Commutative Data Automata[*]

Zhilin Wu

State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences,
P.O. Box 8718, # 4 Zhongguancun South 4th Street, Beijing, 100190, China
wuzl@ios.ac.cn

Abstract

Formalisms over infinite alphabets have recently received much focus in the community of theoretical computer science. Data automaton is a formal model for words over an infinite alphabet, that is, the product of a finite set of labels and an infinite set of data values, proposed by Bojanczyk, Muscholl, Schwentick et. al. in 2006. A data automaton consists of two parts, a nondeterministic letter-to-letter transducer, and a class condition specified by a finite automaton, which acts as a condition on each subword of the outputs of the transducer in corresponding to a maximal set of positions with the same data value. It is open whether the nonemptiness of data automata can be decided with elementary complexity, since this problem is equivalent to the reachability of Petri nets. Very recently, a restriction of data automata with elementary complexity, called weak data automata, was proposed by Kara, Schwentick and Tan and its nonemptiness problem was shown to be in 2-NEXPTIME. In weak data automata, the class conditions are specified by some simple constraints on the number of occurrences of labels occurring in every class. The aim of this paper is to demonstrate that the commutativity of class conditions is the genuine reason accounting for the elementary complexity of weak data automata. For this purpose, we define and investigate commutative data automata, which are data automata with class conditions restricted to commutative regular languages. We show that while the expressive power of commutative data automata is strictly stronger than that of weak data automata, the nonemptiness problem of this model can still be decided with elementary complexity, more precisely, in 3-NEXPTIME. In addition, we extend the results to data ω-words and prove that the nonemptiness of commutative Büchi data automata can be decided in 4-NEXPTIME. We also provide logical characterizations for commutative (Büchi) data automata, similar to those for weak (Büchi) data automata.

1998 ACM Subject Classification F.1.1, F4

Keywords and phrases Data Automata, Commutative regular languages, Presburger arithmetic, Existential Monadic Second-order logic, Büchi automata

Digital Object Identifier 10.4230/LIPIcs.CSL.2012.528

1 Introduction

With the momentums from the XML document processing and the verification of computer programs, formalisms over infinite alphabets have been intensively investigated in recent years. In the database community, XML documents are usually represented by trees, where the nodes can have tags together with several attributes e.g. identifiers. While the tags are from a finite set, the attributes may take values from some infinite domains. On the other hand, in the verification community, take concurrent systems as an example, if there is an unbounded number of processes in the system, then the behavior of the global system consists of the sequences of observed events attached with the process identifiers.

[*] Supported by the National Natural Science Foundation of China under Grant No. 61100062.

With these motivations, researchers in the two communities have investigated various formalisms over infinite alphabets, to name a few, register automata ([11]), pebble automata ([14]), data automata ([1]), XPath with data values ([9, 8]), LTL with freeze quantifiers ([6]), as well as two-variable logic interpreted on words or trees over infinite alphabets ([1, 3]). A survey on this topic can be found in [16].

By infinite alphabet, we mean $\Sigma \times \mathbb{D}$, with Σ a finite set of tags (labels) and $\mathbb{D}$ an infinite data domain. Words and trees over the alphabet $\Sigma \times \mathbb{D}$ are called data words and data trees.

The model of data automata was introduced by Bojanczyk, Muscholl, Schwentick, et. al. in [1] to prove the decidability of two-variable logic over data words. A data automaton $\mathscr{D}$ over data words consists of two parts, a nondeterministic letter-to-letter transducer $\mathscr{A}$, and a class condition specified by a finite automaton $\mathscr{B}$ over the output alphabet of $\mathscr{A}$, which acts as a condition on the subsequence of the outputs of $\mathscr{A}$ in every class, namely, every maximal set of positions with the same data value. By a reduction to the reachability problem of Petri nets (also called multicounter machines), the nonemptiness of data automata was shown to be decidable. On the other hand, data automata are also powerful enough to simulate Petri nets easily. Since it is a well-known open problem whether the complexity of the reachability problem for Petri nets is elementary, it is also not known whether the complexity of the nonemptiness of data automata is elementary.

Aiming at lowering the complexity of data automata, a restriction of data automata, called weak data automata, was introduced very recently by Kara, Schwentick, and Tan ([12]). In weak data automata, the class conditions are replaced by some simple constraints on the number of occurrences of labels occurring in every class. The nonemptiness of weak data automata can be decided with elementary complexity, more precisely, in 2-NEXPTIME.

By comparing data automata with weak data automata, we notice that to simulate Petri nets in data automata, the ability to express the property $L_{a<b}$, "for every occurrence of a, there is an occurrence of b on the right with the same data value", is crucial; on the other hand, as shown in [12], $L_{a<b}$ is not expressible in weak data automata. It is a simple observation that $L_{a<b}$ is a non-commutative language while the class conditions of weak data automata are commutative. This suggests that the commutativity of class conditions might be the *genuine* reason accounting for the elementary complexity of weak data automata. With this observation, we are motivated to define and investigate commutative data automata, which are data automata with class conditions restricted to commutative regular languages. We would like to see that the nonemptiness of commutative data automata can still be decided with elementary complexity, even though they have stronger class conditions than weak data automata. This is indeed the case, as we will show in this paper.

More specifically, the contributions of this paper consist of the following three aspects.

1. At first, we investigate the expressibility of commutative data automata. We show that the expressive power of commutative data automata lies strictly between data automata and weak data automata. In addition, commutative data automata are closed under intersection and union, but not under complementation. We also present a logical characterization of commutative data automata, similar to that for weak data automata.
2. The nonemptiness of commutative data automata can be decided in 3-NEXPTIME, which is the main result of this paper.
3. At last, we extend the results to data ω-words. We define commutative Büchi data automata and prove that the nonemptiness of commutative Büchi data automata can be decided in 4-NEXPTIME.

The main ideas of most of the proofs in this paper come from those for weak data automata ([12, 5]). Nevertheless, some proof steps become much more involved as a result of the stronger class conditions in commutative data automata.

Related work.

Several variants of data automata have been investigated. Bojanczyk and Lasota proposed an (undecidable) extension of data automata, called class automata, to capture the full XPath with data values over data trees; in addition, they established the correspondences of various class conditions of class automata over data words with the various models of counter machines ([2]). We continued this line of research by introducing another decidable extension of data automata over data words and establishing the correspondence with priority multicounter machines ([19]). There is another model, called class counting automata, relevant to this paper. Class counting automata over data words was proposed by Manuel and Ramanujan in [13]. In class counting automata, each data value is assigned a counter; and in each transition step, if the value of the counter corresponding to the current data value satisfies some constraint, then the value of the counter is updated according to a prescribed instruction; a run is accepting if a final state is reached in the end. The nonemptiness of class counting automata was shown to be EXPSPACE-complete. Nevertheless, the expressive power of class counting automata is relatively weak, for instance, the property "Each data value occurs exactly twice" cannot be expressed by class counting automata, while this property can be easily expressed by commutative data automata. Commutative regular languages have been investigated by many researchers. Pin presented a counting characterization of the expressibility of commutative regular languages ([15]). Gomez and Alvarez investigated how commutative regular languages can be learned from positive and negative examples ([10]). Chrobak and To proposed polynomial time algorithms to obtain regular expressions from nondeterministic finite automata over unary alphabets ([4, 18]).

The rest of this paper is organized as follows. Definitions are given in the next section. Then in Section 3, the expressibility of commutative data automata is investigated. Section 4 includes the main result of this paper, a 3-NEXPTIME algorithm for the nonemptiness of commutative data automata. Finally the results are extended to data ω-words in Section 5. The missing proofs can be found in the full version of this paper (http://lcs.ios.ac.cn/~wuzl/pub/cda-wu-12.pdf).

2 Preliminaries

Let Σ be a finite alphabet. A finite word over Σ is an element of Σ^* and an ω-word over Σ is an element of Σ^ω.

2.1 Presburger formulas and Commutative regular languages

Existential Presburger formulas (EP formulas) over a variable set X are formulas of the form $\exists \bar{x} \varphi$, where φ is a quantifier-free Presburger formula, i.e. a Boolean combination of atomic formulas of the form $t \geq c$, or $t \leq c$, or $t = c$, or $t \equiv r \bmod p$, where $c, r, p \in \mathbb{N}$, $p \geq 2, 0 \leq r < p$ and t is a term defined by $t := c \mid cx \mid t_1 + t_2 \mid t_1 - t_2$, where $c \in \mathbb{N}, x \in X$.

