Automata with Semilinear Constraints

Ph.D. defense

Michaël Cadilhac

Université de Montréal

March 18th 2013

Theoretical computer science is 80 years old

[Greibach, 1981]: TCS arose from

- Computability

Theoretical computer science is 80 years old

[Greibach, 1981]: TCS arose from

- Computability
- Logic

Theoretical computer science is 80 years old

[Greibach, 1981]: TCS arose from

- Computability
- Logic
- Language theory

Theoretical computer science is 80 years old

[Greibach, 1981]: TCS arose from

$$\left.\begin{array}{l}\text{▸ Computability}\\ \text{▸ Logic}\\ \text{▸ Language theory}\end{array}\right\}\ \text{unifying notion: languages}$$

Theoretical computer science is 80 years old

[Greibach, 1981]: TCS arose from

- Computability
- Logic
- Language theory

unifying notion: languages

Ex: $L = \{a, ba, abb, babb, \cdots\}$

Theoretical computer science is 80 years old

[Greibach, 1981]: TCS arose from

- Computability
- Logic
- Language theory

unifying notion: languages

Theoretical computer science is 80 years old

[Greibach, 1981]: TCS arose from

- Computability
- Logic
- Language theory

$\left.\begin{array}{l} \\ \\ \\ \end{array}\right\}$ unifying notion: languages

Branches strongly interact:

Theoretical computer science is 80 years old

[Greibach, 1981]: TCS arose from

- Computability
- Logic
- Language theory

unifying notion: languages

Branches strongly interact:

- Automata theory

Theoretical computer science is 80 years old

[Greibach, 1981]: TCS arose from

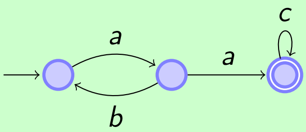
- Computability
- Lo
- La

$\Big\}$ unifying notion: languages

Branch

- Au

Example



- Word: *abaac*

Theoretical computer science is 80 years old

[Greibach, 1981]: TCS arose from

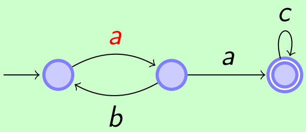
- Computability
- Lo
- La

unifying notion: languages

Branch

- Au

Example



- Word: $abaac$

Theoretical computer science is 80 years old

[Greibach, 1981]: TCS arose from

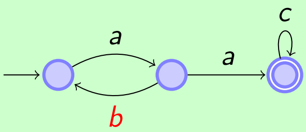
- Computability
- Lo
- La

unifying notion: languages

Branch

- Au

Example



- Word: $a\color{red}{b}\color{black}aac$

Theoretical computer science is 80 years old

[Greibach, 1981]: TCS arose from

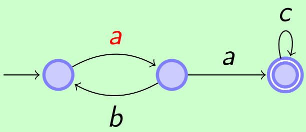
- Computability
- Lo
- La

unifying notion: languages

Branch

- Au

Example



- Word: *abaac*

Theoretical computer science is 80 years old

[Greibach, 1981]: TCS arose from

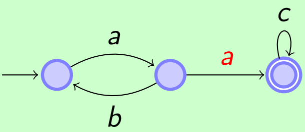
- Computability
- Lo
- La

unifying notion: languages

Branch

- Au

Example



- Word: $abaac$

Theoretical computer science is 80 years old

[Greibach, 1981]: TCS arose from

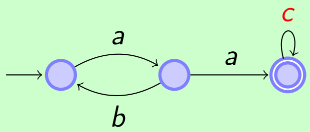
- Computability
- Lo
- La

unifying notion: languages

Branch

- A

Example



- Word: *abaac*

Theoretical computer science is 80 years old

[Greibach, 1981]: TCS arose from

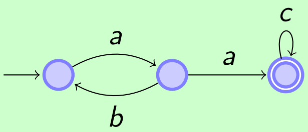
- Computability
- Logic
- La

unifying notion: languages

Branch

- Au

Example



- Word: *abaac* in language $a(ba)^*ac^*$

Theoretical computer science is 80 years old

[Greibach, 1981]: TCS arose from

$$\left.\begin{array}{l}\text{▸ Computability}\\ \text{▸ Logic}\\ \text{▸ Language theory}\end{array}\right\}\ \text{unifying notion: languages}$$

Branches strongly interact:

- Automata theory

Theoretical computer science is 80 years old

[Greibach, 1981]: TCS arose from

$$\left.\begin{array}{l}\text{▸ Computability}\\ \text{▸ Logic}\\ \text{▸ Language theory}\end{array}\right\}\ \text{unifying notion: languages}$$

Branches strongly interact:

- Automata theory: backbone of some complexity classes,

Theoretical computer science is 80 years old

[Greibach, 1981]: TCS arose from

- Computability
- Logic
- Language theory

} unifying notion: languages

Branches strongly interact:

- Automata theory: backbone of some complexity classes, described using logics,

Theoretical computer science is 80 years old

[Greibach, 1981]: TCS arose from

- Computability
- Logic
- Language theory

unifying notion: languages

Branches strongly interact:

- Automata theory: backbone of some complexity classes, described using logics, algebraic structures, ...

Theoretical computer science is 80 years old

[Greibach, 1981]: TCS arose from

- Computability
- Logic
- Language theory

unifying notion: languages

Branches strongly interact:

- Automata theory: backbone of some complexity classes, described using logics, algebraic structures, . . .
  $\rightarrow$ A wealth of variants

Theoretical computer science is 80 years old

[Greibach, 1981]: TCS arose from

- Computability
- Logic
- Language theory

unifying notion: languages

Branches strongly interact:

- Automata theory: backbone of some complexity classes, described using logics, algebraic structures, ...
  → A wealth of variants
- Model-checking: computational task using logics and/or automata

Theoretical computer science is 80 years old

[Greibach, 1981]: TCS arose from

- Computability
- Logic
- Language theory

$\left.\begin{array}{l}\\ \\ \\ \end{array}\right\}$ unifying notion: languages

Branches strongly interact:

- Automata theory: backbone of some complexity classes, described using logics, algebraic structures, ...
  $\rightarrow$ A wealth of variants
- Model-checking: computational task using logics and/or automata

This thesis: add counting mechanisms to automata, continuing a line of work started by [Minsky, 1961]

Outline

Outline

How are context-free and regular similar?

The fundamental result of [Parikh, 1966]

Although $\{a^n b^n \mid n \in \mathbb{N}\} \in \mathrm{CFL} \setminus \mathrm{REG}$:

How are context-free and regular similar?

The fundamental result of [Parikh, 1966]

Although $\{a^n b^n \mid n \in \mathbb{N}\} \in \mathrm{CFL} \setminus \mathrm{REG}$:

Theorem (folklore)

Every unary CFL is regular

How are context-free and regular similar?

The fundamental result of [Parikh, 1966]

Although $\{a^n b^n \mid n \in \mathbb{N}\} \in \mathrm{CFL} \setminus \mathrm{REG}$:

Theorem (folklore)

Every unary CFL is regular

Consequence of a more general result:

Parikh's theorem [1966]

How are context-free and regular similar?

The fundamental result of [Parikh, 1966]

Although $\{a^n b^n \mid n \in \mathbb{N}\} \in \mathrm{CFL} \setminus \mathrm{REG}$:

Theorem (folklore)

Every unary CFL is regular

Consequence of a more general result:

Parikh's theorem [1966]

Disregarding the order of letters, CFL and REG are the same

How are context-free and regular similar?

The fundamental result of [Parikh, 1966]

Although $\{a^n b^n \mid n \in \mathbb{N}\} \in \mathrm{CFL} \setminus \mathrm{REG}$:

Theorem (folklore)

Every unary CFL is regular

Consequence of a more general result:

Parikh's theorem [1966]

Disregarding the order of letters, CFL and REG are the same

More formally: ...

How are context-free and regular similar?

The fundamental result of [Parikh, 1966]

Parikh's theorem [1966]

Disregarding the order of letters, CFL and REG are the same

How are context-free and regular similar?

The fundamental result of [Parikh, 1966]

Parikh's theorem [1966]

Disregarding the order of letters, CFL and REG are the same

How are context-free and regular similar?

The fundamental result of [Parikh, 1966]

Parikh's theorem [1966]

Disregarding the order of letters, CFL and REG are the same

Definition (Parikh image [Parikh, 1966])

With $\Sigma = \{a, b, \ldots\}$, $\text{Pkh}(w) = (|w|_a, |w|_b, \ldots) \in \mathbb{N}^{|\Sigma|}$

How are context-free and regular similar?

