq N_{p_{i-1}}$, it is easy to see that $k_{p_i} \in N_{p_m}$ holds for $m = 1, 2, ..., i-1$. Therefore, we have $w_{k_{p_m},j^*} \in L_{k_{p_i},j^*}$ for $m = 1, 2, ..., i-1$ by the definition of $N_{p_m}$ and $A_{t_{k_{p_m}}}$. Hence, $\{w_{k_{p_1},j^*}, w_{k_{p_2},j^*}, ..., w_{k_{p_{i-1}},j^*}\} \subseteq L_{k_{p_i},j^*}$ holds. On the other hand, by the definition of $P_{j^*}$, we have $w_{k_{p_i},j^*} \notin L_{k_{p_i},j^*}$ for any $k_{p_i} \in P_{j^*}$. These facts imply that the class $\mathcal{C}$ has infinite elasticity, which is a contradiction. This completes the proof. □

**Example 4** Let us consider language classes $\mathcal{C}_0 = \{\{a\}, a^*\}$ and $\mathcal{C}_1 = \{(\lambda + a), a^*\}$ over $\Sigma = \{a\}$, both of which have finite thickness. $Con(\mathcal{C}_0)$ has finite elasticity. However, $Con(\mathcal{C}_1)$ does not have finite elasticity, which is because $\mathcal{C}_1$ does not satisfy the condition (C1). Therefore, the condition (C1) is necessary in this sense. Please note that $\mathcal{C}_1$ is identifiable in the limit from positive data but that $Con(\mathcal{C}_1)$ is not identifiable in the limit from positive data, since there exists no *finite tell-tale*(cf. [Ang80]) for $a^*$. □

It was shown in [MS93] that a *fixed* finite number of language concatenations preserves finite elasticity. Here, we have proved that under a condition (C1), $Con(\mathcal{C})$, i.e., *arbitrary* number of language concatenations preserves finite elasticity. Note, however, that without a condition (C1), $Con(\mathcal{C})$ does not preserve finite elasticity of $\mathcal{C}$, as shown in the example above.

**Theorem 6** $CLT^{\leq k}$ and $CLTS^{\leq k}$ are identifiable in the limit from positive data.

**Proof**

By Lemma 5 and Lemma 6, $CLT^{\leq k}$ has finite elasticity. By Therem 2, $CLTS^{\leq k}$ has finite elasticity, too. By Theorem 4, we have the results. □

## 4.3 Local Parsability

We showed in the previous subsection that the class $CLT^{\leq k}$ and $CLTS^{\leq k}$ are identifiable in the limit from positive data, which does not mean the existence of efficient learning algorithms for $CLT^{\leq k}$ or $CLTS^{\leq k}$. In this subsection, we consider the problem of learning a subclass of $CLTS^{\leq k}$ from positive data with the conjectures updated in polynomial time.

The dificulties of efficient learning of $CLTS^{\leq k}$ seem to lie in the intractability of finding concatenation points of given training examples. Therefore, it would be better to impose some reasonably restrictive conditions on concatinating strictly locally testable languages for obtaining some efficiently learnable subclass of $CLTS^{\leq k}$.

In this paper, we introduce a notion called *local parsability*. This notion has some close relationships to the notion of local parsability originally defined in

[MP71], which is proposed for analyzing *code events*. Intuitively, a language $L$ in $Con(\mathcal{C})$ is said to be locally parsable if we can determine concatenation points of any given string $w$ in $L$ by scanning $w$ with a fixed finite length window. More formally, the notion is defined as follows.

Let $k$ be a positive integer. A parse set of length $k$ is a finite set of pairs of strings $(p, q)$ such that $| p | \leq k, | q | \leq k$. Let $PS$ be a parse set of length $k$. Then, for any string $w$ in $\Sigma^*$, $N(w, PS)$ is defined as the set consisting of 0 and $| w |$, and integers $i$ such that $\exists (p, q) \in PS \, \exists \, x, y \in \Sigma^* \, (w = xpqy \wedge i = | x | + | p |)$. Here, we denote the $i$-th smallest element of $N(w, PS)$ by $j_i$.