Suppose $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$ and $v \in \Sigma^*$. The *Parikh image* of v, denoted by $\mathsf{Parikh}(v)$, is a k-tuple $(\#_{\sigma_1}(v), \ldots, \#_{\sigma_k}(v))$, where for each $i : 1 \leq i \leq k$, $\#_{\sigma_i}(v)$ is the number of occurrences of σ_i in v. Let $V_\Sigma = \{x_{\sigma_1}, \ldots, x_{\sigma_k}\}$ and φ be an EP formula with free variables from V_Σ. The word v is said to satisfy φ, denoted by $v \models \varphi$, iff $\varphi[\mathsf{Parikh}(v)]$ holds. The *language defined by* φ, denoted by $\mathcal{L}(\varphi)$, is the set of words $v \in \Sigma^*$ such that $v \models \varphi$.

A *Presburger automaton* over the alphabet Σ is a binary tuple $(\mathscr{A}, \varphi)$, where $\mathscr{A}$ is a finite automaton over the alphabet Σ and φ is an EP formula with free variables from V_Σ. A word $v \in \Sigma^*$ is accepted by a Presburger automaton $(\mathscr{A}, \varphi)$ iff v is accepted by $\mathscr{A}$ and at the same time $v \models \varphi$. The language accepted by a Presburger automaton $(\mathscr{A}, \varphi)$, denoted by $\mathcal{L}((\mathscr{A}, \varphi))$, is the set of words accepted by $(\mathscr{A}, \varphi)$.

▶ **Theorem 1** ([17]). *The nonemptiness of Presburger automata can be decided in NP.*

Let L be a language over the alphabet Σ. Then L is *commutative* iff for any $\sigma_1, \sigma_2 \in \Sigma$ and $u, v \in \Sigma^*$, $u\sigma_1\sigma_2 v \in L$ iff $u\sigma_2\sigma_1 v \in L$. Commutative regular languages have a characterization in quantifier-free simple Presburger formulas defined in the following.

Quantifier-free simple Presburger formulas (QFSP formulas) over a variable set X are Boolean combinations of atomic formulas of the form $x_1 + \cdots + x_n \leq c$, or $x_1 + \cdots + x_n \geq c$, or $x_1 + \cdots + x_n = c$, or $x_1 + \cdots + x_n \equiv r \bmod p$, where $x_1, \ldots, x_n \in X$, $c, r, p \in \mathbb{N}$, $p \geq 2$, and $0 \leq r < p$.

Let $V_\Sigma = \{x_{\sigma_1}, \ldots, x_{\sigma_k}\}$ and φ be a QFSP formula over the variable set V_Σ. Similar to EP formulas, we can define $\mathcal{L}(\varphi)$, the language defined by φ.

For a set of variables $\{x_1, \ldots, x_k\}$, we use the notation $\varphi(x_1, \ldots, x_k)$ to denote an EP or QFSP formula φ with the free variables from $\{x_1, \ldots, x_k\}$.

▶ Proposition 2 ([15]). *Let L be a language over the alphabet $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$. Then L is a commutative regular language iff L is defined by a QFSP formula $\varphi(x_{\sigma_1}, \ldots, x_{\sigma_k})$.*

The *size* of an EP or QFSP formula φ, denoted by $|\varphi|$, is defined as the length of a binary encoding of φ (where the constants c, r and p are encoded in binary).

▶ Proposition 3. *Let $\varphi(x_1, \ldots, x_k)$ be a QFSP formula. Then there exists an exponential-time algorithm to transform φ into a QFSP formula $\bigvee\limits_{i:1 \leq i \leq m} \varphi_i$ of size $2^{O(k|\varphi|)}$ such that there is $p_0 : 2 \leq p_0 \leq 2^{|\varphi|}$ satisfying that*

- *each φ_i is of the form $\bigwedge\limits_{1 \leq j \leq k} \varphi_{i,j}$;*
- *for each $j : 1 \leq j \leq k$, $\varphi_{i,j}$ is equal to $x_j = c_{i,j}$ or $x_j \geq p_0 \wedge x_j \equiv r_{i,j} \bmod p_0$ for $c_{i,j}, r_{ij} : 0 \leq c_{i,j}, r_{i,j} < p_0$;*
- *in addition, those φ_i's are mutually exclusive.*

For a QFSP formula $\varphi(x_1, \ldots, x_k)$, the number p_0 and the QFSP formula $\bigvee\limits_{i:1 \leq i \leq m} \varphi_i$ in Proposition 3 are called respectively the *normalization* number and the *normal form* of φ.

▶ Remark. A weaker form of Proposition 3 was proved by Ehrenfeucht and Rozenberg in [7]. But they did not give the complexity bound.

2.2 Data words, two-variable logic and data automata

Let Σ be a finite alphabet and $\mathbb{D}$ be an infinite set of data values. A *data word* over Σ is an element of $(\Sigma \times \mathbb{D})^*$ and a *data ω-word* is an element of $(\Sigma \times \mathbb{D})^\omega$. Let $\sigma \in \Sigma$, a position in a data word or a data ω-word is called a *σ-position* if the position is labelled by σ.

Given a data (finite or ω) word $w = \binom{\sigma_1}{d_1}\binom{\sigma_2}{d_2}\ldots$, the *projection* of w to the finite alphabet Σ, denoted by $\mathsf{Proj}(w)$, is the (finite or ω) word $\sigma_1\sigma_2\ldots$. Let X be a set of positions in a word w, we use $w|_X$ to denote the restriction of w to the positions in X. Similarly, $w|_X$ can be defined for ω-words, data words and data ω-words.

Let $FO(+1, \sim, \Sigma)$ denote the first-order logic with the following atomic formulas, $\sigma(x)$ (where $\sigma \in \Sigma$), $x = y$, $x + 1 = y$, and $x \sim y$. Two positions x, y satisfy $x + 1 = y$ if y is the successor of the position x, and two positions satisfy $x \sim y$ if they have the same data value. Let $FO^2(+1, \sim, \Sigma)$ denote the two-variable fragment of $FO(+1, \sim, \Sigma)$. In addition, let $EMSO^2(+1, \sim, \Sigma)$ denote the extension of $FO^2(+1, \sim, \Sigma)$ by existential monadic second-order quantifiers in front of the $FO^2(+1, \sim, \Sigma)$ formulas. The data language defined by an $EMSO^2(+1, \sim, \Sigma)$ sentence φ, denoted by $\mathcal{L}(\varphi)$, is the set of data words satisfying φ.

A *class* of a data (finite or ω) word w is a maximal nonempty set of positions in w with the same data value. Given a class X of a data word w, the *class string* of w corresponding to X is $\mathsf{Proj}(w|_X)$, the projection of $w|_X$.

Let $w = \binom{\sigma_1}{d_1}\binom{\sigma_2}{d_2}\dots\binom{\sigma_n}{d_n}$ be a data word and φ be a QFSP formula over the variable set V_Σ. Then w is said to *satisfy the class condition* φ, denoted by $w \models_c \varphi$, if for each class X of w, $\mathsf{Proj}(w|_X) \models \varphi$.

Given a data (finite or ω) word $w = \binom{\sigma_1}{d_1}\binom{\sigma_2}{d_2}\dots$, the *profile word* of w, denoted by $\mathsf{Profile}(w)$, is a word $(\sigma_1, s_1)(\sigma_2, s_2)\dots$ over the alphabet $\Sigma \times \{\bot, \top\}$ such that for every $i \geq 1$, we have $s_i = \top$ (resp. $s_i = \bot$) iff $d_i = d_{i+1}$ (resp. $d_i \neq d_{i+1}$), moreover, if w is finite and of length n, then $s_n = \bot$. A data (finite or ω) word $w = \binom{\sigma_1}{d_1}\binom{\sigma_2}{d_2}\dots$ is called *locally different* if for every $i \geq 1$, it holds $d_i \neq d_{i+1}$.

▶ **Definition 4.** A *data automaton* (DA) $\mathscr{D}$ is a tuple $(\mathscr{A}, \mathscr{B})$, where $\mathscr{A} = (Q_1, \Sigma \times \{\bot, \top\}, \Gamma, \delta_1, q_{0,1}, F_1)$ is a nondeterministic letter-to-letter transducer with the input alphabet $\Sigma \times \{\bot, \top\}$ and the output alphabet Γ, and $\mathscr{B} = (Q_2, \Gamma, \delta_2, q_{0,2}, F_2)$ is a finite automaton over the alphabet Γ.

A data word $w = \binom{\sigma_1}{d_1}\binom{\sigma_2}{d_2}\dots\binom{\sigma_n}{d_n}$ is *accepted* by a data automaton $\mathscr{D} = (\mathscr{A}, \mathscr{B})$ iff there is an accepting run of $\mathscr{A}$ over $\mathsf{Profile}(w)$ which produces a word $\gamma_1 \dots \gamma_n$ such that for each class X of $w' = \binom{\gamma_1}{d_1}\binom{\gamma_2}{d_2}\dots\binom{\gamma_n}{d_n}$, the class string $\mathsf{Proj}(w'|_X)$ is accepted by $\mathscr{B}$.

