The Quest for a Tight Translation of Büchi to co-Büchi Automata

Udi Boker[⋆] and Orna Kupferman

School of Computer Science and Engineering, Hebrew University, Israel

Abstract. The Büchi acceptance condition specifies a set α of states, and a run is accepting if it visits α infinitely often. The co-Büchi acceptance condition is dual, thus a run is accepting if it visits α only finitely often. Nondeterministic Büchi automata over words (NBWs) are strictly more expressive than nondeterministic co-Büchi automata over words (NCWs). The problem of the blow-up involved in the translation (when possible) of an NBW to an NCW has been open for several decades. Until recently, the best known upper bound was $2^{O(n \log n)}$ and the best lower bound was n. We describe the quest to the tight $2^{\Theta(n)}$ bound.

Key words: Büchi automata, co-Büchi automata, nondeterminism, automata translation

1 Introduction

Finite *automata on infinite objects* were first introduced in the 60's, and were the key to the solution of several fundamental decision problems in mathematics and logic [5, 15, 20]. Today, automata on infinite objects are used for specification verification, and synthesis of nonterminating systems. The automata-theoretic approach to verification views questions about systems and their specifications as questions about languages, and reduces them to automata-theoretic problems like containment and emptiness [13, 26]. Recent industrial-strength property-specification languages such as Sugar, ForSpec, and the recent standard PSL 1.01 include regular expressions and/or automata, making specification and verification tools that are based on automata even more essential and popular [1].

Early automata-based algorithms aimed at showing decidability. The application of automata theory in practice has led to extensive research on the complexity of problems and constructions involving automata [6, 19, 22, 24, 25, 27]. For many problems and constructions, our community was able to come up with satisfactory solutions, in the sense that the upper bound (the complexity of the best algorithm or the blow-up in the best known construction) coincides with the lower bound (the complexity class in which the problem is hard, or the blow-up that is known to be unavoidable). For some problems and constructions, however, the gap between the upper bound and the lower bound is significant. This situation is especially frustrating, as it implies that not only something is

⋆ Supported in part by a Lady Davis postdoctoral fellowship.

missing in our understanding of automata on infinite objects, but also that we may be using algorithms that can be significantly improved.

One such fundamental and longstanding open problem is the translation, when possible, of a nondeterministic Büchi word automaton (NBW) to an equivalent nondeterministic co-Büchi word automaton (NCW).[1] NCWs are less expressive than NBWs. For example, the language $\{w : w$ has infinitely many a's$\}$ over the alphabet $\{a, b\}$ cannot be recognized by an NCW. In fact, NCWs are not more expressive than deterministic co-Büchi automata (DCWs).[2] Hence, since deterministic Büchi automata (DBWs) are dual to DCWs, a language can be recognized by an NCW iff its complement can be recognized by a DBW.

The best translation of an NBW to an NCW (when possible) that was known until recently goes as follows. Consider an NBW $\mathcal{A}$ that has an equivalent NCW. First, co-determinize $\mathcal{A}$ and obtain a deterministic Rabin automaton (DRW) $\tilde{\mathcal{A}}$ for the complement language. By [8], DRWs are *Büchi type*. That is, if a DRW has an equivalent DBW, then the DRW has an equivalent DBW on the same structure. Since $\mathcal{A}$ can be recognized by an NCW, its complement $\tilde{\mathcal{A}}$ has an equivalent DBW, so there is a DBW $\tilde{\mathcal{B}}$ that complements $\mathcal{A}$ and has the same structure as $\tilde{\mathcal{A}}$. By viewing $\tilde{\mathcal{B}}$ as a DCW, one gets a deterministic co-Büchi automaton (DCW) equivalent to $\mathcal{A}$. The co-determinization step involves a super exponential blow-up in the number of states [22]: starting with an NBW with n states, we end up with a DCW with $2^{O(n \log n)}$ states. Beyond the super exponential blow-up, the state space that results from Safra's determinization and co-determinization constructions is awfully complex and is not amenable to optimizations and a symbolic implementation. Also, going through a deterministic automaton requires the introduction of acceptance conditions that are more complex than the Büchi and co-Büchi acceptance conditions. Piterman's construction [18] simplifies the situation only slightly; while the Rabin condition can be replaced by parity, the complication and the $2^{O(n \log n)}$ complexity of Safra's construction are still there. Note that eventhough the problem is of translating an NBW to an NCW, and thus it does not require us to end up in a deterministic automaton, the above procedure actually does result in a DCW. Thus, it is not known how to take advantage of the allowed nondeterminism, and how to keep the translation within the convenient scope of the Büchi and the co-Büchi acceptance conditions.

The $2^{O(n \log n)}$ upper bound is particularly annoying, as no non-trivial lower bound was known. In fact, there was even no counterexample to the conjecture that NBWs are *co-Büchi type*. That is, to a conjecture that if an NBW has an equivalent NCW, then the NBW has an equivalent NCW on the same structure.

[1] In *Büchi* automata, some of the states are designated as accepting states, and a run is accepting iff it visits states from the accepting set infinitely often [5]. Dually, in *co-Büchi* automata, a run is accepting iff it visits the set of accepting states only finitely often.

[2] When applied to universal Büchi automata, the translation in [16], of alternating Büchi automata into NBW, results in DBW. By dualizing it, one gets a translation of NCW to DCW.

The main challenge in proving a non-trivial lower bound for the translation of NBW to NCW is the expressiveness superiority of NBW with respect to NCW. Indeed, a family of languages that is a candidate for proving a lower bound for this translation has to strike a delicate balance: the languages have to somehow take advantage of the Büchi acceptance condition, and still be recognizable by a co-Büchi automaton.[3] In particular, it is not clear how to use the main feature of the Büchi condition, namely its ability to easily track infinitely many occurrences of an event, as a co-Büchi automaton cannot recognize languages that are based on such a tracking.

Beyond the theoretical challenge in tightening the gaps, and the fact they are related to other gaps in our knowledge [9], the translation of NBW to NCW has immediate important applications in formal methods. The premier example in this class is of symbolic LTL model checking. Evaluating specifications in AFMC can be done with linearly many symbolic steps. In contrast, direct LTL model checking reduces to a search for bad-cycles, whose symbolic implementation involves nested fixed-points, and is typically [4] quadratic [21]. It is shown in [12] that given an LTL formula ψ, there is an alternation-free μ-calculus (AFMC) formula equivalent to $\forall\psi$ iff ψ can be recognized by a DBW. Alternatively, an NCW for $\neg\psi$ can be linearly translated to an AFMC formula equivalent to $\exists\neg\psi$, which can be negated to a formula equivalent to $\forall\psi$. Thus, an improvement of the translation of NBW to NCW would immediately imply an improvement of the translation of LTL to AFMC.

We describe the quest to a $2^{\Theta(n)}$ tight bound for the translation. In the upper-bound front, we describe the construction in [3], which translates an NBW $\mathcal{B}$ to an NCW $\mathcal{C}$ whose underlying structure is the product of $\mathcal{B}$ with its subset construction. Thus, given an NBW $\mathcal{B}$ with n states, the translation yields an equivalent NCW with $n2^n$ states, and it has a simple symbolic implementation [17]. In the lower-bound front, we first describe the counterexample given in [11] to the NCW-typeness of NBW. We then describe the "circumventing counting" idea, according to which the ability of NBWs to easily track infinitely many occurrences of an event makes them more succinct than NCWs. The idea is to consider a family of languages $L_1, L_2, L_3, \ldots$ in which an NCW for L_k has to count to some bound that depends on k, whereas an NBW can count instead to infinity. In the first application of the idea, the NBW for the language L_k checks that an event P occurs infinitely often. The language L_k is still NCW-recognizable as other components of L_k make it possible to check instead that P has at least k occurrences. An NCW for L_k can then count occurrences of P, but it needs $O(k)$ more states for this [2]. In order to achieve a super-linear

[3] A general technique for proving lower bounds on the size of automata on infinite words is suggested in [28]. The technique is based on *full automata*, in which a word accepted by the automaton induces a language. The fact NCWs are less expressive than NBWs is a killer for the technique, as full automata cannot be translated to NCWs.

