Parikh Automata and Friends

On the Expressiveness of Parikh Automata and Related Models (NCMA11)

Bounded Parikh Automata (WORDS11)

M. Cadilhac[1], A. Finkel[2], and P. McKenzie[1]

1: Université de Montréal

2: ENS CACHAN

Tübingen, September 22th, 2011

Parikh Automata and Friends

On the Expressiveness of Parikh Automata and Related Models (NCMA11)

Bounded Parikh Automata (WORDS11)

M. Cadilhac[1], A. Finkel[2], and P. McKenzie[1]

1: Université de Montréal

2: ENS CACHAN

Tübingen, September 22th, 2011

Parikh Automata and Friends

On the Expressiveness of Parikh Automata and Related Models (NCMA11)

Bounded Parikh Automata (WORDS11)

M. Cadilhac[1], A. Finkel[2], and P. McKenzie[1]

1: Université de Montréal

2: ENS CACHAN

Tübingen, S … er 22th, 2011

Parikh Automata and Friends

On the Expressiveness of Parikh Automata and Related Models (NCMA11)

Bounded Parikh Automata (WORDS11)

M. Cadilhac[1], A. Finkel[2], and P. McKenzie[1]

1: Université de Montréal

2: ENS Cachan

Tübingen, S er 22th

Context

A practical problem:

- In model checking, properties described by automata

Context

A practical problem:

- In model checking, properties described by automata
- Efficiency depends, in particular, on determinism

Context

A practical problem:

- In model checking, properties described by automata
- Efficiency depends, in particular, on determinism
- Finite automata, CFL offer poor counting expressiveness

Context

A practical problem:

- In model checking, properties described by automata
- Efficiency depends, in particular, on determinism
- Finite automata, CFL offer poor counting expressiveness
- One goal: provide good "deterministic" models

Context

A practical problem:

- In model checking, properties described by automata
- Efficiency depends, in particular, on determinism
- Finite automata, CFL offer poor counting expressiveness
- One goal: provide good “deterministic” models

Natural models of automata allow for a study of:

- Small complexity classes (within NC^2)

Context

A practical problem:

- In model checking, properties described by automata
- Efficiency depends, in particular, on determinism
- Finite automata, CFL offer poor counting expressiveness
- One goal: provide good "deterministic" models

Natural models of automata allow for a study of:

- Small complexity classes (within NC^2)
- Word logics (variants of MSO)

Context

A practical problem:

- In model checking, properties described by automata
- Efficiency depends, in particular, on determinism
- Finite automata, CFL offer poor counting expressiveness
- One goal: provide good "deterministic" models

Natural models of automata allow for a study of:

- Small complexity classes (within NC^2)
- Word logics (variants of MSO)
- Algebra (pseudovarieties of monoids associated with language varieties)

Outline

Parikh Automata

Affine Parikh Automata

Letter Parikh Automata

Bounded languages of PA

Corollaries and Further Work

Outline

Parikh Automata

Affine Parikh Automata

Letter Parikh Automata

Bounded languages of PA

Corollaries and Further Work

Parikh automata

Preliminary: semilinear sets

Definitions

- *Linear* set: of the form $E = \{\vec{c}_0 + \sum_{i=1}^{m} \vec{c}_i \cdot k_i \mid k_i \in \mathbb{N}\}$

Parikh automata

Preliminary: semilinear sets

Definitions

- *Linear* set: of the form $E = \{\vec{c}_0 + \sum_{i=1}^{m} \vec{c}_i \cdot k_i \mid k_i \in \mathbb{N}\}$

  E.g., $\left\{\begin{pmatrix}0\\0\end{pmatrix} + \begin{pmatrix}1\\2\end{pmatrix} \cdot k_1\right\}$ linear

Parikh automata

Preliminary: semilinear sets

Definitions

- *Linear* set: of the form $E = \{\vec{c}_0 + \sum_{i=1}^{m} \vec{c}_i \cdot k_i \mid k_i \in \mathbb{N}\}$

  E.g., $\left\{\begin{pmatrix}0\\0\end{pmatrix} + \begin{pmatrix}1\\2\end{pmatrix} \cdot k_1\right\}$ linear, $\{2^n \mid n \in \mathbb{N}\}$ not linear

Parikh automata

Preliminary: semilinear sets

Definitions

- *Linear* set: of the form $E = \{\vec{c}_0 + \sum_{i=1}^{m} \vec{c}_i \cdot k_i \mid k_i \in \mathbb{N}\}$

  E.g., $\left\{\begin{pmatrix}0\\0\end{pmatrix} + \begin{pmatrix}1\\2\end{pmatrix} \cdot k_1\right\}$ linear, $\{2^n \mid n \in \mathbb{N}\}$ not linear
- *Semilinear* set: finite union of linear sets (equiv. FO[+])

Parikh automata

Preliminary: semilinear sets

Definitions

- *Linear* set: of the form $E = \{\vec{c}_0 + \sum_{i=1}^{m} \vec{c}_i \cdot k_i \mid k_i \in \mathbb{N}\}$

  E.g., $\left\{\begin{pmatrix}0\\0\end{pmatrix} + \begin{pmatrix}1\\2\end{pmatrix} \cdot k_1\right\}$ linear, $\{2^n \mid n \in \mathbb{N}\}$ not linear
- *Semilinear* set: finite union of linear sets (equiv. FO[+])

Parikh automata

Preliminary: semilinear sets

Definitions

- *Linear* set: of the form $E = \{\vec{c_0} + \sum_{i=1}^{m} \vec{c_i} \cdot k_i \mid k_i \in \mathbb{N}\}$

  E.g., $\left\{\begin{pmatrix}0\\0\end{pmatrix} + \begin{pmatrix}1\\2\end{pmatrix} \cdot k_1\right\}$ linear, $\{2^n \mid n \in \mathbb{N}\}$ not linear
- *Semilinear* set: finite union of linear sets (equiv. FO[+])

Why are they natural? One of many reasons:

Theorem ([Parikh, 1966])

With $\Sigma = \{a_1, \ldots, a_n\}$ *and* $w \in \Sigma^*$, *let*
$\text{Parikh}(w) = (|w|_{a_1}, \ldots, |w|_{a_n}) \in \mathbb{N}^n$ *be the Parikh image of* w.

Parikh automata

Preliminary: semilinear sets

Definitions

- *Linear* set: of the form $E = \{\vec{c_0} + \sum_{i=1}^{m} \vec{c_i} \cdot k_i \mid k_i \in \mathbb{N}\}$

  E.g., $\left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 2 \end{pmatrix} \cdot k_1 \right\}$ linear, $\{2^n \mid n \in \mathbb{N}\}$ not linear
- *Semilinear* set: finite union of linear sets (equiv. FO[+])

Why are they natural? One of many reasons:

Theorem ([Parikh, 1966])

With $\Sigma = \{a_1, \ldots, a_n\}$ *and* $w \in \Sigma^*$, *let*
$\text{Parikh}(w) = (|w|_{a_1}, \ldots, |w|_{a_n}) \in \mathbb{N}^n$ *be the Parikh image of* w.

L *context-free* $\Rightarrow$ $\text{Parikh}(L)$ *semilinear*

Start with an example

Parikh automaton: semilinear constraint on Parikh(a path)

Start with an example

Parikh automaton: semilinear constraint on Parikh(a path)

Example

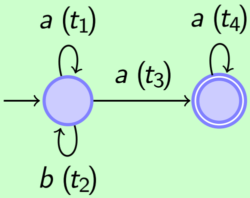
$$L = \{w \cdot a^{|w|+1} \mid w \in \{a,b\}^*\} \in \text{PA}$$

Start with an example

Parikh automaton: semilinear constraint on Parikh(a path)

Example

$$L = \{w \cdot a^{|w|+1} \mid w \in \{a,b\}^*\} \in \text{PA}$$

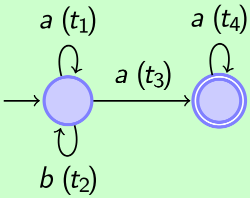


Start with an example

Parikh automaton: semilinear constraint on Parikh(a path)

Example

$$L = \{w \cdot a^{|w|+1} \mid w \in \{a,b\}^*\} \in \text{PA}$$



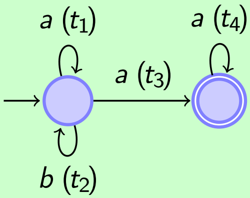
$\delta = \{t_1, t_2, t_3, t_4\}$

Start with an example

Parikh automaton: semilinear constraint on Parikh(a path)

Example

$$L = \{w \cdot a^{|w|+1} \mid w \in \{a, b\}^*\} \in \text{PA}$$



$$\delta = \{t_1, t_2, t_3, t_4\}$$

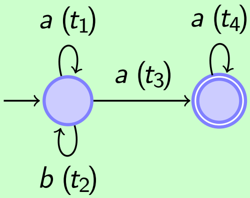
$$C = \{$$

Start with an example

Parikh automaton: semilinear constraint on Parikh(a path)