The *data language* defined by a data automaton $\mathscr{D}$, denoted by $\mathcal{L}(\mathscr{D})$, is the set of data words accepted by $\mathscr{D}$.

▶ **Definition 5** ([12])**.** A *weak data automaton* (WDA) is a tuple $(\mathscr{A}, \mathscr{C})$ such that $\mathscr{A} = (Q, \Sigma \times \{\bot, \top\}, \Gamma, \delta, q_0, F)$ is a letter-to-letter transducer and the class condition $\mathscr{C}$ is specified by a collection of

- key constraints of the form $\mathsf{Key}(\gamma)$ (where $\gamma \in \Gamma$), interpreted as "every two γ-positions have different data values",
- inclusion constraints of the form $D(\gamma) \subseteq \bigcup_{\gamma' \in R} D(\gamma')$ (where $\gamma \in \Gamma, R \subseteq \Gamma$), interpreted as "for every data value occurring in a γ-position, there is $\gamma' \in R$ such that the data value also occurs in a γ'-position",
- and denial constraints of the form $D(\gamma) \cap D(\gamma') = \emptyset$ (where $\gamma, \gamma' \in \Gamma$), interpreted as "no data value occurs in both a γ-position and a γ'-position".

A data word $w = \binom{\sigma_1}{d_1}\binom{\sigma_2}{d_2}\dots\binom{\sigma_n}{d_n}$ is accepted by a weak data automaton $\mathscr{D} = (\mathscr{A}, \mathscr{C})$ iff there is an accepting run of $\mathscr{A}$ over $\mathsf{Profile}(w)$ which produces a word $\gamma_1 \dots \gamma_n$ such that the data word $w' = \binom{\gamma_1}{d_1}\binom{\gamma_2}{d_2}\dots\binom{\gamma_n}{d_n}$ satisfies all the constraints in $\mathscr{C}$.

▶ **Definition 6.** A *commutative data automaton* (CDA) $\mathscr{D}$ is a tuple $(\mathscr{A}, \varphi)$ such that $\mathscr{A} = (Q, \Sigma \times \{\bot, \top\}, \Gamma, \delta, q_0, F)$ is a letter-to-letter transducer and φ is a QFSP formula over the variable set V_Γ.

A data word $w = \binom{\sigma_1}{d_1}\binom{\sigma_2}{d_2}\dots\binom{\sigma_n}{d_n}$ is accepted by a commutative data automaton $\mathscr{D} = (\mathscr{A}, \varphi)$ iff there is an accepting run of $\mathscr{A}$ over $\mathsf{Profile}(w)$ which produces a word $\gamma_1 \dots \gamma_n$ such that the data word $w' = \binom{\gamma_1}{d_1}\binom{\gamma_2}{d_2}\dots\binom{\gamma_n}{d_n}$ satisfies that $w' \models_c \varphi$.

▶ Remark. We choose to define the class conditions of commutative data automata by QFSP formulas, instead of finite automata with commutative transition relations. The main purpose of this choice is to ease the extension of the results to data ω-words (c.f. Section 5). In addition, in the definition of commutative data automata, we choose the input alphabet of the transducer to be $\Sigma \times \{\bot, \top\}$, instead of Σ. It seems for us that this choice strictly increases the expressive power of commutative data automata, but we admit that we do not know how to prove it at present. ◀

3 Expressiveness

In this section, we first show that the expressibility of CDA lies strictly between WDA and DA, then we discuss the closure properties of CDA and provide a logical characterization of CDA.

▶ **Theorem 7.** *WDA $<$ CDA $<$ DA.*

Proof.

CDA $<$ DA.

It was shown in [12] that the language "for every occurrence of a, there is an occurrence of b on the right with the same data value" cannot be expressed in WDA. The same proof can be applied to show that the language is not expressible in CDA. On the other hand, it is easy to see that the language can be defined by a DA.

WDA $<$ CDA.

From any WDA $(\mathscr{A}, \mathscr{C})$, an equivalent CDA $(\mathscr{A}, \varphi_{\mathscr{C}})$ can be constructed such that $\varphi_{\mathscr{C}} := \bigwedge\limits_{C \in \mathscr{C}} \varphi_C$, where φ_C is defined as follows,

- if C is of the form $\mathsf{Key}(\gamma)$, then $\varphi_C := x_\gamma \leq 1$,
- if C is of the form $D(\gamma) \subseteq \bigcup\limits_{\gamma' \in R} D(\gamma')$, then $\varphi_C := x_\gamma \geq 1 \rightarrow \sum\limits_{\gamma' \in R} x_{\gamma'} \geq 1$,
- if C is of the form $D(\gamma) \cap D(\gamma') = \emptyset$, then $\varphi_C := x_\gamma \geq 1 \rightarrow x_{\gamma'} = 0$.

For the strictness of the inclusion, it is easy to observe that the language "In each class of the data word, the letter a occurs an even number of times" is expressible in CDA. By some pumping argument, we can show that the language is not expressible in WDA. ◀

▶ Remark. According to the above reduction of WDA to CDA, we would like to say that in some sense, CDA = WDA + Modulo constraints in class conditions.

▶ **Theorem 8.** *CDAs are closed under union and intersection, but not closed under complementation.*

In the following, we define $EMSO^2_{\#}(+1, \sim, \Sigma)$, a counting extension of $EMSO^2(+1, \sim, \Sigma)$, and show that it is expressively equivalent to CDA.

The logic $EMSO^2_{\#}(+1, \sim, \Sigma)$ includes all the formulas of the form $\exists R_1 \dots R_l(\varphi \wedge \forall x \psi)$ (where $R_1, \dots, R_l$ are unary predicates), such that $\varphi \in FO^2(+1, \sim, \Sigma, R_1, \dots, R_l)$ and ψ is a Boolean combination of atomic formulas of the form $\sum\limits_{\tau \in \Delta} \#_{x \sim y \wedge \tau(y)}(y) \geq c$, or $\sum\limits_{\tau \in \Delta} \#_{x \sim y \wedge \tau(y)}(y) \leq c$, or $\sum\limits_{\tau \in \Delta} \#_{x \sim y \wedge \tau(y)}(y) = c$, or $\sum\limits_{\tau \in \Delta} \#_{x \sim y \wedge \tau(y)}(y) \equiv r \bmod p$, satisfying that $\Delta \subseteq \Sigma \times 2^{\{R_1, \dots, R_l\}}$, $c \in \mathbb{N}$, $p \geq 2$, $0 \leq r < p$, and if $\tau = (\sigma, \mathcal{R})$, then $\tau(y) = \sigma(y) \wedge \bigwedge\limits_{i: 1 \leq i \leq l} \eta_{R_i}(y)$, where $\eta_{R_i}(y) = R_i(y)$ if $R_i \in \mathcal{R}$, and $\eta_{R_i}(y) = \neg R_i(y)$ otherwise.

The semantics of $EMSO^2(+1, \sim, \Sigma)$ formulas can be extended naturally to $EMSO^2_{\#}(+1, \sim, \Sigma)$ formulas by interpreting formulas $\forall x \psi$ as the counting constraints for each class. Let's take the formula $\forall x \left(\#_{x \sim y \wedge \tau(y)}(y) \geq c\right)$ as an example: Given a data word w over the alphabet $\Sigma \times 2^{\{R_1, \dots, R_l\}}$, $w \models \forall x \left(\#_{x \sim y \wedge \tau(y)}(y) \geq c\right)$ iff for each class X of w, the number of τ-positions in X is at least c.

▶ **Theorem 9.** *$EMSO^2_{\#}(+1, \sim, \Sigma)$ and CDA are expressively equivalent.*

- *Given an $EMSO^2_{\#}(+1, \sim, \Sigma)$ formula $\exists R_1 \dots R_l(\varphi \wedge \forall x \psi)$, a CDA $\mathscr{D} = (\mathscr{A}, \varphi')$ of doubly exponential size can be constructed such that $\mathcal{L}(\mathscr{D}) = \mathcal{L}(\exists R_1 \dots R_l(\varphi \wedge \forall x \psi))$. In addition, the size of the output alphabet of $\mathscr{A}$ is at most exponential over the size of $\exists R_1 \dots R_l(\varphi \wedge \forall x \psi)$.*
- *Given a CDA $\mathscr{D} = (\mathscr{A}, \varphi)$, an $EMSO^2_{\#}(+1, \sim, \Sigma)$ formula φ' of polynomial size can be constructed such that $\mathcal{L}(\mathscr{D}) = \mathcal{L}(\varphi')$.*

4 The nonemptiness problem of CDA

In this section, we prove the main result of this paper.

