

On the Computational Complexity of Verifying One-Counter Processes

Stefan Göller

Universität Bremen

joint work with Richard Mayr, and Anthony Widjaja To

MEMICS 2009

Model 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

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: $S \models \varphi?$

Infinite state model checking and its complexity

Class of structures	Logic	Complexity
Automatic structures	FO	NONELEMENTARY
Prefix-recognizable structures	MSO	NONELEMENTARY
Pushdown systems	μ -calculus	EXPTIME-complete
Pushdown systems	LTL	EXPTIME-complete
Pushdown systems	CTL	EXPTIME-complete
Pushdown systems	EF	PSPACE-complete
Pushdown systems	modal logic	PSPACE-complete

Model 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: $S \models \varphi?$

Infinite state model checking and its complexity

Class of structures	Logic	Complexity
Automatic structures	FO	NONELEMENTARY
Prefix-recognizable structures	MSO	NONELEMENTARY
Pushdown systems	μ -calculus	EXPTIME-complete
Pushdown systems	LTL	EXPTIME-complete
Pushdown systems	CTL	EXPTIME-complete
Pushdown systems	EF	PSPACE-complete
Pushdown systems	modal logic	PSPACE-complete

Pushdown systems

A pushdown system P is given by

- ▶ a finite set of **control states** p, q, \dots
- ▶ a finite set of **stack symbols** A, B, C, \dots
- ▶ labeled transitions of the kind
 - ▶ $pA \xrightarrow{l} q$ (pop),
 - ▶ $pA \xrightarrow{l'} qB$ (internal) or
 - ▶ $pA \xrightarrow{l''} qBC$ (push)

One-counter systems

A one-counter system O is given by

- ▶ a finite set of **control states** p, q, \dots
- ▶ a **single stack symbol** A
- ▶ labeled transitions of the kind
 - ▶ $pA \xrightarrow{l} q$ (pop),
 - ▶ $pA \xrightarrow{l'} qA$ (internal) or
 - ▶ $pA \xrightarrow{l''} qAA$ (push)

One-counter systems

Example for a one-counter system

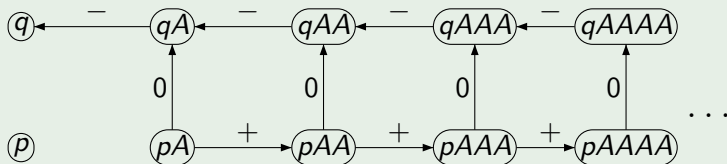
Transitions: $qA \xrightarrow{-} q$ $pA \xrightarrow{0} qA$ $pA \xrightarrow{+} pAA$

One-counter systems

Example for a one-counter system

Transitions: $qA \xrightarrow{-} q$ $pA \xrightarrow{0} qA$ $pA \xrightarrow{+} pAA$

Transition system $T(O)$:

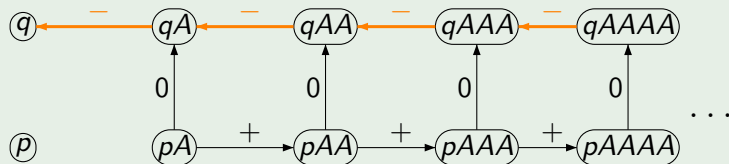


One-counter systems

Example for a one-counter system

Transitions: $qA \xrightarrow{-} q$ $pA \xrightarrow{0} qA$ $pA \xrightarrow{+} pAA$

Transition system $T(O)$:

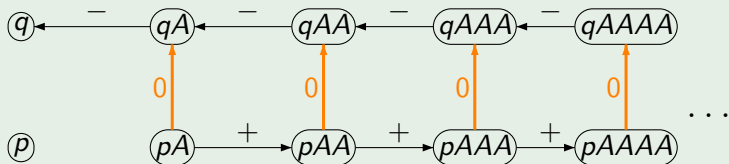


One-counter systems

Example for a one-counter system

Transitions: $qA \xrightarrow{-} q$ $pA \xrightarrow{0} qA$ $pA \xrightarrow{+} pAA$

Transition system $T(O)$:

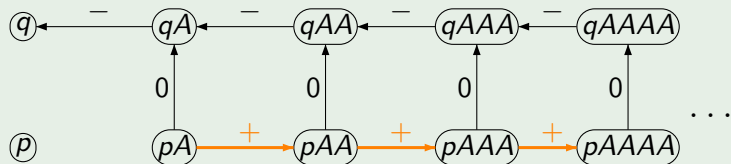


One-counter systems

Example for a one-counter system

Transitions: $qA \xrightarrow{-} q$ $pA \xrightarrow{0} qA$ $pA \xrightarrow{+} pAA$

