Partially Ordered Two-way Büchi Automata*

Manfred Kufleitner   Alexander Lauser

FMI, Universität Stuttgart, Germany
{kufleitner, lauser}@fmi.uni-stuttgart.de

June 14, 2010

Abstract

We introduce partially ordered two-way Büchi automata over infinite words. As for finite words, the nondeterministic variant recognizes the fragment Σ_2 of first-order logic FO[<] and the deterministic version yields the Δ_2-definable ω-languages. As a byproduct of our results, we show that deterministic partially ordered two-way Büchi automata are effectively closed under Boolean operations.

In addition, we have coNP-completeness results for the emptiness problem and the inclusion problem over deterministic partially ordered two-way Büchi automata.

Keywords. infinite words; partially ordered two-way Büchi automaton; first-order logic

1 Introduction

We combine partially ordered two-way (po2) automata with the Büchi acceptance condition. For this new subclass of two-way Büchi automata, we characterize the expressive power of the nondeterministic and the deterministic versions. Moreover, we show that nondeterministic po2-Büchi automata have a *small model property*. This leads to NP-completeness results of the non-emptiness problem for both nondeterministic and deterministic po2-Büchi automata, and to the coNP-completeness of the inclusion problem for deterministic po2-Büchi automata.

Büchi automata have been introduced in order to decide monadic second-order logic over infinite words [2]. Today, they have become one of the most important tools in model-checking sequential finite state systems, see e.g. [1, 3]. Büchi automata are nondeterministic finite automata, accepting infinite words if there exists an infinite run such that some final state occurs infinitely often. A generalization are two-way Büchi automata; Pécuchet has shown that they have the same expressive power as ordinary Büchi automata [9]. Alternating two-way Büchi automata have been used for model checking of temporal logic formulas with past modalities [6, 15]. These automata, too, can recognize nothing but regular ω-languages. With the usual padding technique, the succinctness result for two-way automata over finite words [5] immediately yields an exponential lower bound for the succinctness of two-way Büchi automata.

We characterize the expressive power of po2-Büchi automata in terms of fragments of first-order logic FO[<]. The fragment Σ_2 consists of all FO[<]-sentences in prenex normal form with one block of existential quantifiers followed by one block of universal quantifiers followed by a propositional formula. The fragment Π_2 contains the negations of Σ_2-formulas.

*This work was supported by the German Research Foundation (DFG), grant DI 435/5-1.

By abuse of notation, we identify logical fragments with the classes of ω-languages they define. Hence, it makes sense to define $\Delta_2 = \Sigma_2 \cap \Pi_2$, i.e., an ω-language is Δ_2-definable if it is Σ_2-definable and Π_2-definable. Therefore, Δ_2 is the largest subclass of Σ_2 (or Π_2) which is closed under complementation. Various characterizations of Σ_2 and of Δ_2 over infinite words are known [14, 4]. Note that it makes a difference whether we require that a Σ_2-formula and a Π_2-formula have the same ω-word models, or whether they coincide on finite word models. In some sense, Δ_2 over infinite words is weaker than Δ_2 over finite words. For example over finite words, Δ_2 has the same expressive power as first-order logic with only two variables [13], whereas over infinite words, Δ_2 is weaker than first-order logic with two variables [4]. Moreover, Δ_2 over finite words coincides with a language class called *unambiguous polynomials* [10], whereas over infinite words, only some restricted variant of unambiguous polynomials is definable in Δ_2 [4].

Schwentick, Thérien, and Vollmer introduced po2-automata over finite words [11]; cf. [7] for further characterizations of such automata. A po2-automaton is a two-way automaton with the property that once a state is left, it is never entered again. Every such automaton admits a partial order on its states such that transitions are non-decreasing. In fact, one could use a linear order on the states, but this would distort the length of a longest chain, which in some cases is a useful parameter. Nondeterministic po2-automata recognize exactly the Σ_2-definable languages over finite words whereas deterministic po2-automata correspond to Δ_2-definable languages [11].

In this paper, we present analog results over infinite words. More precisely, for $L \subseteq \Gamma^\omega$ we show that

- L is recognized by some nondeterministic partially ordered two-way Büchi automaton if and only if L is definable in Σ_2 (Theorem 1),
- L is recognized by some deterministic partially ordered two-way Büchi automaton if and only if L is definable in Δ_2 (Theorem 3).

In particular, nondeterministic po2-Büchi automata are more powerful than deterministic po2-Büchi automata, and nondeterministic po2-Büchi automata are not closed under complementation. The proof of Theorem 1 is a straightforward generalization of the respective result for finite words. It is presented here for the sake of completeness. The proof of Theorem 3 is new. It is based on a language description from [4] rather than on so called *turtle languages* as in [11]. The main step in our proof is to show that deterministic po2-Büchi automata are effectively closed under Boolean operations (Theorem 2). This is non-trivial, since the approach of starting a second automaton after the first one has completed its computation does not work for Büchi automata. To this end, we simulate two deterministic po2-Büchi automata simultaneously, and we have to do some bookkeeping of positions if the two automata walk in different directions. In Section 5, we show that various decision problems over po2-Büchi automata are coNP-complete: the emptiness problem for deterministic and for nondeterministic po2-Büchi automata; and the universality, the inclusion, and the equivalence problem for deterministic po2-Büchi automata. Note that for (non-partially-ordered) one-way Büchi automata, both the inclusion problem and the equivalence problem are PSPACE-complete [12].