[4] Better algorithms have been suggested [7, 21], but it turns out that algorithms based on nested fixed-points perform better in practice.

succinctness, we enhance the idea as follows. The NBW for the language L_k still checks that an event P occurs infinitely often. Now, however, in order for L_k to be NCW-recognizable, other components of L_k make it possible to check instead that P repeats at least once in every interval of some bounded length $f(k)$. Thus, while the NBW can detect infinitely many occurrences of P with 2 states, the NCW has to devote $O(f(k))$ states for the counting. We first use ideas from number theory in order to make $f(k)$ quadratic in k, and then use binary encoding in order to make $f(k)$ exponential in k [3].

2 Preliminaries

Given an alphabet Σ, a *word* over Σ is a (possibly infinite) sequence $w = w_1 \cdot w_2 \cdots$ of letters in Σ. For two words, x and y, we use $x \preceq y$ to indicate that x is a prefix of y and $x \prec y$ to indicate that x is a strict prefix of y. An *automaton* is a tuple $\mathcal{A} = \langle \Sigma, Q, \delta, Q_0, \alpha \rangle$, where Σ is the input alphabet, Q is a finite set of states, $\delta : Q \times \Sigma \to 2^Q$ is a transition function, $Q_0 \subseteq Q$ is a set of initial states, and $\alpha \subseteq Q$ is an acceptance condition. We define several acceptance conditions below. Intuitively, $\delta(q, \sigma)$ is the set of states that $\mathcal{A}$ may move into when it is in the state q and it reads the letter σ. The automaton $\mathcal{A}$ may have several initial states and the transition function may specify many possible transitions for each state and letter, and hence we say that $\mathcal{A}$ is *nondeterministic*. In the case where $|Q_0| = 1$ and for every $q \in Q$ and $\sigma \in \Sigma$, we have that $|\delta(q, \sigma)| \leq 1$, we say that $\mathcal{A}$ is *deterministic*. The transition function extends to sets of states and to finite words in the expected way, thus $\delta(S, x)$ is the set of states that $\mathcal{A}$ may move into when it is in a state in S and it reads the finite word x. Formally, $\delta(S, \epsilon) = S$ and $\delta(S, w \cdot \sigma) = \bigcup_{q \in \delta(S, w)} \delta(q, \sigma)$. We abbreviate $\delta(Q_0, x)$ by $\delta(x)$, thus $\delta(x)$ is the set of states that $\mathcal{A}$ may visit after reading x. For an automaton $\mathcal{A}$ and a state a of $\mathcal{A}$, we denote by $\mathcal{A}^a$ the automaton that is identical to $\mathcal{A}$, except for having $\{a\}$ as its set of initial states.

A run $r = r_0, r_1, \cdots$ of $\mathcal{A}$ on $w = w_1 \cdot w_2 \cdots \in \Sigma^\omega$ is an infinite sequence of states such that $r_0 \in Q_0$, and for every $i \geq 0$, we have that $r_{i+1} \in \delta(r_i, w_{i+1})$. Note that while a deterministic automaton has at most a single run on an input word, a nondeterministic automaton may have several runs on an input word. We sometimes refer to r as a word in Q^ω or as a function from the set of prefixes of w to the states of $\mathcal{A}$. Accordingly, we use $r(x)$ to denote the state that r visits after reading the prefix x.

Acceptance is defined with respect to the set $\mathit{inf}(r)$ of states that the run r visits infinitely often. Formally, $\mathit{inf}(r) = \{q \in Q \mid \text{ for infinitely many } i \in \mathbb{N}, \text{ we have } r_i = q\}$. As Q is finite, it is guaranteed that $\mathit{inf}(r) \neq \emptyset$. The run r is *accepting* iff the set $\mathit{inf}(r)$ satisfies the acceptance condition α. We consider here the *Büchi* and the *co-Büchi* acceptance conditions. A set $S \subseteq Q$ satisfies a Büchi acceptance condition $\alpha \subseteq Q$ iff $S \cap \alpha \neq \emptyset$; whereas S satisfies a *co-Büchi* acceptance condition $\alpha \subseteq Q$ iff $S \subseteq \alpha$. Note that the definition of co-Büchi we use is less standard than the $S \cap \alpha = \emptyset$ definition; clearly, $S \subseteq \alpha$ iff $S \cap (Q \setminus \alpha) = \emptyset$, thus the definition is equivalent. We chose to go with the $S \subseteq \alpha$ variant as it better conveys the intuition that, as with the Büchi condition, a visit in α is a "good event". An automaton accepts a word iff it has an accepting run on it. The language of an automaton $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of words that $\mathcal{A}$ accepts. We also say that $\mathcal{A}$ *recognizes* the language $L(\mathcal{A})$. For two automata $\mathcal{A}$ and $\mathcal{A}'$, we say that $\mathcal{A}$ and $\mathcal{A}'$ are *equivalent* if $L(\mathcal{A}) = L(\mathcal{A}')$.

We denote the different classes of automata by three letter acronyms in $\{\text{D}, \text{N}\} \times \{\text{B}, \text{C}\} \times \{\text{W}\}$. The first letter stands for the branching mode of the automaton (deterministic or nondeterministic); the second letter stands for the acceptance-condition type (Büchi, or co-Büchi); the third letter indicates that the automaton runs on words. We say that a language L is in a class γ if L is *γ-recognizable*, that is, L can be recognized by an automaton in the class γ.

Different classes of automata have different expressive power. In particular, while NBWs recognize all ω-regular language [15], DBWs are strictly less expressive than NBWs, and so are DCWs [14]. In fact, a language L is in DBW iff its complement is in DCW. Indeed, by viewing a DBW as a DCW, we get an automaton for the complementing language, and vice versa. The expressiveness superiority of the nondeterministic model over the deterministic one does not apply to the co-Büchi acceptance condition. There, every NCW has an equivalent DCW [16].

3 Upper Bound

In this section we present the upper-bound proof from [3] for the translation of NBW to NCW (when possible).[5] The proof is constructive: given an NBW $\mathcal{B}$ with k states whose language is NCW-recognizable, we construct an equivalent NCW $\mathcal{C}$ with at most $k2^k$ states. The underlying structure of $\mathcal{C}$ is very simple: it runs $\mathcal{B}$ in parallel to its subset construction. We refer to the construction as the *augmented subset construction*, and we describe the rationale behind it below.

Consider an NBW $\mathcal{B}$ with set $\alpha_{\mathcal{B}}$ of accepting states. The subset construction of $\mathcal{B}$ maintains, in each state, all the possible states that $\mathcal{B}$ can be at. Thus, the subset construction gives us full information about $\mathcal{B}$'s *potential* to visit $\alpha_{\mathcal{B}}$ in the future. However, the subset construction loses information about the past. In particular, we cannot know whether fulfilling $\mathcal{B}$'s potential requires us to give up past visits in $\alpha_{\mathcal{B}}$. For that reason, the subset construction is adequate for determinizing automata on finite words, but not good enough for determinizing ω-automata. A naive try to determinize $\mathcal{B}$ could be to build its subset construction and define the acceptance set as all the states for which $\mathcal{B}$ has the potential to be in $\alpha_{\mathcal{B}}$. The problem is that a word might infinitely often gain this potential via different runs. Were we only able to guarantee that the run of the subset construction follows a single run of the original automaton, we would have ensured a correct construction. Well, this is exactly what the augmented subset construction does!

[5] For readers who skipped the preliminaries, let us mention that we work here with a less standard definition of the co-Büchi condition, where a run r satisfies a co-Büchi condition α iff $\mathit{inf}(r) \subseteq \alpha$.