Transition system $T(O)$:



Model checking: Pushdown systems vs. one-counter systems

Class of structures	Logic	Complexity
Pushdown systems One-counter systems	FO + Reach	NONELEMENTARY PSPACE-complete
Pushdown systems One-counter systems	μ -calculus	EXPTIME-complete PSPACE-complete
Pushdown systems One-counter systems	LTL	EXPTIME-complete PSPACE-complete
Pushdown systems One-counter systems	CTL	EXPTIME-complete DP-hard, in PSPACE
Pushdown systems One-counter systems	EF	PSPACE-complete DP-hard, in PSPACE
Pushdown systems One-counter systems	modal logic	PSPACE-complete P-complete

Model checking: Pushdown systems vs. one-counter systems

Class of structures	Logic	Complexity
Pushdown systems One-counter systems	FO + Reach	NONELEMENTARY PSPACE-complete
Pushdown systems One-counter systems	μ -calculus	EXPTIME-complete PSPACE-complete
Pushdown systems One-counter systems	LTL	EXPTIME-complete PSPACE-complete
Pushdown systems One-counter systems	CTL	EXPTIME-complete DP-hard, in PSPACE
Pushdown systems One-counter systems	EF	PSPACE-complete DP-hard, in PSPACE
Pushdown systems One-counter systems	modal logic	PSPACE-complete P-complete

This talk

The logic EF

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

$$\varphi ::= \text{true} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle a \rangle \varphi \mid \text{EF}\varphi,$$

where a ranges over a set of labels Σ .

The logic EF

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

$$\varphi ::= \text{true} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle a \rangle \varphi \mid \text{EF}\varphi,$$

where a ranges over a set of labels Σ .

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

The logic EF

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

$$\varphi ::= \text{true} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle a \rangle \varphi \mid \text{EF}\varphi,$$

where a ranges over a set of labels Σ .

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

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

The logic EF

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

$$\varphi ::= \text{true} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle a \rangle \varphi \mid \text{EF}\varphi,$$

where a ranges over a set of labels Σ .

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

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

$$s \models \text{true} \qquad \text{for all } s \in S$$

The logic EF

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

$$\varphi ::= \text{true} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle a \rangle \varphi \mid \text{EF}\varphi,$$

where a ranges over a set of labels Σ .

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

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

$$\begin{array}{ll} s \models \text{true} & \text{for all } s \in S \\ s \models \neg\varphi & \iff s \not\models \varphi \end{array}$$

The logic EF

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

$$\varphi ::= \text{true} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle a \rangle\varphi \mid \text{EF}\varphi,$$

where a ranges over a set of labels Σ .

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

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

$$s \models \text{true} \quad \text{for all } s \in S$$

$$s \models \neg\varphi \quad \iff \quad s \not\models \varphi$$

$$s \models \varphi_1 \wedge \varphi_2 \quad \iff \quad s \models \varphi_1 \quad \text{and} \quad s \models \varphi_2$$

The logic EF

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

$$\varphi ::= \text{true} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle a \rangle \varphi \mid \text{EF}\varphi,$$

where a ranges over a set of labels Σ .

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

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

$$\begin{array}{ll} s \models \text{true} & \text{for all } s \in S \\ s \models \neg\varphi & \iff s \not\models \varphi \\ s \models \varphi_1 \wedge \varphi_2 & \iff s \models \varphi_1 \text{ and } s \models \varphi_2 \\ s \models \langle a \rangle \varphi & \iff \exists t \in S : s \rightarrow_a t \text{ and } t \models \varphi \end{array}$$

The logic EF

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

$$\varphi ::= \text{true} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle a \rangle\varphi \mid \text{EF}\varphi,$$

where a ranges over a set of labels Σ .

