# Revisiting Synthesis for One-Counter Automata

Ritam Raha

*(Joint work with Guillermo A. Perez)*

MOVEP 2020

Universiteit Antwerpen

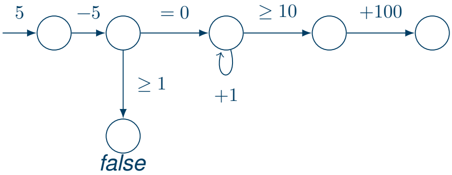
# Outline

# One-Counter Automata

```
n=5
n= max(0,n-5)
if n=0:
    while (n<10):
      n=n+1
    n= n+100
else:
    return false
```

# One-Counter Automata

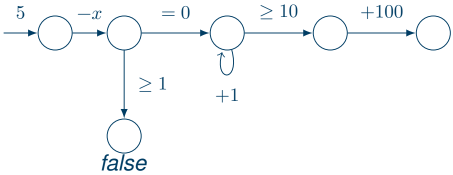
```
n=5
n= max(0,n-5)
if n=0:
    while (n<10):
      n=n+1
    n= n+100
else:
    return false
```




- Configurations: $(q, c), c \geq 0$;

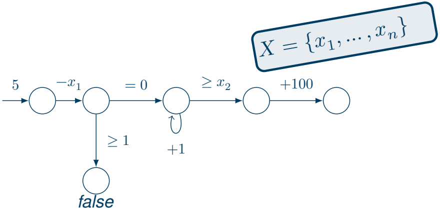
# One-Counter Automata

```
n=5
n= max(0,n-x)
if n=0:
    while (n<10):
      n=n+1
    n= n+100
else:
    return false
```

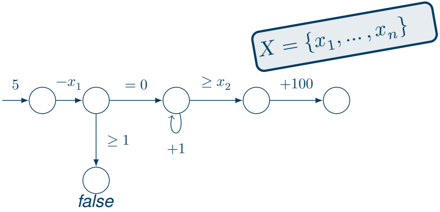



- Configurations: $(q, c), c \geq 0$;

# One-Counter Automata



# One-Counter Automata

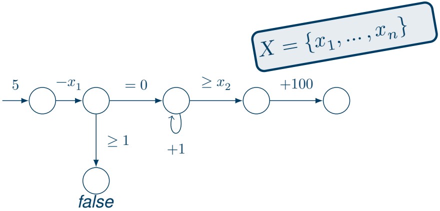



**Definition (<S>OCA<P>)**

$\mathcal{A} = (Q, T, \delta, q_{in}, X) \quad \delta : T \to Op$

$Op :=$

# One-Counter Automata



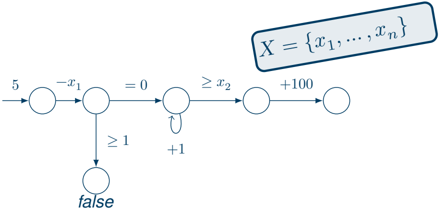

**Definition (<S>OCA<P>)**

$\mathcal{A} = (Q, T, \delta, q_{in}, X) \quad \delta : T \rightarrow Op$

$Op :=$

- $CU := \{+a : a \in \mathbb{Z}\}$
- $CT := \{= 0, \geq a, = a : a \in \mathbb{Z}\}$

# One-Counter Automata



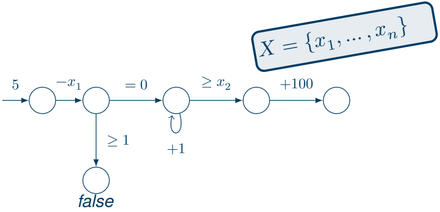

**Definition (<S>OCA<P>)**

$\mathcal{A} = (Q, T, \delta, q_{in}, X) \quad \delta : T \to Op$

$Op :=$

- $CU := \{+a : a \in \mathbb{Z}\}$
- $CT := \{= 0, \geq a, = a : a \in \mathbb{Z}\}$
- $PU := \{+x, -x : x \in X\}$
- $PT := \{= x, \geq x : x \in X\}$

# One-Counter Automata




**Definition (<S>OCA<P>)**

$\mathcal{A} = (Q, T, \delta, q_{in}, X) \quad \delta : T \rightarrow Op$

$Op :=$

- $CU := \{+a : a \in \mathbb{Z}\}$
- $CT := \{= 0, \geq a, = a : a \in \mathbb{Z}\}$
- $PU := \{+x, -x : x \in X\}$
- $PT := \{= x, \geq x : x \in X\}$