Let $\mathcal{C}$ be a class of languages, $PS$ be a parse set of length $k$, $L_1, \cdots, L_n$ be a finite sequnce of languages in $\mathcal{C}$ and $w$ be a string in $L = L_1 \cdots L_n$. Then, $w$ is *parsable to* $L_1, \cdots, L_n$ *based on* $PS$, iff $| N(w, PS) | = n + 1$ and $\forall j_i, j_{i+1} \in N(w, PS) \, (sub(w, j_i + 1, j_{i+1}) \in L_i)$, where, by $sub(w, i, j)$, we denote the substring of $w$ which starts at the $i$th and ends at the $j$th character of $w$. In case of $i > j$, $sub(w, i, j)$ represents $\lambda$. A language $L \in Con(\mathcal{C})$ is said to be $k$-*parsable* iff there exists a finite sequence $L_1, \cdots, L_n$ of languages in $\mathcal{C}$ and a parse set $PS$ of length $k$ such that $L = L_1 \cdots L_n$ and, for any string $w \in L_1 \cdots L_n$, $w$ is parsable to $L_1, \cdots, L_n$ based on $PS$. A language $L \in Con(\mathcal{C})$ is said to be *locally parsable* iff $L$ is $k$-parsable for some positive integer $k$.

The class of languages $(k, l)$-$CLTS$ is defined as the smallest class of languages that contains all languages $L \in CLTS^{\leq k}$ such that $L$ is $l$-parsable. The class of languages $PCLTS$ is defined as the smallest class of languages that contains all languages $L \in CLTS$ such that $L$ is locally parsable.

**Example 5** Let $L_1$ and $L_2$ be strictly 1-testable languages such that $triple(L_1) = \langle \{a\}, \{a\}, \{a\} \rangle$ and $triple(L_2) = \langle \{b\}, \{b\}, \{b\} \rangle$. Let $L_3 = L_1 \cdot L_2 \cdot L_1$. Then we can easily prove $L_3 \in (1,1)$-$CLTS$, since $PS = \{(a, b), (b, a)\}$ is a parse set of length 1 such that for any string $w \in L_3$, $w$ is parsable to $L_1, L_2, L_1$ based on $PS$. Please note here that $L_3$ is in $(k, l)$-$CLTS$ for any positive integers $k$ and $l$.

We can prove $L_3 \notin LT$ as follows. Let us assume $L_3 \in LT$. Then there exists some positive integer $k$ such that $L_3$ is $k$-testable. We have $w_1 = a^{k+1}ba^k \in L_3$ and $w_2 = a^k ba^k ba^k \notin L_3$. It is easy to see that $L_k(w_1) = L_k(w_2)$, $R_k(w_1) = R_k(w_2)$ and $I_k(w_1) = I_k(w_2)$ hold, which is a contradiction. □

Therefore, we have the following.

**Lemma 7** For any positive integers $k$ and $l$, there exists a language $L$ in $(k, l)$-$CLTS$ such that $L \notin LT$.

Please note that any language $L$ in $LTS^{\leq k}$ is in $(k, l)$-$CLTS$, for any positive integer $l$, since any string in $L$ is $l$-parsable to $L$ based on $PS = \emptyset$.

