# Algebraic Reasoning Meets Automata in Solving Linear Integer Arithmetic

Peter Habermehl[1], Vojtěch Havlena[2], Michal Hečko[2], Lukáš Holík[2], Ondřej Lengál[2]

[1] Université Paris Cité, IRIF, Paris, France
[2] Faculty of Information Technology, Brno University of Technology, Brno, Czech Republic

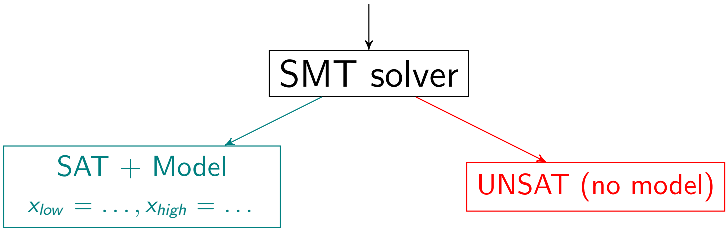
CAV'24

# Motivation: binary search correctness

$$\varphi\colon\ (x_{low} > x_{high} \ \lor\ 0 \le x_{low} < x_{high} < |A|) \land$$
$$(x_{low} \le x_{high} \ \rightarrow\ 0 \le \frac{x_{low} + x_{high}}{2} < |A|)$$

The midpoint must be within array bounds

Are there valid assignments to $x_{low}$ and $x_{high}$ violating the assertion $\varphi$?

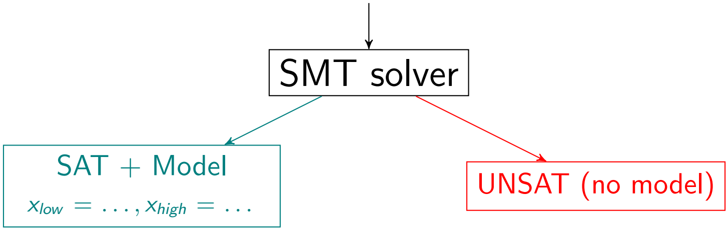


# Motivation: binary search correctness

$$\varphi:\ (x_{low} > x_{high} \ \lor\ 0 \leq x_{low} < x_{high} < |A|) \land$$
$$(x_{low} \leq x_{high} \ \rightarrow\ 0 \leq \frac{x_{low} + x_{high}}{2} < |A|)$$

The midpoint must be within array bounds

Are there valid assignments to $x_{low}$ and $x_{high}$ violating the assertion $\varphi$?



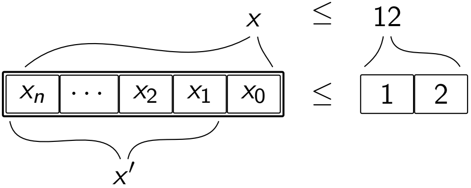

We are interested in *quantified* formulae, as they frequently pose a challenge to the state-of-the-art solvers.

# Intuition: Constructing automata from atomic formulae

Key observation: Any number $x$ can be written as its least-significant digit $x_0$ and remaining digits $x'$, i.e., $x = x_0 + 10x'$

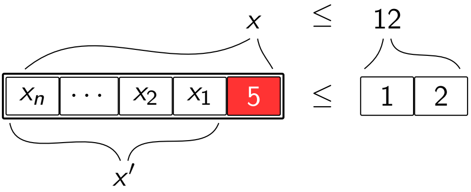
# Intuition: Constructing automata from atomic formulae

Key observation: Any number $x$ can be written as its least-significant digit $x_0$ and remaining digits $x'$, i.e., $x = x_0 + 10x'$



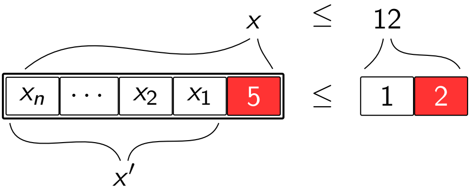
# Intuition: Constructing automata from atomic formulae

Key observation: Any number $x$ can be written as its least-significant digit $x_0$ and remaining digits $x'$, i.e., $x = x_0 + 10x'$



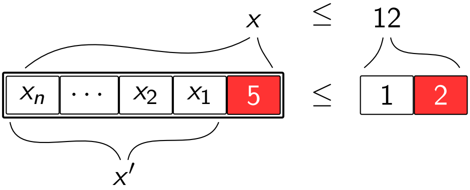
# Intuition: Constructing automata from atomic formulae

