Two-Sided Strictly Locally Testable Languages

Markus Holzer    Martin Kutrib    Friedrich Otto

Institut für Informatik, Universität Giessen, Germany
Fachbereich Elektrotechnik/Informatik, Universität Kassel, Germany
Overview

- Two-Sided Strictly Locally Testable Languages
- Expressive Capacity
- Learnability in the Limit
- Decidability Problems
- Closure Properties
Overview

- Two-Sided Strictly Locally Testable Languages
- Expressive Capacity
- Learnability in the Limit
- Decidability Problems
- Closure Properties
Overview

- Two-Sided Strictly Locally Testable Languages
- Expressive Capacity
- Learnability in the Limit
- Decidability Problems
- Closure Properties
Overview

- Two-Sided Strictly Locally Testable Languages
- Expressive Capacity
- Learnability in the Limit
- Decidability Problems
- Closure Properties
Overview

- Two-Sided Strictly Locally Testable Languages
- Expressive Capacity
- Learnability in the Limit
- Decidability Problems
- Closure Properties
Two-Sided Strictly Locally Testable Languages

Standard Languages

**Standard language:** $L \subseteq \Sigma^*$ can be described by three sets $A, B \subseteq \Sigma$ and $C \subseteq \Sigma^2$ such that

$$L = (A\Sigma^* \cap \Sigma^* B) \setminus \Sigma^* F \Sigma^*$$

where $F = \Sigma^2 \setminus C$ is the set of forbidden factors.
Two-Sided Strictly Locally Testable Languages

Standard Languages

**Standard language:** $L \subseteq \Sigma^*$ can be described by three sets $A, B \subseteq \Sigma$ and $C \subseteq \Sigma^2$ such that

$$L = (A\Sigma^* \cap \Sigma^* B) \setminus \Sigma^* F\Sigma^*$$

where $F = \Sigma^2 \setminus C$ is the set of forbidden factors.

<table>
<thead>
<tr>
<th>Theorem</th>
<th>[Chomsky and Schützenberger 1963]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Every regular language is the letter-to-letter homomorphich image of a standard language.</td>
<td></td>
</tr>
</tbody>
</table>
Strictly $k$-testable language: $L \subseteq \Sigma^*$ can be described by three sets $A, B, C \subseteq \Sigma^k$ such that, if $|w| \geq k$ then $w \in L$ if and only if

its prefix of length $k$ is in $A$, its suffix of length $k$ is in $B$, and all inner factors of length $k$ belong to $C$. 

Example

The language $L = \{a^m b^n | m, n \geq 1\}$ is strictly 2-testable:
Set $A = \{aa, ab\}$, $B = \{ab, bb\}$, and $C = \{aa, ab, bb\}$. 

$a$$a$$a$$a$$b$$b$$b$$\in A \✓$
Two-Sided Strictly Locally Testable Languages

Strictly $k$-testable language: $L \subseteq \Sigma^*$ can be described by three sets $A, B, C \subseteq \Sigma^k$ such that, if $|w| \geq k$ then $w \in L$ if and only if

its prefix of length $k$ is in $A$, its suffix of length $k$ is in $B$, and all inner factors of length $k$ belong to $C$.

Example

The language $L = \{a^m b^n \mid m, n \geq 1\}$ is strictly 2-testable:

Set $A = \{aa, ab\}$, $B = \{ab, bb\}$, and $C = \{aa, ab, bb\}$.

Example of word $aaabbb$:

\[
\begin{array}{ccccccc}
  a & a & a & a & b & b & b \\
\end{array}
\]

$\in A \surd$
Two-Sided Strictly Locally Testable Languages

Strictly $k$-testable language: $L \subseteq \Sigma^*$ can be described by three sets $A, B, C \subseteq \Sigma^k$ such that, if $|w| \geq k$ then $w \in L$ if and only if

its prefix of length $k$ is in $A$, its suffix of length $k$ is in $B$, and all inner factors of length $k$ belong to $C$.

