Piecewise Testable Tree Languages

Mikołaj Bojańczyk, Luc Segoufin, Howard Straubing
This talk is about understanding the expressive power of logics on words and trees. The logics involved can only define (some) regular languages.
This talk is about understanding the expressive power of logics on words and trees. The logics involved can only define (some) regular languages.

Understand logic $X =$
give an algorithm to decide if a language $L$ is definable in $X$
This talk is about understanding the expressive power of logics on words and trees. The logics involved can only define (some) regular languages.

Understand logic $X = \ldots$
give na algorithm to decide if a language $L$ is definable in $X$

all regular languages

languages definable in logic $X$

**Theorem.** (I. Simon, 1975) A word language is piecewise testable iff its syntactic monoid is $J$-trivial.
a c b a c
a b is a piece of a c b a c
Definition.
A word language is called *piecewise testable* if it is a boolean combination of languages “words that contain \(w\) as a piece”
a b is a piece of a c b a c

Definition.
A word language is called piecewise testable if it is a boolean combination of languages “words that contain \( w \) as a piece”

\[
\{ abc \} = \text{contains piece } abc, \text{ but no piece of length } 4
\]

\[
a^*b^* = \text{no piece } ba
\]

\[
a^*b^*a^* = \text{no piece } bab
\]
a b is a piece of a c b a c

**Definition.**
A word language is called *piecewise testable* if it is a boolean combination of languages “words that contain $w$ as a piece”

$$\{abc\} = \text{contains piece } abc, \text{ but no piece of length 4}$$

$$a^*b^* = \text{no piece } ba$$

$$a^*b^*a^* = \text{no piece } bab$$

**Fact.** A language is piecewise testable iff it can be defined by a boolean combination of $\Sigma_1(\leq)$ formulas.

$$\exists x \exists y \ a(x) \land b(y) \land x \leq y$$
Theorem. (I. Simon, 1975)
A word language is piecewise testable
iff
its syntactic monoid is \( J \)-trivial.
Syntactic monoid of $L \subseteq \Sigma^*$
Syntactic monoid of $L \subseteq \Sigma^*$

Consider the two-sided Myhill-Nerode congruence

$$w \sim_L w'$$

holds if for every $u, v \in \Sigma^*$

$$uwv \in L \quad \text{iff} \quad uw'v \in L$$
Consider the two-sided Myhill-Nerode congruence

\[ w \sim_L w' \]

holds if for every \( u,v \in \Sigma^* \)

\[ uwv \in L \quad \text{iff} \quad uw'v \in L \]

Elements of the syntactic monoid are equivalence classes of this congruence, the monoid operation is concatenation.
Consider the two-sided Myhill-Nerode congruence

\[ w \sim_L w' \]

holds if for every \( u, v \in \Sigma^* \)

\[ uwv \in L \quad \text{iff} \quad uw'v \in L \]

Elements of the syntactic monoid are equivalence classes of this congruence, the monoid operation is concatenation.

<table>
<thead>
<tr>
<th>Language</th>
<th>Its syntactic monoid</th>
</tr>
</thead>
<tbody>
<tr>
<td>((aa)^*)</td>
<td>((aa)^<em>)  (a(aa)^</em>)</td>
</tr>
<tr>
<td>(a^<em>ba^</em>)</td>
<td>(a^*)      (a^<em>ba^</em>)  (a^*ba^<em>b(a+b)^</em>)</td>
</tr>
</tbody>
</table>
Theorem. (I. Simon, 1975)
A word language is piecewise testable iff
its syntactic monoid is $J$-trivial.
Infix relation in a monoid

For $s, t, u \in M$, we say $s$ is an infix of $tsu$.

We say $s, t \in M$ are in the same $J$-class if they are mutual infixes.