Once the above intuition is understood, there is still a question of how to define the acceptance condition on top of the augmented subset construction. Since we target for an NCW, we cannot check for infiniteness. However, the premise that the NBW is in DCW guarantees that a word is accepted iff there is a run of the augmented subset construction on it that remains in "potentially good states" from some position. We explain and formalize this property below.

We start with a property relating states of a DCW (in fact, any deterministic automaton) that are reachable via words that lead to the same state in the subset construction of an equivalent NBW.

Lemma 1. *Consider an NBW $\mathcal{B}$ with a transition function $\delta_{\mathcal{B}}$ and a DCW $\mathcal{D}$ with a transition function $\delta_{\mathcal{D}}$ such that $L(\mathcal{B}) = L(\mathcal{D})$. Let d_0 and d_1 be states of $\mathcal{D}$ such that there are two finite words x_0 and x_1 such that $\delta_{\mathcal{D}}(x_0) = d_0$, $\delta_{\mathcal{D}}(x_1) = d_1$, and $\delta_{\mathcal{B}}(x_0) = \delta_{\mathcal{B}}(x_1)$. Then, $L(\mathcal{D}^{d_0}) = L(\mathcal{D}^{d_1})$.*

For automata on finite words, if two states of the automaton have the same language, they can be merged without changing the language of the automaton. While this is not the case for automata on infinite words, the lemma below enables us to do take advantage of such states.

Lemma 2. *Consider a DCW $\mathcal{D} = \langle \Sigma, D, \delta, D_0, \alpha \rangle$. Let d_0 and d_1 be states in D such that $L(\mathcal{D}^{d_0}) = L(\mathcal{D}^{d_1})$. For all finite words u and v, if $\delta(d_0, u) = d_0$ and $\delta(d_1, v) = d_1$ then for all words $w \in (u + v)^*$ and states $d' \in \delta(d_0, w) \cup \delta(d_1, w)$, we have $L(\mathcal{D}^{d'}) = L(\mathcal{D}^{d_0})$.*

Our next observation is the key to the definition of the acceptance condition in the augmented subset construction. Intuitively, it shows that if an NCW language L is indifferent to a prefix in $(u + v)^*$, and L contains the language $(v^* \cdot u^+)^\omega$, then L must also contain the word v^ω.

Lemma 3. *Consider a co-Büchi recognizable language L. For all finite words u and v, if for every finite word $x \in (u + v)^*$ and infinite word w we have that $w \in L$ iff $x \cdot w \in L$, and $(v^* \cdot u^+)^\omega \subseteq L$, then $v^\omega \in L$.*

By considering the language of a specific state of the DCW, Lemma 3 implies the following.

Corollary 1. *Let $\mathcal{D} = \langle \Sigma, D, \delta, D_0, \alpha \rangle$ be a DCW. Consider a state $d \in D$. For all nonempty finite words v and u, if for all words $w \in (v + u)^*$ and states $d' \in \delta(d, w)$, we have $L(\mathcal{D}^{d'}) = L(\mathcal{D}^d)$, and $(v^* \cdot u^+)^\omega \subseteq L(\mathcal{D}^d)$, then $v^\omega \in L(\mathcal{D}^d)$.*

We can now present the construction together with its acceptance condition.

Theorem 1 ([3]). *For every NBW $\mathcal{B}$ with k states that is co-Büchi recognizable there is an equivalent NCW $\mathcal{C}$ with at most $k2^k$ states.*

Proof. Let $\mathcal{B} = \langle \Sigma, B, \delta_{\mathcal{B}}, B_0, \alpha_{\mathcal{B}} \rangle$. We define the NCW $\mathcal{C} = \langle \Sigma, C, \delta_{\mathcal{C}}, C_0, \alpha_{\mathcal{C}} \rangle$ on top of the product of $\mathcal{B}$ with its subset construction. Formally, we have the following.

– $C = B \times 2^B$. That is, the states of $\mathcal{C}$ are all the pairs $\langle b, E \rangle$ where $b \in B$ and $E \subseteq B$.
– For all $\langle b, E \rangle \in C$ and $\sigma \in \Sigma$, we have $\delta_{\mathcal{C}}(\langle b, E \rangle, \sigma) = \delta_{\mathcal{B}}(b, \sigma) \times \{\delta_{\mathcal{B}}(E, \sigma)\}$. That is, $\mathcal{C}$ nondeterministically follows $\mathcal{B}$ on its B-components and deterministically follows the subset construction of $\mathcal{B}$ on its 2^B-component.
– $C_0 = B_0 \times \{B_0\}$.
– A state is a member of $\alpha_{\mathcal{C}}$ if it is reachable from itself along a path whose projection on B visits $\alpha_{\mathcal{B}}$. Formally, $\langle b, E \rangle \in \alpha_{\mathcal{C}}$ if there is a state $\langle b', E' \rangle \in \alpha_{\mathcal{B}} \times 2^B$ and finite words y_1 and y_2 such that $\langle b', E' \rangle \in \delta_{\mathcal{C}}(\langle b, E \rangle, y_1)$ and $\langle b, E \rangle \in \delta_{\mathcal{C}}(\langle b', E' \rangle, y_2)$. We refer to $y_1 \cdot y_2$ as the witness for $\langle b, E \rangle$. Note that all the states in $\alpha_{\mathcal{B}} \times 2^B$ are members of $\alpha_{\mathcal{C}}$ with an empty witness.

We prove the equivalence of $\mathcal{B}$ and $\mathcal{C}$. Note that the 2^B-component of C proceeds in a deterministic manner. Therefore, each run r of $\mathcal{B}$ induces a single run of $\mathcal{C}$ (the run in which the B-component follows r). Likewise, each run r' of $\mathcal{C}$ induces a single run of $\mathcal{B}$, obtained by projecting r' on its B-component.

We first prove that $L(\mathcal{B}) \subseteq L(\mathcal{C})$. Consider a word $w \in L(\mathcal{B})$. Let r be an accepting run of $\mathcal{B}$ on w. We prove that the run r' induced by r is accepting. Consider a state $\langle b, E \rangle \in \mathit{inf}(r')$. We prove that $\langle b, E \rangle \in \alpha_C$. Since $\langle b, E \rangle \in \mathit{inf}(r')$, then $b \in \mathit{inf}(r)$. Thus, there are three prefixes x, $x \cdot y_1$, and $x \cdot y_1 \cdot y_2$ of w such that $r'(x) = r'(x \cdot y_1 \cdot y_2) = \langle b, E \rangle$ and $r'(x \cdot y_1) \in \alpha_{\mathcal{B}} \times 2^B$. Therefore, $y_1 \cdot y_2$ witnesses that $\langle b, E \rangle$ is in $\alpha_{\mathcal{C}}$. Hence, $\mathit{inf}(r) \subseteq \alpha_C$, and we are done.

We now prove that $L(\mathcal{C}) \subseteq L(\mathcal{B})$. Consider a word $w \in L(\mathcal{C})$. Let r' be an accepting run of $\mathcal{C}$ on w, let $\langle b, E \rangle$ be a state in $\mathit{inf}(r')$, and let x be a prefix of w such that $r'(x) = \langle b, E \rangle$. Since r' is accepting, $\mathit{inf}(r') \subseteq \alpha_{\mathcal{C}}$, so $\langle b, E \rangle \in \alpha_{\mathcal{C}}$. Let z be a witness for the membership of $\langle b, E \rangle$ in α_C. By the definition of a witness, $\delta_{\mathcal{B}}(E, z) = E$ and there is a run of $\mathcal{B}^b$ on z that visits $\alpha_{\mathcal{B}}$ and goes back to b. If $z = \epsilon$, then $b \in \alpha_B$, the run of $\mathcal{B}$ induced by r' is accepting, and we are done. Otherwise, $x \cdot z^\omega \in L(\mathcal{B})$, and we proceed as follows.