2 Preliminaries

Throughout this paper, Γ denotes a finite alphabet. For $A \subseteq \Gamma$, the set of finite words over A is A^* and the set of infinite words over A is A^ω. If we want to emphasize that $\alpha \in \Gamma^\omega$ is an infinite word, then we say that α is an *ω-word*. The empty word is ε. We have $\emptyset^* = \{\varepsilon\}$ and $\emptyset^\omega = \emptyset$. The *length* of a finite word $w \in \Gamma^*$ is denoted by $|w|$, i.e., $|w| = n$ if $w = a_1 \cdots a_n$ with $a_i \in \Gamma$. We set $|\alpha| = \infty$ if $\alpha \in \Gamma^\omega$. The *alphabet* of a word $\alpha = a_1 a_2 \cdots \in \Gamma^* \cup \Gamma^\omega$ is denoted by $\mathrm{alph}(\alpha)$. It is the set of letters occurring in α. We say that a position i of α is an *a-position* of α if $a_i = a$.

A *language* is a subset of Γ^* or a subset of Γ^ω. As for ω-words, we emphasize that $L \subseteq \Gamma^\omega$ contains only infinite words by saying that L is an *ω-language*. A *monomial* (of *degree* k) is a language of the form $A_1^* a_1 \cdots A_k^* a_k A_{k+1}^*$. It is *unambiguous* if each word w in the monomial has a unique factorization $w = u_1 a_1 \cdots u_k a_k u_{k+1}$ with $u_i \in A_i^*$. Similarly, an *ω-monomial* is an ω-language of the form $A_1^* a_1 \cdots A_k^* a_k A_{k+1}^\omega$ and it is *unambiguous* if each word $\alpha \in A_1^* a_1 \cdots A_k^* a_k A_{k+1}^\omega$ has a unique factorization $u_1 a_1 \cdots u_k a_k \beta$ with $u_i \in A_i^*$ and $\beta \in A_{k+1}^\omega$. A *restricted unambiguous ω-monomial* is an unambiguous ω-monomial $A_1^* a_1 \cdots A_k^* a_k A_{k+1}^\omega$ such that $\{a_i, \ldots, a_k\} \not\subseteq A_i$ for all $1 \leq i \leq k$. A *polynomial* is a finite union of monomials and an *ω-polynomial* is a finite union of ω-monomials. A *restricted unambiguous ω-polynomial* is a finite union of restricted unambiguous ω-monomials.

By FO[<] we denote the first-order logic over words interpreted as labeled linear orders. As atomic formulas, FO[<] comprises $\top$ (for *true*) and $\bot$ (for *false*), the unary predicate $\lambda(x) = a$ for $a \in \Gamma$, and the binary predicate $x < y$ for variables x and y. The idea is that variables range over the linearly ordered positions of a word and $\lambda(x) = a$ means that x is an a-position. Apart from the Boolean connectives, we allow quantifications over position variables, i.e., existential quantifications $\exists x \colon \varphi$ and universal quantifications $\forall x \colon \varphi$ for $\varphi \in \mathrm{FO}[<]$. The semantics is as usual.

Every formula in FO[<] can be converted into a semantically equivalent formula in prenex normal form by renaming variables and moving quantifiers to the front. This gives rise to the fragment Σ_2 (resp. Π_2) consisting of all FO[<]-formulas in prenex normal form with only two blocks of quantifiers, starting with a block of existential quantifiers (resp. universal quantifiers). Note that the negation of a formula in Σ_2 is equivalent to a formula in Π_2 and vice versa. The fragments Σ_2 and Π_2 are both closed under conjunction and disjunction.

A *sentence* in FO[<] is a formula without free variables. Since there are no free variables in a sentence φ, the truth value of $\alpha \models \varphi$ is well-defined. The *ω-language defined by φ* is $L(\varphi) = \{\alpha \in \Gamma^\omega \mid \alpha \models \varphi\}$. We frequently identify logical fragments with the respective classes of languages. For example, $\Delta_2 = \Sigma_2 \cap \Pi_2$ consist of all languages L such that $L = L(\varphi) = L(\psi)$ for some $\varphi \in \Sigma_2$ and $\psi \in \Pi_2$, i.e., a language L is Δ_2-definable if there are equivalent formulas in Σ_2 and in Π_2 defining L. The notion of *equivalence* depends on the models and it turns out to be a difference whether we use finite or infinite words as models, cf. [4, 13]. Unless stated otherwise, we shall only use infinite word models. In particular, for the remainder of this paper Δ_2 is a class of ω-languages.

2.1 Partially Ordered Two-way Büchi Automata

In the following, we give the Büchi automaton pendant of a two-way automaton. This is basically a Büchi automaton that may change the direction in which it reads the input. A *two-way Büchi automaton* $\mathcal{A} = (Z, \Gamma, \delta, X_0, F)$ is given by:

- a finite set of states $Z = X \dot\cup Y$,
- a finite input alphabet Γ; the tape alphabet is $\Gamma \dot\cup \{\triangleright\}$, where the left end marker $\triangleright$ is a new symbol,
- a transition relation $\delta \subseteq (Z \times \Gamma \times Z) \cup (Y \times \{\triangleright\} \times X)$,
- a set of initial states $X_0 \subseteq X$, and
- a set of final states $F \subseteq Z$.

The states Z are partitioned into "ne$X$t-states" X and "Yesterday-states" Y. The idea is that states in X are entered with a right-move of the head while states in Y are entered with a left-move. For $(z, a, z') \in \delta$ we frequently use the notation $z \xrightarrow{a} z'$. On input $\alpha = a_1 a_2 \cdots \in \Gamma^\omega$ the tape is labeled by $\triangleright\,\alpha$, i.e., positions $i \geq 1$ are labeled by a_i and position 0 is labeled by $\triangleright$. A *configuration* of the automaton is given by a pair (z, i) where $z \in Z$ is a state and $i \in \mathbb{N}$ is the current position of the head. A *transition* $(z, i) \vdash (z', j)$ on configurations (z, i) and (z', j) exists, if

- $z \xrightarrow{a} z'$ for some $a \in \Gamma \cup \{\triangleright\}$ such that i is an a-position, and
- $j = i + 1$ if $z' \in X$, and $j = i - 1$ if $z' \in Y$.