**Example.** The syntactic monoid of $(aa)^*$ has two elements, $(aa)^*$ and $a(aa)^*$, which are in the same $J$-class.

A monoid is $J$-trivial if each $J$-class has one element.
Theorem. (I. Simon, 1975)
A word language is piecewise testable
iff
its syntactic monoid is $J$-trivial.
<table>
<thead>
<tr>
<th>Language</th>
<th>Its syntactic monoid</th>
</tr>
</thead>
<tbody>
<tr>
<td>((aa)^*)</td>
<td>((aa)^<em>) \ a((aa)^</em>)</td>
</tr>
<tr>
<td>(a^<em>ba^</em>)</td>
<td>(a^*) \ a^<em>ba^</em> \ a^*ba^<em>b(a+b)^</em></td>
</tr>
<tr>
<td>(a(a+b)^*)</td>
<td>(\varepsilon) \ a(a+b)^* \ b(a+b)^*</td>
</tr>
</tbody>
</table>

**Theorem.** (I. Simon, 1975)
A word language is piecewise testable iff its syntactic monoid is \(J\)-trivial.
A word language is piecewise testable iff its syntactic monoid is $J$-trivial.

Theorem. (I. Simon, 1975)

<table>
<thead>
<tr>
<th>Language</th>
<th>Its syntactic monoid</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(aa)^*$</td>
<td>$(aa)^<em>$, $a(aa)^</em>$</td>
</tr>
<tr>
<td>$a^<em>ba^</em>$</td>
<td>$a^*$, $a^<em>ba^</em>$, $a^*ba^<em>b(a+b)^</em>$</td>
</tr>
<tr>
<td>$a(a+b)^*$</td>
<td>$\varepsilon$, $a(a+b)^<em>$, $b(a+b)^</em>$</td>
</tr>
</tbody>
</table>
A word language is piecewise testable if its syntactic monoid is $J$-trivial.

**Theorem.** (I. Simon, 1975)  
A word language is piecewise testable if
its syntactic monoid is $J$-trivial.

<table>
<thead>
<tr>
<th>Language</th>
<th>Its syntactic monoid</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(aa)^*$</td>
<td>$(aa)^* \ a(aa)^*$</td>
</tr>
<tr>
<td>$a^<em>ba^</em>$</td>
<td>$a^* \ a^<em>ba^</em> \ a^*ba^<em>b(a+b)^</em>$</td>
</tr>
<tr>
<td>$a(a+b)^*$</td>
<td>$\varepsilon \ a(a+b)^* \ b(a+b)^*$</td>
</tr>
<tr>
<td>Language</td>
<td>Its syntactic monoid</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>$(aa)^*$</td>
<td>$(aa)^<em>$ $a(aa)^</em>$</td>
</tr>
<tr>
<td>$a^<em>ba^</em>$</td>
<td>$a^*$ $a^<em>ba^</em>$ $a^*ba^<em>b(a+b)^</em>$</td>
</tr>
<tr>
<td>$a(a+b)^*$</td>
<td>$\varepsilon$ $a(a+b)^<em>$ $b(a+b)^</em>$</td>
</tr>
</tbody>
</table>

**Theorem.** (I. Simon, 1975)
A word language is piecewise testable iff
its syntactic monoid is $J$-trivial.
A word language is piecewise testable iff its syntactic monoid is $J$-trivial.

**Theorem.** (I. Simon, 1975) A word language is piecewise testable iff its syntactic monoid is $J$-trivial.

<table>
<thead>
<tr>
<th>Language</th>
<th>Its syntactic monoid</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(aa)^*$</td>
<td>$(aa)^<em>$ $a(aa)^</em>$</td>
</tr>
<tr>
<td>$a^<em>ba^</em>$</td>
<td>$a^*$ $a^<em>ba^</em>$ $a^*ba^<em>b(a+b)^</em>$</td>
</tr>
<tr>
<td>$a(a+b)^*$</td>
<td>$\varepsilon$ $a(a+b)^<em>$ $b(a+b)^</em>$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Language</th>
<th>Its syntactic monoid</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(aa)^*$</td>
<td>$(aa)^<em>$ $a(aa)^</em>$</td>
</tr>
<tr>
<td>$a^<em>ba^</em>$</td>
<td>$a^*$ $a^<em>ba^</em>$ $a^*ba^<em>b(a+b)^</em>$</td>
</tr>
<tr>
<td>$a(a+b)^*$</td>
<td>$\varepsilon$ $a(a+b)^<em>$ $b(a+b)^</em>$</td>
</tr>
</tbody>
</table>
Theorem. (I. Simon, 1975)
A word language is piecewise testable iff
its syntactic monoid is $J$-trivial.
<table>
<thead>
<tr>
<th>Language</th>
<th>Its syntactic monoid</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>((aa)^*)</td>
<td>((aa)^<em>) (a(aa)^</em>)</td>
<td>✗</td>
</tr>
<tr>
<td>(a<em>ba^</em>)</td>
<td>(a^<em>) (a</em>ba^<em>) (a</em>ba^<em>b(a+b)^</em>)</td>
<td>✓</td>
</tr>
<tr>
<td>(a(a+b)^*)</td>
<td>(\varepsilon) (a(a+b)^<em>) (b(a+b)^</em>)</td>
<td>✗</td>
</tr>
</tbody>
</table>