The fundamental result of [Parikh, 1966]

Parikh's theorem [1966]

Disregarding the order of letters, CFL and REG are the same

Definition (Parikh image [Parikh, 1966])

With $\Sigma = \{a, b, \ldots\}$, $\mathrm{Pkh}(w) = (|w|_a, |w|_b, \ldots) \in \mathbb{N}^{|\Sigma|}$

Ex: $\mathrm{Pkh}(abaab) = (3, 2)$.

How are context-free and regular similar?

The fundamental result of [Parikh, 1966]

Parikh's theorem [1966]

Disregarding the order of letters, CFL and REG are the same

Definition (Parikh image [Parikh, 1966])

With $\Sigma = \{a, b, \ldots\}$, $\mathrm{Pkh}(w) = (|w|_a, |w|_b, \ldots) \in \mathbb{N}^{|\Sigma|}$

Ex: $\mathrm{Pkh}(abaab) = (3, 2)$.

How are context-free and regular similar?

The fundamental result of [Parikh, 1966]

Parikh's theorem [1966]

Disregarding the order of letters, CFL and REG are the same

Definition (Parikh image [Parikh, 1966])

With $\Sigma = \{a, b, \ldots\}$, $\mathrm{Pkh}(w) = (|w|_a, |w|_b, \ldots) \in \mathbb{N}^{|\Sigma|}$

How are context-free and regular similar?

The fundamental result of [Parikh, 1966]

Parikh's theorem [1966]

Disregarding the order of letters, CFL and REG are the same

Definition (Parikh image [Parikh, 1966])

With $\Sigma = \{a, b, \ldots\}$, $\mathrm{Pkh}(w) = (|w|_a, |w|_b, \ldots) \in \mathbb{N}^{|\Sigma|}$

Definitions (Semilinear set [Parikh, 1966])

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

How are context-free and regular similar?

The fundamental result of [Parikh, 1966]

Parikh's theorem [1966]

Disregarding the order of letters, CFL and REG are the same

Definition (Parikh image [Parikh, 1966])

With $\Sigma = \{a, b, \ldots\}$, $\mathrm{Pkh}(w) = (|w|_a, |w|_b, \ldots) \in \mathbb{N}^{|\Sigma|}$

Definitions (Semilinear set [Parikh, 1966])

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

How are context-free and regular similar?

The fundamental result of [Parikh, 1966]

Parikh's theorem [1966]

Disregarding the order of letters, CFL and REG are the same

Definition (Parikh image [Parikh, 1966])

With $\Sigma = \{a, b, \ldots\}$, $\mathrm{Pkh}(w) = (|w|_a, |w|_b, \ldots) \in \mathbb{N}^{|\Sigma|}$

Definitions (Semilinear set [Parikh, 1966])

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

Ex : $\left\{ \binom{n}{n/2} \mid n \in 2\mathbb{N} \right\}$ is semilinear

$\left\{ (2^n) \mid n \in \mathbb{N} \right\}$ is not

How are context-free and regular similar?

The fundamental result of [Parikh, 1966]

Parikh's theorem [1966]

Disregarding the order of letters, CFL and REG are the same

Definition (Parikh image [Parikh, 1966])

With $\Sigma = \{a, b, \ldots\}$, $\mathrm{Pkh}(w) = (|w|_a, |w|_b, \ldots) \in \mathbb{N}^{|\Sigma|}$

Definitions (Semilinear set [Parikh, 1966])

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

Ex : $\left\{ \binom{n}{n/2} \mid n \in 2\mathbb{N} \right\}$ is semilinear

$\left\{ (2^n) \mid n \in \mathbb{N} \right\}$ is not

How are context-free and regular similar?

The fundamental result of [Parikh, 1966]

Parikh's theorem [1966]

Disregarding the order of letters, CFL and REG are the same

Definition (Parikh image [Parikh, 1966])

With $\Sigma = \{a, b, \ldots\}$, $\mathrm{Pkh}(w) = (|w|_a, |w|_b, \ldots) \in \mathbb{N}^{|\Sigma|}$

Definitions (Semilinear set [Parikh, 1966])

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

How are context-free and regular similar?

The fundamental result of [Parikh, 1966]

Parikh's theorem [1966]

Disregarding the order of letters, CFL and REG are the same

Definition (Parikh image [Parikh, 1966])

With $\Sigma = \{a, b, \ldots\}$, $\mathrm{Pkh}(w) = (|w|_a, |w|_b, \ldots) \in \mathbb{N}^{|\Sigma|}$

Definitions (Semilinear set [Parikh, 1966])

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

Parikh's theorem [1966]

L CFL or REG $\Rightarrow \mathrm{Pkh}(L)$ semilinear
C semilinear $\Rightarrow \exists L \in \mathrm{REG}\ \mathrm{Pkh}(L) = C$

Semilinear sets everywhere!

Pkh(CFL) = Pkh(REG) = semilin., but also. . .

Semilinear sets occur:

Semilinear sets everywhere!

Pkh(CFL) = Pkh(REG) = semilin., but also...

Semilinear sets occur:

- Additive number theory: "generalized arithmetic progressions"

Semilinear sets everywhere!

Pkh(CFL) = Pkh(REG) = semilin., but also...

Semilinear sets occur:

- Additive number theory: "generalized arithmetic progressions"
- Logic: sets expressible in first-order with + [Ginsburg and Spanier, 1966]

Semilinear sets everywhere!

Pkh(CFL) = Pkh(REG) = semilin., but also...

Semilinear sets occur:

- Additive number theory: "generalized arithmetic progressions"
- Logic: sets expressible in first-order with + [Ginsburg and Spanier, 1966]
- Computability: the classical example of a decidable fragment of arithmetic [Presburger, 1927]

Semilinear sets everywhere!

Pkh(CFL) = Pkh(REG) = semilin., but also. . .

Semilinear sets occur:

- Additive number theory: “generalized arithmetic progressions”
- Logic: sets expressible in first-order with + [Ginsburg and Spanier, 1966]
- Computability: the classical example of a decidable fragment of arithmetic [Presburger, 1927]
- Petri nets: decidability of accessibility [Leroux, 2009]

Semilinear sets everywhere!

$\mathrm{Pkh}(\mathrm{CFL}) = \mathrm{Pkh}(\mathrm{REG}) =$ semilin., but also...

Semilinear sets occur:

- Additive number theory: "generalized arithmetic progressions"
- Logic: sets expressible in first-order with + [Ginsburg and Spanier, 1966]
- Computability: the classical example of a decidable fragment of arithmetic [Presburger, 1927]
- Petri nets: decidability of accessibility [Leroux, 2009]
- Generalized: over commutative rings [Kudlek, 2007]; as *semipolynomial* sets [Karianto et al., 2006], ...

Automata with Semilinear Constraints

Our thesis: a systematic study of *semilinear constriction* in variants of automata

Automata with Semilinear Constraints

Our thesis: a systematic study of *semilinear constriction* in variants of automata

- We further the works of [Ginsburg and Spanier, 1964], [Ibarra, 1978], [Bouajjani and Habermehl, 1999], [Klaedtke and Rueß, 2003]

Automata with Semilinear Constraints

Our thesis: a systematic study of *semilinear constriction* in variants of automata

- We further the works of [Ginsburg and Spanier, 1964], [Ibarra, 1978], [Bouajjani and Habermehl, 1999], [Klaedtke and Rueß, 2003]
- We propose new variants

Automata with Semilinear Constraints

Our thesis: a systematic study of *semilinear constriction* in variants of automata

- We further the works of [Ginsburg and Spanier, 1964], [Ibarra, 1978], [Bouajjani and Habermehl, 1999], [Klaedtke and Rueß, 2003]
- We propose new variants
- We locate them within other frameworks

Outline

Regular Constrained Languages

Constraining the Parikh image of words

Definition (Constrained Language)

$L \subseteq \Sigma^*$, C set of vectors:

Regular Constrained Languages

Constraining the Parikh image of words

Definition (Constrained Language)

$L \subseteq \Sigma^*$, C set of vectors: $L{\restriction}_C = \{w \in L \mid \mathrm{Pkh}(w) \in C\}$

Regular Constrained Languages

Constraining the Parikh image of words

Definition (Constrained Language)

$L \subseteq \Sigma^*$, C set of vectors: $L{\upharpoonright}_C = \{w \in L \mid \mathrm{Pkh}(w) \in C\}$

Regular Constrained Languages

Constraining the Parikh image of words

Definition (Constrained Language)

$L \subseteq \Sigma^*$, C set of vectors: $L{\restriction_C} = \{w \in L \mid \mathrm{Pkh}(w) \in C\}$

$\mathrm{REG}{\restriction_{\mathrm{Semilinear}}}$ written as RCL bellow

Regular Constrained Languages

Constraining the Parikh image of words

Definition (Constrained Language)

$L \subseteq \Sigma^*$, C set of vectors: $L{\restriction}_C = \{w \in L \mid \mathrm{Pkh}(w) \in C\}$