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

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

$$\begin{array}{ll} s \models \text{true} & \text{for all } s \in S \\ s \models \neg\varphi & \iff s \not\models \varphi \\ s \models \varphi_1 \wedge \varphi_2 & \iff s \models \varphi_1 \text{ and } s \models \varphi_2 \\ s \models \langle a \rangle\varphi & \iff \exists t \in S : s \rightarrow_a t \text{ and } t \models \varphi \\ s \models \text{EF}\varphi & \iff \exists t \in S : s \rightarrow^* t \text{ and } t \models \varphi \\ & \text{where } \rightarrow = \cup_{a \in \Sigma} \rightarrow_a \end{array}$$

Some complexity classes

$$\text{DP} = \{L_1 \cap L_2 \mid L_1 \in \text{NP}, L_2 \in \text{coNP}\}.$$

Some complexity classes

$$\text{DP} = \{L_1 \cap L_2 \mid L_1 \in \text{NP}, L_2 \in \text{coNP}\}.$$

For complexity classes \mathcal{C} and \mathcal{D} , define $\mathcal{C}^{\mathcal{D}}$ to be the languages decidable by some Turing machine from \mathcal{C} with oracle access to some language from \mathcal{D} .

Some complexity classes

$$\text{DP} = \{L_1 \cap L_2 \mid L_1 \in \text{NP}, L_2 \in \text{coNP}\}.$$

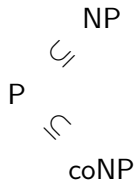
For complexity classes \mathcal{C} and \mathcal{D} , define $\mathcal{C}^{\mathcal{D}}$ to be the languages decidable by some Turing machine from \mathcal{C} with oracle access to some language from \mathcal{D} .

P

Some complexity classes

$$\text{DP} = \{L_1 \cap L_2 \mid L_1 \in \text{NP}, L_2 \in \text{coNP}\}.$$

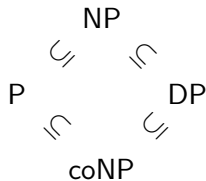
For complexity classes \mathcal{C} and \mathcal{D} , define $\mathcal{C}^{\mathcal{D}}$ to be the languages decidable by some Turing machine from \mathcal{C} with oracle access to some language from \mathcal{D} .



Some complexity classes

$$\text{DP} = \{L_1 \cap L_2 \mid L_1 \in \text{NP}, L_2 \in \text{coNP}\}.$$

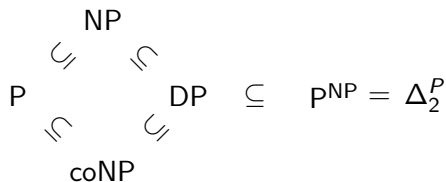
For complexity classes \mathcal{C} and \mathcal{D} , define $\mathcal{C}^{\mathcal{D}}$ to be the languages decidable by some Turing machine from \mathcal{C} with oracle access to some language from \mathcal{D} .



Some complexity classes

$$\text{DP} = \{L_1 \cap L_2 \mid L_1 \in \text{NP}, L_2 \in \text{coNP}\}.$$

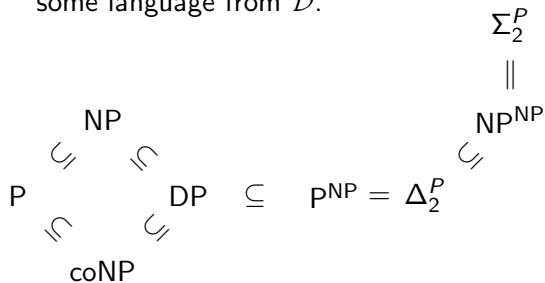
For complexity classes \mathcal{C} and \mathcal{D} , define $\mathcal{C}^{\mathcal{D}}$ to be the languages decidable by some Turing machine from \mathcal{C} with oracle access to some language from \mathcal{D} .



Some complexity classes

$$\text{DP} = \{L_1 \cap L_2 \mid L_1 \in \text{NP}, L_2 \in \text{coNP}\}.$$

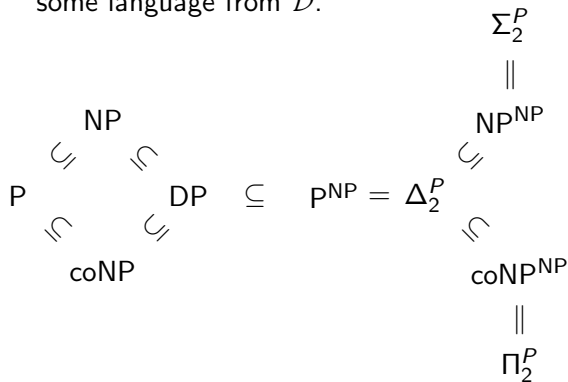
For complexity classes \mathcal{C} and \mathcal{D} , define $\mathcal{C}^{\mathcal{D}}$ to be the languages decidable by some Turing machine from \mathcal{C} with oracle access to some language from \mathcal{D} .