Example
The language $L = \{ a^m b^n | m, n \geq 1 \}$ is strictly 2-testable:

Set $A = \{ aa, ab \}$, $B = \{ ab, bb \}$, and $C = \{ aa, ab, bb \}$.

∈ $C$ ✓
Two-Sided Strictly Locally Testable Languages

Strictly $k$-testable language: $L \subseteq \Sigma^*$ can be described by three sets $A, B, C \subseteq \Sigma^k$ such that, if $|w| \geq k$ then $w \in L$ if and only if

- its prefix of length $k$ is in $A$,
- its suffix of length $k$ is in $B$,
- and all inner factors of length $k$ belong to $C$.

Example

The language $L = \{ a^m b^n \mid m, n \geq 1 \}$ is strictly 2-testable:

Set $A = \{ aa, ab \}$, $B = \{ ab, bb \}$, and $C = \{ aa, ab, bb \}$.

\[
\begin{array}{ccccccc}
 a & a & a & a & b & b & b \\
\end{array}
\]

$\in C \ \checkmark$
Strictly \( k \)-testable language: \( L \subseteq \Sigma^* \) can be described by three sets \( A, B, C \subseteq \Sigma^k \) such that, if \( |w| \geq k \) then \( w \in L \) if and only if

its prefix of length \( k \) is in \( A \),
its suffix of length \( k \) is in \( B \),
and all inner factors of length \( k \) belong to \( C \).

Example
The language \( L = \{ a^m b^n \mid m, n \geq 1 \} \) is strictly 2-testable:

Set \( A = \{ aa, ab \} \), \( B = \{ ab, bb \} \), and \( C = \{ aa, ab, bb \} \).

\[
\begin{array}{cccccc}
  a & a & a & a & b & b & b \\
\end{array}
\]

\( \in C \) \( \checkmark \)
Two-Sided Strictly Locally Testable Languages

Strictly $k$-testable language: $L \subseteq \Sigma^*$ can be described by three sets $A, B, C \subseteq \Sigma^k$ such that, if $|w| \geq k$ then $w \in L$ if and only if

its prefix of length $k$ is in $A$, its suffix of length $k$ is in $B$, and all inner factors of length $k$ belong to $C$.

Example

The language $L = \{a^m b^n \mid m, n \geq 1\}$ is strictly 2-testable:

Set $A = \{aa, ab\}$, $B = \{ab, bb\}$, and $C = \{aa, ab, bb\}$.

$$a \ a \ a \ a \ b \ b \ b \ \in \ C \ \checkmark$$
Strictly \( k \)-testable language: \( L \subseteq \Sigma^* \) can be described by three sets \( A, B, C \subseteq \Sigma^k \) such that, if \( |w| \geq k \) then \( w \in L \) if and only if

- its prefix of length \( k \) is in \( A \),
- its suffix of length \( k \) is in \( B \),
- and all inner factors of length \( k \) belong to \( C \).

Example
The language \( L = \{ a^m b^n \mid m, n \geq 1 \} \) is strictly 2-testable:

Set \( A = \{ aa, ab \}, B = \{ ab, bb \}, \) and \( C = \{ aa, ab, bb \} \).
A. Rosenberg 1967: Double-head finite automata characterize the linear context-free languages.

B. Nagy 2015: Double-head pushdown automata accept linguistically important languages and describe a family of mildly context-sensitive languages.
Two-Sided Strictly Locally Testable Languages

**Idea:** Two windows of some size $k$ move across the input, one from left to right and the other from right to left.
Two-Sided Strictly Locally Testable Languages

**Idea:** Two windows of some size $k$ move across the input, one from left to right and the other from right to left.

**But:** The families of strictly $k$-testable languages are closed under intersection.
Two-Sided Strictly Locally Testable Languages

**Idea:** Two windows of some size $k$ move across the input, one from left to right and the other from right to left.

**But:** The families of strictly $k$-testable languages are closed under intersection.

**Therefore:** The windows move simultaneously and the factors scanned have to be in relation.
Two-Sided Strictly Locally Testable Languages

Idea: Two windows of some size $k$ move across the input, one from left to right and the other from right to left.