**Theorem.** (I. Simon, 1975)

A word language is piecewise testable iff

its syntactic monoid is \(J\)-trivial.
**Theorem.** (I. Simon, 1975)

A word language is piecewise testable iff its syntactic monoid is $J$-trivial.
<table>
<thead>
<tr>
<th>Language</th>
<th>Its syntactic monoid</th>
</tr>
</thead>
<tbody>
<tr>
<td>((aa)^*)</td>
<td>((aa)^* \ a(aa)^*)</td>
</tr>
<tr>
<td>(a^<em>ba^</em>)</td>
<td>(a^* \ a^<em>ba^</em> \ a^*ba^<em>b(a+b)^</em>)</td>
</tr>
<tr>
<td>(a(a+b)^*)</td>
<td>(\varepsilon \ a(a+b)^* \ b(a+b)^*)</td>
</tr>
</tbody>
</table>

**Theorem.** (I. Simon, 1975)

A word language is piecewise testable **iff** its syntactic monoid is $J$-trivial.

If $s$ and $t$ are in the same $J$-class, then for any $n$ one can find representatives of $s$ and $t$ with the same pieces of size $n$. 

\[
\begin{array}{cccccc}
    w & uww & u^\prime uwwv & uu^\prime uwwvvv & u^\prime uu^\prime uwwvvv & v vv \\
    s & t & s & t & s & ... \\
\end{array}
\]
### Theorem. (I. Simon, 1975)
A word language is piecewise testable iff its syntactic monoid is $J$-trivial.

<table>
<thead>
<tr>
<th>Language</th>
<th>Its syntactic monoid</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(aa)^*$</td>
<td>$(aa)^<em>$ $a(aa)^</em>$</td>
</tr>
<tr>
<td>$a^<em>ba^</em>$</td>
<td>$a^*$ $a^<em>ba^</em>$ $a^*ba^<em>b(a+b)^</em>$</td>
</tr>
<tr>
<td>$a(a+b)^*$</td>
<td>$\epsilon$ $a(a+b)^<em>$ $b(a+b)^</em>$</td>
</tr>
<tr>
<td>Language</td>
<td>Its syntactic monoid</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>(aa)*</td>
<td>(aa)*  a(aa)*</td>
</tr>
<tr>
<td>a<em>ba</em></td>
<td>a*  a<em>ba</em>  a<em>ba</em>b(a+b)*</td>
</tr>
<tr>
<td>a(a+b)*</td>
<td>ε  a(a+b)*  b(a+b)*</td>
</tr>
</tbody>
</table>

Theorem. (I. Simon, 1975)
A word language is piecewise testable iff
its syntactic monoid is $J$-trivial.
### Theorem. (I. Simon, 1975)

A word language is piecewise testable iff its syntactic monoid is $J$-trivial.