Recall that the language of $\mathcal{B}$ is NCW-recognizable. Let $\mathcal{D} = \langle \Sigma, D, \delta_D, D_0, \alpha_{\mathcal{D}} \rangle$ be a DCW equivalent to $\mathcal{B}$. Since $L(\mathcal{B}) = L(\mathcal{D})$ and $x \cdot z^\omega \in L(\mathcal{B})$, it follows that the run ρ of $\mathcal{D}$ on $x \cdot z^\omega$ is accepting. Since D is finite, there are two indices i_1 and i_2 such that $i_1 < i_2$, $\rho(x \cdot z^{i_1}) = \rho(x \cdot z^{i_2})$, and for all prefixes y of $x \cdot z^\omega$ such that $x \cdot z^{i_1} \preceq y$, we have $\rho(y) \in \alpha_{\mathcal{D}}$. Let $d_1 = \rho(x \cdot z^{i_1})$.

Consider the run η of $\mathcal{D}$ on w. Since r' visits $\langle b, E \rangle$ infinitely often and D is finite, there must be a state $d_0 \in D$ and infinitely many prefixes $p_1, p_2, \ldots$ of w such that for all $i \geq 1$, we have $r'(p_i) = \langle b, E \rangle$ and $\eta(p_i) = d_0$.

We claim that the states d_0 and d_1 satisfy the conditions of Lemma 1 with x_0 being p_1 and x_1 being $x \cdot z^{i_1}$. Indeed, $\delta_{\mathcal{D}}(p_1) = d_0$, $\delta_{\mathcal{D}}(x \cdot z^{i_1}) = d_1$, and $\delta_{\mathcal{B}}(p_1) = \delta_{\mathcal{B}}(x \cdot z^{i_1}) = E$. For the latter equivalence, recall that $\delta_{\mathcal{B}}(x) = E$ and $\delta_{\mathcal{B}}(E, z) = E$. Hence, by Lemma 1, we have $L(\mathcal{D}^{d_0}) = L(\mathcal{D}^{d_1})$.

Recall the sequence of prefixes $p_1, p_2, \ldots$. For all $i \geq 1$, let $p_{i+1} = p_i \cdot t_i$. We now claim that for all $i \geq 1$, the state d_0 satisfies the conditions of Corollary 1 with u being $z^{i_2 - i_1}$ and v being t_i. The first condition is satisfied by Lemma 2. For the second condition, consider a word $w' \in (v^* \cdot u^+)^\omega$. We prove that $w' \in L(\mathcal{D}^{d_0})$. Recall that there is a run of $\mathcal{B}^b$ on v that goes back to b and there is a run of $\mathcal{B}^b$ on u that visits $\alpha_{\mathcal{B}}$ and goes back to b. Recall also that for the word p_1, we have that $r'(p_1) = \langle b, E\rangle$ and $\eta(p_1) = d_0$. Hence, $p_1 \cdot w' \in L(\mathcal{B})$. Since $L(\mathcal{B}) = L(\mathcal{D})$, we have that $p_1 \cdot w' \in L(\mathcal{B})$. Therefore, $w' \in L(\mathcal{D}^{d_0})$.

Thus, by Corollary 1, for all $i \geq 1$ we have that $t_i^\omega \in L(\mathcal{D}^{d_0})$. Since $\delta_{\mathcal{D}}(d_0, t_i) = d_0$, it follows that all the states that $\mathcal{D}$ visits when it reads t_i from d_0 are in $\alpha_{\mathcal{D}}$. Note that $w = p_1 \cdot t_1 \cdot t_2 \cdots$. Hence, since $\delta_{\mathcal{D}}(p_1) = d_0$, the run of $\mathcal{D}$ on w is accepting, thus $w \in L(\mathcal{D})$. Since $L(\mathcal{D}) = L(\mathcal{B})$, it follows that $w \in L(\mathcal{B})$, and we are done. □

4 Lower Bound

In this section we describe the "circumventing counting" idea and how it has led to a matching lower bound. In the deterministic setting, DBWs are co-Büchi type. Thus, if a DBW $\mathcal{A}$ is DCW-recognizable, then there is a DCW equivalent to $\mathcal{A}$ that agrees with $\mathcal{A}$ on its structure (that is, one only has to modify the acceptance condition). The conjecture that NBW are also co-Büchi type was refuted only in [11]:

Theorem 2 ([11]). *NBWs are not co-Büchi type.*

Proof. Consider the NBW $\mathcal{A}$ described in Fig. 1. The NBW recognizes the language $a^* \cdot b \cdot (a + b)^*$ (at least one b). This language is in NCW, yet it is easy to see that there is no NCW recognizing L on the same structure. □

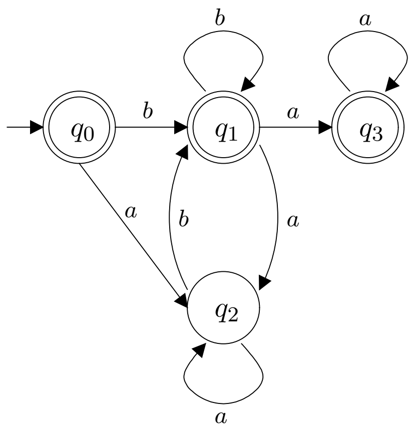

Fig. 1. NBWs for $a^* \cdot b \cdot (a + b)^*$.

The result in [11] shows that there are NBWs that are NCW-recognizable and yet an NCW for them requires a structure that is different from the one of the given NBW. It does not show, however, that the NCW needs to have more states. In particular, the language of the NBW in Fig. 1 can be recognized by an NCW with two states.

4.1 A Linear Lower Bound

The first non-trivial lower bound for the NBW to NCW translation was described in [2]. It is based on an idea to which we refer as the "circumventing counting" idea. We describe the idea along with the family of languages used in [2].

Let $\Sigma = \{a, b\}$. For every $k \geq 1$, we define a language L_k as follows:

$$L_k = \{w \in \Sigma^\omega \mid \text{both } a \text{ and } b \text{ appear at least } k \text{ times in } w\}.$$

Since an automaton recognizing L_k must accept every word in which there are at least k a's and k b's, regardless of how the letters are ordered, it may appear as if the automaton must have two k-counters operating in parallel, which requires $O(k^2)$ states. This would indeed be the case if a and b had not been the only letters in Σ, of if the automaton had been deterministic or on finite words. However, since we are interested in nondeterministic automata on infinite words, and a and b are the only letters in Σ, we can do much better. Since Σ contains only the letters a and b, one of these letters must appear infinitely often in every word in Σ^ω. Hence, $w \in L_k$ iff w has at least k b's and infinitely many a's, or at least k a's and infinitely many b's. An NBW can guess which of the two cases above holds, and proceed to validate its guess (if w has infinitely many a's as well as b's, both guesses would succeed). The validation of each of these guesses requires only one k-counter, and a gadget with two states for verifying that there are infinitely many occurrences of the guessed letter. Implementing this idea results in the NBW with $2k + 1$ states appearing in Fig. 2.

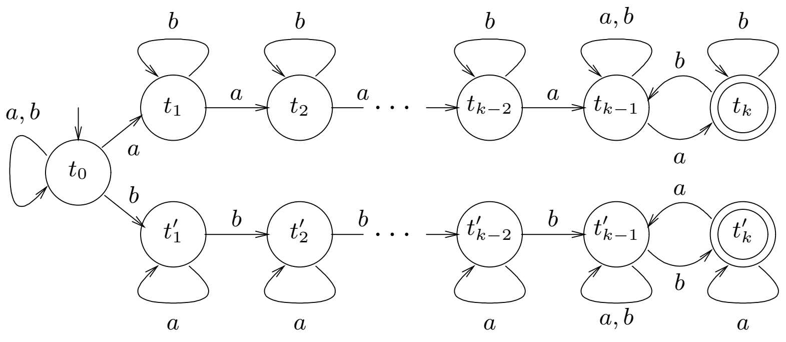

Fig. 2. An NBW for L_k with $2k + 1$ states.

