Feasible Automata for Two-Variable Logic with Successor on Data Words[⋆]

Ahmet Kara[1], Thomas Schwentick[1], and Tony Tan[2]

[1] Technical University of Dortmund
[2] University of Edinburgh

Abstract. We introduce an automata model for data words, that is words that carry at each position a symbol from a finite alphabet and a value from an unbounded data domain. The model is (semantically) a restriction of data automata, introduced by Bojanczyk, et. al. in 2006, therefore it is called *weak data automata*. It is strictly less expressive than data automata and the expressive power is incomparable with register automata. The expressive power of weak data automata corresponds exactly to existential monadic second order logic with successor $+1$ and data value equality $\sim$, $\mathsf{EMSO}^2(+1,\sim)$. It follows from previous work, David, et. al. in 2010, that the nonemptiness problem for weak data automata can be decided in 2-$\mathsf{NEXPTIME}$. Furthermore, we study weak Büchi automata on data ω-strings. They can be characterized by the extension of $\mathsf{EMSO}^2(+1,\sim)$ with existential quantifiers for infinite sets. Finally, the same complexity bound for its nonemptiness problem is established by a nondeterministic polynomial time reduction to the nonemptiness problem of weak data automata.

1 Introduction

Motivated by challenges in XML reasoning and infinite-state Model Checking, an extension of strings and finitely labelled trees by data values has been investigated in recent years. In classical automata theory, a string is a sequence of positions that carry a symbol from some finite alphabet. In a nutshell, *data strings* generalize strings, in that every position additionally carries a data value from some infinite domain. In the same way, *data trees* generalize (finitely) labelled trees. In XML Theory, data trees model XML documents. Here, the data values can be used to represent attribute values or text content. Both, cannot be adequately modelled by a finite alphabet. In a Model Checking[3] scenario, the data values can be used, e.g., to represent process id's or other data.

Early investigations in this area usually considered strings over an "infinite alphabet", that is, each position only have a value, but no finite-alphabet symbol [2, 20, 7, 14, 15, 18]. Many of the automata models and logics that have been studied for data strings and trees

[⋆] The first and the second authors acknowledge the financial support by the German DFG under grant SCHW 678/4-1, and the third author Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under the FET-Open grant agreement FOX, number FP7-ICT-233599.

[3] In the Model Checking setting, a position might carry a finite set of propositional variables, instead of a symbol.

lack the usual nice decidability properties of automata over finite alphabets, unless strong restrictions are imposed [10, 4, 3, 1].

A result that is particularly interesting for our investigations is the decidability of the satisfiability problem for two-variable logic over data strings [4]. Here, as usual, the logical quantifiers range over the positions of the data string and it can be checked whether a position x carries a symbol a (written: $a(x)$), whether it is to the left of a position y ($x + 1 = y$), whether x is somewhere to the left of y ($x < y$) and whether x and y carry the same data value ($x \sim y$). The logic is denoted by $\mathsf{FO}^2(+1, <, \sim)$. The result was shown with the help of a newly introduced automata model for data words, *data automata* (DA). It turned out, that the expressive power of these automata can be actually characterized by the extension of $\mathsf{FO}^2(+1, <, \sim)$ with existential quantification over sets (of positions) and an additional predicate that holds for x and y if y is the next position from x with the same data value.

However, the complexity of the decision procedure for $\mathsf{FO}^2(+1, <, \sim)$ is very high. The problem is equivalent to the Reachability problem for Petri nets [12], a notoriously hard problem whose complexity has not been resolved exactly. Thus, it has been investigated how the complexity can be reduced, by dropping one of the predicates $x < y$ or $x+1 = y$. In the latter case (that is, for $\mathsf{FO}^2(<, \sim)$) the complexity decreases to NEXPTIME [4]. In the former case ($\mathsf{FO}^2(+1, \sim)$) the complexity also becomes elementary. In [3] a 3-NEXPTIME bound was shown for the case of data trees and this bound clearly carries over to data strings. A more direct proof with a 4-NEXPTIME bound was given in [8] and a 2-NEXPTIME bound was obtained in [19].

The high complexity of the satisfiability of $\mathsf{FO}^2(+1, <, \sim)$ in [4] results from the high complexity of the nonemptiness problem for data automata. One of the starting questions for this paper was:

(1) Is there a natural restriction of data automata with (i) a better complexity and (ii) a correspondence to $\mathsf{FO}^2(+1, \sim)$?

We show that such a restriction indeed exists. Data automata consist of two automata $\mathcal{A}$ and $\mathcal{B}$. $\mathcal{A}$ is a non-deterministic letter-to-letter transducer that constructs, given the finite alphabet part of the input data string[4] u, a new data string w (where, for each position, the data value in w is the same as in u). The second automaton $\mathcal{B}$ can then check properties of the subsequences of w that carry the same data value. We define *weak data automata (WDA)* which also use a non-deterministic letter-to-letter transducer but can only test some simple constraints of the subsequences in the second part. These constraints are (unary) key, inclusion and denial constraints and they are evaluated for each class separately (there are no inter-class constraints).

