The Complexity of Verifying Ground Tree Rewrite Systems

Stefan Göller
(University of Bremen)

joint work with Anthony Widjaja Lin
(Oxford University)

LICS 2011, Toronto

Model checking and equivalence checking

Model checking a class of structures $\mathcal{C}$ against a logic $\mathcal{L}$

INPUT: Structure $S \in \mathcal{C}$ + formula $\varphi \in \mathcal{L}$
QUESTION: $S \models \varphi$?

Model checking and equivalence checking

Model checking a class of structures $\mathcal{C}$ against a logic $\mathcal{L}$

INPUT: Structure $S \in \mathcal{C}$ + formula $\varphi \in \mathcal{L}$
QUESTION: $S \models \varphi$?

$\equiv$-Equivalence checking between structures in $\mathcal{C}$

INPUT: Structures $S_1, S_2 \in \mathcal{C}$
QUESTION: $S_1 \equiv S_2$?

Pushdown and prefix-recognizable systems

States: All finite words (over some finite alphabet)

Pushdown and prefix-recognizable systems

States: All finite words (over some finite alphabet)

- **Pushdown systems:**
  - Rewrite rules: $u \overset{a}{\hookrightarrow} v$ (u, v words)
  - Transitions: $uw \overset{a}{\rightarrow} vw$ for all words w.

Pushdown and prefix-recognizable systems

States: All finite words (over some finite alphabet)

- **Pushdown systems:**
  - Rewrite rules: $u \overset{a}{\hookrightarrow} v$ (u, v words)
  - Transitions: $uw \overset{a}{\to} vw$ for all words w.
- **Prefix-recognizable systems:**
  - Rewriting rules: $L_1 \overset{a}{\hookrightarrow} L_2$ (L_1, L_2 regular word languages)
  - Transitions: $uw \overset{a}{\to} vw$ for all $u \in L_1, v \in L_2$ and all words w.

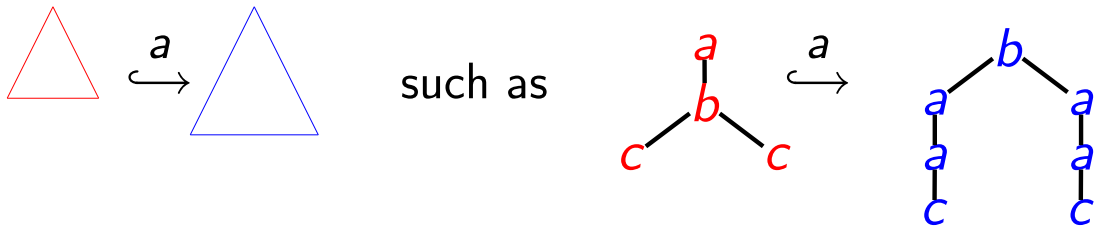
(Regular) ground tree rewrite systems

States: All finite ranked trees (over some ranked alphabet)

(Regular) ground tree rewrite systems

States: All finite ranked trees (over some ranked alphabet)

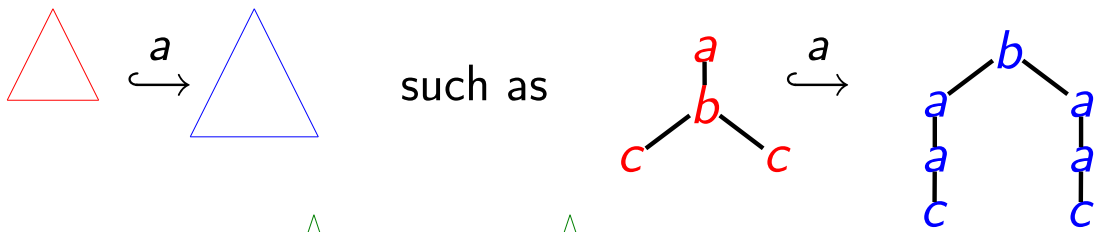
- **Ground tree rewrite systems (GTRS):**
  - Rewrite rules



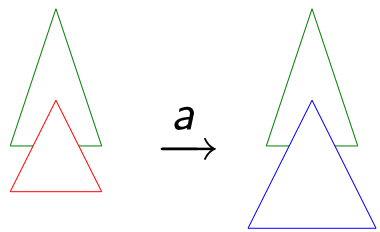
(Regular) ground tree rewrite systems

States: All finite ranked trees (over some ranked alphabet)

- **Ground tree rewrite systems (GTRS):**
  - Rewrite rules



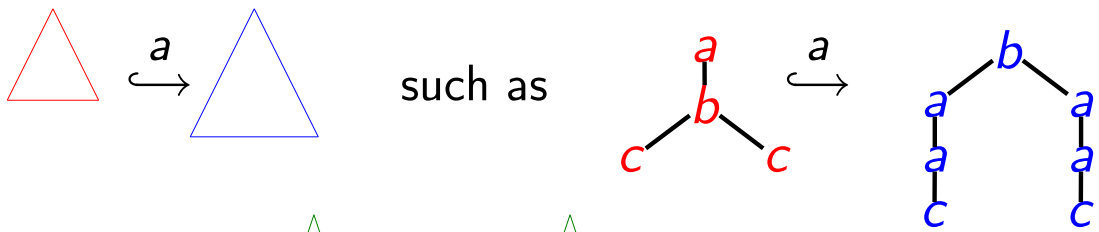
  - Transitions:



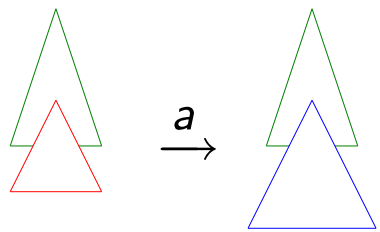
(Regular) ground tree rewrite systems

States: All finite ranked trees (over some ranked alphabet)

