Regular tree languages, cardinality predicates, and addition-invariant FO

Frederik Harwath
Nicole Schweikardt
Goethe-Universität Frankfurt am Main

STACS 2012
Paris
Regular tree languages — sets of finite, ranked, coloured trees recognized by (bottom-up) tree automata

Tree languages are sets of finite trees which are

- ranked (there exists a number $r$ such that each node of a tree in the language has $\leq r$ children)
- coloured with colours from a finite set $\Sigma$.

Example: Trees with odd number of red leaves
Regular tree languages — sets of finite, ranked, coloured trees recognized by (bottom-up) tree automata

Tree languages are sets of finite trees which are

- ranked (there exists a number $r$ such that each node of a tree in the language has $\leq r$ children)
- coloured with colours from a finite set $\Sigma$.

Regular tree languages: tree languages recognized by (bottom–up) tree automata.

Example: Trees with odd number of red leaves
**Regular tree languages** — sets of finite, ranked, coloured trees recognized by (bottom-up) tree automata

Tree languages are sets of finite trees which are

- **ranked** (there exists a number $r$ such that each node of a tree in the language has $\leq r$ children)
- **coloured** with colours from a finite set $\Sigma$.

Regular tree languages: tree languages recognized by (bottom–up) tree automata.

**Example:**

*Trees with odd number of red leaves*
Regular tree languages — sets of finite, ranked, coloured trees recognized by (bottom-up) tree automata

Tree languages are sets of finite trees which are

- ranked (there exists a number $r$ such that each node of a tree in the language has $\leq r$ children)
- coloured with colours from a finite set $\Sigma$.

Regular tree languages: tree languages recognized by (bottom–up) tree automata.

**Example:**

Trees with odd number of red leaves

```
odd  ⊥ ⊥ ⊥ ⊥ ⊥
```

```
```
Regular tree languages — sets of finite, ranked, coloured trees recognized by (bottom-up) tree automata

Tree languages are sets of finite trees which are

- ranked (there exists a number $r$ such that each node of a tree in the language has $\leq r$ children)
- coloured with colours from a finite set $\Sigma$.

Regular tree languages: tree languages recognized by (bottom–up) tree automata.

**Example:**

Trees with odd number of red leaves
Regular tree languages — sets of finite, ranked, coloured trees recognized by (bottom-up) tree automata

Tree languages are sets of finite trees which are

- ranked (there exists a number \( r \) such that each node of a tree in the language has \( \leq r \) children)
- coloured with colours from a finite set \( \Sigma \).

Regular tree languages: tree languages recognized by (bottom–up) tree automata.

Example: Trees with odd number of red leaves

\[
\text{odd} \quad \text{even} \quad \text{odd} \quad \text{odd} \quad \text{even} \quad \text{odd} \quad \text{even}
\]
Regular tree languages — sets of finite, ranked, coloured trees recognized by (bottom-up) tree automata

Tree languages are sets of finite trees which are

- ranked (there exists a number $r$ such that each node of a tree in the language has $\leq r$ children)
- coloured with colours from a finite set $\Sigma$.

Regular tree languages: tree languages recognized by (bottom–up) tree automata.

Example: 

Trees with odd number of red leaves
Regular tree languages — sets of finite, ranked, coloured trees recognized by (bottom-up) tree automata

Tree languages are sets of finite trees which are

- **ranked** (there exists a number \( r \) such that each node of a tree in the language has \( \leq r \) children)
- **coloured** with colours from a finite set \( \Sigma \).

Regular tree languages: tree languages recognized by (bottom–up) tree automata.

**Example:** 
*Trees with odd number of red leaves*
Regular tree languages — sets of finite, ranked, coloured trees recognized by (bottom-up) tree automata

Tree languages are sets of finite trees which are

- ranked (there exists a number $r$ such that each node of a tree in the language has $\leq r$ children)
- coloured with colours from a finite set $\Sigma$.

Regular tree languages: tree languages recognized by (bottom–up) tree automata.

**Example:**

*Trees with odd number of red leaves*
Regular tree languages — sets of finite, ranked, coloured trees recognized by (bottom-up) tree automata

Tree languages are sets of finite trees which are

- ranked (there exists a number \( r \) such that each node of a tree in the language has \( \leq r \) children)

- coloured with colours from a finite set \( \Sigma \).

Regular tree languages: tree languages recognized by (bottom–up) tree automata.

Example: Trees with odd number of red leaves
Regular tree languages — sets of finite, ranked, coloured trees recognized by (bottom-up) tree automata

Tree languages are sets of finite trees which are

- ranked (there exists a number $r$ such that each node of a tree in the language has $\leq r$ children)
- coloured with colours from a finite set $\Sigma$.

Regular tree languages: tree languages recognized by (bottom–up) tree automata.

**Example:**

Trees with odd number of red leaves
FO: First–order logic with successor relations

First–order logic with

- unary colour predicates, e.g. $P_{\text{red}}, P_{\text{blue}}, \ldots$, for each colour from a finite set $\Sigma$,
- binary successor/child predicates $S_1, S_2, \ldots, S_r$, for some rank $r$. 

Example:
Let $\Sigma := \{ \text{red}, \text{blue} \}$ and $r := 2$.

$$\phi := \forall u (P_{\text{red}}(u) \rightarrow (\exists v_1 \exists v_2 S_1(u, v_1) \land S_2(u, v_2) \land P_{\text{blue}}(v_1) \land P_{\text{blue}}(v_2))) \land (P_{\text{blue}}(u) \rightarrow ((\exists v S_1(u, v)) \rightarrow \forall v' \neg S_2(u, v'))) \land ((\exists v S_2(u, v)) \rightarrow \forall v' \neg S_1(u, v'))$$

$\phi \in L_{\text{FO}}$
FO: First–order logic with successor relations

First–order logic with

- unary colour predicates, e.g. $P_{\text{red}}, P_{\text{blue}}, \ldots$, for each colour from a finite set $\Sigma$,
- binary successor/child predicates $S_1, S_2, \ldots, S_r$, for some rank $r$. 