The reason we were able to come up with a small NBW for L_k is that NBWs can abstract precise counting by "counting to infinity" with two states. The fact that NCWs do not share this ability [14] is what ultimately allows us to prove that NBW are more succinct than NCW. As it turns out, however, even an NCW for L_k can do much better than maintaining two k-counters with $O(k^2)$ states. To see how, note that a word w is in L_k iff w has at least k b's *after* the first k a's (this characterizes words in L_k with infinitely many b's), or a finite number of b's that is not smaller than k (this characterizes words in L_k with finitely many b's). Obviously the roles of a and b can also be reversed. Implementing this idea results in the NCW with $3k+1$ states described in Fig. 3. As detailed in [2], up to one state this is indeed the best one can do. Thus, the family of languages $L_1, L_2, \ldots$ implies that translating an NBW with $2k+1$ states may result in an NCW with at least $3k$ states, hence the non-trivial, but still linear, lower bound.

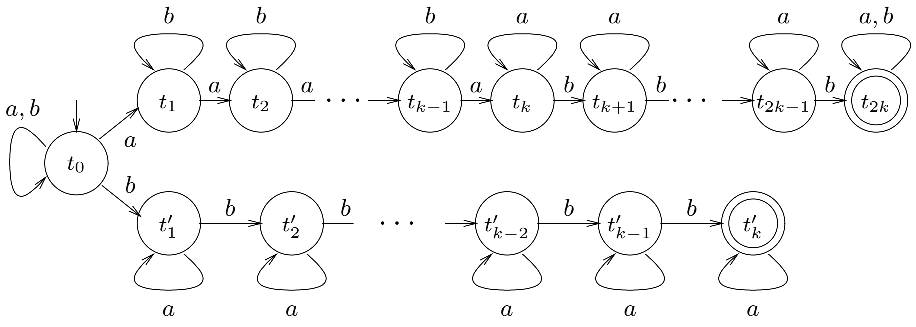

Fig. 3. An NCW for L_k with $3k+1$ states.

4.2 A Quadratic Lower Bound

In this section we enhance the circumventing-counting idea, by letting the NCW count distances between occurrences rather than number of occurrences. We demonstrate how the enhancement can lead to a quadratic lower bound. We first need some results from number theory. For $k \in \mathbb{N}$, let $S_k \subseteq \mathbb{N}$ be the set of all positive integers that can be written as $ik+j(k+1)$, for $i, j \in \mathbb{N}$. Thus, $S_k = \{ik+j(k+1) : i, j \in \mathbb{N}, i+j > 0\}$. For example, $S_4 = \{4, 5, 8, 9, 10, 12, 13, 14, 15, \ldots\}$. The following is a well-known property of S_k, which we prove below for the sake of completeness. (The set S_k is sometimes defined in the literature without the $i+j>0$ requirement, adding the number 0 to the set.)

Theorem 3. *For every $k \in \mathbb{N}$, the number k^2-k-1 is the maximal number not in S_k. That is, $k^2-k-1 \notin S_k$, while for every $t \geq k^2-k$, we have that $t \in S_k$.*

Proof. By definition, $S_k = \{ik + j(k+1) : i, j \in \mathbb{N}, i+j > 0\}$. Equivalently, $S_k = \{(i+j)k + j : i, j \in \mathbb{N}, i+j > 0\}$. Thus, $t \in S_k$ iff $t > 0$ and there are $i', j' \in \mathbb{N}$ such that $i' \geq j'$ and $t = i'k + j'$. We first claim that $k^2-k-1 \notin S_k$. Let i' and j' be such that $i'k + j' = k^2-k-1$. We claim that $i' < j'$, implying that $k^2-k-1 \notin S_k$. To see this, note that $(k-2)k + (k-1) = k^2-k-1$, and that it is impossible to increase $i' = (k-2)$ and decrease $j' = (k-1)$, keeping the same total number. On the other hand, $(k-1)k+0 = k^2-k$, thus for every $t \geq k^2-k$, there are $i' \geq k-1$ and $j' \leq k-1$ such that $t = i'k + j'$. Such i' and j' witness that $t \in S_k$. $\square$

For $k \in \mathbb{N}$, we refer to k^2-k-1 as the *threshold* of k and denote it by $th(k)$. That is, $th(k) = k^2-k-1$.[6]

We can now define the family of languages $L_1, L_2, \ldots$ with which we are going to prove the lower bound. Let $\Sigma = \{a, b\}$. For $k \geq 1$, let $L'_k = \{(\epsilon + \Sigma^* \cdot a) \cdot b^i \cdot a \cdot \Sigma^\omega : i \in S_k\}$. Then, $L_k = L'_k \cup (b^* \cdot a)^\omega$. Thus, $w \in L_k$ iff w starts with $b^i \cdot a$ or has a subword of the form $a \cdot b^i \cdot a$, for $i \in S_k$, or w has infinitely many a's.

Our alert readers are probably bothered by the fact the $(b^* \cdot a)^\omega$ component of L_k is not NCW-recognizable. To see why L_k is still NCW-recognizable, consider a word w with infinitely many a's. Thus, the word is of the form $b^{i_1} \cdot a \cdot b^{i_2} \cdot a \cdot b^{i_3} \cdot a \cdots$, for non-negative integers $i_1, i_2, i_3, \ldots$. If for some $j \geq 1$, we have $i_j \in S_k$, then w is in L'_k. Otherwise, for all $j \geq 1$, we have $i_j \notin S_k$. Hence, by Theorem 3, for all $j \geq 1$, we have $i_k \leq th(k)$. Accordingly, $L_k = L'_k \cup (b^{\leq th(k)} \cdot a)^\omega$. Thus, the "infinitely many a's" disjunct can be replaced by one on which the distance between two successive a's is bounded. As we are going to prove formally, while this implies that L_k can be recognized by an NCW, it forces the NCW to count to $th(k)$, and is the key to the quadratic lower bound.

Theorem 4. *For every $k \geq 1$, the language L_k can be recognized by an NBW with $k+3$ states.*

Proof. We prove that the NBW $\mathcal{B}_k$, appearing in Fig. 4, recognizes L_k.

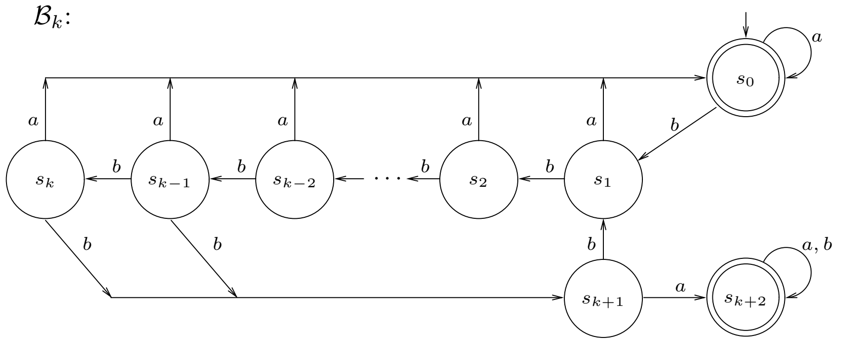

Fig. 4. The NBW $\mathcal{B}_k$ recognizing L_k.