Non-parametric: $X = \emptyset$

# One-Counter Automata

| Models | | *CU* | *PU* | *ZT* | *PT* |
|---|---|---|---|---|---|
| Non-Parametric | SOCA | ✓ | ✗ | ✓ | ✗ |
| Parametric | OCAPT | $\{-1, 0, 1\}$ | ✗ | ✓ | ✓ |
| | SOCAP | ✓ | ✓ | ✓ | ✓ |

# Decision Problems

**Non-Parametric:**

**Parametric:**

# Decision Problems

**Non-Parametric:**

**Reach**

$\exists \rho$ such that

$(q_{in}, 0) \xrightarrow{\rho} q_f$

**Parametric:**

# Decision Problems

**Non-Parametric:** 🙂

**Reach**

$\exists\rho$ such that

$(q_{in}, 0) \xrightarrow{\rho} q_f$

[NP-complete
(Haase-Kreutzer-Joel-Worrell '09)]

**Parametric:**

# Decision Problems

**Non-Parametric:** 🙂

**Reach**

$\exists\rho$ such that

$(q_{in}, 0) \xrightarrow{\rho} q_f$

[NP-complete
(Haase-Kreutzer-Joel-Worrell '09)]

**Parametric:**

**Par-Reach**

$\exists V(X)$ such that $\exists\rho$,

$(q_{in}, 0) \xrightarrow{\rho}_V q_f$

# Decision Problems

**Non-Parametric:**

**Reach**

$\exists\rho$ such that

$(q_{in}, 0) \xrightarrow{\rho} q_f$

[NP-complete]

(Haase-Kreutzer-Joel-Worrell '09)]

**Parametric:**

**Par-Reach**

$\exists V(X)$ such that $\exists\rho$,

$(q_{in}, 0) \xrightarrow{\rho}_V q_f$

[in NEXPTIME]

(Haase-Kreutzer-Joel-Worrell '09)

# Decision Problems

**Non-Parametric:** 🙂

**Reach**

$\exists\rho$ such that
$(q_{in}, 0) \xrightarrow{\rho} q_f$

[NP-complete]
(Haase-Kreutzer-Joel-Worrell '09)]

**Parametric:** 🙂

**Par-Reach**

$\exists V(X)$ such that $\exists\rho$,
$(q_{in}, 0) \xrightarrow{\rho}_V q_f$

[in NEXPTIME]
(Haase-Kreutzer-Joel-Worrell '09)

**UnivReach**

For all infinite $\rho$,
$(q_{in}, 0) \xrightarrow{\rho} q_f$

# Decision Problems

**Non-Parametric:** 🙂

**Reach**

$\exists\rho$ such that

$(q_{in}, 0) \xrightarrow{\rho} q_f$

[NP-complete]
(Haase-Kreutzer-Joel-Worrell '09)]

**Parametric:** 🙂

**Par-Reach**

$\exists V(X)$ such that $\exists\rho$,

$(q_{in}, 0) \xrightarrow{\rho}_V q_f$

[in NEXPTIME]
(Haase-Kreutzer-Joel-Worrell '09)

**UnivReach**

For all infinite $\rho$,

$(q_{in}, 0) \xrightarrow{\rho} q_f$

**SynthReach**

$\exists V(X)$ such that for all infinite $\rho$, $(q_{in}, 0) \xrightarrow{\rho}_V q_f$

# Decision Problems

**Non-Parametric:**

**REACH**

$\exists\rho$ such that
$(q_{in}, 0) \xrightarrow{\rho} q_f$

[NP-complete]
(Haase-Kreutzer-Joel-Worrell '09)]

**Parametric:**

**PAR-REACH**

$\exists V(X)$ such that $\exists\rho$,
$(q_{in}, 0) \xrightarrow{\rho}_V q_f$

[in NEXPTIME]
(Haase-Kreutzer-Joel-Worrell '09)

**UNIVREACH**

For all infinite $\rho$,
$(q_{in}, 0) \xrightarrow{\rho} q_f$

**SYNTHREACH**

$\exists V(X)$ such that for all infinite $\rho$, $(q_{in}, 0) \xrightarrow{\rho}_V q_f$

# UnivReach for SOCA

*Op* $=$ *CU* $\cup$ *ZT*; UNIVREACH: all infinite paths reach $q_f$;

# UnivReach for SOCA

$Op = CU \cup ZT$; UNIVREACH: all infinite paths reach $q_f$;

$\neg$REACH := two REACH queries

# UnivReach for SOCA

*Op* $=$ *CU* $\cup$ *ZT*;  UnivReach: all infinite paths reach $q_f$;

$\neg$Reach $:=$ two Reach queries (coNP);

# UnivReach for SOCA

$Op = CU \cup ZT$;     UnivReach: all infinite paths reach $q_f$;

$\neg$Reach := two Reach queries (coNP);

Hardness from reduction from co-subsetsum.