Key observation: Any number $x$ can be written as its least-significant digit $x_0$ and remaining digits $x'$, i.e., $x = x_0 + 10x'$

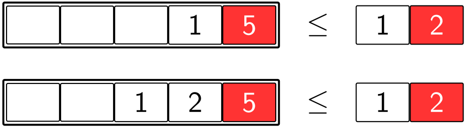


# Intuition: Constructing automata from atomic formulae

Key observation: Any number $x$ can be written as its least-significant digit $x_0$ and remaining digits $x'$, i.e., $x = x_0 + 10x'$

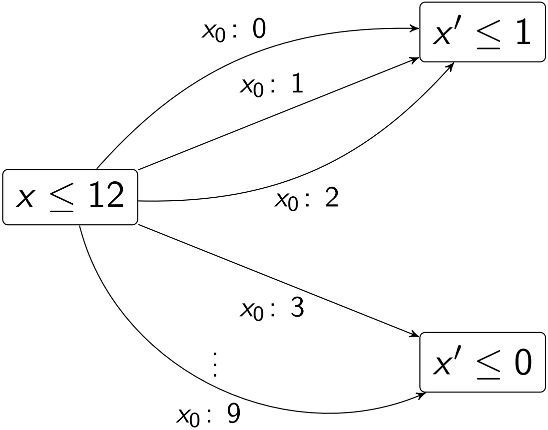



Since $5 > 2$, it must hold that $x' \leq 0$, otherwise we would get, e.g.,



# Intuition: Constructing automata from atomic formulae

Key observation: Any number $x$ can be written as its least-significant digit $x_0$ and remaining digits $x'$, i.e., $x = x_0 + 10x'$



# From atoms to automata

## Encoding assignments as words

- Encoding variable assignments as words using Least Significant Bit First (LSBF) encoding

$$\begin{aligned}\sigma(x) &= (-6)_{10} = (\underline{0}101)_2 = (\underline{0}101)_2 \\ \sigma(y) &= \phantom{-}(2)_{10} = (\underline{0}10)_2 \phantom{1} = (\underline{0}100)_2\end{aligned} \quad \rightsquigarrow \quad w_\sigma = \begin{matrix} x: \\ y: \end{matrix} \begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}$$

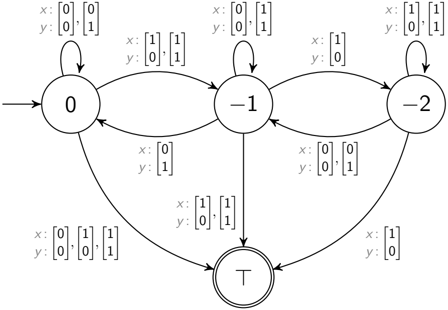
# From atoms to automata

## Encoding assignments as words

- Encoding variable assignments as words using Least Significant Bit First (LSBF) encoding

$$\sigma(x) = (-6)_{10} = (\underline{0}101)_2 = (\underline{0}101)_2$$
$$\sigma(y) = (2)_{10} = (\underline{0}10)_2 = (\underline{0}100)_2$$
$$\rightsquigarrow \quad w_\sigma = \begin{matrix} x: \\ y: \end{matrix} \begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}$$

- NFA accepting the solutions of $2x - y \leq 0$



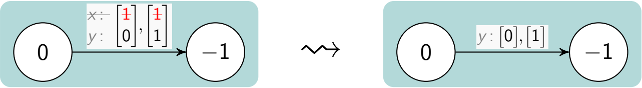
# Deciding linear integer arithmetic (LIA)

. . . the automata way

- Construct an NFA for every atom in the input formula
- Proceed inductively: construct $\mathcal{A}_{\varphi \diamond \psi}$ from $\mathcal{A}_\varphi$ and $\mathcal{A}_\psi$ using an $\mathcal{A}$-construction corresponding to $\diamond$

$\varphi \wedge \psi \rightsquigarrow \mathcal{L}(\mathcal{A}_\varphi) \cap \mathcal{L}(\mathcal{A}_\varphi)$    $\varphi \vee \psi \rightsquigarrow \mathcal{L}(\mathcal{A}_\varphi) \cup \mathcal{L}(\mathcal{A}_\varphi)$    $\neg\varphi \rightsquigarrow \Sigma^* \setminus \mathcal{L}(\mathcal{A}_\varphi)$