Example:

Let $\Sigma := \{\text{red}, \text{blue}\}$ and $r := 2$.

$$\psi := \forall u (P_{\text{red}}(u) \rightarrow (\exists v_1 \exists v_2 S_1(u, v_1) \land S_2(u, v_2) \land P_{\text{blue}}(v_1) \land P_{\text{blue}}(v_2))) \land (P_{\text{blue}}(u) \rightarrow ((\exists v S_1(u, v)) \rightarrow \forall v' \neg S_2(u, v')) \land ((\exists v S_2(u, v)) \rightarrow \forall v' \neg S_1(u, v'))).$$
FO: First–order logic with successor relations

First–order logic with

- unary colour predicates, e.g. $P_{\text{red}}, P_{\text{blue}}, \ldots$, for each colour from a finite set $\Sigma$,
- binary successor/child predicates $S_1, S_2, \ldots, S_r$, for some rank $r$. 

FO: First–order logic with successor relations

First–order logic with

- unary colour predicates, e.g. $P_{\text{red}}, P_{\text{blue}}, \ldots$, for each colour from a finite set $\Sigma$,
- binary successor/child predicates $S_1, S_2, \ldots, S_r$, for some rank $r$. 

Example:

Let $\Sigma := \{\text{red}, \text{blue}\}$ and $r := 2$.

\[
\phi := \forall u (P_{\text{red}}(u) \rightarrow (\exists v_1 \exists v_2 S_1(u, v_1) \land S_2(u, v_2) \land P_{\text{blue}}(v_1) \land P_{\text{blue}}(v_2)))
\land (P_{\text{blue}}(u) \rightarrow ((\exists v S_1(u, v)) \rightarrow \forall v' \neg S_2(u, v')) \land ((\exists v S_2(u, v)) \rightarrow \forall v' \neg S_1(u, v'))) \]
**FO: First–order logic with successor relations**

First–order logic with
- unary *colour predicates*, e.g. $P_{\text{red}}, P_{\text{blue}}, \ldots$, for each colour from a finite set $\Sigma$,
- binary *successor/child predicates* $S_1, S_2, \ldots, S_r$, for some rank $r$.

**Example:**

Let $\Sigma := \{\text{red, blue}\}$ and $r := 2$.

$$
\varphi := \forall u \left( P_{\text{red}}(u) \rightarrow (\exists v_1 \exists v_2 S_1(u, v_1) \land S_2(u, v_2) \land P_{\text{blue}}(v_1) \land P_{\text{blue}}(v_2)) \right)
$$

$$
\land \left( P_{\text{blue}}(u) \rightarrow \left( (\exists v S_1(u, v)) \rightarrow \forall v' \neg S_2(u, v') \right) \land \left( (\exists v S_2(u, v)) \rightarrow \forall v' \neg S_1(u, v') \right) \right)
$$
FO: First–order logic with successor relations

First–order logic with

• unary colour predicates, e.g. $P_{\text{red}}$, $P_{\text{blue}}$, . . ., for each colour from a finite set $\Sigma$,
• binary successor/child predicates $S_1$, $S_2$, . . . , $S_r$, for some rank $r$.

Example:

Let $\Sigma := \{\text{red}, \text{blue}\}$ and $r := 2$.

$$\varphi := \forall u \left( P_{\text{red}}(u) \rightarrow (\exists v_1 \exists v_2 \ S_1(u, v_1) \land S_2(u, v_2) \land P_{\text{blue}}(v_1) \land P_{\text{blue}}(v_2)) \right)$$

$$\land \left( P_{\text{blue}}(u) \rightarrow \left( (\exists v \ S_1(u, v)) \rightarrow \forall v' \neg S_2(u, v') \right) \land \left( (\exists v \ S_2(u, v)) \rightarrow \forall v' \neg S_1(u, v') \right) \right)$$

$\in L_\varphi$
**FO: First–order logic with successor relations**

First–order logic with
- unary **colour predicates**, e.g. $P_{red}, P_{blue}, \ldots$, for each colour from a finite set $\Sigma$,
- binary **successor/child predicates** $S_1, S_2, \ldots, S_r$, for some rank $r$.

**Example:**

Let $\Sigma := \{\text{red, blue}\}$ and $r := 2$.

$$\varphi := \forall u \left( P_{red}(u) \rightarrow (\exists v_1 \exists v_2 S_1(u, v_1) \land S_2(u, v_2) \land P_{blue}(v_1) \land P_{blue}(v_2)) \right)$$

$$\land \left( P_{blue}(u) \rightarrow \left( (\exists v S_1(u, v)) \rightarrow \forall v' \neg S_2(u, v') \right) \land \left( (\exists v S_2(u, v)) \rightarrow \forall v' \neg S_1(u, v') \right) \right)$$

![Diagram 1](image1)

$\in L_\varphi$

![Diagram 2](image2)

$\notin L_\varphi$
Logic On Words — i.e. trees of rank 1

MSO(\textless) = regular languages  \quad (\text{Büchi/Elgot/Trakhtenbrot})

(\text{MSO}(\textless) = \text{MSO})

\text{FO}(\textless) = \text{star-free regular languages}  \quad (\text{McNaughton, Papert 71})

\text{FO} = \text{locally threshold testable languages}  \quad (\text{Thomas 78})
Logic On Words — i.e. trees of rank 1

Question:
Given an automaton, is it decidable whether its language is definable in a given logic?

