Deciding Presburger Arithmetic Using Finite Automata

Michal Hečko, Ondřej Lengál, Vojta Havlena, Lukáš Holík

Faculty of Information Technology
Brno University of Technology

LTA23, Dec 2023

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 φ?

SMT solver

SAT + Model

$x_{low} = \ldots, x_{high} = \ldots$

UNSAT (no model)

Outline

First-order logic, theories and decision procedures

Automata-based decision procedure

Amaya - a novel implementation $\mathcal{A}$-based decision procedure

Conclusion, future work

First-order logic primer

First-order logic (FOL)

- collection of formal languages distinguished by $\langle \mathcal{F}, \mathcal{P} \rangle$

[1] recursively enumerable

First-order logic primer

First-order logic (FOL)

- collection of formal languages distinguished by $\langle \mathcal{F}, \mathcal{P} \rangle$
- allows reasoning about properties and relations between elements of the domain of discourse

[1] recursively enumerable

First-order logic primer

First-order logic (FOL)

- collection of formal languages distinguished by $\langle \mathcal{F}, \mathcal{P} \rangle$
- allows reasoning about properties and relations between elements of the domain of discourse
- a FOL theory $\mathcal{T}$ is given by its signature $\Sigma_{\mathcal{T}}$, and a set[1] of closed formulae $A_{\mathcal{T}}$ called the axioms.

[1] recursively enumerable

First-order logic primer

First-order logic (FOL)

- collection of formal languages distinguished by $\langle \mathcal{F}, \mathcal{P} \rangle$
- allows reasoning about properties and relations between elements of the domain of discourse
- a FOL theory $\mathcal{T}$ is given by its signature $\Sigma_{\mathcal{T}}$, and a set[1] of closed formulae $A_{\mathcal{T}}$ called the axioms.

$$
\begin{aligned}
t &::= x \mid f(x_1, \ldots, x_n) \\
\varphi &::= p(t_1, \ldots t_m) \mid t_1 = t_2 \mid \\
&\qquad (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \leftrightarrow \varphi) \mid (\varphi \rightarrow \varphi) \mid \\
&\qquad (\exists x \varphi) \mid (\forall x \varphi)
\end{aligned}
$$

for a variable $x \in \mathbb{X}$, a predicate symbol $p_{/m} \in \mathcal{P}$, and a function symbol $f_{/n} \in \mathcal{F}$.

[1] recursively enumerable

Presburger arithmetic (PrA)

Presburger arithmetic (PrA)

- FOL theory with the signature $\Sigma_{PrA} = \langle 0, 1, + \rangle$

Presburger arithmetic (PrA)

- FOL theory with the signature $\Sigma_{PrA} = \langle 0, 1, + \rangle$
- decidability established in 1929 by Presburger

Presburger arithmetic (PrA)

- FOL theory with the signature $\Sigma_{PrA} = \langle 0, 1, + \rangle$
- decidability established in 1929 by Presburger
  - shown by quantifier elimination

Presburger arithmetic (PrA)

- FOL theory with the signature $\Sigma_{PrA} = \langle 0, 1, + \rangle$
- decidability established in 1929 by Presburger
  - shown by quantifier elimination
- trivially extendable from $\mathbb{N}$ to $\mathbb{Z}$

Presburger arithmetic (PrA)

- FOL theory with the signature $\Sigma_{PrA} = \langle 0, 1, + \rangle$
- decidability established in 1929 by Presburger
  - shown by quantifier elimination
- trivially extendable from $\mathbb{N}$ to $\mathbb{Z}$
  - PrA nowadays refers to $Th(\mathbb{Z}, 0, 1, +)$
  - also known as linear integer arithmetic (LIA)

Decision procedures, SMT and SMT solvers

Decision procedure $\mathcal{P}(\varphi)$ for a theory $\mathcal{T}$:

- an algorithm that returns SAT when there is a solution (model) to φ, UNSAT otherwise

Decision procedures, SMT and SMT solvers

Decision procedure $\mathcal{P}(\varphi)$ for a theory $\mathcal{T}$:

- an algorithm that returns SAT when there is a solution (model) to φ, UNSAT otherwise

SMT solver:

- a tool implementing decision procedures for theories $\mathcal{T}_1$, $\mathcal{T}_2, \ldots$ and their combinations

Decision procedures, SMT and SMT solvers

Decision procedure $\mathcal{P}(\varphi)$ for a theory $\mathcal{T}$:

- an algorithm that returns SAT when there is a solution (model) to φ, UNSAT otherwise

SMT solver:

- a tool implementing decision procedures for theories $\mathcal{T}_1$, $\mathcal{T}_2, \ldots$ and their combinations
- typically a set of dedicated theory solvers combined using the Nelson-Oppen approach

Decision procedures, SMT and SMT solvers

Decision procedure $\mathcal{P}(\varphi)$ for a theory $\mathcal{T}$:

- an algorithm that returns SAT when there is a solution (model) to φ, UNSAT otherwise

SMT solver:

- a tool implementing decision procedures for theories $\mathcal{T}_1$, $\mathcal{T}_2, \ldots$ and their combinations
- typically a set of dedicated theory solvers combined using the Nelson-Oppen approach
- quantifiers handled using quantifier instantiation

Decision procedures, SMT and SMT solvers

Decision procedure $\mathcal{P}(\varphi)$ for a theory $\mathcal{T}$:

- an algorithm that returns SAT when there is a solution (model) to φ, UNSAT otherwise

SMT solver:

- a tool implementing decision procedures for theories $\mathcal{T}_1$, $\mathcal{T}_2, \ldots$ and their combinations
- typically a set of dedicated theory solvers combined using the Nelson-Oppen approach
- quantifiers handled using quantifier instantiation
- actively employed in the industry, e.g., at AWS
- standardized input format SMTLIB

SMTLIB example

```
(set-logic LIA)
(declare-fun P () Int)
(assert
  (and
    (<= 0 P)
    (forall ((x0 Int) (x1 Int))
      (=>
        (and (<= 0 x0) (<= 0 x1))
        (not (= (+ (* x0 13) (* x1 17)) P))))
    (forall ((R Int))
      (=>
          (forall ((x0 Int) (x1 Int))
            (=>
              (and (<= 0 x0) (<= 0 x1))
              (not (= (+ (* x0 13) (* x1 17)) R))))
        (<= R P)))))
(check-sat)
```

Outline

First-order logic, theories and decision procedures

Automata-based decision procedure

Amaya - a novel implementation $\mathcal{A}$-based decision procedure

Conclusion, future work

Logic-automata connection

$$\mathcal{A} \longleftrightarrow \varphi$$

- first $\mathcal{A}$-based decision procedure is due to Büchi (1960) [2]

Logic-automata connection

$$\mathcal{A} \longleftrightarrow \varphi$$

- first $\mathcal{A}$-based decision procedure is due to Büchi (1960) [2]
- Büchi developed the approach to show decidability of WS1S

Logic-automata connection

$$\boxed{\mathcal{A}} \longleftrightarrow \boxed{\varphi}$$

- first $\mathcal{A}$-based decision procedure is due to Büchi (1960) [2]
- Büchi developed the approach to show decidability of WS1S
- a direct construction $\mathcal{P}_{\mathcal{A}}$ for Presburger arithmetic given by Boudet & Comon (1996) [1]

Logic-automata connection

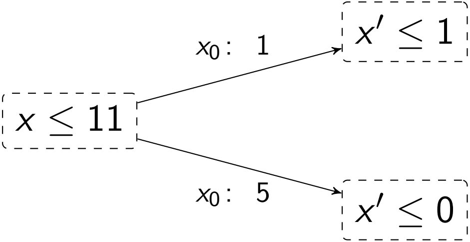
$$\mathcal{A} \longleftrightarrow \varphi$$

- first $\mathcal{A}$-based decision procedure is due to Büchi (1960) [2]
- Büchi developed the approach to show decidability of WS1S
- a direct construction $\mathcal{P}_{\mathcal{A}}$ for Presburger arithmetic given by Boudet & Comon (1996) [1]
- the time-complexity $\mathcal{O}(2^{2^{2^n}})$ of $\mathcal{P}_{\mathcal{A}}$ established by Durand-Gasselin & Habermehl (2010) [3]

Constructing NFAs from atomic formulae - intuition

Idea: Any number x can be written as its least-significant digit x_0 and remaining digits x'.

$$x = x_0 + 10x'$$

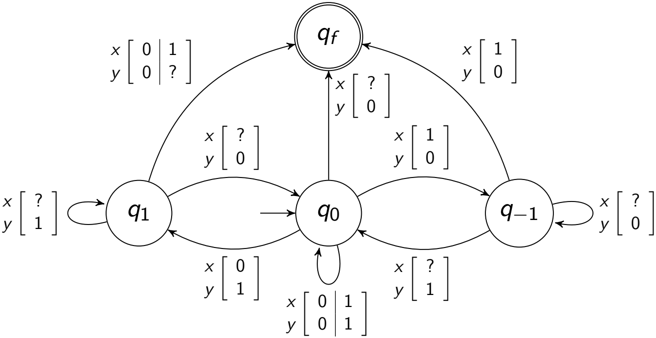


$\varphi_{atom} \mapsto \mathcal{A}$ – binary encoding, coefficients

The previous observation is flexible:

- number encoding (basis) is arbitrary: $\Sigma = \{0,1\}^n$ has advantages (BDDs)
- variables can have coefficients

Automaton $\mathcal{A}_\varphi$ for φ: $x - 2y \leq 0$

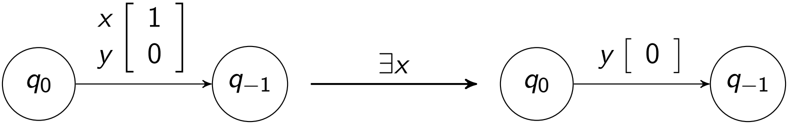


Beyond atomic formulae

The automaton $\mathcal{A}_\psi$ for a formula ψ is created inductively by mapping logical connectives to corresponding language operations:

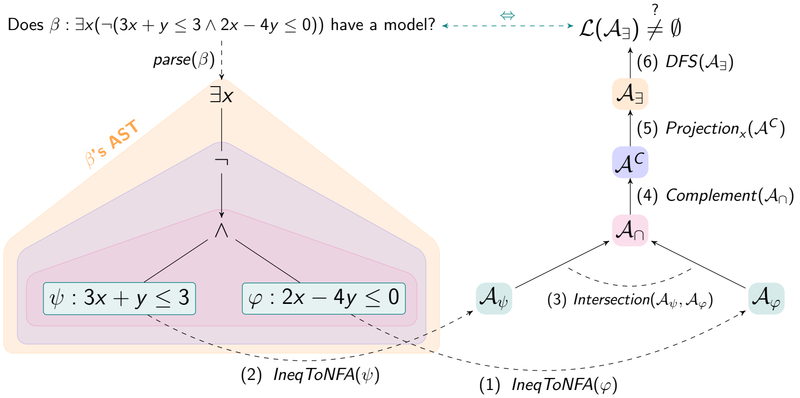
- $\mathcal{A}_{\varphi \wedge \varphi'} = \mathcal{A}_\varphi \cap \mathcal{A}_{\varphi'}$
- $\mathcal{A}_{\varphi \vee \varphi'} = \mathcal{A}_\varphi \cup \mathcal{A}_{\varphi'}$
- $\mathcal{A}_{\neg\varphi} = \mathcal{A}_\varphi^C$

Existential quantification $\exists x(\varphi)$ corresponds to projecting away the track corresponding to variable x [2].