- Thus, for every subformula $\varphi$, construct an NFA $\mathcal{A}_\varphi$ accepting all of its solutions
- Quantifiers $\exists x$ are handled by projecting away the variable track corresponding to $x$



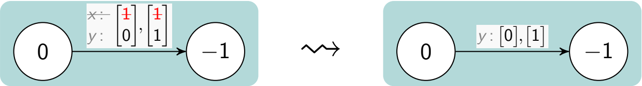
# Deciding linear integer arithmetic (LIA)

. . . the automata way

- Construct an NFA for every atom in the input formula
- Proceed inductively: construct $\mathcal{A}_{\varphi \diamond \psi}$ from $\mathcal{A}_\varphi$ and $\mathcal{A}_\psi$ using an $\mathcal{A}$-construction corresponding to $\diamond$

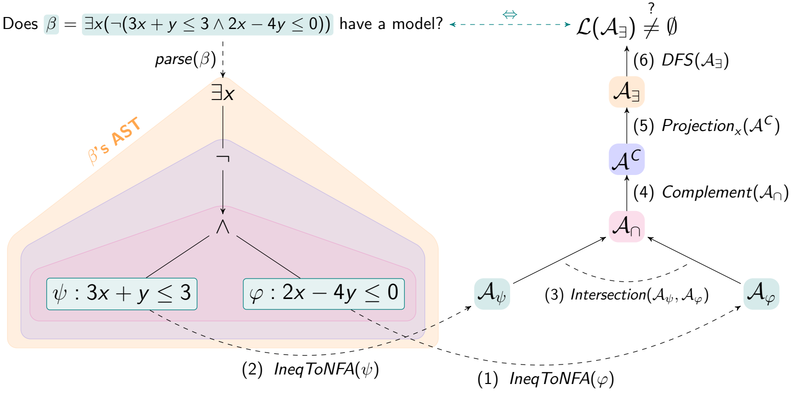
$\varphi \wedge \psi \rightsquigarrow \mathcal{L}(\mathcal{A}_\varphi) \cap \mathcal{L}(\mathcal{A}_\varphi)$ $\quad$ $\varphi \vee \psi \rightsquigarrow \mathcal{L}(\mathcal{A}_\varphi) \cup \mathcal{L}(\mathcal{A}_\varphi)$ $\quad$ $\neg\varphi \rightsquigarrow \Sigma^* \setminus \mathcal{L}(\mathcal{A}_\varphi)$

- Thus, for every subformula $\varphi$, construct an NFA $\mathcal{A}_\varphi$ accepting all of its solutions
- Quantifiers $\exists x$ are handled by projecting away the variable track corresponding to $x$




- Very simple procedure $\rightsquigarrow$ can have a poor performance even at the induction base when constructing an NFA for an atom
  - $\varphi = 55x + 77y \leq 0 \quad \rightsquigarrow \quad |\mathcal{A}_\varphi| = 55 + 77 + 1$

# A comprehensive example



# Introducing algebraic reasoning to the $\mathcal{A}$-based procedure

An intuitive overview

1. Rewriting formulae into equivalent ones
   - Core theme: **finding a value of an existentially quantified variable** that restricts the free variables the least
   - Result is much easier to decide using automata (smaller number of intermediate automata with less states)

# Introducing algebraic reasoning to the $\mathcal{A}$-based procedure

## An intuitive overview

1. Rewriting formulae into equivalent ones
   - Core theme: **finding a value of an existentially quantified variable** that restricts the free variables the least
   - Result is much easier to decide using automata (smaller number of intermediate automata with less states)

2. Algebraic reasoning during the decision procedure
   - **states = LIA formulae** precisely describing their languages
   - compact representation of the language of every state $\rightsquigarrow$ on-the-fly pruning without the need to have the entire automaton upfront

