Restrictions and Extensions of Data Automata

Zhilin Wu

Institute of Software,
Chinese Academy of Sciences

LOCALI 2013,
Fragrant Hill Hotel,
Nov. 04-07, 2011

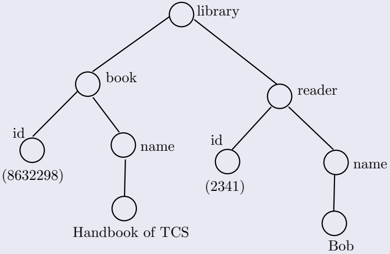
Motivation

Words or trees with infinite alphabet

XML documents

Trees with tags and attributes (data) on nodes

<library>
<book id ="8632298">
<name>Handbook of TCS </name>
</book>
<reader id ="2341">
<name>Bob</name>
</reader>
</library>



Verification

Timed system: timed words

$$(request, 1)(response, 1.5)(request, 2)(response, 4)\dots$$

Data words and languages

Data words

Infinite alphabet: $\Sigma \times \mathbb{D}$

- Σ: A finite letter domain,
- $\mathbb{D}$: An infinite data domain $\mathbb{D}$ (only (in)equality comparisons allowed)

Data word (w, d): a word over $\Sigma \times \mathbb{D}$, e.g. $\begin{matrix} a & b & a & b & a & b & b & a & b \\ 1 & 2 & 2 & 3 & 1 & 4 & 3 & 1 & 7 \end{matrix}$

A class of a data word: A maximal set of positions with the same data value.

$$\begin{matrix} \color{red}{a} & b & a & b & \color{red}{a} & b & b & \color{red}{a} & b \\ \color{red}{1} & 2 & 2 & 3 & \color{red}{1} & 4 & 3 & \color{red}{1} & 7 \end{matrix}$$

Data languages

Example. For every two a in the same class, there is a b between them in a different class.

$$\begin{matrix} \color{red}{a} & \color{blue}{b} & a & b & \color{red}{a} & \color{blue}{b} & b & \color{red}{a} & b \\ \color{red}{1} & \color{blue}{2} & 2 & 3 & \color{red}{1} & \color{blue}{4} & 3 & \color{red}{1} & 7 \end{matrix}$$

Automata models over data words

- Register automata (Kaminski & Francez 1994, Demri & Lazić 2006)

  Data values stored in the registers

- *Pebble automata* (Neven & Schwentick & Vianu 2001, Tan 2009)

  Pebbles placed on the positions of data words

- *Variable automata* (Grumberg & Kupferman & Sheinvald 2010)

  Add variables into the alphabet to symbolically represent data values

- Data automata and class automata (Bojańczyk & Muscholl & Schwentick & Segoufin 2006, Bojańczyk & Lasota 2010)

  Nondeterministic transducer + class condtion
  Introduced to prove the decidability of $FO^2[+1, <, \sim]$

Data automata

Profile of data words ($profile(w, d)$)

data word	$\begin{array}{ccccccccc} a & b & a & b & a & b & b & a & b \\ 1 \neq 2 = 2 \neq 3 \neq 1 \neq 4 \neq 3 \neq 1 \neq 7 \end{array}$
profile	$\begin{array}{ccccccccc} a & b & a & b & a & b & b & a & b \\ \bot & \bot & \top & \bot & \bot & \bot & \bot & \bot & \bot \end{array}$

Data automaton

A data automaton $\mathcal{D} = (\mathcal{A}, \mathcal{B})$

- a nondeterministic letter-to-letter transducer $\mathcal{A} : (\Sigma \times \{\bot, \top\})^* \to \Gamma^*$,
- class condition: a finite automaton $\mathcal{B}$ over the alphabet Γ.

Acceptance of a data word (w, d) by $\mathcal{D}$

- $\mathcal{A}$ generates a w' from $profile(w, d)$, and
- for each class X, the class string $w'|_X$ is accepted by $\mathcal{B}$

  $w'|_X$: the substring of w' restricted to positions in X

Data automata

Example

Let $\Sigma = \{a, b\}$. The language

$$\forall x(a(x) \to \exists y(x < y \wedge b(y) \wedge x \sim y))$$

is accepted by $\mathcal{D} = (\mathcal{A}, \mathcal{B})$

- $\mathcal{A}$ is the identity transducer: $(a, \{\perp, \top\}) \to a, (b, \{\perp, \top\}) \to b$,
- $\mathcal{B}$ is the automaton accepting $\Sigma^* b$.

$$\begin{matrix} 1 & 5 & 3 & 2 & 1 & 4 & 1 & 2 & 2 \\ a & b & b & a & a & b & b & a & b \end{matrix} \quad \begin{matrix} 1 & 1 & 1 \\ a & a & b \end{matrix} \quad \begin{matrix} 5 \\ b \end{matrix} \quad \begin{matrix} 3 \\ b \end{matrix} \quad \begin{matrix} 4 \\ b \end{matrix} \quad \begin{matrix} 2 & 2 & 2 \\ a & a & b \end{matrix}$$

Data automata

Example

Let $\Sigma = \{a, b\}$. The language

$$\forall x(a(x) \to \exists y(x < y \wedge b(y) \wedge x \sim y))$$

is accepted by $\mathcal{D} = (\mathcal{A}, \mathcal{B})$

- $\mathcal{A}$ is the identity transducer: $(a, \{\bot, \top\}) \to a, (b, \{\bot, \top\}) \to b$,
- $\mathcal{B}$ is the automaton accepting $\Sigma^* b$.

$$\begin{matrix} 1 & 5 & 3 & 2 & 1 & 4 & 1 & 2 & 2 \\ a & b & b & a & a & b & b & a & b \end{matrix} \qquad \begin{matrix} 1 & 1 & 1 \\ a & a & b \end{matrix} \qquad \begin{matrix} 5 \\ b \end{matrix} \quad \begin{matrix} 3 \\ b \end{matrix} \quad \begin{matrix} 4 \\ b \end{matrix} \qquad \begin{matrix} 2 & 2 & 2 \\ a & a & b \end{matrix}$$

Fact

Nonemptiness of data automata is decidable.

Open question

Whether the nonemptiness of data automata can be solved in elementary time?