**Theorem 7** **(1)** $PCLTS$ and $(k, l)$-$CLTS$ are incomparable to $LT$.

**(2)** $PCLTS((k, l)$-$CLTS)$ properly includes $LTS(LTS^{\leq k}$, respectively).

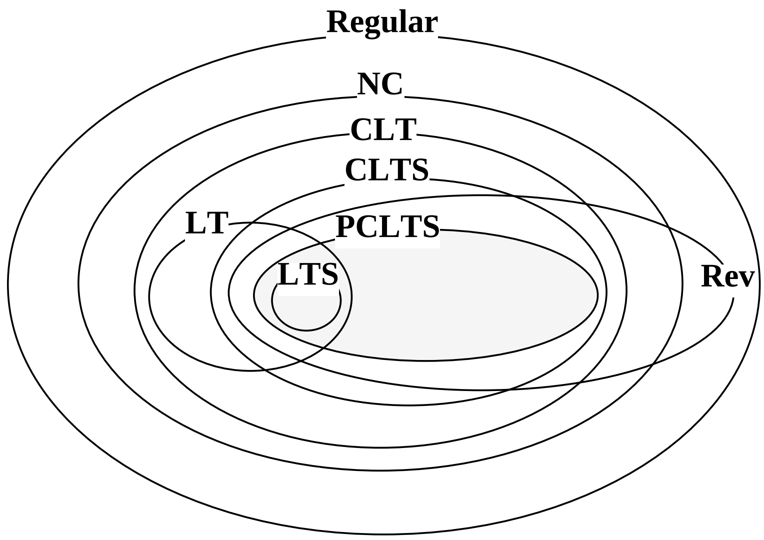


Figure 1: Relationships between subclasses of regular languages

### 4.4 Efficient Learning of $(k,l)$-$CLTS$ from Positive Data

In this subsection, we will show that $(k,l)$-$CLTS$ is identifiable in the limit from positive data using reversible automata with the conjectures updated in polynomial time. This proof is based on a relationship between $(k,l)$-$CLTS$ and $Rev(k{+}2l)$.

**Lemma 8** Let $L$ be any language in $LTS^{\leq k}$ and $u_1, u_2, v, x$, and $y$ be any strings over $\Sigma$ such that $u_1vx, u_2vx, u_1vy \in L$, and $\mid v \mid = k+1$. Then $u_2vy \in L$ holds.

**Proof**

There exists some positive integer $j$ $(\leq k)$ such that $L$ is strictly $j$-testable by the definition of $LTS^{\leq k}$. By Lemma 4, $L$ is $(j+1)$-reversible. Then we have that $L$ is $(k+1)$-reversible by Lemma 2. Therefore, we have $u_2vy \in L$ by the definition of $k$-reversible language. □

We then can prove an interesting relationship between $(k,l)$-$CLTS$ and $Rev(k{+}2l)$.

**Theorem 8** For any positive integers $k$ and $l$, $(k,l)$-$CLTS$ is contained in $Rev(k{+}2l)$.

**Proof**

Let $L$ be a language in $(k,l)$-$CLTS$. Then there exist a positive integer $n$, a sequence of languages $L_1, ..., L_n$ in $LTS^{\leq k}$, and a parse set $PS$ of length $l$ such that $L = L_1 \cdots L_n$, and, for all $w \in L$, $w$ is parsable to $L_1, ..., L_n$ based on $PS$. Let us consider any strings $u_1, u_2, v, x, y \in \Sigma^*$ such that $w_1 = u_1vx \in L, w_2 = u_2vx \in L, w_3 = u_1vy \in L$ and $\mid v \mid = k{+}2l$. It is sufficient for us to show $w_4 = u_2vy \in L$.

Here, we denote the $i$-th smallest element in $N(w_p, PS)$ by $j_i^p$, for $p = 1, 2, 3, 4$. For proving $w_4 = u_2vy \in L$, it suffices to show $|\ N(w_4, PS)\ | = n + 1$ and $\forall j_i^4, j_{i+1}^4 \in N(w_4, PS)\ sub(w_4, j_i^4 + 1, j_{i+1}^4) \in L_i$.

From the assumption above, we have,

**(1)** $|\ N(w_p, PS)\ | = n + 1$ (for $p = 1, 2, 3$)

**(2)** $\forall j_i^p, j_{i+1}^p \in N(w_p, PS)\ sub(w_p, j_i^p + 1, j_{i+1}^p) \in L_i$ (for $p = 1, 2, 3$)

Further, for a finite set of integers $N$ and an integer $m$, by $N_{\leq m}(N_{>m})$, we denote the set of all elements in $N$ which are less than or equal to $m$ (grater than $m$), and, by $N/m$, we denote the set $\{e - m \mid e \in N\}$. Then we have the followings.(cf. Figure 2)