Some complexity classes

$$\text{DP} = \{L_1 \cap L_2 \mid L_1 \in \text{NP}, L_2 \in \text{coNP}\}.$$

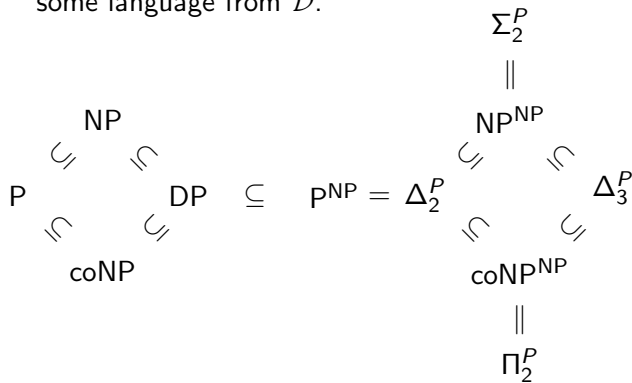
For complexity classes \mathcal{C} and \mathcal{D} , define $\mathcal{C}^{\mathcal{D}}$ to be the languages decidable by some Turing machine from \mathcal{C} with oracle access to some language from \mathcal{D} .



Some complexity classes

$$\text{DP} = \{L_1 \cap L_2 \mid L_1 \in \text{NP}, L_2 \in \text{coNP}\}.$$

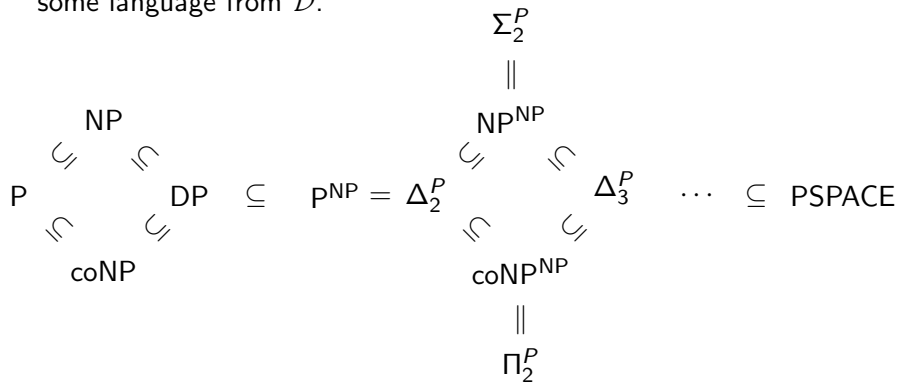
For complexity classes \mathcal{C} and \mathcal{D} , define $\mathcal{C}^{\mathcal{D}}$ to be the languages decidable by some Turing machine from \mathcal{C} with oracle access to some language from \mathcal{D} .



Some complexity classes

$$\text{DP} = \{L_1 \cap L_2 \mid L_1 \in \text{NP}, L_2 \in \text{coNP}\}.$$

For complexity classes \mathcal{C} and \mathcal{D} , define $\mathcal{C}^{\mathcal{D}}$ to be the languages decidable by some Turing machine from \mathcal{C} with oracle access to some language from \mathcal{D} .



Model checking EF over one-counter processes

State of the art:

- ▶ Lower bound: DP
(P. Jančar, A. Kučera, F. Moller, and Z. Sawa, 2004)

Model checking EF over one-counter processes

State of the art:

- ▶ Lower bound: **DP**
(P. Jančar, A. Kučera, F. Moller, and Z. Sawa, 2004)
- ▶ Upper bound: **PSPACE** inherited from modal μ -calculus
(O. Serre, 2006)

Model checking EF over one-counter processes

State of the art:

- ▶ Lower bound: **DP**
(P. Jančar, A. Kučera, F. Moller, and Z. Sawa, 2004)
- ▶ Upper bound: **PSPACE** inherited from modal μ -calculus
(O. Serre, 2006)

Theorem

Model checking EF over one-counter processes is P^{NP} -complete.

Model checking EF over one-counter processes

State of the art:

- ▶ Lower bound: **DP**
(P. Jančar, A. Kučera, F. Moller, and Z. Sawa, 2004)
- ▶ Upper bound: **PSPACE** inherited from modal μ -calculus
(O. Serre, 2006)

Theorem

Model checking EF over one-counter processes is P^{NP} -complete.