- **Ground tree rewrite systems (GTRS):**
  - Rewrite rules



  - Transitions:

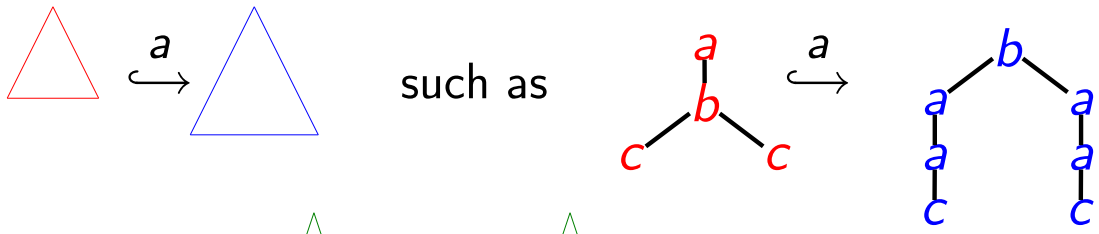


- **Regular ground tree rewrite systems (RGTRS)**
  - Rewrite rules: $L_1 \overset{a}{\hookrightarrow} L_2$ for regular tree languages L_1 and L_2.

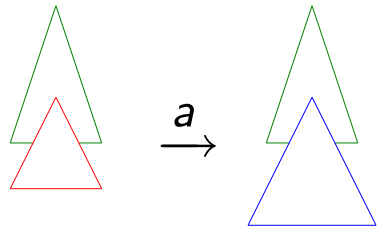
(Regular) ground tree rewrite systems

States: All finite ranked trees (over some ranked alphabet)

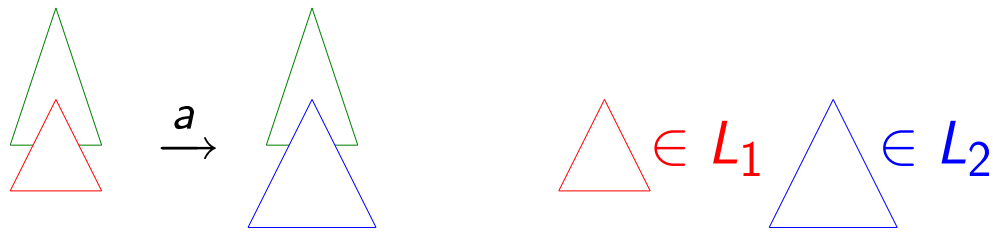
- **Ground tree rewrite systems (GTRS):**
  - Rewrite rules



  - Transitions:



- **Regular ground tree rewrite systems (RGTRS)**
  - Rewrite rules: $L_1 \overset{a}{\hookrightarrow} L_2$ for regular tree languages L_1 and L_2.
  - Transitions:



Why study pushdown systems and GTRS?

- Pushdown systems: Allow to model behavior of recursive programs.

Why study pushdown systems and GTRS?

- Pushdown systems: Allow to model behavior of recursive programs.
- **GTRS** = Pushdown systems plus unbounded parallelism.

Why study pushdown systems and GTRS?

- Pushdown systems: Allow to model behavior of recursive programs.
- **GTRS** = Pushdown systems plus unbounded parallelism.

Why study pushdown systems and GTRS?

- Pushdown systems: Allow to model behavior of recursive programs.
- **GTRS** = Pushdown systems plus unbounded parallelism.

Theorem

Bisimulation equivalence between pushdown systems is

- *decidable (Sénizergues 2005)*

Why study pushdown systems and GTRS?

- Pushdown systems: Allow to model behavior of recursive programs.
- **GTRS** = Pushdown systems plus unbounded parallelism.

Theorem

Bisimulation equivalence between pushdown systems is

- *decidable (Sénizergues 2005)*
- EXP-*hard (Kučera, Mayr 2010)*

Why study pushdown systems and GTRS?

- Pushdown systems: Allow to model behavior of recursive programs.
- **GTRS** = Pushdown systems plus unbounded parallelism.

Theorem

Bisimulation equivalence between pushdown systems is

- *decidable (Sénizergues 2005)*
- EXP-*hard (Kučera, Mayr 2010)*

Open problem

Is bisimulation equivalence on **GTRS** decidable?

Why study pushdown systems and GTRS?

- Pushdown systems: Allow to model behavior of recursive programs.
- **GTRS** = Pushdown systems plus unbounded parallelism.

Theorem

Bisimulation equivalence between pushdown systems is

- *decidable (Sénizergues 2005)*
- EXP-*hard (Kučera, Mayr 2010)*

Open problem

Is bisimulation equivalence on **GTRS** decidable?

This paper:

- How difficult is it to decide **GTRS** $\equiv F$ for *finite* systems F?

Why study pushdown systems and GTRS?

- Pushdown systems: Allow to model behavior of recursive programs.
- **GTRS** = Pushdown systems plus unbounded parallelism.

Theorem

Bisimulation equivalence between pushdown systems is

- *decidable (Sénizergues 2005)*
- EXP-*hard (Kučera, Mayr 2010)*

Open problem

Is bisimulation equivalence on **GTRS** decidable?

This paper:

- How difficult is it to decide **GTRS** $\equiv F$ for *finite* systems F?
- Main tool: Study model-checking problem of EF on **GTRS**.

Decidability and complexity: State of the art

TOWER = DTIME(Tower(ELEMENTARY)).

	Pushdown	**GTRS**	**RGTRS**
MSO	TOWER-c.	undecidable	
FO + reach	TOWER-c.		

Decidability and complexity: State of the art

TOWER = DTIME(Tower(ELEMENTARY)).