But: The families of strictly $k$-testable languages are closed under intersection.

Therefore: The windows move simultaneously and the factors scanned have to be in relation.

Two-sided strictly $k$-testable language: $L \subseteq \Sigma^*$ is given through a strictly $k$-testable language $H$ and a binary relation $R \subseteq \Sigma^k \times \Sigma^k$. If $|w| \geq k$ then $w \in L$ if and only if $w \in H$ and, for all indices $i \in \{1, 2, \ldots, |w| - k + 1\}$, $(w[i, k], w[|w| + 2 - k - i, k]) \in R$. 
Two-Sided Strictly Locally Testable Languages

Example
There is a two-sided strictly 1-testable language $L$ such that
$L \cap a^* b^* = \{ a^n b^n \mid n \geq 1 \}$:

Set $A = B = C = \{a, b\}$ and $R = \{(a, b), (b, a)\}$.

```
  a  a  a  b  b  b
```
Example
There is a two-sided strictly 1-testable language \( L \) such that
\[ L \cap a^* b^* = \{ a^n b^n \mid n \geq 1 \} : \]
Set \( A = B = C = \{ a, b \} \) and \( R = \{ (a, b), (b, a) \} \).
Example

There is a two-sided strictly 1-testable language $L$ such that $L \cap a^*b^* = \{a^n b^n \mid n \geq 1\}$:

Set $A = B = C = \{a, b\}$ and $R = \{(a, b), (b, a)\}$.
Two-Sided Strictly Locally Testable Languages

Example
There is a two-sided strictly 1-testable language $L$ such that $L \cap a^*b^* = \{ a^n b^n \mid n \geq 1 \}$:

Set $A = B = C = \{a, b\}$ and $R = \{(a, b), (b, a)\}$.

\[
\begin{array}{cccccc}
  a & a & a & b & b & b \\
\end{array}
\]

$\in R \checkmark$

$\in C \checkmark \in C \checkmark$
Two-Sided Strictly Locally Testable Languages

Example

There is a two-sided strictly 1-testable language $L$ such that $L \cap a^*b^* = \{ a^n b^n \mid n \geq 1 \}$:

Set $A = B = C = \{a, b\}$ and $R = \{(a, b), (b, a)\}$.

$\in R \checkmark$

$\in C \checkmark \in C \checkmark$
Two-Sided Strictly Locally Testable Languages

Example
There is a two-sided strictly 1-testable language $L$ such that
$L \cap a^*b^* = \{ a^n b^n \mid n \geq 1 \}$:

Set $A = B = C = \{a, b\}$ and $R = \{(a, b), (b, a)\}$.

\[
\begin{array}{c}
\in R \checkmark \\
\in C \checkmark & \in C \checkmark \\
\end{array}
\]
Example

There is a two-sided strictly 1-testable language $L$ such that $L \cap a^*b^* = \{ a^n b^n \mid n \geq 1 \}$:

Set $A = B = C = \{a, b\}$ and $R = \{(a, b), (b, a)\}$.
The classes $2\text{SLT}(k)$, for $k \geq 1$, are incomparable to the classes $\text{REG}$, $\text{LIN}$, $\text{DLIN}$, $\text{DCFL}$, and $\text{CRL}$.