**Proposition**

*The UnivReach problem for SOCA is coNP-complete.*

# Decision Problems

**Non-Parametric:**

**REACH**

$\exists \rho$ such that $(q_{in}, 0) \xrightarrow{\rho} q_f$

[NP-complete]
(Haase-Kreutzer-Joel-Worrell '09)]

**UNIVREACH**

For all infinite $\rho$, $(q_{in}, 0) \xrightarrow{\rho} q_f$

**Parametric:**

**PAR-REACH**

$\exists V(X)$ such that $\exists \rho$, $(q_{in}, 0) \xrightarrow{\rho}_V q_f$

[in NEXPTIME]
(Haase-Kreutzer-Joel-Worrell '09)

**SYNTHREACH**

$\exists V(X)$ such that for all infinite $\rho$, $(q_{in}, 0) \xrightarrow{\rho}_V q_f$

# Decision Problems

**Non-Parametric:**

**REACH**

$\exists \rho$ such that

$(q_{in}, 0) \xrightarrow{\rho} q_f$

[NP-complete]

(Haase-Kreutzer-Joel-Worrell '09)]

**UNIVREACH**

For all infinite $\rho$,

$(q_{in}, 0) \xrightarrow{\rho} q_f$

[coNP-complete]

**Parametric:**

**PAR-REACH**

$\exists V(X)$ such that $\exists \rho$,

$(q_{in}, 0) \xrightarrow{\rho}_V q_f$

[in NEXPTIME]

(Haase-Kreutzer-Joel-Worrell '09)

**SYNTHREACH**

$\exists V(X)$ such that for all infinite $\rho$, $(q_{in}, 0) \xrightarrow{\rho}_V q_f$

# Decision Problems

**Non-Parametric:** ☺

**REACH**
$\exists \rho$ such that
$(q_{in}, 0) \xrightarrow{\rho} q_f$

[NP-complete]
(Haase-Kreutzer-Joel-Worrell '09)]

☺

**UNIVREACH**
For all infinite $\rho$,
$(q_{in}, 0) \xrightarrow{\rho} q_f$

[coNP-complete]

**Parametric:** ☺

**PAR-REACH**
$\exists V(X)$ such that $\exists \rho$,
$(q_{in}, 0) \xrightarrow{\rho}_V q_f$

[in NEXPTIME]
(Haase-Kreutzer-Joel-Worrell '09)

☺?

**SYNTHREACH**
$\exists V(X)$ such that for all infinite $\rho$, $(q_{in}, 0) \xrightarrow{\rho}_V q_f$

- Presburger arithmetic (PA) $:= \langle \mathbb{N}, 0, 1, +, < \rangle$

# PAD & Extensions

- Presburger arithmetic (PA) $:= \langle \mathbb{N}, 0, 1, +, < \rangle$
- Presburger arithmetic with Divisibility (PAD) $:=$ PA $+ |$
  ($a \mid b \iff \exists c \in \mathbb{Z} : b = ac$)

# PAD & Extensions

- Presburger arithmetic (PA) $:= \langle \mathbb{N}, 0, 1, +, < \rangle$
- Presburger arithmetic with Divisibility (PAD) $:=$ PA $+|$
  $(a \mid b \iff \exists c \in \mathbb{Z} : b = ac)$

**Theorem (Lechner-Ouaknine-Worrell'15)**

*EPAD is decidable in NEXPTIME.*

# Outline

1. One-Counter Automata(Parametric) and Synthesis Problem
2. Previous Approach
3. Approach with Alternating Two Way Automata for a subclass
4. Approach with Partial Observation Games

# $\forall\exists_R$PAD & Undecidability

- $\forall\exists_R$PAD $:= \forall z_1 \ldots \forall z_n \exists x_1 \ldots \exists x_m \varphi(\vec{x}, \vec{z})$

# $\forall\exists_R$PAD & Undecidability

- $\forall\exists_R$PAD $:= \forall z_1 \ldots \forall z_n \exists x_1 \ldots \exists x_m \varphi(\vec{x}, \vec{z})(f(\vec{z}) \mid g(\vec{x}, \vec{z}))$.