Outline

1. Restriction
   - Weak data automata (WDA)
   - Commutative data automata (CDA)
   - Expressibility
   - Nonemptiness problem
2. Extension
   - Class automata
   - Priority multicounter automata (PMA)
   - Class automata with priority class condition (PCA)
   - Expressibility
   - Correspondence between PMA and PCA
3. Conclusion

Outline

1. Restriction
   - Weak data automata (WDA)
   - Commutative data automata (CDA)
   - Expressibility
   - Nonemptiness problem
2. Extension
   - Class automata
   - Priority multicounter automata (PMA)
   - Class automata with priority class condition (PCA)
   - Expressibility
   - Correspondence between PMA and PCA
3. Conclusion

Weak data automata (WDA)

A WDA $\mathcal{D} = (\mathcal{A}, \mathcal{C})$

- a nondeterministic letter-to-letter transducer $\mathcal{A} : (\Sigma \times \{\bot, \top\})^* \to \Gamma^*$,
- the condition $\mathcal{C}$: A collection of
  - key constraints $key(\gamma)$:

    Every two γ-positions have different data values
  - inclusion constraints $D(\gamma) \subseteq \bigcup_{\gamma' \in R} D(\gamma')$:

    For every data value occurring in a γ-position,
    there is $\gamma' \in R$ s.t. the data value also occurs in a γ'-position
  - and denial constraints $D(\gamma) \cap D(\gamma') = \emptyset$:

    No data value occurs in both a γ-position and a γ'-position

Example: Let (w, d) be the following data word

$$\begin{array}{cccccc} 1 & 2 & 1 & 3 & 2 & 4 \\ a & a & b & c & c & b \end{array}$$

Then $(w, d) \models key(a) \ \wedge \ D(a) \subseteq D(b) \cup D(c) \ \wedge \ D(b) \cap D(c) = \emptyset$.

Weak data automata (WDA)

A WDA $\mathcal{D} = (\mathcal{A}, \mathcal{C})$

- a nondeterministic letter-to-letter transducer $\mathcal{A} : (\Sigma \times \{\perp, \top\})^* \to \Gamma^*$,
- the condition $\mathcal{C}$: A collection of
  - key constraints $key(\gamma)$:

    Every two γ-positions have different data values
  - inclusion constraints $D(\gamma) \subseteq \bigcup_{\gamma' \in R} D(\gamma')$:

    For every data value occurring in a γ-position, there is $\gamma' \in R$ s.t. the data value also occurs in a γ'-position
  - and denial constraints $D(\gamma) \cap D(\gamma') = \emptyset$:

    No data value occurs in both a γ-position and a γ'-position

Example: Let (w, d) be the following data word

$$\begin{matrix} 1 & 2 & 1 & 3 & 2 & 4 \\ a & a & b & c & c & b \end{matrix}$$

Then $(w, d) \models key(a) \;\wedge\; D(a) \subseteq D(b) \cup D(c) \;\wedge\; D(b) \cap D(c) = \emptyset$.

Theorem (Kara, Schwentick and Tan 2012).
Nonemptiness of WDA can be decided in 2NEXPTIME.

Weak data automata (WDA)

WDA $\mathcal{D} = (\mathcal{A}, \mathcal{C})$ seen as a DA

- a nondeterministic letter to letter transducer $\mathcal{A} : \Sigma^* \rightarrow \Gamma^*$,
- the condition $\mathcal{C}$: Intersection of class conditions
  - key constraints $key(\gamma)$:

    In each class, γ occurs at most once,
  - inclusion constraints $D(\gamma) \subseteq \bigcup_{\gamma' \in R} D(\gamma')$:

    In each class, if γ occurs at least once, then γ' occurs at least once for some $\gamma' \in R$,
  - and denial constraints $D(\gamma) \cap D(\gamma') = \emptyset$:

    In each class, if γ occurs at least once, then γ' does not occur.

Weak data automata (WDA)

WDA $\mathcal{D} = (\mathcal{A}, \mathcal{C})$ seen as a DA

- a nondeterministic letter to letter transducer $\mathcal{A} : \Sigma^* \to \Gamma^*$,
- the condition $\mathcal{C}$: Intersection of class conditions
  - key constraints $key(\gamma)$:

    In each class, γ occurs at most once,
  - inclusion constraints $D(\gamma) \subseteq \bigcup_{\gamma' \in R} D(\gamma')$:

    In each class, if γ occurs at least once,
    then γ' occurs at least once for some $\gamma' \in R$,
  - and denial constraints $D(\gamma) \cap D(\gamma') = \emptyset$:

    In each class, if γ occurs at least once, then γ' does not occur.

All these class conditions are **Commutative**

$$\forall \gamma_1, \gamma_2 \in \Gamma, \forall x, y \in \Gamma^*, x\gamma_1\gamma_2 y \in L \Leftrightarrow x\gamma_2\gamma_1 y \in L$$

Weak data automata (WDA)

WDA $\mathcal{D} = (\mathcal{A}, \mathcal{C})$ seen as a DA

- a nondeterministic letter to letter transducer $\mathcal{A} : \Sigma^* \rightarrow \Gamma^*$,
- the condition $\mathcal{C}$: Intersection of class conditions
  - key constraints $key(\gamma)$:

    In each class, γ occurs at most once,
  - inclusion constraints $D(\gamma) \subseteq \bigcup_{\gamma' \in R} D(\gamma')$:

    In each class, if γ occurs at least once,
    then γ' occurs at least once for some $\gamma' \in R$,
  - and denial constraints $D(\gamma) \cap D(\gamma') = \emptyset$:

    In each class, if γ occurs at least once, then γ' does not occur.

All these class conditions are **Commutative**

$$\forall \gamma_1, \gamma_2 \in \Gamma, \forall x, y \in \Gamma^*, x\gamma_1\gamma_2 y \in L \Leftrightarrow x\gamma_2\gamma_1 y \in L$$

Commutative Data Automata

Outline

Commutative Data automata (CDA)

A CDA $(\mathcal{A}, \mathcal{B})$: A data automaton $(\mathcal{A}, \mathcal{B})$ s.t.

$L(\mathcal{B})$ is a commutative regular language.

Commutative regular languages

Quantifier free simple Presburger formulas (QFSP):

Boolean combination of formulas of the form
$x_1 + \dots x_n \leq c$, $x_1 + \cdots + x_n = c$, $x_1 + \dots x_n \geq c$
$x_1 + \cdots + x_n \equiv r \mod q$.