<table>
<thead>
<tr>
<th>Language</th>
<th>Its syntactic monoid</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$(aa)^*$</td>
<td>$(aa)^<em>$, $a(aa)^</em>$</td>
<td>X</td>
</tr>
<tr>
<td>$a^<em>ba^</em>$</td>
<td>$a^*$, $a^<em>ba^</em>$, $a^*ba^<em>b(a+b)^</em>$</td>
<td>✓</td>
</tr>
<tr>
<td>$a(a+b)^*$</td>
<td>$\varepsilon$, $a(a+b)^<em>$, $b(a+b)^</em>$</td>
<td>X</td>
</tr>
</tbody>
</table>

Several arguments, all difficult.
What’s the point of all this?

There is a rich theory connecting logic, regular languages, and algebra.
What’s the point of all this?

There is a rich theory connecting logic, regular languages, and algebra.

**Theorem.** (Schützenberger, McNaughton/Papert)
The following are equivalent for a word language:
– $L$ is definable in first-order logic
– $L$ is star-free
– the syntactic monoid of $L$ is group-free
What’s the point of all this?

There is a rich theory connecting logic, regular languages, and algebra.

**Theorem.** (Schützenberger, McNaughton/Papert)
The following are equivalent for a word language:
– $L$ is definable in first-order logic
– $L$ is star-free
– the syntactic monoid of $L$ is group-free

**Theorem.** (Schützenberger, Thérien / Wilke)
The following are equivalent for a word language:
– $L$ is definable in two-variable first-order logic
– $L$ can be defined by a type of unambiguous expression
– the syntactic monoid of $L$ is in DA
What’s the point of all this?

There is a rich theory connecting logic, regular languages, and algebra.

**Theorem.** (Schützenberger, McNaughton/Papert)
The following are equivalent for a word language:
– $L$ is definable in first-order logic
– $L$ is star-free
– the syntactic monoid of $L$ is group-free

**Theorem.** (Schützenberger, Thérien / Wilke)
The following are equivalent for a word language:
– $L$ is definable in two-variable first-order logic
– $L$ can be defined by a type of unambiguous expression
– the syntactic monoid of $L$ is in DA

... more results, including modulo quantifiers, the quantifier alternation hierarchy, etc.
What’s the point of all this?

There is a rich theory connecting logic, regular languages, and algebra.

**Theorem.** (Schützenberger, McNaughton/Papert)
The following are equivalent for a word language:
– $L$ is definable in first-order logic
– $L$ is star-free
– the syntactic monoid of $L$ is group-free

What about trees?

**Theorem.** (Schützenberger, Thérien / Wilke)
The following are equivalent for a word language:
– $L$ is definable in two-variable first-order logic
– $L$ can be defined by a type of unambiguous expression
– the syntactic monoid of $L$ is in DA

... more results, including modulo quantifiers, the quantifier alternation hierarchy, etc.
What’s the point of all this?

There is a rich theory connecting logic, regular languages, and algebra.

**Theorem.** (Schützenberger, McNaughton/Papert)
The following are equivalent for a word language:
– $L$ is definable in first-order logic
– $L$ is star-free
– the syntactic monoid of $L$ is group-free

**Theorem.** (Schützenberger, Trévisan/Wilke)
The following are equivalent for a word language:
– $L$ is definable in two-variable first-order logic
– $L$ can be defined by a type of unambiguous expression
– the syntactic monoid of $L$ is in $DA$

What about trees?

This paper is part of a program to extend the algebra-logic connection to trees...

... more results, including modulo quantifiers, the quantifier alternation hierarchy, etc.
A tree is finite, unranked and labeled
A tree is finite, unranked and labeled

A forest is a sequence of trees
A *tree* is finite, unranked and labeled.

A *forest* is a sequence of trees.

A *context* is a forest with a hole in a leaf.
A tree is finite, unranked and labeled

A forest is a sequence of trees

A context is a forest with a hole in a leaf
Notion of piece for forests and contexts.

is a piece of

is a piece of
Notion of piece for forests and contexts.

Definition.
A forest language is called *piecewise testable* if it is a boolean combination of languages “forests that contain $t$ as a piece”
Notion of piece for forests and contexts.