The $\triangleright$-position can only be encountered in a state from Y and left via a state from X. In particular, $\mathcal{A}$ can never overrun the left end marker $\triangleright$. Due to the partition of the states Z, we can never have a change in direction without changing the state. A configuration (z, i) is *initial*, if $z \in X_0$ and $i = 1$. A *computation* of $\mathcal{A}$ on input α is an infinite sequence of transitions

$$(z_0, i_0) \vdash (z_1, i_1) \vdash (z_2, i_2) \vdash \cdots$$

such that (z_0, i_0) is initial. It is *accepting*, if there exists some state $y \in F$ which occurs infinitely often in this computation. Now, $\mathcal{A}$ *accepts* an input α if there is an accepting computation of $\mathcal{A}$ on input α. As usual, the language recognized by $\mathcal{A}$ is $L(\mathcal{A}) = \{\alpha \in \Gamma^\omega \mid \mathcal{A} \text{ accepts } \alpha\}$.

A two-way Büchi automaton is *deterministic* if $|X_0| = 1$ and if for every state $z \in Z$ and every symbol $a \in \Gamma \cup \{\triangleright\}$ there is at most one $z' \in Z$ with $z \xrightarrow{a} z'$. A two-way Büchi automaton is *complete* if for every state $z \in Z$ and every symbol $a \in \Gamma$ there is at least one $z' \in Z$ with $z \xrightarrow{a} z'$, and for every $z \in Y$ there is at least one $z' \in X$ with $z \xrightarrow{\triangleright} z'$.

We are now ready to define *partially ordered* two-way Büchi automata. We use the abbreviation "po2" for "partially ordered two-way". A two-way Büchi automaton $\mathcal{A}$ is a *po2-Büchi automaton*, if there is a partial order $\preccurlyeq$ on the set of states Z such that every transition is non-descending, i.e., if $z \xrightarrow{a} z'$ then $z \preccurlyeq z'$. In po2-Büchi automata, every computation enters a state at most once and it defines a non-decreasing sequence of states. Since there can be no infinite chain of states, every computation has a unique state $z \in Z$ which occurs infinitely often and this state is maximal among all states in the computation. Moreover, $z \in X$ since the automaton cannot loop in a left-moving state forever. We call this state z *stationary.* A computation is accepting if and only if its stationary state z is a final state. In particular, we can always assume $F \subseteq X$ in po2-Büchi automata.

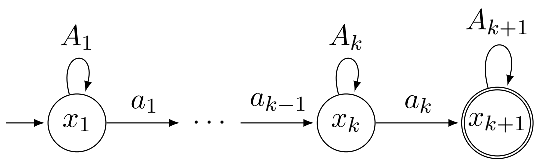
3 Nondeterministic po2-Büchi Automata

In this section, we show that nondeterministic po2-Büchi automata recognize exactly the class of Σ_2-definable languages. Moreover, it turns out that nondeterministic po2-Büchi automata and nondeterministic partially ordered one-way Büchi automata (i.e., $Y = \emptyset$ in our definition of nondeterministic po2-Büchi automata) have the same expressive power. The proof is a straightforward extension of the respective result for finite words [11]. It is presented here only for the sake of completeness.

Theorem 1 *Let $L \subseteq \Gamma^\omega$. The following assertions are equivalent:*

1. *L is recognized by a nondeterministic po2-Büchi automaton.*
2. *L is Σ_2-definable.*
3. *L is recognized by a nondeterministic partially ordered Büchi automaton.*

Proof: "1 $\Rightarrow$ 2": Let $\mathcal{A}$ be a partially ordered two-way Büchi automaton. It suffices to show that $L(\mathcal{A})$ is an ω-polynomial, since every ω-polynomial is Σ_2-definable. This follows from Lemma 1 below (with $\mathcal{A} = \mathcal{B}$). "2 $\Rightarrow$ 3": Every Σ_2-definable ω-language is an ω-polynomial [14]. The ω-monomial $A_1^* a_1 \cdots A_k^* a_k A_{k+1}^\omega$ is recognized by the following Büchi automaton:



Now, every ω-polynomial can be recognized by a finite union of such automata. "3 $\Rightarrow$ 1": Every partially ordered one-way Büchi automaton is a special case of a po2-Büchi automaton. $\square$

Lemma 1 *Let $\mathcal{A}$ and $\mathcal{B}$ be complete po2-Büchi automata and let $n_\mathcal{A}$ and $n_\mathcal{B}$ be the lengths of the longest chains in the state sets of $\mathcal{A}$ and $\mathcal{B}$, respectively. Then for every $\alpha \in L(\mathcal{A}) \cap L(\mathcal{B})$ there exists an ω-monomial P_α of degree at most $n_\mathcal{A} + n_\mathcal{B} - 2$ such that $\alpha \in P_\alpha \subseteq L(\mathcal{A}) \cap L(\mathcal{B})$. In particular,*

$$L(\mathcal{A}) \cap L(\mathcal{B}) \quad = \bigcup_{\alpha \in L(\mathcal{A}) \cap L(\mathcal{B})} P_\alpha$$

is an ω-polynomial, since there are only finitely many ω-monomials of degree at most $n_\mathcal{A} + n_\mathcal{B} - 2$.