Remark. If formulas of the form $x_i - x_j \leq c$ are added,
then we get quantifier free Presburger formulas (QFP).

Proposition (Pin 86). Let $\Gamma = \{\gamma_1, \dots, \gamma_k\}$ and $L \subseteq \Gamma^*$ be regular. Then
L is commutative iff L is defined by a QFSP formula $\varphi(x_{\gamma_1}, \dots, x_{\gamma_k})$.

Example. "Words of even length over the alphabet $\{a, b\}$": $x_a + x_b \equiv 0 \mod 2$.

Commutative Data automata (CDA)

A CDA $(\mathcal{A}, \mathcal{B})$: A data automaton $(\mathcal{A}, \mathcal{B})$ s.t.

$L(\mathcal{B})$ is a commutative regular language.

Commutative regular languages

Quantifier free simple Presburger formulas (QFSP):

Boolean combination of formulas of the form

$x_1 + \dots x_n \leq c$, $x_1 + \dots + x_n = c$, $x_1 + \dots x_n \geq c$

$x_1 + \dots + x_n \equiv r \mod q$.

Remark. If formulas of the form $x_i - x_j \leq c$ are added, then we get quantifier free Presburger formulas (QFP).

Proposition (Pin 86). Let $\Gamma = \{\gamma_1, \dots, \gamma_k\}$ and $L \subseteq \Gamma^*$ be regular. Then

L is commutative iff L is defined by a QFSP formula $\varphi(x_{\gamma_1}, \dots, x_{\gamma_k})$.

Example. "Words of even length over the alphabet $\{a, b\}$": $x_a + x_b \equiv 0 \mod 2$.

A CDA $\mathcal{D} = (\mathcal{A}, \varphi)$ s.t. φ is a QFSP formula

CDA: Example

L:

$$\exists x \exists y (a(x) \wedge x < y \wedge b(y) \wedge x \sim y).$$

L is defined by the CDA $\mathcal{D} = (\mathcal{A}, \varphi)$ defined as follows.

- *the transducer $\mathcal{A}$:*
  When reading the profile of a data word from left to right, $\mathcal{A}$
  - *nondeterministically chooses an occurrence of a, then an occurrence of b,*
  - *relabel them by \$,*
  - *and keep unchanged the letters in all the other positions.*
- $\varphi := x_{\$} = 2 \vee x_{\$} = 0$.

$$\begin{array}{ccccc} & d & & d & \\ \cdots & a & \cdots & b & \cdots \\ & \$ & & \$ & \end{array}$$

Outline

Expressibility comparison and closure properties

Theorem. $WDA < CDA < DA$.

Proof(sketch)

WDA $<$ CDA: "In each class of the data word,
the letter a occurs an even number of times"

CDA $<$ DA: "For each occurrence of a,
$\exists$ an occurrence of b on the right with the same data value"

Theorem. CDAs are closed under union and intersection, but not under complementation.

Proof(sketch)

Closed under union and intersection: Easy to show.
Non-closed under complementation:

The language L: $\begin{matrix} a & \dots & a & b & \dots & b \\ d_1 & \dots & d_k & d_1 & \dots & d_k \end{matrix}$

- L is not definable by DA,
- but the complement $\overline{L}$ can be defined by a CDA.

Outline

1. Restriction
   - Weak data automata (WDA)
   - Commutative data automata (CDA)
   - Expressibility
   - Nonemptiness problem

2. Extension
   - Class automata
   - Priority multicounter automata (PMA)
   - Class automata with priority class condition (PCA)
   - Expressibility
   - Correspondence between PMA and PCA

3. Conclusion

Nonemptiness

Theorem. Nonemptiness of CDA can be decided in 3NEXPTIME.

We illustrate the proof by using the following example.

Let $\Sigma = \{a, b\}$. Consider the CDA $\mathcal{D} = (\mathcal{A}, \varphi)$, where

- the transducer $\mathcal{A}$:
  - $(a, \{\perp, \top\}) \rightarrow a, (b, \{\perp, \top\}) \rightarrow b$,
  - over a data word (w, d),

    $\mathcal{A}$ verifies that $w \in (ab)^*$ and (w, d) is locally different $(\forall i. d_i \neq d_{i+1})$,

    that is, $profile(w, d) \in ((a, \perp)(b, \perp))^*$.
- $\varphi = (x_a \leq 1 \wedge x_b = 1) \vee (x_a \geq 2 \wedge x_b \equiv 1 \mod 2)$.

Nonemptiness

Theorem. Nonemptiness of CDA can be decided in 3NEXPTIME.

Proof.

1. Transform φ into a normal form $\bigvee_{1 \leq i \leq 6} \varphi_i$, where

- $\varphi_1 = (x_a = 0 \wedge x_b = 1)$,
- $\varphi_2 = (x_a = 1 \wedge x_b = 1)$,
- $\varphi_3 = (x_a \geq 2 \wedge x_a \equiv 0 \mod 2 \wedge x_b = 1)$,
- $\varphi_4 = (x_a \geq 2 \wedge x_a \equiv 0 \mod 2 \wedge x_b \geq 2 \wedge x_b \equiv 1 \mod 2)$,
- $\varphi_5 = (x_a \geq 2 \wedge x_a \equiv 1 \mod 2 \wedge x_b = 1)$,
- $\varphi_6 = (x_a \geq 2 \wedge x_a \equiv 1 \mod 2 \wedge x_b \geq 2 \wedge x_b \equiv 1 \mod 2)$.

Note that all those φ_i's are mutually exclusive.

Nonemptiness

Theorem. Nonemptiness of CDA can be decided in 3NEXPTIME.

Proof.

2. Forget "locally different" and consider the following problem.

Is there a data word (w, d) such that
$w \in (ab)^*$ and each class of (w, d) satisfies φ?

Lemma. $\exists$ data word (w, d) s.t. $w \in (ab)^*$ and each class of (w, d) satisfies φ iff $\exists$ word $w \in (ab)^*$ such that $w \models \exists y_1 \ldots \exists y_6 \psi$ (data-free).