Assume first that $w \in L_k$. Then, w either have infinitely many a's, or starts with $b^r \cdot a$ or has a subword of the form $a \cdot b^r \cdot a$, for $r \in S_k$. In the first case, w

[6] In general, for a finite set of positive integers $\{n_1, n_2, \ldots, n_l\}$, we have that all integers above $\max^2\{n_1, n_2, \ldots, n_l\}$ can be written as linear combinations of the n_i's iff the greater common divisor of the n_i's is 1. For our purpose, it is sufficient to restrict attention to linear combinations of two subsequent integers.

is accepted by $\mathcal{B}_k$, since the automaton's transition function is total and an a-transition always goes to an accepting state. Now, assume that w has a subword of the form $a \cdot b^r \cdot a$, starting at a position t, for $r \in S_k$. Then, as argued above, a run of $\mathcal{B}_k$ on w will either visit s_0 or s_{k+2} at position $t+1$. If it visits s_{k+2} it is obviously an accepting run. If it visits s_0, then at position $t+1+r$ it can visit s_{k+1} if there are natural numbers i and j such that $i+j > 0$ and $r = ik+j(k+1)$, which is the case by the assumption that $r \in S_k$. Thus, the run can visit s_{k+2} at position $t+r+2$, making it an accepting run. Hence, $w \in L_k$. The case in which w starts with $b^r \cdot a$, for $r \in S_k$, is handled analogously.

As for the other direction, assume that there is an accepting run of $\mathcal{B}_k$ on w. Then, the run either visits infinitely often s_0 or s_{k+2}. Since s_0 has only a-in-transitions, it follows that in the first case w has infinitely many a's, thus belonging to L_k. For the second case, note that a run visits the state s_{k+1}, only if there are natural numbers i and j such that $i+j > 0$ and $\mathcal{B}_k$ has read the subword $b^{ik+j(k+1)}$ since its last visit to the state s_0. Since a run visits s_0 only at initialization or after an a is read, it follows that a word visits s_{k+2} only if it starts with $b^i \cdot a$ or has a subword of the form $a \cdot b^i \cdot a$, for $i \in S_k$. Hence, $w \in L_k$. □

Next, we show that while L_k can be recognized by an NCW, every NCW recognizing L_k cannot take advantage of its non-determinism. Formally, we present a DCW (Fig. 5) for L_k that has $k^2 - k + 2$ states, and prove that an NCW recognizing L_k needs at least that many states. For simplicity, we show that the NCW must count up to $th(k)$, resulting with at least $k^2 - k$ states, and do not consider the two additional states of the DCW.

Theorem 5. *For every $k \geq 1$, the language L_k can be recognized by a DCW with $k^2 - k + 2$ states, and cannot be recognized by an NCW with fewer than $k^2 - k$ states.*

Proof. Consider the DCW $\mathcal{D}_k$, appearing in Fig. 5. In the figure, a state s_i has an a-transition to the state $s_{th(k)+2}$ if and only if $i \in S_k$. We leave to the reader the easy task of verifying that that $L(D_k) = L_k$.

We now turn to prove the lower bound. Assume by way of contradiction that there is an NCW $\mathcal{C}_k$ with at most $k^2 - k - 1$ states that recognizes L_k. The word $w = (b^{(k^2-k-1)} \cdot a)^\omega$ belongs to L_k since it has infinitely many a's. Thus, there is an accepting run r of $\mathcal{C}_k$ on w. Let t be a position such that $r_{t'} \in \alpha$ for all $t' \geq t$. Let $t' \geq t$ be the first position in which a occurs in w after t. Then, between positions $t'+1$ and $t'+k^2-k$, the run r is in α, making only b-transitions. Since $\mathcal{C}_k$ has at most k^2-k-1 states, it follows that there are positions t_1 and t_2, with $t' < t_1 < t_2 \leq t' + k^2 - k$ such that $r_{t_1} = r_{t_2}$. Consider now the word $w' = w_1 \cdot w_2 \cdots w_{t_1} \cdot b^\omega$. On the one hand, w' is accepted by $\mathcal{C}_k$ via the run $r' = r_0, r_1, \ldots, r_{t_1}, (r_{t_1+1}, r_{t_1+2}, \ldots, r_{t_2})^\omega$. On the other hand, w' has only finitely many a's, and by Theorem 3, it has no i consequent b's followed by an a, such that $i \in S_k$. Indeed, all the subwords of the form $a \cdot b^i \cdot a$ in w' have $i = k^2 - k - 1$. Hence, $w' \notin L_k$, which leads to a contradiction. □

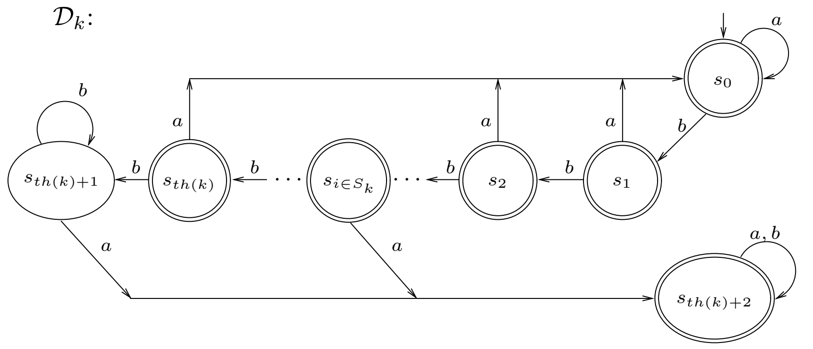

Fig. 5. The DCW $\mathcal{D}_k$ recognizing L_k.

4.3 An Exponential Lower Bound

We now push the circumventing-counting idea to its limit, and use it in order to describe a family of languages $L_2, L_3, \ldots$ such that for all $k \geq 2$, an NBW for L_k has $O(k)$ states whereas an NCW for L_k requires at least $k2^k$ states. As in the quadratic lower bound, the NCW has to bound the distance between occurrences of an event. Now, however, the distance is exponential in k.

Let $\Sigma = \{0, 1, \$, \#\}$. The language L_k is going to be the union of a language L'_k with the language $(\Sigma^* \cdot \#)^\omega$. Before we define L'_k formally, we describe the intuition behind it. Note that the $(\Sigma^* \cdot \#)^\omega$ component of L_k is not NCW-recognizable. Thus, one task of L'_k is to neutralize the non NCW-recognizability of this component. We do this by letting L'_k contain all the words in $(\Sigma^* \cdot \#)^\omega$ that have a subword $(0+1+\$)^h$, for $h > th(k)$, for some threshold $th(k)$. As with the quadratic lower bound, this would make it possible to replace the $(\Sigma^* \cdot \#)^\omega$ component by $(\Sigma^{\leq th(k)} \cdot \#)^\omega$, which is NCW-realizable. The second task of L'_k would be to accomplish the first task with an exponential threshold.

The language L'_k is going to fulfill its second task as follows. Consider a word in Σ^ω and a subword $u \in (0 + 1 + \$)^*$ of it. The subword u is of the form $v_0\$v_1\$v_2\$v_3 \cdots$, for $v_i \in (0+1)^*$. Thus, u can be viewed as an attempt to encode a binary k-bit cyclic counter in which two adjacent values are separated by \$. For example, when $k = 3$, a successful attempt might be 100\$101\$110\$111\$000. Each subword in $(0+1+\$)^*$ of length $(k+1)2^k$ must reach the value 1^k or contain an error (in its attempt to encode a counter). There are two types of errors. One type is a "syntax error", namely a value v_i of length different from k. The second type is an "improper-increase error", namely a subword $v_i \cdot \$ \cdot v_{i+1} \in (0+1)^k \cdot \$ \cdot (0+1)^k$ such that v_{i+1} is not the successor of v_i in a correct binary encoding of a cyclic k-bit counter. The language L'_k consists of all words that contain the value 1^k or an error, eventually followed by $\#$.

We now define L'_k formally. For $v, v' \in (0 + 1)^*$, we use $not\_succ_k(v, v')$ to indicate that v and v' are in $(0 + 1)^k$ but v' is not the successor of v in the binary encoding of a k-bit counter. For example, $not\_succ_3(101, 111)$ holds, but $not\_succ_3(101, 110)$ does not hold. We define the following languages over Σ.