Proof: Let $\alpha \in L(\mathcal{A}) \cap L(\mathcal{B})$ and consider an accepting computation of $\mathcal{A}$ and an accepting computation of $\mathcal{B}$. For these computations, we define the factorization $\alpha = u_1 a_1 \cdots u_k a_k \beta$ with $a_i \in \Gamma$, $u_i \in \Gamma^*$, and $\beta \in \Gamma^\omega$ such that the positions of the markers a_i are exactly those where a state change happens in at least one of the computations. In each traversal of one of the factors u_i and β, the letters in these factors correspond to self-loops on the respective states in both computations. Hence, $P_\alpha = A_1^* a_1 \cdots A_k^* a_k B^\omega$ with $A_i = \mathrm{alph}(u_i)$ and $B = \mathrm{alph}(\beta)$ satisfies $\alpha \in P_\alpha$, $P_\alpha \subseteq L(\mathcal{A})$, and $P_\alpha \subseteq L(\mathcal{B})$. $\square$

4 Deterministic po2-Büchi Automata

This section contains the main contribution of our paper, namely that the class of languages recognizable by deterministic po2-Büchi automata is exactly the fragment Δ_2 of first-order logic. Our proof relies on a characterization of Δ_2 in terms of restricted unambiguous ω-polynomials [4].

As an intermediate step, we show in Theorem 2 that deterministic po2-Büchi automata are effectively closed under Boolean operations. The proof is split into two parts: First, we show the closure under complementation in Lemma 2. This result is surprising in the sense that for general deterministic one-way Büchi automata (not necessarily partially ordered), the same result does not hold. Proposition 1 gives an effective construction on deterministic po2-Büchi automata recognizing the union and the intersection of two languages given by deterministic po2-Büchi automata.

Theorem 2 *The class of languages recognized by deterministic po2-Büchi automata is effectively closed under complementation, union, and intersection.*

Proof: Effective closure under complementation of po2-Büchi automata will be shown in Lemma 2. Effective closure under positive finite Boolean combinations is Proposition 1. □

Lemma 2 *The class of languages recognized by deterministic po2-Büchi automata is effectively closed under complementation. Moreover, the complement automaton has at most one additional state and it can be computed in polynomial time.*

Proof: Let $\mathcal{A}$ be a complete deterministic po2-automaton. For every word α we have a unique computation. Therefore, every word α uniquely determines a stationary state x_α, and α is accepted by $\mathcal{A}$ if and only if x_α is final. Thus complementing the set of final states yields a deterministic po2-automaton $\overline{\mathcal{A}}$ with $L(\overline{\mathcal{A}}) = \Gamma^\omega \setminus L(\mathcal{A})$. We note that complementing F with respect ot the right-moving states X instead of all states Z is sufficient. Obviously, the necessary computations can be done in polynomial time. We might need to add one state in order to make $\mathcal{A}$ complete. □

Proposition 1 *The class of languages recognized by deterministic po2-Büchi automata is effectively closed under union and intersection.*

Proof: Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be complete deterministic po2-Büchi automata. We give a product automaton construction $\mathcal{A}$ recognizing $L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$. With a different choice of the final states, the same automaton also recognizes $L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$. We start with a description of the general idea of our construction. Details are given below. The automaton $\mathcal{A}$ operates in two modes: the *synchronous mode* and the *asynchronous mode*. In the synchronous mode $\mathcal{A}$ executes both automata at the same time until at least one of them changes to a left-moving state. Then $\mathcal{A}$ changes to the asynchronous mode by activating a left-moving automaton and suspending the other one. The position where this divergence happens is called the *synchronization point*. We stay in the asynchronous mode until the synchronization point is reached again. In a complete partially ordered automaton this must happen eventually. If the two automata now agree on going to the right, we switch back to the synchronous mode; else the process is repeated.

In order to recognize the synchronization point while executing the active automaton in the asynchronous mode, $\mathcal{A}$ administers a stack of letters and a pointer on this stack. The stack records the letters which led to a state change during synchronous mode in at least one of the automata. The corresponding positions of the word are called *marker positions* and its labels are *markers*. Let $a_1 \cdots a_m$ be the sequence of markers encountered during the computation and let $p_1 < \cdots < p_m$ be the respective marker positions. Changing from synchronous mode to asynchronous mode involves a state change of one of the automata $\mathcal{A}_1$ and $\mathcal{A}_2$. In particular, if $\mathcal{A}$ is in the asynchronous mode, then a_m is the label of the synchronization point p_m. Since both automata are deterministic, we have that for every $1 \leq k \leq m$ the prefix of the input of length p_k is the shortest prefix admitting $a_1 \cdots a_k$ as a (scattered) subword. Our construction takes advantage of this observation for detecting the synchronization point and in order to keep the pointer up to date while simulating the active automaton. The semantics of the pointer is as follows: If it points to a marker a_k then the current position q of $\mathcal{A}$ is in the interval $[p_{k-1}; p_m]$ and moreover, $a_k \cdots a_m$ is a scattered subword of the factor induced by the interval $[q; p_m]$. Here, we set $p_0 = 0$ to be the position of the left end marker $\triangleright$ for convenience. If the head is on an a_m-position and the pointer points to this marker, i.e., to the top of the stack, then we can deduce $q = p_m$, i.e., that we have reached the synchronization point. Now, if $\mathcal{A}$ is at an a_{k-1}-position and moves to the left afterward, then it is quite possible that we are to the left of p_{k-1}. But we cannot be to the left of p_{k-2} and we know that now the subword $a_{k-1} \cdots a_m$ appears before p_m. Thus we adjust the pointer to a_{k-1}. On the other hand, if we scan a_k, then we know that we are at a position $\geq p_k$ since a_k cannot appear in the interval $(p_{k-1}; p_k)$. Moreover, the subword $a_{k+1} \cdots a_m$ still appears before p_m. Therefore, we adjust the pointer to a_{k+1}, if after reading a_k the automaton moves to the right.