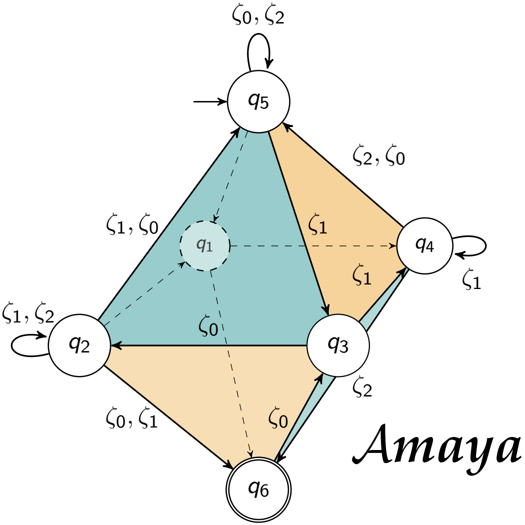
[2] Omitting technical details about padding

High-level example of $\mathcal{A}$-based procedure



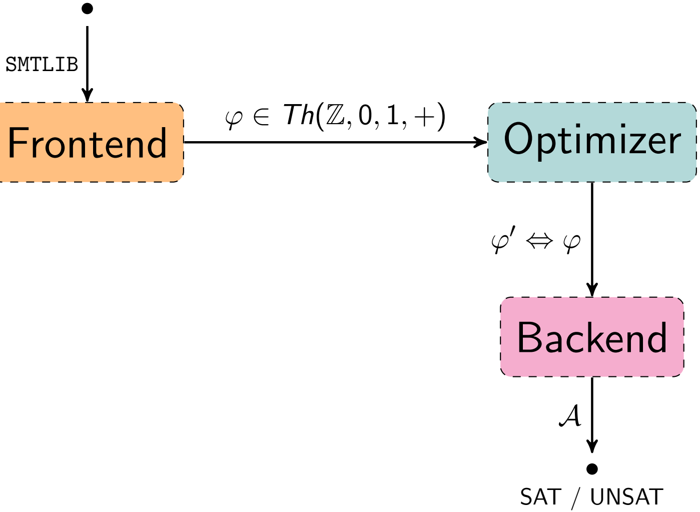
Outline

Enter Amaya

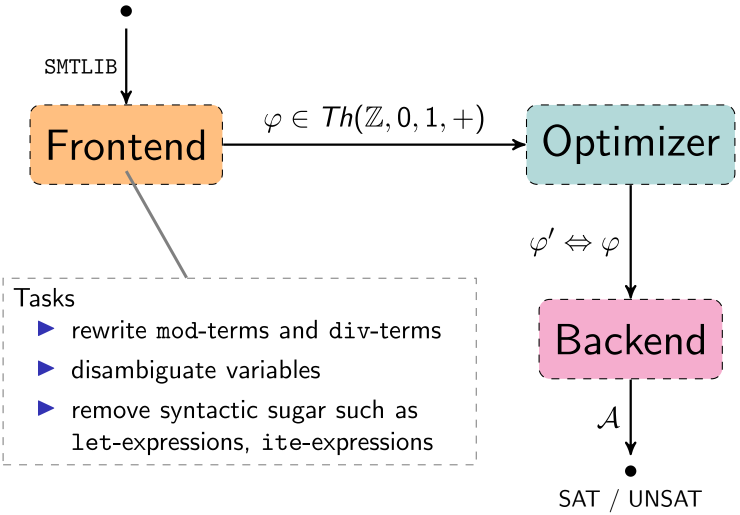


- LIA SMT solver based on finite automata
- novel optimizations of the classical $\mathcal{A}$-based decision procedure
- implementation:
  Python (7.7 KLOC),
  C++ (8.3 KLOC)

