Parameter Synthesis for One-Counter Automata

Guillermo A. Pérez
joint work with Ritam Raha

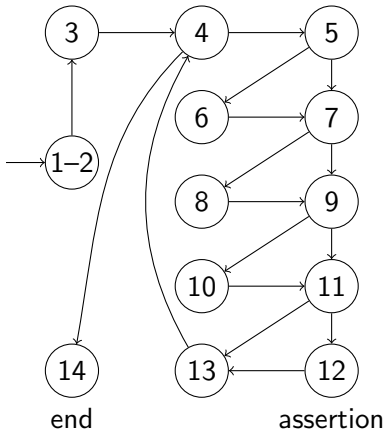
April 2021

Static reachability analysis via the control flow graph

```
1  x = 0
2  i = 0
3  i += x
4  while i >= 0:
5      if i == 0:
6          print("Hello world!")
7      if i == 1:
8          print("Lockdown + babies = pain")
9      if i == 2:
10         print("Please, help me!")
11     if i >= 3:
12         assert(False)
13     i -= 1
14 # end program
```

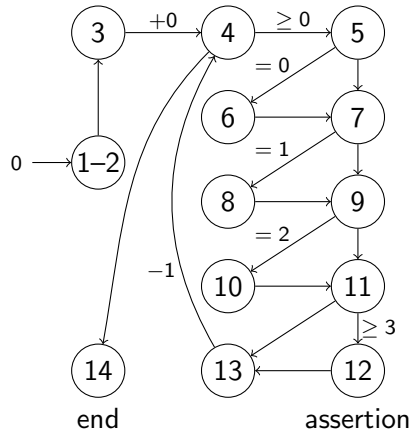
Static reachability analysis via the control flow graph

```
1  x = 0
2  i = 0
3  i += x
4  while i >= 0:
5      if i == 0:
6          print("Hello world!")
7      if i == 1:
8          print("Lockdown + babies = pain")
9      if i == 2:
10         print("Please, help me!")
11     if i >= 3:
12         assert(False)
13     i -= 1
14 # end program
```



Extending the CFG with a counter

```
1  x = 0
2  i = 0
3  i += x
4  while i >= 0:
5      if i == 0:
6          print("Hello world!")
7      if i == 1:
8          print("Lockdown + babies = pain")
9      if i == 2:
10         print("Please, help me!")
11     if i >= 3:
12         assert(False)
13     i -= 1
14 # end program
```

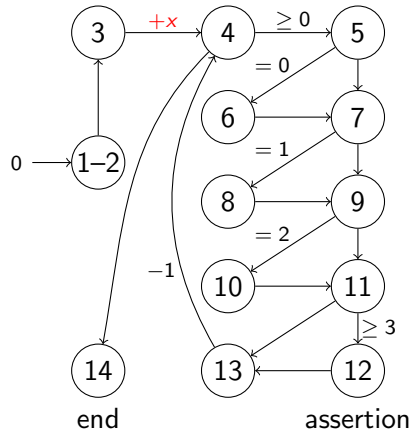


Parametric one-counter automata

```
def funprint(x):
    i = 0
    i += x
    while i >= 0:
        if i == 0:
            print("Hello world!")
        if i == 1:
            print("... babies = pain")
        if i == 2:
            print("Please, help me!")
        if i >= 3:
            assert(False)
        i -= 1
    # end program
```

Parametric one-counter automata

```
def funprint(x):
    i = 0
    i += x
    while i >= 0:
        if i == 0:
            print("Hello world!")
        if i == 1:
            print("... babies = pain")
        if i == 2:
            print("Please, help me!")
        if i >= 3:
            assert(False)
        i -= 1
    # end program
```