What follows are the technical details of this construction. For $i \in \{1, 2\}$ let $\mathcal{A}_i = (Z_i, \Gamma, \delta_i, x_i^0, F_i)$ with $Z_i = X_i \dot\cup Y_i$. We construct $\mathcal{A} = (Z, \Gamma, \delta, x^0, F)$ with $Z = X \dot\cup Y$ satisfying the following constraints:

- $Z \subseteq (\Gamma^* \times X_1 \times X_2) \cup (\Gamma^* \times Z_1 \times Z_2 \times \mathbb{N} \times \{\mathcal{A}_1, \mathcal{A}_2\})$. The states of the first term in the union are for the synchronous mode. The first component is the stack of markers. Its size is bounded by $|X_1| + |X_2|$. For the asynchronous states, the fourth component is the pointer to the stack of markers and the fifth component specifies the active automaton.

- $Y = Z \cap \big((\Gamma^* \times Y_1 \times Z_2 \times \mathbb{N} \times \{\mathcal{A}_1\}) \cup (\Gamma^* \times Z_1 \times Y_2 \times \mathbb{N} \times \{\mathcal{A}_2\})\big)$ and $X = Z \setminus Y$. So the left-moving states of $\mathcal{A}$ are exactly those where in asynchronous mode the active component is left-moving.

- $x^0 = (\varepsilon, x_1^0, x_2^0)$, i.e., at the beginning $\mathcal{A}$ is in the synchronous mode, the stack of markers is empty, and both automata are in their initial state.

- For recognizing the intersection we set $F = Z \cap (\Gamma^* \times F_1 \times F_2)$. For recognizing the union we set $F = Z \cap \big((\Gamma^* \times F_1 \times X_2) \cup (\Gamma^* \times X_1 \times F_2)\big)$.

Next, we describe the transitions $z \xrightarrow{a} z'$ of $\mathcal{A}$. Let $z = (w, z_1, z_2)$ when $\mathcal{A}$ is in synchronous mode, and $z = (w, z_1, z_2, k, \mathcal{C})$ otherwise. Furthermore, let $z_1 \xrightarrow{a} z_1'$ in $\mathcal{A}_1$ and let $z_2 \xrightarrow{a} z_2'$ in $\mathcal{A}_2$. Suppose that $\mathcal{A}$ is in synchronous mode, i.e., $z \in \Gamma^* \times X_1 \times X_2$. Let

$$w' = \begin{cases} w & \text{if } z_1' = z_1 \text{ and } z_2' = z_2, \\ wa & \text{else,} \end{cases}$$

i.e., push the symbol to the stack if the state of at least one automaton changes. We set

$$(w, z_1, z_2) \xrightarrow{a} \begin{cases} (w', z_1', z_2') & \text{if } z_1' \in X_1 \text{ and } z_2' \in X_2, \\ (w', z_1', z_2, |w'|, \mathcal{A}_1) & \text{if } z_1' \in Y_1, \\ (w', z_1, z_2', |w'|, \mathcal{A}_2) & \text{else,} \end{cases}$$

i.e., we stay in synchronous mode if both automata agree on moving right for the next step, we suspend the second automaton if $\mathcal{A}_1$ wants to move to the left (independent of the direction of $\mathcal{A}_2$), and we suspend the first automaton when it wants to move to the right but $\mathcal{A}_2$ wants to move to the left. Consider now an asynchronous state $z \in \Gamma^* \times Z_1 \times Z_2 \times \mathbb{N} \times \{\mathcal{A}_1, \mathcal{A}_2\}$. First we deal with the special case of reading the last remaining letter of the stack, i.e., a is the last letter of w and the pointer is $|w|$:

$$(w, z_1, z_2, |w|, \mathcal{C}) \xrightarrow{a} \begin{cases} (w, z_1', z_2') & \text{if } z_1' \in X_1 \text{ and } z_2' \in X_2, \\ (w, z_1', z_2, |w|, \mathcal{A}_1) & \text{if } z_1' \in Y_1, \\ (w, z_1, z_2', |w|, \mathcal{A}_2) & \text{else.} \end{cases}$$

The first case is that both automata now agree on the direction of moving to the right and then we change to synchronous mode. If not, the right-moving automaton is suspended. If both are left-moving, then $\mathcal{A}_2$ is suspended. For the other situations we only consider the case of $\mathcal{C} = \mathcal{A}_1$ being active. The case $\mathcal{C} = \mathcal{A}_2$ is similar.

$$(w, z_1, z_2, k, \mathcal{A}_1) \xrightarrow{a} \begin{cases} (w, z_1', z_2, k-1, \mathcal{A}_1) & \text{if } z_1' \in Y_1 \text{ and } a_{k-1} = a, \\ (w, z_1', z_2, k+1, \mathcal{A}_1) & \text{if } z_1' \in X_1 \text{ and } a_k = a, \\ (w, z_1', z_2, k \qquad, \mathcal{A}_1) & \text{else.} \end{cases}$$

Since $\mathcal{A}_1$ is active, we simulate this automaton. The fourth component never gets greater than $|w|$, since scanning the last remaining symbol is treated differently.

One can verify that $\mathcal{A}$ is partially ordered. The main idea is that between any increase and any decrease of the pointer (and also between any decrease and any increase), the state of the active automaton changes.