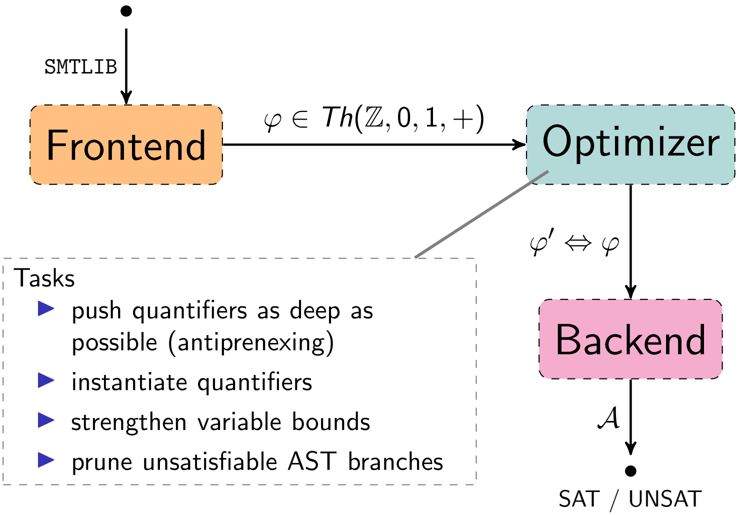
Amaya: an interpreter's perspective



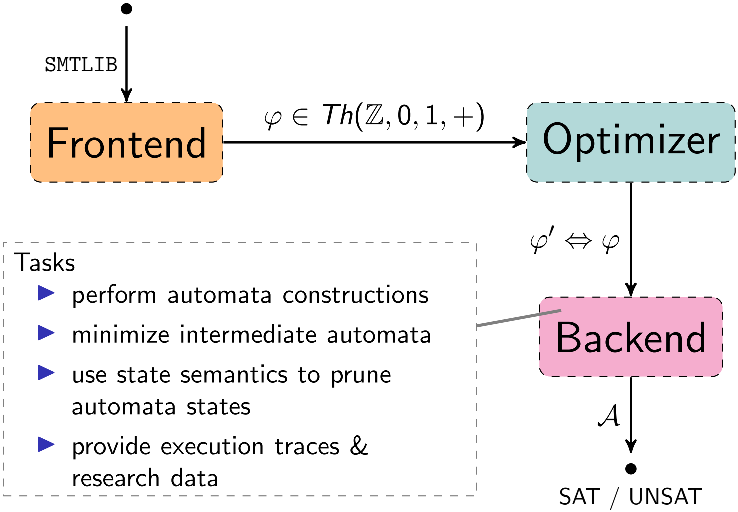
Amaya: an interpreter's perspective



Amaya: an interpreter's perspective



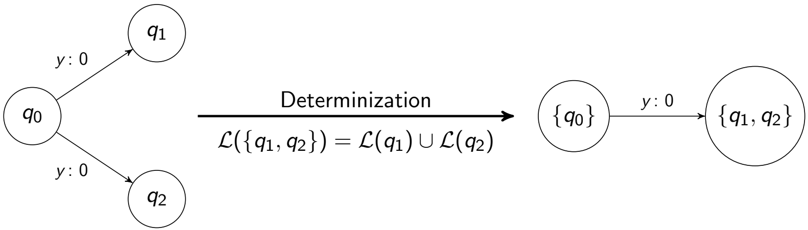
Amaya: an interpreter's perspective



Tasks

- perform automata constructions
- minimize intermediate automata
- use state semantics to prune automata states
- provide execution traces & research data

Subset construction primer



Backend - addressing performance bottlenecks

Time complexity of many automata constructions is linear in $|\Sigma|$:

...
for each $\sigma \in \Sigma$ **do**
    ...
end for
...

Backend - addressing performance bottlenecks

Time complexity of many automata constructions is linear in $|\Sigma|$:

...
for each $\sigma \in \Sigma$ **do**
    ...
end for
...

$\Sigma = \{0, 1\}^n$ where n is the number of variables.

Backend - addressing performance bottlenecks

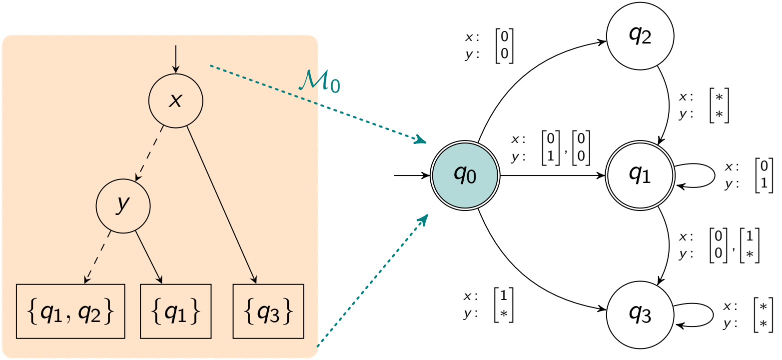
Time complexity of many automata constructions is linear in $|\Sigma|$:

...
for each $\sigma \in \Sigma$ **do**
    ...
end for
...

$\Sigma = \{0, 1\}^n$ where n is the number of variables.

- $|\Sigma|$ grows exponentially with n

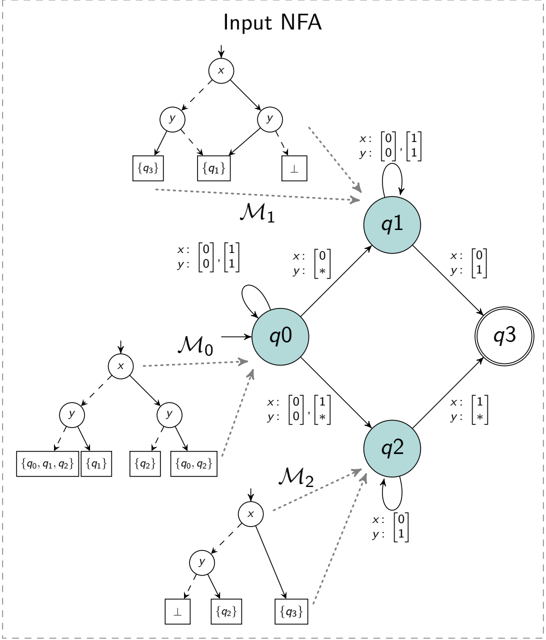
MTBDDs - symbolic representation of automata