The existential Presburger(EP) formula $\exists y_1 \ldots y_6 \psi$: $\psi = \psi_1 \wedge \psi_2 \wedge \psi_3$,

$$\psi_1 = \begin{array}{l} x_a \geq y_2 + 2y_3 + 2y_4 + 3y_5 + 3y_6 \ \wedge \\ x_b \geq y_1 + y_2 + y_3 + 3y_4 + y_5 + 3y_6 \end{array},$$

$$\psi_2 = \begin{array}{l} y_3 + y_4 + y_5 + y_6 = 0 \rightarrow x_a = y_2 \ \wedge \\ y_4 + y_6 = 0 \rightarrow x_b = y_1 + y_2 + y_3 + y_5 \end{array},$$

$$\psi_3 = \begin{array}{l} x_a - (y_2 + 2y_3 + 2y_4 + 3y_5 + 3y_6) \equiv 0 \mod 2 \ \wedge \\ x_b - (y_1 + y_2 + y_3 + 3y_4 + y_5 + 3y_6) \equiv 0 \mod 2 \end{array}.$$

The intuition of y_i:

Number of data values d s.t. the class corresponding to d satisfies φ_i.

Nonemptiness

Theorem. Nonemptiness of CDA can be decided in 3NEXPTIME.

Proof.

2. Forget "locally different" and consider the following problem.

   Is there a data word (w, d) such that

   $w \in (ab)^*$ *and each class of* (w, d) *satisfies* φ*?*

Presburger automaton $(\mathcal{A}, \psi)$:

- $\mathcal{A}$: finite-state automaton over Σ,
- $\psi((x_a)_{a \in \Sigma})$: Existential Presburger formula.

Acceptance: w is accepted by $(\mathcal{A}, \psi)$ iff
w is accepted by $\mathcal{A}$ and its Parikh image satisfies ψ

Theorem (Seidl, Schwentick, Muscholl and Habermehl 2004).
Nonemptiness of Presburger automata can be decided in NP.

Nonemptiness

Theorem. Nonemptiness of CDA can be decided in 3NEXPTIME.

Proof.

3. How about "locally different" ?

Lemma. There is a number $M : M \leq 2^{|\varphi|}$ s.t.

$\exists$ a locally different data word (w, d) satisfying
$w \in (ab)^*$ and $(w, d) \models \varphi$ with "many" data values w.r.t. M

iff $\exists$ a word $w \in (ab)^*$ satisfying
$w \models \exists y_1 \ldots \exists y_6 \psi$ with "large" numbers w.r.t. M.

Satisfaction with large numbers and many data values

- $w \models \exists y_1 \ldots y_6 \psi$ with large numbers (w.r.t. M):

  $w \models \psi[k_1/y_1, \ldots, k_6/y_6]$ *s.t. for every* i, *either* $k_i = 0$ *or* $k_i \geq M$.

- $(w, d) \models \varphi = \bigvee_{1 \leq i \leq 6} \varphi_i$ with many data values (w.r.t. M):

  $\forall i : 1 \leq i \leq 6$, *either there are no data values satisfying* φ_i, *or there are at least* M *data values satisfying* φ_i.

Nonemptiness

Theorem. Nonemptiness of CDA can be decided in 3NEXPTIME.

Proof.

3. How about "locally different" ?

Lemma. There is a number M : $M \leq 2^{|\varphi|}$ s.t.

$\exists$ a locally different data word (w, d) satisfying
$w \in (ab)^*$ and $(w, d) \models \varphi$ with "many" data values w.r.t. M

iff $\exists$ a word $w \in (ab)^*$ satisfying
$w \models \exists y_1 \ldots \exists y_6 \psi$ with "large" numbers w.r.t. M

iff $\exists$ a word $w \in (ab)^*$ satisfying

$$w \in (ab)^* \text{ and } w \models \exists y_1 \ldots \exists y_6 \left(\psi \wedge \bigwedge_{1 \leq i \leq 6} (y_i = 0 \vee y_i \geq M) \right).$$

Nonemptiness

Theorem. Nonemptiness of CDA can be decided in 3NEXPTIME.

Proof.

3. How about "locally different" ?

The general situation:

> *For some i, there are only "a few" data values satisfying φ_i.*

The idea:

> *Guess those indices i s.t. there are only "a few" data values satisfying φ_i.*

The algorithm.

i) Guess a subset J and set of constants D_j's,

- guess $J \subseteq [m] = \{1, \ldots, m\}$,
- for each $j \in J$, guess an integer $s_j \leq M$,
- for each $j \in J$, fix a set $D_j = \{\alpha^j_1, \ldots, \alpha^j_{s_j}\}$. Let $D_J = \bigcup_{j \in J} D_j$.

Nonemptiness

Theorem. Nonemptiness of CDA can be decided in 3NEXPTIME.

Proof.

3. How about "locally different" ?

The algorithm.

i) Guess a subset J and set of constants D_j's,

ii) Construct the NFA $\mathcal{A}'$ over the alphabet $\{a, b\} \cup \{a, b\} \times D_J$ s.t. $\mathcal{A}'$ accepts $w = \lambda_1 \ldots \lambda_n$ iff

- a symbol (γ, c) occurs in w iff $c \in D_j$ and either $x_\gamma = 1$ or $x_\gamma \geq 2$ occurs in φ_j,
- the projection of v over $\{a, b\}$ belongs to $(ab)^*$,
- for every i, if $\lambda_i = (\gamma, c)$ and $\lambda_{i+1} = (\gamma', c')$, then $c \neq c'$,
- for every $j \in J$ and $\gamma \in \{a, b\}$,
  if $x_\gamma = 1$ occurs in φ_j, then $\forall c \in D_j$, (γ, c) occurs exactly once,
  if $x_\gamma \geq 2 \wedge x_\gamma \equiv 0 \mod 2$ (resp. $x_\gamma \equiv 1 \mod 2$), then
  (γ, c) occurs at least twice and an even (resp. odd) number of times.

Nonemptiness

Theorem. Nonemptiness of CDA can be decided in 3NEXPTIME.

Proof.

3. How about "locally different" ?

The algorithm.

i) Guess a subset J and set of constants D_j's,

ii) Construct the NFA $\mathcal{A}'$ over the alphabet $\{a, b\} \cup \{a, b\} \times D_J$,

iii) Construct $\psi_J = \exists y_1 \ldots y_8 \left(\psi \wedge \bigwedge_{j \in J} y_j = 0 \wedge \bigwedge_{j \notin J} y_j \geq M \right)$,

iv) Decide the nonemptiness of the Presburger automaton $(\mathcal{A}', \psi_J)$.