Assumption: EF-formulas are given as directed acyclic graphs!

Model checking EF over one-counter processes

State of the art:

- ▶ Lower bound: **DP**
(P. Jančar, A. Kučera, F. Moller, and Z. Sawa, 2004)
- ▶ Upper bound: **PSPACE** inherited from modal μ -calculus
(O. Serre, 2006)

Theorem

Model checking EF over one-counter processes is P^{NP} -complete.

Assumption: EF-formulas are given as directed acyclic graphs!

But:

Theorem (G., Lohrey 2009)

Model-checking of one-counter processes against EF formulas in tree representation is hard for P^{NP} .

Model checking EF over one-counter processes

State of the art:

- ▶ Lower bound: **DP**
(P. Jančar, A. Kučera, F. Moller, and Z. Sawa, 2004)
- ▶ Upper bound: **PSPACE** inherited from modal μ -calculus
(O. Serre, 2006)

Theorem

Model checking EF over one-counter processes is P^{NP} -complete.

Assumption: EF-formulas are given as directed acyclic graphs!

But:

Theorem (G., Lohrey 2009)

Model-checking of one-counter processes against EF formulas in tree representation is hard for P^{NP} .

Proof strategy for the P^{NP} upper bound

1. Find a quantifier-free logic \mathcal{L} over \mathbb{N} such that the following problem is in P^{NP} :

\mathcal{L} -MEMBERSHIP

Input: $n \in \mathbb{N}$ in binary, $\alpha \in \mathcal{L}$

Question: $n \in \llbracket \alpha \rrbracket$?

Proof strategy for the P^{NP} upper bound

1. Find a quantifier-free logic \mathcal{L} over \mathbb{N} such that the following problem is in P^{NP} :

\mathcal{L} -MEMBERSHIP

Input: $n \in \mathbb{N}$ in binary, $\alpha \in \mathcal{L}$

Question: $n \in \llbracket \alpha \rrbracket$?

2. Compute the following in polynomial time:

FROM MODEL CHECKING TO \mathcal{L} -MEMBERSHIP

Input: One-counter process O , control state q , $\varphi \in EF$

Output: $\alpha(q, \varphi) \in \mathcal{L}$ such that

$$\llbracket \alpha(q, \varphi) \rrbracket = \{n \in \mathbb{N} \mid (q, n) \models \varphi\}.$$

Proof strategy for the P^{NP} upper bound

1. Find a quantifier-free logic \mathcal{L} over \mathbb{N} such that the following problem is in P^{NP} :

\mathcal{L} -MEMBERSHIP

Input: $n \in \mathbb{N}$ in binary, $\alpha \in \mathcal{L}$

Question: $n \in \llbracket \alpha \rrbracket$?

2. Compute the following in polynomial time:

FROM MODEL CHECKING TO \mathcal{L} -MEMBERSHIP

Input: One-counter process O , control state q , $\varphi \in EF$

Output: $\alpha(q, \varphi) \in \mathcal{L}$ such that

$$\llbracket \alpha(q, \varphi) \rrbracket = \{n \in \mathbb{N} \mid (q, n) \models \varphi\}.$$

Proof strategy for the P^{NP} upper bound

1. Find a quantifier-free logic \mathcal{L} over \mathbb{N} such that the following problem is in P^{NP} :

\mathcal{L} -MEMBERSHIP

Input: $n \in \mathbb{N}$ in binary, $\alpha \in \mathcal{L}$

Question: $n \in \llbracket \alpha \rrbracket$?

2. Compute the following in polynomial time:

FROM MODEL CHECKING TO \mathcal{L} -MEMBERSHIP

Input: One-counter process O , control state q , $\varphi \in EF$

Output: $\alpha(q, \varphi) \in \mathcal{L}$ such that

$$\llbracket \alpha(q, \varphi) \rrbracket = \{n \in \mathbb{N} \mid (q, n) \models \varphi\}.$$

MMA: MinMax Arithmetic

Formulas of MMA are given as follows, where $a, b \in \mathbb{N}$:

$$\alpha ::= \equiv a \bmod b \mid \geq b \mid \neg\alpha \mid \alpha \wedge \alpha \mid \geq \min \alpha \mid \geq \max \alpha$$

MMA: MinMax Arithmetic

Formulas of MMA are given as follows, where $a, b \in \mathbb{N}$:

$$\alpha ::= \equiv a \bmod b \mid \geq b \mid \neg\alpha \mid \alpha \wedge \alpha \mid \geq \min \alpha \mid \geq \max \alpha$$