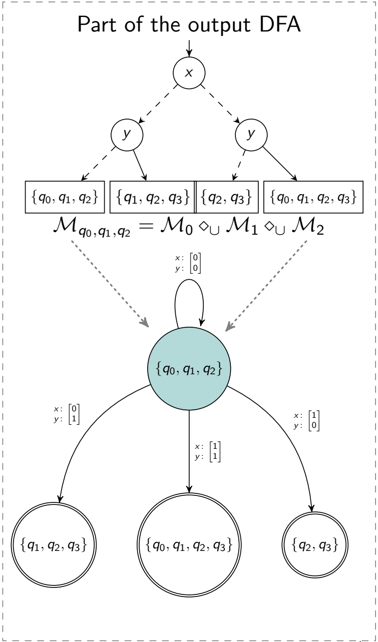


Binary-decision diagram representing transitions from q_0

Amaya relies on the Sylvan library [4] to provide an MTBDD implementation.

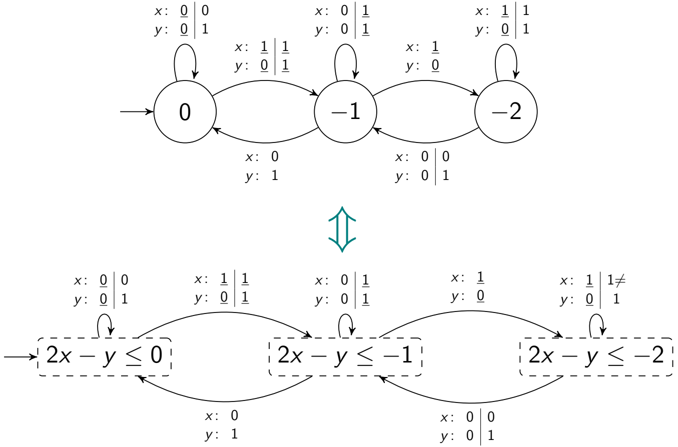
MTBDD-based automata constructions





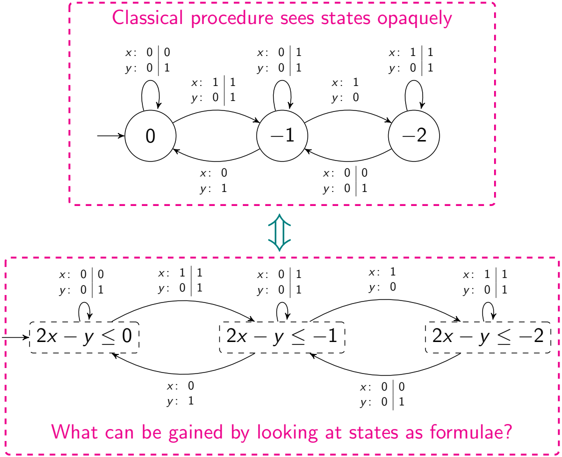
Duality between formulae and states

TFA $\mathcal{A}_\varphi$ for φ: $2x - y \leq 0$



Duality between formulae and states

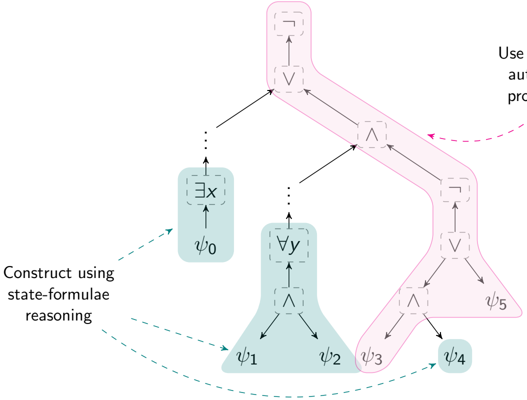
TFA $\mathcal{A}_\varphi$ for φ: $2x - y \leq 0$



A birth of a framework

- Having $Post(\varphi, \sigma)$ where $\sigma \in \Sigma$ and φ is an atomic predicate allows for an inductive definition of $Post(\psi, \sigma)$ for arbitrary ψ e.g.

$$Post(2x - y \leq 3 \wedge y + 2z \leq 42) =$$
$$Post(2x - y \leq 3) \wedge Post(y + 2z \leq 42)$$



Framework - structural subsumption

A state $\psi_1 \vee \psi_2 \vee \psi_3 \ldots \psi_k$ can be rewritten into an equivalent state $\psi_2 \vee \psi_3 \ldots \psi_k$ given $\psi_2 \vee \psi_3 \ldots \psi_k \Rightarrow \psi_1$.

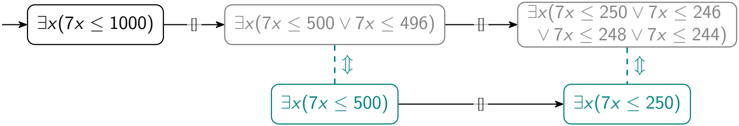
- Testing $\varphi \Rightarrow \psi$ is hard, therefore, we underapproximate using structural subsumption $\preceq_s$

$$\vec{a}_1 \cdot \vec{x}_1 \leq c_1 \preceq_s \vec{a}_2 \cdot \vec{x}_2 \leq c_2 \overset{def}{\Leftrightarrow} \vec{a}_1 = \vec{a}_2 \wedge \vec{x}_1 = \vec{x}_2 \wedge c_1 \leq c_2$$

$$\vec{a}_1 \cdot \vec{x}_1 = c_1 \preceq_s \vec{a}_2 \cdot \vec{x}_2 = c_2 \overset{def}{\Leftrightarrow} \vec{a}_1 = \vec{a}_2 \wedge \vec{x}_1 = \vec{x}_2 \wedge c_1 = c_2$$

$$\vec{a}_1 \cdot \vec{x}_1 \equiv_{m_1} c_1 \preceq_s \vec{a}_2 \cdot \vec{x}_2 \equiv_{m_2} c_2 \overset{def}{\Leftrightarrow} \vec{a}_1 = \vec{a}_2 \wedge \vec{x}_1 = \vec{x}_2 \wedge c_1 = c_2 \wedge m_1 = m_2$$

(Can be extended to arbitrary ψ)



Framework - rewriting into equivalent formulae

A formula ψ can be rewritten into an equivalent ψ' whenever suitable.