Definition.
A forest language is called piecewise testable if it is a boolean combination of languages “forests that contain $t$ as a piece”

Fact. A forest language is piecewise testable iff it can be defined by a boolean combination of $\Sigma_1(\leq, \leq_{lex})$ formulas.
contains piece

contains no piece with 5 nodes
all leaves are

contains no piece

contains no piece with 5 nodes

contains piece
contains piece
contains no piece with 5 nodes
contains no piece
contains no piece
all leaves are
forest is a word (vertically)
contains piece

contains no piece with 5 nodes

all leaves are

contains no piece

forest is a word (vertically)

contains no piece

forest is a word (horizontally)

contains no piece
We want the forest extension of:

**Theorem.** (I. Simon, 1975)
A word language is piecewise testable iff
its syntactic monoid is $J$-trivial.
We want the forest extension of:

**Theorem.** (I. Simon, 1975)
A word language is piecewise testable
  iff
its syntactic monoid is $J$-trivial.

What is a syntactic monoid for forest languages?

Although a definition exists (forest algebra), here we will only talk about Myhill-Nerode equivalence.
Myhill-Nerode congruence for a forest language $L$. 
Myhill-Nerode congruence for a forest language $L$.

Two contexts and are called $L$-equivalent if
Myhill-Nerode congruence for a forest language $L$.

Two contexts and are called $L$-equivalent if for every context and every forest
Myhill-Nerode congruence for a forest language $L$.

Two contexts $\text{ and }$ are called $L$-equivalent if
for every context $\text{ and every forest }$ iff $\in L$ $\in L$.
Main Theorem.
A forest language is piecewise testable iff the following holds for all sufficiently large $n$
Main Theorem.
A forest language is piecewise testable iff
the following holds for all sufficiently large $n$

if

is a piece of

, then
Main Theorem.
A forest language is piecewise testable iff the following holds for all sufficiently large $n$

if \[ \text{is a piece of} \] \[ \text{, then} \]

\[ n \text{ times} \] \[ \text{is equivalent} \] \[ n \text{ times} \]

\[ n \text{ times} \] \[ \text{is equivalent} \] \[ n \text{ times} \]
Main Theorem.

A forest language is piecewise testable if the following holds for all sufficiently large $n$:

If this criterion is decidable, then we also have variants of the theorem for:
- tree languages
- commutative pieces
- pieces with closest common ancestor

\[ \text{is equivalent to} \]
The language $*$ has a $J$-trivial syntactic monoid, but is not piecewise testable is confused with
Big project: understand the expressive power of first-order logic on trees.
Big project: understand the expressive power of first-order logic on trees.
Big project: understand the expressive power of first-order logic on trees.
Big project: understand the expressive power of first-order logic on trees.

Easy exercise

regular languages

$\Sigma_1(\leq)$

$\Pi_1(\leq)$

$FO(\leq)$
Big project: understand the expressive power of first-order logic on trees.

Regular languages

Easy exercise
Big project: understand the expressive power of first-order logic on trees.

This paper

Easy exercise

regular languages

$\Sigma_1(\leq)$

$\Pi_1(\leq)$

$\text{Bool}(\Sigma_1(\leq))$

$FO(\leq)$
Big project: understand the expressive power of first-order logic on trees.

This paper

Easy exercise

regular languages

$FO(\leq)$

$\Sigma_2(\leq)$

$\Sigma_1(\leq)$

$\Pi_1(\leq)$

$\Delta_2(\leq)$

$\Pi_2(\leq)$

$Bool(\Sigma_1(\leq))$
Big project: understand the expressive power of first-order logic on trees.

This paper

BS, ICALP 08

Easy exercise

regular languages

$\Sigma_1(\leq)$

$\Pi_1(\leq)$

$\Delta_2(\leq)$

$\Sigma_2(\leq)$

$\Pi_2(\leq)$

$\text{FO}(\leq)$

$\text{Bool}(\Sigma_1(\leq))$
Big project: understand the expressive power of first-order logic on trees.

Regular languages

This paper

BS, ICALP 08

Easy excercise

\[ \Sigma_1(\leq) \rightarrow \Pi_1(\leq) \rightarrow \Delta_2(\leq) \]

\[ FO(\leq) =? \]

\[ \Sigma_2(\leq) =? \]

\[ \Pi_2(\leq) =? \]