# $\forall\exists_R$PAD & Undecidability

- $\forall\exists_R$PAD $:= \forall z_1 \dots \forall z_n \exists x_1 \dots \exists x_m \varphi(\vec{x}, \vec{z})(f(\vec{z}) \mid g(\vec{x}, \vec{z}))$.
- $\forall\exists_R$PAD$^+$ $:= \forall\exists_R$PAD with $\neg$ not allowed before divisibility.

# $\forall\exists_R$PAD & Undecidability

- $\forall\exists_R$PAD $:= \forall z_1 \ldots \forall z_n \exists x_1 \ldots \exists x_m \varphi(\vec{x}, \vec{z})(f(\vec{z}) \mid g(\vec{x}, \vec{z}))$.
- $\forall\exists_R$PAD$^+$ $:= \forall\exists_R$PAD with $\neg$ not allowed before divisibility.

**Claim(Lechner'15)**

SynthReach for SOCAP is decidable by a reduction to $\forall\exists_R$PAD$^+$.

# $\forall\exists_R$PAD & Undecidability

- $\forall\exists_R$PAD := $\forall z_1 \ldots \forall z_n \exists x_1 \ldots \exists x_m \varphi(\vec{x}, \vec{z})(f(\vec{z}) \mid g(\vec{x}, \vec{z}))$.
- $\forall\exists_R$PAD$^+$ := $\forall\exists_R$PAD with $\neg$ not allowed before divisibility.

**Claim(Lechner'15)**

SynthReach for SOCAP is decidable by a reduction to $\forall\exists_R$PAD$^+$.

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

# $\forall\exists_R$PAD & Undecidability

- $\forall\exists_R$PAD := $\forall z_1 \ldots \forall z_n \exists x_1 \ldots \exists x_m \varphi(\vec{x}, \vec{z})(f(\vec{z}) \mid g(\vec{x}, \vec{z}))$.
- $\forall\exists_R$PAD$^+$ := $\forall\exists_R$PAD with $\neg$ not allowed before divisibility.

**Claim(Lechner'15)**

SynthReach for SOCAP is decidable by a reduction to $\forall\exists_R$PAD$^+$.

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

Idea: We have to rewrite $\neg(a \mid b)$

# $\forall\exists_R$PAD & Undecidability

- $\forall\exists_R$PAD := $\forall z_1 \ldots \forall z_n \exists x_1 \ldots \exists x_m \varphi(\vec{x}, \vec{z})(f(\vec{z}) \mid g(\vec{x}, \vec{z}))$.
- $\forall\exists_R$PAD$^+$ := $\forall\exists_R$PAD with $\neg$ not allowed before divisibility.

**Claim(Lechner'15)**

SynthReach for SOCAP is decidable by a reduction to $\forall\exists_R$PAD$^+$.

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

Idea: We have to rewrite $\neg(a \mid b)$

$\neg(a \mid b) \equiv b = aq + r$ where, $0 < r < b$.

# $\forall\exists_R$PAD $\&$ Undecidability

Using techniques from Bozga-Iosif'05 and Lechner'15:

$\forall\exists_R\text{PAD}^+ \iff \forall\exists_R\text{PAD}$

# $\forall\exists_R$PAD $\&$ Undecidability

Using techniques from Bozga-Iosif'05 and Lechner'15:

$\forall\exists_R$PAD$^+$ $\iff$ $\forall\exists_R$PAD $\implies$ LCM

# $\forall\exists_R$PAD $\&$ Undecidability

Using techniques from Bozga-Iosif'05 and Lechner'15:

$\forall\exists_R$PAD$^+$ $\iff$ $\forall\exists_R$PAD $\implies$ LCM $\implies$ Square

# $\forall\exists_R$PAD & Undecidability

Using techniques from Bozga-Iosif'05 and Lechner'15:

$\forall\exists_R$PAD$^+$ $\iff$ $\forall\exists_R$PAD $\implies$ LCM $\implies$ Square $\implies$ Multiplication

# $\forall\exists_R$PAD $\&$ Undecidability

Using techniques from Bozga-Iosif'05 and Lechner'15:

$\forall\exists_R$PAD$^+$ $\iff$ $\forall\exists_R$PAD $\implies$ LCM $\implies$ Square $\implies$ Multiplication

Undecidable!!