Example

$$L = \{w \cdot a^{|w|+1} \mid w \in \{a, b\}^*\} \in \text{PA}$$



$$\delta = \{t_1, t_2, t_3, t_4\}$$

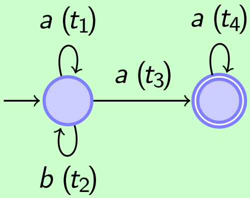
$$C = \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \cdot k_1 + \right.$$

Start with an example

Parikh automaton: semilinear constraint on Parikh(a path)

Example

$$L = \{w \cdot a^{|w|+1} \mid w \in \{a,b\}^*\} \in \mathrm{PA}$$



$$\delta = \{t_1, t_2, t_3, t_4\}$$

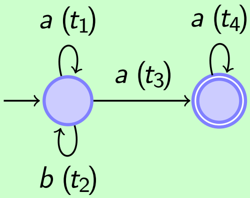
$$C = \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \cdot k_1 + \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \cdot k_2 + \right.$$

Start with an example

Parikh automaton: semilinear constraint on Parikh(a path)

Example

$$L = \{w \cdot a^{|w|+1} \mid w \in \{a,b\}^*\} \in \text{PA}$$



$$\delta = \{t_1, t_2, t_3, t_4\}$$

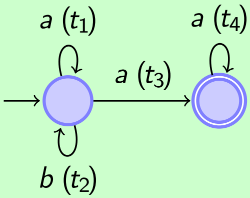
$$C = \left\{ \begin{pmatrix}1\\0\\0\\1\end{pmatrix} \cdot k_1 + \begin{pmatrix}0\\1\\0\\1\end{pmatrix} \cdot k_2 + \begin{pmatrix}0\\0\\1\\0\end{pmatrix} \right\}$$

Start with an example

Parikh automaton: semilinear constraint on Parikh(a path)

Example

$$L = \{w \cdot a^{|w|+1} \mid w \in \{a,b\}^*\} \in \text{PA}$$



$$\delta = \{t_1, t_2, t_3, t_4\}$$

$$C = \left\{ \begin{pmatrix}1\\0\\0\\1\end{pmatrix} \cdot k_1 + \begin{pmatrix}0\\1\\0\\1\end{pmatrix} \cdot k_2 + \begin{pmatrix}0\\0\\1\\0\end{pmatrix} \right\}$$

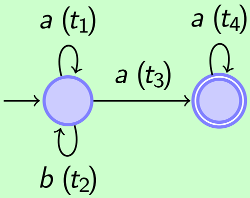
Word: *aabaaaa*

Start with an example

Parikh automaton: semilinear constraint on Parikh(a path)

Example

$$L = \{w \cdot a^{|w|+1} \mid w \in \{a,b\}^*\} \in \mathrm{PA}$$



$$\delta = \{t_1, t_2, t_3, t_4\}$$

$$C = \left\{ \begin{pmatrix}1\\0\\0\\1\end{pmatrix} \cdot k_1 + \begin{pmatrix}0\\1\\0\\1\end{pmatrix} \cdot k_2 + \begin{pmatrix}0\\0\\1\\0\end{pmatrix} \right\}$$

Word: *aabaaaa*

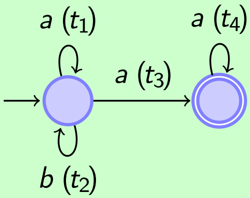
Parikh(traced run):
$\pi =$

Start with an example

Parikh automaton: semilinear constraint on Parikh(a path)

Example

$$L = \{w \cdot a^{|w|+1} \mid w \in \{a,b\}^*\} \in \mathrm{PA}$$



$$\delta = \{t_1, t_2, t_3, t_4\}$$

$$C = \left\{ \begin{pmatrix}1\\0\\0\\1\end{pmatrix} \cdot k_1 + \begin{pmatrix}0\\1\\0\\1\end{pmatrix} \cdot k_2 + \begin{pmatrix}0\\0\\1\\0\end{pmatrix} \right\}$$

Word: *aabaaaa* Parikh(traced run): (0, 0, 0, 0)

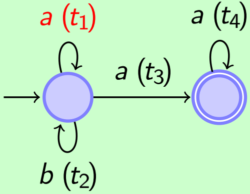
$\pi =$

Start with an example

Parikh automaton: semilinear constraint on Parikh(a path)

Example

$$L = \{w \cdot a^{|w|+1} \mid w \in \{a, b\}^*\} \in \text{PA}$$



$$\delta = \{t_1, t_2, t_3, t_4\}$$

$$C = \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \cdot k_1 + \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \cdot k_2 + \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \right\}$$

Word: *aabaaaa* ▲

Parikh(traced run): (1, 0, 0, 0)

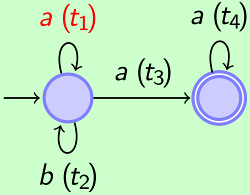
$\pi = t_1$

Start with an example

Parikh automaton: semilinear constraint on Parikh(a path)

Example

$$L = \{w \cdot a^{|w|+1} \mid w \in \{a, b\}^*\} \in \text{PA}$$



$$\delta = \{t_1, t_2, t_3, t_4\}$$

$$C = \left\{ \begin{pmatrix}1\\0\\0\\1\end{pmatrix} \cdot k_1 + \begin{pmatrix}0\\1\\0\\1\end{pmatrix} \cdot k_2 + \begin{pmatrix}0\\0\\1\\0\end{pmatrix} \right\}$$

Word: *aabaaaa* ▲

Parikh(traced run): (2, 0, 0, 0)

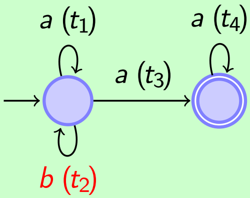
$\pi = t_1 t_1$

Start with an example

Parikh automaton: semilinear constraint on Parikh(a path)

Example

$$L = \{w \cdot a^{|w|+1} \mid w \in \{a, b\}^*\} \in \text{PA}$$



$$\delta = \{t_1, t_2, t_3, t_4\}$$

$$C = \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \cdot k_1 + \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \cdot k_2 + \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \right\}$$

Word: *aabaaaa*

Parikh(traced run): (2, 1, 0, 0)

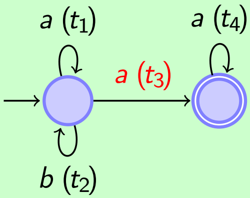
$\pi = t_1 t_1 t_2$

Start with an example

Parikh automaton: semilinear constraint on Parikh(a path)

Example

$$L = \{ w \cdot a^{|w|+1} \mid w \in \{a, b\}^* \} \in \text{PA}$$



$$\delta = \{t_1, t_2, t_3, t_4\}$$

$$C = \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \cdot k_1 + \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \cdot k_2 + \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \right\}$$

Word: *aabaaaa*

Parikh(traced run): (2, 1, 1, 0)

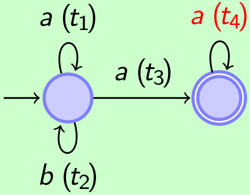
$\pi = t_1 t_1 t_2 t_3$

Start with an example

Parikh automaton: semilinear constraint on Parikh(a path)

Example

$$L = \{w \cdot a^{|w|+1} \mid w \in \{a, b\}^*\} \in \mathrm{PA}$$



$$\delta = \{t_1, t_2, t_3, t_4\}$$

$$C = \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \cdot k_1 + \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \cdot k_2 + \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \right\}$$

Word: *aabaaaa*

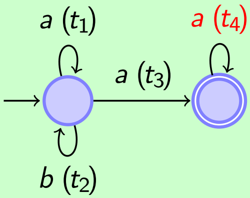
Parikh(traced run): (2, 1, 1, 1)
$\pi = t_1 t_1 t_2 t_3 t_4$

Start with an example

Parikh automaton: semilinear constraint on Parikh(a path)

Example

$$L = \{ w \cdot a^{|w|+1} \mid w \in \{a, b\}^* \} \in \text{PA}$$



$$\delta = \{ t_1, t_2, t_3, t_4 \}$$

$$C = \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \cdot k_1 + \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \cdot k_2 + \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \right\}$$

Word: *aabaaaa*

Parikh(traced run): (2, 1, 1, 2)

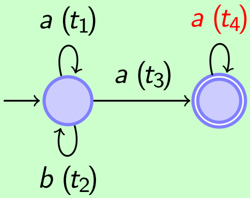
$\pi = t_1 t_1 t_2 t_3 t_4 t_4$

Start with an example

Parikh automaton: semilinear constraint on Parikh(a path)

Example

$$L = \{w \cdot a^{|w|+1} \mid w \in \{a, b\}^*\} \in \text{PA}$$



$$\delta = \{t_1, t_2, t_3, t_4\}$$

$$C = \left\{ \begin{pmatrix}1\\0\\0\\1\end{pmatrix} \cdot k_1 + \begin{pmatrix}0\\1\\0\\1\end{pmatrix} \cdot k_2 + \begin{pmatrix}0\\0\\1\\0\end{pmatrix} \right\}$$