It turns out that WDA are expressively weaker than data automata, incomparable with register automata [14, 1] and that their expressiveness can be precisely characterized by the extension of $\mathsf{FO}^2(+1, \sim)$ by existential set quantification, that is, $\mathsf{EMSO}^2(+1, \sim)$. As the property that we use to separate the expressive power of WDA and DA can be defined in $\mathsf{EMSO}^2(+1, <, \sim)$ we get that $\mathsf{EMSO}^2(+1, \sim) \not\equiv \mathsf{EMSO}^2(+1, <, \sim)$ as opposed to the classical setting (without data values) where $\mathsf{EMSO}^2(+1) \equiv \mathsf{EMSO}^2(+1, <)$. Indeed, one of the benefits of the logical characterization is that it gives an easy means to show

[4] The transducer also sees whether a position has the same data value as the next one.

non-expressibility for $\mathsf{EMSO}^2(+1,\sim)$ (and $\mathsf{FO}^2(+1,\sim)$). From results in [8] it immediately follows that the nonemptiness problem for WDA can be solved in 2-NEXPTIME.

As mentioned above, one motivation to study data strings comes from Model Checking. In that context, systems are usually considered to run forever and to produce infinite traces. Thus, data ω-words need to be considered as well, and this was actually one of the main motivations of this research. In particular we address the following questions.

(2) Do the complexity results of [8] carry over to data ω-strings?
(3) Can the expressibility results and logical characterizations of the first part of the paper also be established for data ω-strings?

It is straightforward to adapt weak data automata for data ω-strings. The transducer can simply be equipped with a Büchi acceptance mechanism. We refer to the resulting model as *weak Büchi data automata (WBDA)*. It turns out that the answer to both questions, (2) and (3), is affirmative. For (3), this is not hard to prove. The separation of WDA from DA also separates WBDA from Büchi data automata. It is also not too hard to get a logical characterization of WBDA by extending $\mathsf{EMSO}^2(+1,\sim)$ with existential set quantifiers that are semantically restricted to bind to infinite sets. The answer to question (2) required considerably more effort. However, we establish a 2-NEXPTIME upper bound for the nonemptiness problem for WBDAs by a nondeterministic polynomial time reduction to the nonemptiness for WDA.

Related work. Some related work was already mentioned above. The pioneering works in Linear Temporal Logic for ω-words with data are the papers [10, 9]. In [9] an extension of Linear Temporal Logic (LTL) to handle data values is proposed and its satisfiability problem is shown to be decidable. The decision procedure is a reduction to the reachability problem in Petri nets, thus resulting in a similarly unknown complexity as for data automata. The logic and automata considered in [10] are decidable for finite data words, but not primitive recursive, and undecidable for ω-words. In [17] it is shown that with a *safety* restriction both the logic and the automata become decidable, even in EXPSPACE. In [9] a logic with PSPACE complexity is considered. In [5], MSO logic on data words (with possibly multiple data values per position) is compared to automata models for various types of successor relations.

Organization. We give basic definitions in Section 2. In Section 3, weak data automata are defined, their complexity is given, and their expressive power is compared with other models. Section 4 gives the logical characterization of WDA by $\mathsf{EMSO}^2(+1,\sim)$. Section 5 studies data ω-strings and shows how the nonemptiness problem of WBDA can be nondeterministically reduced in polynomial time to the nonemptiness of WDA. Section 6 states some open problems. Proofs omitted due to space constraints can be found in the full version of this paper ([16]).

Acknowledgement. We thank Christof Löding for helpful remarks on automata and logics for ω-words and Thomas Zeume for thorough proof reading.

2 Notation

Data words. Let Σ be a finite alphabet and $\mathfrak{D}$ an infinite set of data values. A *finite* word is an element of Σ^*, while an ω-word is an element of Σ^ω. A finite *data word* is an element of $(\Sigma \times \mathfrak{D})^*$, while a *data ω-word* is an element of $(\Sigma \times \mathfrak{D})^\omega$. We often refer to data words also as *data strings*.

We write a data (finite or ω-) word w as $\binom{a_1}{d_1}\binom{a_2}{d_2}\cdots$, where $a_1, a_2, \ldots \in \Sigma$ and $d_1, d_2, \ldots \in \mathfrak{D}$. The symbol a_i is the label of position i, while the value d_i is the data value of position i. The projection of w to the alphabet Σ is denoted by $\mathsf{Str}(w) = a_1 a_2 \ldots$. A position in w is called an a-position, if the label of that position is a. We denote by $V_w(a)$, the set of data values found in a-positions in w, i.e., $V_w(a) = \{d_i \mid a_i = a\}$, for each $a \in \Sigma$. Note that some $V_w(a)$'s may be infinite, while some others finite.

A maximal set of positions with the same data value d is called a *class* c^d of the word and the Σ-string induced by the symbols at its positions is called the *class string* w^d. The *profile word* of a data ω-word $w = \binom{a_1}{d_1}\binom{a_2}{d_2}\cdots$ is $\mathsf{Profile}(w) = (a_1, s_1), (a_2, s_2), \ldots \in (\Sigma \times \{\top, \bot\})^\omega$, where for each position $i \geq 1$ the component s_i is $\top$ if and only if $d_i = d_{i+1}$. The profile word of a finite data word $\binom{a_1}{d_1}\binom{a_2}{d_2}\cdots\binom{a_n}{d_n}$ is defined similarly, with the addition that the component s_n is $\bot$.

Automata and Büchi automata. An *automaton* $\mathcal{A}$ over the alphabet Σ is a tuple $\mathcal{A} = \langle \Sigma, Q, q_0, \Delta, F \rangle$, where Q is a finite set of states, $q_0 \in Q$ is the initial state, $\Delta \subseteq Q \times \Sigma \times Q$ is a set of transitions and $F \subseteq Q$ is a set of accepting states. A run of $\mathcal{A}$ on a word $w = a_1 a_2 \ldots a_n$ is a sequence $\rho = q_1 \ldots q_n$ of states from $Q - \{q_0\}$ such that $(q_0, a_1, q_1) \in \Delta$ and $(q_i, a_{i+1}, q_{i+1}) \in \Delta$ for each $1 \leq i < n$. The run ρ is accepting, if $q_n \in F$.

