Forest Algebras

Igor Walukiewicz
joint work with Mikolaj Bojanczyk and Howard Straubing

CNRS, LaBRI Bordeaux

Bordeaux, November 2007
The goal

Understand logics on (finite) trees.

Some more prominent tree logics

- CTL, CTL*, μ-calculus,
- FOL, MSOL, chain logic, anti-chain logic.

We understand relations between these logics

\[
\begin{align*}
\mu\text{-calculus} & \equiv \text{MSOL} & \equiv \text{anti-chain logic} \\
\text{CTL}^* & \equiv \text{FOL} & \subset \text{chain logic}
\end{align*}
\]

The case of binary trees
The goal

Understand logics on (finite) trees.

Some more prominent tree logics

- CTL, CTL\(^*\), \(\mu\)-calculus,
- FOL, MSOL, chain logic, anti-chain logic.

We understand relations between these logics

\[\mu\text{-calculus} \nsubseteq \mathcal{C} \nsubseteq \text{MSOL} \nsubseteq \mathcal{C} \nsubseteq \text{anti-chain logic}\]

\[\text{CTL}^* \nsubseteq \mathcal{C} \nsubseteq \text{FOL} \nsubseteq \mathcal{C} \nsubseteq \text{chain logic}\]

The case of unranked, unordered trees

Problem

Given a property, decide in which logic it can be expressed.
The goal

**Understand logics on (finite) trees.**

**Some more prominent tree logics**

- CTL, CTL*, µ-calculus,
- FOL, MSOL, chain logic, anti-chain logic.

**We understand relations between these logics**

<table>
<thead>
<tr>
<th>µ-calculus</th>
<th>⊊</th>
<th>MSOL</th>
<th>⊊</th>
<th>anti-chain logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>⊊</td>
<td>FOL</td>
<td>⊊</td>
<td>chain logic</td>
<td></td>
</tr>
<tr>
<td>⊊</td>
<td>CTL</td>
<td>⊊</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The case of unranked, unordered trees

**Problem**

Given a property, decide in which logic it can be expressed.
An example of a surprising expressive power

**Even depth property (Potthoff)**
- Consider the property of binary trees saying that all leaves in the forest are at even depth.
- This property can be expressed in FO[$\leq$] (with the descendant order).

**Mixed parity nodes**
A mixed parity node has a path to a leaf of even length, and another one of odd length.

**The case when there are no mixed parity nodes**

**How to check if there are mixed parity nodes**
An example of a language not in first-order

**Positive boolean expressions**

Let $A = \{\lor, \land, 0, 1\}$, and consider the set of trees that evaluate to 1 (and are well-formed).

**Theorem (Heuter, Potthoff)**

The above language is not definable in chain logic. (This language is aperiodic.)

**Theorem (Potthoff, Thomas)**

The above language is definable in antichain logic.
**How it is done for words**

**Definition (Recognition)**

A language $L$ is recognized by a monoid $S$ if there are $\alpha : \Sigma^* \rightarrow S$ and $F \subseteq S$ such that $\alpha^{-1}(F) = L$.

**Definition (Syntactic monoid for $L$)**

- Define $v_1 \sim_L v_2$ iff for all $u, w \in \Sigma^*$: $uv_1w \in L$ iff $uv_2w \in L$.
- This is an equivalence relation so we can take $\langle \Sigma^*/\sim_L, \cdot, \varepsilon \rangle$.

**Definition (Apperiodicity)**

A monoid $\langle S, \cdot \rangle$ is aperiodic iff there is $n$ such that $s^n = s^{n+1}$ for all $s \in S$.

**Theorem (Schützenberger, McNaughton & Papert)**

A language is FOL definable iff its syntactic monoid is aperiodic.
A naive example comparing automata and algebra

A FOL definable language

Let $L = (ab)^* c\Sigma^*$. It is first-order definable.

The minimal automaton for $L$

Remarks

- The aperiodicity property is not that visible from the structure of the automaton.
- Every property of syntactic algebra is also a property of the minimal automaton. The converse is not true.
- This is good, as long as we know that properties of interest are properties of syntactic algebras.
How to do equally well for trees?