Word: *aabaaaa*

Parikh(traced run): (2, 1, 1, 3)

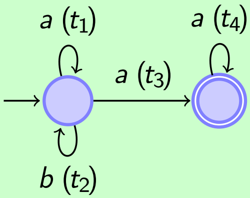
$\pi = t_1 t_1 t_2 t_3 t_4 t_4 t_4$

Start with an example

Parikh automaton: semilinear constraint on Parikh(a path)

Example

$$L = \{w \cdot a^{|w|+1} \mid w \in \{a, b\}^*\} \in \text{PA}$$



$$\delta = \{t_1, t_2, t_3, t_4\}$$

$$C = \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \cdot k_1 + \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \cdot k_2 + \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \right\}$$

Word: *aabaaaa*

Parikh(traced run): $(2, 1, 1, 3) \in C$

$\pi = t_1 t_1 t_2 t_3 t_4 t_4 t_4$

Parikh automata

The formal definition

Definition ([Klaedtke and Rueß, 2003])

- *Parikh automaton* (PA): a pair (A, C) with:
  - A a finite automaton of transition set δ
  - $C \subseteq \mathbb{N}^{|\delta|}$ semilinear

Parikh automata

The formal definition

Definition ([Klaedtke and Rueß, 2003])

- *Parikh automaton* (PA): a pair (A, C) with:
  - A a finite automaton of transition set δ
  - $C \subseteq \mathbb{N}^{|\delta|}$ semilinear
- $L(A, C) = \{\text{Label}(\pi) \mid \pi \in \text{Runs}(A) \wedge \text{Parikh}(\pi) \in C\}$

Parikh automata

The formal definition

Definition ([Klaedtke and Rueß, 2003])

- *Parikh automaton* (PA): a pair (A, C) with:
  - A a finite automaton of transition set δ
  - $C \subseteq \mathbb{N}^{|\delta|}$ semilinear
- $L(A, C) = \{\text{Label}(\pi) \mid \pi \in \text{Runs}(A) \wedge \text{Parikh}(\pi) \in C\}$
- *Deterministic Parikh automaton* (DetPA) if A is

Parikh automata

The formal definition

Definition ([Klaedtke and Rueß, 2003])

- *Parikh automaton* (PA): a pair (A, C) with:
  - A a finite automaton of transition set δ
  - $C \subseteq \mathbb{N}^{|\delta|}$ semilinear
- $L(A, C) = \{\text{Label}(\pi) \mid \pi \in \text{Runs}(A) \wedge \text{Parikh}(\pi) \in C\}$
- *Deterministic Parikh automaton* (DetPA) if A is

Parikh automata

The formal definition

Definition ([Klaedtke and Rueß, 2003])

- *Parikh automaton* (PA): a pair (A, C) with:
  - A a finite automaton of transition set δ
  - $C \subseteq \mathbb{N}^{|\delta|}$ semilinear
- $L(A, C) = \{\text{Label}(\pi) \mid \pi \in \text{Runs}(A) \wedge \text{Parikh}(\pi) \in C\}$
- *Deterministic Parikh automaton* (DetPA) if A is

- $L = \{w \cdot a^{|w|+1}\} \in \text{PA}$ but $\notin$ DetPA

Parikh automata

The formal definition

Definition ([Klaedtke and Rueß, 2003])

- *Parikh automaton* (PA): a pair (A, C) with:
  - A a finite automaton of transition set δ
  - $C \subseteq \mathbb{N}^{|\delta|}$ semilinear
- $L(A, C) = \{\text{Label}(\pi) \mid \pi \in \text{Runs}(A) \wedge \text{Parikh}(\pi) \in C\}$
- *Deterministic Parikh automaton* (DetPA) if A is

- $L = \{w \cdot a^{|w|+1}\} \in$ PA but $\notin$ DetPA, same for $\overline{\text{PAL}}$

Parikh automata

The formal definition

Definition ([Klaedtke and Rueß, 2003])

- *Parikh automaton* (PA): a pair (A, C) with:
  - A a finite automaton of transition set δ
  - $C \subseteq \mathbb{N}^{|\delta|}$ semilinear
- $L(A, C) = \{\text{Label}(\pi) \mid \pi \in \text{Runs}(A) \wedge \text{Parikh}(\pi) \in C\}$
- *Deterministic Parikh automaton* (DetPA) if A is

- $L = \{w \cdot a^{|w|+1}\} \in$ PA but $\notin$ DetPA, same for $\overline{\text{PAL}}$
- PAL $\notin$ PA

Parikh automata

Definition: Parikh automata

Why is PA a relevant model?

Parikh automata

Definition: Parikh automata

Why is PA a relevant model?

- Nice decidability properties (emptiness)

Parikh automata

Definition: Parikh automata

Why is PA a relevant model?

- Nice decidability properties (emptiness)
- Nice closure properties for DetPA

Parikh automata

Definition: Parikh automata

Why is PA a relevant model?

- Nice decidability properties (emptiness)
- Nice closure properties for DetPA
- Equivalent to reversal bounded counter machines (RBCM) of [Ibarra, 1978]

Parikh automata

Definition: Parikh automata

Why is PA a relevant model?

- Nice decidability properties (emptiness)
- Nice closure properties for DetPA
- Equivalent to reversal bounded counter machines (RBCM) of [Ibarra, 1978]
- Equivalent to extended automata over $(\mathbb{Z}^k, +, 0)$ of [Mitrana and Stiebe, 2001]

Parikh automata

Definition: Parikh automata

Why is PA a relevant model?

- Nice decidability properties (emptiness)
- Nice closure properties for DetPA
- Equivalent to reversal bounded counter machines (RBCM) of [Ibarra, 1978]
- Equivalent to extended automata over $(\mathbb{Z}^k, +, 0)$ of [Mitrana and Stiebe, 2001]
- Related deterministic models used for model checking

Parikh automata

Definition: Parikh automata

Why is PA a relevant model?

- Nice decidability properties (emptiness)
- Nice closure properties for DetPA
- Equivalent to reversal bounded counter machines (RBCM) of [Ibarra, 1978]
- Equivalent to extended automata over $(\mathbb{Z}^k, +, 0)$ of [Mitrana and Stiebe, 2001]
- Related deterministic models used for model checking
- Logical characterization: WS1S + cardinalities — or ∃MSOunC

Parikh automata

Definition: Parikh automata

Why is PA a relevant model?

- Nice decidability properties (emptiness)
- Nice closure properties for DetPA
- Equivalent to reversal bounded counter machines (RBCM) of [Ibarra, 1978]
- Equivalent to extended automata over $(\mathbb{Z}^k, +, 0)$ of [Mitrana and Stiebe, 2001]
- Related deterministic models used for model checking
- Logical characterization: WS1S + cardinalities — or ∃MSOunC
- Low complexity: PA $\subsetneq$ NL, DetPA $\subsetneq$ NC^1

Parikh automata

Definition: Parikh automata

Why is PA a relevant model?

- Nice decidability properties (emptiness)
- Nice closure properties for DetPA
- Equivalent to reversal bounded counter machines (RBCM) of [Ibarra, 1978]
- Equivalent to extended automata over $(\mathbb{Z}^k, +, 0)$ of [Mitrana and Stiebe, 2001]
- Related deterministic models used for model checking
- Logical characterization: WS1S + cardinalities — or ∃MSOunC
- Low complexity: PA $\subsetneq$ NL, DetPA $\subsetneq$ NC^1
- Algebraic characterization?

Parikh automata

Expressiveness: main (simple) tool

Lemma

For $L \in$ *PA* , $\exists p, l \in \mathbb{N}^+$ *s.t. all* $w \in L \cap \Sigma^{>l}$ *can be written* *uvxvz with:*

Parikh automata

Expressiveness: main (simple) tool

Lemma

For $L \in$ PA, $\exists p, l \in \mathbb{N}^+$ s.t. all $w \in L \cap \Sigma^{>l}$ can be written uvxvz with:

- $0 < |v| \geq p$
- $p < |x|$
- $|uvxv| < l$
- $uv^2xz, uxv^2z \in L$

- Ex.: COPY $\notin$ PA $(a^p b)^l \#(a^p b)^l$

Parikh automata

Expressiveness: main (simple) tool

Lemma

For $L \in$ DetPA, $\exists p, l \in \mathbb{N}^+$ s.t. all $w \in \Sigma^{>l}$ can be written uvxvz with:

- $0 < |v| \geq p$
- $p < |x|$
- $|uvxv| < l$
- $uv^2x \equiv uvxv \equiv uxv^2$

- Ex.: COPY $\notin$ PA $\quad (a^p b)^l \#(a^p b)^l$

Parikh automata

Expressiveness: main (simple) tool

Lemma

For $L \in$ DetPA, $\exists p, l \in \mathbb{N}^+$ s.t. all $w \in \Sigma^{>l}$ can be written uvxvz with:

- $0 < |v| \geq p$
- $p < |x|$
- $|uvxv| < l$
- $uv^2x \equiv uvxv \equiv uxv^2$

- Ex.: COPY $\notin$ PA $\quad (a^p b)^l \# (a^p b)^l$
- Ex.: $\{w \cdot a^{|w|+1}\} \notin$ DetPA $\quad (a^p b)^l$

Parikh automata

Decidability and closure properties

- Decidability results:

	$= \varnothing$	$= \Sigma^*$	is finite	$\subseteq$	is regular
DetPA	D	D	**D**	**D**	?
PA	D	U	**D**	**U**	**U**

Parikh automata

Decidability and closure properties

- Decidability results:

	$= \emptyset$	$= \Sigma^*$	is finite	$\subseteq$	is regular
DetPA	D	D	**D**	**D**	?
PA	D	U	**D**	**U**	**U**

- Closure properties:

	$\cup$	$\cap$	$\cdot$	$\rightarrow$	h	h^{-1}	c	$*$
DetPA	Y	Y	**N**	Y	**N**	Y	**Y**	**N**
PA	Y	Y	Y	N	Y	Y	**Y**	N

Parikh automata

PA and RBCM

Definitions

- Counter machine: two-way automaton with 0-test counters

Parikh automata

PA and RBCM

Definitions

- Counter machine: two-way automaton with 0-test counters
- Reversal-bounded: constant number of back-and-forth and increment-decrement

Parikh automata

PA and RBCM

Definitions

- Counter machine: two-way automaton with 0-test counters
- Reversal-bounded: constant number of back-and-forth and increment-decrement

Parikh automata

PA and RBCM

Definitions

- Counter machine: two-way automaton with 0-test counters
- Reversal-bounded: constant number of back-and-forth and increment-decrement

Theorem

- $PA = RBCM$

Parikh automata

PA and RBCM

Definitions

- Counter machine: two-way automaton with 0-test counters
- Reversal-bounded: constant number of back-and-forth and increment-decrement

Theorem

- $PA = RBCM$
- $DetPA \subsetneq DetRBCM$

Parikh automata

PA and RBCM

Definitions

- Counter machine: two-way automaton with 0-test counters
- Reversal-bounded: constant number of back-and-forth and increment-decrement

Theorem

- $PA = RBCM$
- $DetPA \subsetneq DetRBCM$
- $NSUM = \{a^n \spadesuit b^{m_1} \# \cdots \# b^{m_k} \clubsuit c^{m_1 + \cdots + m_n}\} \in DetRBCM \setminus DetPA$

Outline

Parikh Automata

Affine Parikh Automata

Letter Parikh Automata

Bounded languages of PA

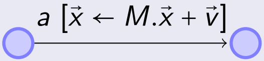
Corollaries and Further Work

Affine Parikh automata

A quick definition

Definition

- *Affine Parikh automaton* given by:
  - A finite automaton
  - A labelling of the transitions by affine functions (on $\mathbb{Q}$ or $\mathbb{N}$)
  - A semilinear set (for $\mathbb{Q}$, FO)

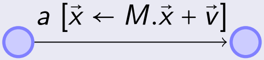


Affine Parikh automata

A quick definition

Definition

- *Affine Parikh automaton* given by:
  - A finite automaton
  - A labelling of the transitions by affine functions (on $\mathbb{Q}$ or $\mathbb{N}$)
  - A semilinear set (for $\mathbb{Q}$, FO)



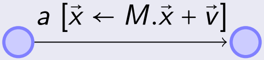
- Its language: accepted words which take $\vec{0}$ to some $\vec{x}$ in the semilinear set

Affine Parikh automata

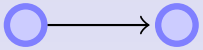
A quick definition

Definition

- *Affine Parikh automaton* given by:
  - A finite automaton
  - A labelling of the transitions by affine functions (on $\mathbb{Q}$ or $\mathbb{N}$)
  - A semilinear set (for $\mathbb{Q}$, FO)



- Its language: accepted words which take $\vec{0}$ to some $\vec{x}$ in the semilinear set



Affine Parikh automata

Expressiveness

- PA: APA in which every M = Identity

Affine Parikh automata

Expressiveness

- PA: APA in which every M = Identity
- PAL, $\{a^{2^n}\} \in$ APA

SL set

$\vec{c_0} + \sum_i \vec{c_i} \cdot k_i$

(Det)PA

automaton constraining #transitions

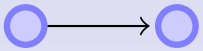
RBCM

Rev. bounded 0-test counters

(Det)APA

PA with func

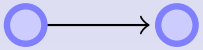
$a \ [\vec{x} \leftarrow M.\vec{x} + \vec{v}]$



Affine Parikh automata

Expressiveness

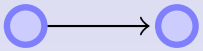
- PA: APA in which every M = Identity
- PAL, $\{a^{2^n}\} \in$ APA
- APA $\subsetneq$ NP $\cap$ CSL



Affine Parikh automata

Expressiveness

- PA: APA in which every M = Identity
- PAL, $\{a^{2^n}\} \in$ APA
- APA $\subsetneq$ NP $\cap$ CSL
- DetAPA do not need their automaton

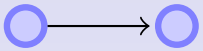


Affine Parikh automata

Expressiveness

- PA: APA in which every M = Identity
- PAL, $\{a^{2^n}\} \in$ APA
- APA $\subseteq$ NP $\cap$ CSL
- DetAPA do not need their automaton
- (Det)APA using $\mathbb{Q}$ included in $\mathbb{N}$

$a\ [\vec{x} \leftarrow M.\vec{x} + \vec{v}]$



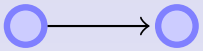
Affine Parikh automata

Expressiveness

- PA: APA in which every M = Identity
- PAL, $\{a^{2^n}\} \in$ APA
- APA $\subseteq$ NP $\cap$ CSL
- DetAPA do not need their automaton
- (Det)APA using $\mathbb{Q}$ included in $\mathbb{N}$

Open:

- Dyck $\notin$ APA
- $\{a, b\}^n a \{a, b\}^n, \{a^{2^n}\} \notin$ DetAPA
- Separate $\mathbb{Q}$-(Det)APA and $\mathbb{N}$-(Det)APA



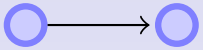
Affine Parikh automata

Expressiveness

- PA: APA in which every M = Identity
- PAL, $\{a^{2^n}\} \in$ APA
- APA $\subseteq$ NP $\cap$ CSL
- DetAPA do not need their automaton
- (Det)APA using $\mathbb{Q}$ included in $\mathbb{N}$

Open:

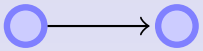
- Dyck $\notin$ APA
- $\{a, b\}^n a \{a, b\}^n, \{a^{2^n}\} \notin$ DetAPA
- Separate $\mathbb{Q}$-(Det)APA and $\mathbb{N}$-(Det)APA
- Logical characterization?
- Algebraic characterization?



Affine Parikh automata

Decidability and closure properties

- Positive properties: (Det)APA closed under $\cap$, $\cup$, h^{-1}; APA under concat, DetAPA under complement



Affine Parikh automata

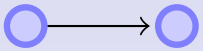
Decidability and closure properties

- Positive properties: (Det)APA closed under $\cap$, $\cup$, h^{-1}; APA under concat, DetAPA under complement

But:

Lemma

L recognizable $\rightarrow$ L morphic image of some $L' \in$ DetAPA



Affine Parikh automata

Decidability and closure properties

- Positive properties: (Det)APA closed under $\cap$, $\cup$, h^{-1}; APA under concat, DetAPA under complement

But:

Lemma

L recognizable $\rightarrow$ L morphic image of some $L' \in$ DetAPA

- (Det)APA not closed under morphism
- Emptiness, finiteness undecidable

SL set

$\vec{c_0} + \sum_i \vec{c_i} \cdot k_i$

(Det)PA

automaton constraining #transitions

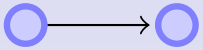
RBCM

Rev. bounded 0-test counters

(Det)APA

PA with func

$a\ [\vec{x} \leftarrow M.\vec{x} + \vec{v}]$



Outline

Parikh Automata

Affine Parikh Automata

Letter Parikh Automata

Bounded languages of PA

Corollaries and Further Work

SL set

$\vec{c_0} + \sum_i \vec{c_i} \cdot k_i$

(Det)PA

automaton constraining #transitions

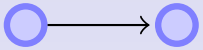
RBCM

Rev. bounded 0-test counters

(Det)APA

PA with func

$a\ [\vec{x} \leftarrow M.\vec{x} + \vec{v}]$



Letter Parikh automata

A simple restriction

Definition

PA without distinguishing transitions with same label

Letter Parikh automata

A simple restriction

Definition

PA without distinguishing transitions with same label

With $R \subseteq \Sigma^*$ regular, $C \subseteq \mathbb{N}^{|\Sigma|}$ semilinear,

$$R\restriction_C = R \cap \text{Parikh}^{-1}(C)$$

Letter Parikh automata

A simple restriction

Definition

PA without distinguishing transitions with same label
With $R \subseteq \Sigma^*$ regular, $C \subseteq \mathbb{N}^{|\Sigma|}$ semilinear,

$$R\restriction_C = R \cap \mathrm{Parikh}^{-1}(C)$$

SL set

$\vec{c_0} + \sum_i \vec{c_i} \cdot k_i$

(Det)PA

automaton constraining #transitions

RBCM

Rev. bounded 0-test counters

(Det)APA

PA with func

$a\ [\vec{x} \leftarrow M.\vec{x} + \vec{v}]$

LPA

automaton constraining #letters

Letter Parikh automata

A simple restriction

Definition

PA without distinguishing transitions with same label

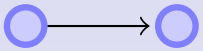
With $R \subseteq \Sigma^*$ regular, $C \subseteq \mathbb{N}^{|\Sigma|}$ semilinear,

$$R{\upharpoonright}_C = R \cap \text{Parikh}^{-1}(C)$$

Poor expressiveness:

Lemma

$R{\upharpoonright}_C \cap E$ *nonregular* $\rightarrow \text{Parikh}(E) \notin \text{Parikh}(R{\upharpoonright}_C)$



Letter Parikh automata

A simple restriction

Definition

PA without distinguishing transitions with same label

With $R \subseteq \Sigma^*$ regular, $C \subseteq \mathbb{N}^{|\Sigma|}$ semilinear,

$$R{\restriction}_C = R \cap \text{Parikh}^{-1}(C)$$

Poor expressiveness:

Lemma

$R{\restriction}_C \cap E$ *nonregular* $\rightarrow \text{Parikh}(E) \notin \text{Parikh}(R{\restriction}_C)$

Leads to:

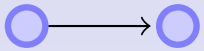
Theorem

Letter PA not closed under $\cup$, *complement, squaring, nonerasing morphism*

Letter Parikh automata

A link with PA

But PA $= h(\text{LPA})$ helps in proving Parikh-boundedness:



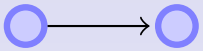
Letter Parikh automata

A link with PA

But PA $=$ h(LPA) helps in proving Parikh-boundedness:

Lemma

$L \in PA \rightarrow \exists L' \subseteq L$ *bounded with* $\text{Parikh}(L) = \text{Parikh}(L')$



Outline

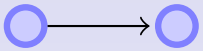
Parikh Automata

Affine Parikh Automata

Letter Parikh Automata

Bounded languages of PA

Corollaries and Further Work



Bounded languages of PA

Definition: bounded languages

Definition

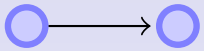
- *L bounded*: $L \subseteq w_1^* \cdots w_n^*$ for some words w_i's

Bounded languages of PA

Definition: bounded languages

Definition

- *L bounded*: $L \subseteq w_1^* \cdots w_n^*$ for some words w_i's
- Define $\text{Iter}_{\vec{w}}(L) = \{(i_1, \ldots, i_n) \mid w_1^{i_1} \cdots w_n^{i_n} \in L\} \subseteq \mathbb{N}^n$

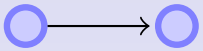


Bounded languages of PA

Definition: bounded languages

Definition

- L *bounded*: $L \subseteq w_1^* \cdots w_n^*$ for some words w_i's
- Define $\mathrm{Iter}_{\vec{w}}(L) = \{(i_1, \ldots, i_n) \mid w_1^{i_1} \cdots w_n^{i_n} \in L\} \subseteq \mathbb{N}^n$
- $\mathrm{BSL} = \{L \subseteq w_1^* \cdots w_n^* \mid \mathrm{Iter}_{\vec{w}}(L) \text{ semilinear}\}$

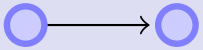


Bounded languages of PA

Definition: bounded languages

Definition

- *L bounded*: $L \subseteq w_1^* \cdots w_n^*$ for some words w_i's
- Define $\mathrm{Iter}_{\vec{w}}(L) = \{(i_1, \ldots, i_n) \mid w_1^{i_1} \cdots w_n^{i_n} \in L\} \subseteq \mathbb{N}^n$
- $\mathrm{BSL} = \{L \subseteq w_1^* \cdots w_n^* \mid \mathrm{Iter}_{\vec{w}}(L) \text{ semilinear}\}$



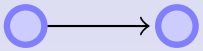
Bounded languages of PA

Definition: bounded languages

Definition

- *L bounded*: $L \subseteq w_1^* \cdots w_n^*$ for some words w_i's
- Define $\text{Iter}_{\vec{w}}(L) = \{(i_1, \ldots, i_n) \mid w_1^{i_1} \cdots w_n^{i_n} \in L\} \subseteq \mathbb{N}^n$
- $\text{BSL} = \{L \subseteq w_1^* \cdots w_n^* \mid \text{Iter}_{\vec{w}}(L) \text{ semilinear}\}$

- $\{a^i b^{2i}\} \in \text{BSL}$



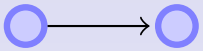
Bounded languages of PA

Definition: bounded languages

Definition

- *L bounded*: $L \subseteq w_1^* \cdots w_n^*$ for some words w_i's
- Define $\mathrm{Iter}_{\vec{w}}(L) = \{(i_1, \ldots, i_n) \mid w_1^{i_1} \cdots w_n^{i_n} \in L\} \subseteq \mathbb{N}^n$
- $\mathrm{BSL} = \{L \subseteq w_1^* \cdots w_n^* \mid \mathrm{Iter}_{\vec{w}}(L) \text{ semilinear}\}$

- $\{a^i b^{2i}\} \in \mathrm{BSL}$
- Σ^* not bounded



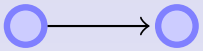
Bounded languages of PA

Definition: bounded languages

Definition

- *L bounded*: $L \subseteq w_1^* \cdots w_n^*$ for some words w_i's
- Define $\text{Iter}_{\vec{w}}(L) = \{(i_1, \ldots, i_n) \mid w_1^{i_1} \cdots w_n^{i_n} \in L\} \subseteq \mathbb{N}^n$
- $\text{BSL} = \{L \subseteq w_1^* \cdots w_n^* \mid \text{Iter}_{\vec{w}}(L) \text{ semilinear}\}$

- $\{a^i b^{2i}\} \in \text{BSL}$
- Σ^* not bounded
- $\{a^{2^n}\}$ bounded $\notin$ BSL



Bounded languages of PA

Definition: bounded languages

Definition

- *L bounded*: $L \subseteq w_1^* \cdots w_n^*$ for some words w_i's
- Define $\text{Iter}_{\vec{w}}(L) = \{(i_1, \ldots, i_n) \mid w_1^{i_1} \cdots w_n^{i_n} \in L\} \subseteq \mathbb{N}^n$
- $\text{BSL} = \{L \subseteq w_1^* \cdots w_n^* \mid \text{Iter}_{\vec{w}}(L) \text{ semilinear}\}$

- $\{a^i b^{2i}\} \in \text{BSL}$
- Σ^* not bounded
- $\{a^{2^n}\}$ bounded $\notin$ BSL

- ▸▸ BSL intensively studied, e.g., Ginsburg & Spanier, 60's

Bounded languages of PA

. . . are determinizable

Theorem

Parikh automata and their deterministic variant recognize the same bounded languages: those with a semilin iteration set

Bounded languages of PA

. . . are determinizable

Theorem

Parikh automata and their deterministic variant recognize the same bounded languages: those with a semilin iteration set

Theorem, restated:

Theorem

$$\text{PA} \cap \text{BOUNDED} \quad \subseteq \quad \text{BSL} \quad \subseteq \quad \text{DetPA} \cap \text{BOUNDED}$$

SL set

$\vec{c_0} + \sum_i \vec{c_i} \cdot k_i$

(Det)PA

automaton constraining #transitions

RBCM

Rev. bounded 0-test counters

(Det)APA

PA with func

$a\ [\vec{x} \leftarrow M.\vec{x} + \vec{v}]$

LPA

automaton constraining #letters

Bounded

$L \subseteq w_1^* \cdots w_n^*$

Iter. set

$\{\vec{i} \mid \vec{w}^{\vec{i}} \in L\}$

BSL

Bounded ∧ SL iter. set

Bounded languages of PA

. . . are determinizable

Theorem

Parikh automata and their deterministic variant recognize the same bounded languages: those with a semilin iteration set

Theorem, restated:

Theorem

$\mathrm{PA} \cap \mathrm{BOUNDED} \quad \subseteq \quad \mathrm{BSL} \quad \subseteq \quad \mathrm{DetPA} \cap \mathrm{BOUNDED}$

Parikh(*any L in PA*) *semilinear*

PA closed under h^{-1}, $\cap$

Bounded languages of PA

. . . are determinizable

Theorem

Parikh automata and their deterministic variant recognize the same bounded languages: those with a semilin iteration set

Theorem, restated:

Theorem

$\text{PA} \cap \text{BOUNDED} \subsetneq \text{BSL} \subsetneq \text{DetPA} \cap \text{BOUNDED}$