A *Büchi automaton* $\mathcal{A}$ is syntactically just an automaton. A run of $\mathcal{A}$ on an ω-word $w = a_1 a_2 \ldots$ is an infinite sequence $\rho = q_1 q_2 \ldots$ of states from $Q - \{q_0\}$ such that $(q_0, a_1, q_1) \in \Delta$ and $(q_i, a_{i+1}, q_{i+1}) \in \Delta$, for each $i \geq 1$. Let $\mathsf{Inf}(\rho)$ denote the set of states that appear infinitely many times in ρ. The run ρ is accepting if $\mathsf{Inf}(\rho) \cap F \neq \emptyset$.

A word (resp. an ω-word) w is accepted by an automaton (resp. Büchi automaton) $\mathcal{A}$, if there exists an accepting run of $\mathcal{A}$ on w. As usual, $\mathcal{L}(\mathcal{A})$ (resp. $\mathcal{L}^\omega(\mathcal{A})$) denotes the set of words (resp. ω-words) accepted by the automaton $\mathcal{A}$.

Letter-to-letter transducers. A *letter-to-letter transducer* over the input alphabet Σ and the output alphabet Γ is a tuple $\mathcal{T} = \langle \Sigma, \Gamma, Q, q_0, \Delta, F \rangle$, where Q, q_0, F are the set of states, the initial state, and the set of accepting states, respectively, and $\Delta \subseteq Q \times \Sigma \times Q \times \Gamma$ is the set of transitions. The intuitive meaning of a transition (q, a, q', γ) is that when the automaton is in state q, reading the symbol a, then it can move to the state q' and output γ. A *run* of $\mathcal{T}$ on a word $w = a_1 a_2 \ldots a_n$ is a sequence $(q_1, \gamma_1), \ldots, (q_n, \gamma_n)$ over $(Q - \{q_0\}) \times \Gamma$ such that $(q_0, a_1, q_1, \gamma_1) \in \Delta$ and $(q_i, a_{i+1}, q_{i+1}, \gamma_{i+1}) \in \Delta$, for each $1 \leq i < n$. Likewise, a *run* of $\mathcal{T}$ on an ω-word $w = a_1 a_2 \ldots$ is a sequence $(q_1, \gamma_1), (q_2, \gamma_2), \ldots$ over $(Q - \{q_0\}) \times \Gamma$ such that $(q_0, a_1, q_1, \gamma_1) \in \Delta$ and $(q_i, a_{i+1}, q_{i+1}, \gamma_{i+1}) \in \Delta$, for each $i \geq 1$. A run is *accepting* if it is accepting in the sense of (Büchi) automata. We say that $v = \gamma_1 \gamma_2 \ldots$ is an output of $\mathcal{T}$ on w, if there exists an accepting run $(q_1, \gamma_1), (q_2, \gamma_2), \ldots$ of $\mathcal{T}$ on w.

Data automata. A *data automaton (DA)* is a pair $(\mathcal{A}, \mathcal{B})$, where $\mathcal{A}$ is a letter-to-letter transducer with input alphabet $\Sigma \times \{\top, \bot\}$ and output alphabet Γ and $\mathcal{B}$ is a finite state automaton over the alphabet Γ. A data word w is accepted by $(\mathcal{A},\mathcal{B})$ if the following holds.

– $\mathsf{Profile}(w)$ is accepted by $\mathcal{A}$, yielding an output u.
– For each data value d of w, the class string u^d is accepted by $\mathcal{B}$.

Data automata were introduced in the stated form in [4]. In [1] it was shown that their expressive power is not affected, if $\mathcal{A}$ gets $\mathsf{Str}(w)$ as input as opposed to $\mathsf{Profile}(w)$. In more recent papers, data automata are therefore defined in the (syntactically) weaker form with input $\mathsf{Str}(w)$.

3 Weak data automata

In this section we define a new automata model for finite data words and study its expressive power and its complexity. The model follows a similar approach as the model of data automata. The profile of the input data word is transformed by a letter-to-letter transducer and then further conditions on the resulting class strings are imposed. However, the conditions that can be stated in the new automata model are much more limited than those of a data automaton (hence the name *weak* data automata).

Let Γ be an alphabet. Weak data automata allow three kinds of data constraints over Γ:

1. *key constraints*, written in the form: $\mathsf{key}(\gamma)$, where $\gamma \in \Gamma$.
2. *inclusion constraints*, written in the form: $V(\gamma) \subseteq \bigcup_{\gamma' \in R} V(\gamma')$, where $\gamma \in \Gamma$, $R \subseteq \Gamma$.
3. *denial constraints*, written in the form: $V(\gamma) \cap V(\gamma') = \emptyset$, where $\gamma, \gamma' \in \Gamma$.

Whether a data word w satisfies a data constraint C, written as $w \models C$, is defined as follows.

1. $w \models \mathsf{key}(\gamma)$, if every two γ-positions in w have different data values.
2. $w \models V(\gamma) \subseteq \bigcup_{\gamma' \in R} V(\gamma')$, if $V_w(\gamma) \subseteq \bigcup_{\gamma' \in R} V_w(\gamma')$.
3. $w \models V(\gamma) \cap V(\gamma') = \emptyset$, if $V_w(\gamma) \cap V_w(\gamma') = \emptyset$.