- (Büchi/Elgot/Trakhtenbrot)
- (McNaughton, Papert 71)
- (Thomas 78)
Logic On Words — i.e. trees of rank 1

**Question:**
Given an automaton, is it decidable whether its language is definable in a given logic?

**Solution:**
Characterisation of language classes by algebraic properties of their transition monoid.
Introduction

**FO with cardinality predicates**

**Addition–invariant FO on trees**

Logic On Words — i.e. trees of rank 1

**Question:**

Given an automaton, is it decidable whether its language is definable in a given logic?

**Solution:**

Characterisation of language classes by algebraic properties of their transition monoid.

**Transition monoid:**

- Each word induces a function from states to states (state at beginning of word $\rightarrow$ state after word)
- The set of all such functions forms a finite monoid w.r.t function composition.
Logic On Words — i.e. trees of rank 1

\[ \text{MSO}(\prec) = \text{regular languages} \quad \text{(Büchi/Elgot/Trakhtenbrot)} \]

\[ (\text{MSO}(\prec) = \text{MSO}) \]

\[ \text{FO}(\prec) = \text{star-free regular languages} \quad \text{(McNaughton, Papert 71)} \]

\[ \text{FO} = \text{locally threshold testable languages} \quad \text{(Thomas 78)} \]
Logic On Words — i.e. trees of rank 1

$$\text{MSO}(\prec) = \text{regular languages}$$

$$\text{(MSO}(\prec) = \text{MSO})$$

$$\text{FO}(\prec) = \text{star-free regular languages}$$

$$= \text{aperiodic} \quad \text{(Schützenberger 65)}$$

$$\text{FO} = \text{locally threshold testable languages}$$

$$= \text{aperiodic & closed under guarded swaps} \quad \text{(Beauquier, Pin 89)}$$
Logic On Words — i.e. trees of rank 1

\[ \text{MSO}(\prec) = \text{regular languages} \]
\[ (\text{MSO}(\prec) = \text{MSO}) \]
\[ \text{FO}(\prec) = \text{star-free regular languages} \]
\[ = \text{aperiodic} \]
\[ \text{FO} = \text{locally threshold testable languages} \]
\[ = \text{aperiodic & closed under guarded swaps} \quad \text{(Schützenberger 65)} \]
\[ = \text{aperiodic & closed under guarded swaps} \quad \text{(Beauquier, Pin 89)} \]

Decidable
Logic On Words — i.e. trees of rank 1

MSO(⟨) = regular languages
(MSO(⟨) = MSO)

FO(⟨) = star-free regular languages
    = aperiodic

FO = locally threshold testable languages
    = aperiodic & closed under guarded swaps

Decidable

Many further decidable characterisations known for extensions (e.g. modular quantifiers) and restrictions, e.g. FO^2(⟨), Δ_2(⟨), ΠΣ_1(⟨) of FO(⟨).
Logic on trees

<: descendant relation, i.e. transitive closure of parent–child relation.

\[ \text{MSO}(<) = \text{regular tree languages} \]

\[ (\text{MSO}(<) = \text{MSO}) \]

\[ \text{FO}(<) \not\subseteq \text{aperiodic} \]

\[ \text{FO} = \text{locally threshold testable languages} = \text{aperiodic & closed under guarded swaps} \]

(Thatcher, Wright 68/Doner 70)

(Potthoff 95)

(Benedikt, Segoufin 09)
Logic on trees

<: descendant relation, i.e. transitive closure of parent–child relation.

\[ \text{MSO}(<) = \text{regular tree languages} \]
\[ (\text{MSO}(<) = \text{MSO}) \]
\[ \text{FO}(<) \not\subseteq \text{aperiodic} \]
\[ \text{FO} = \text{locally threshold testable languages} \]
\[ = \text{aperiodic & closed under guarded swaps} \]

(Thatcher, Wright 68/Doner 70)
(Potthoff 95)
(Benedikt, Segoufin 09)
Logic on trees

$\prec$: descendant relation, i.e. transitive closure of parent–child relation.

\[ \text{MSO}(\prec) = \text{regular tree languages} \]  
\[ (\text{MSO}(\prec) = \text{MSO}) \]  
\[ \text{FO}(\prec) \subset \text{aperiodic} \]  
\[ \text{FO} = \text{locally threshold testable languages} \]  
\[ = \text{aperiodic & closed under guarded swaps} \]  

Decidable

(Thatcher, Wright 68/Doner 70)

(Potthoff 95)

(Benedikt, Segoufin 09)
Logic on trees

$\prec$: descendant relation, i.e. transitive closure of parent–child relation.

$\text{MSO}(\prec) = \text{regular tree languages}$  
($\text{Thatcher, Wright 68/Doner 70}$)

$(\text{MSO}(\prec) = \text{MSO})$

$\text{FO}(\prec) \subsetneq \text{aperiodic}$  
($\text{Potthoff 95}$)

$\text{FO} = \text{locally threshold testable languages}$
$\Rightarrow \text{aperiodic} \& \text{closed under guarded swaps}$  
($\text{Benedikt, Segoufin 09}$)

Decidable

- Further decidable characterisations known, e.g. for $\mathbb{B}\Sigma_1(\prec, \prec_{dfs})$ [Bojańczyk, Segoufin, Straubing ICALP 08] and $\Delta_2(\prec)$ [Bojańczyk, Segoufin LICS 08].
Logic on trees

\(<\): descendant relation, i.e. transitive closure of parent–child relation.