The big plan

- Define an algebraic notion of recognition.
- This should give a notion of syntactic algebra (invariant, for a given language).
- Decide the properties of the language looking at the properties of its syntactic algebra.
  - It is convenient to look at equational properties.
  - If we have an Eilenberg like theorem we can deduce that there is an equational characterization without knowing what it is.
Related work

- [Nivat & Podelski’87] Looking only at the structure contexts with application. Not enough to characterize FOL.
- [Salehi & Steinby’ 07] Variety theorem for the Wilke’s setting.
- [Esik’99]
- [Esik & Weil’ 05] Preclones. Algebra of all term, i.e., contexts of arbitrary arity.
- [Thomas’84] Regular expressions for trees.
- [Heuter’89] Regular expressions for FOL.
- [Potthoff’] Star free=regular. Apperiodic is more than star free.
- [Bojańczyk & W.’04] Characterization of EF.
- [Benedikt & Segufin’05-’07] Characterization of FO\(\text{succ}\).
- [Bojańczyk ’07] Characterization of EF + F\(^{-1}\).
Plan

- Forests, contexts, and some operations.
- Forest algebras.
- Recognition.
- Automata over forests.
- Syntactic forest algebras.
- Simple applications.
- Pseudovarieties.
- Eilenberg theorem for forest algebras.
- Applications.
- Related and further work.
Plan

- Forests, contexts, and some operations.
- Forest algebras.
- Recognition.
- Automata over forests.
- Syntactic forest algebras.
- Simple applications.
- Pseudovarieties.
- Eilenberg theorem for forest algebras.
- Applications.
- Related and further work.
**Definition (Trees, Forests)**

- A **A-tree** is a partial mapping \( t : \mathbb{N}^* \rightarrow A \) with finite and prefix closed domain.
- A **Forest** is a finite sequence of trees.
**Definition (Contexts)**

A $A$-context is a $(A \cup \{\ast\})$-forest, with $\ast$ occurring in exactly one leaf; called a **hole**.
Definition (Contexts)

A $A$-context is a $(A \cup \{\ast\})$-forest, with $\ast$ occurring in exactly one leaf; called a hole.
**Definition (Contexts)**

A *A-context* is a \((A \cup \{\ast\})\)-forest, with \(\ast\) occurring in exactly one leaf; called a **hole**.

We have two operations:

- **context substitution** \(p(t)\), and
- **context composition** \(p \cdot q\).
**Definition (Contexts)**

A $A$-context is a $(A \cup \{\ast\})$-forest, with $\ast$ occurring in exactly one leaf; called a hole.

We have two operations:

- **context substitution** $p(t)$, and **context composition** $p \cdot q$. 

![Diagram of a context structure]
**Definition (Contexts)**

A *$A$-context* is a $(A \cup \{\ast\})$-forest, with $\ast$ occurring in exactly one leaf; called a *hole*.

We have two operations:
- **Context substitution** $p(t)$, and **context composition** $p \cdot q$.

![Image of context substitution and composition](image-url)
Definition (Contexts)

A $A$-context is a $(A \cup \{\star\})$-forest, with $\star$ occurring in exactly one leaf; called a hole.

We have two operations:
context substitution $p(t)$, and context composition $p \cdot q$. 

\[ a \]
\[ \begin{array}{c}
  b \\
  b'
\end{array} \]
\[ c \]
\[ \begin{array}{c}
  * \\
  *
\end{array} \]
Forest and context monoid

**Forest monoid**

Forest monoid consists of the set of $A$-forests, with concatenation operation (it is not commutative), and the empty tree.

**Context monoid**

Context monoid consists of the set of $A$-contexts with context composition.

**Notation**

Forest concatenation will be denoted by $+$ and context composition by $\cdot$:

\[ s + t, \quad p \cdot q \]

One letter trees and contexts are just denoted by letters:

\[ b(t_1 + t_2 + \cdots + t_n) \]
**Actions of forests and contexts**

### Action of contexts on forests

If $p$ is a context and $t$ is a forest then $p(t)$ is the tree obtained by the substitution of $p$ in the hole of $t$. We have $(p \cdot q)(t) = p(q(t))$.