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

$$\Big\downarrow\, m = 0$$

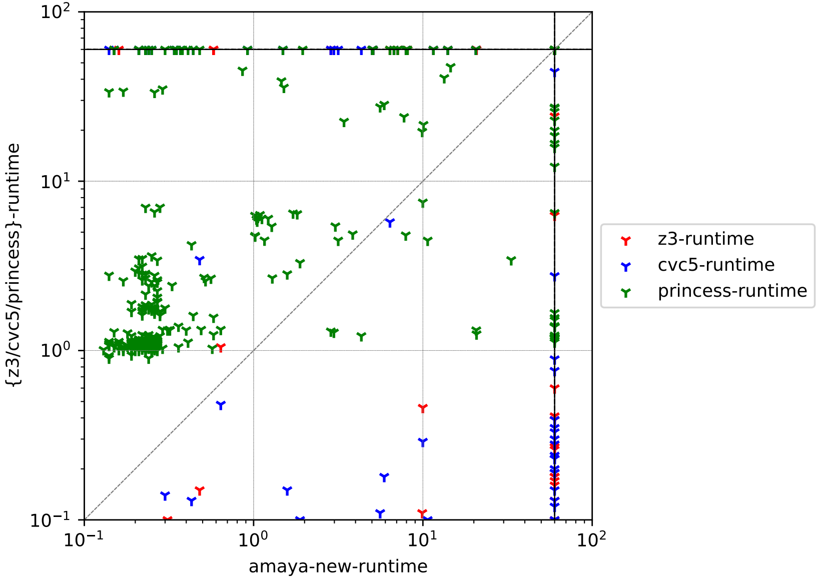
$$\psi'\colon \exists y(f_0 \leq y \land 0 \leq f_1 + 42 \land y \leq -1 \land 0 \equiv_7 y)$$

$$\Big\downarrow\, y = -7$$

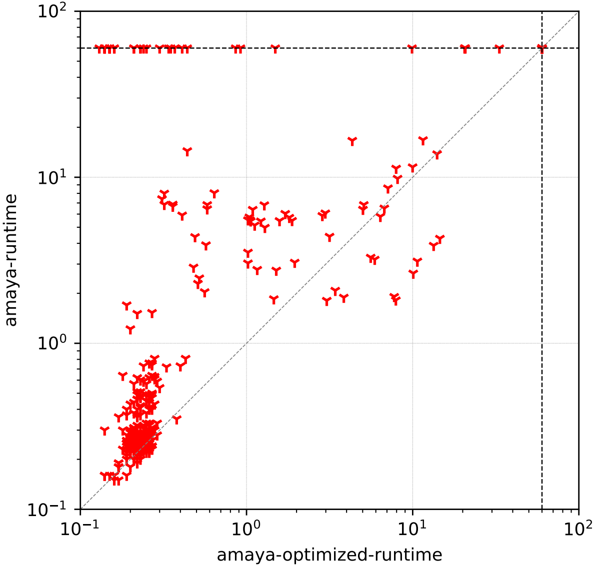
$$\psi''\colon f_0 \leq -7 \land 0 \leq f_1 + 42$$

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

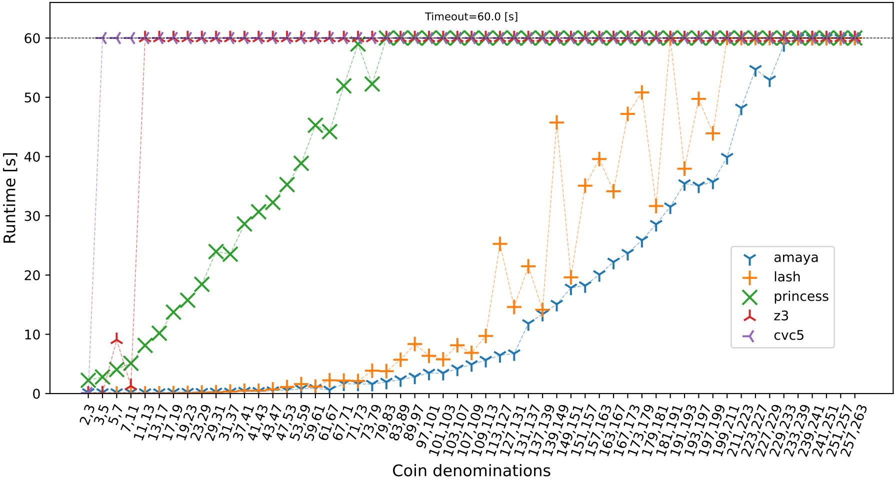
Performance evaluation - state-of-the-art SMT solvers