Let n_1 and n_2 be the length of a maximal chain of states in X_1 and X_2, respectively. The size of the stack in the first component is bounded by $n = n_1 + n_2 - 2$. Therefore, the construction can be realized by an automaton with at most $|\Gamma|^n |Z_1||Z_2|(1 + 2n)$ states. Moreover, the construction is effective. □

Proposition 2 *Every restricted unambiguous ω-monomial is recognized by a deterministic po2-Büchi automaton.*

Proof: Let $L = A_1^* a_1 \cdots A_k^* a_k A_{k+1}^\omega$ be unambiguous such that $\{a_i, \ldots, a_k\} \not\subseteq A_i$ for all $1 \leq i \leq k$. This implies $a_i \notin A_1$ for some $i \geq 1$. Let i be minimal with this property. For each $\alpha \in L$ we consider the factorization $\alpha = u a_i \beta$ with $a_i \notin \mathrm{alph}(u)$. There are two cases:

$$\begin{array}{lll} u \in A_1^* a_1 \cdots A_i^*, & & \beta \in A_{i+1}^* a_{i+1} \cdots A_k^* a_k A_{k+1}^\omega \quad \text{or} \\ u \in A_1^* a_1 \cdots A_j^*, & a_i \in A_j, & \beta \in A_j^* a_j \cdots A_k^* a_k A_{k+1}^\omega \end{array}$$

with $2 \leq j \leq i$. In each case, the expression $B = A_j^* a_j \cdots A_k^* a_k A_{k+1}^\omega$ is unambiguous, because L is. Moreover, it is shorter than the expression for L, and we have $\{a_\ell, \ldots, a_k\} \not\subseteq A_\ell$ for all $j \le \ell \le k$. By induction, B is recognized by some complete deterministic po2-Büchi automaton $\mathcal{B}$.

The unambiguous monomials $A = A_1^* a_1 \cdots A_j^* \cap (\Gamma \setminus \{a_i\})^*$ are accepted by a deterministic po2-Büchi automaton $\mathcal{A}$ operating on finite words [11]. We modify this automaton in order to use the letter a_i instead of $\lhd$ as a right end marker.

From these two automata, we now construct a new automaton $\mathcal{C}$ accepting the ω-language Aa_iB. First, $\mathcal{C}$ checks whether there exists some a_i-position. If so, $\mathcal{C}$ returns to the first letter of the word and starts a simulation of $\mathcal{A}$. If this automaton accepts the word, i.e., $u \in A$, then $\mathcal{C}$ moves its head to the position after the first a_i-position and starts an automaton $\mathcal{B}'$. The automaton $\mathcal{B}'$ simulates $\mathcal{B}$ but ensures that left-scanning for a letter is only successful if this letter is found before the first a_i, i.e., if there is still an a_i on the left. There are at most i cases from above for a word $\alpha \in L$ and therefore, L is a union of finitely many languages of the form Aa_iB recognized by deterministic po2-Büchi automata. Using Proposition 1 the statement follows.

In the following, we describe the construction of $\mathcal{B}'$ from $\mathcal{B}$. The basic idea is that before we make a transition from a left-moving state to a right-moving state, we verify that there is an a_i on the left-hand side. If this verification is successful, then the automaton returns to the position from where it started the verification (this is possible since $\mathcal{B}$ is deterministic), and finally, the automaton makes the original transition of $\mathcal{B}$. Note that in a left-moving state, there must eventually be a transition to a right-moving state, since $\mathcal{B}$ is complete.

For a state z of $\mathcal{B}$, we define $\mathcal{B}_z$ to be the automaton induced by all states which occur in some path from the initial state to z. Consider a transition $y \xrightarrow{b} x$ from $y \in Y$ to $x \in X$ in $\mathcal{B}$. We replace this transition by a sequence of transitions checking that there is an a_i to the left of the current position. If this is successful, we return to the first a_i and give control to $\mathcal{B}_y$ which brings the automaton directly to the position where the procedure started (without any occurrence checking of a_i involved). Finally, we add a transition $\hat{y} \xrightarrow{b} x$ where $\hat{y}$ is the state corresponding to y in $\mathcal{B}_y$. This transition is eventually performed in $\mathcal{B}'$ if the transition $y \xrightarrow{b} x$ is performed in $\mathcal{B}$ and thereafter $\mathcal{B}'$ continues simulating $\mathcal{B}$. $\square$

The following lemma shows the converse of Proposition 2. Our proof reuses techniques from the proof of Lemma 1 which in turn yields a different proof as the one for finite words in [11].

Lemma 3 *Let $\mathcal{A}$ be a deterministic po2-Büchi automaton. Then $L(\mathcal{A})$ is a restricted unambiguous ω-polynomial.*

Proof: Let $\alpha \in L(\mathcal{A})$ and consider the accepting computation of $\mathcal{A}$ on α. For this computation, we define the factorization $\alpha = u_1 a_1 \cdots u_k a_k \beta$ with $a_i \in \Gamma$, $u_i \in \Gamma^*$, and $\beta \in \Gamma^\omega$ such that the positions of the markers a_i are exactly those where a state change happens in the computation. In each traversal of one of the factors u_i and of the suffix β, the letters in these factors correspond to self-loops on the respective states in the accepting computation. Hence $P_\alpha = A_1^* a_1 \cdots A_k^* a_k B^\omega \subseteq L(\mathcal{A})$ for $A_i = \mathrm{alph}(u_i)$ and $B = \mathrm{alph}(\beta)$. Moreover, P_α is unambiguous, since $\mathcal{A}$ is deterministic. When moving from the starting position 1 to some a_i-position with a state change, then there exists a state change at some marker a_j with $j \ge i$ and $a_j \notin \mathrm{alph}(u_i)$, otherwise there would be no marker positions after the factor u_i. Hence, P_α is a restricted unambiguous ω-monomial. It follows

$$L(\mathcal{A}) = \bigcup_{\alpha \in L(\mathcal{A})} P_\alpha$$

and this union is finite, since the degree of each ω-monomial P_α is bounded by the number of states in $\mathcal{A}$ and there are only finitely many ω-monomials of bounded degree. $\square$