![Diagram showing the action of contexts on forests](attachment:diagram.png)

### From forests to contexts

If $t$ is a forest then we have the context $in_l(t)$. Similarly we have $in_r(t)$.
Plan

- Forests, contexts, and some operations.
- Forest algebras.
- Recognition.
- Automata over forests.
- Syntactic forest algebras.
- Simple applications.
- Pseudovarieties.
- Eilenberg theorem for forest algebras.
- Applications.
- Related and further work.
Algebraic structure

**Forest algebra** \( \langle H, V, \text{act}, \text{in}_l, \text{in}_r \rangle \)

- Two monoids: \( H \) and \( V \). We denote their operations by \(+\) and \(\cdot\), respectively.
- An action \( \text{act} : H \times V \to H \). We write \( vh \) for \( \text{act}(h, v) \).
- Two operations \( \text{in}_l, \text{in}_r : H \to V \).
- Axioms:

  - **ACTION** \((v \cdot w)h = v(wh)\);
  - **INSERTION** \( \text{in}_l(g)h = g + h \) and \( \text{in}_r(g)h = h + g \);
  - **FAITHFULNESS** for every two distinct \( v, w \in V \) there is \( h \in H \) with \( vh \neq wh \);

**Some conventions**

- \( H \) will be called **horizontal monoid**, and \( V \) **vertical monoid**.
- The neutral element of the horizontal monoid is \( 0 \) and the operation is \(+\).
- The neutral element of the vertical monoid is \( 1 \) and the operation is \(\cdot\).
- Action is written on the left: so \( \text{act}(h, v) \) is written \( vh \).
**Algebraic structure**

**Forest algebra** \( \langle H, V, act, in_l, in_r \rangle \)

- Two monoids: \( H \) and \( V \). We denote their operations by \(+\) and \(\cdot\), respectively.
- An action \( act : H \times V \to H \). We write \( vh \) for \( act(h, v) \).
- Two operations \( in_l, in_r : H \to V \).
- Axioms:
  - **ACTION** \( (v \cdot w)h = v(wh) \);
  - **INSERTION** \( in_l(g)h = g + h \) and \( in_r(g)h = h + g \);
  - **FAITHFULNESS** for every two distinct \( v, w \in V \) there is \( h \in H \) with \( vh \neq wh \);

**Example**

- \( H \) any monoid, \( V = H \to H \) with composition as multiplication.
- The action is application.
- The insertions are determined by the insertion axiom.
Forest algebra \( \langle H, V, \text{act}, \text{in}_l, \text{in}_r \rangle \)

- Two monoids: \( H \) and \( V \). We denote their operations by \( + \) and \( \cdot \), respectively.
- An action \( \text{act} : H \times V \rightarrow H \). We write \( vh \) for \( \text{act}(h, v) \).
- Two operations \( \text{in}_l, \text{in}_r : H \rightarrow V \).
- Axioms:
  - **ACTION** \((v \cdot w)h = v(wh)\);
  - **INSERTION** \( \text{in}_l(g)h = g + h \) and \( \text{in}_r(g)h = h + g \);
  - **FAITHFULNESS** for every two distinct \( v, w \in V \) there is \( h \in H \) with \( vh \neq wh \);

Remarks

- Every element from \( H \) is of the form \( v0 \) for some \( v \in V \): take \( v = \text{in}_l h \).
  \[ v0 = (\text{in}_l h)0 = h + 0 = h. \]
- Mappings \( \text{in}_l, \text{in}_r : H \rightarrow V \) are, injective, monoid homomorphisms:
  \[ \text{in}_l (h_1 + h_2) = \text{in}_l (h_1) \text{in}_l (h_2) \quad \text{and} \quad \text{in}_l (0) = 1. \]
Free forest algebra

For an alphabet $A$, the free forest algebra $A^\Delta$ is:

- The horizontal monoid is the set of forests over $A$.
- The vertical monoid is the set of contexts over $A$.
- The action is the substitution of forests in contexts.
- The $\text{in}_l$ function takes a forest and transforms it into a context with a hole to the right of all the roots in the forest. Similarly for $\text{in}_r$ but the hole is to the left of the roots.
Morphisms

**Morphism** \((\alpha, \beta) : (H, V) \to (G, V)\)

It is a pair of monoid morphisms \((\alpha : H \to G, \beta : V \to W)\) with additional requirements ensuring that the operations are preserved:

\[
\alpha(vh) = \beta(v)\alpha(h) \quad \beta(\text{in}_l(h)) = \text{in}_l(\alpha(h)) \quad \text{and} \quad \beta(\text{in}_r(h)) = \text{in}_r(\alpha(h))
\]

**Remark**

The component \(\alpha\) is determined by \(\beta\) via

\[
\alpha(h) = \alpha(h + 0) = \alpha(\text{in}_l(h)0) = \beta(\text{in}_l(h))\alpha(0) ,
\]

where \(\alpha(0)\) must be the neutral element in \(G\) as \(\alpha\) is a monoid morphism.

**Lemma (Free algebra)**

For every forest algebra \((H, V)\): every function \(f : A \to V\) can be uniquely extended to a morphism \((\alpha, \beta) : A^\Delta \to (H, V)\) such that \(\beta(a) = f(a)\) for every \(a \in A\).
Plan

- Forests, contexts, and some operations.
- Forest algebras.
- **Recognition.**
- Automata over forests.
- Syntactic forest algebras.
- Simple applications.
- Pseudovarieties.
- Eilenberg theorem for forest algebras.
- Applications.
- Related and further work.
Recognizing a Language $L$ of $A$-Forests

$L$ is recognized by a surjective morphism $(\alpha, \beta) : A^\Delta \rightarrow (H, V)$ if $L$ is the inverse image $\alpha^{-1}(F)$ of some $F \subseteq H$. $L$ is recognizable if it is recognized by a finite $(H, V)$.

Even Number of Nodes

- Let $L$ be the set of forest with even number of nodes.
- Consider the forest algebra where $H = V = \langle \{0, 1\}, 0, + (\text{mod} 2) \rangle$
- The action is also addition mod 2 and insertions are determined uniquely.
- The morphism maps a context to 0 if it has even number of nodes.
- The accepting set of $L$ is 0.

\[
\beta(a) = 1 \\
\beta(b) = 1
\]
Recognizing a language $L$ of $A$-forests

$L$ is recognized by a surjective morphism $(\alpha, \beta) : A^\Delta \to (H, V)$ if $L$ is the inverse image $\alpha^{-1}(F)$ of some $F \subseteq H$. $L$ is recognizable if it is recognized by a finite $(H, V)$.

Even number of nodes

- Let $L$ be the set of forest with even number of nodes.
- Consider the forest algebra where $H = V = \langle\{0, 1\}, 0, +(\text{mod} 2)\rangle$.
- The action is also addition mod 2 and insertions are determined uniquely.
- The morphism maps a context to 0 if it has even number of nodes.
- The accepting set of $L$ is 0.

\[
\beta(a) = 1 \\
\beta(b) = 1
\]
**Recognition**

**Recognizing a language \( L \) of \( A \)-forests**

\( L \) is recognized by a surjective morphism \((\alpha, \beta) : A^\Delta \rightarrow (H, V)\) if \( L \) is the inverse image \( \alpha^{-1}(F) \) of some \( F \subseteq H \). \( L \) is recognizable if it is recognized by a finite \((H, V)\).

**Even number of nodes**

- Let \( L \) be the set of forest with even number of nodes.
- Consider the forest algebra where \( H = V = \langle \{0, 1\}, 0, + (\mod 2) \rangle \)
- The action is also addition and insertions are determined uniquely.
- The morphism maps a context to 0 if it has even number of nodes.
- The accepting set of \( L \) is 0.