Performance evaluation - classical $\mathcal{P}_{\mathcal{A}}$



Performance evaluation - Frobenius coin problem



Outline

First-order logic, theories and decision procedures

Automata-based decision procedure

Amaya - a novel implementation $\mathcal{A}$-based decision procedure

Conclusion, future work

Future work, open problems

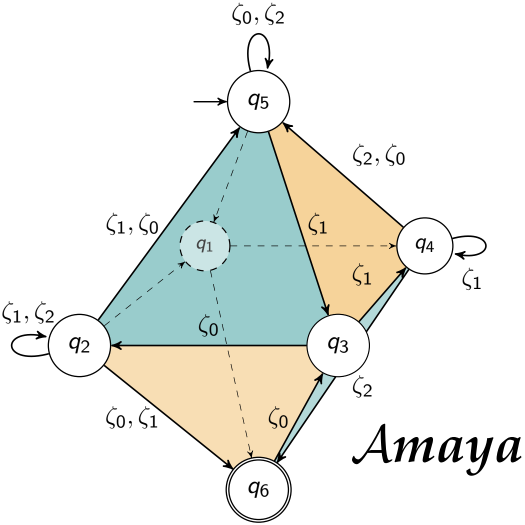
Open problems:

- combination with other SMT theories, e.g., theory of uninterpreted functions
  - $\rightsquigarrow$ existential second-order theory over automatic structures
  - decidability of combinations $\times$ practical usefulness
- extending PrA with a predicate $\mathit{IsPow2}(x) \overset{\mathit{def}}{\Leftrightarrow} \exists k(x = 2^k)$
  - trivial, but $\mathcal{O}(\cdot)$ is unknown
- Can the duality between states and formulae be used in different theories, e.g., WS1S?

Engineering challenges:

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

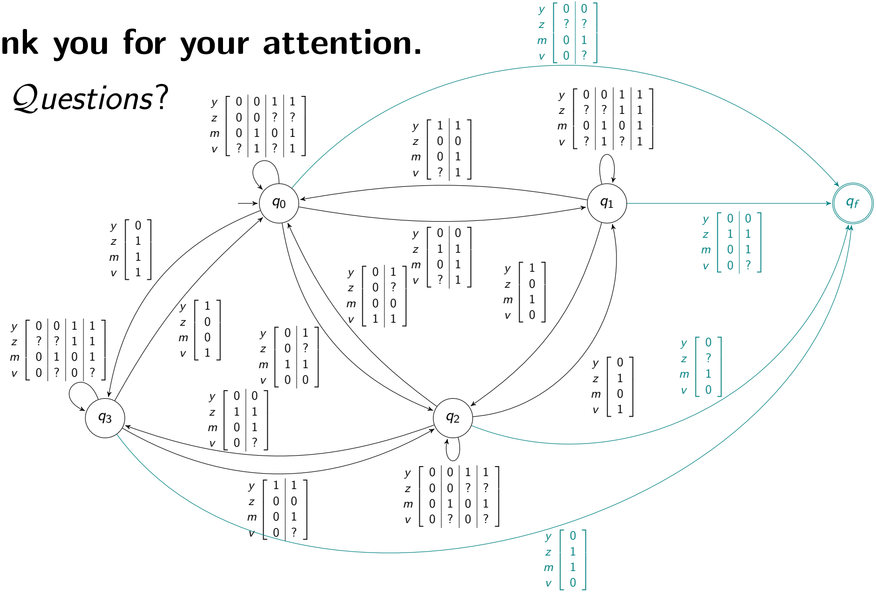
Conclusion



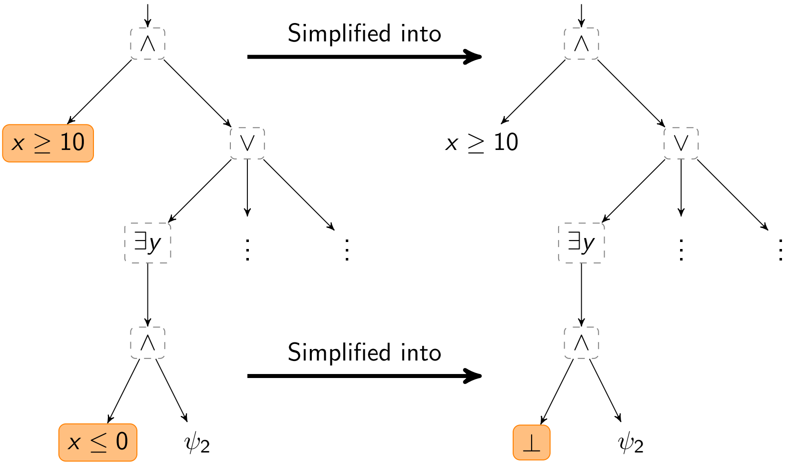
- PrA can be decided using finite automata
- $\mathcal{A}$-based approach exhibits interesting properties wrt. quantifiers
- automata-logic connection can be used to improve the performance of $\mathcal{P}_{\mathcal{A}}(\varphi)$

Thank you for your attention.