Outline

1. Restriction
   - Weak data automata (WDA)
   - Commutative data automata (CDA)
   - Expressibility
   - Nonemptiness problem

2. Extension
   - Class automata
   - Priority multicounter automata (PMA)
   - Class automata with priority class condition (PCA)
   - Expressibility
   - Correspondence between PMA and PCA

3. Conclusion

Outline

An equivalent definition of data automata

A data automaton $\mathcal{D} = (\mathcal{A}, \mathcal{B})$

- a nondeterministic letter-to-letter transducer $\mathcal{A} : \Sigma^* \to \Gamma^*$,
- class condition: a finite automaton $\mathcal{B}$ over the alphabet Γ.

Acceptance of a data word (w, d) by $\mathcal{D}$

- $\mathcal{A}$ generates a w' from w, and
- for each class X, the class string $w'|_X$ is accepted by $\mathcal{B}$

  $w'|_X$: the substring of w' restricted to positions in X

The reason:

Given a data word (w, d),
a data automaton $(\mathcal{A}, \mathcal{B})$ can be constructed s.t.
over w, $\mathcal{A}$ can guess profile(w, d) and
$\mathcal{B}$ can verify the correctness of the guessing.

Class automata

Definition

A class automaton $\mathcal{D} = (\mathcal{A}, \mathcal{B})$

- a nondeterministic letter-to-letter transducer $\mathcal{A} : \Sigma^* \to \Gamma^*$,
- class condition: A finite automaton $\mathcal{B}$ over the alphabet $\Gamma \times \{0, 1\}$.

Acceptance of a data word (w, d) by $\mathcal{D}$

- $\mathcal{A}$ generates a w' from w, and
- for each class X, the class string $w' \otimes X$ is accepted by $\mathcal{B}$, where $w' \otimes X$ is obtained from w': $w'_i \to (w'_i, 1)$ if $i \in X$, otherwise $w'_i \to (w'_i, 0)$.

Fact

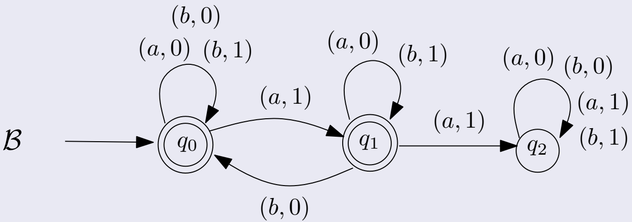
Nonemptiness of class automata is undecidable.

Class automata

Example

"For every two *a* in the same class, there is a *b* between them in a different class"
$\mathcal{D} = (\mathcal{A}, \mathcal{B})$,

- $\mathcal{A}$ is the identity transducer,
- $\mathcal{B}$: a finite automaton over the alphabet $\{a, b\} \times \{0, 1\}$



Class conditions and counter automata

	class condition		models of counter automata
$\mathfrak{A}_1$	no restriction	$\mathfrak{C}_1$	multicounter automata
$\mathfrak{A}_2$	local	$\mathfrak{C}_2$	multicounter automata without zero tests
$\mathfrak{A}_3$	tail	$\mathfrak{C}_4$	multicounter automata with increasing errors

Let $\pi : \Sigma \to \Gamma \cup \{\varepsilon\}$.
The projection of a data language $L \in (\Sigma \times \mathbb{D})^*$ under π:
The set of words $\pi(w)$ in Γ^*, where $(w, d) \in L$ for some d.

Correspondence (Bojańczyk & Lasota 2010)

For each $i = 1, 2, 3$, the following two language classes are the same,

- projections of data languages accepted by $\mathfrak{A}_i$,
- languages of words accepted by $\mathfrak{C}_i$.

Outline

1. Restriction
   - Weak data automata (WDA)
   - Commutative data automata (CDA)
   - Expressibility
   - Nonemptiness problem
2. Extension
   - Class automata
   - Priority multicounter automata (PMA)
   - Class automata with priority class condition (PCA)
   - Expressibility
   - Correspondence between PMA and PCA
3. Conclusion

Priority multicounter automata (PMA)

Priority multicounter automata

A multicounter automaton with the restricted zero tests:
The k counters in $\mathbb{C}$ are ordered into a sequence $C_1, \ldots, C_k$.
Restricted zero tests:

Select one $i \leq k$, and test whether for each $j \leq i$, $C_j = 0$.

Decidability (Reinhardt 2005)

The emptiness of PMA is decidable.

Priority multicounter automata (PMA)

Priority multicounter automata

A multicounter automaton with the restricted zero tests:
The k counters in $\mathbb{C}$ are ordered into a sequence $C_1, \ldots, C_k$.
Restricted zero tests:

Select one $i \leq k$, and test whether for each $j \leq i$, $C_j = 0$.

Decidability (Reinhardt 2005)

The emptiness of PMA is decidable.

Our goal: Priority class condition $\Leftrightarrow$ Priority multicounter automata

Outline

Intuition

Data automata as Class automata

Data automaton $\mathcal{D} = (\mathcal{A}, \mathcal{B}) \Rightarrow$ Class automaton $\mathcal{D}' = (\mathcal{A}, \mathcal{B}')$
$\mathcal{B}'$ is a finite automaton over the alphabet $\Gamma \times \{0, 1\}$.

$$\mathcal{B} : q \xrightarrow{\gamma} q' \quad \Rightarrow \quad \mathcal{B}' : q \xrightarrow{(\gamma,1)} q'$$

$$\mathcal{B}' : q \xrightarrow{(\gamma,0)} q$$

Intuition

Data automata as Class automata

Data automaton $\mathcal{D} = (\mathcal{A}, \mathcal{B}) \Rightarrow$ Class automaton $\mathcal{D}' = (\mathcal{A}, \mathcal{B}')$
$\mathcal{B}'$ is a finite automaton over the alphabet $\Gamma \times \{0, 1\}$.

$$\mathcal{B} : q \xrightarrow{\gamma} q' \quad \Rightarrow \quad \mathcal{B}' : q \overset{(\gamma,1)}{\longrightarrow} q'$$

$$\mathcal{B}' : q \overset{(\gamma,0)}{\longrightarrow} q$$

Priority class condition:
Restriction on the $(\gamma, 0)$-transitions of the class condition $\mathcal{B}$ to gain decidability.