	Pushdown	**GTRS**	**RGTRS**
MSO	TOWER-c.	undecidable	
FO + reach	TOWER-c.		
CTL	EXP-c.	undecidable	
EF	PSPACE-c.	PSPACE...TOWER	EXP...TOWER

Decidability and complexity: State of the art

TOWER = DTIME(Tower(ELEMENTARY)).

	Pushdown	**GTRS**	**RGTRS**
MSO	TOWER-c.	undecidable	
FO + reach	TOWER-c.		
CTL	EXP-c.	undecidable	
EF	PSPACE-c.	PSPACE...TOWER	EXP...TOWER
$\sim$ vs. fin. syst.	PSPACE-c.	PSPACE...TOWER	EXP...TOWER
$\approx$ vs. fin. syst.	PSPACE-c.	PSPACE...TOWER	EXP...TOWER

Decidability and complexity: State of the art

TOWER = DTIME(Tower(ELEMENTARY)).

	Pushdown	**GTRS**	**RGTRS**
MSO	TOWER-c.	undecidable	
FO + reach	TOWER-c.		
CTL	EXP-c.	undecidable	
EF	PSPACE-c.	PSPACE...TOWER	EXP...TOWER
$\sim$ vs. fin. syst.	PSPACE-c.	PSPACE...TOWER	EXP...TOWER
$\approx$ vs. fin. syst.	PSPACE-c.	PSPACE...TOWER	EXP...TOWER

The branching-time logic EF

Formulas φ of the logic EF are given by the following grammar

$$\varphi \ ::= \quad \texttt{true} \quad | \quad \neg\varphi \quad | \quad \varphi \wedge \varphi \quad | \quad \mathsf{EX}_A\varphi \quad | \quad \mathsf{EF}\varphi,$$

where $A \subseteq \Sigma$ for some set of edge labels Σ.

The branching-time logic EF

Formulas φ of the logic EF are given by the following grammar

$$\varphi \ ::= \quad \texttt{true} \quad | \quad \neg\varphi \quad | \quad \varphi \wedge \varphi \quad | \quad \mathsf{EX}_A\varphi \quad | \quad \mathsf{EF}\varphi,$$

where $A \subseteq \Sigma$ for some set of edge labels Σ.

Let $T = (S, \{\rightarrow_a | \ a \in \Sigma\})$ be a transition system.

The branching-time logic EF

Formulas φ of the logic EF are given by the following grammar

$$\varphi \ ::= \quad \texttt{true} \quad | \quad \neg\varphi \quad | \quad \varphi \wedge \varphi \quad | \quad \mathsf{EX}_A\varphi \quad | \quad \mathsf{EF}\varphi,$$

where $A \subseteq \Sigma$ for some set of edge labels Σ.

Let $T = (S, \{\rightarrow_a | \ a \in \Sigma\})$ be a transition system.

For each state $s \in S$ and $\varphi \in \mathsf{EF}$ define $s \models \varphi$ inductively:

The branching-time logic EF

Formulas φ of the logic EF are given by the following grammar

$$\varphi \ ::= \quad \texttt{true} \quad | \quad \neg\varphi \quad | \quad \varphi \wedge \varphi \quad | \quad \mathsf{EX}_A\varphi \quad | \quad \mathsf{EF}\varphi,$$

where $A \subseteq \Sigma$ for some set of edge labels Σ.

Let $T = (S, \{\rightarrow_a | \ a \in \Sigma\})$ be a transition system.

For each state $s \in S$ and $\varphi \in \mathsf{EF}$ define $s \models \varphi$ inductively:

$$s \models \mathsf{EX}_A\varphi \iff \exists t \in S, a \in A : \quad s \rightarrow_a t \ \text{ and } \ t \models \varphi$$

The branching-time logic EF

Formulas φ of the logic EF are given by the following grammar

$$\varphi \ ::= \quad \texttt{true} \quad | \quad \neg\varphi \quad | \quad \varphi \wedge \varphi \quad | \quad \mathsf{EX}_A\varphi \quad | \quad \mathsf{EF}\varphi,$$

where $A \subseteq \Sigma$ for some set of edge labels Σ.

Let $T = (S, \{\rightarrow_a | \ a \in \Sigma\})$ be a transition system.

For each state $s \in S$ and $\varphi \in \mathsf{EF}$ define $s \models \varphi$ inductively:

$$s \models \mathsf{EX}_A\varphi \iff \exists t \in S, a \in A : \quad s \rightarrow_a t \ \text{ and } \ t \models \varphi$$

$$s \models \mathsf{EF}\varphi \iff \exists t \in S : \quad s \rightarrow^* t \ \text{ and } \ t \models \varphi$$

$$\text{where } \rightarrow = \bigcup_{a \in \Sigma} \rightarrow_a$$

Decidability and complexity

TOWER = DTIME(Tower($O(n)$)).

	Pushdown	**GTRS**	**RGTRS**
MSO	TOWER-c.	undecidable	
FO + reach	TOWER-c.		
CTL	EXP-c.	undecidable	
EF	PSPACE-c.	PSPACE...TOWER	EXP...TOWER
$\sim$ vs. fin. syst.	PSPACE-c.	PSPACE...TOWER	EXP...TOWER
$\approx$ vs. fin. syst.	PSPACE-c.	PSPACE...TOWER	EXP...TOWER

Decidability and complexity

TOWER $=$ DTIME(Tower($O(n)$)).

	Pushdown	**GTRS**	**RGTRS**
MSO	TOWER-c.	undecidable	
FO + reach	TOWER-c.		
CTL	EXP-c.	undecidable	
EF	PSPACE-c.	**TOWER − c.**	
$\sim$ vs. fin. syst.	PSPACE-c.	PSPACE...TOWER	EXP...TOWER
$\approx$ vs. fin. syst.	PSPACE-c.	PSPACE...TOWER	EXP...TOWER

EF model checking is nonelementary on GTRS

Theorem

Model checking EF is nonelementary on GTRS.