Semantics:

$$\llbracket \equiv a \bmod b \rrbracket = \{n \in \mathbb{N} \mid n \equiv a \bmod b\}$$

MMA: MinMax Arithmetic

Formulas of MMA are given as follows, where $a, b \in \mathbb{N}$:

$$\alpha ::= \equiv a \bmod b \mid \geq b \mid \neg\alpha \mid \alpha \wedge \alpha \mid \geq \min \alpha \mid \geq \max \alpha$$

Semantics:

$$\llbracket \geq b \rrbracket = \{n \in \mathbb{N} \mid n \geq b\}$$

MMA: MinMax Arithmetic

Formulas of MMA are given as follows, where $a, b \in \mathbb{N}$:

$$\alpha ::= \equiv a \bmod b \mid \geq b \mid \neg\alpha \mid \alpha \wedge \alpha \mid \geq \min \alpha \mid \geq \max \alpha$$

Semantics:

$$\llbracket \neg\alpha \rrbracket = \mathbb{N} \setminus \llbracket \alpha \rrbracket$$

MMA: MinMax Arithmetic

Formulas of MMA are given as follows, where $a, b \in \mathbb{N}$:

$$\alpha ::= \equiv a \bmod b \mid \geq b \mid \neg \alpha \mid \alpha \wedge \alpha \mid \geq \min \alpha \mid \geq \max \alpha$$

Semantics:

$$\llbracket \alpha_1 \wedge \alpha_2 \rrbracket = \llbracket \alpha_1 \rrbracket \cap \llbracket \alpha_2 \rrbracket$$

MMA: MinMax Arithmetic

Formulas of MMA are given as follows, where $a, b \in \mathbb{N}$:

$$\alpha ::= \equiv a \bmod b \mid \geq b \mid \neg\alpha \mid \alpha \wedge \alpha \mid \geq \mathbf{\min} \alpha \mid \geq \mathbf{\max} \alpha$$

Semantics:

$$\llbracket \geq \mathbf{\min} \alpha \rrbracket = \{n \in \mathbb{N} \mid n \geq \min\llbracket \alpha \rrbracket\}$$

MMA: MinMax Arithmetic

Formulas of MMA are given as follows, where $a, b \in \mathbb{N}$:

$$\alpha ::= \equiv a \bmod b \mid \geq b \mid \neg\alpha \mid \alpha \wedge \alpha \mid \geq \min \alpha \mid \geq \max \alpha$$

Semantics:

$$\llbracket \geq \max \alpha \rrbracket = \{n \in \mathbb{N} \mid n \geq \max\llbracket \alpha \rrbracket\}$$

MMA: MinMax Arithmetic

Formulas of MMA are given as follows, where $a, b \in \mathbb{N}$:

$$\alpha ::= \equiv a \bmod b \mid \geq b \mid \neg\alpha \mid \alpha \wedge \alpha \mid \geq \min \alpha \mid \geq \max \alpha$$

Important for our translation later:

Formulas are given as directed acyclic graphs!

MMA: MinMax Arithmetic

Formulas of MMA are given as follows, where $a, b \in \mathbb{N}$:

$$\alpha ::= \equiv a \bmod b \mid \geq b \mid \neg\alpha \mid \alpha \wedge \alpha \mid \geq \min \alpha \mid \geq \max \alpha$$

Important for our translation later:

Formulas are given as directed acyclic graphs!

Note on expressiveness

The sets definable in MMA are precisely the sets definable in Presburger Arithmetic.

Membership in MMA

Lemma (Periodicity of MMA)

The following is computable in polynomial time:

Input: An MMA formula α

Output: *Threshold* t and *period* p such that for all $n_1, n_2 > t$:

$$n_1 \equiv n_2 \pmod{p} \implies (n_1 \in \llbracket \alpha \rrbracket \Leftrightarrow n_2 \in \llbracket \alpha \rrbracket)$$

Membership in MMA

Lemma (Periodicity of MMA)

The following is computable in polynomial time:

Input: An MMA formula α

Output: *Threshold* t and *period* p such that for all $n_1, n_2 > t$:

$$n_1 \equiv n_2 \pmod{p} \implies (n_1 \in \llbracket \alpha \rrbracket \Leftrightarrow n_2 \in \llbracket \alpha \rrbracket)$$

Proposition (membership problem for MMA)

The following problem is in P^{NP} :

Input: $n \in \mathbb{N}$ in binary, $\alpha \in \text{MMA}$

Question: $n \in \llbracket \alpha \rrbracket$?