▶ **Theorem 10.** *The nonemptiness of CDA can be decided in 3-NEXPTIME.*

The rest of this section is devoted to the proof of Theorem 10. Although the structure of the proof is similar to that for WDA in [12, 5], the proofs of several lemmas become more complicated.

Through this section, let $\mathscr{D} = (\mathscr{A}, \varphi)$ be a commutative data automaton such that $\mathscr{A} = (Q, \Sigma \times \{\bot, \top\}, \Gamma, \delta, q_0, F)$ and φ is a QFSP formula over the variable set V_Γ.

Because we are concerned with the nonemptiness problem, without loss of generality, we can assume that $\mathscr{A} = (Q, \Gamma \times \{\bot, \top\}, \delta, q_0, F)$ is just a finite automaton over the alphabet $\Gamma \times \{\bot, \top\}$. Then the nonemptiness of $\mathscr{D}$ is reduced to the following problem.

PROBLEM:	NONEMPTINESS-PROFILE
INPUT:	A finite automaton $\mathscr{A} = (Q, \Gamma \times \{\bot, \top\}, \delta, q_0, F)$ and a QFSP formula φ over V_Γ
QUESTION:	Is there a data word w over Γ such that $\mathsf{Profile}(w)$ is accepted by $\mathscr{A}$ and $w \models_c \varphi$?

The outline of the proof goes as follows.

- At first, a finite automaton $\mathscr{A}'$ of exponential size over the alphabet Γ', and a QFSP formula φ' in the normal form of doubly exponential size over the variable set $V_{\Gamma'}$, are constructed from $\mathscr{D} = (\mathscr{A}, \varphi)$ such that the problem of NONEMPTINESS-PROFILE is reduced to the following problem,

  "is there a *locally different* data word w over the alphabet Γ' such that $\mathsf{Proj}(w)$ is accepted by $\mathscr{A}'$ and $w \models_c \varphi'$?"

  We would like to point out that the finite automaton $\mathscr{A}'$ runs directly on the *projections* of data words, instead of the profile words of them.

  Let's call this problem NONEMPTINESS-LOCALLY-DIFFERENT, which is formally defined as follows.

PROBLEM:	NONEMPTINESS-LOCALLY-DIFFERENT
INPUT:	A finite automaton $\mathscr{A} = (Q, \Gamma, \delta, q_0, F)$ and a QFSP formula φ over V_Γ in the normal form
QUESTION:	Is there a locally different data word w over Γ such that $\mathsf{Proj}(w)$ is accepted by $\mathscr{A}$ and $w \models_c \varphi$?

- Then a 2-NEXPTIME algorithm is presented to solve the problem of NONEMPTINESS-LOCALLY-DIFFERENT.

From the above description of the proof outline, it is evident that NONEMPTINESS-PROFILE can be decided in 4-NEXPTIME. By a finer analysis, the complexity can be shown in 3-NEXPTIME.

Since the reduction of the problem of NONEMPTINESS-PROFILE to the problem of NONEMPTINESS-LOCALLY-DIFFERENT completely mimics that for WDA in [12, 5], it is omitted here due to the lack of space.

In the rest of this section, we will focus on the problem of NONEMPTINESS-LOCALLY-DIFFERENT. Before presenting an algorithm to solve the problem, we will state and prove two lemmas.

4.1 Two lemmas

We first introduce some notations.

▶ **Definition 11.** *Let $\varphi = \bigvee\limits_{1 \le i \le m} \varphi_i$ be a QFSP formula in the normal form over the variable set V_Γ, p_0 be the normalization number of φ, and for every $i : 1 \le i \le m$, $\varphi_i = \bigwedge\limits_{\gamma \in \Gamma} \varphi_{i,\gamma}$, where $\varphi_{i,\gamma}$ is either $x_\gamma = c_{i,\gamma}$ or $x_\gamma \ge p_0 \wedge x_\gamma \equiv r_{i,\gamma} \bmod p_0$ for some $c_{i,\gamma}, r_{i,\gamma} : 0 \le c_{i,\gamma}, r_{i,\gamma} < p_0$. Then for each $\gamma \in \Gamma$, define two subsets of $\{1, \dots, m\}$, denoted by $I_E(\varphi, \gamma)$ and $I_M(\varphi, \gamma)$, as follows: For every $i : 1 \le i \le m$,*

- *$i \in I_E(\varphi, \gamma)$ iff $\varphi_{i,\gamma}$ is $x_\gamma = c_{i,\gamma}$ and $c_{i,\gamma} > 0$,*
- *$i \in I_M(\varphi, \gamma)$ iff $\varphi_{i,\gamma}$ is $x_\gamma \ge p_0 \wedge x_\gamma \equiv r_{i,\gamma} \bmod p_0$.*

Note that if $i \notin I_E(\varphi, \gamma) \cup I_M(\varphi, \gamma)$, then it holds that $\varphi_{i,\gamma}$ is $x_\gamma = c_{i,\gamma}$ and $c_{i,\gamma} = 0$.

▶ **Definition 12.** *Let w be a data word over Γ and $\varphi = \bigvee\limits_{1 \le i \le m} \varphi_i$ be a QFSP formula over the variable set V_Γ in the normal form. If $w \models_c \varphi$, then for each data value d occurring in w, there is a unique $i : 1 \le i \le m$ such that $\mathsf{Proj}(w|_X) \models \varphi_i$, where X is the class of w corresponding to d. This unique number i is called* the index of the class condition φ *for d, denoted by* $\mathsf{idx}_\varphi(d)$.

We are ready to state and prove the two lemmas.

▶ **Lemma 13.** *For every QFSP formula $\varphi = \bigvee\limits_{1 \le i \le m} \varphi_i$ over the variable set V_Γ in the normal form, there is an EP formula $\psi = \exists y_1 \dots \exists y_m \psi'$ of polynomial size such that for each word v over Γ, $v \models \psi$ iff there is a data word w such that $\mathsf{Proj}(w) = v$ and $w \models_c \varphi$.*

Proof. Suppose φ is a QFSP formula in the normal form over the variable set V_Γ with the normalization number p_0. Then $\varphi = \bigvee\limits_{i : 1 \le i \le m} \varphi_i$, where φ_i is of the form $\bigwedge\limits_{\gamma \in \Gamma} \varphi_{i,\gamma}$ such that $\varphi_{i,\gamma}$ is equal to $x_\gamma = c_{i,\gamma}$ or $x_\gamma \ge p_0 \wedge x_\gamma \equiv r_{i,\gamma} \bmod p_0$ for some $c_{i,\gamma}, r_{i,\gamma} : 0 \le c_{i,\gamma}, r_{i,\gamma} < p_0$. In addition, those φ_i's are mutually exclusive.

Let $\psi = \exists y_1 \dots \exists y_m \psi'$ such that ψ' is a conjunction of the quantifier-free Presburger formulas ψ'_1, ψ'_2, and ψ'_3, where

1. $\psi'_1 := \bigwedge\limits_{\gamma \in \Gamma} \left(x_\gamma - \left(\sum\limits_{i \in I_E(\varphi,\gamma)} c_{i,\gamma} y_i \right) - \left(\sum\limits_{i \in I_M(\varphi,\gamma)} (p_0 + r_{i,\gamma}) y_i \right) \ge 0 \right)$,
2. $\psi'_2 := \bigwedge\limits_{\gamma \in \Gamma} \left(\left(\bigwedge\limits_{i \in I_M(\varphi,\gamma)} y_i = 0 \right) \to x_\gamma - \left(\sum\limits_{i \in I_E(\varphi,\gamma)} c_{i,\gamma} y_i \right) = 0 \right)$,
3. $\psi'_3 := \bigwedge\limits_{\gamma \in \Gamma} \left(x_\gamma - \left(\sum\limits_{i \in I_E(\varphi,\gamma)} c_{i,\gamma} y_i + \sum\limits_{i \in I_M(\varphi,\gamma)} r_{i,\gamma} y_i \right) \equiv 0 \bmod p_0 \right)$.

Intuitively,

- $y_1, \dots, y_m$ represent the numbers of classes satisfying respectively $\varphi_1, \dots, \varphi_m$,
- the formula ψ'_1 specifies the lower bound of γ-positions for every $\gamma \in \Gamma$, which is the sum of the lower bounds of γ-positions in all classes, more precisely, $c_{i,\gamma}$ for $i \in I_E(\varphi, \gamma)$ or $p_0 + r_{i,\gamma}$ for $i \in I_M(\varphi, \gamma)$,
- the formula ψ'_2 specifies that for each $\gamma \in \Gamma$, if there are no classes in which modular constraints for γ are required (this is specified by the condition $y_i = 0$ for every $i \in I_M(\varphi, \gamma)$), then the number of γ-positions is equal to the sum of $c_{i,\gamma}$ for $i \in I_E(\varphi, \gamma)$,
- the formula ψ'_3 says that for every $\gamma \in \Gamma$, the number of γ-positions, subtracting the lower bound specified in ψ'_1, should be equal to zero modulo p_0.