EF model checking is nonelementary on GTRS

Theorem

Model checking EF is nonelementary on GTRS.

Proof.

Idea: Reduction from first-order satisfiability over words.

.

EF model checking is nonelementary on GTRS

Theorem

Model checking EF is nonelementary on GTRS.

Proof.

Idea: Reduction from first-order satisfiability over words.

Fix some first order sentence $\varphi = \exists x_1 \forall x_2 \cdots \exists x_{n-1} \forall x_n \; \psi(x_1, \ldots, x_n)$ over signature $(P_0, P_1, <)$.

.

EF model checking is nonelementary on GTRS

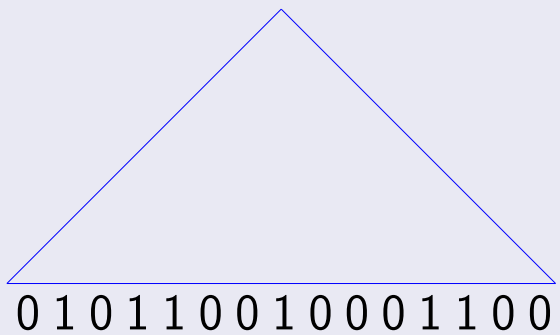
Theorem

Model checking EF is nonelementary on GTRS.

Proof.

Idea: Reduction from first-order satisfiability over words.

Fix some first order sentence $\varphi = \exists x_1 \forall x_2 \cdots \exists x_{n-1} \forall x_n \; \psi(x_1, \ldots, x_n)$ over signature $(P_0, P_1, <)$.



Yield string of tree corresponds to word

.

□

EF model checking is nonelementary on GTRS

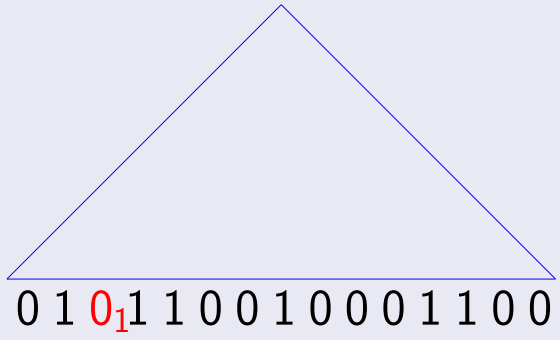
Theorem

Model checking EF is nonelementary on GTRS.

Proof.

Idea: Reduction from first-order satisfiability over words.

Fix some first order sentence $\varphi = \exists x_1 \forall x_2 \cdots \exists x_{n-1} \forall x_n \; \psi(x_1, \ldots, x_n)$ over signature $(P_0, P_1, <)$.



Yield string of tree corresponds to word

EF **formula:**

EX_{a_1}

Rewrite rules:

$0 \overset{a_1}{\hookrightarrow} 0_1$ (and $1 \overset{a_1}{\hookrightarrow} 1_1$)

.

□

EF model checking is nonelementary on GTRS

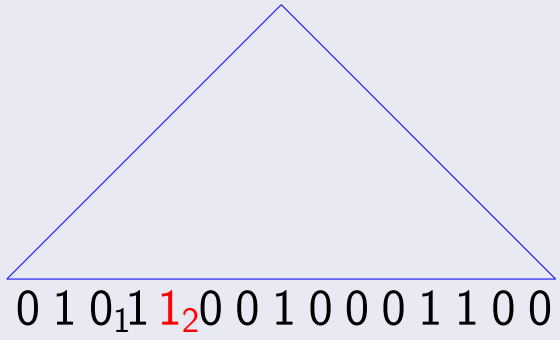
Theorem

Model checking EF is nonelementary on GTRS.

Proof.

Idea: Reduction from first-order satisfiability over words.

Fix some first order sentence $\varphi = \exists x_1 \forall x_2 \cdots \exists x_{n-1} \forall x_n \; \psi(x_1, \ldots, x_n)$ over signature $(P_0, P_1, <)$.



Yield string of tree corresponds to word

EF **formula:**

$\mathsf{EX}_{a_1}\mathsf{AX}_{a_2}$

Rewrite rules:

$1 \overset{a_2}{\hookrightarrow} 1_2$ (and $0 \overset{a_2}{\hookrightarrow} 0_2$)

.

□

EF model checking is nonelementary on GTRS

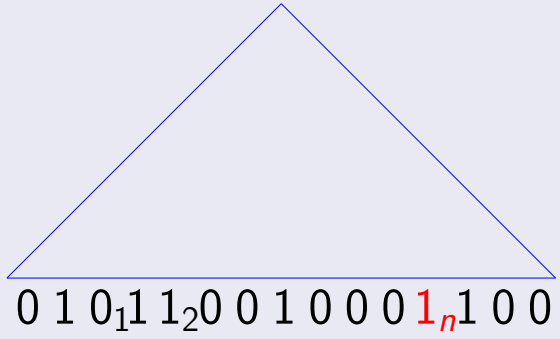
Theorem

Model checking EF is nonelementary on GTRS.

Proof.

Idea: Reduction from first-order satisfiability over words.

Fix some first order sentence $\varphi = \exists x_1 \forall x_2 \cdots \exists x_{n-1} \forall x_n \; \psi(x_1, \ldots, x_n)$ over signature $(P_0, P_1, <)$.



Yield string of tree corresponds to word

EF **formula:**

$$\mathsf{EX}_{a_1}\mathsf{AX}_{a_2} \cdots \mathsf{AX}_{a_n}$$

.