**(3)** $N(w_3, PS)_{\leq |u_1|+l} \ = \ N(w_1, PS)_{\leq |u_1|+l}$

**(4)** $N(w_2, PS)_{>|u_2|+l} / \mid u_2 \mid \ = \ N(w_1, PS)_{>|u_1|+l} / \mid u_1 \mid$

**(5)** $N(w_2, PS)_{\leq |u_2|+l} \ = \ N(w_4, PS)_{\leq |u_2|+l}$

**(6)** $N(w_3, PS)_{>|u_1|+l} / \mid u_1 \mid \ = \ N(w_4, PS)_{>|u_2|+l} / \mid u_2 \mid$

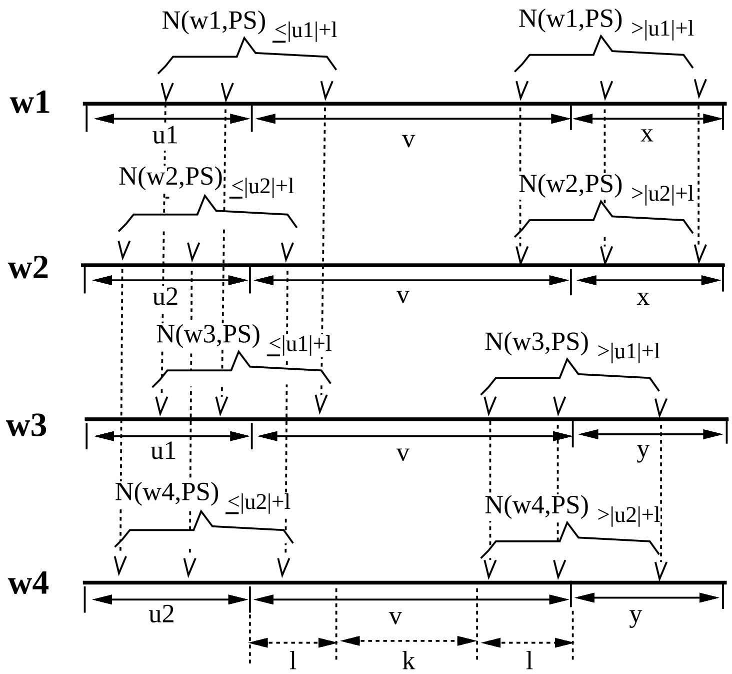


Figure 2: Concatenation points of each string ( in case $j_{i^*+1}^4 \geq |\ u_2\ | + k + l + 1$)

By summing up (3), (4), (5), (6) after taking cardinalities of both sides of the equations, and then using (1), we obtain the equation $|\ N(w_4, PS)\ | = n + 1$.

Let $i^* =| N(w_4, PS)_{\leq |u_2|+l} |$. Then, by (1), (3), (4), (5),and (6), we have the following relations.

**(7)** $i^* =| N(w_1, PS)_{\leq |u_1|+l} |=| N(w_2, PS)_{\leq |u_2|+l} |=| N(w_3, PS)_{\leq |u_1|+l} |$.

**(8)** $\forall i \leq i^* \, (j_i^1 = j_i^3 \leq| u_1 | +l \ \wedge \ j_i^2 = j_i^4 \leq| u_2 | +l)$

**(9)** $\forall i > i^* \, (j_i^1- | u_1 |= j_i^2- | u_2 |> l \ \wedge \ j_i^3- | u_1 |= j_i^4- | u_2 |> l)$.

By (2), (8), (9), we have

$$\forall j_i^4, j_{i+1}^4 \in N(w_4, PS) \ s.t. \ i+1 \leq i^* \ \ sub(w_4, j_i^4+1, j_{i+1}^4) = sub(u_2v, j_i^4+1, j_{i+1}^4) = sub(u_2v, j_i^2+1, j_{i+1}^2) = sub(w_2, j_i^2+1, j_{i+1}^2) \in L_i$$

$$\forall j_i^4, j_{i+1}^4 \in N(w_4, PS) \ s.t. \ i > i^* \ \ sub(w_4, j_i^4+1, j_{i+1}^4) = sub(vy, j_i^4+1- | u_2 |, j_{i+1}^4- | u_2 |) = sub(vy, j_i^3+1- | u_1 |, j_{i+1}^3- | u_1 |) = sub(w_3, j_i^3+1, j_{i+1}^3) \in L_i$$

Therefore, it is only left for us to show $sub(w_4, j_{i^*}^4+1, j_{i^*+1}^4) \in L_{i^*}$,

We consider the two cases of $j_{i^*+1}^4 \leq| u_2 | +k+l$ and $j_{i^*+1}^4 \geq| u_2 | +k+l+1$.