– $S_k = \{\$ \cdot (0+1)^m \cdot \$: m < k\} \cup \{(0+1)^m : m > k\}$,
– $I_k = \{v \cdot \$ \cdot v' \ : \ \mathit{not\_succ}_k(v, v')\}$, and
– $L'_k = \Sigma^* \cdot (S_k \cup I_k \cup \{1^k\}) \cdot \Sigma^* \cdot \# \cdot \Sigma^\omega$.

Finally, we define $L_k = L'_k \cup (\Sigma^* \cdot \#)^\omega$. For example, taking $k = 3$, we have that $010\$011\#110\$111\#\cdots$ is in L_3 since it is in L'_3 with a 111 subword, the word $010\$\$011\#\cdots$ is in L_3 since it is in L'_3 by a syntax error, the word $\$010\$010\$\#\cdots$ is in L_3 since it is in L'_3 by an improper-increase error, the word $(010\$011\#)^\omega$ is in L_3 since it has infinitely many #'s, and the word $010\$011\#000\$001\$010\#1^\omega$ is not in L_3, as it has only finitely many #'s, it does not contain an error, and while it does contain the subword 111, it does not contain a subword 111 that is eventually followed by #.

Lemma 4. *For every $k \geq 1$, the language L'_k can be recognized by an NBW with $O(k)$ states and by an NCW with $O(k)$ states.*

Proof. We show that there is an NFW with $O(k)$ states recognizing $S_k \cup I_k \cup \{1^k\}$. Completing the NFW to an NBW or an NCW for L'_k is straightforward. It is easy to construct NFWs with $O(k)$ states for S_k and for $\{1^k\}$. An NFW with $O(k)$ states for I_k is fairly standard too (see, for example, [10]). The idea is that if v' is the successor of v in a binary k-bit cyclic counter, then v' can be obtained from v by flipping the bits of the $0 \cdot 1^*$ suffix of v, and leaving all other bits unchanged (the only case in which v does not have a suffix in $0 \cdot 1^*$ is when $v \in 1^*$, in which case all bits are flipped). For example, the successor of 1001 is obtained by flipping the bits of the suffix 01, which results in 1010. Accordingly, there is an improper-increase error in $v \cdot \$ \cdot v'$ if there is at least one bit of v that does not respect the above rule. An NFW can guess the location of this bit and reveals the error by checking the bit located $k+1$ bits after it, along with the bits read in the suffix of v that starts in this bit. □

An immediate corollary of Lemma 4 is that L_k can be recognized by an NBW with $O(k)$ states. Next, we show that while L_k is NCW-recognizable, an NCW for it must be exponentially larger.

Lemma 5. *For every $k \geq 2$, the language L_k is NCW-recognizable, and every NCW recognizing L_k must have at least $k2^k$ states.*

Proof. We first prove that L_k is NCW-recognizable. Let $th(k) = (k+1)2^k$. Consider the language $B_k = (\Sigma^{\leq th(k)} \cdot \#)^\omega$. It is easy to see that B_k is NCW-recognizable. We prove that $L_k = L'_k \cup B_k$. Since, by Lemma 4, the language L'_k is NCW-recognizable, it would follow that L_k is NCW-recognizable. Clearly, $B_k \subseteq (\Sigma^* \cdot \#)^\omega$. Thus, $L'_k \cup B_k \subseteq L_k$, and we have to prove that $L_k \subseteq L'_k \cup B_k$. For that, we prove that $(\Sigma^* \cdot \#)^\omega \subseteq L'_k \cup B_k$. Consider a word $w \in (\Sigma^* \cdot \#)^\omega$. If $w \in B_k$, then we are done. Otherwise, w contains a subword $u \in (0+1+\$)^h$, for $h > th(k)$. Thus, either u does not properly encode a k-bit cyclic counter (that is, it contains a syntactic or an improper-increase error) or u has the subword 1^k. Hence, $u \in \Sigma^* \cdot (S_k \cup I_k \cup \{1^k\}) \cdot \Sigma^*$. Since $w \in (\Sigma^* \cdot \#)^\omega$, it has infinitely many occurrences of #'s. In particular, there is an occurrence of # after the subword u. Thus, $w \in L'_k$, and we are done.

We now turn to prove the lower bound. Assume by way of contradiction that there is an NCW $\mathcal{C}_k$ with acceptance set α and at most $k2^k - 1$ states that recognizes L_k. Consider the word $w = (00 \cdots 0\$00 \cdots 01\$ \cdots \$11 \cdots 10\#)^\omega$, in which the distance between two consequent #'s is $d = (k+1)(2^k - 1)$. Note that for all $k \geq 2$, we have that $d > k2^k$. The word w has infinitely many #'s and it therefore belongs to L_k. Thus, there is an accepting run r of $\mathcal{C}_k$ on w. Let t be a position such that $r_{t'} \in \alpha$ for all $t' \geq t$. Let $t_0 \geq t$ be the first position after t such that $w_{t_0} = \#$. Since $\mathcal{C}_k$ has at most $k2^k - 1$ states, there are two positions t_1 and t_2, with $t_0 < t_1 < t_2 \leq t_0 + k2^k$, such that $r_{t_1} = r_{t_2}$.

Consider the word $w' = w_1 \cdot w_2 \cdots w_{t_1} \cdot (w_{t_1+1} \cdots w_{t_2})^\omega$. The NCW $\mathcal{C}_k$ accepts w' with a run r' that pumps r between the positions t_1 and t_2. Formally, $r' = r_0, r_1, \ldots, r_{t_1}, (r_{t_1+1}, \ldots, r_{t_2})^\omega$. Note that since $r_{t'} \in \alpha$ for all $t' \geq t$, the run r' is indeed accepting. We would get to a contradiction by proving that $w' \notin L_k$.

Since $t_2 \leq t_0 + k2^k$ and $k2^k < d$, we have that $w_{t_1+1} \cdots w_{t_2}$ has no occurrence of #, thus w' has no occurrences of # after position t_0. Recall that $L_k = L'_k \cup (\Sigma^* \cdot \#)^\omega$. By the above, $w' \notin (\Sigma^* \cdot \#)^\omega$. Furthermore, since $L'_k = \Sigma^* \cdot (S_k \cup I_k \cup \{1^k\}) \cdot \Sigma^* \cdot \# \cdot \Sigma^\omega$, the fact w' has no occurrences of # after position t_0 implies that the only chance of w' to be in L_k is to have a prefix of $w_1 \cdots w_{t_0}$ in $\Sigma^* \cdot (S_k \cup I_k \cup \{1^k\}) \cdot \Sigma^* \cdot \#$. Such a prefix, however, does not exist. Indeed, all the subwords in $(0 + 1 + \$)^*$ of $w_1 \cdots w_{t_0}$ do not contain errors in their encoding of a k-bit counter, nor they reach the value 1^k. It follows that $w \notin L_k$, and we are done. $\square$

Lemmas 4 and 5 imply the desired exponential lower bound:

Theorem 6 ([3]). *There is a family of languages $L_2, L_3, \ldots$, over an alphabet of size 4, such that for every $k \geq 2$, the language L_k is NCW-recognizable, it can be recognized by an NBW with $O(k)$ states, and every NCW that recognizes it has at least $k2^k$ states.*

Combining the above lower bound with the upper bound in Theorem 1, we can conclude with the following.[7]

Theorem 7 ([3]). *The asymptotically tight bound for the state blow up in the translation, when possible, of an NBW to an equivalent NCW is $2^{\Theta(n)}$.*

5 Discussion

It is well known that nondeterministic automata are exponentially more succinct than deterministic ones. The succinctness is robust and it applies to all known classes of automata on finite or infinite objects. Restricting attention to nondeterministic automata makes the issue of succinctness more challenging, as now

[7] Note that the lower and upper bounds are only asymptotically tight, leaving a gap in the constants. This is because the NBW that recognizes L_k requires $O(k)$ states and not strictly k states.