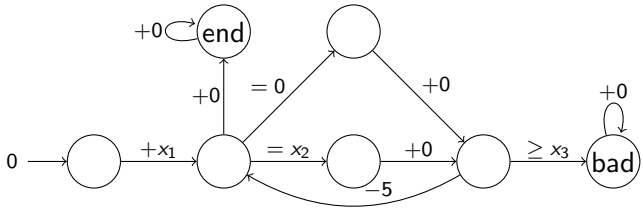


Outline

Parametric One-Counter Automata

Natural-valued parameters

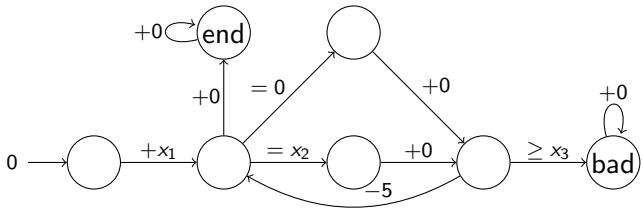
$X = \{x_1, \ldots, x_n\}$



Parametric One-Counter Automata

Natural-valued parameters

$X = \{x_1, \ldots, x_n\}$



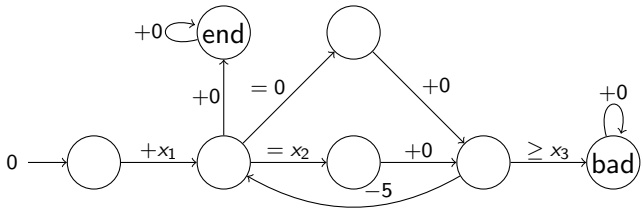
Definition (Succinct OCA with Parameters)

$\mathcal{A} = (Q, q_0, T, \delta, X)$ where $\delta : T \to Op$ with Op the union of

Parametric One-Counter Automata

Natural-valued parameters

$X = \{x_1, \ldots, x_n\}$



Definition (Succinct OCA with Parameters)

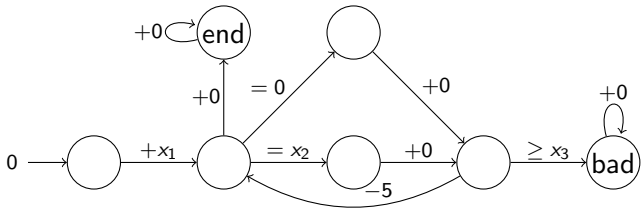
$\mathcal{A} = (Q, q_0, T, \delta, X)$ where $\delta : T \to Op$ with Op the union of

- $CU := \{+a : a \in \mathbb{Z}\}$, $CT := \{= a, \geq a : a \in \mathbb{N}\}$

Parametric One-Counter Automata

Natural-valued parameters

$X = \{x_1, \ldots, x_n\}$



Definition (Succinct OCA with Parameters)

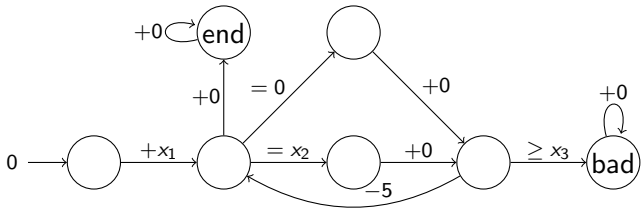
$\mathcal{A} = (Q, q_0, T, \delta, X)$ where $\delta : T \to Op$ with Op the union of

- $CU := \{+a : a \in \mathbb{Z}\}$, $CT := \{= a, \geq a : a \in \mathbb{N}\}$
- $PU := \{+x, -x : x \in X\}$, $PT := \{= x, \geq x : x \in X\}$

Parametric One-Counter Automata

Natural-valued parameters

$X = \{x_1, \ldots, x_n\}$



Definition (Succinct OCA with Parameters)

$\mathcal{A} = (Q, q_0, T, \delta, X)$ where $\delta : T \to Op$ with Op the union of

- $CU := \{+a : a \in \mathbb{Z}\}$, $CT := \{= a, \geq a : a \in \mathbb{N}\}$
- $PU := \{+x, -x : x \in X\}$, $PT := \{= x, \geq x : x \in X\}$

! Runs are restricted to keeping the counter value non-negative

Synthesis problems

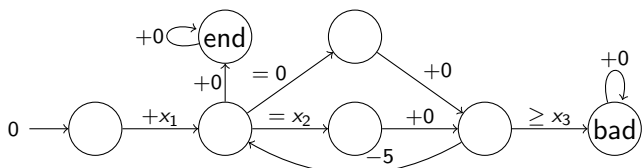
Definition (Parameter-value synthesis)

Is there some valuation $V : X \to \mathbb{N}$ such that all (infinite) runs of $\mathcal{A}$ satisfy a given ω-regular property?

Synthesis problems

Definition (Parameter-value synthesis)

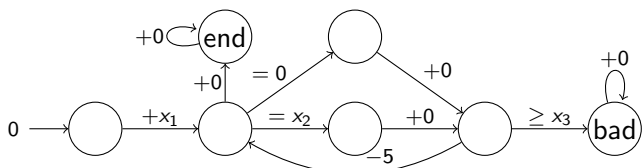
Is there some valuation $V : X \to \mathbb{N}$ such that all (infinite) runs of $\mathcal{A}$ satisfy a given ω-regular property?