$\mathrm{REG}{\restriction}_{\mathrm{Semilinear}}$ written as RCL bellow

Regular Constrained Languages

Constraining the Parikh image of words

Definition (Constrained Language)

$L \subseteq \Sigma^*$, C set of vectors: $L\restriction_C = \{w \in L \mid \mathrm{Pkh}(w) \in C\}$

$\mathrm{REG}\restriction_{\mathrm{Semilinear}}$ written as RCL bellow

Examples:

- In RCL: $\{a^n b^n c^n \mid n \in \mathbb{N}\} = a^* b^* c^* \restriction_{\{(n,n,n)\} \mid n \in \mathbb{N}\}}$

Regular Constrained Languages

Constraining the Parikh image of words

Definition (Constrained Language)

$L \subseteq \Sigma^*$, C set of vectors: $L{\restriction_C} = \{w \in L \mid \mathrm{Pkh}(w) \in C\}$

$\mathrm{REG}{\restriction_{\mathrm{Semilinear}}}$ written as RCL bellow

Examples:

- In RCL: $\{a^n b^n c^n \mid n \in \mathbb{N}\} = a^* b^* c^* {\restriction}_{\underbrace{\{(n,n,n)\} \mid n \in \mathbb{N}\}}_{= \mathrm{Pkh}((abc)^*)}}$

Regular Constrained Languages

Constraining the Parikh image of words

Definition (Constrained Language)

$L \subseteq \Sigma^*$, C set of vectors: $L\restriction_C = \{w \in L \mid \mathrm{Pkh}(w) \in C\}$

$\mathrm{REG}\restriction_{\mathrm{Semilinear}}$ written as RCL bellow

Examples:

- In RCL: $\{a^n b^n c^n \mid n \in \mathbb{N}\} = a^* b^* c^* \restriction_{\{(n,n,n)\} \mid n \in \mathbb{N}\}}$

Regular Constrained Languages

Constraining the Parikh image of words

Definition (Constrained Language)

$L \subseteq \Sigma^*$, C set of vectors: $L{\restriction_C} = \{w \in L \mid \mathrm{Pkh}(w) \in C\}$

$\mathrm{REG}{\restriction_{\mathrm{Semilinear}}}$ written as RCL bellow

Examples:

- In RCL: $\{a^n b^n c^n \mid n \in \mathbb{N}\} = a^* b^* c^* {\restriction_{\{(n,n,n)\} \mid n \in \mathbb{N}\}}}$
- Out of RCL: $\{a^n b^n\} \cup b^* a^*$

Regular Constrained Languages

Constraining the Parikh image of words

Definition (Constrained Language)

$L \subseteq \Sigma^*$, C set of vectors: $L{\restriction}_C = \{w \in L \mid \mathrm{Pkh}(w) \in C\}$

$\mathrm{REG}{\restriction}_{\mathrm{Semilinear}}$ written as RCL bellow

Examples:

- In RCL: $\{a^n b^n c^n \mid n \in \mathbb{N}\} = a^* b^* c^* {\restriction}_{\{(n,n,n)\} \mid n \in \mathbb{N}\}}$
- Out of RCL: $\{a^n b^n\} \cup b^* a^*$
  Indeed $\{a^n b^n\} \cup b^* a^* = L{\restriction}_C$ for regular L implies $a^m b^n \in L$ for some $m \neq n$

Regular Constrained Languages

Constraining the Parikh image of words

Definition (Constrained Language)

$L \subseteq \Sigma^*$, C set of vectors: $L\restriction_C = \{w \in L \mid \mathrm{Pkh}(w) \in C\}$

$\mathrm{REG}\restriction_{\mathrm{Semilinear}}$ written as RCL bellow

Examples:

- In RCL: $\{a^n b^n c^n \mid n \in \mathbb{N}\} = a^* b^* c^* \restriction_{\{(n,n,n)\} \mid n \in \mathbb{N}\}}$
- Out of RCL: $\{a^n b^n\} \cup b^* a^*$
  Indeed $\{a^n b^n\} \cup b^* a^* = L\restriction_C$ for regular L implies $a^m b^n \in L$ for some $m \neq n$

(Otherwise: $L \cap a^* b^* = \{a^n b^n\}$)

Regular Constrained Languages

Constraining the Parikh image of words

Definition (Constrained Language)

$L \subseteq \Sigma^*$, C set of vectors: $L\restriction_C = \{w \in L \mid \mathrm{Pkh}(w) \in C\}$

$\mathrm{REG}\restriction_{\mathrm{Semilinear}}$ written as RCL bellow

Examples:

- In RCL: $\{a^n b^n c^n \mid n \in \mathbb{N}\} = a^* b^* c^* \restriction_{\{(n,n,n)\} \mid n \in \mathbb{N}\}}$
- Out of RCL: $\{a^n b^n\} \cup b^* a^*$
  Indeed $\{a^n b^n\} \cup b^* a^* = L\restriction_C$ for regular L implies $a^m b^n \in L$ for some $m \neq n$

Regular Constrained Languages

Constraining the Parikh image of words

Definition (Constrained Language)

$L \subseteq \Sigma^*$, C set of vectors: $L{\restriction}_C = \{w \in L \mid \mathrm{Pkh}(w) \in C\}$

$\mathrm{REG}{\restriction}_{\mathrm{Semilinear}}$ written as RCL bellow

Examples:

- In RCL: $\{a^n b^n c^n \mid n \in \mathbb{N}\} = a^* b^* c^* {\restriction}_{\{(n,n,n)\} \mid n \in \mathbb{N}\}}$
- Out of RCL: $\{a^n b^n\} \cup b^* a^*$
  Indeed $\{a^n b^n\} \cup b^* a^* = L{\restriction}_C$ for regular L implies $a^m b^n \in L$ for some $m \neq n$ thus $(m, n) \notin C$,

Regular Constrained Languages

Constraining the Parikh image of words

Definition (Constrained Language)

$L \subseteq \Sigma^*$, C set of vectors: $L{\upharpoonright}_C = \{w \in L \mid \mathrm{Pkh}(w) \in C\}$

$\mathrm{REG}{\upharpoonright}_{\mathrm{Semilinear}}$ written as RCL bellow

Examples:

- In RCL: $\{a^n b^n c^n \mid n \in \mathbb{N}\} = a^* b^* c^* {\upharpoonright}_{\{(n,n,n)\} \mid n \in \mathbb{N}\}}$
- Out of RCL: $\{a^n b^n\} \cup b^* a^*$
  Indeed $\{a^n b^n\} \cup b^* a^* = L{\upharpoonright}_C$ for regular L implies $a^m b^n \in L$ for some $m \neq n$ thus $(m, n) \notin C$, hence $b^n a^m$ not in $L{\upharpoonright}_C$

Regular Constrained Languages

RCL is not a robust class

- RCL is not closed under union

Regular Constrained Languages

RCL is not a robust class

- RCL is not closed under union

Theorem

	$\cup$	$\cap$	$-$	$\cdot$	*rev*
RCL	*N*	*Y*	*N*	*N*	*Y*

Regular Constrained Languages

RCL is not a robust class

- RCL is not closed under union

Theorem

	$\cup$	$\cap$	$-$	$\cdot$	*rev*
RCL	*N*	*Y*	*N*	*N*	*Y*

- In hindsight: the *automaton* should take better advantage of counting

Outline

Semilinear sets: our counters

Regular Constrained Languages

Constrained Automata

Affine Constrained Automata

Conclusion

The language of accepting runs

Automaton $A = (Q, \Sigma, \delta, q_0, F)$,

The language of accepting runs

Automaton $A = (Q, \Sigma, \delta, q_0, F)$, $\delta \subseteq Q \times \Sigma \times Q$

The language of accepting runs

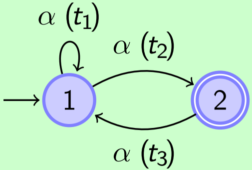
Automaton $A = (Q, \Sigma, \delta, q_0, F)$, $\delta \subseteq Q \times \Sigma \times Q$
$\rightarrow \delta$ is an alphabet, $\mathrm{AccRuns}(A)$ is a language

The language of accepting runs

Automaton $A = (Q, \Sigma, \delta, q_0, F)$, $\delta \subseteq Q \times \Sigma \times Q$

$\rightarrow$ δ is an alphabet, $\mathrm{AccRuns}(A)$ is a language

Example



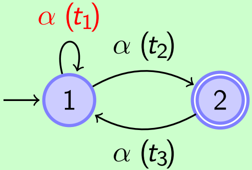
$$\begin{cases} \delta = \{t_1, t_2, t_3\} \\ t_1 = (1, \alpha, 1) \\ t_2 = (1, \alpha, 2) \\ t_3 = (2, \alpha, 1) \end{cases}$$