Membership in MMA

Lemma (Periodicity of MMA)

The following is computable in polynomial time:

Input: An MMA formula α

Output: *Threshold* t and *period* p such that for all $n_1, n_2 > t$:

$$n_1 \equiv n_2 \pmod{p} \implies (n_1 \in \llbracket \alpha \rrbracket \Leftrightarrow n_2 \in \llbracket \alpha \rrbracket)$$

Proposition (membership problem for MMA)

The following problem is in P^{NP} :

Input: $n \in \mathbb{N}$ in binary, $\alpha \in \text{MMA}$

Question: $n \in \llbracket \alpha \rrbracket$?

Proof.

Successively eliminate min and max operators by replacing them with constants: Doable in P^{NP} . □

Proof strategy for the P^{NP} upper bound

1. Find a quantifier-free logic \mathcal{L} over \mathbb{N} such that the following problem is in P^{NP} :

\mathcal{L} -MEMBERSHIP

Input: $n \in \mathbb{N}$ in binary, $\alpha \in \mathcal{L}$

Question: $n \in \llbracket \alpha \rrbracket$?

2. Compute the following in polynomial time:

FROM MODEL CHECKING TO \mathcal{L} -MEMBERSHIP

Input: One-counter process O , control state q , $\varphi \in EF$

Output: $\alpha(q, \varphi) \in \mathcal{L}$ such that

$$\llbracket \alpha(q, \varphi) \rrbracket = \{n \in \mathbb{N} \mid (q, n) \models \varphi\}.$$

Proof strategy for the P^{NP} upper bound

1. Find a quantifier-free logic \mathcal{L} over \mathbb{N} such that the following problem is in P^{NP} :

\mathcal{L} -MEMBERSHIP

Input: $n \in \mathbb{N}$ in binary, $\alpha \in \mathcal{L}$

Question: $n \in \llbracket \alpha \rrbracket$?

2. Compute the following in polynomial time:

FROM MODEL CHECKING TO \mathcal{L} -MEMBERSHIP

Input: One-counter process O , control state q , $\varphi \in EF$

Output: $\alpha(q, \varphi) \in \mathcal{L}$ such that

$$\llbracket \alpha(q, \varphi) \rrbracket = \{n \in \mathbb{N} \mid (q, n) \models \varphi\}.$$

From model checking to MMA-membership

An example for translating $EF\varphi$

Transitions: $q_0A \xrightarrow{-} q_2$ $q_2A \xrightarrow{-} q_1$ $q_1A \xrightarrow{-} q_0$

From model checking to MMA-membership

An example for translating $EF\varphi$

Transitions: $q_0 A \xrightarrow{-} q_2$ $q_2 A \xrightarrow{-} q_1$ $q_1 A \xrightarrow{-} q_0$

Assume we have already computed:

$$\alpha(q_i, \varphi) = \{n \in \mathbb{N} \mid (q_i, n) \models \varphi\} \quad \text{for each } i \in \{0, 1, 2\}$$

From model checking to MMA-membership

An example for translating $EF\varphi$

Transitions: $q_0 A \xrightarrow{-} q_2$ $q_2 A \xrightarrow{-} q_1$ $q_1 A \xrightarrow{-} q_0$

Assume we have already computed:

$$\alpha(q_i, \varphi) = \{n \in \mathbb{N} \mid (q_i, n) \models \varphi\} \quad \text{for each } i \in \{0, 1, 2\}$$

How to define $\alpha(q_0, EF\varphi)$?

From model checking to MMA-membership

An example for translating $EF\varphi$

Transitions: $q_0A \xrightarrow{-} q_2$ $q_2A \xrightarrow{-} q_1$ $q_1A \xrightarrow{-} q_0$

Assume we have already computed:

$$\alpha(q_i, \varphi) = \{n \in \mathbb{N} \mid (q_i, n) \models \varphi\} \quad \text{for each } i \in \{0, 1, 2\}$$

How to define $\alpha(q_0, EF\varphi)$?

$$\bigvee_{b \in \{0, 1, 2\}} \equiv b \pmod{3} \wedge$$

From model checking to MMA-membership

An example for translating $EF\varphi$

Transitions: $q_0A \xrightarrow{-} q_2$ $q_2A \xrightarrow{-} q_1$ $q_1A \xrightarrow{-} q_0$