Synthesis problems

Definition (Parameter-value synthesis)

Is there some valuation $V : X \to \mathbb{N}$ such that all (infinite) runs of $\mathcal{A}$ satisfy a given ω-regular property?



	LTL	Reachability	Safety	Büchi	coBüchi
Lower bound	PSPACE-hard	coNP-hard	— NP^{NP}-hard —		
Upper bound	in N3EXP	— in N2EXP —			

Outline

Presburger arithmetic with divisibility (PAD)

- Presburger arithmetic (PA) $= \mathrm{FO}(\mathbb{Z}, 0, 1, +, <)$

Presburger arithmetic with divisibility (PAD)

- Presburger arithmetic (PA) $= \mathrm{FO}(\mathbb{Z}, 0, 1, +, <)$
- Presburger arithmetic with divisibility (PAD) $=$ PA $+ \mid$
  $(a \mid b \iff \exists c \in \mathbb{Z} : b = ac)$

Presburger arithmetic with divisibility (PAD)

- Presburger arithmetic (PA) $= \mathrm{FO}(\mathbb{Z}, 0, 1, +, <)$
- Presburger arithmetic with divisibility (PAD) $=$ PA $+ \mid$
  $(a \mid b \iff \exists c \in \mathbb{Z} : b = ac)$

Theorem (Robinson'49, Lipshitz'81)

Full PAD is undecidable; one alternation suffices for undecidability.

Presburger arithmetic with divisibility (PAD)

- Presburger arithmetic (PA) $= \mathrm{FO}(\mathbb{Z}, 0, 1, +, <)$
- Presburger arithmetic with divisibility (PAD) $=$ PA $+ \mid$
  $(a \mid b \iff \exists c \in \mathbb{Z} : b = ac)$

Theorem (Robinson'49, Lipshitz'81)

Full PAD is undecidable; one alternation suffices for undecidability.

Theorem (Lipshitz'78, Lechner-Ouaknine-Worrell'15)

The existential fragment of PAD (EPAD) is decidable and in NEXP.

A restricted one-alternation fragment: $\forall\exists_R$PAD

- $\forall\exists_R\mathrm{PAD} = \forall z_1 \ldots \forall z_n \exists x_1 \ldots \exists x_m . \varphi(\boldsymbol{x}, \boldsymbol{z})$
  - in φ, divisibilities of the form $g(\boldsymbol{z}) \mid h(\boldsymbol{x}, \boldsymbol{z})$

A restricted one-alternation fragment: $\forall\exists_R$PAD

- $\forall\exists_R\text{PAD} = \forall z_1 \ldots \forall z_n \exists x_1 \ldots \exists x_m . \varphi(\boldsymbol{x}, \boldsymbol{z})$
  - in φ, divisibilities of the form $g(\boldsymbol{z}) \mid h(\boldsymbol{x}, \boldsymbol{z})$
- $\forall\exists_R\text{PAD}^+ = \forall\exists_R\text{PAD}$ with $\neg$ not allowed before divisibility
  - A negation normal form where $\mid$ cannot be negated

A restricted one-alternation fragment: $\forall\exists_R$PAD

- $\forall\exists_R\text{PAD} = \forall z_1 \ldots \forall z_n \exists x_1 \ldots \exists x_m .\, \varphi(\boldsymbol{x}, \boldsymbol{z})$
  - in φ, divisibilities of the form $g(\boldsymbol{z}) \mid h(\boldsymbol{x}, \boldsymbol{z})$
- $\forall\exists_R\text{PAD}^+ = \forall\exists_R\text{PAD}$ with $\neg$ not allowed before divisibility
  - A negation normal form where $\mid$ cannot be negated

Claim (Bozga-Iosif'05, Lechner'15)

The synthesis problems for SOCAP are decidable via an encoding into $\forall\exists_R\text{PAD}^+$.

Outline

Encoding the safety synthesis problem

Safety synthesis

Is there some valuation V such that all runs of $\mathcal{A}$ do not reach t?

Encoding the safety synthesis problem

Safety synthesis

Is there some valuation V such that all runs of $\mathcal{A}$ do not reach t?