Let $\overline{y}$ denote the tuple $y_1, \ldots, y_m$ in the following.

"If" part:

Suppose there is a data word w such that $\mathsf{Proj}(w) = v$ and $w \models_c \varphi$, namely, for each class X in w, $\mathsf{Proj}(w|_X) \models \varphi$.

For each $i : 1 \leq i \leq m$, let D_i be the set of data values d occurring in w such that $\mathsf{idx}_\varphi(d) = i$. Note that $(D_i)_{1 \leq i \leq m}$ forms a partition of the set of all the data values occurring in w. In addition, let $k_i = |D_i|$ for each $i : 1 \leq i \leq m$. We also use $\overline{k}$ to denote the tuple $k_1, \ldots, k_m$.

It is sufficient to verify that $v \models \psi'[\overline{y} \leftarrow \overline{k}]$ in order to show $v \models \psi$.

Let's exemplify the argument by demonstrating that $v \models \psi'_2[\overline{y} \leftarrow \overline{k}]$.

Suppose $k_i = 0$ for each $i \in I_M(\varphi, \gamma)$. Then $D_i = \emptyset$ for each $i \in I_M(\varphi, \gamma)$. We want to show that $\#_\gamma(v) = \sum\limits_{i \in I_E(\varphi,\gamma)} c_{i,\gamma} k_i$.

For each data value $d \in D_i$ such that $i \notin I_M(\varphi, \gamma)$,

- if $i \in I_E(\varphi, \gamma)$, i.e. $\varphi_{i,\gamma}$ is equal to $x_\gamma = c_{i,\gamma}$ and $c_{i,\gamma} > 0$, then the letter γ occurs exactly $c_{i,\gamma}$ times in the class of w corresponding to d;
- if $i \notin I_E(\varphi, \gamma) \cup I_M(\varphi, \gamma)$, i.e. $\varphi_{i,\gamma}$ is equal to $x_\gamma = c_{i,\gamma}$ and $c_{i,\gamma} = 0$, then the letter γ does not occur in the class of w corresponding to d.

Since $(D_i)_{1 \leq i \leq m}$ is a partition of the set of all the data values occurring in w, it follows that $\#_\gamma(v) = \sum\limits_{i \in I_E(\varphi,\gamma)} c_{i,\gamma} k_i$. So $v \models \psi'_2[\overline{y} \leftarrow \overline{k}]$.

"Only if" part:

Suppose $v \models \psi$. Then there are numbers $\overline{k} = k_1, \ldots, k_m$ such that $v \models \psi'[\overline{y} \leftarrow \overline{k}]$.

Let $K = k_1 + \cdots + k_m$. Define a function $\xi : \{1, \ldots, K\} \to \{1, \ldots, m\}$ such that $|\xi^{-1}(i)| = k_i$ for each $i : 1 \leq i \leq m$.

In the following, we assign the data values from $\{1, \ldots, K\}$ to the positions in v to get a data word w such that $w \models_c \varphi$, namely, for each class X of w, $\mathsf{Proj}(w|_X) \models \varphi$.

From the fact that $v \models \psi'_1[\overline{y} \leftarrow \overline{k}]$, we know that for each $\gamma \in \Gamma$,

$$\#_\gamma(v) \geq \sum_{i \in I_E(\varphi,\gamma)} c_{i,\gamma} k_i \ + \sum_{i \in I_M(\varphi,\gamma)} (p_0 + r_{i,\gamma}) k_i.$$

We assign the data values in $\{1, \ldots, K\}$ to the positions in v through the following two-step procedure.

Step 1 For every $\gamma \in \Gamma$ and every $i : 1 \leq i \leq m$, assign the data values in $\xi^{-1}(i)$ to the γ-positions in v such that each data value in $\xi^{-1}(i)$ is assigned to exactly $(c_{i,\gamma})$ γ-positions if $i \in I_E(\varphi, \gamma)$, and is assigned to exactly $(p_0 + r_{i,\gamma})$ γ-positions if $i \in I_M(\varphi, \gamma)$.

Step 2 For every $\gamma \in \Gamma$ such that there is $i \in I_M(\varphi, \gamma)$ satisfying that $k_i > 0$, select such an index i and a data value from $\xi^{-1}(i)$, denoted by d_γ, and assign d_γ to all the γ-positions which have not been assigned data values after Step 1.

Now all the positions of v have been assigned data values from $\{1, \ldots, K\}$, let w be the resulting data word.

For every $\gamma \in \Gamma$, if there are still γ-positions that have not been assigned data values after Step 1, then $\#_\gamma(v) > \sum\limits_{i \in I_E(\varphi,\gamma)} c_{i,\gamma} k_i \ + \sum\limits_{i \in I_M(\varphi,\gamma)} (p_0 + r_{i,\gamma}) k_i \geq \sum\limits_{i \in I_E(\varphi,\gamma)} c_{i,\gamma} k_i$. From the fact that $v \models \psi'_2[\overline{y} \leftarrow \overline{k}]$, it follows that there is $i \in I_M(\varphi, \gamma)$ such that $k_i > 0$. So a data value d_γ can be selected and assigned to all the pending γ-positions in Step 2.

It remains to show that $w \models_c \varphi$. It is sufficient to prove that for every $i : 1 \leq i \leq m$ and every data value $d \in \xi^{-1}(i)$, $\mathsf{Proj}(w|_X) \models \varphi_i$, where X is the class of w corresponding to d. Because $\mathsf{Proj}(w|_X) = v|_X$ and $\varphi_i = \bigwedge\limits_{\gamma \in \Gamma} \varphi_{i,\gamma}$, it is equivalent to show that for every

$i : 1 \leq i \leq m$, $d \in \xi^{-1}(i)$, and $\gamma \in \Gamma$, we have $v|_X \models \varphi_{i,\gamma}$, where X is the class of w corresponding to d.

Suppose $i : 1 \leq i \leq m$, $d \in \xi^{-1}(i)$, and $\gamma \in \Gamma$. Let X be the class of w corresponding to d. In the following, we show that $v|_X \models \varphi_{i,\gamma}$.

From the data value assignment procedure, we know that there are still $\big(\#_\gamma(v) - \sum\limits_{i \in I_E(\varphi,\gamma)} c_{i,\gamma} k_i - \sum\limits_{i \in I_M(\varphi,\gamma)} (p_0 + r_{i,\gamma}) k_i\big)$ γ-positions which have not been assigned data values after Step 1. Because $v \models \psi'_3[\overline{y} \leftarrow \overline{k}]$, it follows that $\#_\gamma(v) - \sum\limits_{i \in I_E(\varphi,\gamma)} c_{i,\gamma} k_i - \sum\limits_{i \in I_M(\varphi,\gamma)} (p_0 + r_{i,\gamma}) k_i \equiv 0 \bmod p_0$. So there is $t_\gamma \in \mathbb{N}$ such that $\#_\gamma(v) - \sum\limits_{i \in I_E(\varphi,\gamma)} c_{i,\gamma} k_i - \sum\limits_{i \in I_M(\varphi,\gamma)} (p_0 + r_{i,\gamma}) k_i = t_\gamma p_0$.

We distinguish between the following three cases.

Case $i \in I_E(\varphi,\gamma)$. Then $\varphi_{i,\gamma}$ is $x_\gamma = c_{i,\gamma}$ and $c_{i,\gamma} > 0$. From the data value assignment procedure, we know that each data value in $\xi^{-1}(i)$, including d, has been assigned to exactly $(c_{i,\gamma})$ γ-positions. This implies that $\#_\gamma(v|_X) = c_{i,\gamma}$. So $v|_X \models \varphi_{i,\gamma}$.

Case $i \in I_M(\varphi,\gamma)$. Then $\varphi_{i,\gamma}$ is $x_\gamma \geq p_0 \wedge x_\gamma \equiv r_{i,\gamma} \bmod p_0$. From the data value assignment procedure, we know that the data value d is assigned to $(p_0 + r_{i,\gamma})$ γ-positions if $d \neq d_\gamma$, and assigned to $(p_0 + r_{i,\gamma} + t_\gamma p_0)$ γ-positions otherwise. Therefore, $\#_\gamma(v|_X) = p_0 + r_{i,\gamma}$ or $p_0 + r_{i,\gamma} + t_\gamma p_0$. It follows that $v|_X \models \varphi_{i,\gamma}$.