Assume we have already computed:

$$\alpha(q_i, \varphi) = \{n \in \mathbb{N} \mid (q_i, n) \models \varphi\} \quad \text{for each } i \in \{0, 1, 2\}$$

How to define $\alpha(q_0, EF\varphi)$?

$$\bigvee_{b \in \{0, 1, 2\}} \equiv b \pmod{3} \wedge \left(\geq \min(\equiv b \pmod{3} \wedge \alpha(q_0, \varphi)) \vee \right)$$

From model checking to MMA-membership

An example for translating $EF\varphi$

Transitions: $q_0 A \xrightarrow{-} q_2$ $q_2 A \xrightarrow{-} q_1$ $q_1 A \xrightarrow{-} q_0$

Assume we have already computed:

$$\alpha(q_i, \varphi) = \{n \in \mathbb{N} \mid (q_i, n) \models \varphi\} \quad \text{for each } i \in \{0, 1, 2\}$$

How to define $\alpha(q_0, EF\varphi)$?

$$\bigvee_{b \in \{0, 1, 2\}} \equiv b \pmod{3} \wedge \left(\geq \min(\equiv b \pmod{3} \wedge \alpha(q_0, \varphi)) \vee \right. \\ \left. \geq \min(\equiv b + 1 \pmod{3} \wedge \alpha(q_1, \varphi)) \right) \vee$$

From model checking to MMA-membership

An example for translating $EF\varphi$

Transitions: $q_0 A \xrightarrow{-} q_2$ $q_2 A \xrightarrow{-} q_1$ $q_1 A \xrightarrow{-} q_0$

Assume we have already computed:

$$\alpha(q_i, \varphi) = \{n \in \mathbb{N} \mid (q_i, n) \models \varphi\} \quad \text{for each } i \in \{0, 1, 2\}$$

How to define $\alpha(q_0, EF\varphi)$?

$$\bigvee_{b \in \{0, 1, 2\}} \equiv b \pmod{3} \wedge \left(\begin{aligned} &\geq \min(\equiv b \pmod{3} \wedge \alpha(q_0, \varphi)) \vee \\ &\geq \min(\equiv b + 1 \pmod{3} \wedge \alpha(q_1, \varphi)) \vee \\ &\geq \min(\equiv b + 2 \pmod{3} \wedge \alpha(q_2, \varphi)) \end{aligned} \right)$$

Main result on weak bisimilarity

Theorem

The following problem is complete for P^{NP} :

WEAK BISIMILARITY CHECKING AGAINST FINITE SYSTEMS

Input: *State s of a one-counter process, state t of a finite transition system.*

Question: $s \approx t?$

Main result on weak bisimilarity

Theorem

The following problem is complete for P^{NP} :

WEAK BISIMILARITY CHECKING AGAINST FINITE SYSTEMS

Input: *State s of a one-counter process, state t of a finite transition system.*

Question: $s \approx t?$

Remarks:

- ▶ Upper bound: Reduction to EF model checking over one-counter processes (formulas represented as dags).

Main result on weak bisimilarity

Theorem

The following problem is complete for P^{NP} :

WEAK BISIMILARITY CHECKING AGAINST FINITE SYSTEMS

Input: *State s of a one-counter process, state t of a finite transition system.*

Question: $s \approx t?$

Remarks:

- ▶ Upper bound: Reduction to EF model checking over one-counter processes (formulas represented as dags).
- ▶ Lower bound: Involved reduction from DSAT:

Input: $x_i = \exists \mathbf{Y}_i : \varphi_i(x_1, \dots, x_{i-1}, \mathbf{Y}_i)$, where $1 \leq i \leq n$

Question: $x_n = 1?$

Verification of one-counter processes: Outlook

Theorem (G., Lohrey 2009)

CTL model checking over one-counter processes is PSPACE-hard, even if one fixes either the formula or the one-counter processes.

Verification of one-counter processes: Outlook

Theorem (G., Lohrey 2009)

CTL model checking over one-counter processes is PSPACE-hard, even if one fixes either the formula or the one-counter processes.

Open problems:

- ▶ Model checking **fixed** EF formulas over one-counter processes:
 $P^{NP[\log]}$ -hard and in P^{NP}
- ▶ Weak bisimilarity against **fixed** finite systems:
 $P^{NP[\log]}$ -hard and in P^{NP}