Encoding reachability [Lechner'15]

- PAD formula $\Phi(\boldsymbol{x}) = \exists \boldsymbol{f} \bigvee_i \bigwedge_j g_j(\boldsymbol{x}) \mid h_j(\boldsymbol{f}, \boldsymbol{x}) \wedge \varphi_i(\boldsymbol{x}) \wedge \boldsymbol{f}, \boldsymbol{x} \geq 0$
- s.t. safety holds iff $\forall \boldsymbol{x}.\Phi(\boldsymbol{x})$ is false

Encoding the safety synthesis problem

Safety synthesis

Is there some valuation V such that all runs of $\mathcal{A}$ do not reach t?

Encoding reachability [Lechner'15]

- PAD formula $\Phi(\boldsymbol{x}) = \exists \boldsymbol{f} \bigvee_i \bigwedge_j g_j(\boldsymbol{x}) \mid h_j(\boldsymbol{f}, \boldsymbol{x}) \wedge \varphi_i(\boldsymbol{x}) \wedge \boldsymbol{f}, \boldsymbol{x} \geq 0$
- s.t. safety holds iff $\forall \boldsymbol{x}.\Phi(\boldsymbol{x})$ is false

Constructing the formula [Haase et al.'09]

Boolean combination of sub-formulas for 3 subcases: reach t via a run that

- has no positive cycles
- ...

$$\exists \boldsymbol{f} \bigvee_i \varphi^{(i)}_{\text{flow}}(\boldsymbol{f}) \wedge \varphi^{(i)}_{\text{weight}}(\boldsymbol{f}, \boldsymbol{x}) \wedge \varphi^{(i)}_{\text{nopos}}(\boldsymbol{x})$$

The flow sub-formula

$$\exists \boldsymbol{f} \bigvee_i \varphi^{(i)}_{\text{flow}}(\boldsymbol{f}) \wedge \varphi^{(i)}_{\text{weight}}(\boldsymbol{f}, \boldsymbol{x}) \wedge \varphi^{(i)}_{\text{nopos}}(\boldsymbol{x})$$

Path formulas

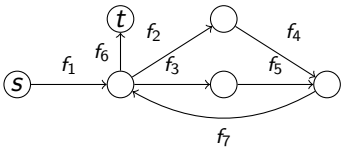
V should satisfy φ_{flow} iff $\mathcal{A}$ has a path reaching t from s

The flow sub-formula

$$\exists \boldsymbol{f} \bigvee_i \varphi^{(i)}_{\text{flow}}(\boldsymbol{f}) \wedge \varphi^{(i)}_{\text{weight}}(\boldsymbol{f}, \boldsymbol{x}) \wedge \varphi^{(i)}_{\text{nopos}}(\boldsymbol{x})$$

Path formulas

V should satisfy φ_{flow} iff $\mathcal{A}$ has a path reaching t from s



$$\bigwedge_{q \notin \{s,t\}} \left(\sum_{p \in T(\cdot,q)} f(p,q) = \sum_{r \in T(q,\cdot)} f(q,r) \right) \quad (1)$$

Theorem (Euler's theorem for digraphs)

There is an s–t path iff there is a valuation of the f_i such that

- *the subgraph induced by edges with nonzero flow and $\{(t,s)\}$ is strongly connected,*
- *it satisfies (1) and $\sum_{p \in T(\cdot,s)} f(p,s) - \sum_{r \in T(s,\cdot)} f(s,r) = 1$.*

The flow sub-formula

$$\exists \boldsymbol{f} \bigvee_i \varphi^{(i)}_{\text{flow}}(\boldsymbol{f}) \wedge \varphi^{(i)}_{\text{weight}}(\boldsymbol{f}, \boldsymbol{x}) \wedge \varphi^{(i)}_{\text{nopos}}(\boldsymbol{x})$$

A formula per subgraph

Each $\varphi^{(i)}_{\text{flow}}(\boldsymbol{f})$ captures the flow constraints over one subgraph that satisfies the support condition.

The flow sub-formula

$$\exists \boldsymbol{f} \bigvee_i \varphi^{(i)}_{\text{flow}}(\boldsymbol{f}) \wedge \varphi^{(i)}_{\text{weight}}(\boldsymbol{f}, \boldsymbol{x}) \wedge \varphi^{(i)}_{\text{nopos}}(\boldsymbol{x})$$

A formula per subgraph

Each $\varphi^{(i)}_{\text{flow}}(\boldsymbol{f})$ captures the flow constraints over one subgraph that satisfies the support condition.

No positive cycles

Now $\varphi^{(i)}_{\text{nopos}}(\boldsymbol{x})$ becomes easy: a conjunction of weight constraints over all simple cycles in the subgraph.