Case $i \notin I_E(\varphi,\gamma) \cup I_M(\varphi,\gamma)$. Then $\varphi_{i,\gamma}$ is $x_\gamma = c_{i,\gamma}$ and $c_{i,\gamma} = 0$. From the data value assignment procedure, we know that each data value in $\xi^{-1}(i)$, including d, has not been assigned to any γ-position in v. Therefore, $\#_\gamma(v|_X) = 0$ and $v|_X \models \varphi_{i,\gamma}$. ◀

▶ **Definition 14.** *Let $\varphi = \bigvee\limits_{1 \leq i \leq m} \varphi_i$ be a QFSP formula in the normal form over the variable set V_Γ with the normalization number p_0 such that for each $i : 1 \leq i \leq m$, $\varphi_i = \bigwedge\limits_{\gamma \in \Gamma} \varphi_{i,\gamma}$, where $\varphi_{i,\gamma}$ is $x_\gamma = c_{i,\gamma}$ or $x_\gamma \geq p_0 \wedge x_\gamma \equiv r_{i,\gamma} \bmod p_0$ for $0 \leq c_{i,\gamma}, r_{i,\gamma} < p_0$. Moreover, for each $i : 1 \leq i \leq m$, let*

$$h_i = \sum_{\gamma : i \in I_E(\varphi,\gamma)} c_{i,\gamma} + \sum_{\gamma : i \in I_M(\varphi,\gamma)} (p_0 + r_{i,\gamma}).$$

Let w be a data word over the alphabet Γ, then w is said to satisfy the class condition φ with "many" data values if $w \models_c \varphi$ and for each $i : 1 \leq i \leq m$, either $k_i = 0$ or $k_i \geq \max(2p_0 + 1,\ 2h_i + 3)$, where k_i is the number of data values d occurring in w such that $\mathsf{idx}_\varphi(d) = i$.

Let $v \in \Gamma^*$ and $\psi = \exists y_1 \ldots \exists y_m \psi'$ be the EP formula obtained from φ as in Lemma 13. Then v is said to satisfy ψ with "large" numbers if there are a tuple of numbers $\overline{k} = k_1, \ldots, k_m$ such that $v \models \psi'[\overline{y} \leftarrow \overline{k}]$ and for each $i : 1 \leq i \leq m$, either $k_i = 0$ or $k_i \geq \max(2p_0 + 1,\ 2h_i + 3)$.

▶ **Lemma 15.** *Let $\varphi = \bigvee\limits_{1 \leq i \leq m} \bigwedge\limits_{\gamma \in \Gamma} \varphi_{i,\gamma}$ be a QFSP formula in the normal form with the normalization number p_0. Moreover, let $\psi = \exists y_1 \ldots \exists y_m \psi'$ be the EP formula obtained from φ as stated in Lemma 13. Then for any $v \in \Gamma^*$, $v \models \psi$ with large numbers iff there is a locally different data word w such that $\mathsf{Proj}(w) = v$ and $w \models_c \varphi$ with many data values.*

Proof. "If" part: Obvious.

"Only if" part:

Suppose v satisfies ψ with large numbers, i.e. there are numbers $\overline{k} = k_1, \ldots, k_m$ such that $v \models \psi'[\overline{y} \leftarrow \overline{k}]$ and for each $i : 1 \leq i \leq m$, either $k_i = 0$ or $k_i \geq \max(2p_0 + 1, 2h_i + 3)$.

Let $K = k_1 + \cdots + k_m$. Define a function $\xi : \{1, \ldots, K\} \to \{1, \ldots, m\}$ such that $|\xi^{-1}(i)| = k_i$ for each $i : 1 \leq i \leq m$.

As in the proof of Lemma 13, we assign data values in $\{1, \ldots, K\}$ to the positions of v to get a desired data word w. The assignment procedure is divided into two steps, Step 1 and 2.

Step 1:

The same as Step 1 of the data value assignment procedure in the proof of the "Only if" part of Lemma 13.

After Step 1, we get a partial data word where some positions still have no data values. Let's assign a special data value, say $\sharp$, to all those positions without data values, then we get a data word $w_1 = \binom{\gamma_1}{d_1}\binom{\gamma_2}{d_2} \ldots \binom{\gamma_n}{d_n}$.

In w_1, there may exist positions j such that $d_j = d_{j+1}$ and $d_j, d_{j+1} \neq \sharp$. Let's call these positions as *conflicting* positions of w_1.

Claim. The data word w_1 can be turned into a data word w_1'' such that w_1'' contains no conflicting positions, w_1 and w_1'' have the same set of data values (including $\sharp$), and for each $\gamma \in \Gamma$ and each class X of w_1, $\#_\gamma(w_1|_X) = \#_\gamma(w_1''|_X)$.

Proof of the claim.

Let j be a conflicting position of w_1, $a = \gamma_j$, and $i : 1 \leq i \leq m$ such that $d_j \in \xi^{-1}(i)$. From Step 1, we know that d_j occurs exactly $h_i = \sum\limits_{\gamma : i \in I_E(\varphi,\gamma)} c_{i,\gamma} + \sum\limits_{\gamma : i \in I_M(\gamma)} (p_0 + r_{i,\gamma})$ times in w_1. It follows that there are at most $2h_i$ positions adjacent to a position with the data value d_j. On the other hand, we have that $k_i \geq \max(2p_0 + 1, 2h_i + 3)$ and for each data value in $d \in \xi^{-1}(i)$, there is at least one occurrence of a with the data value d. It follows that there are (at least) *three* positions j_1', j_2', j_3' such that $d_{j_1'}, d_{j_2'}, d_{j_3'} \in \xi^{-1}(i)$, $d_{j_1'}, d_{j_2'}, d_{j_3'}$ are pairwise distinct, $\gamma_{j_1'} = \gamma_{j_2'} = \gamma_{j_3'} = a$, and $d_{j_1'-1}, d_{j_2'-1}, d_{j_3'-1}, d_{j_1'+1}, d_{j_2'+1}, d_{j_3'+1} \neq d_j$. From this, we deduce that there is a position j' such that $\gamma_{j'} = a$, $d_{j'} \in \xi^{-1}(i)$, $d_{j'} \neq d_{j-1}, d_j$, and $d_j \neq d_{j'-1}, d_{j'+1}$. Because $d_{j'} \neq d_{j-1}, d_{j+1}$ ($d_{j+1} = d_j$ since j is conflicting) and $d_j \neq d_{j'-1}, d_{j'+1}$, we can swap the data value d_j in the position j and the data value $d_{j'}$ in the position j' to make the two positions j and j' non-conflicting. Let w_1' be the data word after the swapping. It follows that w_1' has less conflicting positions than w_1.

Continue like this, we finally get a data word w_1'' without conflicting positions. ◀

Now we return to the proof of the lemma.

From the claim, we know that a data word w_1'' containing no conflicting positions can be obtained from w_1. But w_1'' may still contain the special data value $\sharp$. If this is the case, then from the description of Step 1, we know that there exists at least one $\gamma \in \Gamma$ such that $\#_\gamma(v) > \sum\limits_{i \in I_E(\varphi,\gamma)} c_{i,\gamma} k_i + \sum\limits_{i \in I_M(\gamma)} (p_0 + r_{i,\gamma}) k_i$.

Let $\gamma \in \Gamma$ such that $\#_\gamma(v) > \sum\limits_{i \in I_E(\varphi,\gamma)} c_{i,\gamma} k_i + \sum\limits_{i \in I_M(\gamma)} (p_0 + r_{i,\gamma}) k_i$.

From $v \models \psi_3'[\overline{y} \leftarrow \overline{k}]$, it follows that $\#_\gamma(v) - \sum\limits_{i \in I_E(\varphi,\gamma)} c_{i,\gamma} k_i - \sum\limits_{i \in I_M(\gamma)} (p_0 + r_{i,\gamma}) k_i \equiv 0 \bmod p_0$. So there is $t_\gamma \geq 1$ such that $\#_\gamma(v) - \sum\limits_{i \in I_E(\varphi,\gamma)} c_{i,\gamma} k_i - \sum\limits_{i \in I_M(\gamma)} (p_0 + r_{i,\gamma}) k_i = t_\gamma p_0$. Therefore, there are $(t_\gamma p_0)$ γ-positions in w_1 of the data value $\sharp$. Because w_1 and w_1'' have the same set of positions of the data value $\sharp$, it follows that there are also $(t_\gamma p_0)$ γ-positions in w_1'' of the data value $\sharp$. Let $j_{\gamma,1} < \cdots < j_{\gamma,t_\gamma p_0}$ be a list of all such γ-positions in w_1''.

On the other hand, because $v \models \psi_2'[\overline{y} \leftarrow \overline{k}]$ and $\#_\gamma(v) > \sum\limits_{i \in I_E(\varphi,\gamma)} c_{i,\gamma} k_i$, it follows that there is $i \in I_M(\varphi, \gamma)$ such that $k_i > 0$. Let i_γ be such an index i. Then from the assumption that v satisfies ψ with large numbers, we know that $k_{i_\gamma} \geq \max(2p_0 + 1, 2h_{i_\gamma} + 3)$.