If $\mathcal{C}$ is a collection of data constraints, then we write $w \models \mathcal{C}$, if $w \models C$ for all $C \in \mathcal{C}$.

A *weak data automaton (WDA)* over the alphabet Σ is a pair $(\mathcal{A},\mathcal{C})$, where $\mathcal{A}$ is a letter-to-letter transducer with input alphabet $\Sigma \times \{\top,\bot\}$ and output alphabet Γ and $\mathcal{C}$ is a collection of data constraints over the alphabet Γ. A data word $w = \binom{a_1}{d_1}\binom{a_2}{d_2}\cdots\binom{a_n}{d_n}$ is accepted by a WDA $(\mathcal{A},\mathcal{C})$, if

– there is an accepting run of $\mathcal{A}$ on Profile(w), with an output $\gamma_1\gamma_2\ldots\gamma_n$, and
– the induced data word $w = \binom{\gamma_1}{d_1}\binom{\gamma_2}{d_2}\cdots\binom{\gamma_n}{d_n}$ satisfies all the constraints in $\mathcal{C}$.

We write $\mathcal{L}(\mathcal{A},\mathcal{C})$ to denote the language that consists of all data words accepted by $(\mathcal{A},\mathcal{C})$.

We first discuss some extensions of WDA by the constraints that were studied in [8].

– *Disjunctive key constraints* are written in the form: $\mathsf{key}(K)$, where $K \subseteq \Gamma$. Such a constraint is satisfied by a data word if each of its classes has at most one position with a symbol from K.

– *Disjunctive inclusion constraints* are written in the form: $\bigcup_{\gamma\in S} V(\gamma) \subseteq \bigcup_{\gamma'\in R} V(\gamma')$, where $S, R \subseteq \Gamma$. Such a constraint is satisfied by a data word if each class with a position with a symbol from S also has a position with a symbol from R.

An *extended weak data automaton* is defined like a WDA but it further allows disjunctive key and inclusion constraints.

Lemma 1. *From each extended WDA $(\mathcal{A}, \mathcal{C})$ an equivalent WDA of polynomial size can be constructed in polynomial time.*

The proof can be found in the full version of this paper ([16]).

Next, we compare the expressive power of weak data automata with other automata models for data words. More precisely we compare it with register automata [14, 1] and data automata. Register automata are an extension of finite state automata with a fixed number of registers in which they can store data values and compare them with the data value of subsequent positions. For a precise definition we refer[1] the reader to [1].

We consider the following two data languages.

– $L_{a<b}$ consists of all data words over the alphabet $\{a, b\}$ with the property that for every a-position i there is a b-position $j > i$ with the same data value;
– L_{a*b} is the subset of $L_{a<b}$ where the next b-position j with the same data value as i always satisfies $j = i + 2$.

Lemma 2. *Neither L_{a*b} nor $L_{a<b}$ can be decided by a WDA.*

Proof. We first show that no WDA decides L_{a*b}. Towards a contradiction, we thus assume that L_{a*b} is decided by some weak data automata $(\mathcal{A}, \mathcal{C})$.

To this end, let $n = |\Gamma|^4 + 1$ and let $d_1, d_1', d_2, d_2', \ldots d_n, d_n'$ be pairwise different data values. We consider the data word

$$w = \binom{a}{d_1}\binom{a}{d_1'}\binom{b}{d_1}\binom{b}{d_1'}\binom{a}{d_2}\binom{a}{d_2'}\binom{b}{d_2}\binom{b}{d_2'}\cdots\binom{a}{d_n}\binom{a}{d_n'}\binom{b}{d_n}\binom{b}{d_n'}$$

of length $4n$. Clearly, w is in L_{a*b} and its profile is $((a,\bot)(a,\bot)(b,\bot)(b,\bot))^n$.

Let $\gamma = \gamma_1\gamma_2\cdots\gamma_{4n}$ be an output of $\mathcal{A}$ on the profile of w such that $\binom{\gamma_1}{d_1}\cdots\binom{\gamma_{4n}}{d_n'}$ satisfies all constraints in $\mathcal{C}$. By the choice of n, there exist numbers i, j with $0 \leq i < j < n$ such that $\gamma_{4i+1}\gamma_{4i+2}\gamma_{4i+3}\gamma_{4i+4} = \gamma_{4j+1}\gamma_{4j+2}\gamma_{4j+3}\gamma_{4j+4}$.

Let u be the data word obtained from w by swapping the positions of the data values $d_{i+1}d_{i+1}'$ and $d_{j+1}d_{j+1}'$. That is,

$$u = \binom{a}{d_1}\cdots\binom{a}{d_{i+1}}\binom{a}{d_{i+1}'}\binom{b}{d_{j+1}}\binom{b}{d_{j+1}'}\cdots\binom{a}{d_{j+1}}\binom{a}{d_{j+1}'}\binom{b}{d_{i+1}}\binom{b}{d_{i+1}'}\cdots\binom{b}{d_n'}.$$

Clearly, $u \notin L_{a*b}$. However, because $\mathsf{Profile}(u) = \mathsf{Profile}(w)$, $\gamma_1\gamma_2\ldots\gamma_{4n}$ is also an output of $\mathcal{A}$ on $\mathsf{Profile}(u)$. Moreover, the sets of $V_u(\gamma) = V_w(\gamma)$, for each $\gamma \in \Gamma$, and therefore the validity of inclusion and denial constraints does not change. Furthermore, as in u and w

[1] The precursor model *finite-memory automata* was defined on "strings over infinite alphabets", that is, essentially data strings without a Σ-component [14].

every data value occurs at exactly one a-position and at exactly one b-position, they cannot be distinguished by key constraints, either. Thus, $u \in \mathcal{L}(\mathcal{A}, \mathcal{C})$, the desired contradiction.