all classes of automata may guess the future, and the question is whether certain acceptance conditions can use this feature better than others. For example, translating a nondeterministic Rabin word automaton with n states and index k to an NBW, results in an automaton with $O(nk)$ states, whereas translating a nondeterministic Streett automaton with n states and index k, results in an NBW with $O(n2^k)$ states. The difference between the blow-ups in the case of Rabin and Streett can be explained by viewing the acceptance condition as imposing additional nondeterminism. Indeed, simulating a Rabin automaton, an NBW has to guess not only the accepting run, but also the pair $\langle G, B\rangle$ that is going to be satisfied and the position after which no states in B are visited (hence the $O(k)$ factor in the blow up), whereas in a Streett automaton, the simulating NBW has to guess, for each pair, the way in which it is going to be satisfied (hence the $O(2^k)$ factor). This intuition is supported by matching lower bounds [23]. Starting with a Büchi automaton, no such additional nondeterminism hides in the acceptance condition, so one would not expect Büchi automata to be more succinct than other nondeterministic automata. The challenge grows with an acceptance condition like co-Büchi, whose expressive power is strictly weaker, and thus not all languages are candidates for proving succinctness. The exponential lower-bound described in Sect. 4 shows that the Büchi condition is still exponentially more succinct than its dual co-Büchi condition. The explanation to this succinctness is the ability of the Büchi condition to easily specify the fact that some event P should repeat infinitely often. Languages that involve such a specification may still be NCW-recognizable, as other components of the language force the distance between successive occurrences of P to be bounded by some fixed threshold 2^k. While an NBW for the language does not have to count to the threshold and can be of size $O(k)$, an NCW for the language has to count, which requires it to have at least 2^k states.

Co-Büchi automata can be determinized with a $2^{O(n)}$ blow up [16]. A $2^{O(n \log n)}$ translation of NBW to DCW was known, but a matching lower bound was known only for the translation of NBW to deterministic Rabin or Streett automata. As detailed in [3], the improved $2^{O(n)}$ translation of NBW to NCW described in Sect. 3, also suggests an improved $2^{O(n)}$ translation of NBW to DCW. Indeed, since the exponential component of the constructed NCW is deterministic, then applying the break-point subset construction of [16] on it does not involve an additional exponential blow-up. In particular, this implies a Safraless and symbolic translation of LTL formulas to DBW, when possible. Furthermore, by [4], one cannot expect to do better than the breakpoint construction, making the translation of NBW to DCW [3] optimal. In addition, as detailed in [3], the translation described in Sect. 3 has a one-sided error. Thus, when applied to an NBW that is not NCW-recognizable, the constructed NCW contains the language of the NBW. Accordingly, translating LTL formulas that are not DBW-recognizable to a DBW, one gets a DBW that under-approximates the specification. For many applications, and in particular synthesis, one can work with such an under-approximating automaton, and need not worry about the specification being DBW-recognizable.

References

1. Accellera: Accellera organization inc. http://www.accellera.org (2006)
2. Aminof, B., Kupferman, O., Lev, O.: On the relative succinctness of nondeterministic Büchi and co-Büchi word automata. In: Proc. 15th Int. Conf. on Logic for Programming Artificial Intelligence and Reasoning. Lecture Notes in Computer Science, Vol. 5330. Springer (2008) 183–197
3. Boker, U., Kupferman, O.: Co-ing Büchi made tight and useful. In: Proc. 24th IEEE Symp. on Logic in Computer Science. (2009)
4. Boker, U., Kupferman, O., Rosenberg, A.: Alternation removal in Büchi automata. In: Proc. 37th Int. Colloq. on Automata, Languages, and Programming. (2010)
5. Büchi, J.: On a decision method in restricted second order arithmetic. In: Proc. Int. Congress on Logic, Method, and Philosophy of Science. 1960. Stanford University Press (1962) 1–12
6. Emerson, E., Jutla, C.: The complexity of tree automata and logics of programs. In: Proc. 29th IEEE Symp. on Foundations of Computer Science. (1988) 328–337
7. Gentilini, R., Piazza, C., Policriti, A.: Computing strongly connected components in a linear number of symbolic steps. In: 14th ACM-SIAM Symp. on Discrete Algorithms. (2003) 573–582
8. Krishnan, S., Puri, A., Brayton, R.: Deterministic ω-automata vis-a-vis deterministic Büchi automata. In: Algorithms and Computations. Lecture Notes in Computer Science, Vol. 834. Springer (1994) 378–386
9. Kupferman, O.: Tightening the exchange rate beteen automata. In: Proc. 16th Annual Conf. of the European Association for Computer Science Logic. Lecture Notes in Computer Science, Vol. 4646. Springer (2007) 7–22
10. Kupferman, O., Lustig, Y., Vardi, M.: On locally checkable properties. In: Proc. 13th Int. Conf. on Logic for Programming Artificial Intelligence and Reasoning. Lecture Notes in Computer Science, Vol. 4246. Springer (2006) 302–316
11. Kupferman, O., Morgenstern, G., Murano, A.: Typeness for ω-regular automata. International Journal on the Foundations of Computer Science **17** (2006) 869–884
12. Kupferman, O., Vardi, M.: From linear time to branching time. ACM Transactions on Computational Logic **6** (2005) 273–294
13. Kurshan, R.: Computer Aided Verification of Coordinating Processes. Princeton Univ. Press (1994)
14. Landweber, L.: Decision problems for ω–automata. Mathematical Systems Theory **3** (1969) 376–384
15. McNaughton, R.: Testing and generating infinite sequences by a finite automaton. Information and Control **9** (1966) 521–530
16. Miyano, S., Hayashi, T.: Alternating finite automata on ω-words. Theoretical Computer Science **32** (1984) 321–330
17. Morgenstern, A., Schneider, K.: From LTL to symbolically represented deterministic automata. In: Proc. 9th Int. Conf. on Verification, Model Checking, and Abstract Interpretation. Lecture Notes in Computer Science, Vol. 4905. (2008) 279–293
18. Piterman, N.: From nondeterministic Büchi and Streett automata to deterministic parity automata. In: Proc. 21st IEEE Symp. on Logic in Computer Science. IEEE press (2006) 255–264
19. Pnueli, A., Rosner, R.: On the synthesis of a reactive module. In: Proc. 16th ACM Symp. on Principles of Programming Languages. (1989) 179–190

20. Rabin, M.: Decidability of second order theories and automata on infinite trees. Transaction of the AMS **141** (1969) 1–35
21. Ravi, K., Bloem, R., Somenzi, F.: A comparative study of symbolic algorithms for the computation of fair cycles. In: Proc. 3rd Int. Conf. on Formal Methods in Computer-Aided Design. Lecture Notes in Computer Science, Vol. 1954. Springer (2000) 143–160
22. Safra, S.: On the complexity of ω-automata. In: Proc. 29th IEEE Symp. on Foundations of Computer Science. (1988) 319–327
23. Safra, S., Vardi, M.: On ω-automata and temporal logic. In: Proc. 21st ACM Symp. on Theory of Computing. (1989) 127–137
24. Street, R., Emerson, E.: An elementary decision procedure for the μ-calculus. In: Proc. 11th Int. Colloq. on Automata, Languages, and Programming. Vol. 172. Springer (1984) 465–472
25. Vardi, M., Wolper, P.: Automata-theoretic techniques for modal logics of programs. Journal of Computer and Systems Science **32** (1986) 182–221
26. Vardi, M., Wolper, P.: Reasoning about infinite computations. Information and Computation **115** (1994) 1–37
27. Wolper, P., Vardi, M., Sistla, A.: Reasoning about infinite computation paths. In: Proc. 24th IEEE Symp. on Foundations of Computer Science. (1983) 185–194
28. Yan, Q.: Lower bounds for complementation of ω-automata via the full automata technique. In: Proc. 33rd Int. Colloq. on Automata, Languages, and Programming. Lecture Notes in Computer Science, Vol. 4052. Springer (2006) 589–600