# Rewriting using monotonicity

## Exploiting variable relations to improve performance

# Rewriting using monotonicity

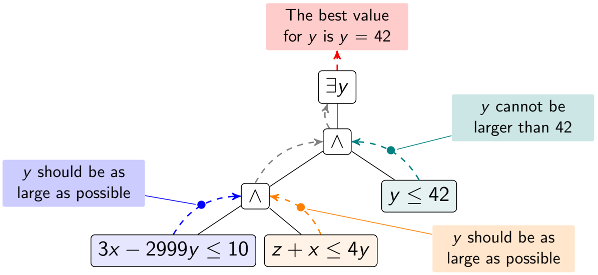
## Exploiting variable relations to improve performance

- Core idea: Given $\varphi$, find $\psi$ such that $\varphi \Leftrightarrow \psi$ with $\mathcal{A}_\psi$ being easier to construct than $\mathcal{A}_\varphi$

# Rewriting using monotonicity

## Exploiting variable relations to improve performance

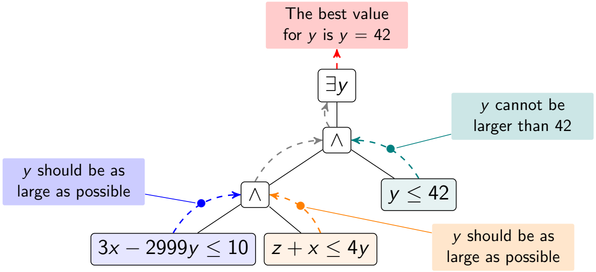
- Core idea: Given $\varphi$, find $\psi$ such that $\varphi \Leftrightarrow \psi$ with $\mathcal{A}_\psi$ being easier to construct than $\mathcal{A}_\varphi$
- Use dataflow analysis to extract useful variable relations
  - $\rightsquigarrow$ notion of monotonicity ($c$-best-from-{below, above})



# Rewriting using monotonicity

## Exploiting variable relations to improve performance

- Core idea: Given $\varphi$, find $\psi$ such that $\varphi \Leftrightarrow \psi$ with $\mathcal{A}_\psi$ being easier to construct than $\mathcal{A}_\varphi$
- Use dataflow analysis to extract useful variable relations
  - $\rightsquigarrow$ notion of monotonicity ($c$-best-from-{below, above})

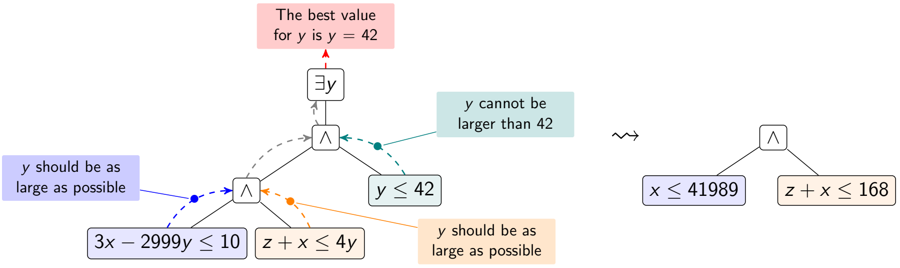



- Use the results to rewrite $\varphi$, removing existential quantifiers
  - $\varphi$ is $c$-best-from-below w.r.t. $y \rightsquigarrow \exists y(\varphi(\vec{x}, y)) \Leftrightarrow \varphi[c/y]$
  - basis for other tricks such as modulo linearization

# Rewriting using monotonicity

## Exploiting variable relations to improve performance

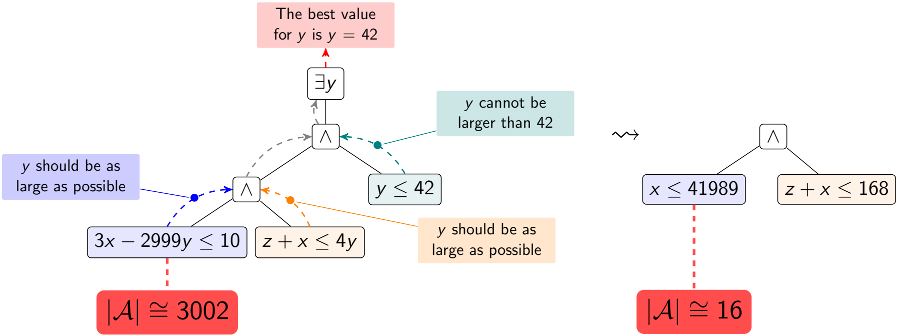
- Core idea: Given $\varphi$, find $\psi$ such that $\varphi \Leftrightarrow \psi$ with $\mathcal{A}_\psi$ being easier to construct than $\mathcal{A}_\varphi$
- Use dataflow analysis to extract useful variable relations
  - $\rightsquigarrow$ notion of monotonicity ($c$-best-from-{below, above})