The proof for $L_{a<b}$ is exactly the same, as $w \in L_{a<b}$ and $u \notin L_{a<b}$ (because of $\binom{a}{d_{j+1}}$). ☐

Theorem 1. *(a) The class of data languages that are decided by WDA is strictly included in the class of data languages decided by DA.*
(b) The classes of languages decided by WDA and by register automata are incomparable.

Proof. Towards (a) we first show that every WDA can be translated into a DA and thus WDA decide a subclass of DA. That the subclass is strict can then be concluded from (b) as register automata are captured by DA [1] and thus there is a data language that can be decided by a DA but not a WDA.

Let thus $(\mathcal{A}, \mathcal{C})$ be a WDA. Then $(\mathcal{A}, \mathcal{B})$ is a data automaton for $L(\mathcal{A}, \mathcal{C})$, where the automaton $\mathcal{B}$ tests the constraints in $\mathcal{C}$ as follows.

– For every key constraint $\mathsf{key}(\gamma)$ of $\mathcal{C}$, $\mathcal{B}$ tests that every class string has at most one γ-position.
– For every inclusion constraint $V(\gamma) \subseteq \bigcup_{\gamma' \in R} V(\gamma')$, $\mathcal{B}$ tests that every class string with a γ-position also has a γ'-position, for some $\gamma' \in R$.
– For every denial constraint $V(\gamma) \cap V(\gamma') = \emptyset$, $\mathcal{B}$ checks that classes with a γ-position do not have any γ'-positions.

To show statement (b) we first consider the separation language $L = L_{a*b}$ which cannot be decided by a WDA by Lemma 2. However, L_{a*b} can be easily decided by a register automaton that always stores the last two data values in two registers and the information about their symbols in its state.

On the other hand, the set of all data strings over $\Sigma = \{a\}$ in which every data value occurs only once can easily be decided by a WDA by the identity-transducer and the key constraint $\mathsf{key}(a)$ but not by a register automaton [14]. ☐

The complexity of the nonemptiness problem for WDA follows directly from results in [8].

Theorem 2. *The nonemptiness problem for WDA is decidable in 2-NEXPTIME.*

Proof. In [8], it was shown that given an automaton $\mathcal{A}$ that reads profile strings and a set $\mathcal{C}$ of disjunctive key and inclusion constraints, to decide whether there is a data word w such that $\mathcal{A}$ accepts $\mathsf{Profile}(w)$ and $w \models \mathcal{C}$ can be done in nondeterministic double exponential time.

Clearly, this is basically the same as the nonemptiness problem for WDA with disjunctive key and inclusion constraints only. It thus only remains to show that denial constraints can be translated into disjunctive constraints in a nonemptiness respecting fashion. To this end, a denial constraint $V(\gamma_1) \cap V(\gamma_2) = \emptyset$ can be replaced as follows. We add two new symbols γ_1', γ_2' and require that in each class with γ_i one γ_i' occurs but γ_1' and γ_2' do not co-occur by two inclusion constraints $V(\gamma_1) \subseteq V(\gamma_1')$ and $V(\gamma_2) \subseteq V(\gamma_2')$ and a disjunctive key constraint for $\{\gamma_1', \gamma_2'\}$. ☐

4 A logical characterization of weak data automata

In this section, we give a logical characterization of the data languages decided by weak data automata in terms of existential second order logic. The characterization is an analogue of the Theorem of Büchi, Elgot and Trakhtenbrot [6, 11, 22] for string languages. This theorem can be stated for various logics, the most interesting one for our context is that $\mathsf{EMSO}^2(+1)$ characterizes exactly the regular languages.

We represent data words by logical structures $w = \langle\{1, \ldots, n\}, +1, <, \{a(\cdot)\}_{a\in\Sigma}, \sim\rangle$, where $\{1, \ldots, n\}$ is the set of positions, $+1$ is the successor relation (i.e., $+1(i, j)$ if $i+1 = j$), $<$ is the order relation (i.e., $< (i, j)$ if $i < j$), the $a(\cdot)$'s are the label relations, and $i \sim j$ holds if positions i and j have the same data value. As the empty data word can not be properly represented, the logical characterization of WDA ignores empty data words. That is, if some WDA $(\mathcal{A}, \mathcal{C})$ accepts the empty data string then its language is different from the language of the corresponding formula φ: $\mathcal{L}(\mathcal{A}, \mathcal{C}) = L(\varphi) \cup \{\epsilon\}$.

For a set $\mathcal{S} \subseteq \{+1, <, \sim\}$ of relation symbols, we write $\mathsf{FO}(\mathcal{S})$ for first-order logic with the vocabulary $\mathcal{S}$, $\mathsf{MSO}(\mathcal{S})$ for monadic second-order logic (which extends $\mathsf{FO}(\mathcal{S})$ with quantification over sets of positions), and $\mathsf{EMSO}(\mathcal{S})$ for existential monadic second order logic, that is, all sentences of the form $\exists R_1 \ldots \exists R_m\ \psi$, where ψ is an $\mathsf{FO}(\mathcal{S})$ formula extended with the unary predicates $R_1, \ldots, R_m$. By $\mathsf{FO}^2(\mathcal{S})$ we denote the restriction of $\mathsf{FO}(\mathcal{S})$ to sentences with two variables x and y, and by $\mathsf{EMSO}^2(\mathcal{S})$ the restriction of $\mathsf{EMSO}(\mathcal{S})$ where the first-order part uses only two variables.