Rewrite rules:

$1 \overset{a_n}{\hookrightarrow} 1_n$ (and $0 \overset{a_n}{\hookrightarrow} 0_n$)

□

EF model checking is nonelementary on GTRS

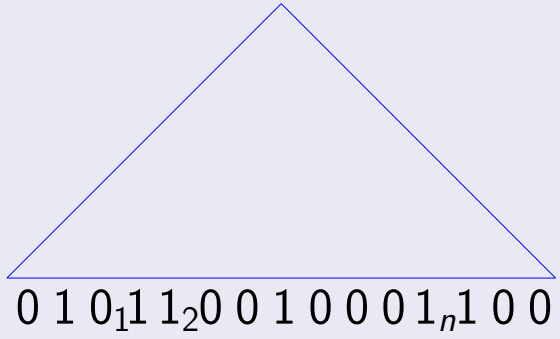
Theorem

Model checking EF is nonelementary on GTRS.

Proof.

Idea: Reduction from first-order satisfiability over words.

Fix some first order sentence $\varphi = \exists x_1 \forall x_2 \cdots \exists x_{n-1} \forall x_n \; \psi(x_1, \ldots, x_n)$ over signature $(P_0, P_1, <)$.



Yield string of tree corresponds to word

EF **formula:**

$$\mathsf{EX}_{a_1}\mathsf{AX}_{a_2} \cdots \mathsf{AX}_{a_n}\mathsf{EF}\,\mathsf{EX}_{\mathsf{acc}}$$

Rewrite rules:

transitions of tree automaton accepting $[\![\psi]\!]$

.

□

EF model checking is nonelementary on GTRS

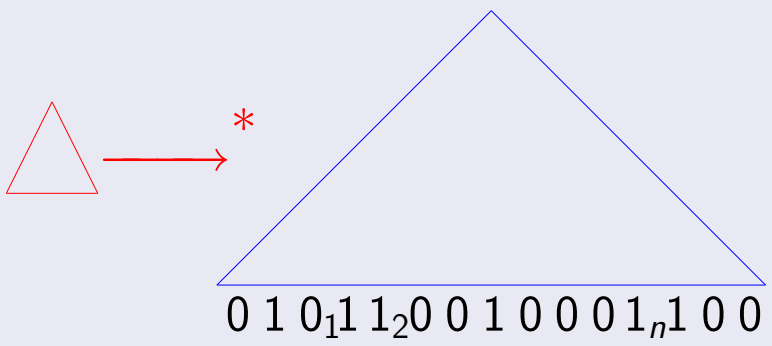
Theorem

Model checking EF is nonelementary on GTRS.

Proof.

Idea: Reduction from first-order satisfiability over words.

Fix some first order sentence $\varphi = \exists x_1 \forall x_2 \cdots \exists x_{n-1} \forall x_n \; \psi(x_1, \ldots, x_n)$ over signature $(P_0, P_1, <)$.



Yield string of tree corresponds to word

EF **formula:**

$$\mathsf{EF}\ \mathsf{EX}_{a_1}\mathsf{AX}_{a_2}\ \cdots\ \mathsf{AX}_{a_n}\mathsf{EF}\ \mathsf{EX}_{\mathsf{acc}}$$

.

□

Model checking syntactic fragments of EF on GTRS

- EF on GTRS: Nonelementary already for **two** nested EF operators ($\Rightarrow \mathsf{EF}_2$).

Model checking syntactic fragments of EF on GTRS

- EF on GTRS: Nonelementary already for **two** nested EF operators ($\Rightarrow \mathrm{EF}_2$).
- What happens with at most one nesting ($\Rightarrow \mathrm{EF}_1$)?

Model checking syntactic fragments of EF on GTRS

- EF on GTRS: Nonelementary already for **two** nested EF operators ($\Rightarrow \mathrm{EF}_2$).
- What happens with at most one nesting ($\Rightarrow \mathrm{EF}_1$)?

  Motivation:
  - Find the nonelementary border for EF.

Model checking syntactic fragments of EF on GTRS

- EF on GTRS: Nonelementary already for **two** nested EF operators ($\Rightarrow \mathrm{EF}_2$).
- What happens with at most one nesting ($\Rightarrow \mathrm{EF}_1$)?

  Motivation:
  - Find the nonelementary border for EF.
  - **Theorem:** (Jančar, Kučera, Moller)
    Strong bisimilarity against finite systems is polytime-reducible to model checking EF_1.

Decidability and complexity

TOWER = DTIME(Tower($O(n)$)).

	Pushdown	**GTRS**	**RGTRS**
MSO	TOWER-c.	undecidable	
FO + reach	TOWER-c.		
CTL	EXP-c.	undecidable	
EF	PSPACE-c.	TOWER-c.	
$\sim$ vs. fin. syst.	PSPACE-c.	PSPACE...TOWER	EXP...TOWER
$\approx$ vs. fin. syst.	PSPACE-c.	PSPACE...TOWER	EXP...TOWER

Decidability and complexity

TOWER $=$ DTIME(Tower($O(n)$)).

	Pushdown	**GTRS**	**RGTRS**
MSO	TOWER-c.	undecidable	
FO + reach	TOWER-c.		
CTL	EXP-c.	undecidable	
EF_2, EF	PSPACE-c.	TOWER-c.	
EF_1	PSPACE-c.	?	TOWER-c.
$\sim$ vs. fin. syst.	PSPACE-c.	PSPACE...TOWER	EXP...TOWER
$\approx$ vs. fin. syst.	PSPACE-c.	PSPACE...TOWER	EXP...TOWER

Decidability and complexity

TOWER = DTIME(Tower($O(n)$)).