The weight sub-formula: from paths to runs

$$\exists \boldsymbol{f} \bigvee_i \varphi^{(i)}_{\text{flow}}(\boldsymbol{f}) \wedge \varphi^{(i)}_{\text{weight}}(\boldsymbol{f}, \boldsymbol{x}) \wedge \varphi^{(i)}_{\text{nopos}}(\boldsymbol{x})$$

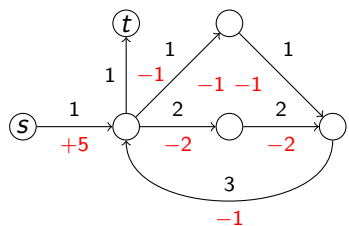
The weight sub-formula: from paths to runs

$$\exists \boldsymbol{f} \bigvee_i \varphi^{(i)}_{\text{flow}}(\boldsymbol{f}) \wedge \varphi^{(i)}_{\text{weight}}(\boldsymbol{f}, \boldsymbol{x}) \wedge \varphi^{(i)}_{\text{nopos}}(\boldsymbol{x})$$

A path is not a run

Not every path can be lifted to a run because:

- of equality tests,
- the lower-bound tests,
- the counter value cannot go negative



Decomposed flows

$$\exists \boldsymbol{f} \bigvee_i \varphi^{(i)}_{\text{flow}}(\boldsymbol{f}) \wedge \varphi^{(i)}_{\text{weight}}(\boldsymbol{f}, \boldsymbol{x}) \wedge \varphi^{(i)}_{\text{nopos}}(\boldsymbol{x})$$

Decomposed reachability certificates

Instead of a single flow f, we want $f_1, f_2, \ldots, f_{|Q|}$ such that $f = \sum_{i=1}^{|Q|} f_i$ and:

- f_i is a flow witnessing a q_i–q_{i+1} path,
- $f_j(p, q_i) = 0$ for all $i \leq j$ and all $(p, q_i) \in T$.

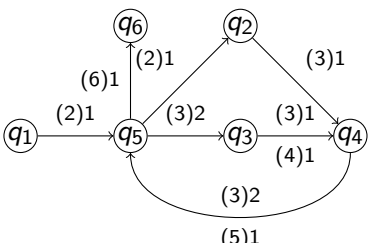
Decomposed flows

$$\exists \boldsymbol{f} \bigvee_i \varphi^{(i)}_{\text{flow}}(\boldsymbol{f}) \wedge \varphi^{(i)}_{\text{weight}}(\boldsymbol{f}, \boldsymbol{x}) \wedge \varphi^{(i)}_{\text{nopos}}(\boldsymbol{x})$$

Decomposed reachability certificates

Instead of a single flow f, we want $f_1, f_2, \ldots, f_{|Q|}$ such that $f = \sum_{i=1}^{|Q|} f_i$ and:

- f_i is a flow witnessing a q_i–q_{i+1} path,
- $f_j(p, q_i) = 0$ for all $i \leq j$ and all $(p, q_i) \in T$.



Case 1: No positive cycles

To check the path can be lifted to a run:

$$\bigwedge_{m=1}^{|Q|} \sum_{i=1}^{m} \sum_{t \in T} f_i(t)\delta(t) \geq 0$$

Weight formulas using divisibility

$$\exists \boldsymbol{f} \bigvee_i \varphi^{(i)}_{\text{flow}}(\boldsymbol{f}) \wedge \varphi^{(i)}_{\text{weight}}(\boldsymbol{f}, \boldsymbol{x}) \wedge \varphi^{(i)}_{\text{nopos}}(\boldsymbol{x})$$

A formula per decomposition

Each $\varphi^{(i)}_{\text{weight}}(\boldsymbol{x}, \boldsymbol{f})$ captures the weight constraints over one decomposition of the flow.

Weight constraints use "weak multiplication"! $\sum f_j x_j \geq 0$

Weight formulas using divisibility

$$\exists \boldsymbol{f} \bigvee_i \varphi^{(i)}_{\text{flow}}(\boldsymbol{f}) \wedge \varphi^{(i)}_{\text{weight}}(\boldsymbol{f}, \boldsymbol{x}) \wedge \varphi^{(i)}_{\text{nopos}}(\boldsymbol{x})$$

A formula per decomposition

Each $\varphi^{(i)}_{\text{weight}}(\boldsymbol{x}, \boldsymbol{f})$ captures the weight constraints over one decomposition of the flow.