4.1 From weak data automata to $\mathsf{EMSO}^2(+1, \sim)$

Theorem 3. *For every weak data automaton $(\mathcal{A}, \mathcal{C})$, an equivalent $\mathsf{EMSO}^2(+1, \sim)$-formula φ is constructible in polynomial time.*

The construction can be found in the full version of this paper ([16]). It is the same as the classical translation from NFAs to MSO formulas. See, for example, [21].

4.2 From $\mathsf{EMSO}^2(+1, \sim)$ to weak data automata

In the following, we use the abbreviation $F(x, y)$ for the formula $\neg y = x + 1 \wedge \neg x = y + 1 \wedge x \neq y$, which states that the distance of x and y is at least two.

Theorem 4. *There is an algorithm that translates every $\mathsf{EMSO}^2(+1, \sim)$-formula φ into an equivalent weak data automaton $(\mathcal{A}, \mathcal{C})$ in doubly exponential time. In particular, the output alphabet Γ of $\mathcal{A}$ and the number of constraints in $\mathcal{C}$ is at most exponenotial.*

Proof. In the first step, the algorithm transforms φ into an equivalent $\mathsf{EMSO}^2(+1, \sim)$ formula in Scott normal form (SNF) of the form $\psi = \exists R_1 \ldots \exists R_n [\forall x \forall y\ \chi' \wedge \bigwedge_{i=1}^{m} \forall x \exists y\ \chi'_i]$, where χ' and each χ'_i are quantifier-free [13]. The size of ψ is linear in the size of φ, in particular, $n = \mathcal{O}(|\varphi|)$ and $m = \mathcal{O}(|\varphi|)$.

Then it rewrites formula χ' into an, at most exponential, conjunction $\chi = \bigwedge_j \neg(\alpha_j(x) \wedge \beta_j(y) \wedge \delta_j(x, y) \wedge \epsilon_j(x, y))$, where, for every j, α_j, β_j are conjunctions of literals with unary relation symbols, δ_j is $x \sim y$ or $x \not\sim y$ and $\epsilon_j(x, y)$ is one[2] of $x = y$, $y = x + 1$ and $F(x, y)$.

[2] The case $x = y + 1$ does not need to be considered as it can be obtained by swapping x and y.

Likewise, it rewrites every χ'_i into an, at most exponential, disjunction $\chi_i = \bigvee_j (\alpha^i_j(x) \wedge \beta^i_j(y) \wedge \delta^i_j(x,y) \wedge \epsilon^i_j(x,y))$, where the atomic formulas are of the respective forms as above.

The idea of the construction is that $\mathcal{A}$ guesses some relations that allow to state some of the properties expressed in ψ by constraints of $\mathcal{C}$. The details are given in [16]. $\square$

We note that in the upper bound of the algorithm for nonemptiness of WDA transferred from [8], the doubly exponential term only depends on the alphabet size. By combing this with the bounds of Theorem 4 we obtain a 3-NEXPTIME upper bound for satisfiability of $\mathsf{FO}^2(+1,\sim)$ (which is worse than the bound in [19]). We also note that the construction underlying the proof of Theorem 4 can be turned into a nondeterministic exponential time reduction from satisfiability for $\mathsf{FO}^2(+1,\sim)$ to nonemptiness for WDA resulting in an automaton with a singly exponential number of states. The reduction guesses the order in which types appear in the accepted string (as opposed to the construction in the proof of Theorem 4).

The previous two theorems yield the following logical characterization of WDA.

Theorem 5. *Weak data automata and $\mathsf{EMSO}^2(+1,\sim)$ are equivalent in expressive power.*

We note that on strings $\mathsf{EMSO}^2(+1)$ and $\mathsf{EMSO}^2(+1,<)$ are expressively equivalent. It is an interesting consequence of the above characterization that this equivalence does not hold for data strings.

Corollary 1. *The logic $\mathsf{EMSO}^2(+1,\sim)$ is strictly less expressible than $\mathsf{EMSO}^2(+1,<,\sim)$.*

Proof. The inclusion holds by definition. It is strict because the language $L_{a<b}$ cannot be decided by an WDA (Lemma 2) and thus cannot be defined in $\mathsf{EMSO}^2(+1,\sim)$, but it can be expressed by the simple formula $\forall x \exists y (a(x) \rightarrow (b(y) \wedge x < y \wedge x \sim y))$. $\square$

5 Weak Büchi data automata

In this section we consider automata and logics for *data ω-words*, that is, data words of infinite length. Weak data automata $(\mathcal{A},\mathcal{C})$ can easily be adapted for data ω-words. The automaton $\mathcal{A}$ is simply interpreted as a letter-to-letter Büchi transducer. A run is accepting if it visits infinitely often a state from F. We refer to the resulting model as weak Büchi data automata (WBDA). We write $\mathcal{L}^\omega(\mathcal{A},\mathcal{C})$ for the set of data ω-words accepted by $(\mathcal{A},\mathcal{C})$. The results regarding expressive power of WDA compared with other automata models easily carry over to WBDA.