Parikh(*any L in PA*) *semilinear*
PA closed under h^{-1}, $\cap$

Rest of this section

SL set

$\vec{c_0} + \sum_i \vec{c_i} \cdot k_i$

(Det)PA

automaton constraining #transitions

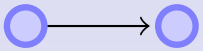
RBCM

Rev. bounded 0-test counters

(Det)APA

PA with func

$a\ [\vec{x} \leftarrow M.\vec{x} + \vec{v}]$



LPA

automaton constraining #letters

Bounded

$L \subseteq w_1^* \cdots w_n^*$

Iter. set

$\{\vec{i} \mid \vec{w}^{\vec{i}} \in L\}$

BSL

Bounded ∧ SL iter. set

Outline

Parikh Automata

Affine Parikh Automata

Letter Parikh Automata

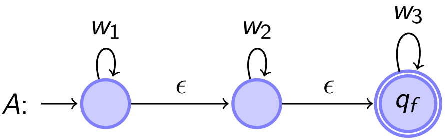
Bounded languages of PA

- BSL ⊆ ℕ-DetAPA with some property X
- ℕ-DetAPA with this property ⊆ DetPA
- Corollaries

Corollaries and Further Work

BSL ⊆ ℕ-DetAPA with some property X

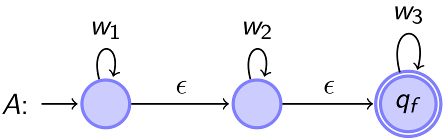
- Let $L \subseteq w_1^* w_2^* w_3^* \in$ BSL. Then a PA describes L:



and $C = \text{Iter}_{\vec{w}}(L)$

BSL ⊆ ℕ-DetAPA with some property X

- Let $L \subseteq w_1^* w_2^* w_3^* \in$ BSL. Then a PA describes L:



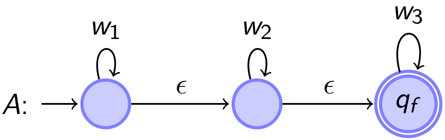
and $C = \text{Iter}_{\vec{w}}(L)$

- Let π_1, π_2 two accepting paths with same label:

$$\text{Parikh}(\pi_1) \in C \Leftrightarrow \text{Parikh}(\pi_2) \in C$$

BSL $\subseteq$ $\mathbb{N}$-DetAPA with some property X

- Let $L \subseteq w_1^* w_2^* w_3^* \in$ BSL. Then a PA describes L:



and $C = \mathrm{Iter}_{\vec{w}}(L)$

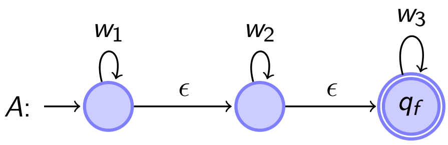
- Let π_1, π_2 two accepting paths with same label:

$$\mathrm{Parikh}(\pi_1) \in C \Leftrightarrow \mathrm{Parikh}(\pi_2) \in C$$

- In SubsetDeterminize(A), when reaching a final state $\{q_f, \ldots\}$, we need only recall the Parikh image of *one* possible run to q_f

BSL ⊆ ℕ-DetAPA with some property X

- Let $L \subseteq w_1^* w_2^* w_3^* \in \text{BSL}$. Then a PA describes L:

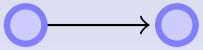


and $C = \text{Iter}_{\vec{w}}(L)$

- Let π_1, π_2 two accepting paths with same label:

$$\text{Parikh}(\pi_1) \in C \Leftrightarrow \text{Parikh}(\pi_2) \in C$$

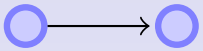
- In SubsetDeterminize(A), when reaching a final state $\{q_f, \ldots\}$, we need only recall the Parikh image of *one* possible run to q_f
- In SubsetDeterminize(A), when reaching some state $\{q_1, \ldots, q_k\}$, we need only recall the Parikh image of *one* possible run to each q_i



BSL ⊆ $\mathbb{N}$-DetAPA with some property X

Thus we do the following:

1. Let (A, C) be the PA for $L \in$ BSL



BSL ⊆ $\mathbb{N}$-DetAPA with some property X

Thus we do the following:

1. Let (A, C) be the PA for $L \in$ BSL
2. Determinize A by the subset construction

BSL ⊆ ℕ-DetAPA with some property X

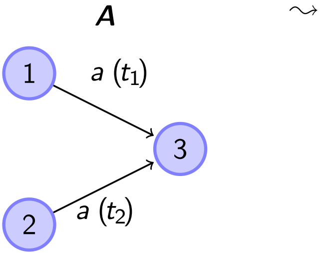
Thus we do the following:

1. Let (A, C) be the PA for $L \in \text{BSL}$
2. Determinize A by the subset construction
3. Associate functions to compute the Parikh image:

BSL ⊆ $\mathbb{N}$-DetAPA with some property X

Thus we do the following:

1. Let (A, C) be the PA for $L \in$ BSL
2. Determinize A by the subset construction
3. Associate functions to compute the Parikh image:

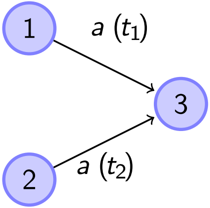


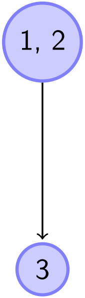
BSL ⊆ ℕ-DetAPA with some property X

Thus we do the following:

1. Let (A, C) be the PA for $L \in$ BSL
2. Determinize A by the subset construction
3. Associate functions to compute the Parikh image:

$\boldsymbol{A}$ $\rightsquigarrow$ SubsetDeterminize($\boldsymbol{A}$)



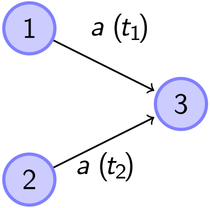


BSL ⊆ ℕ-DetAPA with some property X

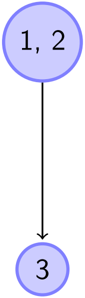
Thus we do the following:

1. Let (A, C) be the PA for $L \in$ BSL
2. Determinize A by the subset construction
3. Associate functions to compute the Parikh image:

$\boldsymbol{A}$ ⇝ **SubsetDeterminize(A)**



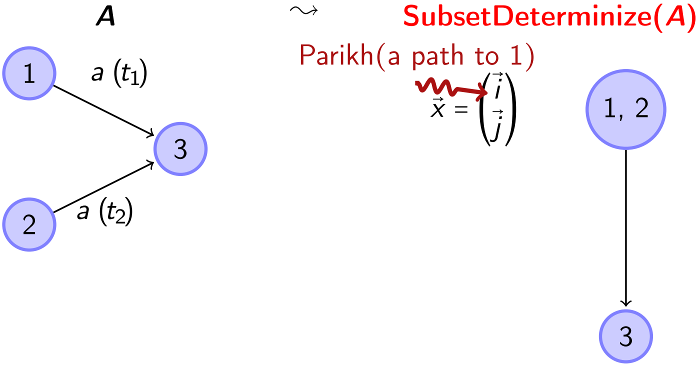
$$\vec{x} = \begin{pmatrix} \vec{i} \\ \vec{j} \end{pmatrix}$$



BSL ⊆ ℕ-DetAPA with some property X

Thus we do the following:

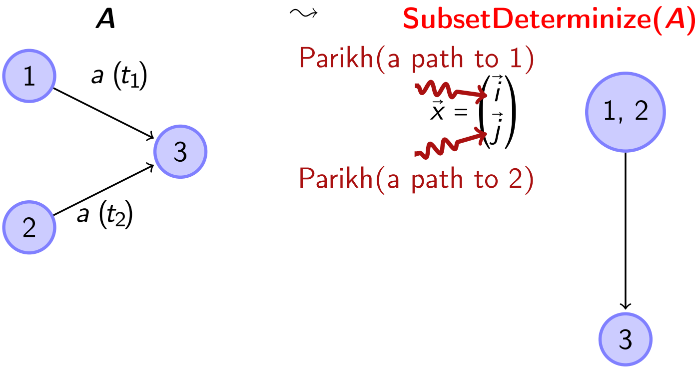
1. Let (A, C) be the PA for $L \in$ BSL
2. Determinize A by the subset construction
3. Associate functions to compute the Parikh image:



BSL ⊆ ℕ-DetAPA with some property X

Thus we do the following:

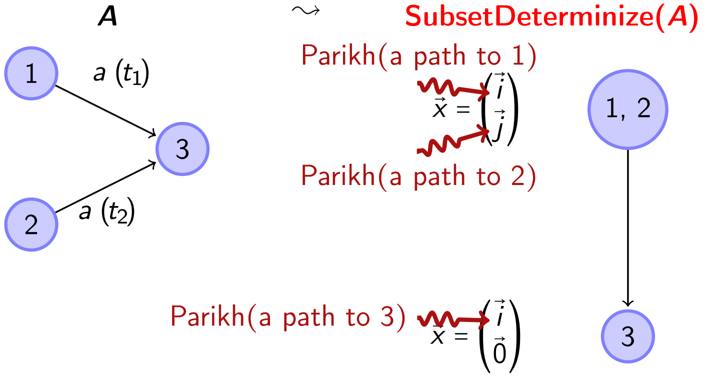
1. Let (A, C) be the PA for $L \in$ BSL
2. Determinize A by the subset construction
3. Associate functions to compute the Parikh image:



BSL ⊆ ℕ-DetAPA with some property X

Thus we do the following:

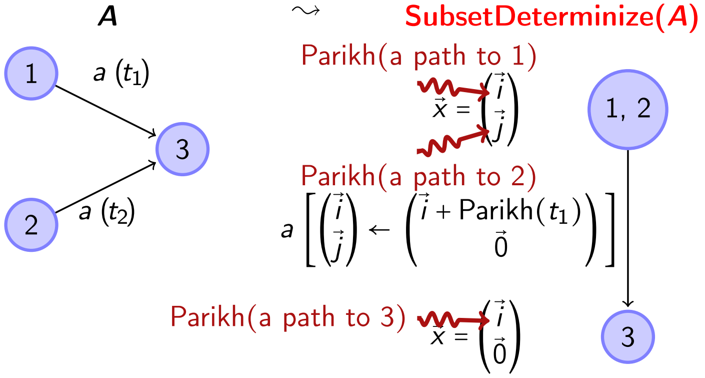
1. Let (A, C) be the PA for $L \in$ BSL
2. Determinize A by the subset construction
3. Associate functions to compute the Parikh image:



BSL ⊆ ℕ-DetAPA with some property X

Thus we do the following:

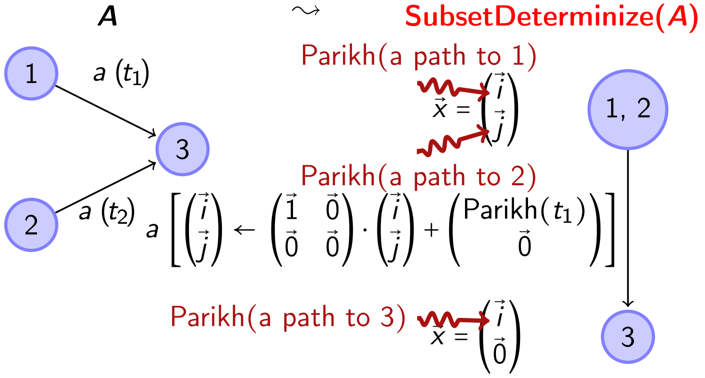
1. Let (A, C) be the PA for $L \in \mathrm{BSL}$
2. Determinize A by the subset construction
3. Associate functions to compute the Parikh image:



BSL ⊆ ℕ-DetAPA with some property X

Thus we do the following:

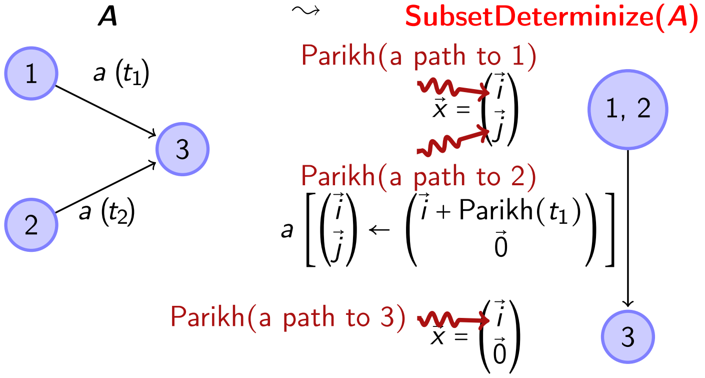
1. Let (A, C) be the PA for $L \in$ BSL
2. Determinize A by the subset construction
3. Associate functions to compute the Parikh image:



BSL ⊆ ℕ-DetAPA with some property X

Thus we do the following:

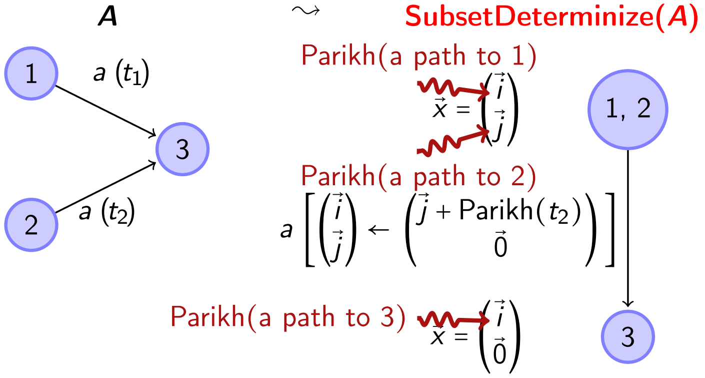
1. Let (A, C) be the PA for $L \in$ BSL
2. Determinize A by the subset construction
3. Associate functions to compute the Parikh image:

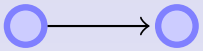


BSL ⊆ ℕ-DetAPA with some property X

Thus we do the following:

1. Let (A, C) be the PA for $L \in$ BSL
2. Determinize A by the subset construction
3. Associate functions to compute the Parikh image:





Outline

Parikh Automata

Affine Parikh Automata

Letter Parikh Automata

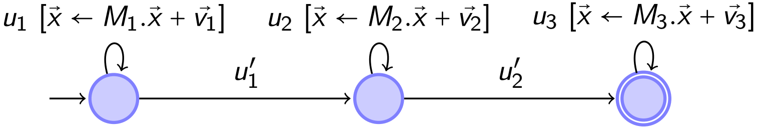
Bounded languages of PA

- BSL ⊆ ℕ-DetAPA with some property X
- ℕ-DetAPA with this property ⊆ DetPA
- Corollaries

Corollaries and Further Work

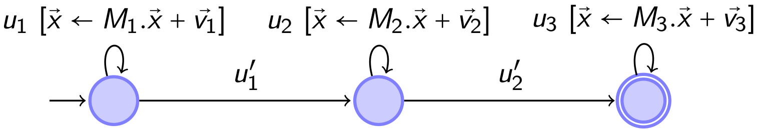
$\mathbb{N}$-DetAPA with the property X ⊆ DetPA

- We can assume the DetAPA, of language L, is of the form:



$\mathbb{N}$-DetAPA with the property X ⊆ DetPA

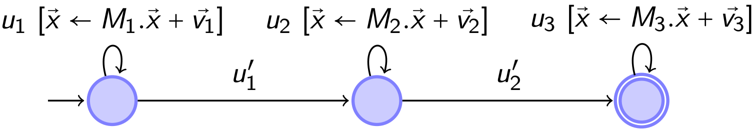
- We can assume the DetAPA, of language L, is of the form:



- Property X: For all i, there are p_i, k_i s.t. $M_i^{p_i} = M_i^{p_i+k_i}$

ℕ-DetAPA with the property X ⊆ DetPA

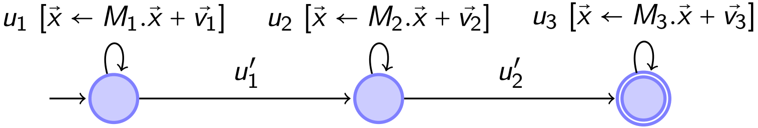
- We can assume the DetAPA, of language L, is of the form:



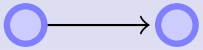
- Property X: For all i, there are p_i, k_i s.t. $M_i^{p_i} = M_i^{p_i+k_i}$
- For any $\vec{a} \in \{p_i, \ldots, p_i + k_i\}_{\{1,2,3\}}$, we give a DetPA for $L \cap (u_1^{a_1})(u_1^{k_1})^* \cdot u'_1 \cdot (u_2^{a_2})(u_2^{k_2})^* \cdot u'_2 \cdot (u_3^{a_3})(u_3^{k_3})^*$

$\mathbb{N}$-DetAPA with the property X ⊆ DetPA

- We can assume the DetAPA, of language L, is of the form:

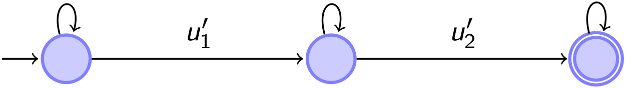


- Property X: For all i, there are p_i, k_i s.t. $M_i^{p_i} = M_i^{p_i+k_i}$
- For any $\vec{a} \in \{p_i, \ldots, p_i + k_i\}_{\{1,2,3\}}$, we give a DetPA for $L \cap (u_1^{a_1})(u_1^{k_1})^* \cdot u_1' \cdot (u_2^{a_2})(u_2^{k_2})^* \cdot u_2' \cdot (u_3^{a_3})(u_3^{k_3})^*$
- ▸▸ Construction main idea: suppose a word contains t times $u_2^{k_2}$. Final value of $\vec{x}$ contains:



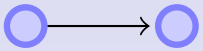
$\mathbb{N}$-DetAPA with the property X ⊆ DetPA

- We can assume the DetAPA, of language L, is of the form:



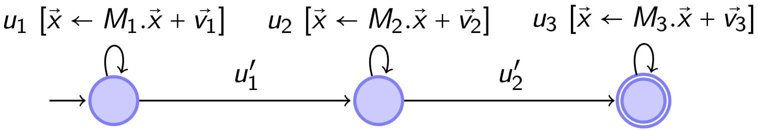
- Property X: For all i, there are p_i, k_i s.t. $M_i^{p_i} = M_i^{p_i+k_i}$
- For any $\vec{a} \in \{p_i, \ldots, p_i + k_i\}_{\{1,2,3\}}$, we give a DetPA for $L \cap (u_1^{a_1})(u_1^{k_1})^* \cdot u'_1 \cdot (u_2^{a_2})(u_2^{k_2})^* \cdot u'_2 \cdot (u_3^{a_3})(u_3^{k_3})^*$
- Construction main idea: suppose a word contains t times $u_2^{k_2}$. Final value of $\vec{x}$ contains:

$$\sum_{i=1}^{t} \cdots M_2^{i \times k_2} M_2^{a_2} \cdots$$



$\mathbb{N}$-DetAPA with the property X ⊆ DetPA

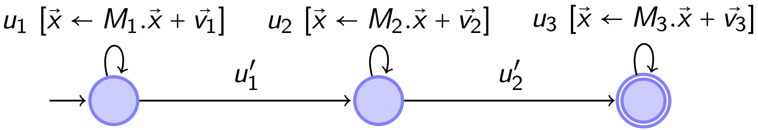
- We can assume the DetAPA, of language L, is of the form:



- Property X: For all i, there are p_i, k_i s.t. $M_i^{p_i} = M_i^{p_i+k_i}$
- For any $\vec{a} \in \{p_i, \ldots, p_i + k_i\}_{\{1,2,3\}}$, we give a DetPA for $L \cap (u_1^{a_1})(u_1^{k_1})^* \cdot u'_1 \cdot (u_2^{a_2})(u_2^{k_2})^* \cdot u'_2 \cdot (u_3^{a_3})(u_3^{k_3})^*$
  - Construction main idea: suppose a word contains t times $u_2^{k_2}$. Final value of $\vec{x}$ contains:
$$\textstyle\sum_{i=1}^{t} \cdots M_2^{i \times k_2} M_2^{a_2} \cdots = t \times \cdots M_2^{a_2} \cdots$$

$\mathbb{N}$-DetAPA with the property X ⊆ DetPA

- We can assume the DetAPA, of language L, is of the form:



- Property X: For all i, there are p_i, k_i s.t. $M_i^{p_i} = M_i^{p_i + k_i}$
- For any $\vec{a} \in \{p_i, \ldots, p_i + k_i\}_{\{1,2,3\}}$, we give a DetPA for $L \cap (u_1^{a_1})(u_1^{k_1})^* \cdot u'_1 \cdot (u_2^{a_2})(u_2^{k_2})^* \cdot u'_2 \cdot (u_3^{a_3})(u_3^{k_3})^*$
  - Construction main idea: suppose a word contains t times $u_2^{k_2}$. Final value of $\vec{x}$ contains:
    $$\sum_{i=1}^{t} \cdots M_2^{i \times k_2} M_2^{a_2} \cdots = t \times \cdots M_2^{a_2} \cdots$$
  - Contribution of $(u_2^{k_2})$ *constant*

Outline

Parikh Automata

Affine Parikh Automata

Letter Parikh Automata

Bounded languages of PA

- BSL ⊊ ℕ-DetAPA with some property X
- ℕ-DetAPA with this property ⊊ DetPA
- Corollaries

Corollaries and Further Work

Bounded languages of PA

Some corollaries

- Recall PA = RBCM; moreover DetPA $\subsetneq$ DetRBCM, thus RBCM accepting a bounded language can be determinized

SL set

$\vec{c_0} + \sum_i \vec{c_i} \cdot k_i$

(Det)PA

automaton constraining #transitions

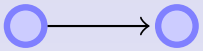
RBCM

Rev. bounded 0-test counters

(Det)APA

PA with func

$a\ [\vec{x} \leftarrow M.\vec{x} + \vec{v}]$



LPA

automaton constraining #letters

Bounded

$L \subseteq w_1^* \cdots w_n^*$

Iter. set

$\{\vec{i} \mid \vec{w}^{\vec{i}} \in L\}$

BSL

Bounded $\wedge$ SL iter. set

Bounded languages of PA

Some corollaries

- Recall PA = RBCM; moreover DetPA $\subsetneq$ DetRBCM, thus RBCM accepting a bounded language can be determinized
- In fact, the resulting DetPA has a special form: union of *flat* automata $\rightarrow$ CQDD [Bouajjani and Habermehl, 1999]; thus CQDD = BSL

SL set

$\vec{c_0} + \sum_i \vec{c_i} \cdot k_i$

(Det)PA

automaton constraining #transitions

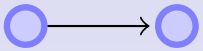
RBCM

Rev. bounded 0-test counters

(Det)APA

PA with func

$a\ [\vec{x} \leftarrow M.\vec{x} + \vec{v}]$



LPA

automaton constraining #letters

Bounded

$L \subseteq w_1^* \cdots w_n^*$

Iter. set

$\{\vec{i} \mid \vec{w}^{\vec{i}} \in L\}$

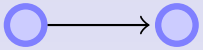
BSL

Bounded ∧ SL iter. set

Bounded languages of PA

Some corollaries

- Recall PA = RBCM; moreover DetPA $\subsetneq$ DetRBCM, thus RBCM accepting a bounded language can be determinized
- In fact, the resulting DetPA has a special form: union of *flat* automata → CQDD [Bouajjani and Habermehl, 1999]; thus CQDD = BSL
- Properties of BSL: closure under concat, under h^{-1} ∩ BSL,



Bounded languages of PA

Some corollaries

- Recall PA = RBCM; moreover DetPA $\subsetneq$ DetRBCM, thus RBCM accepting a bounded language can be determinized
- In fact, the resulting DetPA has a special form: union of *flat* automata → CQDD [Bouajjani and Habermehl, 1999]; thus CQDD = BSL
- Properties of BSL: closure under concat, under $h^{-1} \cap$ BSL, and $L \in \text{BSL} \Rightarrow (\forall \vec{w})[L \subseteq w_1^* \cdots w_n^* \rightarrow \text{Iter}_{\vec{w}}(L) \text{ is SL}]$

Bounded languages of PA

Some corollaries

- Recall PA = RBCM; moreover DetPA $\subsetneq$ DetRBCM, thus RBCM accepting a bounded language can be determinized
- In fact, the resulting DetPA has a special form: union of *flat* automata → CQDD [Bouajjani and Habermehl, 1999]; thus CQDD = BSL
- Properties of BSL: closure under concat, under $h^{-1} \cap$ BSL, and $L \in \text{BSL} \Rightarrow (\forall \vec{w})[L \subseteq w_1^* \cdots w_n^* \rightarrow \text{Iter}_{\vec{w}}(L) \text{ is SL}]$

What implications to algebraic/logic characterizations?

Outline

Parikh Automata

Affine Parikh Automata

Letter Parikh Automata

Bounded languages of PA

Corollaries and Further Work

SL set

$\vec{c_0} + \sum_i \vec{c_i} \cdot k_i$

(Det)PA

automaton constraining #transitions

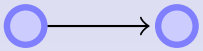
RBCM

Rev. bounded 0-test counters

(Det)APA

PA with func

$a\ [\vec{x} \leftarrow M.\vec{x} + \vec{v}]$



LPA

automaton constraining #letters

Bounded

$L \subseteq w_1^* \cdots w_n^*$

Iter. set

$\{\vec{i} \mid \vec{w}^{\vec{i}} \in L\}$

BSL

Bounded ∧ SL iter. set

Corollaries and Further Work

Thank you

Parikh Automata

Affine Parikh Automata

Letter Parikh Automata

Bounded languages of PA

- BSL $\subseteq$ $\mathbb{N}$-DetAPA with some property X
- $\mathbb{N}$-DetAPA with this property $\subseteq$ DetPA
- Corollaries

Corollaries and Further Work

Bibliography I

Bouajjani, A. and Habermehl, P. (1999).
Symbolic reachability analysis of FIFO-channel systems with nonregular sets of configurations.
In *Theoretical Computer Science*.

Ibarra, O. H. (1978).
Reversal-bounded multicounter machines and their decision problems.
J. ACM, 25(1):116–133.

Klaedtke, F. and Rueß, H. (2003).
Monadic second-order logics with cardinalities.
In *Proceedings of the 30th International Colloquium on Automata, Languages, and Programming (ICALP 2003)*, volume 2719 of *Lecture Notes in Computer Science*, pages 681–696. Springer-Verlag.

Bibliography II

Mitrana, V. and Stiebe, R. (2001).
Extended finite automata over groups.
Discrete Appl. Math., 108(3):287–300.

Parikh, R. J. (1966).
On context-free languages.
Journal of the ACM, 13(4):570–581.