# Rewriting using monotonicity

## Exploiting variable relations to improve performance

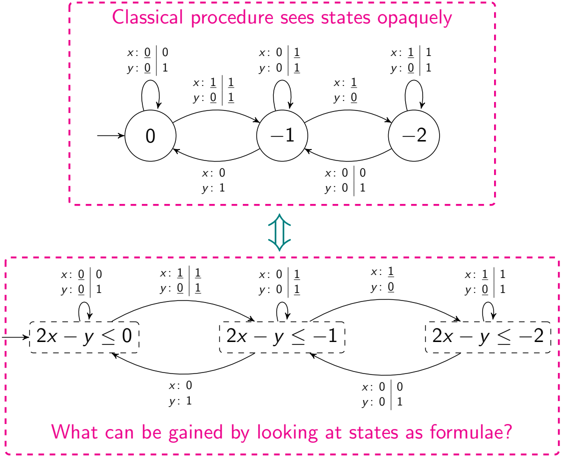
- Core idea: Given $\varphi$, find $\psi$ such that $\varphi \Leftrightarrow \psi$ with $\mathcal{A}_\psi$ being easier to construct than $\mathcal{A}_\varphi$
- Use dataflow analysis to extract useful variable relations
  - $\rightsquigarrow$ notion of monotonicity ($c$-best-from-{below, above})



# Top-down reformulation of the decision procedure

## Duality between formulae and states

NFA $\mathcal{A}_\varphi$ for $\varphi = 2x - y \leq 0$



# Top-down reformulation of the decision procedure

## Using state semantics to improve efficiency

# Top-down reformulation of the decision procedure

## Using state semantics to improve efficiency

- Given a non-atomic $\varphi$, the procedure constructs successors of $\mathcal{A}_\varphi$'s states directly, e.g.,

$$Post(3x - y \leq 2 \land x \equiv_3 1, \sigma) = Post(3x - y \leq 2, \sigma) \land Post(x \equiv_3 1, \sigma)$$

# Top-down reformulation of the decision procedure

## Using state semantics to improve efficiency

- Given a non-atomic $\varphi$, the procedure constructs successors of $\mathcal{A}_\varphi$'s states directly, e.g.,

  $$Post(3x - y \leq 2 \land x \equiv_3 1, \sigma) = Post(3x - y \leq 2, \sigma) \land Post(x \equiv_3 1, \sigma)$$

- applications: disjunction pruning, state rewriting

# Top-down reformulation of the decision procedure

## Using state semantics to improve efficiency

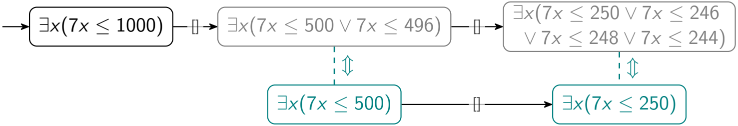
- Given a non-atomic $\varphi$, the procedure constructs successors of $\mathcal{A}_\varphi$'s states directly, e.g.,

$$Post(3x - y \leq 2 \land x \equiv_3 1, \sigma) = Post(3x - y \leq 2, \sigma) \land Post(x \equiv_3 1, \sigma)$$

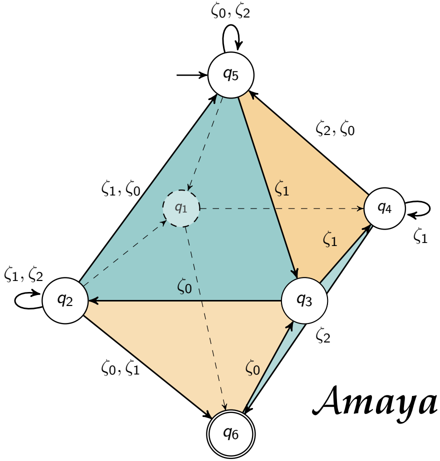
- applications: disjunction pruning, state rewriting

Disjunction pruning:

- A state $\psi_1 \lor \psi_2 \lor \cdots \lor \psi_k$ can be rewritten into an equivalent state $\psi_2 \lor \cdots \lor \psi_k$ given $\psi_2 \lor \cdots \lor \psi_k \Rightarrow \psi_1$.
- Testing $\varphi \Rightarrow \psi$ is hard, therefore, we underapproximate using structural subsumption $\preceq_s$



# Enter Amaya




- new open-source LIA SMT solver based on finite automata
- novel optimizations of the classical $\mathcal{A}$-based decision procedure
- implemented in Python and C++
  - uses the `sylvan`[1] library providing an MTBDD implementation

[1] van Dijk, T., van de Pol, J. TACAS'2015

# Performance evaluation

## Discussion of used benchmarks

Performance evaluated on 2 benchmark families:

- `SMT-COMP`: 372 arithmetic-heavy quantified formulae from SMT-COMP's `LIA` and `NIA` categories
  - from the `20190429-UltimateAutomizerSvcomp2019` and `UltimateAutomizer` directories
- `Frobenius`: 55 instances of the Frobenius coin problem for two coins

$$\forall \mathbf{n}(x \neq \mathbf{w} \cdot \mathbf{n}^\intercal) \wedge (\forall y((\forall \mathbf{m}(y \neq \mathbf{w} \cdot \mathbf{m}^\intercal)) \rightarrow y \leq x))$$

where $\mathbf{w} \in \mathbb{N}^2$ are parameters (two consequent primes)

# Performance evaluation

## Runtime ([s]) comparison with the state of the art

SMT-COMP (372)

| solver | timeouts | mean | median | std. dev. | wins | | losses | |
|---|---|---|---|---|---|---|---|---|
| AMAYA | **17** | 1.12 | 0.26 | 3.58 | | | | |
| $\text{AMAYA}_{\text{noopt}}$ | 73 | 2.32 | 0.27 | 8.16 | 232 | (56) | 113 | (0) |
| LASH | 114 | 3.04 | **0.01** | 9.94 | 178 | (98) | 178 | (1) |
| Z3 | 31 | **0.11** | **0.01** | 1.35 | 31 | (28) | 338 | (14) |
| CVC5 | 28 | 0.20 | 0.02 | 2.42 | 32 | (28) | 340 | (17) |
| PRINCESS | 50 | 4.14 | 1.14 | 9.31 | 354 | (40) | 8 | (7) |

# Performance evaluation

## Runtime ([s]) comparison with the state of the art

SMT-COMP (372)

| solver | timeouts | mean | median | std. dev. | wins | | losses | |
|---|---|---|---|---|---|---|---|---|
| Amaya | **17** | 1.12 | 0.26 | 3.58 | | | | |
| $\text{Amaya}_{\text{noopt}}$ | 73 | 2.32 | 0.27 | 8.16 | 232 | (56) | 113 | (0) |
| Lash | 114 | 3.04 | **0.01** | 9.94 | 178 | (98) | 178 | (1) |
| Z3 | 31 | **0.11** | **0.01** | 1.35 | 31 | (28) | 338 | (14) |
| cvc5 | 28 | 0.20 | 0.02 | 2.42 | 32 | (28) | 340 | (17) |
| Princess | 50 | 4.14 | 1.14 | 9.31 | 354 | (40) | 8 | (7) |

Frobenius (55)

| solver | timeouts | mean | median | std. dev. | wins | | losses | |
|---|---|---|---|---|---|---|---|---|
| Amaya | **5** | 11.79 | 3.54 | 16.03 | | | | |
| $\text{Amaya}_{\text{noopt}}$ | **5** | 11.54 | 4.06 | 14.65 | 27 | (0) | 21 | (0) |
| Lash | 9 | 15.72 | 5.74 | 20.32 | 37 | (5) | 14 | (0) |
| Z3 | 51 | 1.66 | 0.49 | 2.69 | 48 | (46) | 2 | (0) |
| cvc5 | 54 | **0.05** | **0.05** | — | 49 | (49) | 1 | (0) |
| Princess | 13 | 46.32 | 45.92 | 29.03 | 50 | ( 8) | 0 | (0) |