<table>
<thead>
<tr>
<th>REG</th>
<th>LIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLT</td>
<td>2SLT</td>
</tr>
<tr>
<td>$\text{SLT}(k + 1)$</td>
<td>$\text{2SLT}(k + 1)$</td>
</tr>
<tr>
<td>SLT($k$)</td>
<td>2SLT($k$)</td>
</tr>
</tbody>
</table>
Expressive Capacity

- $\text{SLT} \subset \text{REG}$
- $\text{SLT}(k) \subset \text{SLT}(k+1)$
- $\text{SLT}(k) \subset 2\text{SLT}(k)$
Expressive Capacity

→ SLT ⊂ REG
→ SLT\(^{(k)}\) ⊂ SLT\(^{(k + 1)}\)
→ SLT\(^{(k)}\) ⊂ 2SLT\(^{(k)}\)

**Theorem**

The classes 2SLT and 2SLT\(^{(k)}\), for \(k \geq 1\), are incomparable to the classes REG, DLIN, DCFL, and CRL.
**Lemma**

For all $k \geq 1$, 
\[
\{ab^{k+1}\} \in \text{SLT}(k + 1) \setminus 2\text{SLT}(k).
\]

**Theorem**

The classes $\text{2SLT}$ and $\text{2SLT}(k)$, for $k \geq 1$, are incomparable to the classes $\text{REG}$, $\text{DLIN}$, $\text{DCFL}$, and $\text{CRL}$.
Expressive Capacity

Theorem

2SLT ⊂ LIN.

Theorem

The classes 2SLT and 2SLT(k), for k ≥ 1, are incomparable to the classes REG, DLIN, DCFL, and CRL.
Expressive Capacity

→ A unary language belongs to SLT($k$) if and only if it belongs to 2SLT($k$).

**Question:** Do SLT($k$) and 2SLT($k$) coincide for regular languages?
Expressive Capacity

→ A unary language belongs to SLT($k$) if and only if it belongs to 2SLT($k$).

**Question:** Do SLT($k$) and 2SLT($k$) coincide for regular languages?

**Answer:** No

**Theorem**

There is a language belonging to (2SLT(1) ∩ REG) \ SLT.

**Witness language:** $L_r = \{ ac^n b, bc^n a \mid n \geq 0 \}$. 
Learnability in the Limit

A positive presentation of a language $L$ is an infinite sequence \( \{w_j\}_{j=1}^{\infty} \) of words from $L$ such that every $w \in L$ occurs at least once in the sequence.
A positive presentation of a language $L$ is an infinite sequence $\{w_j\}_{j=1}^{\infty}$ of words from $L$ such that every $w \in L$ occurs at least once in the sequence.

Identifying in the limit from positive data: There must exist an algorithm $A$ that returns a conjecture automaton (grammar) $M_j$ for any input $\{w_1, w_2, \ldots, w_j\}$ such that, for any positive presentation of $L$, there is a $j_0 \geq 1$ with $L(M_j) = L(M_{j+1}) = L$, for all $j \geq j_0$. 
Learnability in the Limit

A **positive presentation** of a language $L$ is an infinite sequence $\{w_j\}_{j=1}^\infty$ of words from $L$ such that every $w \in L$ occurs at least once in the sequence.

**Identifying in the limit from positive data:** There must exist an algorithm $A$ that returns a conjecture automaton (grammar) $M_j$ for any input $\{w_1, w_2, \ldots, w_j\}$ such that, for any positive presentation of $L$, there is a $j_0 \geq 1$ with $L(M_j) = L(M_{j+1}) = L$, for all $j \geq j_0$.

**Learnable in the limit from positive data:** A class of languages $\mathcal{L}$ is learnable in the limit from positive data if there exists an algorithm $A$ that, for each $L \in \mathcal{L}$, identifies $L$ in the limit from positive data.
Learnability in the Limit

Theorem [Gold 1967]

Any class of languages including all finite languages and at least one infinite language is not learnable in the limit from positive data only.
Learnability in the Limit

**Theorem [Gold 1967]**

Any class of languages including all finite languages and at least one infinite language is **not** learnable in the limit from positive data only.