# Decision Problems

**Non-Parametric:**

**Reach**

$\exists \rho$ such that

$(q_{in}, 0) \xrightarrow{\rho} q_f$

[NP-complete]

(Haase-Kreutzer-Joel-Worrell '09)]

**UnivReach**

For all infinite $\rho$,

$(q_{in}, 0) \xrightarrow{\rho} q_f$

[coNP-complete]

**Parametric:**

**Par-Reach**

$\exists V(X)$ such that $\exists \rho$,

$(q_{in}, 0) \xrightarrow{\rho}_V q_f$

[in NEXPTIME]

(Haase-Kreutzer-Joel-Worrell '09)

**SynthReach**

$\exists V(X)$ such that for all

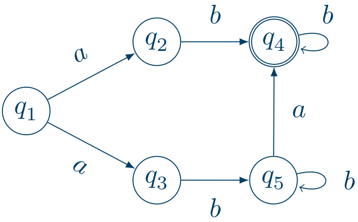
infinite $\rho$, $(q_{in}, 0) \xrightarrow{\rho}_V q_f$

# Outline

1. One-Counter Automata(Parametric) and Synthesis Problem
2. Previous Approach
3. Approach with Alternating Two Way Automata for a subclass
4. Approach with Partial Observation Games

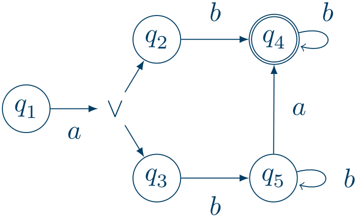
# Alternating Two Way Automaton

$w = ab^\omega$



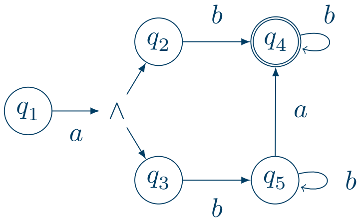
# Alternating Two Way Automaton

$w = ab^{\omega}$



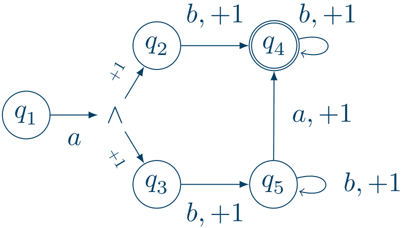
# Alternating Two Way Automaton

$w = ab^\omega$



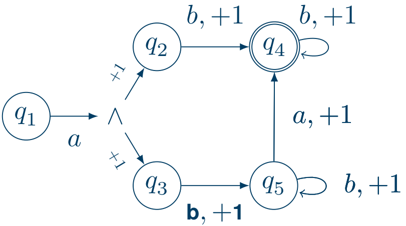
# Alternating Two Way Automaton

$w = ab^{\omega}$



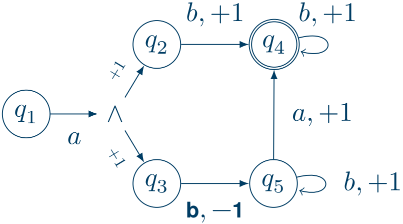
# Alternating Two Way Automaton

$w = ab^\omega$



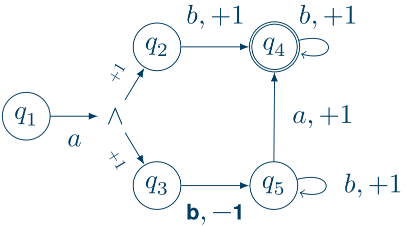
# Alternating Two Way Automaton

$w = ab^{\omega}$



# Alternating Two Way Automaton

$w = ab^\omega$




**Theorem (Serre'06)**

*The non-emptiness problem for A2As is in PSPACE.*

# From OCAPT to A2A

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

# From OCAPT to A2A

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

**Proposition (Similar Idea to Bollig-Quaas-Sangnier'19)**

*For every OCAPT $\mathcal{A} \Rightarrow$ an A2A $\mathcal{T}$ of poly-size such that, it accepts words corresponding to valuations for which SynthReach is true.*

# From OCAPT to A2A

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

**Proposition (Similar Idea to Bollig-Quaas-Sangnier'19)**

*For every OCAPT $\mathcal{A} \Rightarrow$ an A2A $\mathcal{T}$ of poly-size such that, it accepts words corresponding to valuations for which SYNTHREACH is true.*

Idea :

# From OCAPT to A2A

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

**Proposition (Similar Idea to Bollig-Quaas-Sangnier'19)**

*For every OCAPT $\mathcal{A} \Rightarrow$ an A2A $\mathcal{T}$ of poly-size such that, it accepts words corresponding to valuations for which SynthReach is true.*

Idea :