**Label testable**

- \( L \) is label-testable if membership of \( t \) in \( L \) depends on the labels that occur in \( t \).
- We set \( H = V = \mathcal{P}(A) \) with union as operation.
- The action is also union, the insertions are determined.
- Recognizing morphism takes a context to the set of labels appearing in it.
Plan

- Forests, contexts, and some operations.
- Forest algebras.
- Recognition.
- **Automata over forests.**
- Syntactic forest algebras.
- Simple applications.
- Pseudovarieties.
- Eilenberg theorem for forest algebras.
- Applications.
- Related and further work.
Automata

Automaton for unranked trees (first approximation)

\[ \mathcal{A} = \langle Q, A, \delta, F \subseteq Q \rangle \]

where \( \delta : (A \times Q^* \rightarrow Q) \)
**Automata**

**Automaton for unranked trees (first approximation)**

\[ A = \langle Q, A, \delta, F \subseteq Q \rangle \]

where \( \delta : (A \times Q^* \rightarrow Q) \)

**Definition (Automaton for unranked trees)**

\[ A = \langle (Q, 0, +), A, \delta, F \subseteq Q \rangle \]

where \( \delta : (A \times Q \rightarrow Q) \)

**Example**

```
\begin{align*}
q_b &= \delta(b, q_b^1 + q_b^2) \\
q_b^1 &= \delta(b, q_a^3) \\
q_b^2 &= \delta(b, q_a^2) \\
q_a &= q_a + q_a + q_a
\end{align*}
```
Between algebra and automata

**Algebra → Automata**

Take a morphism \((\alpha, \beta) : A^\Delta \to (H, V)\) and \(F \subseteq H\).
We construct the automaton
\[ A^{(\alpha, \beta)} = \langle H, A, \delta, F \rangle \quad \text{where:} \quad \delta(a, h) = \beta(a) h. \]
Claim: \(L(A^{(\alpha, \beta)}) = \alpha^{-1}(F)\).

**Automata → Algebra**

Take an automaton \(A = \langle (Q, +, 0), A, \delta : A \times Q \to Q, F \rangle\).
We construct the algebra \((H, V)\) with
- \(H = (Q, +, 0)\);
- \(V : Q \to Q\) with the composition operation.
- \(act\) is the application \(v(h)\).
Claim: Take the unique homomorphism \((\alpha_A, \beta_A) : A^\Delta \to (H, V)\) s.t.:
\[ \beta_A(a) = \delta(a) \quad \text{for all} \quad a \in A. \]
We have \(\alpha_A(t) = t^A\).
Remark: \((H, V)\) so defined may be not faithful.
Plan

- Forests, contexts, and some operations.
- Forest algebras.
- Recognition.
- Automata over forests.
- **Syntactic forest algebras.**
- Simple applications.
- Pseudovarieties.
- Eilenberg theorem for forest algebras.
- Applications.
- Related and further work.
Syntactic algebra for forest languages

Fix a language $L$ of $A$-forests.

**Equivalences**

- Two $A$-forests $s, t$ are $L$-equivalent if for every context $p$, either both or none of the forests $ps, pt$ belong to $L$.
- Two $A$-contexts $p, q$ are $L$-equivalent if for every forest $t$, the forests $pt$ and $qt$ are $L$-equivalent.

**Definition (Syntactic forest algebra for $L$)**

The syntactic forest algebra for $L$ is the quotient of $A^\Delta$ with respect to $L$-equivalence.

The syntactic morphism $(\alpha^L, \beta^L)$ assigns to every element of $A^\Delta$ its equivalence class.

**Fact**

A language $L$ of $A$-forests is recognized by a the syntactic morphism $(\alpha^L, \beta^L)$.

Moreover, any morphism $(\alpha, \beta) : A^\Delta \to (H, V)$ that recognizes $L$ can be extended by a morphism $(\alpha', \beta') : (H, V) \to (H^L, V^L)$ so that $\beta' \circ \beta = \beta^L$.