# Performance evaluation

## Runtime ([s]) comparison with the state of the art

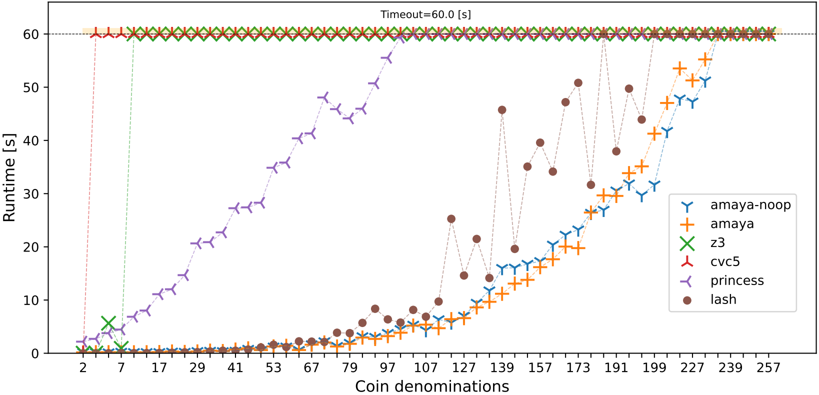
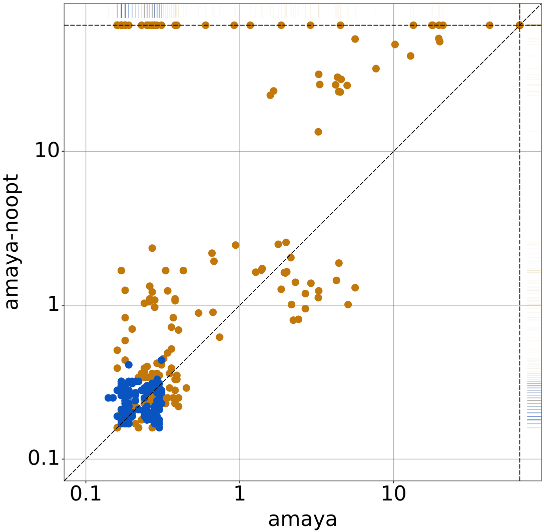



Figure: Runtime comparison on the `Frobenius` benchmark

# Performance evaluation

## Runtime ([s]) improvements over the classical constructions



# Future work, open problems

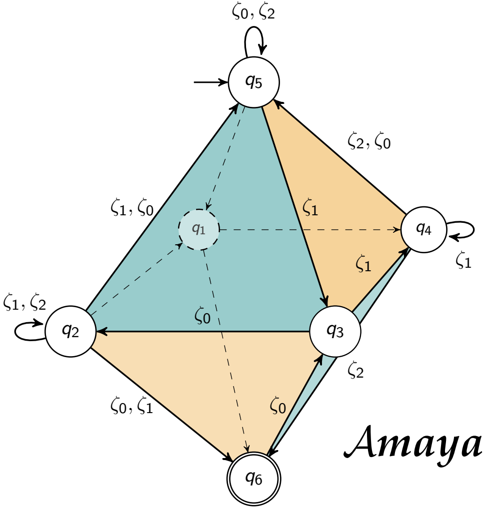
Open problems:

- combination with other SMT theories, e.g., theory of uninterpreted functions
- extending LIA with a predicate $\mathit{IsPow2}(x) \overset{def}{\Leftrightarrow} \exists k(x = 2^k)$
  - trivial, but (a good) $\mathcal{O}(\cdot)$ of the $\mathcal{A}$-based approach is unknown
- Can the duality between states and formulae be used in different theories, e.g., WS1S?

Engineering challenges:

- Parallelization based on the formula structure
- Second-order DAGification of formula