Step 2:

For each $\gamma \in \Gamma$ such that $\#_\gamma(v) > \sum_{i \in I_E(\varphi,\gamma)} c_{i,\gamma} k_i + \sum_{i \in I_M(\gamma)} (p_0 + r_{i,\gamma}) k_i$, assign the data values from $\{1, \ldots, K\}$ to the γ-positions with the data value $\sharp$ in w_1'' as follows. We distinguish between the following two cases.

- *Case $t_\gamma \geq 2$.*
  Initially set $s := 1$. Repeat following procedure until $s > t_\gamma$.
  Let $J = \{j_{\gamma,s}, j_{\gamma,t_\gamma+s}, \ldots, j_{\gamma,t_\gamma(p_0-1)+s}\}$ and J' be the set of all positions adjacent to a position in J. In addition, let D be the set of data values (except $\sharp$) occurring in the positions belonging to J'. Because $|J| = p_0$, we have $|D| \leq 2p_0$. On the other hand, $k_{i_\gamma} \geq 2p_0 + 1$, it follows that there is $d \in \xi^{-1}(i_\gamma) \setminus D$.
  Assign the data value d to every position in J. Then we still get a non-conflicting data word, since all the positions in J are not adjacent to each other.
  Set $s := s + 1$.
- *Case $t_\gamma = 1$.*
  Let $J = \{j_{\gamma,1}, j_{\gamma,2}, \ldots, j_{\gamma,p_0}\}$ and J' be the set of all positions adjacent to a position in J. In addition, let D be the set of data values (except $\sharp$) occurring in the positions belonging to J'. Because $|J| = p_0$, we have $|D| \leq 2p_0$. On the other hand, $k_{i_\gamma} \geq 2p_0 + 1$, it follows that there is $d \in \xi^{-1}(i_\gamma) \setminus D$.
  Because $i_\gamma \in I_M(\varphi, \gamma)$, each data value in $\xi^{-1}(i_\gamma)$ has been assigned to exactly $(p_0 + r_{i_\gamma,\gamma})$ γ-positions in Step 1. During Step 1, we can do the assignments in a way so that all the positions in J, i.e. the p_0 γ-positions without data value, are not adjacent to each other. Therefore, we can assign the data value d to every position in J and still get a non-conflicting data word.

Let w be the resulting data word after the two steps of data value assignments. Then w is locally different. Similar to the proof of the "Only if" part of Lemma 13, we can show that for each $i : 1 \leq i \leq m$ and each data value $d \in \xi^{-1}(i)$, $\mathsf{Proj}(w|_X) \models_c \varphi_i$, where X is the class of w corresponding to d. From this, it follows that for each $i : 1 \leq i \leq m$, the number of data values in w such that $i \in \mathsf{idx}_\varphi(d)$ is equal to k_i. Since for each $i : 1 \leq i \leq m$, either $k_i = 0$ or $k_i \geq \max(2p_0 + 1, 2h_i + 3)$, we conclude that $w \models_c \varphi$ with many data values. ◀

4.2 Algorithm for NONEMPTINESS-LOCALLY-DIFFERENT

We first give an algorithm for the following problem.

PROBLEM:	NONEMPTINESS-LOCALLY-DIFFERENT-MANY
INPUT:	A finite automaton $\mathscr{A} = (Q, \Gamma, \delta, q_0, F)$ and a QFSP formula φ over V_Γ in the normal form
QUESTION:	is there a locally different data word w over the alphabet Γ such that $\mathsf{Proj}(w)$ is accepted by $\mathscr{A}$ and $w \models_c \varphi$ with many data values?

From Lemma 15, it follows that NONEMPTINESS-LOCALLY-DIFFERENT-MANY can be solved by the following algorithm.

Suppose the normalization number of φ is p_0 and $\varphi = \bigvee_{i:1\leq i\leq m} \bigwedge_{\gamma\in\Gamma} \varphi_{i,\gamma}$ such that each $\varphi_{i,\gamma}$ is either $x_\gamma = c_{i,\gamma}$ or $x_\gamma \geq p_0 \wedge x_\gamma \equiv r_{i,\gamma} \bmod p_0$ for some $c_{i,\gamma}, r_{i,\gamma} : 0 \leq c_{i,\gamma}, r_{i,\gamma} < p_0$. Let $\psi = \exists y_1 \ldots \exists y_m \psi'$ be the existential Presburger formula obtained *from φ as stated in Lemma 13. For every $i : 1 \leq i \leq m$, let $h_i = \sum\limits_{\gamma: i \in I_E(\varphi,\gamma)} c_{i,\gamma} + \sum\limits_{\gamma: i \in I_M(\varphi,\gamma)} (p_0 + r_{i,\gamma})$.*

1. *Construct the following EP formula ψ_g,*

$$\psi_g := \exists y_1 \ldots \exists y_m \left(\psi' \wedge \bigwedge_{1 \leq i \leq m} (y_i = 0 \vee (y_i \geq 2p_0 + 1 \wedge y_i \geq 2h_i + 3)) \right).$$

2. *Decide the nonemptiness of the Presburger automaton $(\mathscr{A}, \psi_g)$.*

Now we consider the problem of NONEMPTINESS-LOCALLY-DIFFERENT.

For a data word $w \in (\Gamma \times \mathbb{D})^*$, if $w \models_c \varphi$, then for each $i : 1 \leq i \leq m$, let k_i be the number of data values d occurring in w such that $\mathsf{idx}_\varphi(d) = i$. For each $i : 1 \leq i \leq m$ such that $k_i < \max(2p_0 + 1, 2h_i + 3)$, if we take the k_i data values d such that $\mathsf{idx}_\varphi(d) = i$ as constants, then the problem of NONEMPTINESS-LOCALLY-DIFFERENT can be solved similar to the problem of NONEMPTINESS-LOCALLY-DIFFERENT-MANY. More specifically, the algorithm goes as follows.

1. *Guess a set $J \subseteq \{1, \ldots, m\}$ and sets of constants D_j's.*
   a) *Guess a set $J \subseteq \{1, \ldots, m\}$.*
   b) *For each $j \in J$, guess an integer $s_j < \max(2p_0 + 1, 2h_i + 3)$.*
   c) *For each $j \in J$, fix a set $D_j = \{\alpha^j_1, \ldots, \alpha^j_{s_j}\}$ of constants such that D_j's are mutually disjoint and $D_j \cap \mathbb{D} = \emptyset$. Let $D_J = \cup_{j \in J} D_j$.*
2. *Construct an automaton $\mathscr{A}'$ over the alphabet $\Gamma \cup \Gamma \times D_J$ from $(\mathscr{A}, \varphi)$ such that $\mathscr{A}'$ accepts a word $v = \lambda_1 \ldots \lambda_n \in (\Gamma \cup \Gamma \times D_J)^*$ iff the following conditions hold.*
   - *A symbol (γ, d) appears in v iff there exists $j \in J$ such that $j \in I_E(\varphi, \gamma) \cup I_M(\varphi, \gamma)$ and $d \in D_j$.*
   - *Let $u = \gamma_1 \ldots \gamma_n \in \Gamma^*$ such that*

$$\gamma_i = \begin{cases} \lambda_i & \text{if } \lambda_i \in \Gamma, \\ \gamma & \text{if } \lambda_i = (\gamma, d) \in \Gamma \times D_J. \end{cases}$$

     Then u is accepted by $\mathscr{A}$.
   - *For any $i : 1 \leq i < n$, if $\lambda_i = (\gamma, d)$ and $\lambda_{i+1} = (\gamma', d')$, then $d \neq d'$.*
   - *For any $j \in J$ and any $\gamma \in \Gamma$, the following holds: If $j \in I_E(\varphi, \gamma)$, then for each $d \in D_j$, the letter (γ, d) occurs exactly $c_{j,\gamma}$ times in v. If $j \in I_M(\varphi, \gamma)$, then for each $d \in D_j$, the number of occurrences of the letter (γ, d) is at least p_0 and equal to $r_{i,\gamma}$ modulo p_0.*
3. *Construct the following EP formula $\psi_{g,J}$,*

$$\psi_{g,J} = \exists y_1 \ldots \exists y_m \left(\psi' \wedge \bigwedge_{i \in J} y_i = 0 \wedge \bigwedge_{i \notin J} (y_i \geq 2p_0 + 1 \wedge y_i \geq 2h_i + 3) \right).$$

   Note that $\psi_{g,J}$ is an EP formula with free variables from V_Γ, and contains no variables $x_{(\gamma,d)}$ with $(\gamma, d) \in \Gamma \times D_J$.
4. *Decide the nonemptiness of the Presburger automaton $(\mathscr{A}', \psi_{g,J})$.*

The proof of the correctness of the above algorithm for NONEMPTINESS-LOCALLY-DIFFERENT follows the same line as the proof for SAT-LOCALLY-DIFFERENT in [5].