▸▸ $\mathrm{AccRuns}(A) =$

The language of accepting runs

Automaton $A = (Q, \Sigma, \delta, q_0, F)$, $\delta \subseteq Q \times \Sigma \times Q$
$\rightarrow$ δ is an alphabet, $\mathrm{AccRuns}(A)$ is a language

Example



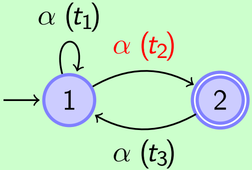
$$\begin{cases} \delta = \{t_1, t_2, t_3\} \\ t_1 = (1, \alpha, 1) \\ t_2 = (1, \alpha, 2) \\ t_3 = (2, \alpha, 1) \end{cases}$$

▸▸ $\mathrm{AccRuns}(A) = t_1^*$

The language of accepting runs

Automaton $A = (Q, \Sigma, \delta, q_0, F)$, $\delta \subseteq Q \times \Sigma \times Q$
$\rightarrow$ δ is an alphabet, AccRuns(A) is a language

Example



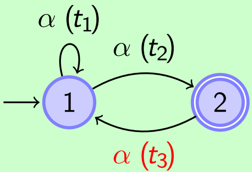
$$\begin{cases} \delta = \{t_1, t_2, t_3\} \\ t_1 = (1, \alpha, 1) \\ t_2 = (1, \alpha, 2) \\ t_3 = (2, \alpha, 1) \end{cases}$$

- AccRuns(A) = $t_1^* t_2$

The language of accepting runs

Automaton $A = (Q, \Sigma, \delta, q_0, F)$, $\delta \subseteq Q \times \Sigma \times Q$
$\rightarrow$ δ is an alphabet, $\mathrm{AccRuns}(A)$ is a language

Example



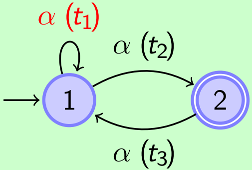
$$\begin{cases} \delta = \{t_1, t_2, t_3\} \\ t_1 = (1, \alpha, 1) \\ t_2 = (1, \alpha, 2) \\ t_3 = (2, \alpha, 1) \end{cases}$$

- $\mathrm{AccRuns}(A) = t_1^* t_2 (t_3$

The language of accepting runs

Automaton $A = (Q, \Sigma, \delta, q_0, F)$, $\delta \subseteq Q \times \Sigma \times Q$
$\to$ δ is an alphabet, $\mathrm{AccRuns}(A)$ is a language

Example



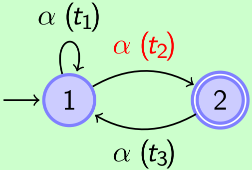
$$\begin{cases} \delta = \{t_1, t_2, t_3\} \\ t_1 = (1, \alpha, 1) \\ t_2 = (1, \alpha, 2) \\ t_3 = (2, \alpha, 1) \end{cases}$$

- $\mathrm{AccRuns}(A) = t_1^* t_2 (t_3 t_1^*$

The language of accepting runs

Automaton $A = (Q, \Sigma, \delta, q_0, F)$, $\delta \subseteq Q \times \Sigma \times Q$
$\rightarrow$ δ is an alphabet, $\mathrm{AccRuns}(A)$ is a language

Example



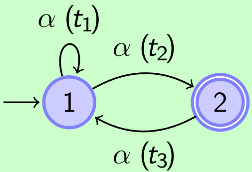
$$\begin{cases} \delta = \{t_1, t_2, t_3\} \\ t_1 = (1, \alpha, 1) \\ t_2 = (1, \alpha, 2) \\ t_3 = (2, \alpha, 1) \end{cases}$$

- $\mathrm{AccRuns}(A) = t_1^* t_2 (t_3 t_1^* t_2)$

The language of accepting runs

Automaton $A = (Q, \Sigma, \delta, q_0, F)$, $\delta \subseteq Q \times \Sigma \times Q$
$\rightarrow$ δ is an alphabet, $\mathrm{AccRuns}(A)$ is a language

Example



$$\begin{cases} \delta = \{t_1, t_2, t_3\} \\ t_1 = (1, \alpha, 1) \\ t_2 = (1, \alpha, 2) \\ t_3 = (2, \alpha, 1) \end{cases}$$

- $\mathrm{AccRuns}(A) = t_1^* t_2 (t_3 t_1^* t_2)^*$

Constrained automata

- Semilinear constraint on Pkh(a run)

Constrained automata

- Semilinear constraint on Pkh(a run)

Example

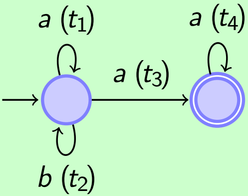
$$L = \{w \cdot a^{|w|+1} \mid w \in \{a, b\}^*\}$$

Constrained automata

- Semilinear constraint on Pkh(a run)

Example

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

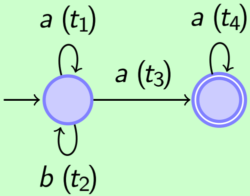


Constrained automata

- Semilinear constraint on Pkh(a run)

Example

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



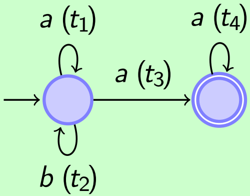
$$\delta = \{t_1, t_2, t_3, t_4\}$$

Constrained automata

- Semilinear constraint on Pkh(a run)

Example

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



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

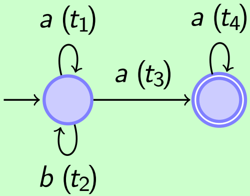
$C = \Bigg\{$

Constrained automata

- Semilinear constraint on Pkh(a run)

Example

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



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

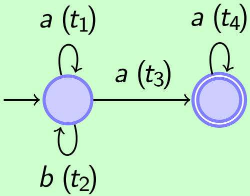
$$C = \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \cdot k_1 + \right.$$

Constrained automata

- Semilinear constraint on Pkh(a run)

Example

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



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

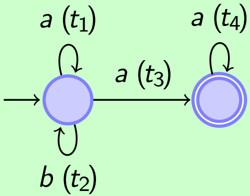
$$C = \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \cdot k_1 + \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \cdot k_2 + \right.$$

Constrained automata

- Semilinear constraint on Pkh(a run)

Example

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



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

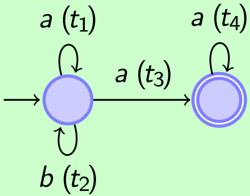
$$C = \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \cdot k_1 + \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \cdot k_2 + \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \right\}$$

Constrained automata

- Semilinear constraint on Pkh(a run)

Example

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



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

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

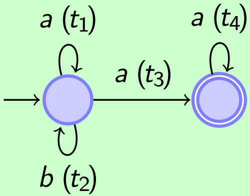
Word: *aabaaaa*

Constrained automata

- Semilinear constraint on Pkh(a run)

Example

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



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

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

Word: *aabaaaa*

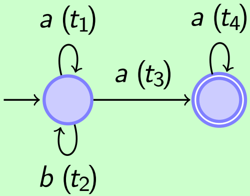
Pkh(traced run):
$\pi =$

Constrained automata

- Semilinear constraint on Pkh(a run)

Example

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



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

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

Word: *aabaaaa*

Pkh(traced run): $(0, 0, 0, 0)$

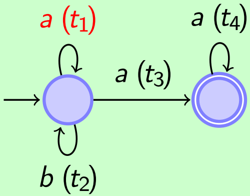
$\pi =$

Constrained automata

- Semilinear constraint on Pkh(a run)

Example

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



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

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

Word: *aabaaaa*

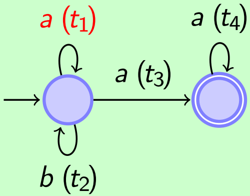
Pkh(traced run): (1, 0, 0, 0)
$\pi = t_1$

Constrained automata

- Semilinear constraint on Pkh(a run)

Example

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



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

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

Word: *aabaaaa*

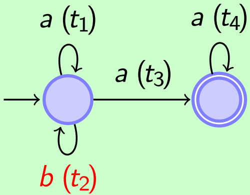
Pkh(traced run): (2, 0, 0, 0)
$\pi = t_1 t_1$

Constrained automata

- Semilinear constraint on Pkh(a run)

Example

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



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

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

Word: *aabaaaa*

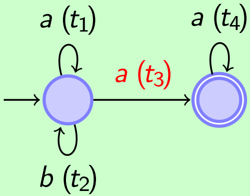
Pkh(traced run): (2, 1, 0, 0)
$\pi = t_1 t_1 t_2$

Constrained automata

- Semilinear constraint on Pkh(a run)