**Computing syntactic algebras**

If $L$ is recognizable then the syntactic algebra is finite. Given an automaton for $L$, its syntactic algebra can be computed by a straightforward fixpoint algorithm.
Plan

- Forests, contexts, and some operations.
- Forest algebras.
- Recognition.
- Automata over forests.
- Syntactic forest algebras.
- Simple applications.
- Pseudovarieties.
- Eilenberg theorem for forest algebras.
- Applications.
- Related and further work.
An example: label-testable languages

**Theorem**

A language is label testable iff its syntactic algebra satisfies the equations:

\[ vv = v \quad vw = wv. \]

**Proof**

- If \( L \) is label testable then its syntactic algebra satisfies the equations.
- If an algebra satisfies the equations then \( H \) is also idempotent and commutative.
- It also holds that \( v(h) = h + v0 \).
- This allows to show that every forest is equivalent to the one of the form:
  \[ a_10 + \cdots + a_n0. \]
An example: $\Sigma_1$-languages

$\Sigma_1$ formulas

A $\Sigma_1$ formula is a formula of first-order logic, where only existential quantifiers appear in the quantifier prenex normal form. (There is the descendant order in the signature).

Theorem

Let $L$ be a forest language, and let $(\alpha, \beta)$ be its syntactic morphism. A language $L$ is definable in $\Sigma_1$ if and only if $vh \in \alpha(L)$ implies $vwh \in \alpha(L)$.

Proof

$\Rightarrow$ If $t \in L$ then any extension of $t$ is in $L$.

$\Leftarrow$ For every $h \in H$ consider minimal size forests mapped to $h$. Every such forest is of bounded size.

A forest is in $L$ if it is an extension of a minimal forest that is mapped to $h \in \alpha(L)$. 
**An example: EF-logic**

**Syntax and semantics**
- If $a$ is a letter, then $a$ is a formula true in trees whose root label is $a$.
- EF formulas are closed under boolean connectives.
- If $\varphi$ is an EF formula, then $EF\varphi$ is an EF formula true in trees having a proper subtree satisfying $\varphi$.

**Definition (Definability in EF)**

A forest language $L$ is definable in EF if for some $a \in A$ and for some $\varphi$ of $EF$:

$$L = \{ t : at \models \varphi \}.$$ 

**Theorem**

A forest language is definable in EF if and only if its syntactic forest algebra satisfies the following equations

$$g + h = h + g$$
$$vh = h + vh.$$
Example: EF + F⁻¹

**Definition (EF + F⁻¹)**

To the definition of EF we add the clause:
- If φ is a formula then F⁻¹φ is a formula true in a node of a tree if there is a proper ancestor satisfying φ.

**Theorem (Bojańczyk)**

Forest language is definable in EF + F⁻¹ iff its syntactic forest algebra satisfies the following:

\[
\begin{align*}
  h + h &= h \\
  g + h &= h + g \\
  (vw)\omega &= (vw)^{\omega}w(vw)^{\omega} \\
  (u_1 w_1)^{\omega}(u_2 w_2)^{\omega} &= (u_1 w_1)^{\omega}u_1 w_2(u_2 w_2)^{\omega} \quad \text{for } u_1 \vdash u_2, w_1 \vdash w_2
\end{align*}
\]

we need to assume here that \(v_i, w_i \neq 1\).
Plan

- Forests, contexts, and some operations.
- Forest algebras.
- Recognition.
- Automata over forests.
- Syntactic forest algebras.
- Simple applications.
- Pseudovarieties.
- Eilenberg theorem for forest algebras.
- Applications.
- Related and further work.
Towards varieties of forest algebras

Forest algebra \( \langle H, V, \text{act}, \text{in}_l, \text{in}_r \rangle \)

- Two monoids: \( H \) and \( V \). We denote their operations by \(+\) and \(\cdot\).
- An action \( \text{act} : H \times V \rightarrow H \). We write \( vh \) for \( \text{act}(h, v) \).
- Two operations \( \text{in}_l, \text{in}_r : H \rightarrow V \).
- Axioms:
  