5 Commutative Büchi data automata

In this section, we consider data automata with commutative class conditions over data ω-words.

Let $\mathbb{N}_\omega = \mathbb{N} \cup \{\omega\}$ with the linear order ($<$) and the addition ($+$) operation of $\mathbb{N}$ extended in a natural way, i.e. $n < \omega$ for any $n \in \mathbb{N}$, and $\omega + n = \omega$ for any $n \in \mathbb{N}_\omega$.

The definition of the Parikh images of finite words can be easily extended to ω-words: Given an ω-word v over an alphabet $\Gamma = \{\gamma_1, \dots, \gamma_l\}$, $\mathsf{Parikh}(v) = (\#_{\gamma_1}(v), \dots, \#_{\gamma_l}(v))$, where for each $i : 1 \le i \le l$, $\#_{\gamma_i}(v)$ is still the number of occurrences of γ_i in v, in particular, if γ_i occurs infinitely many times in v, then $\#_{\gamma_i}(v) = \omega$.

Similar to QFSP formulas, we define ω-QFSP formulas over a variable set X as follows.

The syntax of ω-QFSP formulas is the same as QFSP formulas, except that the atomic formulas can also be of the form $x = \omega$ (where $x \in X$).

The ω-QFSP formulas are interpreted on $\mathbb{N}_\omega$: Let $\pi : X \to \mathbb{N}_\omega$, then the atomic ω-QFSP formulas are interpreted as follows,

- $\pi \models x_1 + \cdots + x_n$ op c if $\pi(x_1) + \cdots + \pi(x_n)$ op c, where op $\in \{\le, \ge, =\}$,
- $\pi \models x_1 + \cdots + x_n \equiv r \bmod p$ if $\pi(x_1) + \cdots + \pi(x_n) < \omega$ and $\pi(x_1) + \cdots + \pi(x_n) \equiv r \bmod p$,
- $\pi \models x = \omega$ if $\pi(x) = \omega$.

In addition, the Boolean operators are interpreted in a standard way.

Similar to Proposition 3, there is a normal form for ω-QFSP formulas.

▶ Proposition 16. *Let $\varphi(x_1, \dots, x_k)$ be a ω-QFSP formula. Then there exists an exponential-time algorithm to transform φ into a ω-QFSP formula $\bigvee\limits_{i:1 \le i \le m} \varphi_i$ of size $2^{O(k|\varphi|)}$ such that there is $p_0 : 2 \le p_0 \le 2^{|\varphi|}$ satisfying that*

- *each φ_i is of the form $\bigwedge\limits_{1 \le j \le k} \varphi_{i,j}$;*
- *for each $j : 1 \le j \le k$, $\varphi_{i,j}$ is equal to $x_j = c_{i,j}$ or $x_j \ge p_0 \wedge x_j \equiv r_{i,j} \bmod p_0$, or $x_j = \omega$ for $c_{i,j}, r_{ij} : 0 \le c_{i,j}, r_{i,j} < p_0$;*
- *in addition, those φ_i's are mutually exclusive.*

Let w be a data ω-word over an alphabet Γ and φ be a ω-QFSP formula over the variable set V_Γ, then the definition of $w \models_c \varphi$, i.e. w satisfies the class condition φ, is a natural extension of that for data words.

A *commutative Büchi data automaton* (CBDA) is a binary tuple $(\mathscr{A}, \varphi)$, where $\mathscr{A} = (Q, \Sigma \times \{\bot, \top\}, \Gamma, \delta, q_0, F)$ is a Büchi letter-to-letter transducer and φ is a ω-QFSP formula over the variable set V_Γ.

A CBDA $(\mathscr{A}, \varphi)$ accepts a data ω-word $w = \binom{\sigma_1}{d_1}\binom{\sigma_2}{d_2}\dots$ if there is an accepting run of $\mathscr{A}$ over $\mathsf{Profile}(w)$ which produces an ω-word $\gamma_1\gamma_2\dots$ such that the data ω-word $w' = \binom{\gamma_1}{d_1}\binom{\gamma_2}{d_2}\dots\dots$ satisfies that $w' \models_c \varphi$.

Similar to the logic $EMSO^2_\#(+1, \sim, \Sigma)$ in Section 3, we define the logic $E_\infty MSO^2_\#(+1, \sim, \Sigma)$ as follows: It includes all the formulas $\exists_\infty R_1 \dots \exists_\infty R_k \exists S_1 \dots \exists S_l (\varphi \wedge \forall x \psi)$, where $\varphi \in FO^2(+1, \sim, \Sigma, R_1, \dots, R_k, S_1, \dots, S_l)$ and ψ is the same as the ψ in $EMSO^2_\#(+1, \sim, \Sigma)$ formulas, except that the atomic formulas in ψ can be also of the form $\#_{x \sim y \wedge \tau(y)}(y) = \omega$.

The semantics of $E_\infty MSO^2(+1, \sim, \Sigma)$ formulas are defined similar to $EMSO^2_\#(+1, \sim, \Sigma)$ formulas, except that the unary relation symbols $R_1, \dots, R_k$ are restricted to bind to infinite sets and $\#_{x \sim y \wedge \tau(y)}(y) = \omega$ are interpreted as the fact that the symbol τ appears infinitely many times in the class that contains the position x.

Similar to CDA, we also have the following logical characterization of CBDA.

▶ **Theorem 17.** *$E_\infty MSO^2_\#(+1, \sim, \Sigma)$ and CBDA are expressively equivalent.*

The proof of Theorem 17 is similar to that for WBDA in [12].

▶ **Theorem 18.** *The nonemptiness of CBDA can be decided in 4-NEXPTIME.*

The proof of Theorem 18 is by a nondeterminstic exponential time reduction to the nonemptiness of CDA on data words.

Acknowledgements. The author thanks Anca Muscholl for the comments and suggestions on this work. The author also thanks anonymous referees for their valuable comments and suggestions to improve the quality of this paper.

References

1 M. Bojańczyk, C. David, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data words. *ACM Trans. Comput. Logic*, 12(4):27:1–27:26, 2011.

2 M. Bojanczyk and S. Lasota. An extension of data automata that captures xpath. *Logic. Method. in Comput. Sci.*, 8(1), 2012.

3 M. Bojańczyk, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data trees and XML reasoning. *J. ACM*, 56(3):1–48, 2009.

4 M Chrobak. Finite automata and unary languages. *Theor. Comput. Sci.*, 47(2):149–158, 1986.

5 C. David, L. Libkin, and T. Tan. On the satisfiability of two-variable logic over data words. In *LPAR'10*, pages 248–262, 2010.

6 S. Demri and R. Lazić. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Logic*, 10(3):16:1–16:30, 2009.

7 A. Ehrenfeucht and G. Rozenberg. Commutative linear languages. Technical Report CU-CS-209-81, Department of Computer Science, University of Colorado, 1981.

8 D. Figueira. Alternating register automata on finite data words and trees. *Logic. Method. in Comput. Sci.*, 8(1), 2012.

9 D. Figueira. Satisfiability of downward XPath with data equality tests. In *PODS*, pages 197–206, 2009.

10 A. C. Gómez and G. I. Alvarez. Learning commutative regular languages. In *ICGI*, pages 71–83, 2008.

11 M. Kaminski and N. Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134(2):329–363, November 1994.

12 A. Kara, T. Schwentick, and T. Tan. Feasible automata for two-variable logic with successor on data words. In *LATA*, pages 351–362, 2012. A long version can be found at http://arxiv.org/abs/1110.1221.

13 A. Manuel and R. Ramanujam. Class counting automata on datawords. *Int. J. Found. Comput. Sci.*, 22(4):863–882, 2011.

14 F. Neven, T. Schwentick, and V. Vianu. Finite state machines for strings over infinite alphabets. *ACM Trans. Comput. Logic*, 5(3):403–435, 2004.

15 J. E. Pin. *Varieties of formal languages.* Plenum Publishers, 1986.

16 L. Segoufin. Automata and logics for words and trees over an infinite alphabet. In *CSL, LNCS 4207*, pages 41–57, 2006.

17 H. Seidl, T. Schwentick, and A. Muscholl. Counting in trees. In *Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas]*, pages 575–612, 2008.

18 A. W. To. Unary finite automata vs. arithmetic progressions. *Inf. Process. Lett.*, 109(17):1010–1014, 2009.

19 Z. Wu. A decidable extension of data automata. In *GandALF*, pages 116–130, 2011.