\(\text{MSO}(\text{<}) = \) regular tree languages \hspace{1cm} \text{(Thatcher, Wright 68/Doner 70)}

\((\text{MSO}(\text{<}) = \text{MSO})\)

\(\text{FO}(\text{<}) \subsetneq \) aperiodic \hspace{1cm} \text{(Potthoff 95)}

\(\text{FO} = \) locally threshold testable languages

\(=\) aperiodic & closed under guarded swaps \hspace{1cm} \text{(Benedikt, Segoufin 09)}

\textbf{Decidable}

- Further decidable characterisations known, e.g. for \(\mathcal{B}\Sigma_1(\text{<, <}_\text{dfs})\) \[\text{Bojańczyk, Segoufin, Straubing ICALP 08}\] and \(\Delta_2(\text{<})\) \[\text{Bojańczyk, Segoufin LICS 08}\].

- **Longstanding open problem:** Find a decidable characterisation of \(\text{FO}(\text{<})\)!
First main result

$\text{FO}_{\text{card}}$ : $\text{FO}$ with predicates expressing the cardinality of a structure modulo arbitrary integers.

We extend the following result from words to trees:

- (Schweikardt, Segoufin LICS 10)
  A regular (word) language is $\text{FO}_{\text{card}}$-definable iff it is closed under guarded swaps and closed under transfer.

and the following result from $\text{FO}$ to $\text{FO}_{\text{card}}$:

- (Benedikt, Segoufin ToCL 09)
  A regular tree language is $\text{FO}$-definable iff it is closed under guarded swaps and aperiodic.

We prove:

**Theorem:** (H., Schweikardt STACS 12)

A regular tree language is $\text{FO}_{\text{card}}$-definable iff it is closed under guarded swaps and closed under transfer.
First main result

$\text{FO}_{\text{card}} : \text{FO}$ with predicates expressing the cardinality of a structure modulo arbitrary integers.

We extend the following result from words to trees:

- (Schweikardt, Segoufin LICS 10)
  A regular (word) language is $\text{FO}_{\text{card}}$-definable iff it is closed under guarded swaps and closed under transfer.

and the following result from FO to $\text{FO}_{\text{card}}$:

- (Benedikt, Segoufin ToCL 09)
  A regular tree language is FO-definable iff it is closed under guarded swaps and aperiodic.

We prove:

**Theorem:** (H., Schweikardt STACS 12)
A regular tree language is $\text{FO}_{\text{card}}$-definable iff it is closed under guarded swaps and closed under transfer.
First main result

$\text{FO}_{\text{card}}$ : $\text{FO}$ with predicates expressing the cardinality of a structure modulo arbitrary integers.

We extend the following result from words to trees:

- (Schweikardt, Segoufin LICS 10)
  A regular (word) language is $\text{FO}_{\text{card}}$-definable iff it is closed under guarded swaps and closed under transfer.

and the following result from FO to $\text{FO}_{\text{card}}$:

- (Benedikt, Segoufin ToCL 09)
  A regular tree language is FO-definable iff it is closed under guarded swaps and aperiodic.

We prove:

**Theorem:**

(H., Schweikardt STACS 12)

A regular tree language is $\text{FO}_{\text{card}}$-definable iff it is closed under guarded swaps and closed under transfer.
First main result

$\text{FO}_{\text{card}} :$ $\text{FO}$ with predicates expressing the cardinality of a structure modulo arbitrary integers.

We extend the following result from words to trees:

- (Schweikardt, Segoufin LICS 10)
  A regular (word) language is $\text{FO}_{\text{card}}$-definable iff it is closed under guarded swaps and closed under transfer.

and the following result from $\text{FO}$ to $\text{FO}_{\text{card}}$:

- (Benedikt, Segoufin ToCL 09)
  A regular tree language is $\text{FO}$-definable iff it is closed under guarded swaps and aperiodic.

We prove:

**Theorem:** (H., Schweikardt STACS 12)
A regular tree language is $\text{FO}_{\text{card}}$-definable iff it is closed under guarded swaps and closed under transfer.
First main result

\( \text{FO}_{\text{card}} : \) \( \text{FO} \) with predicates expressing the cardinality of a structure modulo arbitrary integers.

We extend the following result from words to trees:

- (Schweikardt, Segoufin LICS 10)
  A regular (word) language is \( \text{FO}_{\text{card}} \)-definable iff it is closed under guarded swaps and closed under transfer.

and the following result from FO to \( \text{FO}_{\text{card}} \):

- (Benedikt, Segoufin ToCL 09)
  A regular tree language is \( \text{FO} \)-definable iff it is closed under guarded swaps and aperiodic.

We prove:

**Theorem:** (H., Schweikardt STACS 12)
A regular tree language is \( \text{FO}_{\text{card}} \)-definable iff it is closed under guarded swaps and closed under transfer.
First main result

$\text{FO}_{\text{card}}$ : $\text{FO}$ with predicates expressing the cardinality of a structure modulo arbitrary integers.

We extend the following result from words to trees:

- (Schweikardt, Segoufin LICS 10)
  A regular (word) language is $\text{FO}_{\text{card}}$-definable iff it is closed under guarded swaps and closed under transfer.

and the following result from $\text{FO}$ to $\text{FO}_{\text{card}}$:

- (Benedikt, Segoufin ToCL 09)
  A regular tree language is $\text{FO}$-definable iff it is closed under guarded swaps and aperiodic.