0-priority finite state automata

Let $\mathcal{B}$ be a deterministic complete finite automaton over the alphabet $\Gamma \times \{0, 1\}$.

- G_0:

  Transition subgraph of $\mathcal{B}$ restricted to arcs labeled by $\Gamma \times \{0\}$.
- 0-cyclic state:

  A state belonging to a cycle in G_0, otherwise 0-acyclic.
- $G_{(\gamma,0)}$:

  Transition subgraph of $\mathcal{B}$ restricted to arcs labeled by $(\gamma, 0)$.
- $(\gamma, 0)$-cyclic state:

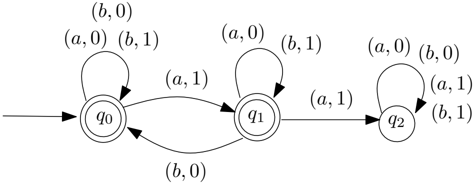
  A state belonging to a cycle in $G_{(\gamma,0)}$, otherwise $(\gamma, 0)$-acyclic.

0-priority finite state automata

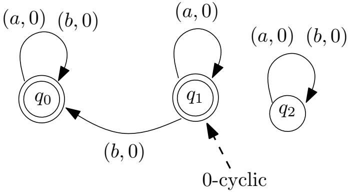
Let $\mathcal{B}$ be a deterministic complete finite automaton over the alphabet $\Gamma \times \{0, 1\}$.

Example

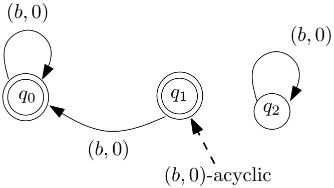
$\mathcal{B}$



G_0



$G_{(b,0)}$



0-priority finite state automata (continued)

Homogeneous G_0

Homogeneous SCC(strongly-connected-component) C in G_0:

$\forall \gamma \in \Gamma$, *either all states in* C *are* $(\gamma, 0)$*-cyclic or all are* $(\gamma, 0)$*-acyclic.*

Let $lab(C)$ denote the set of $\gamma \in \Gamma$ s.t. all states in C are $(\gamma, 0)$-cyclic.

G_0 is homogeneous if all its SCCs are homogeneous.

Suppose G_0 is homogeneous. Construct a labeled graph $\mathcal{D}_{scc}(G_0) = (V', E', L')$:

- V' is the set of all SCCs of G_0,
- $(C_1, \gamma, C_2) \in E'$ iff $\exists q_1 \in C_1, q_2 \in C_2$ s.t. $(q_1, (\gamma, 0), q_2) \in G_0$ (Note $\gamma \notin lab(C_1)$),
- $L'(C) = lab(C)$.

0-priority finite state automata (continued)

0-priority finite state automata

Let $\mathcal{A}$ be a deterministic complete finite state automaton over $\Gamma \times \{0, 1\}$.
$\mathcal{A}$ is a 0-priority finite state automaton if

there is an order of Γ, say $\gamma_1 \ldots \gamma_k$, s.t.

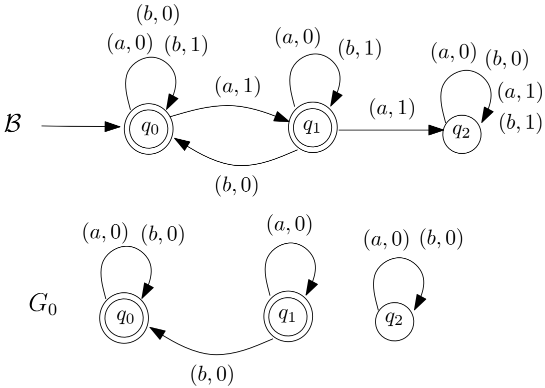
- *G_0 is homogeneous,*
- *every path $C_0 \gamma_{i_1} C_1 \ldots \gamma_{i_m} C_m$ in $\mathcal{G}_{scc}(G_0)$ s.t. C_0 is non-trivial respect the order of Γ, more specifically,*
  - *for every $1 \leq j_1 < j_2 \leq m$, $i_{j_1} < i_{j_2}$,*
  - *for every $i : 0 \leq j_1 < j_2 \leq m$ and every $\gamma_\ell \in lab(C_{i_{j_1}})$, it holds $\ell < i_{j_2}$.*

Proposition. Suppose $\mathcal{A}$ is a 0-priority finite state automaton.

- For every nontrivial SCC C in $\mathcal{G}_{scc}(G_0)$,
  $\exists i : 1 \leq i \leq k$ s.t. $lab(C) = \{\gamma_1, \ldots, \gamma_i\}$ (i: the index of C).
- No 0-acyclic states are reachable from 0-cyclic states in G_0.
- For every $(C, \gamma_i, C') \in E'$ s.t. C is nontrivial, it holds $\gamma_i \in lab(C')$.

0-priority finite state automata (continued)

Example:



Under the order $\gamma_1\gamma_2 = ab$, $\mathcal{B}$ is a 0-priority finite automaton

0-priority regular languages

Definition

$L \subseteq (\Gamma \times \{0,1\})^*$ is a 0-priority regular language if

there is a 0-priority finite automaton over the alphabet $\Gamma \times \{0,1\}$ *accepting* L.

Property

$L \subseteq (\Gamma \times \{0,1\})^*$ is a 0-priority regular language
iff
the unique minimal deterministic complete automaton accepting L
is a 0-priority finite automaton.

Class automata with priority class condition (PCA)

Definition

A class automaton $(\mathcal{A}, \mathcal{B})$ such that the output alphabet Γ of $\mathcal{A}$ can be partitioned into k (disjoint) subsets $\Gamma_1, \ldots, \Gamma_k$ satisfying that

$\mathcal{L}(\mathcal{B}) = L_1 \cup \cdots \cup L_k$, and for each $i : 1 \leq i \leq k$,
$L_i \subseteq (\Gamma_i \times \{0,1\})^*$ is a 0-priority regular language.

In particular, if $k = 1$, then $\mathcal{L}(\mathcal{B})$ is a 0-priority regular language.

Remark: It can be assumed that $\mathcal{A}$ satisfies the following condition,

Over each $w \in \Sigma^*$, $\mathcal{A}$ outputs a word w' in Γ_1^* or Γ_2^*, or $\ldots$, Γ_k^*.