- Encode valuations as parameter word

# From OCAPT to A2A

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

**Proposition (Similar Idea to Bollig-Quaas-Sangnier'19)**

*For every OCAPT $\mathcal{A} \Rightarrow$ an A2A $\mathcal{T}$ of poly-size such that, it accepts words corresponding to valuations for which SynthReach is true.*

Idea :

- Encode valuations as parameter word
- For every operation, we build an A2A

# From OCAPT to A2A

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

**Proposition (Similar Idea to Bollig-Quaas-Sangnier'19)**

*For every OCAPT $\mathcal{A} \Rightarrow$ an A2A $\mathcal{T}$ of poly-size such that, it accepts words corresponding to valuations for which SYNTHREACH is true.*

Idea :

- Encode valuations as parameter word
- For every operation, we build an A2A
- Accept the reaching runs

# OCAPT to A2A

Valuation to Words:

# OCAPT to A2A

Valuation to Words:

$V : X \to \mathbb{N}; \qquad \Sigma = X \cup \{\Box\};$

# OCAPT to A2A

Valuation to Words:

$V : X \to \mathbb{N}; \qquad \Sigma = X \cup \{\square\};$

$\square$

# OCAPT to A2A

Valuation to Words:

$V : X \to \mathbb{N}$;       $\Sigma = X \cup \{\square\}$;

$\square\square\square x_1 \square x_2 \square^\omega$

# OCAPT to A2A

Valuation to Words:

$V : X \to \mathbb{N}; \qquad \Sigma = X \cup \{\Box\};$

$\Box\Box\Box x_1 \Box x_2 \Box^\omega \Rightarrow x_1 \mapsto 2, x_2 \mapsto 3;$

# OCAPT to A2A

Valuation to Words:

$V : X \to \mathbb{N}; \qquad \Sigma = X \cup \{\Box\};$

$\Box\Box\Box x_1 \Box x_2 \Box^\omega \Rightarrow x_1 \mapsto 2, x_2 \mapsto 3;$

*Op* to A2A:

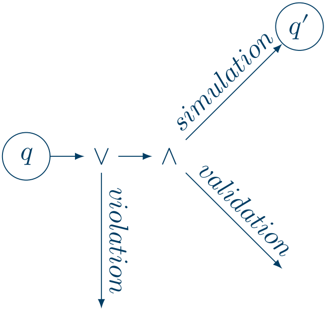
# OCAPT to A2A

Valuation to Words:

$V : X \to \mathbb{N}; \qquad \Sigma = X \cup \{\square\};$

$\square\square\square x_1 \square x_2 \square^\omega \Rightarrow x_1 \mapsto 2, x_2 \mapsto 3;$

*Op* to A2A:



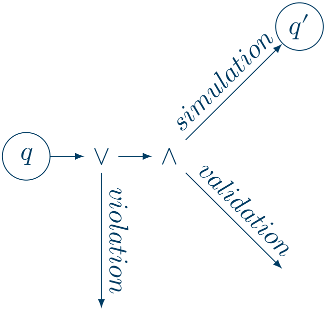
# OCAPT to A2A

Valuation to Words:

$V : X \to \mathbb{N}; \qquad \Sigma = X \cup \{\square\};$

$\square\square\square x_1 \square x_2 \square^\omega \Rightarrow x_1 \mapsto 2, x_2 \mapsto 3;$

*Op* to A2A:



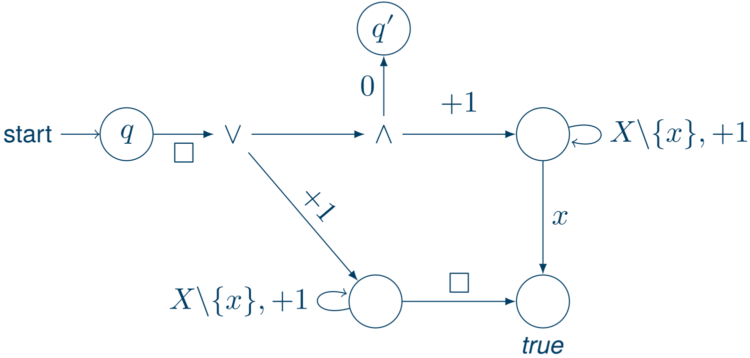

Accept reaching runs:

# OCAPT to A2A

- Encoding Equality Test:

# OCAPT to A2A

- Encoding Equality Test:

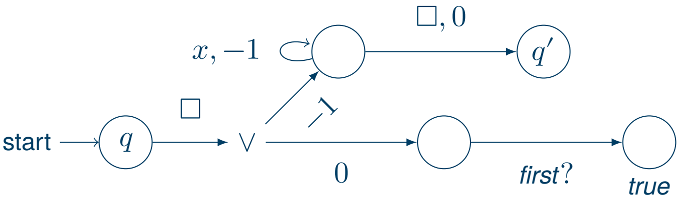


# OCAPT to A2A

- Encoding Decrement:

# OCAPT to A2A

- Encoding Decrement:



# OCAPT to A2A

We want to check all paths:

# OCAPT to A2A

We want to check all paths:

$$q \xrightarrow{op_1} q_1 \Rightarrow (q, \Box, \mathcal{T}_1)$$

# OCAPT to A2A

We want to check all paths:

$$q \xrightarrow{op_1} q_1 \Rightarrow (q, \Box, \mathcal{T}_1)$$
$$q \xrightarrow{op_2} q_2 \Rightarrow (q, \Box, \mathcal{T}_2)$$

# OCAPT to A2A

We want to check all paths:

$$q \xrightarrow{op_1} q_1 \Rightarrow (q, \Box, \mathcal{T}_1)$$

$$q \xrightarrow{op_2} q_2 \Rightarrow (q, \Box, \mathcal{T}_2)$$

$$(q, \Box, \mathcal{T}_1 \wedge \mathcal{T}_2) \in \mathcal{T}$$

# OCAPT to A2A

- We check if the word is valid.

# OCAPT to A2A

- We check if the word is valid.
- We gradually build A2A for every op.

# OCAPT to A2A

- We check if the word is valid.
- We gradually build A2A for every op.
- Finally we have the global A2A.

# OCAPT to A2A

- We check if the word is valid.
- We gradually build A2A for every op.
- Finally we have the global A2A.

# OCAPT to A2A

- We check if the word is valid.
- We gradually build A2A for every op.
- Finally we have the global A2A.

PSPACE!!

# OCAPT to A2A

- We check if the word is valid.
- We gradually build A2A for every op.
- Finally we have the global A2A.

PSPACE!!

For an A2A $\mathcal{T} \Rightarrow$ an NBA $\mathcal{B}$ of exponential size accpeting same language!!

# OCAPT to A2A

- We check if the word is valid.
- We gradually build A2A for every op.
- Finally we have the global A2A.

PSPACE!!

For an A2A $\mathcal{T} \Rightarrow$ an NBA $\mathcal{B}$ of exponential size accpeting same language!!

**Lemma**

*If SYNTHREACH is true then we can bound the valuation to be exponential and thus poly size in succinct encoding.*

# OCAPT to A2A

- We check if the word is valid.
- We gradually build A2A for every op.
- Finally we have the global A2A.

PSPACE!!

For an A2A $\mathcal{T} \Rightarrow$ an NBA $\mathcal{B}$ of exponential size accpeting same language!!

**Lemma**

*If SynthReach is true then we can bound the valuation to be exponential and thus poly size in succinct encoding.*

Guess the valuation

# OCAPT to A2A

- We check if the word is valid.
- We gradually build A2A for every op.
- Finally we have the global A2A.

PSPACE!!

For an A2A $\mathcal{T} \Rightarrow$ an NBA $\mathcal{B}$ of exponential size accpeting same language!!

**Lemma**

*If* SynthReach *is true then we can bound the valuation to be exponential and thus poly size in succinct encoding.*

Guess the valuation $\Rightarrow$ UnivReach of SOCA

# OCAPT to A2A

- We check if the word is valid.
- We gradually build A2A for every op.
- Finally we have the global A2A.

PSPACE!!

For an A2A $\mathcal{T} \Rightarrow$ an NBA $\mathcal{B}$ of exponential size accpeting same language!!

**Lemma**

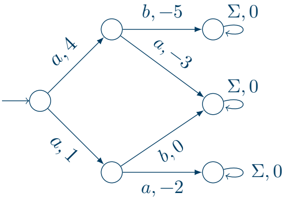
*If SynthReach is true then we can bound the valuation to be exponential and thus poly size in succinct encoding.*

Guess the valuation $\Rightarrow$ UnivReach of SOCA $\Rightarrow$ $\text{NP}^{\text{NP}}$!!