  **ACTION** \( (v \cdot w)h = v(wh) \);
  
  **INSERTION** \( \text{in}_l(g)h = g + h \) and \( \text{in}_r(g)h = h + g \);
  
  **FAITHFULNESS** for every two distinct \( v, w \in V \) there is \( h \in H \) with \( vh \neq wh \);

Remark

A subalgebra, quotient, or a homomorphic image of a forest algebra may not be a forest algebra (because of faithfulness.).
The solution is to take the faithful quotient of the result.
Varieties of forest algebras

Pseudovariety

A pseudovariety of finite forest algebras is a collection $\mathcal{V}$ of finite forest algebras with the following properties:

- $\mathcal{V}$ is closed under binary product.
  If $(H, V), (G, U) \in \mathcal{V}$ then $(H, V) \times (G, U) \in \mathcal{V}$.

- $\mathcal{V}$ is closed under faithful quotients of homomorphic images.
  If $(H, V) \in \mathcal{V}$ and $(H', V')$ is its homomorphic image, then $faith(H', V') \in \mathcal{V}$.

- $\mathcal{V}$ is closed under faithful quotients of subalgebras.
  If $(H, V) \in \mathcal{V}$ and $(H', V')$ is its subalgebra then $faith(H', V') \in \mathcal{V}$.

Example

- Equation is a pair of terms in a signature of forest algebras (with variables of two types).
- An algebra satisfies an equation if for all valuations of variables the values of the two terms are the same.
- Everything definable by a finite set of equations is a variety.
Examples

Theorem

A language is label testable iff its syntactic algebra satisfies the equations:

\[ vv = v \quad vw = wv. \]

Theorem

Let \( L \) be a forest language, and let \((\alpha, \beta)\) be its syntactic morphism. A language \( L \) is definable in \( \Sigma_1 \) if and only if \( vh \in \alpha(L) \) implies \( vwh \in \alpha(L) \).

Theorem

A forest language is definable in \( EF \) if and only if its syntactic forest algebra satisfies the following equations

\[
\begin{align*}
g + h &= h + g \\
vh &= h + vh.
\end{align*}
\]
Plan

- Forests, contexts, and some operations.
- Forest algebras.
- Recognition.
- Automata over forests.
- Syntactic forest algebras.
- Simple applications.
- Pseudovarieties.
- Eilenberg theorem for forest algebras.
- Applications.
- Related and further work.
Let $\mathcal{V}$ be a variety of algebras. For every finite alphabet $A$ define
\[
\mathcal{V}(A) = \{ L \subseteq H_A : (H^L, V^L) \in \mathcal{V} \}.
\]
We call $\mathcal{V}$ the \textit{variety of forest languages} associated to $\mathcal{V}$, and write
\[
\mathcal{V} \mapsto \mathcal{V}.
\]

\textbf{Theorem}

\textit{The mapping $\mathcal{V} \mapsto \mathcal{V}$ is one-to-one.}
Second correspondence theorem

A prefix operator

If $L$ is a language of $A$-forests and $p$ is a $A$-context then we define

$$p^{-1}L = \{ t : pt \in L \}.$$ 

Theorem

Let $\mathcal{W}$ be an operator assigning to each finite alphabet $A$ a family $\mathcal{W}(A)$ of $A$-languages. $\mathcal{W}$ is a variety of languages if and only if the following three conditions hold:

1. for all finite alphabets $A$, $\mathcal{W}(A)$ is closed under boolean operations.
2. for all finite alphabets $A$, if $L \in \mathcal{W}(A)$ and $p$ is an $A$-context then $p^{-1}L \in \mathcal{W}(A)$.
3. for all finite alphabets $A$ and $B$, if $(\alpha, \beta) : A^\Delta \rightarrow B^\Delta$ is a homomorphism, and $L \in \mathcal{W}(B)$, then $\alpha^{-1}(L) \in \mathcal{W}(A)$. 
Plan

- Forests, contexts, and some operations.
- Forest algebras.
- Recognition.
- Automata over forests.
- Syntactic forest algebras.
- Simple applications.
- Pseudovarieties.
- Eilenberg theorem for forest algebras.
- Applications.
- Related and further work.
**Application: First-order logic**

**First-order logic over forest, \( \text{FO}[\leq] \)**
- We have a unary predicate \( Q_a \) for every letter in the alphabet.
- We have \( \leq \) relation interpreted as a descendant relation in the tree.
- Example: \( \exists x \forall y. (x \leq y) \) defines trees.