Intuitively, over a data word (w, d), a PCA $\mathcal{D}$

- nondeterministically chooses a number $i : 1 \leq i \leq k$,
- guesses a word $w' \in \Gamma_i^*$,
- verifies each class string belongs to the 0-priority regular language L_i.

Class automata with priority class condition (PCA)

Definition

A class automaton $(\mathcal{A}, \mathcal{B})$ such that the output alphabet Γ of $\mathcal{A}$ can be partitioned into k (disjoint) subsets $\Gamma_1, \ldots, \Gamma_k$ satisfying that

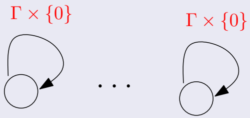
$\mathcal{L}(\mathcal{B}) = L_1 \cup \cdots \cup L_k$, and for each $i : 1 \leq i \leq k$,
$L_i \subseteq (\Gamma_i \times \{0, 1\})^*$ is a 0-priority regular language.

In particular, if $k = 1$, then $\mathcal{L}(\mathcal{B})$ is a 0-priority regular language.

Data automata as PCA

$\mathcal{B}$ can be seen as a 0-priority finite automaton $\mathcal{B}'$

$$q \xrightarrow{\gamma} q' \quad \Rightarrow \quad q \xrightarrow{(\gamma,1)} q' \qquad G_0$$



Outline

Expressibility of PCA

Theorem

PCAs are strictly more expressive than data automata.

Closure properties of PCA

- Closed under letter projection $h : \Sigma \to \Sigma'$: Nondeterminism of the transducer $\mathcal{A}$.
- Closed under union: By definition.
- Not closed under intersection or complementation:
  Otherwise, two-counter machines can be simulated,
  contradicting to the decidability of PCA.

Fact

Data automata are closed under intersection.

Outline

PMA $\Rightarrow$ PCA

A run of a PMA $\mathcal{C}$ can be encoded by a data word

From a PMA $\mathcal{C}$, a PCA $\mathcal{D} = (\mathcal{A}, \mathcal{B})$ can be constructed such that

- $\mathcal{A}$: The identity transducer
  check some regular (non-data) properties of data words,
- $\mathcal{B}$: A 0-priority finite automaton
  check the validity of all zero tests of $\mathcal{C}$.

PCA $\Rightarrow$ PMA

Let $\mathcal{D} = (\mathcal{A}, \mathcal{B})$ s.t. $\mathcal{A} = (Q_g, \Sigma, \Gamma, \delta_g, q_0^g, F_g)$ and $\mathcal{B} = (Q_c, \Gamma \times \{0,1\}, \delta_c, q_0^c, F_c)$.

A run of $\mathcal{D}$ over a data word (w, d) is a parallel running of

the transducer $\mathcal{A}$ and the copies of $\mathcal{B}$ over (w, d),
with one copy for each data value occurring in (w, d).

PCA $\Rightarrow$ PMA

Let $\mathcal{D} = (\mathcal{A}, \mathcal{B})$ s.t. $\mathcal{A} = (Q_g, \Sigma, \Gamma, \delta_g, q_0^g, F_g)$ and $\mathcal{B} = (Q_c, \Gamma \times \{0,1\}, \delta_c, q_0^c, F_c)$.

A run of $\mathcal{D}$ over a data word (w, d) is a parallel running of

the transducer $\mathcal{A}$ and the copies of $\mathcal{B}$ over (w, d),
with one copy for each data value occurring in (w, d).

Speficially, a run of $\mathcal{D}$ over (w, d) is a sequence

$$(q_1^g, q_1^c, \gamma_1, R_1)(q_2^g, q_2^c, \gamma_2, R_2) \ldots (q_{|w|}^g, q_{|w|}^c, \gamma_{|w|}, R_{|w|}) \text{ s.t.}$$

- the sequence $(q_1^g, \gamma_1) \ldots (q_{|w|}^g, \gamma_{|w|})$ corresponds to a run of $\mathcal{A}$,
  for each $1 \le i \le |w|$, $(q_{i-1}^g, w_i, q_i^g, \gamma_i) \in \delta_g$.
- $q_i^c = \delta^c(q_{i-1}^c, (\gamma_i, 0))$ records the state of a copy of $\mathcal{B}$
  for a data value that has not been met until the position i,
- R_i records the states of the copies of $\mathcal{B}$ for the data values that have been met until the position i:
  - If d_i has not been met before, then $R_i(d_i) = \delta_c(q_{i-1}^c, (\gamma_i, 1))$.
  - If d_i has been met before, then $R_i(d_i) = \delta_c(R_{i-1}(d_i), (\gamma_i, 1))$.
  - For each data value $d' \neq d_i$ that has been met before,
    $R_i(d') = \delta_c(R_{i-1}(d'), (\gamma_i, 0))$.

PCA $\Rightarrow$ PMA

Abstract runs

A run: $(q^g_1, q^c_1, \gamma_1, R_1)(q^g_2, q^c_2, \gamma_2, R_2)\dots(q^g_{|w|}, q^c_{|w|}, \gamma_{|w|}, R_{|w|})$.

Functions $R_1, \dots, R_{|w|} \Rightarrow$ Functions $C_1, \dots, C_{|w|}$

each C_i is a function $Q_c \to \mathbb{N}$ satisfying that
for each $q \in Q_c$, $C_i(q)$ is the number of data values
that have been met before the position i such that $R_i(d) = q$.

Intuitively,

each C_i is a tuple of counter values,
with one counter for each state in Q_c.

Abstract runs (continued)

The sequence $(q^g_1, q^c_1, \gamma_1, C_1)(q^g_2, q^c_2, \gamma_2, C_2) \ldots (q^g_{|w|}, q^c_{|w|}, \gamma_{|w|}, C_{|w|})$
can be seen in a more abstract way,
without directly referring to the data values in (w, d), as follows:

For each $1 < i \leq |w|$, C_i is obtained from C_{i-1} by nondeterministically choosing one of the following two possibilities:

- either (corresponding to the situation that d_i has been met before)
  - select some counter q' with non-zero value (i.e. $C_{i-1}(q') > 0$), decrement the counter q',
  - then for each counter q'',
    the value of $q'' \Leftarrow$ the sum of those of the counters p s.t. $\delta_c(p, (\gamma_i, 0)) = q''$,
  - finally increment the counter $\delta_c(q', (\gamma_i, 1))$.
- or (corresponding to the situation that d_i has not been met)
  - for each counter q'',
    the value of $q'' \Leftarrow$ the sum of those of the counters p s.t. $\delta_c(p, (\gamma_i, 0)) = q''$,
  - increment the counter $\delta_c(q^c_{i-1}, (\gamma_i, 1))$.