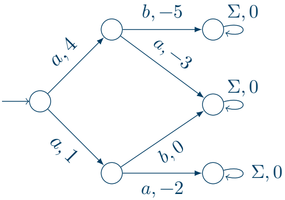
# Outline

1. One-Counter Automata (Parametric) and Synthesis Problem
2. Previous Approach
3. Approach with Alternating Two Way Automata for a subclass
4. Approach with Partial Observation Games

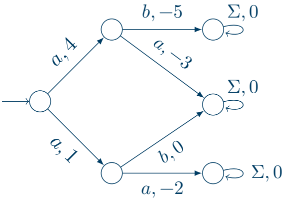
# Partial Observation Energy Game



# Partial Observation Energy Game



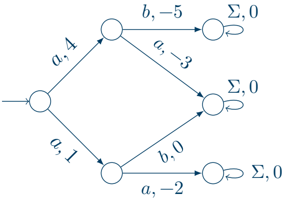

# Partial Observation Energy Game




- Chooses an action

- Resolves Non-determinism

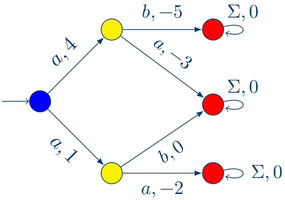
# Partial Observation Energy Game




- Chooses an action
- Keeps the energy level positive always at infinite play

- Resolves Non-determinism
- Makes the energy level negative at some point

# Partial Observation Energy Game




- Chooses an action
- Keeps the energy level positive always at infinite play

- Resolves Non-determinism
- Makes the energy level negative at some point

# SOCAP to POEG

**Lemma**

*For every SOCAP $\mathcal{A}$ $\Rightarrow$ an POEG $\mathcal{G}$ such that for all valuations $V$ there exists a reaching run of $\mathcal{A}$ iff Eve has a winning strategy in $\mathcal{G}$.*

# SOCAP to POEG

**Lemma**

*For every SOCAP $\mathcal{A}$ $\Rightarrow$ an POEG $\mathcal{G}$ such that for all valuations $V$ there exists a reaching run of $\mathcal{A}$ iff Eve has a winning strategy in $\mathcal{G}$.*

Idea:

- Adam chooses valuation and Eve has to simulate reaching run for that.

# SOCAP to POEG

**Lemma**

*For every SOCAP $\mathcal{A} \Rightarrow$ an POEG $\mathcal{G}$ such that for all valuations $V$ there exists a reaching run of $\mathcal{A}$ iff Eve has a winning strategy in $\mathcal{G}$.*

Idea:

- Adam chooses valuation and Eve has to simulate reaching run for that.
- We create gadgets in the game simulating *Op*.

# SOCAP to POEG

**Lemma**

*For every SOCAP $\mathcal{A}$ $\Rightarrow$ an POEG $\mathcal{G}$ such that for all valuations $V$ there exists a reaching run of $\mathcal{A}$ iff Eve has a winning strategy in $\mathcal{G}$.*

Idea:

- Adam chooses valuation and Eve has to simulate reaching run for that.
- We create gadgets in the game simulating *Op*.
- Using partial observation, Adam can force Eve to defeat in a gadget if she cheats.

# SOCAP to POEG

**Lemma**

*For every SOCAP $\mathcal{A} \Rightarrow$ an POEG $\mathcal{G}$ such that for all valuations $V$ there exists a reaching run of $\mathcal{A}$ iff Eve has a winning strategy in $\mathcal{G}$.*

Idea:

- Adam chooses valuation and Eve has to simulate reaching run for that.
- We create gadgets in the game simulating *Op*.
- Using partial observation, Adam can force Eve to defeat in a gadget if she cheats.

**Theorem**

*SynthReach problem for SOCAP is decidable.*

# Decision Problems

**Non-Parametric:**

**REACH**

$\exists \rho$ such that $(q_{in}, 0) \xrightarrow{\rho} q_f$

[NP-complete]

**UNIVREACH**

For all infinite $\rho$, $(q_{in}, 0) \xrightarrow{\rho} q_f$

[coNP-complete]

**Parametric:**

**PAR-REACH**

$\exists V(X)$ such that $\exists \rho$, $(q_{in}, 0) \xrightarrow{\rho}_V q_f$

[in NEXPTIME]

**SYNTHREACH**

$\exists V(X)$ such that for all infinite $\rho$, $(q_{in}, 0) \xrightarrow{\rho}_V q_f$

[Decidable]

# Conclusion

- Clarified the decidability status for the synthesis problem for OCA.
- If parameters are only allowed on tests, the problem is in $NP^{NP}$.
- In full generality, for SOCA it is decidable.

# Conclusion

- Clarified the decidability status for the synthesis problem for OCA.
- If parameters are only allowed on tests, the problem is in $NP^{NP}$.
- In full generality, for SOCA it is decidable.

  Approach with Multi Energy Game with resets and transfers- Better Complexity?