In case $j_{i^*+1}^4 \leq| u_2 | +k+l$, we have $j_{i^*+1}^2 = j_{i^*+1}^4 \leq| u_2 | +k+l$, since $N(w_2, PS)_{\leq |u_2|+k+l} = N(w_4, PS)_{\leq |u_2|+k+l}$ holds. Therefore, by (2) and (8), it holds that $sub(w_4, j_{i^*}^4+1, j_{i^*+1}^4) = sub(u_2v, j_{i^*}^4+1, j_{i^*+1}^4) = sub(u_2v, j_{i^*}^2+1, j_{i^*+1}^2) = sub(w_2, j_{i^*}^2+1, j_{i^*+1}^2) \in L_{i^*}$.

Let us consider the case $j_{i^*+1}^4 \geq| u_2 | +k+l+1$. In this case, $j_{i^*+1}^2 \geq | u_2 | +k+l+1$ holds since otherwise $j_{i^*+1}^4 = j_{i^*+1}^2 \leq| u_2 | +k+l$ holds by the the relation $N(w_2, PS)_{\leq |u_2|+k+l} = N(w_4, PS)_{\leq |u_2|+k+l}$. Therefore, we have $j_{i^*+1}^1, j_{i^*+1}^3 \geq| u_1 | +k+l+1$ by (9).

Let $q_1 = sub(u_1v, j_{i^*}^1+1, | u_1 | +l)$, $q_2 = sub(u_2v, j_{i^*}^2+1, | u_2 | +l)$, $z_1 = sub(vx, k+l+2, j_{i^*+1}^1- | u_1 |)$, $z_2 = sub(vy, k+l+2, j_{i^*+1}^3- | u_1 |)$, and $r = sub(v, l+1, k+l+1)$. Then, by (2), (8), (9), we have $sub(w_1, j_{i^*}^1+1, j_{i^*+1}^1) = q_1rz_1 \in L_{i^*}$, $sub(w_2, j_{i^*}^2+1, j_{i^*+1}^2) = q_2rz_1 \in L_{i^*}$, $sub(w_3, j_{i^*}^3+1, j_{i^*+1}^3) = q_1rz_2 \in L_{i^*}$, and $sub(w_4, j_{i^*}^4+1, j_{i^*+1}^4) = q_2rz_2$. Here note that $| r |= k+1$. Therefore, by Lemma 8 and $L_{i^*} \in LTS^{\leq k}$, we have $sub(w_4, j_{i^*}^4+1, j_{i^*+1}^4) = q_2rz_2 \in L_{i^*}$. This completes the proof. □

**Theorem 9** [Ang82] The class of $k$-reversible languages is identifiable in the limit from positive data using $k$-reversible automata with the conjectures updated in polynomial time.

The learning algorithm for $Rev(k)$ is called $k$-$RI$ in [Ang82].

**Theorem 10** $(k,l)$-$CLTS$ is identifiable in the limit from positive data using $(k+2l)$-reversible automata with the conjectures updated in polynomial time.
**Proof**
By Theorem 8 and Theorem 9, we have only to apply the learning algorithm $(k+2l)$-$RI$ for learning $(k,l)$-$CLTS$. □

Note here that the $(k+2l)$-$RI$ algorithm does not always output a conjecture $g_i$ such that $L(g_i) \in (k,l)$-$CLTS$. Therefore, Theorem 10 does not imply the existence of a *class preserving* efficent learning algorithm, but the *class comprising* efficient learnability of the class $(k,l)$-$CLTS$. It is an open question whether the class preserving efficient learnability holds for $(k,l)$-$CLTS$.

# 5 Concluding Remarks

In [Yok90], Yokomori presented a learning algorithm for the class $LTS^{=k}$ from positive data. In the current paper, we have introduced some extended classes, $CLT^{\leq k}$, $CLTS^{\leq k}$, $(k,l)$-$CLTS$, etc., by concatenating locally testable languages, and established their relationships. These classes, $CLT^{\leq k}$, $CLTS^{\leq k}$, $(k,l)$-$CLTS$ are proved to properly include $LTS^{=k}$ and to be identifiable in the limit from positive data. Especially, we have shown that the class $(k,l)$-$CLTS$ is identifiable in the limit from positive data using reversible automata with the conjectures updated in polynomial time, which is based on a close relationship between the class $(k,l)$-$CLTS$ and the class of $(k+2l)$-reversible languages.