Weight constraints use "weak multiplication"! $\sum f_j x_j \geq 0$

Divisibility to the rescue

We replace every appearance of fx with a product variable z_{fx}. Then the weight constraints can be rewritten:

$$(x_i \mid z_{f_i x_i}) \wedge (x_i > 0 \leftrightarrow z_{f_i x_i} > 0) \wedge \left(\sum z_{f_i x_i} \geq 0 \right)$$

Final PAD encoding

$$\exists \boldsymbol{f} \bigvee_i \bigwedge_j g_j(\boldsymbol{x}) \mid h_j(\boldsymbol{f}, \boldsymbol{x}) \wedge \varphi_i(\boldsymbol{x}) \wedge \boldsymbol{f}, \boldsymbol{x} \geq 0$$

The safety synthesis problem

Consider an instance of the safety synthesis problem.

- It has a positive answer iff $\forall \boldsymbol{x}.\Phi(\boldsymbol{x})$ is false
- $\forall \boldsymbol{x}.\Phi(\boldsymbol{x})$ is a sentence in $\forall\exists_R \mathrm{PAD}^+$

A restricted one-alternation fragment: $\forall\exists_R$PAD

- $\forall\exists_R\text{PAD} = \forall z_1 \ldots \forall z_n \exists x_1 \ldots \exists x_m.\, \varphi(\boldsymbol{x}, \boldsymbol{z})$
  - in φ, divisibilities of the form $g(\boldsymbol{z}) \mid h(\boldsymbol{x}, \boldsymbol{z})$
- $\forall\exists_R\text{PAD}^+ = \forall\exists_R\text{PAD}$ with $\neg$ not allowed before divisibility
  - A negation normal form where $\mid$ cannot be negated

Claim (Bozga-Iosif'05, Lechner'15)

The synthesis problems for SOCAP are decidable via an encoding into $\forall\exists_R\text{PAD}^+$.

$\forall\exists_R \mathrm{PAD}^+$ is undecidable

Lemma (Lechner'15)

$\forall\exists_R PAD \equiv \forall\exists_R PAD^+$

Sketch: We can rewrite $\neg(a \mid b)$ using the identity

$$\neg(a \mid b) \iff \exists q \exists r.(b = aq + r) \wedge (0 < r < b)$$

aq can be replaced by a product variable

$\forall\exists_R \mathrm{PAD}^+$ is undecidable

Lemma (Lechner'15)

$\forall\exists_R PAD \equiv \forall\exists_R PAD^+$

Sketch: We can rewrite $\neg(a \mid b)$ using the identity

$$\neg(a \mid b) \iff \exists q \exists r.(b = aq + r) \wedge (0 < r < b)$$

aq can be replaced by a product variable

Theorem (Bozga-Iosif'05)

Hilbert's 10th problem can be encoded in $\forall\exists_R PAD$, hence it is undecidable.

Sketch: Using the single restricted alternation we define

1. LCM
2. Square (x^2)
3. Multiplication

A stronger restriction of $\forall\exists_R$PAD

- $\forall\exists_R\text{PAD} = \forall z_1 \ldots \forall z_n \exists x_1 \ldots \exists x_m.\, \varphi(\boldsymbol{x}, \boldsymbol{z})$ with divisibilities $g(\boldsymbol{z}) \mid h(\boldsymbol{x}, \boldsymbol{z})$

A stronger restriction of $\forall\exists_R$PAD

- $\forall\exists_R\text{PAD} = \forall z_1 \ldots \forall z_n \exists x_1 \ldots \exists x_m.\, \varphi(\boldsymbol{x}, \boldsymbol{z})$ with divisibilities $g(\boldsymbol{z}) \mid h(\boldsymbol{x}, \boldsymbol{z})$
- The Bozga-Iosif-Lechner fragment of $\forall\exists_R$PAD:

$$\forall \boldsymbol{z} \exists \boldsymbol{x} \bigvee_{i \in I} \bigwedge_{j \in J_i} (g_j(\boldsymbol{z}) \mid h_j(\boldsymbol{x}, \boldsymbol{z}) \wedge g_j(\boldsymbol{z}) \neq 0) \wedge \varphi_i(\boldsymbol{z}) \wedge \boldsymbol{x}, \boldsymbol{z} \geq 0$$