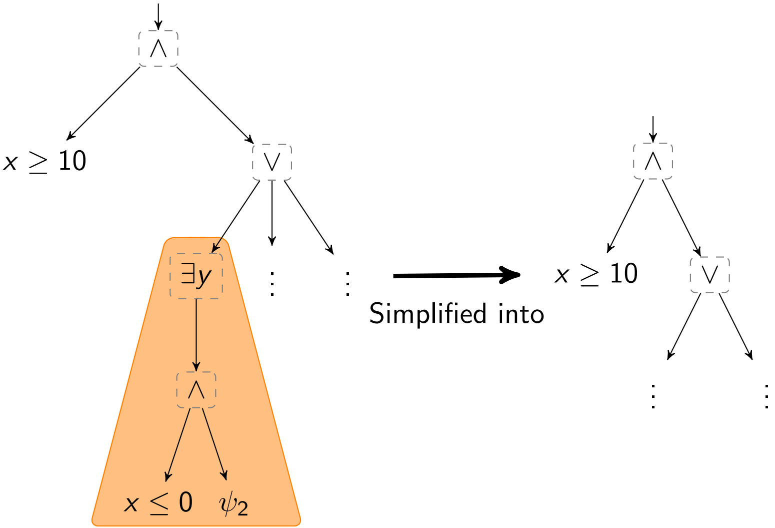
Questions?



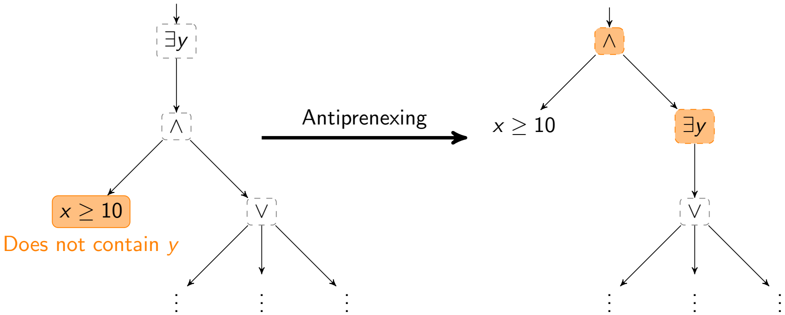
Optimizer: formula pruning, bound strengthening



Optimizer: formula pruning, bound strengthening



Optimizer: antiprenexing



Formula monotonicity

A formula $\psi(\vec{x}, y)$ is c-increasing w.r.t. y where $c \in \mathbb{Z} \cup \{\pm\infty\}$ iff

1. $[\![\psi(\vec{x}, y_1)]\!] \subseteq [\![\psi(\vec{x}, y_2)]\!]$ for all $y_1 \leq y_2 \leq c$ and
2. $[\![\psi(\vec{x}, y)]\!] = \emptyset$ for all $y > c$.

For example, $\psi(x, z, y)$ is 42-increasing w.r.t. y:

$$\psi \colon x - 2z \leq 3 \wedge z < y \wedge x - 13y \leq 2z \wedge y \leq 42$$

Monotonicity-based optimizations

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

- $\exists y(\psi(\vec{x}, y)) \Leftrightarrow \psi(\vec{x}, 42)$

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

$$\Updownarrow$$

$$x - 2z \leq 3 \wedge z < 42 \wedge x - 13 \cdot 42 \leq 2z$$

Monotonicity-based optimizations

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

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

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

$$\Updownarrow$$

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

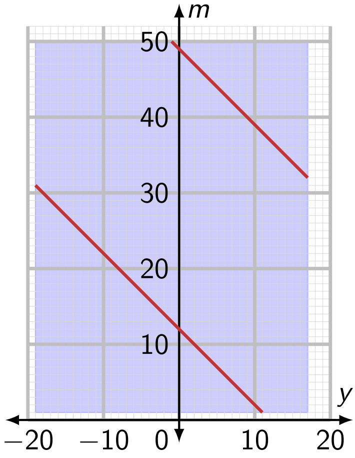
Monotonicity-based optimizations - modulo linearization

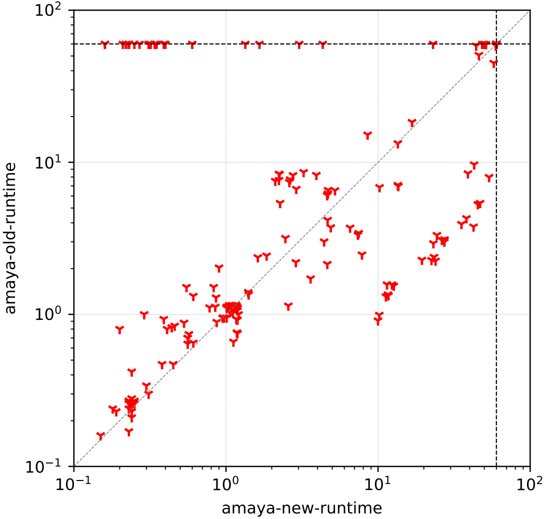
Let $\psi(\vec{x}, y)$ be a 17-increasing w.r.t. y

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

$\Updownarrow$

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





10^2
10^1
10^0
10^{-1}
amaya-old-runtime
10^{-1}
10^0
10^1
10^2
amaya-new-runtime

Literature

[1] Alexandre Boudet and Hubert Comon.
Diophantine equations, presburger arithmetic and finite automata.
In Hélène Kirchner, editor, *Trees in Algebra and Programming — CAAP '96*, pages 30–43, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[2] J Richard Büchi.
Weak second-order arithmetic and finite automata.
Mathematical Logic Quarterly, 6(1-6), 1960.

[3] Antoine Durand-Gasselin and Peter Habermehl.
On the use of non-deterministic automata for presburger arithmetic.
In Paul Gastin and François Laroussinie, editors, *CONCUR 2010 - Concurrency Theory*, pages 373–387, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[4] Tom van Dijk and Jaco van de Pol.
Sylvan: Multi-core decision diagrams.
In Christel Baier and Cesare Tinelli, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 677–691, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.