	Pushdown	**GTRS**	**RGTRS**
MSO	TOWER-c.	undecidable	
FO + reach	TOWER-c.		
CTL	EXP-c.	undecidable	
EF_2, EF	PSPACE-c.	TOWER-c.	
EF_1	PSPACE-c.	$\mathbf{P}^{\mathbf{NEXP}}$-c.	TOWER-c.
$\sim$ vs. fin. syst.	PSPACE-c.	PSPACE...TOWER	EXP...TOWER
$\approx$ vs. fin. syst.	PSPACE-c.	PSPACE...TOWER	EXP...TOWER

Model checking EF_1 on GTRS is P^{NEXP}-complete

Upper bound: Fix some GTRS G.

Model checking EF_1 on GTRS is $\mathrm{P}^{\mathrm{NEXP}}$-complete

Upper bound: Fix some GTRS G.

How to represent tree lang. $[\![\varphi]\!]$ for each **modal formula** $\varphi \in \mathrm{EF}_0$?

Model checking EF_1 on GTRS is $\mathrm{P}^{\mathrm{NEXP}}$-complete

Upper bound: Fix some GTRS G.

How to represent tree lang. $[\![\varphi]\!]$ for each **modal formula** $\varphi \in \mathrm{EF}_0$?

- First approach: Compute $[\![\varphi]\!]$ using closure of tree languages by boolean operations and pre (for dealing with EX).

Model checking EF_1 on GTRS is P^{NEXP}-complete

Upper bound: Fix some GTRS G.

How to represent tree lang. $[\![\varphi]\!]$ for each **modal formula** $\varphi \in EF_0$?

- First approach: Compute $[\![\varphi]\!]$ using closure of tree languages by boolean operations and pre (for dealing with EX).

  (Nonelementary blowup).

Model checking EF_1 on GTRS is P^{NEXP}-complete

Upper bound: Fix some GTRS G.

How to represent tree lang. $[\![\varphi]\!]$ for each **modal formula** $\varphi \in EF_0$?

- First approach: Compute $[\![\varphi]\!]$ using closure of tree languages by boolean operations and pre (for dealing with EX).

  (Nonelementary blowup).

- Second approach: Study relation $\simeq_i$ on trees, where $T_1 \simeq_i T_2$ iff T_1 and T_2 cannot be distinguished by modal formulas of EX-rank at most i.

Model checking EF_1 on GTRS is P^{NEXP}-complete

Upper bound: Fix some GTRS G.

How to represent tree lang. $[\![\varphi]\!]$ for each **modal formula** $\varphi \in EF_0$?

- First approach: Compute $[\![\varphi]\!]$ using closure of tree languages by boolean operations and pre (for dealing with EX).

  (Nonelementary blowup).

- Second approach: Study relation $\simeq_i$ on trees, where $T_1 \simeq_i T_2$ iff T_1 and T_2 cannot be distinguished by modal formulas of EX-rank at most i.

  How compute NTA for each "positive" equiv. class w.r.t. φ?

Model checking EF_1 on GTRS is P^{NEXP}-complete

Upper bound: Fix some GTRS G.

How to represent tree lang. $[\![\varphi]\!]$ for each **modal formula** $\varphi \in EF_0$?

- First approach: Compute $[\![\varphi]\!]$ using closure of tree languages by boolean operations and pre (for dealing with EX).

  (Nonelementary blowup).

- Second approach: Study relation $\simeq_i$ on trees, where $T_1 \simeq_i T_2$ iff T_1 and T_2 cannot be distinguished by modal formulas of EX-rank at most i.

  How compute NTA for each "positive" equiv. class w.r.t. φ?

  How can one bound the index of $\simeq_i$?

Model checking EF_1 on GTRS is P^{NEXP}-complete

Our solution: Define relation $\equiv_i$ where $T_1 \equiv_i T_2$ iff T_1 and T_2 have the same number of subtrees of depth $\leq i \cdot \text{poly}(|G|)$ up to some threshold θ.

Model checking EF_1 on GTRS is P^{NEXP}-complete

Our solution: Define relation $\equiv_i$ where $T_1 \equiv_i T_2$ iff T_1 and T_2 have the same number of subtrees of depth $\leq i \cdot \text{poly}(|G|)$ up to some threshold θ.

- $\equiv_i$ is finer than $\simeq_i$.

Model checking EF_1 on GTRS is $\mathrm{P}^{\mathrm{NEXP}}$-complete

Our solution: Define relation $\equiv_i$ where $T_1 \equiv_i T_2$ iff T_1 and T_2 have the same number of subtrees of depth $\leq i \cdot \mathrm{poly}(|G|)$ up to some threshold θ.

- $\equiv_i$ is finer than $\simeq_i$.
- Testing if $f : \mathrm{Trees}^{\leq i \cdot \mathrm{poly}(|G|)} \to \{0, \ldots, \theta\}$ describes positive equivalence class w.r.t. φ is decidable in time $|f|^{\mathrm{poly}(i+|G|)}$.

Model checking EF_1 on GTRS is $\mathrm{P}^{\mathrm{NEXP}}$-complete

Our solution: Define relation $\equiv_i$ where $T_1 \equiv_i T_2$ iff T_1 and T_2 have the same number of subtrees of depth $\leq i \cdot \mathrm{poly}(|G|)$ up to some threshold θ.

- $\equiv_i$ is finer than $\simeq_i$.
- Testing if $f : \mathrm{Trees}^{\leq i \cdot \mathrm{poly}(|G|)} \to \{0, \ldots, \theta\}$ describes positive equivalence class w.r.t. φ is decidable in time $|f|^{\mathrm{poly}(i+|G|)}$.
- $\forall$ positive f $\exists$ small NTA (computable) accepting $[\![f]\!]$.

Model checking EF_1 on GTRS is $\mathrm{P}^{\mathrm{NEXP}}$-complete

Our solution: Define relation $\equiv_i$ where $T_1 \equiv_i T_2$ iff T_1 and T_2 have the same number of subtrees of depth $\leq i \cdot \mathrm{poly}(|G|)$ up to some threshold θ.