A stronger restriction of $\forall\exists_R$PAD

- $\forall\exists_R\text{PAD} = \forall z_1 \ldots \forall z_n \exists x_1 \ldots \exists x_m.\, \varphi(\boldsymbol{x}, \boldsymbol{z})$ with divisibilities $g(\boldsymbol{z}) \mid h(\boldsymbol{x}, \boldsymbol{z})$
- The Bozga-Iosif-Lechner fragment of $\forall\exists_R$PAD:

$$\forall \boldsymbol{z} \exists \boldsymbol{x} \bigvee_{i \in I} \bigwedge_{j \in J_i} (g_j(\boldsymbol{z}) \mid h_j(\boldsymbol{x}, \boldsymbol{z}) \wedge g_j(\boldsymbol{z}) \neq 0) \wedge \varphi_i(\boldsymbol{z}) \wedge \boldsymbol{x}, \boldsymbol{z} \geq 0$$

Theorem (Bozga-Iosif'05, Lechner'15, P-Raha'20)

The BIL fragment of $\forall\exists_R$PAD is decidable and in coN2EXP.

A stronger restriction of $\forall\exists_R$PAD

- $\forall\exists_R\text{PAD} = \forall z_1 \ldots \forall z_n \exists x_1 \ldots \exists x_m.\, \varphi(\boldsymbol{x}, \boldsymbol{z})$ with divisibilities $g(\boldsymbol{z}) \mid h(\boldsymbol{x}, \boldsymbol{z})$
- The Bozga-Iosif-Lechner fragment of $\forall\exists_R$PAD:

$$\forall \boldsymbol{z} \exists \boldsymbol{x} \bigvee_{i \in I} \bigwedge_{j \in J_i} (g_j(\boldsymbol{z}) \mid h_j(\boldsymbol{x}, \boldsymbol{z}) \wedge g_j(\boldsymbol{z}) \neq 0) \wedge \varphi_i(\boldsymbol{z}) \wedge \boldsymbol{x}, \boldsymbol{z} \geq 0$$

Theorem (Bozga-Iosif'05, Lechner'15, P-Raha'20)

The BIL fragment of $\forall\exists_R$PAD is decidable and in coN2EXP.

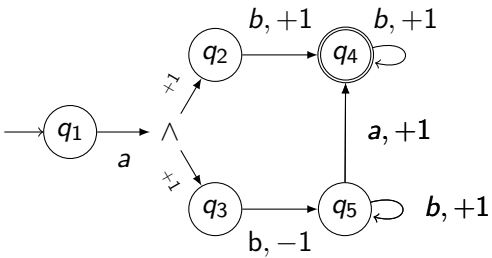
We massage $\forall \boldsymbol{x}.\Phi(\boldsymbol{x})$ to get it into the BIL-fragment.

Corollary

The safety synthesis problem is decidable and in N2EXP.

Outline

Alternating two-way automata (A2A)



- existential and universal non-determinism
- a run is a tree
- accepting if all infinite branches are Büchi-accepting

Theorem (Serre'06)

The emptiness problem for A2As is in PSPACE.

From SOCAP to A2A

$$Op = \{-1, 0, +1\} \cup ZT \cup PT$$

From SOCAP to A2A

$Op = \{-1, 0, +1\} \cup ZT \cup PT$

Lemma (Based on Bollig-Quaas-Sangnier'19)

For every SOCAP $\mathcal{A}$ we construct an A2A $\mathcal{T}$ of poly-size which accepts words corresponding to valuations for which universal reachability holds.

From SOCAP to A2A

$Op = \{-1, 0, +1\} \cup ZT \cup PT$

Lemma (Based on Bollig-Quaas-Sangnier'19)

For every SOCAP $\mathcal{A}$ we construct an A2A $\mathcal{T}$ of poly-size which accepts words corresponding to valuations for which universal reachability holds.

Sketch:

- Encode valuations as parameter words

From SOCAP to A2A

$Op = \{-1, 0, +1\} \cup ZT \cup PT$

Lemma (Based on Bollig-Quaas-Sangnier'19)

For every SOCAP $\mathcal{A}$ we construct an A2A $\mathcal{T}$ of poly-size which accepts words corresponding to valuations for which universal reachability holds.

Sketch:

- Encode valuations as parameter words
- For every transition we build an A2A

From SOCAP to A2A

$Op = \{-1, 0, +1\} \cup ZT \cup PT$

Lemma (Based on Bollig-Quaas-Sangnier'19)

For every SOCAP $\mathcal{A}$ we construct an A2A $\mathcal{T}$ of poly-size which accepts words corresponding to valuations for which universal reachability holds.