Example

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



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

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

Word: *aabaaaa*

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

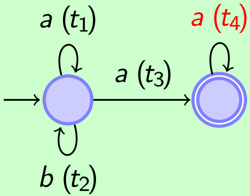
$\pi = t_1 t_1 t_2 t_3$

Constrained automata

- Semilinear constraint on Pkh(a run)

Example

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



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

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

Word: *aabaaaa*

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

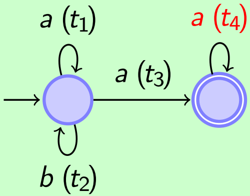
$\pi = t_1 t_1 t_2 t_3 t_4$

Constrained automata

- Semilinear constraint on Pkh(a run)

Example

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



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

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

Word: *aabaaaa*

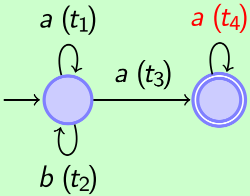
Pkh(traced run): (2, 1, 1, 2)
$\pi = t_1 t_1 t_2 t_3 t_4 t_4$

Constrained automata

- Semilinear constraint on Pkh(a run)

Example

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



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

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

Word: *aabaaaa*

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

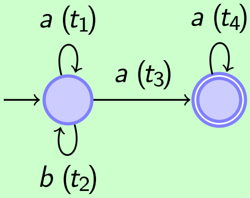
$\pi = t_1 t_1 t_2 t_3 t_4 t_4 t_4$

Constrained automata

- Semilinear constraint on Pkh(a run)

Example

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



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

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

Word: *aabaaaa*

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

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

Constrained automata formally

Definition (Constr. Automata [Klaedtke and Rueß, 2003])

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

Constrained automata formally

Definition (Constr. Automata [Klaedtke and Rueß, 2003])

- *Constrained automaton* (CA): a pair (A, C) with:
  - A a finite automaton of transition set δ
  - $C \subseteq \mathbb{N}^{|\delta|}$ semilinear
- $L(A, C) = \mathrm{Label}(\mathrm{AccRuns}(A)\restriction_C)$

Constrained automata formally

Definition (Constr. Automata [Klaedtke and Rueß, 2003])

- *Constrained automaton* (CA): a pair (A, C) with:
  - A a finite automaton of transition set δ
  - $C \subseteq \mathbb{N}^{|\delta|}$ semilinear
- $L(A, C) = \mathrm{Label}(\mathrm{AccRuns}(A){\restriction}_C)$
- *Deterministic, unambiguous* if A is

Constrained automata formally

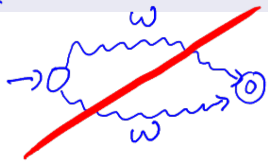
Definition (Constr. Automata [Klaedtke and Rueß, 2003])

- *Constrained automaton* (CA): a pair (A, C) with:
  - A a finite automaton of transition set δ
  - $C \subseteq \mathbb{N}^{|\delta|}$ semilinear
- $L(A, C) = \mathrm{Label}(\mathrm{AccRuns}(A)\restriction_C)$
- *Deterministic, unambiguous* if A is

Constrained automata formally

Definition (Constr. Automata [Klaedtke and Rueß, 2003])

- *Constrained automaton* (CA): a pair (A, C) with:
  - A a finite automaton of transition set δ
  - $C \subseteq \mathbb{N}^{|\delta|}$ semilinear
- $L(A, C) = \mathrm{Label}(\mathrm{AccRuns}(A){\upharpoonright}_C)$
- *Deterministic, unambiguous* if A is



Constrained automata formally

Definition (Constr. Automata [Klaedtke and Rueß, 2003])

- *Constrained automaton* (CA): a pair (A, C) with:
  - A a finite automaton of transition set δ
  - $C \subseteq \mathbb{N}^{|\delta|}$ semilinear
- $L(A, C) = \text{Label}(\text{AccRuns}(A)\restriction_C)$
- *Deterministic, unambiguous* if A is

Constrained automata formally

Definition (Constr. Automata [Klaedtke and Rueß, 2003])

- *Constrained automaton* (CA): a pair (A, C) with:
  - A a finite automaton of transition set δ
  - $C \subseteq \mathbb{N}^{|\delta|}$ semilinear
- $L(A, C) = \mathrm{Label}(\mathrm{AccRuns}(A){\restriction_C})$
- *Deterministic, unambiguous* if A is

Constrained automata formally

Definition (Constr. Automata [Klaedtke and Rueß, 2003])

- *Constrained automaton* (CA): a pair (A, C) with:
  - A a finite automaton of transition set δ
  - $C \subseteq \mathbb{N}^{|\delta|}$ semilinear
- $L(A, C) = \mathrm{Label}(\mathrm{AccRuns}(A){\restriction_C})$
- *Deterministic, unambiguous* if A is

Examples:

- $L = \{w \cdot a^{|w|+1}\} \in \mathrm{CA}$ but $\notin$ DetCA

Constrained automata formally

Definition (Constr. Automata [Klaedtke and Rueß, 2003])

- *Constrained automaton* (CA): a pair (A, C) with:
  - A a finite automaton of transition set δ
  - $C \subseteq \mathbb{N}^{|\delta|}$ semilinear
- $L(A, C) = \mathrm{Label}(\mathrm{AccRuns}(A){\upharpoonright_C})$
- *Deterministic, unambiguous* if A is

Examples:

- $L = \{w \cdot a^{|w|+1}\} \in \mathrm{CA}$ but $\notin$ DetCA, same for $\overline{\mathrm{COPY}}$

Constrained automata formally

Definition (Constr. Automata [Klaedtke and Rueß, 2003])

- *Constrained automaton* (CA): a pair (A, C) with:
  - A a finite automaton of transition set δ
  - $C \subseteq \mathbb{N}^{|\delta|}$ semilinear
- $L(A, C) = \mathrm{Label}(\mathrm{AccRuns}(A)\restriction_C)$
- *Deterministic, unambiguous* if A is

Examples:

- $L = \{w \cdot a^{|w|+1}\} \in \mathrm{CA}$ but $\notin$ DetCA, same for $\overline{\mathrm{COPY}}$
- COPY $\notin$ CA

Constrained automata formally

Definition (Constr. Automata [Klaedtke and Rueß, 2003])

- *Constrained automaton* (CA): a pair (A, C) with:
  - A a finite automaton of transition set δ
  - $C \subseteq \mathbb{N}^{|\delta|}$ semilinear
- $L(A, C) = \mathrm{Label}(\mathrm{AccRuns}(A)\restriction_C)$
- *Deterministic, unambiguous* if A is

Examples:

- $L = \{w \cdot a^{|w|+1}\} \in \mathrm{CA}$ but $\notin$ DetCA, same for $\overline{\mathrm{COPY}}$
- COPY $\notin$ CA

$\to \forall$ positions $u_i = v_i$

$\exists$ position $u_i \neq v_i$

Constrained automata historically

CA were introduced by [Klaedtke and Rueß, 2003], generalizing [Bouajjani and Habermehl, 1999]:

Constrained automata historically

CA were introduced by [Klaedtke and Rueß, 2003], generalizing [Bouajjani and Habermehl, 1999]:

- Linked to reversal-bounded counter machines of [Ibarra, 1978]

Constrained automata historically

CA were introduced by [Klaedtke and Rueß, 2003], generalizing [Bouajjani and Habermehl, 1999]:

- Linked to reversal-bounded counter machines of [Ibarra, 1978]
- Gave closure properties, decidability results

Constrained automata historically

CA were introduced by [Klaedtke and Rueß, 2003], generalizing [Bouajjani and Habermehl, 1999]:

- Linked to reversal-bounded counter machines of [Ibarra, 1978]
- Gave closure properties, decidability results
- Connected with variants of the canonical logic for REG

Constrained automata historically

CA were introduced by [Klaedtke and Rueß, 2003], generalizing [Bouajjani and Habermehl, 1999]:

- Linked to reversal-bounded counter machines of [Ibarra, 1978]
- Gave closure properties, decidability results
- Connected with variants of the canonical logic for REG
- Used in model-checking

Closure properties

We continue the study of closure:

Theorem (with [Klaedtke and Rueß, 2003])

	$\cup$	$\cap$	$-$	$\cdot$	*rev*
DetCA	Y	Y	Y	N	N
UnCA	Y	Y	Y	N	Y
CA	Y	Y	N	Y	Y

Closure properties

We continue the study of closure:

Theorem (with [Klaedtke and Rueß, 2003])

	$\cup$	$\cap$	$-$	$\cdot$	*rev*
DetCA	*Y*	*Y*	*Y*	*N*	*N*
UnCA	*Y*	*Y*	*Y*	*N*	*Y*
CA	*Y*	*Y*	*N*	*Y*	*Y*

- Use of expressiveness lemmata reminiscent of pumping

Decidability properties

We also contribute (un)decidability properties:

Theorem (with [Klaedtke and Rueß, 2003])

	$\emptyset$	Σ^*	*fin.*	$\subseteq$	*reg.*
DetCA	*D*	*D*	*D*	*D*	*D*
UnCA	*D*	*D*	*D*	*D*	*D*
CA	*D*	*U*	*D*	*U*	*U*

Decidability properties

We also contribute (un)decidability properties:

Theorem (with [Klaedtke and Rueß, 2003])

	$\emptyset$	Σ^*	*fin.*	$\subseteq$	*reg.*
DetCA	*D*	*D*	*D*	*D*	*D*
UnCA	*D*	*D*	*D*	*D*	*D*
CA	*D*	*U*	*D*	*U*	*U*

Decidability properties

The decidability of regularity

- In fact, we show "$\mathrm{AccRuns}(A)\restriction_C \in \mathrm{REG}$?" decidable

Decidability properties

The decidability of regularity

- In fact, we show "$\mathrm{AccRuns}(A){\upharpoonright}_C \in \mathrm{REG}$?" decidable

Lemma

For A deterministic or unambiguous:

$$L(A, C) \in \mathrm{REG} \textit{ iff } \mathrm{AccRuns}(A){\upharpoonright}_C \in \mathrm{REG}$$

Decidability properties

The decidability of regularity

- In fact, we show "$\mathrm{AccRuns}(A)\restriction_C \in \mathrm{REG}$?" decidable

Lemma

For A deterministic or unambiguous:

$$L(A, C) \in \mathrm{REG} \textit{ iff } \mathrm{AccRuns}(A)\restriction_C \in \mathrm{REG}$$

- $\leftarrow$: clear

Decidability properties

The decidability of regularity

- In fact, we show "$\mathrm{AccRuns}(A)\restriction_C \in \mathrm{REG}$?" decidable

Lemma

For A deterministic or unambiguous:

$$L(A, C) \in \mathrm{REG} \textit{ iff } \mathrm{AccRuns}(A)\restriction_C \in \mathrm{REG}$$

- $\leftarrow$: clear
- $\rightarrow$:
  $\mathrm{AccRuns}(A)\restriction_C = \{\pi \in \mathrm{AccRuns}(A) \mid \mathrm{Label}(\pi) \in L(A, C)\}$

Decidability properties

The decidability of regularity

- In fact, we show "$\mathrm{AccRuns}(A){\restriction_C} \in \mathrm{REG}$?" decidable

Lemma

For A deterministic or unambiguous:
$$L(A, C) \in \mathrm{REG} \textit{ iff } \mathrm{AccRuns}(A){\restriction_C} \in \mathrm{REG}$$

- $\leftarrow$: clear
- $\rightarrow$:

$$\begin{aligned} \mathrm{AccRuns}(A){\restriction_C} &= \{\pi \in \mathrm{AccRuns}(A) \mid \mathrm{Label}(\pi) \in L(A, C)\} \\ &= \mathrm{AccRuns}(A) \cap \mathrm{Label}^{-1}(L(A, C)) \end{aligned}$$

CA over Bounded Languages

Definition (Bounded language [Ginsburg and Spanier, 1964])

L bounded iff $L \subseteq w_1^* w_2^* \cdots w_n^*$

CA over Bounded Languages

Definition (Bounded language [Ginsburg and Spanier, 1964])

L bounded iff $L \subseteq w_1^* w_2^* \cdots w_n^*$

- Exhibit properties "we wistfully wish CFL would have" [Ginsburg, 1966]

CA over Bounded Languages

Definition (Bounded language [Ginsburg and Spanier, 1964])

L bounded iff $L \subseteq w_1^* w_2^* \cdots w_n^*$

- Exhibit properties "we wistfully wish CFL would have" [Ginsburg, 1966]
- Model-checking of sequential machines [Esparza et al., 2012]

CA over Bounded Languages

Definition (Bounded language [Ginsburg and Spanier, 1964])

L bounded iff $L \subseteq w_1^* w_2^* \cdots w_n^*$

- Exhibit properties "we wistfully wish CFL would have" [Ginsburg, 1966]
- Model-checking of sequential machines [Esparza et al., 2012]

Theorem

DetCA, UnCA and CA express the same bounded languages

CA over Bounded Languages

Definition (Bounded language [Ginsburg and Spanier, 1964])

L bounded iff $L \subseteq w_1^* w_2^* \cdots w_n^*$

- Exhibit properties "we wistfully wish CFL would have" [Ginsburg, 1966]
- Model-checking of sequential machines [Esparza et al., 2012]

Theorem

DetCA, UnCA and CA express the same bounded languages

- Similar to [Ibarra and Seki, 2011]

An algebraic theory for CA

Using *finitely typed monoids* [Krebs et al., 2007]

Theorem (Algebraic characterizations of DetCA and UnCA)

An algebraic theory for CA

Using *finitely typed monoids* [Krebs et al., 2007]

Theorem (Algebraic characterizations of DetCA and UnCA)

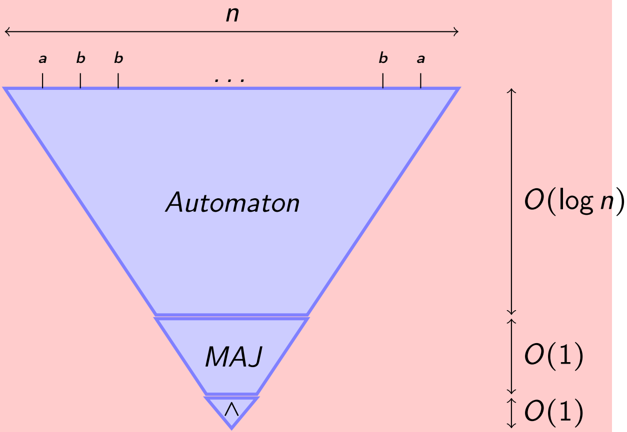
- *Consequence for UnCA, expressible as:*

An algebraic theory for CA

Using *finitely typed monoids* [Krebs et al., 2007]

Theorem (Algebraic characterizations of DetCA and UnCA)

- *Consequence for UnCA, expressible as:*



An algebraic theory for CA

Using *finitely typed monoids* [Krebs et al., 2007]

Theorem (Algebraic characterizations of DetCA and UnCA)

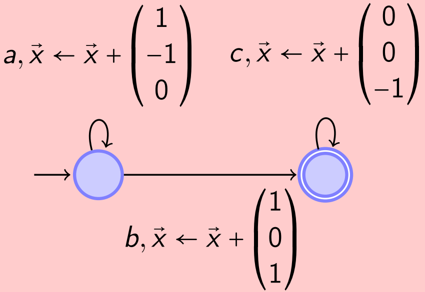
- *Consequence for DetCA, expressible as:*

An algebraic theory for CA

Using *finitely typed monoids* [Krebs et al., 2007]

Theorem (Algebraic characterizations of DetCA and UnCA)

- *Consequence for DetCA, expressible as:*



Constraint: Bool($x_i > 0$)

Outline

Affine constrained automata

Adding linear transformations to CA transitions

Definition (Affine Constrained Automata (ACA))

- *Affine constrained automaton* given by:
  - A finite automaton
  - A map from the transitions to affine functions
  - A semilinear set

Affine constrained automata

Adding linear transformations to CA transitions

Definition (Affine Constrained Automata (ACA))

- *Affine constrained automaton* given by:
  - A finite automaton
  - A map from the transitions to affine functions
  - A semilinear set

$$\bigcirc \xrightarrow{a,\ \vec{x} \leftarrow M.\vec{x} + \vec{v}} \bigcirc$$

Affine constrained automata

Adding linear transformations to CA transitions

Definition (Affine Constrained Automata (ACA))

- *Affine constrained automaton* given by:
  - A finite automaton
  - A map from the transitions to affine functions
  - A semilinear set

$$\bigcirc \xrightarrow{a,\ \vec{x} \leftarrow M.\vec{x} + \vec{v}} \bigcirc$$

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

Affine constrained automata

Adding linear transformations to CA transitions

Definition (Affine Constrained Automata (ACA))

- *Affine constrained automaton* given by:
  - A finite automaton
  - A map from the transitions to affine functions
  - A semilinear set

$$\bigcirc \xrightarrow{a, \vec{x} \leftarrow M.\vec{x} + \vec{v}} \bigcirc$$

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

Affine constrained automata

Adding linear transformations to CA transitions

Definition (Affine Constrained Automata (ACA))

- *Affine constrained automaton* given by:
  - A finite automaton
  - A map from the transitions to affine functions
  - A semilinear set

$$\bigcirc \xrightarrow{a, \vec{x} \leftarrow M.\vec{x} + \vec{v}} \bigcirc$$

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

Related to *linear systems with Presburger guards* [Finkel and Leroux, 2002]

A comparison with CA

How much more powerful are ACA?