We prove:

\textbf{Theorem:} \hspace{1cm} (H., Schweikardt STACS 12)

A regular tree language is $\text{FO}_{\text{card}}$-definable iff it is closed under guarded swaps and closed under transfer.
Concatenation and transition monoids for trees

- **Context**: tree with designated leaf ("hole").
Concatenation and transition monoids for trees

- **Context**: tree with designated leaf ("hole").
Concatenation and transition monoids for trees

- **Context**: tree with designated leaf ("hole").
- Contexts can be concatenated like words.
Concatenation and transition monoids for trees

- **Context**: tree with designated leaf (“hole”).
- Contexts can be concatenated like words.
Concatenation and transition monoids for trees

- **Context**: tree with designated leaf ("hole").
- Contexts can be concatenated like words.
- Given a tree automaton, a context induces a function from states to states (state at hole $\rightarrow$ state at root).
Concatenation and transition monoids for trees

- **Context**: tree with designated leaf ("hole").
- Contexts can be concatenated like words.
- Given a tree automaton, a context induces a function from states to states (state at hole $\mapsto$ state at root).
- Generalises the notion of a transition monoid to tree automata.
Closure under guarded swaps

(Benedikt, Segoufin ToCL 09)

A tree language $L$ is closed under guarded swaps, if $L$ is closed under $k$-guarded vertical swaps and $k$-guarded horizontal swaps, for some $k \in \mathbb{N}$. 
$k$-guarded vertical swaps
$k$-guarded vertical swaps

Nodes $u, v$ are $k$-similar: $u, v$ have isomorphic subtrees up to depth $k$.
*k*-guarded vertical swaps

Nodes $u, v$ are $k$-similar: $u, v$ have isomorphic subtrees up to depth $k$
nodes $u, v$ are $k$-similar: $\iff$
$u, v$ have isomorphic subtrees up to depth $k$
$k$-guarded vertical swaps

$L$ is closed under $k$-guarded vertical swaps, if

\[ \in L \iff \in L \]
$k$-guarded horizontal swaps
$k$-guarded horizontal swaps

$L$ is closed under $k$-guarded horizontal swaps, if
First main result

$\text{FO}_{\text{card}}$ : $FO$ with predicates expressing the cardinality of a structure modulo arbitrary integers.

We extend the following result from words to trees:

- (Schweikardt, Segoufin LICS 10)
  A regular (word) language is $\text{FO}_{\text{card}}$-definable iff it is closed under guarded swaps and closed under transfer.

and the following result from $FO$ to $\text{FO}_{\text{card}}$:

- (Benedikt, Segoufin ToCL 09)
  A regular tree language is $FO$-definable iff it is closed under guarded swaps and aperiodic.

We prove:

**Theorem:**

(H., Schweikardt STACS 12)

A regular tree language is $\text{FO}_{\text{card}}$-definable iff it is closed under guarded swaps and closed under transfer.
Aperiodicity

$L$ is aperiodic, if there exists a number $\ell$ such that:
Aperiodicity

$L$ is aperiodic, if there exists a number $\ell$ such that:

$$\ell\text{-times} \in L \iff \ell+1\text{-times} \in L$$
Closure under transfer — vertical transfer

$L$ is closed under vertical transfer, if there exists a number $\ell$ such that:

\[
\ell \text{-times } \triangle \rightarrow \ell + 1 \text{-times } \triangle
\]
Closure under transfer — vertical transfer

$L$ is closed under **vertical transfer**, if there exists a number $\ell$ such that:

\[
|\triangle| = |\triangle| \quad \ell\text{-times} \quad \in L \quad \iff \quad \ell+1\text{-times} \quad \in L
\]
Closure under transfer — horizontal transfer

$L$ is closed under **horizontal transfer**, if there exists a number $\ell$ such that:

$$\ell \text{-times } \triangle = \ell+1 \text{-times } \triangle \quad \subseteq \quad \ell \text{-times } \triangle \quad \ell+1 \text{-times } \triangle \quad \in L \quad \iff \quad \ell \text{-times } \triangle \quad \ell+1 \text{-times } \triangle \quad \in L$$
Proof of the result

**Theorem:** (H., Schweikardt STACS 12)

A regular tree language is $\text{FO}_{\text{card}}$-definable iff it is closed under guarded swaps and closed under transfer.

**Proof methods:**

⇒ Easy direction

⇐ Interesting direction

- A concept (“tree templates“) that allows to unify vertical and horizontal transfer.

- Techniques & results of [Benedikt, Segoufin ToCL 09] (take care of size of trees, lift embedding lemmas to multiple contexts etc.).

- Lift techniques of [Schweikardt, Segoufin LICS 10] using tree templates.
Proof of the result

**Theorem:**  
(H., Schweikardt STACS 12)  
A regular tree language is \( \text{FO}_{\text{card}} \)-definable iff it is closed under guarded **swaps** and closed under **transfer**.

**Proof methods:**

“⇒“  
*Easy direction*

“⇐“  
*Interesting direction*

- A concept ("tree templates") that allows to unify vertical and horizontal transfer.
- Techniques & results of [Benedikt, Segoufin ToCL 09]  
  (take care of size of trees, lift embedding lemmas to multiple contexts etc.).
- Lift techniques of [Schweikardt, Segoufin LICS 10] using tree templates.
Theorem: \( (H., \text{Schweikardt STACS 12}) \)

A regular tree language is \( \text{FO}_{\text{card}} \)-definable iff it is closed under guarded \textit{swaps} and closed under \textit{transfer}.