- $\equiv_i$ is finer than $\simeq_i$.
- Testing if $f : \mathrm{Trees}^{\leq i \cdot \mathrm{poly}(|G|)} \to \{0, \ldots, \theta\}$ describes positive equivalence class w.r.t. φ is decidable in time $|f|^{\mathrm{poly}(i+|G|)}$.
- $\forall$ positive f $\exists$ small NTA (computable) accepting $[\![f]\!]$.

Model checking $T_0 \stackrel{?}{\models} \mathrm{EF}\varphi$ (φ modal formula) in NEXP:

Model checking EF_1 on GTRS is P^{NEXP}-complete

Our solution: Define relation $\equiv_i$ where $T_1 \equiv_i T_2$ iff T_1 and T_2 have the same number of subtrees of depth $\leq i \cdot \text{poly}(|G|)$ up to some threshold θ.

- $\equiv_i$ is finer than $\simeq_i$.
- Testing if $f : \text{Trees}^{\leq i \cdot \text{poly}(|G|)} \to \{0, \ldots, \theta\}$ describes positive equivalence class w.r.t. φ is decidable in time $|f|^{\text{poly}(i+|G|)}$.
- $\forall$ positive f $\exists$ small NTA (computable) accepting $[\![f]\!]$.

Model checking $T_0 \stackrel{?}{\models} \text{EF}\varphi$ (φ modal formula) in NEXP:

1. Guess a function $f : \text{Trees}^{\leq i \cdot \text{poly}(|G|)} \to \{0, \ldots, \theta\}$.

Model checking EF_1 on GTRS is $\mathrm{P}^{\mathrm{NEXP}}$-complete

Our solution: Define relation $\equiv_i$ where $T_1 \equiv_i T_2$ iff T_1 and T_2 have the same number of subtrees of depth $\leq i \cdot \mathrm{poly}(|G|)$ up to some threshold θ.

- $\equiv_i$ is finer than $\simeq_i$.
- Testing if $f : \mathrm{Trees}^{\leq i \cdot \mathrm{poly}(|G|)} \to \{0, \ldots, \theta\}$ describes positive equivalence class w.r.t. φ is decidable in time $|f|^{\mathrm{poly}(i+|G|)}$.
- $\forall$ positive f $\exists$ small NTA (computable) accepting $[\![f]\!]$.

Model checking $T_0 \stackrel{?}{\models} \mathrm{EF}\varphi$ (φ modal formula) in NEXP:

1. Guess a function $f : \mathrm{Trees}^{\leq i \cdot \mathrm{poly}(|G|)} \to \{0, \ldots, \theta\}$.
2. Check whether f describes positive equivalence class w.r.t. φ.

Model checking EF_1 on GTRS is P^{NEXP}-complete

Our solution: Define relation $\equiv_i$ where $T_1 \equiv_i T_2$ iff T_1 and T_2 have the same number of subtrees of depth $\leq i \cdot \text{poly}(|G|)$ up to some threshold θ.

- $\equiv_i$ is finer than $\simeq_i$.
- Testing if $f : \text{Trees}^{\leq i \cdot \text{poly}(|G|)} \to \{0, \ldots, \theta\}$ describes positive equivalence class w.r.t. φ is decidable in time $|f|^{\text{poly}(i+|G|)}$.
- $\forall$ positive f $\exists$ small NTA (computable) accepting $[\![f]\!]$.

Model checking $T_0 \stackrel{?}{\models} \text{EF}\varphi$ (φ modal formula) in NEXP:

1. Guess a function $f : \text{Trees}^{\leq i \cdot \text{poly}(|G|)} \to \{0, \ldots, \theta\}$.
2. Check whether f describes positive equivalence class w.r.t. φ.
3. Compute small NTA accepting $[\![f]\!]$.

Model checking EF_1 on GTRS is P^{NEXP}-complete

Our solution: Define relation $\equiv_i$ where $T_1 \equiv_i T_2$ iff T_1 and T_2 have the same number of subtrees of depth $\leq i \cdot \text{poly}(|G|)$ up to some threshold θ.

- $\equiv_i$ is finer than $\simeq_i$.
- Testing if $f : \text{Trees}^{\leq i \cdot \text{poly}(|G|)} \to \{0, \ldots, \theta\}$ describes positive equivalence class w.r.t. φ is decidable in time $|f|^{\text{poly}(i+|G|)}$.
- $\forall$ positive f $\exists$ small NTA (computable) accepting $[\![f]\!]$.

Model checking $T_0 \stackrel{?}{\models} \text{EF}\varphi$ (φ modal formula) in NEXP:

1. Guess a function $f : \text{Trees}^{\leq i \cdot \text{poly}(|G|)} \to \{0, \ldots, \theta\}$.
2. Check whether f describes positive equivalence class w.r.t. φ.
3. Compute small NTA accepting $[\![f]\!]$.
4. Check if $T_0 \in \text{pre}^*([\![f]\!])$.

Model checking EF_1 on GTRS is P^{NEXP}-complete

Our solution: Define relation $\equiv_i$ where $T_1 \equiv_i T_2$ iff T_1 and T_2 have the same number of subtrees of depth $\leq i \cdot \text{poly}(|G|)$ up to some threshold θ.

- $\equiv_i$ is finer than $\simeq_i$.
- Testing if $f : \text{Trees}^{\leq i \cdot \text{poly}(|G|)} \to \{0, \ldots, \theta\}$ describes positive equivalence class w.r.t. φ is decidable in time $|f|^{\text{poly}(i+|G|)}$.
- $\forall$ positive f $\exists$ small NTA (computable) accepting $[\![f]\!]$.