# Conclusion



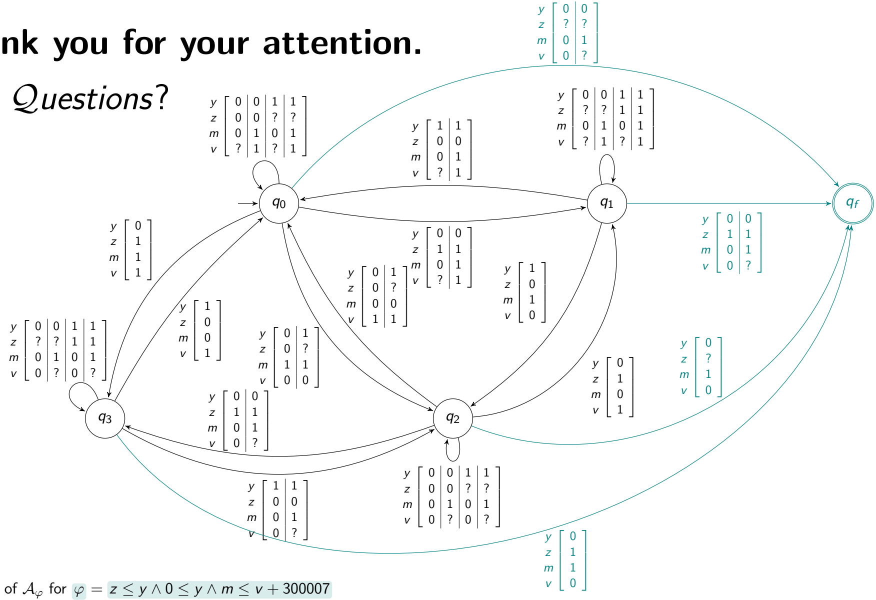

CAV Artifact Evaluation Available

CAV Artifact Evaluation Reusable

- LIA can be decided efficiently using finite automata
- $\mathcal{A}$-based approach exhibits interesting properties w.r.t. quantifiers
- automata-logic connection can be used to greatly improve the performance of the original procedure
- SMT-COMP'24 - 2nd place in NIA, 1st place in NIA(24s)

# Thank you for your attention.

*Questions?*




SCC of $\mathcal{A}_\varphi$ for $\varphi = z \leq y \wedge 0 \leq y \wedge m \leq v + 300007$

# Monotonicity-based optimizations - modulo linearization

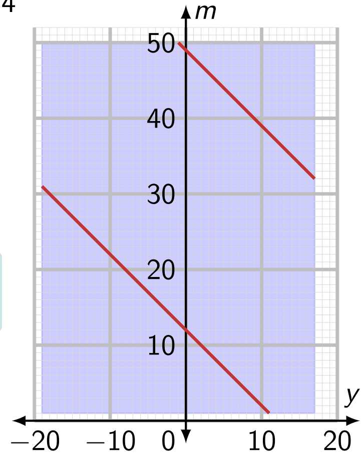
Let $\psi(\vec{x}, y, m)$ be 17-best-from-below w.r.t. $y$

- larger $y \rightsquigarrow$ more $\vec{x}$ values satisfy $\varphi(\vec{x}, m)$, but $y$ cannot be larger than 17
- $2x - y \leq 3 \land 3x - 2y \leq 3 \land 2y \leq 34$

$$\exists y, m(\psi(\vec{x}, y, m) \land y + m \equiv_{37} 12 \land 1 \leq m \leq 50)$$

$$\Updownarrow$$

$$\exists y, m(\psi \land ((y \geq -19 \land y \leq 11 \land y + m = 12) \lor (y \geq -1 \land y \leq 17 \land y + m = 49))$$



# Monotonicity-based optimizations

Let $\psi(\vec{x}, y)$ be a 42-increasing w.r.t. $y$

- $\exists y(\psi(\vec{x}, y) \wedge y \equiv_M k) \Leftrightarrow \psi(\vec{x}, c')$ where $c' = \max\{\ell \in \mathbb{Z} \mid \ell \equiv_M k, \ell \leq c\}$

$$\exists y(x - 2z \leq 3 \wedge z < y \wedge x - 13y \leq 2z \wedge y \leq 42 \wedge y \equiv_9 0)$$

$$\Updownarrow$$

$$x - 2z \leq 3 \wedge z < 36 \wedge x - 13 \cdot 36 \leq 2z$$

# Fromulae $\rightleftharpoons$ states — rewriting into equivalent formulae

A formula $\psi$ can be rewritten into an equivalent $\psi'$ whenever suitable.

$$\psi\colon \exists y, m(f_0 \leq y \land m \leq f_1 + 42 \land y \leq -1 \land m \geq 0 \land m \leq 0 \land m \equiv_7 y)$$

$$\big\downarrow\; m = 0$$

$$\psi'\colon \exists y(f_0 \leq y \land 0 \leq f_1 + 42 \land y \leq -1 \land 0 \equiv_7 y)$$

$$\big\downarrow\; y = -7$$

$$\psi''\colon f_0 \leq -7 \land 0 \leq f_1 + 42$$

And continue building the automaton using $Post(\psi'', \sigma)$.