Theorem 3 *Let $L \subseteq \Gamma^\omega$. The following assertions are equivalent:*

1. *L is recognized by a deterministic po2-Büchi automaton.*
2. *L is Δ_2-definable.*

Proof: An ω-language L is Δ_2-definable if and only if L is a restricted unambiguous ω-polynomial [4]. The implication "1 $\Rightarrow$ 2" is Lemma 3. For "2 $\Rightarrow$ 1" let L be a restricted unambiguous ω-polynomial, i.e., a finite union of restricted unambiguous ω-monomials. Proposition 2 shows that each of these ω-monomials is recognized by a deterministic po2-Büchi automaton, and Proposition 1 yields an automaton for their union. $\square$

Example 1 The ω-language $L = \{a, b\}^* \, a \, \emptyset^* c \, \{c\}^\omega$ is a restricted unambiguous ω-monomial. By Theorem 3 it is recognized by a deterministic po2-Büchi automaton. Moreover, L is not recognizable by a deterministic partially ordered *one-way* Büchi automaton. Hence, the class of ω-languages recognizable by deterministic partially ordered one-way Büchi automata is a strict subclass of the class recognizable by deterministic po2-Büchi automata. $\Diamond$

5 Complexity Results

In this section, we prove some complexity results for the following decision problems (given po2-Büchi automata $\mathcal{A}$ and $\mathcal{B}$):

- Inclusion: Decide whether $L(\mathcal{A}) \subseteq L(\mathcal{B})$.
- Equivalence: Decide whether $L(\mathcal{A}) = L(\mathcal{B})$.
- Emptiness: Decide whether $L(\mathcal{A}) = \emptyset$.
- Universality: Decide whether $L(\mathcal{A}) = \Gamma^\omega$.

Theorem 4 Emptiness *is* coNP*-complete for both nondeterministic and deterministic po2-Büchi automata.* Inclusion*,* Equivalence *and* Universality *are* coNP*-complete for deterministic po2-Büchi automata; for* Inclusion *this still holds for nondeterministic $\mathcal{A}$.*

Lemma 4 Inclusion *is in* coNP *for nondeterministic $\mathcal{A}$ and deterministic $\mathcal{B}$.*

Proof: Let $Z_\mathcal{A}$ and $Z_\mathcal{B}$ be the states of $\mathcal{A}$ and $\mathcal{B}$, respectively. We have $L(\mathcal{A}) \subseteq L(\mathcal{B})$ if and only if $L(\mathcal{A}) \setminus L(\mathcal{B}) = \emptyset$. By Lemma 2 we see that we can easily compute a deterministic po2-Büchi automaton $\overline{\mathcal{B}}$ such that $L(\overline{\mathcal{B}}) = \Gamma^\omega \setminus L(\mathcal{B})$. If $L(\mathcal{A}) \cap L(\overline{\mathcal{B}}) \neq \emptyset$, then, by Lemma 1, there is a word u with $|u| \leq |Z_\mathcal{A}| + |Z_\mathcal{B}|$ and a letter $a \in \Gamma$ such that $ua^\omega \in L(\mathcal{A}) \cap L(\overline{\mathcal{B}}) = L(\mathcal{A}) \setminus L(\mathcal{B})$. We might have to add one state in each of $\mathcal{A}$ and $\mathcal{B}$ for making them complete. Therefore, in order to test $L(\mathcal{A}) \not\subseteq L(\mathcal{B})$, it suffices to guess a word u of length at most $|Z_\mathcal{A}| + |Z_\mathcal{B}|$ and a letter $a \in \Gamma$ with $ua^\omega \in L(\mathcal{A}) \cap L(\overline{\mathcal{B}})$. Hence, non-inclusion can be verified in NP, i.e., Inclusion is in coNP. $\square$

Lemma 5 EMPTINESS *is* coNP*-hard for deterministic po2-Büchi automata.*

Proof: We shall reduce the complement of satisfiability to EMPTINESS. Let φ be a propositional formula and let $v_1, \ldots, v_m$ be the variables used in φ. We give the construction of a deterministic po2-automaton $\mathcal{A}_\varphi$ over the alphabet $\{0, 1\}$ such that $L(\mathcal{A}_\varphi) = \emptyset$ if and only if there is no valuation satisfying φ. The idea is that we identify the position i of the input with the valuation of variable i for $1 \leq i \leq m$. The rest of the input has no effect on the computation.

Inductively we construct an automaton with the following characteristics: There are two distinguished states x_t and x_f with a loop for both letters 0 and 1. No other right-moving state has a self-loop. The state x_t is eventually entered if φ evaluates to true under the input, else x_f is eventually entered. Moreover, x_t and x_f are only entered by transitions reading $\triangleright$ and x_t is the sole final state. Hence an input is accepted if and only if eventually x_t is entered, i.e., the input defines a satisfying valuation of φ. In case it is rejected it eventually enters x_f.