Proof methods:

\( \implies \) Easy direction

\( \iff \) Interesting direction

- A concept ("tree templates") that allows to unify vertical and horizontal transfer.
- Techniques & results of [Benedikt, Segoufin ToCL 09]
  (take care of size of trees, lift embedding lemmas to multiple contexts etc.).
- Lift techniques of [Schweikardt, Segoufin LICS 10] using tree templates.
Proof of the result

Theorem: (H., Schweikardt STACS 12)
A regular tree language is $\text{FO}_{\text{card}}$-definable iff it is closed under guarded swaps and closed under transfer.

Proof methods:

$\Rightarrow$ Easy direction

$\Leftarrow$ Interesting direction

- A concept ("tree templates") that allows to unify vertical and horizontal transfer.
- Techniques & results of [Benedikt, Segoufin ToCL 09] (take care of size of trees, lift embedding lemmas to multiple contexts etc.).
- Lift techniques of [Schweikardt, Segoufin LICS 10] using tree templates.
Proof of the result

**Theorem:** (H., Schweikardt STACS 12)

A regular tree language is $\text{FO}_{\text{card}}$-definable iff it is closed under guarded swaps and closed under transfer.

**Proof methods:**

“⇒” Easy direction

“⇐” Interesting direction

- A concept ("tree templates") that allows to unify vertical and horizontal transfer.
- Techniques & results of [Benedikt, Segoufin ToCL 09] (take care of size of trees, lift embedding lemmas to multiple contexts etc.).
- Lift techniques of [Schweikardt, Segoufin LICS 10] using tree templates.
Proof of the result

**Theorem:** (H., Schweikardt STACS 12)

A regular tree language is $\text{FO}_{\text{card}}$-definable iff it is closed under guarded swaps and closed under transfer.

**Proof methods:**

“⇒” Easy direction

“⇐” Interesting direction

- A concept (“tree templates“) that allows to unify vertical and horizontal transfer.
- Techniques & results of [Benedikt, Segoufin ToCL 09] (take care of size of trees, lift embedding lemmas to multiple contexts etc.).
- Lift techniques of [Schweikardt, Segoufin LICS 10] using tree templates.
Decidability of closure under transfer

**Lemma:**

There exists an algorithm that, given a (bottom–up) tree automaton $A$ as input, decides whether $L(A)$ is closed under transfer.

Putting this together with our first main result and the decidability of closure under guarded swaps proved in [Benedikt, Segoufin ToCL 2009] this yields:

**Corollary:**

There exists an algorithm that, given a (bottom–up) tree automaton $A$ as input, decides whether $L(A)$ is $\text{FO}_{\text{card}}$–definable.
Decidability of closure under transfer

Lemma:
There exists an algorithm that, given a (bottom–up) tree automaton $A$ as input, decides whether $L(A)$ is closed under transfer.

Putting this together with our first main result and the decidability of closure under guarded swaps proved in [Benedikt, Segoufin ToCL 2009] this yields:

Corollary:
There exists an algorithm that, given a (bottom–up) tree automaton $A$ as input, decides whether $L(A)$ is $\text{FO}_{\text{card}}$–definable.
Deciding closure under transfer

- The algorithm simply enumerates all transition functions realized by contexts which satisfy the preconditions for applying transfer.
Deciding closure under transfer

- The algorithm simply enumerates all transition functions realized by contexts which satisfy the preconditions for applying transfer.
Deciding closure under transfer

- The algorithm simply enumerates all transition functions realized by contexts which satisfy the preconditions for applying transfer.

\[
\begin{align*}
\triangle = \triangle
\end{align*}
\]

\[
\begin{align*}
\ell\text{-times} & \quad \ell\text{-times} \\
\ell+1\text{-times} & \quad \ell+1\text{-times}
\end{align*}
\]
Deciding closure under transfer

- The algorithm simply enumerates all transition functions realized by contexts which satisfy the preconditions for applying transfer.

\[ \triangle = \triangle \]

- Because of the condition \( |\triangle| = |\triangle| \), it is not clear a priori how many different contexts \( \triangle, \triangledown \) need to be considered.
Deciding closure under transfer — key idea

**Key idea: “Parallel Pumping“ (for tree automaton $\mathcal{A}$)**

There exists a computable bound $n$ such that, for all contexts $\blacktriangle$, $\bigtriangleup$ with $|\blacktriangle| = |\bigtriangleup| > n$, there exist contexts $\blacktriangle$, $\bigtriangleup$ satisfying:

- $|\blacktriangle| = |\bigtriangleup| \leq n,$
- $\blacktriangle$ induces the same transition function as $\blacktriangle$ on the state set $Q$ of $\mathcal{A}$,
- $\bigtriangleup$ induces the same transition function as $\bigtriangleup$ on $Q$. 
Second main result

We extend the following result from words to trees:

- (Schweikardt, Segoufin LICS 2010)
  A regular (word) language is $\text{FO}_{\text{card}}$–definable
  iff it is $+-+-\text{invariant-FO}$–definable.

That is, we prove:

**Theorem:** (H., Schweikardt STACS 2012)

A regular tree language is $\text{FO}_{\text{card}}$–definable
iff it is closed under guarded swaps and closed under transfer
iff it is $+-+-\text{invariant-FO}$–definable
Second main result

We extend the following result from words to trees:

- (Schweikardt, Segoufin LICS 2010)
  A regular (word) language is $\text{FO}_{\text{card}}$–definable
  iff it is $+\text{-invariant-FO}$–definable.