CA variants are restrictions of ACA:

Theorem

- *DetCA = [DetACA where M is a permutation matrix]*

A comparison with CA

How much more powerful are ACA?

CA variants are restrictions of ACA:

Theorem

- *DetCA = [DetACA where M is a permutation matrix]*
- *UnCA = [DetACA where $\prod M$ is in a finite set]*

Parikh image

$\mathrm{Pkh}(w) = (|w|_a, \ldots)$

SL set

$\cup(\vec{c_0} + \sum_i \vec{c_i} \cdot k_i)$

Constr. Lang.

$L\restriction_C = \{w \in L \mid \mathrm{Pkh}(w) \in C\}$

RCL

constraining #letters

Unambiguous

One acc. run per word

(Det|Un)CA

automaton constraining #transitions

(Det)ACA

CA with func

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

A comparison with CA

How much more powerful are ACA?

CA variants are restrictions of ACA:

Theorem

- *DetCA = [DetACA where M is a permutation matrix]*
- *UnCA = [DetACA where $\prod M$ is in a finite set]*
- *CA = [ACA where $\prod M$ is in a finite set]*

Parikh image

$\mathrm{Pkh}(w) = (|w|_a, \ldots)$

SL set

$\bigcup(\vec{c_0} + \sum_i \vec{c_i} \cdot k_i)$

Constr. Lang.

$L{\restriction}_C = \{w \in L \mid \mathrm{Pkh}(w) \in C\}$

RCL

constraining #letters

Unambiguous

One acc. run per word

(Det|Un)CA

automaton constraining #transitions

(Det)ACA

CA with func

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

A comparison with CA

How much more powerful are ACA?

CA variants are restrictions of ACA:

Theorem

- *DetCA = [DetACA where M is a permutation matrix]*
- *UnCA = [DetACA where $\prod M$ is in a finite set]*
- *CA = [ACA where $\prod M$ is in a finite set]*
- *Over unary languages: CA $\subsetneq$ DetACA*

A comparison with CA

How much more powerful are ACA?

CA variants are restrictions of ACA:

Theorem

- *DetCA = [DetACA where M is a permutation matrix]*
- *UnCA = [DetACA where $\prod M$ is in a finite set]*
- *CA = [ACA where $\prod M$ is in a finite set]*
- *Over unary languages: CA $\subsetneq$ DetACA*
- *DetACA = UnACA*

A comparison with CA

How much more powerful are ACA?

CA variants are restrictions of ACA:

Theorem

- *DetCA = [DetACA where M is a permutation matrix]*
- *UnCA = [DetACA where $\prod M$ is in a finite set]*
- *CA = [ACA where $\prod M$ is in a finite set]*
- *Over unary languages: CA $\subsetneq$ DetACA*
- *DetACA = UnACA*

Matrix manipulation → Unambiguity

Closure properties of ACA

We investigate the closure properties of ACA:

Theorem

	$\cup$	$\cap$	$-$	$\cdot$	*rev*
DetACA	*Y*	*Y*	*Y*	*N*	*Y*
ACA	*Y*	*Y*	*N*	*Y*	*Y*

Closure properties of ACA

We investigate the closure properties of ACA:

Theorem

	$\cup$	$\cap$	$-$	$\cdot$	*rev*
DetACA	*Y*	*Y*	*Y*	*N*	*Y*
ACA	*Y*	*Y*	*N*	*Y*	*Y*

- *Conditional* nonclosures

Closure properties of ACA

We investigate the closure properties of ACA:

Theorem

	$\cup$	$\cap$	$-$	$\cdot$	*rev*
DetACA	*Y*	*Y*	*Y*	*N*	*Y*
ACA	*Y*	*Y*	*N*	*Y*	*Y*

- *Conditional* nonclosures
- $\exists$ NP-complete languages in ACA, so

Closure properties of ACA

We investigate the closure properties of ACA:

Theorem

	$\cup$	$\cap$	$-$	$\cdot$	*rev*
DetACA	*Y*	*Y*	*Y*	*N*	*Y*
ACA	*Y*	*Y*	*N*	*Y*	*Y*

- *Conditional* nonclosures
- $\exists$ NP-complete languages in ACA, so

$$\mathrm{DetACA} = \mathrm{ACA} \Rightarrow \mathrm{P} = \mathrm{NP}$$

An algebraic theory of ACA

Using *finitely typed monoids* [Krebs et al., 2007]

Parikh image

$\mathrm{Pkh}(w) = (|w|_a, \ldots)$

SL set

$\bigcup(\vec{c_0} + \sum_i \vec{c_i} \cdot k_i)$

Constr. Lang.

$L{\restriction_C} = \{w \in L \mid \mathrm{Pkh}(w) \in C\}$

RCL

constraining #letters

Unambiguous

One acc. run per word

(Det|Un)CA

automaton constraining #transitions

(Det)ACA

CA with func

$\bigcirc \xrightarrow{a, \vec{x} \leftarrow M.\vec{x} + \vec{v}} \bigcirc$

An algebraic theory of ACA

Using *finitely typed monoids* [Krebs et al., 2007]

- Consequence for DetACA:

An algebraic theory of ACA

Using *finitely typed monoids* [Krebs et al., 2007]

- Consequence for DetACA:

Definition (Rational series [Schützenberger, 1961])

Series: *infinite polynomials* with words $\in \Sigma^*$ as unknowns, $\mathbb{Z}$ as coefficients

An algebraic theory of ACA

Using *finitely typed monoids* [Krebs et al., 2007]

- Consequence for DetACA:

Definition (Rational series [Schützenberger, 1961])

Series: *infinite polynomials* with words $\in \Sigma^*$ as unknowns, $\mathbb{Z}$ as coefficients

$s = 3.\varepsilon + 6ab - ba + 2aa + 3aaa + \ldots$

An algebraic theory of ACA

Using *finitely typed monoids* [Krebs et al., 2007]

- Consequence for DetACA:

Definition (Rational series [Schützenberger, 1961])

Series: *infinite polynomials* with words $\in \Sigma^*$ as unknowns, $\mathbb{Z}$ as coefficients

An algebraic theory of ACA

Using *finitely typed monoids* [Krebs et al., 2007]

- Consequence for DetACA:

Definition (Rational series [Schützenberger, 1961])

Series: *infinite polynomials* with words $\in \Sigma^*$ as unknowns, $\mathbb{Z}$ as coefficients
Rational series: closure of (usual, finite) polynomials under $+$, $\times$, and $s^* = \sum_n s^n$

An algebraic theory of ACA

Using *finitely typed monoids* [Krebs et al., 2007]

- Consequence for DetACA:

Definition (Rational series [Schützenberger, 1961])

Series: *infinite polynomials* with words $\in \Sigma^*$ as unknowns, $\mathbb{Z}$ as coefficients

Rational series: closure of (usual, finite) polynomials under $+$, $\times$, and $s^* = \sum_n s^n$

Theorem

$L \in$ DetACA iff L is a Boolean combination of
$\{w \mid \text{coeff. of } w \text{ in } s > 0\}$ *for a rational series s*

An algebraic theory of ACA

Using *finitely typed monoids* [Krebs et al., 2007]

- Consequence for DetACA:

Definition (Rational series [Schützenberger, 1961])

Series: *infinite polynomials* with words $\in \Sigma^*$ as unknowns, $\mathbb{Z}$ as coefficients
Rational series: closure of (usual, finite) polynomials under $+$, $\times$, and $s^* = \sum_n s^n$

Theorem

$L \in$ DetACA iff L is a Boolean combination of
$\{w \mid \text{coeff. of } w \text{ in } s > 0\}$ *for a rational series s*

  - Connection with probabilistic languages, Markov chains, ... [Paz, 1971]

Parikh image

$\mathrm{Pkh}(w) = (|w|_a, \ldots)$

SL set

$\bigcup(\vec{c_0} + \sum_i \vec{c_i} \cdot k_i)$

Constr. Lang.

$L{\restriction_C} = \{w \in L \mid \mathrm{Pkh}(w) \in C\}$

RCL

constraining #letters

Unambiguous

One acc. run per word

(Det|Un)CA

automaton constraining #transitions

(Det)ACA

CA with func

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

Outline

Formal content of the thesis

- MC, A. Finkel, P. McKenzie. Affine Parikh Automata.

  In 3rd workshop on Non-Classical Models of Automata (NCMA'11)
  In RAIRO – Theoretical Informatics and Applications,
  NCMA'11 special issue, vol. ***46****(4), pp. 511–545, 2012*

Formal content of the thesis

- MC, A. Finkel, P. McKenzie. Affine Parikh Automata.