Data ω-words can be represented by logical structures $w = \langle \mathbb{N}, +1, <, \{a(\cdot)\}_{a\in\Sigma}, \sim\rangle$, where $\mathbb{N}$ is the set $\{1,2,\ldots\}$ of natural numbers which represent the positions and the other relations are as in the case of data words. For a set $\mathcal{S} \subseteq \{+1,<,\sim\}$ of relation symbols $\mathsf{E}_\infty\mathsf{MSO}(\mathcal{S})$ consists of all formulas of the form $\exists_\infty R_1 \ldots \exists_\infty R_m \exists S_1 \ldots \exists S_\ell\, \varphi$ where $\varphi \in \mathsf{FO}^2(\mathcal{S})$. Here all relation symbols R_i, S_i are unary. The $\exists_\infty$ are semantically restricted to bind to infinite sets only.

Remark 1. It is folklore that languages (without data) accepted by Büchi automata are precisely languages expressible in formulae of the form:

$$\exists_\infty R_1 \cdots \exists_\infty R_m \exists S_1 \cdots \exists S_\ell \quad \varphi$$

for some $\varphi \in \mathsf{FO}^2(+1)$. However, we have not found an explicit reference for this result in the literature.

The following theorem is a straightforward generalization of Theorem 5.

Theorem 6. *Weak Büchi data automata and $\mathsf{E}_\infty\mathsf{MSO}^2(+1,\sim)$ are equivalent in expressive power.*

The proof is sketched in [16].

Theorem 7. *The nonemptiness problem for weak Büchi data automata is decidable in 2-NEXPTIME.*

Proof. We show in the following that the nonemptiness problem for WBDA can be polynomially reduced to the nonemptiness problem for WDA. The result then follows from Theorem 2. The approach is a classical one. We show that if the language of a WBDA $(\mathcal{A},\mathcal{C})$ is non-empty then a finite data string of the form uv can be constructed such that there is a run of $\mathcal{A}$ which loops over v. The "unravelling" uv^ω is then also accepted by the automaton. However, some care is needed to assign data values in a suitable manner.

Let $(\mathcal{A},\mathcal{C})$ be a WBDA with $\mathcal{A} = \langle \Sigma, \Gamma, Q, q_0, \Delta, F\rangle$. Since we are only interested in whether $\mathcal{L}^\omega(\mathcal{A},\mathcal{C}) = \emptyset$, we can assume, without loss of generality, that the transitions of $\mathcal{A}$ are all of the form (q,γ,q',γ). Otherwise, we can replace it by a transducer which reads Γ-strings and guesses, for every position i, a symbol $a_i \in \Sigma$, its profile symbol s_i (and store them in the state) and verifies that its output would be (the actual input symbol) γ_i. Therefore, we consider $\mathcal{A}$ in this proof just as a normal Büchi automaton that gets a Γ-string as input. The constraints are applied to the same string.

We first fix some notation. We refer to the symbols that occur in key constraints of $\mathcal{C}$ as *key symbols*.

A *zone* is a finite data string over Γ in which all positions carry the same data value. An *ω-zone* is an infinite data string over Γ in which all positions carry the same data value. The *zones of a data string* w are the maximal zones of w. An *adorned zone* is a zone together with a pair (q,q') of states of $\mathcal{A}$. We write $\mathsf{a\text{-}Proj}(z)$ for the triple $(\mathsf{Str}(z),q,q')$ of a zone z that is adorned with the pair (q,q').

We next define an important notion for this proof, (singular and non-singular) witnesses. We will show that the nonemptiness of $(\mathcal{A},\mathcal{C})$ boils down to deciding whether such witnesses exist. Singular witnesses correspond to data strings in $\mathcal{L}^\omega(\mathcal{A},\mathcal{C})$ with an infinite zone whereas non-singular witnesses correspond to data strings with finite zones only.

A *singular witness* for $(\mathcal{A},\mathcal{C})$ is a data string uv over Γ the following properties.

- $uv \models \mathcal{C}$.
- There is a state $\hat{q} \in F$ and a (partial) run $\rho = \rho_u\rho_v$ of $\mathcal{A}$ on input $\mathsf{Profile}(uv)_\top$ in which the state after reading u and after reading v is $\hat{q}$. Here, $\mathsf{Profile}(uv)_\top$ denotes the profile string that is obtained from $\mathsf{Profile}(uv)$ by setting the last profile symbol to $\top$.
- All positions of v and the last zone of u carry the same data value and v does not carry any key symbol.

A *non-singular witness* for $(\mathcal{A},\mathcal{C})$ is a data string uv over Γ which fulfills the following conditions.

– All zones in uv are of length at most $|Q|(|\Gamma|+1)$.
– The data value of the last position of u is different from the value of the first position of v.
– There is a state $\hat{q}$ and a (partial) run $\rho = \rho_u\rho_v$ of $\mathcal{A}$ on input uv in which the state after reading u and after reading v is $\hat{q}$. Furthermore, ρ_v contains some state from F. In the following, each zone z of w is adorned by the pair (q, q') where q is the state of ρ before reading z and q' is the state after reading z.
– The classes of uv can be colored[3] with the four colors black, yellow, white and blue such that all black, yellow and white classes satisfy [4] all constraints from $\mathcal{C}$ and furthermore the following conditions hold.

(black) There are at most $3|Q|^2$ black classes. There are no key symbols in black zones of v. Furthermore, it is not the case that the first zone and the last zone of v are from the same black class.
(yellow) There are at most $|Q|^2$ yellow classes and they consist of at most $|\Gamma|$ zones. All these zones are located in v.
(white) All zones of the white classes are located in u.
(blue) For each blue zone z there is a yellow zone z' such that $\mathsf{a\text{-}Proj}(z)$=$\mathsf{a\text{-}Proj}(z')$.