Model checking $T_0 \stackrel{?}{\models} \text{EF}\varphi$ (φ modal formula) in NEXP:

1. Guess a function $f : \text{Trees}^{\leq i \cdot \text{poly}(|G|)} \to \{0, \ldots, \theta\}$.
2. Check whether f describes positive equivalence class w.r.t. φ.
3. Compute small NTA accepting $[\![f]\!]$.
4. Check if $T_0 \in \text{pre}^*([\![f]\!])$.

Finally use $P^{NEXP} = \text{PSPACE}^{NEXP}$ (Hemaspaandra,Allender et al).

Applications of the upper bound proof idea

Corollary

For a GTRS G and a finite system F one can decide in coNEXP *whether $G \sim F$.*

Applications of the upper bound proof idea

Corollary

For a GTRS G and a finite system F one can decide in coNEXP *whether $G \sim F$.*

Theorem

For a PA process P and a finite system F, one can decide in coNEXP *whether $P \sim F$.*

(gives a first elementary upper bound for this problem)

Model checking EF_1 on GTRS is P^{NEXP}-complete

Lower bound:

The proof is a combination of the following:

Model checking EF_1 on GTRS is P^{NEXP}-complete

Lower bound:

The proof is a combination of the following:

1. $2^n \times 2^n$-tiling problem is reducible to model checking formulas of the kind $EF\varphi$, where φ is a modal formula.
   (Uses ideas from satisfiability checking)

Model checking EF_1 on GTRS is P^{NEXP}-complete

Lower bound:

The proof is a combination of the following:

1. $2^n \times 2^n$-tiling problem is reducible to model checking formulas of the kind $EF\varphi$, where φ is a modal formula.
   (Uses ideas from satisfiability checking)
2. Encode Circuit Value for boolean circuits with access to $2^n \times 2^n$-tiling problem.

Decidability and complexity

$\mathrm{TOWER} = \mathrm{DTIME}(\mathrm{Tower}(O(n)))$.

	Pushdown	**GTRS**	**RGTRS**
MSO	TOWER-c.	undecidable	
FO + reach	TOWER-c.		
CTL	EXP-c.	undecidable	
$\mathrm{EF}_2, \mathrm{EF}$	PSPACE-c.	TOWER-c.	
EF_1	PSPACE-c.	$\mathrm{P}^{\mathrm{NEXP}}$-c.	TOWER-c.
$\sim$ vs. fin. syst.	PSPACE-c.	PSPACE...coNEXP	EXP...TOWER
$\approx$ vs. fin. syst.	PSPACE-c.	PSPACE...TOWER	EXP...TOWER

Decidability and complexity

$\mathrm{TOWER} = \mathrm{DTIME}(\mathrm{Tower}(O(n)))$.

	Pushdown	**GTRS**	**RGTRS**
MSO	TOWER-c.	undecidable	
FO + reach	TOWER-c.		
CTL	EXP-c.	undecidable	
$\mathrm{EF}_2, \mathrm{EF}$	PSPACE-c.	TOWER-c.	
EF_1	PSPACE-c.	$\mathrm{P}^{\mathrm{NEXP}}$-c.	TOWER-c.
$\sim$ vs. fin. syst.	PSPACE-c.	PSPACE...coNEXP	**TOWER**-c.
$\approx$ vs. fin. syst.	PSPACE-c.	**TOWER**-c.	

Nonelementary lower bounds for bisimilarity checking

Theorem

Given **RGTRS** G *and a finite system* F*, checking* $G \sim F$ *is nonelementary.*

Nonelementary lower bounds for bisimilarity checking

Theorem

Given **RGTRS** G *and a finite system* F*, checking* $G \sim F$ *is nonelementary.*

Corollary

Given **GTRS** G *and a finite system* F*, checking* $G \approx F$ *is nonelementary.*

Nonelementary lower bounds for bisimilarity checking

Theorem

Given **RGTRS** G *and a finite system* F*, checking* $G \sim F$ *is nonelementary.*

Corollary

Given **GTRS** G *and a finite system* F*, checking* $G \approx F$ *is nonelementary.*

- Attacker chooses witness tree "satisfying" first-order sentence.

Nonelementary lower bounds for bisimilarity checking

Theorem

Given **RGTRS** G *and a finite system* F*, checking* $G \sim F$ *is nonelementary.*

Corollary

Given **GTRS** G *and a finite system* F*, checking* $G \approx F$ *is nonelementary.*

- Attacker chooses witness tree "satisfying" first-order sentence.
- $\exists x_i$ (resp. $\forall x_i$) means Attacker (resp. Defender) labels leaf.

Nonelementary lower bounds for bisimilarity checking

Theorem

Given **RGTRS** G *and a finite system* F*, checking* $G \sim F$ *is nonelementary.*

Corollary

Given **GTRS** G *and a finite system* F*, checking* $G \approx F$ *is nonelementary.*

- Attacker chooses witness tree "satisfying" first-order sentence.
- $\exists x_i$ (resp. $\forall x_i$) means Attacker (resp. Defender) labels leaf.
- **Main obstacles for the proof:**
  - Order of variable assignments not controllable.

Nonelementary lower bounds for bisimilarity checking

Theorem

Given **RGTRS** G *and a finite system* F*, checking* $G \sim F$ *is nonelementary.*

Corollary

Given **GTRS** G *and a finite system* F*, checking* $G \approx F$ *is nonelementary.*

- Attacker chooses witness tree "satisfying" first-order sentence.
- $\exists x_i$ (resp. $\forall x_i$) means Attacker (resp. Defender) labels leaf.
- **Main obstacles for the proof:**
  - Order of variable assignments not controllable.
  - (R)GTRS not closed under products with finite systems to implement standard Defender's Forcing technique.

Thanks for your attention