That is, we prove:

**Theorem:** (H., Schweikardt STACS 2012)

A regular tree language is $\text{FO}_{\text{card}}$–definable
iff it is closed under guarded swaps and closed under transfer
iff it is $+\text{-invariant-FO}$–definable
Addition–invariant FO

Definition: A FO(≺)-sentence \( \varphi \) is \( \prec \)-invariant on a tree \( t \) if for all linear orders \( \prec_1 \) and \( \prec_2 \) on the set of nodes of \( t \)
Definition: A FO($\preceq$)-sentence $\varphi$ is $\preceq$-invariant on a tree $t$ if for all linear orders $\preceq_1$ and $\preceq_2$ on the set of nodes of $t$

$$ (t, \preceq_1) \models \varphi \iff (t, \preceq_2) \models \varphi. $$
Addition–invariant FO

**Definition:** A FO(≺)-sentence \( \varphi \) is \( \prec \)-invariant on a tree \( t \) if for all linear orders \( \prec_1 \) and \( \prec_2 \) on the set of nodes of \( t \):

\[
(t, \prec_1) \models \varphi \iff (t, \prec_2) \models \varphi.
\]

\( \prec \)-invariant-FO: FO(≺)–sentences which are \( \prec \)-invariant on all trees.
**Addition–invariant FO**

**Definition:** A FO(≺, +)-sentence $\varphi$ is $\langle+\rangle$-invariant on a tree $t$ $\iff$ for all linear orders $≺_1$ and $≺_2$ on the set of nodes of $t$ and for the matching addition relations $+_1$, $+_2$

\[(t, ≺_1, +_1) \models \varphi \iff (t, ≺_2, +_2) \models \varphi.\]

$≺$-invariant-FO: FO($≺$)–sentences which are $≺$-invariant on all trees.
Addition–invariant FO

**Definition:** A FO($≺, +$)-sentence $\varphi$ is $+$-invariant on a tree $t$ if for all linear orders $≺_1$ and $≺_2$ on the set of nodes of $t$ and for the matching addition relations $+_1$, $+_2$

$$(t, ≺_1, +_1) \models \varphi \iff (t, ≺_2, +_2) \models \varphi.$$

$+$-invariant-FO: FO($≺, +$)-sentences which are $+$-invariant on all trees.
Expressiveness of invariant FO

- On finite labeled graphs, $\prec$-invariant-FO is more expressive than FO (Gurevich)
Expressiveness of invariant FO

- On finite labeled graphs, $\preceq$-invariant-FO is more expressive than FO (Gurevich)
- On trees $\preceq$-invariant-FO $= FO$ ................. (Benedikt, Segoufin JSL 09)

What about $\pm$-invariant FO?
Expressiveness of invariant FO

- On finite labeled graphs, $\prec$-invariant-FO is more expressive than FO (Gurevich)
- On trees $\prec$-invariant-FO $=$ FO ...................... (Benedikt, Segoufin JSL 09)

What about $+$-invariant FO?

**Example:**

A $+$-invariant-FO–sentence $\varphi$ such that

\[ t \models \varphi \iff |t| \text{ is even.} \]

\[ \varphi := \exists x \exists z \left( x + x = z \land \forall y \ (y \prec z \lor y = z) \right) \]
Expressiveness of invariant FO

- On finite labeled graphs, $\prec$-invariant-FO is more expressive than FO (Gurevich)
- On trees $\prec$-invariant-FO = FO ................. (Benedikt, Segoufin JSL 09)

What about $+$-invariant FO?

**Example:**

A $+$-invariant-FO–sentence $\varphi$ such that

\[
t \models \varphi \iff |t| \text{ is even.}
\]

\[
\varphi := \exists x \exists z \left( x + x = z \land \forall y \left( y \prec z \lor y = z \right) \right)
\]
Proof of second main result

**Theorem:** \( (H.,\text{ Schweikardt STACS 2012}) \)

A regular tree language is \( \text{FO}_{\text{card}} \)-definable iff it is \( +\text{-invariant-FO} \)-definable.

**Proof idea:**

\( \implies \) Each cardinality predicate is \( +\text{-invariant-FO} \)-definable, as seen in our previous example.

\( \impliedby \) Using our first main result, it suffices to prove closure under guarded swaps and transfer.

- Proving closure under transfer can be achieved by a reduction to results of [Schweikardt, Segoufin LICS 2010] by interpreting trees in words.
- Proving closure under guarded swaps is the most involved part of the proof.

The proof uses:

- locality of \( \prec \)-invariant-FO from [Grohe, Schwentick MFCS 1998],
- results allowing to lift equivalence of ordered structures to a certain extension of Presburger arithmetic,
- interpretation of trees in these structures.
Proof of second main result

**Theorem:** (H., Schweikardt STACS 2012)

A regular tree language is $\text{FO}_{\text{card}}$–definable iff it is $\text{+-invariant-FO}$–definable.

**Proof idea:**

$\Rightarrow$ Each cardinality predicate is $\text{+-invariant-FO}$–definable, as seen in our previous example.

$\Leftarrow$ Using our first main result, it suffices to prove closure under guarded swaps and transfer.

- Proving closure under transfer can be achieved by a reduction to results of [Schweikardt, Segoufin LICS 2010] by interpreting trees in words.
- Proving closure under guarded swaps is the most involved part of the proof. The proof uses:
  - locality of $\prec$-invariant-FO from [Grohe, Schwentick MFCS 1998],
  - results allowing to lift equivalence of ordered structures to a certain extension of Presburger arithmetic,
  - interpretation of trees in these structures.
Proof of second main result

**Theorem:** \( (H., \text{Schweikardt STACS 2012}) \)

A regular tree language is \( \text{FO}_{\text{card}} \)-definable iff it is \( +\)-invariant-\( \text{FO} \)-definable.