Sketch:

- Encode valuations as parameter words
- For every transition we build an A2A
- Accept the reaching runs
- (. . . and runs that "die off")

SOCAP to A2A

Valuations as words

- $V : X \to \mathbb{N}$ encoded with $\Sigma = X \cup \{\square\}$

SOCAP to A2A

Valuations as words

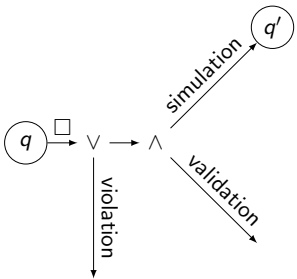
- $V : X \to \mathbb{N}$ encoded with $\Sigma = X \cup \{\square\}$
- $\square\square\square x_1 \square x_2 \square^\omega$ encodes $x_1 \mapsto 2, x_2 \mapsto 3$

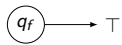
SOCAP to A2A

Valuations as words

- $V : X \to \mathbb{N}$ encoded with $\Sigma = X \cup \{\square\}$
- $\square\square\square x_1 \square x_2 \square^\omega$ encodes $x_1 \mapsto 2, x_2 \mapsto 3$

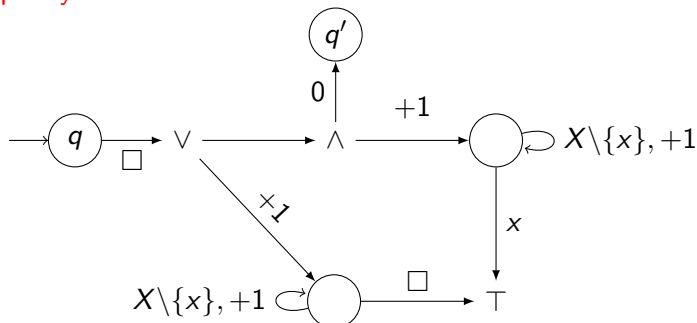
A2As for each transition (q, q')





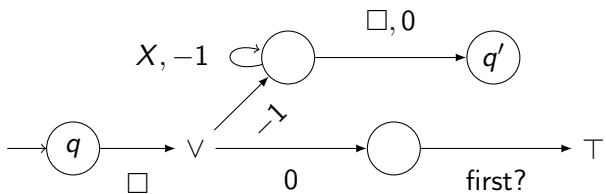
SOCAP to A2A: Equality tests

Encoding an equality test



SOCAP to A2A: Decrements

Encoding a decrement



SOCAP to A2A: Putting things together

To check all runs:

If we created subautomata and transitions

- $(q, \square, \mathcal{T}_1)$ to simulate $q \xrightarrow{op_1} q_1$,
- $(q, \square, \mathcal{T}_2)$ to simulate $q \xrightarrow{op_2} q_2$.

Then, we replace the transitions with: $(q, \square, \mathcal{T}_1 \wedge \mathcal{T}_2) \in \mathcal{T}$

SOCAP to A2A: An algorithm for reachability

Lemma

The reachability synthesis problem for SOCAP with non-parametric updates is in PSPACE.

SOCAP to A2A: An algorithm for reachability

Lemma

The reachability synthesis problem for SOCAP with non-parametric updates is in PSPACE.

- For every A2A $\mathcal{T}$ there is an NWA $\mathcal{B}$ of exponential size accepting same language
- Non-emptiness witnesses for NWAs are simple "lassos"
- $\implies$ the reachability synthesis problem admits exponential (w.r.t. the SOCAP) witnesses and thus polynomial in binary encoding

SOCAP to A2A: An algorithm for reachability

Lemma

The reachability synthesis problem for SOCAP with non-parametric updates is in PSPACE.

- For every A2A $\mathcal{T}$ there is an NWA $\mathcal{B}$ of exponential size accepting same language
- Non-emptiness witnesses for NWAs are simple "lassos"
- $\implies$ the reachability synthesis problem admits exponential (w.r.t. the SOCAP) witnesses and thus polynomial in binary encoding
- Guess a valuation and check universal reachability for resulting SOCA

Theorem

The reachability synthesis problem for SOCAP is in $\mathrm{NP}^{\mathrm{coNP}}$.

Conclusion

	LTL	Reachability	Safety	Büchi	coBüchi
Lower bound	PSPACE-hard	coNP-hard	— NP^{NP}-hard —		
Upper bound	in N3EXP	— in N2EXP —			

Open questions

- What is the exact complexity of the synthesis problems?
- What about approximations via continuous relaxations?
- What is the complexity of the BIL fragment?
- Can the A2A approach be extended to handle parametric updates?
  - by adding a single pebble?