PCA $\Rightarrow$ PMA

Abstract runs (continued)

The sequence $(q^g_1, q^c_1, \gamma_1, C_1)(q^g_2, q^c_2, \gamma_2, C_2) \ldots (q^g_{|w|}, q^c_{|w|}, \gamma_{|w|}, C_{|w|})$
can be seen in a more abstract way,
without directly referring to the data values in (w, d), as follows:

For each $1 < i \leq |w|$, C_i is obtained from C_{i-1} by nondeterministically choosing one of the following two possibilities:

- either (corresponding to the situation that d_i has been met before)
- or (corresponding to the situation that d_i has not been met)

With such an abstract view of runs, $\mathcal{D}$ can be transformed into a counter automaton (with unrestricted zero tests) containing $k = |Q_c|$ counters.

PCA $\Rightarrow$ PMA

A property of abstract runs of PCA

Every abstract run of $\mathcal{D}$ satisfies that:

For each $i : 1 \leq i \leq |w|$,

$$\sum_{q:\ \textit{0-acyclic}} C_i(q) \leq \#_{scc}(G_0),$$

where $\#_{scc}(G_0)$ is the maximal length of paths in $D_{scc}(G_0)$.

Intuitively, for each i, at position i,
the number of data values d
such that $\mathcal{B}_d$ is in a 0-acyclic state
is bounded by a constant (independent of the data word).

PCA $\Rightarrow$ PMA

Abstract runs of PCA $\mathcal{D} = (\mathcal{A}, \mathcal{B}) \Longrightarrow$
PMA $\mathcal{C}'$ with one counter for each 0-cyclic state of $\mathcal{B}$,
with the info for 0-acyclic states recorded in the finite state control.

If $\mathcal{B}$ is a 0-priority finite automaton, then the counters of $\mathcal{C}'$ are ordered as follows:

$$Cyc_1 \ Cyc_2 \ \ldots \ Cyc_k.$$

where for every $i : 1 \leq i \leq k$,

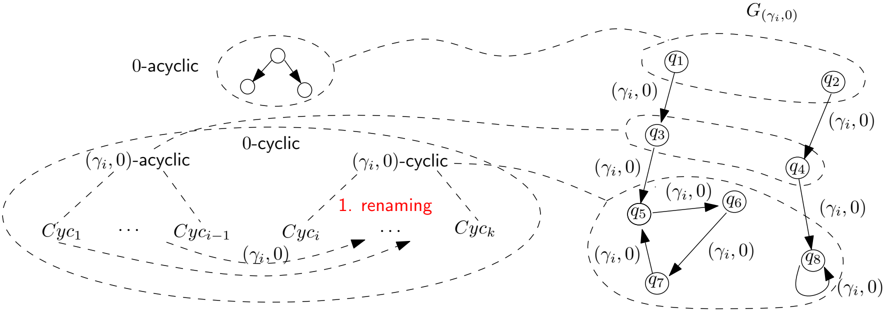
Cyc_i is the set of 0-cyclic states in Q_c belonging to an SCC of index i.

Remark. For every $j : j > i$, all states in Cyc_i are $(\gamma_j, 0)$-acyclic.

Similarly for the more general case that $\mathcal{B}$ is a union of 0-priority regular languages.

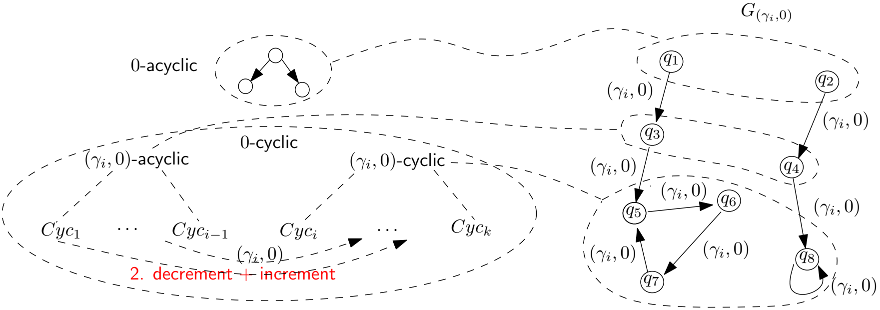
PCA ⇒ PMA (continued)

With this orderring of counters,
 the updates of counter values can be fullfilled
  with the restricted zero tests of priority multicounter automata.



PCA ⇒ PMA (continued)

With this orderring of counters,
the updates of counter values can be fullfilled
with the restricted zero tests of priority multicounter automata.

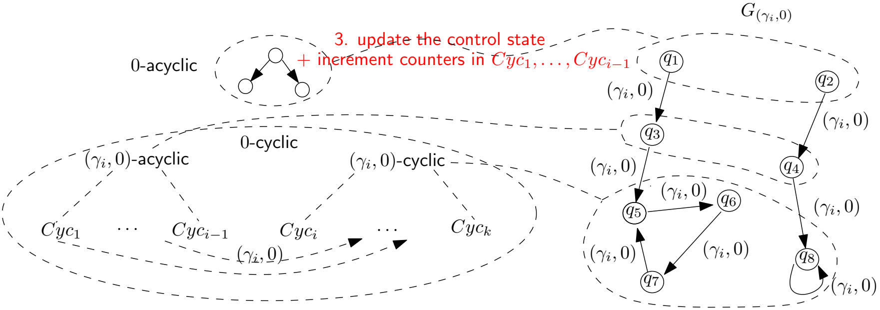


PCA ⇒ PMA (continued)

With this orderring of counters,

the updates of counter values can be fullfilled

with the restricted zero tests of priority multicounter automata.



Outline

Conclusion

Summary

1. Commutative data automata: Expressibility, complexity.
2. Class automata with priority class condition:
   Correspondence with priority multicounter automata (PMA).

Future work

- Lower bound for commutative data automata.
- "Partially" commutative data automata ?
- Commutative data automata over data trees ?