The proof of decidability of the nonemptiness problem for WBDA now reduces to proving the following three claims.

(Claim 1) If there exists a witness for $(\mathcal{A}, \mathcal{C})$ then $\mathcal{L}^\omega(\mathcal{A}, \mathcal{C}) \neq \emptyset$.
(Claim 2) If $\mathcal{L}^\omega(\mathcal{A}, \mathcal{C}) \neq \emptyset$ then there exists a witness for $(\mathcal{A}, \mathcal{C})$.
(Claim 3) There is a nondeterministic algorithm which constructs, for every WBDA $(\mathcal{A}, \mathcal{C})$, in polynomial time some WDA $(\mathcal{A}', \mathcal{C}')$ such that every possible $(\mathcal{A}', \mathcal{C}')$ accepts only witnesses for $(\mathcal{A}, \mathcal{C})$ and for each witness uv there is a run of the algorithm producing some $(\mathcal{A}', \mathcal{C}')$ that accepts uv.

Therefore, the nonemptiness problem for WBDA can indeed be reduced non-deterministically in polynomial time to the nonemptiness problem for WDA. The proofs of these claims are given in [16]. □

6 Conclusion

We conclude this paper with two open problems for future directions. An obvious open problem is the exact complexity of the nonemptiness problem for weak data automata. The current 2-NEXPTIME yields a 3-NEXPTIME upper bound for the satisfiability problem for $\mathsf{EMSO}^2(+1, \sim)$. However, as it is known that this problem can be solved in 2-NEXPTIME [19], some room for improvement is left.

Another interesting question is how our results can be applied to temporal logics. In [10], a restriction of LTL with one register, *simple* LTL, was considered with the same expressive power as some two variable logic. We conjecture that there is a correspondence between our logics and the restriction of simple LTL to the operators X, X^{-1} and an operator that allows navigation to some other position.

[3] Each class gets exactly one color. We refer to zones and positions in a black class as black zones and positions, respectively, and likewise for the other colors.
[4] We do not require that the blue classes satisfy $\mathcal{C}$.

References

1. Henrik Björklund and Thomas Schwentick. On notions of regularity for data languages. *Theor. Comput. Sci.*, 411(4-5):702–715, 2010.
2. Luc Boasson. Some applications of CFL's over infinte alphabets. In *Theoretical Computer Science*, pages 146–151, 1981.
3. Mikolaj Bojanczyk, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. Two-variable logic on data trees and XML reasoning. *J. ACM*, 56(3), 2009.
4. Mikolaj Bojanczyk, Anca Muscholl, Thomas Schwentick, Luc Segoufin, and Claire David. Two-variable logic on words with data. In *LICS*, pages 7–16, 2006.
5. Benedikt Bollig. An automaton over data words that captures EMSO logic. *CoRR*, abs/1101.4475, 2011.
6. J. R. Büchi. Weak second-order arithmetic and finite automata. *Z. Math. Logik Grundl. Math.*, 6:66–92, 1960.
7. Edward Y. C. Cheng and Michael Kaminski. Context-free languages over infinite alphabets. *Acta Inf.*, 35(3):245–267, 1998.
8. Claire David, Leonid Libkin, and Tony Tan. On the satisfiability of two-variable logic over data words. In *LPAR (Yogyakarta)*, pages 248–262, 2010.
9. Stéphane Demri, Deepak D'Souza, and Régis Gascon. A decidable temporal logic of repeating values. In *LFCS*, pages 180–194, 2007.
10. Stéphane Demri and Ranko Lazic. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Log.*, 10(3), 2009.
11. Calvin C. Elgot. Decision problems of finite automata design and related arithmetics. *Transactions of The American Mathematical Society*, 98:21–21, 1961.
12. Jay L. Gischer. Shuffle languages, Petri nets, and context-sensitive grammars. *Commun. ACM*, 24(9):597–605, 1981.
13. Erich Grädel and Martin Otto. On logics with two variables. *Theor. Comput. Sci.*, 224(1-2):73–113, 1999.
14. Michael Kaminski and Nissim Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134(2):329–363, 1994.
15. Michael Kaminski and Tony Tan. Regular expressions for languages over infinite alphabets. *Fundam. Inform.*, 69(3):301–318, 2006.
16. Ahmet Kara, Thomas Schwentick, and Tony Tan. Feasible automata for two-variable logic with successor on data words. Available from *arXiv:1110.1221v1*.
17. Ranko Lazic. Safety alternating automata on data words. *ACM Trans. Comput. Log.*, 12(2):10, 2011.
18. Frank Neven, Thomas Schwentick, and Victor Vianu. Finite state machines for strings over infinite alphabets. *ACM Trans. Comput. Log.*, 5(3):403–435, 2004.
19. Matthias Niewerth and Thomas Schwentick. Two-variable logic and key constraints on data words. In *ICDT*, pages 138–149, 2011.
20. Friedrich Otto. Classes of regular and context-free languages over countably infinite alphabets. *Discrete Applied Mathematics*, 12(1):41 – 56, 1985.
21. Wolfgang Thomas. Languages, automata, and logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages, Vol. III*, pages 389–455. Springer, New York, 1997.
22. Boris Trakhtenbrot. Finite automata and logic of monadic predicates. *Doklady Akademii Nauk SSSR*, 140:326–329, 1961.