**Theorem [Yokomori, Kobayashi 1998]**

For any $k \geq 1$, the family of strictly $k$-testable languages is learnable in the limit from positive data.
Learnability in the Limit

Learning Algorithm:

Input: an integer $k \geq 1$ and a positive presentation $\{w_j\}_{j=1}^{\infty}$ of a two-sided strictly $k$-testable language $L$.

Output: a sequence of linear grammars $G_j$ generating two-sided strictly $k$-testable languages such that

(i) for each $j \geq 0$, $L(G_j) \subseteq L(G_{j+1}) \subseteq L$, and

(ii) there exists $j_0 \geq 1$ such that, for each $j \geq j_0$, $G_j = G_{j+1}$, and $L(G_j) = L$. 

Learnability in the Limit

**Theorem**

For any $k \geq 1$, the family of two-sided strictly $k$-testable languages is learnable in the limit from positive data.
Theorem

For any $k \geq 1$, the family of two-sided strictly $k$-testable languages is learnable in the limit from positive data.

Remark

The families SLT and 2SLT are not learnable in the limit from positive data.
Decidability Problems

Basic properties

- Finiteness, infiniteness, and emptiness are decidable in polynomial time for two-sided strictly locally testable languages.
Decidability Problems

Universality

- Universality is undecidable for linear context-free languages.

**Theorem**

Universality is decidable in linear time for two-sided strictly locally testable languages.
Decidability Problems

Universality – Regular Inclusion Problem

**INSTANCE:** A regular language $S$ and a two-sided strictly $k$-testable language $L$.

**QUESTION:** Is $S$ contained in $L$?
Decidability Problems

Universality – Regular Inclusion Problem

**INSTANCE:** A regular language $S$ and a two-sided strictly $k$-testable language $L$.

**QUESTION:** Is $S$ contained in $L$?

**Theorem**

The regular inclusion problem is decidable for two-sided strictly locally testable languages.
Decidability Problems

Regular Equality Problem

**INSTANCE:** A regular language \( S \) and a two-sided strictly \( k \)-testable language \( L \).

**QUESTION:** Is \( S = L \)?

→ Essentially, this question reduces to the questions of whether the inclusion \( L \subseteq S \) is true.

→ \( L \subseteq S \) if and only if \( L \cap \overline{S} = \emptyset \).
Decidability Problems

Regular Equality Problem

INSTANCE: A regular language $S$ and a two-sided strictly $k$-testable language $L$.

QUESTION: Is $S = L$?

Essentially, this question reduces to the questions of whether the inclusion $L \subseteq S$ is true.

$L \subseteq S$ if and only if $L \cap \overline{S} = \emptyset$.

Theorem

The regular equality problem is decidable for two-sided strictly locally testable languages.
Decidability Problems

General Inclusion and Equivalence

**Theorem**

For any $k \geq 1$, the inclusion and equivalence problems for two-sided strictly $k$-testable languages are decidable in polynomial time.
Decidability Problems

**Theorem**

It is decidable whether a given two-sided strictly $k$-testable language is strictly $k$-testable.

**Open Problem**

The decidability status of the regularity problem for two-sided strictly $k$-testable languages is open.
## Closure Properties

<table>
<thead>
<tr>
<th>Language class</th>
<th>(c)</th>
<th>(\cup)</th>
<th>(\cap)</th>
<th>(R)</th>
<th>(\cdot)</th>
<th>(*)</th>
<th>(h_{\text{len.pres.}})</th>
<th>(h_{\text{len.pres.}}^{-1})</th>
<th>(h_\lambda^{-1})</th>
</tr>
</thead>
<tbody>
<tr>
<td>2SLT</td>
<td>(\times)</td>
<td>(\times)</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
<td>(\times)</td>
<td>(\times)</td>
<td>(\times)</td>
<td>(\checkmark)</td>
<td>(\times)</td>
</tr>
<tr>
<td>2SLT((k))</td>
<td>(\times)</td>
<td>(\times)</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
<td>(\times)</td>
<td>(\times)</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
<td>(\times)</td>
</tr>
</tbody>
</table>