**Fact**
\( \text{FO}[\leq] \) is a variety of forest languages.

**Fact**
Chain logic is a variety of forest languages.

**Remark**
\( \text{FO}[\text{succ}] \) is not a variety of forest languages. Closure under inverse homomorphic images fails.
Defining varieties by equations

Let \( \{ l_i = r_i \}_{i>0} \) be an infinite set of equations.

An algebra ultimately satisfies \( \{ l_i = r_i \}_{i>0} \) if it satisfies all but finitely many of these equations.

A variety ultimately defined by \( \{ l_i = r_i \}_{i>0} \) is the set of finite algebras ultimately satisfying these equations.

**Theorem**

If \( \forall \) is a pseudovariety, then it is ultimately defined by some set of equations.

**Corollary**

There is a set of equations ultimately defining FO[\( \leq \)].
Plan

- Forests, contexts, and some operations.
- Forest algebras.
- Recognition.
- Automata over forests.
- Syntactic forest algebras.
- Simple applications.
- Pseudovarieties.
- Eilenberg theorem for forest algebras.
- Applications.
- Related and further work.
In unranked trees we have two orders:

- $\leq_H$ the horizontal order, i.e., order between siblings.
- $\leq_V$ the vertical order, i.e., ancestor-descendant order.

We write $FO[\leq_H, \leq_V]$ or $FO[\leq_V]$ to show which order is present.

UHV: Given a regular language $L$ of unranked trees, decide if $L$ can be defined in $FO[\leq_H, \leq_V]$.

$\uparrow$

BHV: Given a regular language $L$ of binary trees, decide if $L$ can be defined in $FO[\leq_H, \leq_V]$.

$\downarrow$

BV: Given a regular language $L$ of binary trees, decide if $L$ can be defined in $FO[\leq_V]$.

$\downarrow$

UV: Given a regular language $L$ of unranked trees, decide if $L$ can be defined in $FO[\leq_V]$.

$\uparrow$

CTL*: Given a regular language $L$ of unranked trees, decide if $L$ can be defined in CTL*.
Comments on missing reductions

**Fact**
The language “unranked trees with even number of nodes” is definable in MSOL[≤_H, ≤_V] but not in MSOL[≤_V].

**Fact**
The language “binary trees where all leaves have even depth” can be defined in FO[≤_H, ≤_V] but not in FO[≤_H].
Related work

- [Nivat & Podelski’87] Looking only at the structure contexts with application. Not enough to characterize FOL.
- [Salehi & Steinby’ 07] Variety theorem for the Wilke’s setting.
- [Esik’99]
- [Esik & Weil’ 05] Preclones. Algebra of all term, i.e., contexts of arbitrary arity.
- [Thomas’84] Regular expressions for trees.
- [Heuter’89] Regular expressions for FOL.
- [Potthoff’] Star free=regular. Apperiodic is more than star free.
- [Bojańczyk & W.’04] Characterization of EF.
- [Benedikt & Segufin’05-07] Characterization of FO[$\text{succ}$].
- [Bojańczyk ’07] Characterization of EF + $F^{-1}$. 
Conclusions and some other results

- An algebraic framework for recognizing forest languages. It is based on a new interpretation of a transformation semigroup.
- Basic results from the theory of semigroups carry over to forest algebras.
- The framework permits to express some characterizations in a simple way.
- “By design” there are some things not expressible in the framework. Ex. deterministic tree languages, or frontier languages.
- The variety theorem guarantees that there is an equational characterisation for some interesting logics.
- Considering unranked forests is interesting also for the case of binary forests.
- The notion of the wreath product of transformation semigroups generalizes naturally to forest algebras.
- As in the word case one gets the wreath product principle connecting the wreath product with the cascade product of automata, and the formula substitution.
- This gives characterizations of many known logics in terms of wreath products of simple varieties.