For variables v_i the automaton $\mathcal{A}_{v_i}$ skips the first $i - 1$ letters of the input, remembers the letter a_i at position i and returns to the beginning of the word. If $a_i = 1$ then $\mathcal{A}$ enters x_t else it enters x_f. For the negation we simply swap the states x_f and x_t. For $\varphi \wedge \psi$ we compose the automata $\mathcal{A}_\varphi$ and $\mathcal{A}_\psi$ in the following way: The states x_t and x_f of $\mathcal{A}_\varphi$ are deleted. Transitions of $\mathcal{A}_\varphi$ leading into state x_f are redirected to the corresponding state x_f of $\mathcal{A}_\psi$; transitions leading into state x_t are redirected to the initial state of $\mathcal{A}_\psi$. With a similar composition, we get an automaton for $\varphi \vee \psi$. All these constructions can be done in polynomial time in the size of the input formula. $\square$

Proof (Theorem 4): Taking $L(\mathcal{B}) = \emptyset$, Lemma 4 yields that EMPTINESS is in coNP for nondeterministic po2-Büchi automata. Lemma 5 shows that EMPTINESS is coNP-hard even for deterministic po2-Büchi automata.

From INCLUSION $\in$ coNP for deterministic po2-Büchi automata, we immediately get that EQUIVALENCE and UNIVERSALITY are in coNP. Moreover, the trivial reductions from EMPTINESS to UNIVERSALITY to EQUIVALENCE and from EMPTINESS to INCLUSION show that all problems under consideration are coNP-hard for deterministic po2-Büchi automata.

For nondeterministic $\mathcal{A}$ and deterministic $\mathcal{B}$, Lemma 4 shows that INCLUSION is in coNP and of course it is coNP-hard since this is already true if both automata are deterministic. $\square$

6 Conclusion

In this paper, we introduced partially ordered two-way Büchi automata (po2-Büchi automata). The nondeterministic variant corresponds to the fragment Σ_2 of first-order logic, whereas the deterministic variant is characterized by the fragment $\Delta_2 = \Sigma_2 \cap \Pi_2$. The characterization of nondeterministic automata uses similar techniques as for finite words [11]. For deterministic automata, our proof uses new techniques and it relies on a novel language description of Δ_2 involving restricted unambiguous ω-polynomials [4]. As an intermediate step it turns out that the class of ω-languages recognized by deterministic po2-Büchi automata is effectively closed under Boolean operations.

The complexity of the EMPTINESS problem for both deterministic and nondeterministic po2-Büchi automata is coNP-complete. For deterministic po2-Büchi automata the decision problems INCLUSION, EQUIVALENCE, and UNIVERSALITY are coNP-complete. To date, no non-trivial upper bounds are known for these decision problems over nondeterministic automata. Moreover, the complexity of the decision problems for general two-way Büchi automata as well as the succinctness of this model have not yet been considered in the literature.

Considering fragments with successor would be a natural extension of our results. An automaton model for the fragment Δ_2 with successor over finite words has been given by Lodaya, Pandya, and Shah [8] in terms of *deterministic partially ordered two-way automata with look-around*. We conjecture that extending such automata with a Büchi acceptance condition yields a characterization of Δ_2 with successor over infinite words.

References

[1] Ch. Baier and J.-P. Katoen. *Principles of Model Checking*. The MIT Press, 2008.

[2] J.R. Büchi. On a decision method in restricted second-order arithmetic. In *Proc. Int. Congr. for Logic, Methodology, and Philosophy of Science*, pages 1–11. Stanford Univ. Press, 1962.

[3] E.M. Clarke Jr., O. Grumberg, and D.A. Peled. *Model Checking*. The MIT Press, 1999.

[4] V. Diekert and M. Kufleitner. Fragments of first-order logic over infinite words. *STACS 2009*, volume 09001 of *Dagstuhl Seminar Proceedings*, pages 325–336, 2009.

[5] Ch.A. Kapoutsis. Removing bidirectionality from nondeterministic finite automata. *MFCS 2005*, volume 3618 of *LNCS*, pages 544–555. Springer, 2005.

[6] O. Kupferman, N. Piterman, and M.Y. Vardi. Extended temporal logic revisited. *CONCUR 2001*, pages 519–535, 2001. Springer.

[7] K. Lodaya, P.K. Pandya, and S.S. Shah. Marking the chops: an unambiguous temporal logic. *IFIP TCS*, 273:461–476, 2008.

[8] K. Lodaya, P.K. Pandya, and S.S. Shah. Around dot depth two. *DLT 2010*, volume 6224 of *LNCS*, pages 305–316, 2010.

[9] J.-P. Pécuchet. Automates boustrophédon et mots infinis. *Theoretical Computer Science*, 35:115–122, 1985.

[10] J.-É. Pin and P. Weil. Polynomial closure and unambiguous product. *Theory of Computing Systems*, 30(4):383–422, 1997.

[11] Th. Schwentick, D. Thérien, and H. Vollmer. Partially-ordered two-way automata: A new characterization of DA. *DLT 2001*, volume 2295 of *LNCS*, pages 239–250. Springer, 2001.

[12] A. P. Sistla, M. Y. Vardi, and P. L. Wolper. The complementation problem for Büchi automata with applications to temporal logic. *Theoretical Computer Science*, 49(2-3):217–237, 1987.

[13] D. Thérien and Th. Wilke. Over words, two variables are as powerful as one quantifier alternation. *STOC 1998*, pages 234–240, 1998.

[14] W. Thomas. Classifying regular events in symbolic logic. *Journal of Computer and System Sciences*, 25:360–376, 1982.

[15] M.Y. Vardi. Reasoning about the past with two-way automata. *ICALP 1998*, pages 628–641, 1998. Springer.