  In 3rd workshop on Non-Classical Models of Automata (NCMA'11)
  In RAIRO – Theoretical Informatics and Applications,
  NCMA'11 special issue, vol. ***46****(4), pp. 511–545, 2012*

- ——————. Bounded Parikh Automata.

  In 8th international conference on Words (WORDS'11)
  In International Journal of Foundations of Computer Science,
  WORDS'11 special issue, to appear

Formal content of the thesis

- MC, A. Finkel, P. McKenzie. Affine Parikh Automata.

  In 3rd workshop on Non-Classical Models of Automata (NCMA'11)
  In RAIRO – Theoretical Informatics and Applications, NCMA'11 special issue, vol. ***46****(4), pp. 511–545, 2012*

- ———————. Bounded Parikh Automata.

  In 8th international conference on Words (WORDS'11)
  In International Journal of Foundations of Computer Science, WORDS'11 special issue, to appear

- ———————. Unambiguous Constrained Automata.

  In 16th international conference on Developments in Language Theory (DLT'12)
  Invited to International Journal of Foundations of Computer Science, DLT'12 special issue

Formal content of the thesis

- MC, A. Finkel, P. McKenzie. Affine Parikh Automata.

  In 3rd workshop on Non-Classical Models of Automata (NCMA'11)
  In RAIRO – Theoretical Informatics and Applications, NCMA'11 special issue, vol. ***46****(4), pp. 511–545, 2012*

- ———————. Bounded Parikh Automata.

  In 8th international conference on Words (WORDS'11)
  In International Journal of Foundations of Computer Science, WORDS'11 special issue, to appear

- ———————. Unambiguous Constrained Automata.

  In 16th international conference on Developments in Language Theory (DLT'12)
  Invited to International Journal of Foundations of Computer Science, DLT'12 special issue

- MC, A. Krebs, P. McKenzie. The Algebraic Theory of Parikh Automata (in prep.).

What has been done, what is to be done

- Study of 3 models and their variants

What has been done, what is to be done

- Study of 3 models and their variants
- Decidability, closure, comparisons, restrictions

Parikh image
$\mathrm{Pkh}(w) = (|w|_a, \ldots)$

SL set
$\cup(\vec{c_0} + \sum_i \vec{c_i} \cdot k_i)$

Constr. Lang.
$L\restriction_C = \{w \in L \mid \mathrm{Pkh}(w) \in C\}$

RCL
constraining #letters

Unambiguous
One acc. run per word

(Det|Un)CA
automaton constraining #transitions

(Det)ACA
CA with func

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

What has been done, what is to be done

- Study of 3 models and their variants
- Decidability, closure, comparisons, restrictions
- An algebraic theory

What has been done, what is to be done

- Study of 3 models and their variants
- Decidability, closure, comparisons, restrictions
- An algebraic theory

Further work:

- Show $\Sigma^* \cdot \{a^n b^n\} \cdot \Sigma^* \notin$ UnCA, DetACA

What has been done, what is to be done

- Study of 3 models and their variants
- Decidability, closure, comparisons, restrictions
- An algebraic theory

Further work:

- Show $\Sigma^* \cdot \{a^n b^n\} \cdot \Sigma^* \notin$ UnCA, DetACA
- Characterize unary languages of ACA

Parikh image
$\text{Pkh}(w) = (|w|_a, \ldots)$

SL set
$\bigcup(\vec{c_0} + \sum_i \vec{c_i} \cdot k_i)$

Constr. Lang.
$L{\upharpoonright}_C = \{w \in L \mid \text{Pkh}(w) \in C\}$

RCL
constraining #letters

Unambiguous
One acc. run per word

(Det|Un)CA
automaton constraining #transitions

(Det)ACA
CA with func
$\bigcirc \xrightarrow{a, \vec{x} \leftarrow M.\vec{x} + \vec{v}} \bigcirc$

What has been done, what is to be done

- Study of 3 models and their variants
- Decidability, closure, comparisons, restrictions
- An algebraic theory

Further work:

- Show $\Sigma^* \cdot \{a^n b^n\} \cdot \Sigma^* \notin$ UnCA, DetACA
- Characterize unary languages of ACA
- Give tight upper bounds on DetACA

What has been done, what is to be done

- Study of 3 models and their variants
- Decidability, closure, comparisons, restrictions
- An algebraic theory

Further work:

- Show $\Sigma^* \cdot \{a^n b^n\} \cdot \Sigma^* \notin$ UnCA, DetACA
- Characterize unary languages of ACA
- Give tight upper bounds on DetACA
- Study algebraic structures for CA, ACA

What has been done, what is to be done

- Study of 3 models and their variants
- Decidability, closure, comparisons, restrictions
- An algebraic theory

Further work:

- Show $\Sigma^* \cdot \{a^n b^n\} \cdot \Sigma^* \notin$ UnCA, DetACA
- Characterize unary languages of ACA
- Give tight upper bounds on DetACA
- Study algebraic structures for CA, ACA
- Do separation results impact complexity classes?

Parikh image
$\mathrm{Pkh}(w) = (|w|_a, \ldots)$

SL set
$\cup(\vec{c_0} + \sum_i \vec{c_i} \cdot k_i)$

Constr. Lang.
$L\restriction_C = \{w \in L \mid \mathrm{Pkh}(w) \in C\}$

RCL
constraining #letters

Unambiguous
One acc. run per word

(Det|Un)CA
automaton constraining #transitions

(Det)ACA
CA with func

$\bigcirc \xrightarrow{a, \vec{x} \leftarrow M.\vec{x} + \vec{v}} \bigcirc$

Thank you!

Bibliography I

Bouajjani, A. and Habermehl, P. (1999).
Symbolic reachability analysis of FIFO-channel systems with nonregular sets of configurations.
Theoretical Computer Science, 221(1–2):211–250.

Esparza, J., Ganty, P., and Majumdar, R. (2012).
A perfect model for bounded verification.
In *Proceedings of the 2012 27th Annual IEEE/ACM Symposium on Logic in Computer Science*, LICS '12, pages 285–294, Washington, DC, USA. IEEE Computer Society.

Bibliography II

- Finkel, A. and Leroux, J. (2002).
  How to compose Presburger-accelerations: applications to broadcast protocols.
  In *Proc. 22nd Conf. Found. of Software Technology and Theor. Comp. Sci. (FST&TCS'2002), Kanpur*, pages 145–156. Springer.
- Ginsburg, S. (1966).
  The Mathematical Theory of Context-Free Languages.
  McGraw-Hill, Inc., New York, NY, USA.
- Ginsburg, S. and Spanier, E. H. (1964).
  Bounded ALGOL-like languages.
  Transactions of the American Mathematical Society, 113(2):333–368.

Bibliography III

Ginsburg, S. and Spanier, E. H. (1966).
Semigroups, Presburger formulas and languages.
Pacific Journal of Mathematics, 16(2):285–296.

Greibach, S. A. (1981).
Formal languages: Origins and directions.
Annals of the History of Computing, 3(1):14–41.

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

Bibliography IV

Ibarra, O. H. and Seki, S. (2011).
Characterizations of bounded semilinear languages by one-way and two-way deterministic machines.
In *Automata and Formal Languages*, pages 211–224. Institute of Mathematics and Computer Science of Nyíregyháza College.

Karianto, W., Krieg, A., and Thomas, W. (2006).
On intersection problems for polynomially generated sets.
In *Automata, Languages and Programming*, volume 4052 of *Lecture Notes in Computer Science*, pages 516–527. Springer.

Bibliography V

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

- Krebs, A., Lange, K.-J., and Reifferscheid, S. (2007).
  Characterizing TC^0 in terms of infinite groups.
  Theory of Computing Systems, 40(4):303–325.

- Kudlek, M. (2007).
  On semilinear sets over commutative semirings.
  Fundam. Inf., 79(3-4):447–452.

Bibliography VI

Leroux, J. (2009).
The general vector addition system reachability problem by presburger inductive invariants.
In *Logic In Computer Science, 2009.*, pages 4–13.

Minsky, M. L. (1961).
Recursive unsolvability of Post's problem of "tag" and other topics in theory of Turing machines.
Annals of Mathematics, 74(3):pp. 437–455.

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

Bibliography VII

Paz, A. (1971).
Introduction to probabilistic automata (Computer science and applied mathematics).
Academic Press, Inc., Orlando, FL, USA.

Presburger, M. (1927).
Über de vollständigkeit eines gewissen systems der arithmetik ganzer zahlen, in welchen, die addition als einzige operation hervortritt.
In *Comptes Rendus du Premier Congrès des Mathématiciens des Pays Slaves*, pages 92–101, Warsaw.

Schützenberger, M.-P. (1961).
On the definition of a family of automata.
Information and Control, 4(2-3):245–270.