On the other hand, in [YIK94], we applied the learning algorithm for $LTS^{=k}$ to the problem of identifying the $\alpha$-chain region of amino acid sequences of hemoglobin and obtained the overall success rate of more than 90% correct prediction for unknown $\alpha$-chain region of amino acid sequences. This work, motivated by the theoretical result by [Hea87], is interesting in that it bridges the gap between the mathematical analysis of splicing process of DNA sequences and formal language theory.

This paper presents some theoretical results on the learnability of some subclasses of concatenations of locally testable languages from positive data. It is strongly suggested by the experimental work [YIK94], that the class of concatenations of locally testable languages may effectively model some classes of amino acid or DNA sequences where sequencial locality changes exist. In fact, the notion of local parsability is strongly motivated from a biological observation that exon-intron boundaries are characterized by a finite set of pairs of base sequences. Therefore, Theorem 10 suggests that Angluin's $k$-$RI$ algorithm has some potential abilities to find sequentially changing common localities in given samples of amino acid or DNA sequences, provided that local feature changes of biological data are locally parsable. These application issues to amino acid or DNA sequence analysis are left for future works.

It is also left as a theoretical interest to find an efficient learning algorithm for $(k,l)$-$CLTS$ in which a representation class $\mathcal{R}$ is class preserving with respect to the target class $(k,l)$-$CLTS$.

# Acknowledgement

We would like to thank anonymous referees for their valuable comments. This work was supported in part by Grants-in-Aid for Scientific Research No.06780302 and No.06249202 from the Ministry of Education, Science and Culture, Japan.

## References

[Ang80] D. Angluin. Inductive inference of formal languages from positive data. *Information and Control*, vol.45, pp.117-135, 1980

[Ang82] D. Angluin. Inference of reversible languages. *Journal of the ACM*, vol.29, pp.741-765, 1982

[BB75] L. Blum and M. Blum. Toward a Mathematical Theory of Inductive Inference. *Information and Control*, vol.28, pp.125-155, 1975

[Gol67] E. Mark Gold. Language identification in the limit. *Information and Control*, vol.10, pp.447-474, 1967

[Hea87] T. Head. Formal language theory and DNA : An analysis of the generative capacity of specific recombinant behaviors. *Bulletin of Mathematical Biology*, vol.49, pp.737-759, 1987

[LZ93] S. Lange and T. Zeugmann. Language Learning in Dependence on the Space of Hypotheses. *Proc. of 6th Annual ACM Workshop on Computational Learning Theory*, pp.127-136, 1993

[MS93] T. Moriyama and M. Sato. Properties of Language Classes with Finite Elasticity. *Proc. 4th Workshop on Algorithmic Learning Theory*, Lecture Notes in Artificial Intelligence **744**, pp.187-196, 1993

[MSW90] T. Motoki, T. Shinohara and K. Wright. The correct definition of finite elasticity: corrigendum to identification of unions. *Proc. of 4th Workshop on Computational Learning Theory*, pp.375-375, 1991

[MP71] R. McNaughton and S. Papert. Counter-Free Automata. *MIT Press*, Cambridge, MA, 1971

[Yok90] T. Yokomori. A note on the polynomial-time identification of strictly local languages in the limit. Report CSIM90-03, Dept. of Comput. Sci. and Inf. Math., Univ. of Elect.-Communi., 1990

[YIK94] T. Yokomori, N. Ishida and S. Kobayashi. Learning Local Languages and Its Application to Protein $\alpha$-chain Identification. *Proc. of 27th Hawaii International Conference on System Sciences*, Hawaii, pp.113-122, 1994, Janualy

[Wri89] K. Wright. Identification of unions of languages drawn from an identifiable class. *Proc. of 2nd Workshop on Computational Learning Theory*, pp.328-333, 1989