**Proof idea:**

\( \Rightarrow \) Each cardinality predicate is \( +\)-invariant-\( \text{FO} \)-definable, as seen in our previous example.

\( \Leftarrow \) Using our first main result, it suffices to prove closure under guarded swaps and transfer.

- Proving closure under transfer can be achieved by a reduction to results of [Schweikardt, Segoufin LICS 2010] by interpreting trees in words.
- Proving closure under guarded swaps is the most involved part of the proof. The proof uses:
  - locality of \( \prec \)-invariant-\( \text{FO} \) from [Grohe, Schwentick MFCS 1998],
  - results allowing to lift equivalence of ordered structures to a certain extension of Presburger arithmetic,
  - interpretation of trees in these structures.
Proof of second main result

**Theorem:** (H., Schweikardt STACS 2012)

A regular tree language is $\text{FO}_{\text{card}}$--definable iff it is $\text{+-invariant-FO}$--definable.

**Proof idea:**

“⇒” Each cardinality predicate is $\text{+-invariant-FO}$--definable, as seen in our previous example.

“⇐” Using our first main result, it suffices to prove closure under guarded swaps and transfer.

- Proving closure under transfer can be achieved by a reduction to results of [Schweikardt, Segoufin LICS 2010] by interpreting trees in words.

- Proving closure under guarded swaps is the most involved part of the proof. The proof uses:
  - locality of $\prec$-invariant-FO from [Grohe, Schwentick MFCS 1998],
  - results allowing to lift equivalence of ordered structures to a certain extension of Presburger arithmetic,
  - interpretation of trees in these structures.
Proof of second main result

**Theorem:** (H., Schweikardt STACS 2012)

A regular tree language is \( \text{FO}_{\text{card}} \)-definable iff it is \(+\)-invariant-\( \text{FO} \)-definable.

**Proof idea:**

"⇒" Each cardinality predicate is \(+\)-invariant-\( \text{FO} \)-definable, as seen in our previous example.

"⇐" Using our first main result, it suffices to prove closure under guarded swaps and transfer.

- Proving closure under transfer can be achieved by a reduction to results of [Schweikardt, Segoufin LICS 2010] by interpreting trees in words.
- Proving closure under guarded swaps is the most involved part of the proof. The proof uses:
  - locality of \( \prec \)-invariant-\( \text{FO} \) from [Grohe, Schwentick MFCS 1998],
  - results allowing to lift equivalence of ordered structures to a certain extension of Presburger arithmetic,
  - interpretation of trees in these structures.
Proof of second main result

**Theorem:** (H., Schweikardt STACS 2012)
A regular tree language is \( \text{FO}_{\text{card}} \)-definable iff it is \(+\text{-invariant-FO}\)-definable.

**Proof idea:**

\[ \Rightarrow \] Each cardinality predicate is \(+\text{-invariant-FO}\)-definable, as seen in our previous example.

\[ \Leftarrow \] Using our first main result, it suffices to prove closure under guarded swaps and transfer.

- Proving closure under transfer can be achieved by a reduction to results of [Schweikardt, Segoufin LICS 2010] by interpreting trees in words.
- Proving closure under guarded swaps is the most involved part of the proof.

The proof uses:

- locality of \( \prec\)-invariant-FO from [Grohe, Schwentick MFCS 1998],
- results allowing to lift equivalence of ordered structures to a certain extension of Presburger arithmetic,
- interpretation of trees in these structures.
Proof of second main result

Theorem: (H., Schweikardt STACS 2012)
A regular tree language is $\text{FO}_{\text{card}}$–definable iff it is $+\text{-invariant}$-$\text{FO}$–definable.

Proof idea:

$\Rightarrow$ Each cardinality predicate is $+\text{-invariant}$-$\text{FO}$–definable, as seen in our previous example.

$\Leftarrow$ Using our first main result, it suffices to prove closure under guarded swaps and transfer.

- Proving closure under transfer can be achieved by a reduction to results of [Schweikardt, Segoufin LICS 2010] by interpreting trees in words.
- Proving closure under guarded swaps is the most involved part of the proof. The proof uses:
  - locality of $\text{-invariant-FO}$ from [Grohe, Schwentick MFCS 1998],
  - results allowing to lift equivalence of ordered structures to a certain extension of Presburger arithmetic,
  - interpretation of trees in these structures.
**Proof of second main result**

**Theorem:** \((H.,\) Schweikardt \ STACS 2012\)

A regular tree language is \(\text{FO}_{\text{card}}\)-definable iff it is \(+\text{-invariant-FO}\)-definable.

**Proof idea:**

\(\Rightarrow\) Each cardinality predicate is \(+\text{-invariant-FO}\)-definable, as seen in our previous example.

\(\Leftarrow\) Using our first main result, it suffices to prove closure under guarded swaps and transfer.

- Proving closure under transfer can be achieved by a reduction to results of [Schweikardt, Segoufin LICS 2010] by interpreting trees in words.
- Proving closure under guarded swaps is the most involved part of the proof. The proof uses:
  - locality of \(\prec\)-invariant-FO from [Grohe, Schwentick MFCS 1998],
  - results allowing to lift equivalence of ordered structures to a certain extension of Presburger arithmetic,
  - interpretation of trees in these structures.
Open questions

- What is the complexity of deciding closure under transfer?
Open questions

- What is the complexity of deciding closure under transfer?
- Are all $+$-invariant tree languages regular?
Open questions

- What is the complexity of deciding closure under transfer?
- Are all \(+\)-invariant tree languages regular?

**Conjecture:**

A regular tree language is \(\text{FO}_{\text{card}}\)-definable iff it is \(+\)-invariant-\(\text{FO}\)-definable.