Co-ing Büchi Made Tight and Useful

Udi Boker and Orna Kupferman
Hebrew University*

Abstract

We solve the longstanding open problems of the blow-up involved in the translations (when possible) of a nondeterministic Büchi word automaton (NBW) to a nondeterministic co-Büchi word automaton (NCW) and to a deterministic co-Büchi word automaton (DCW). For the NBW to NCW translation, the currently known upper bound is $2^{O(n \log n)}$ *and the lower bound is* $1.5n$*. We improve the upper bound to* $n2^n$ *and describe a matching lower bound of* $2^{\Omega(n)}$*. For the NBW to DCW translation, the currently known upper bound is* $2^{O(n \log n)}$*. We improve it to* $2^{O(n)}$*, which is asymptotically tight. Both of our upper-bound constructions are based on a simple subset construction, do not involve intermediate automata with richer acceptance conditions, and can be implemented symbolically. We point to numerous applications of the new constructions. In particular, they imply a simple subset-construction based translation (when possible) of LTL to deterministic Büchi word automata.*

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 [4, 18, 25]. 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 [15, 30]. 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 [5, 24, 27, 28, 29, 32]. 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 missing in our understanding of automata on infinite objects, but also that we may be using algorithms that can be significantly improved.

Two such fundamental and longstanding open problems are the translation, when possible, of a nondeterministic Büchi word automaton (NBW) to an equivalent nondeterministic co-Büchi word automaton (NCW) and to an equivalent deterministic co-Büchi word automaton (DCW).[1] NCWs are less expressive than NBWs. For example, the language $\{w : w \text{ has infinitely many } a\text{'s}\}$ 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 is currently known goes via an intermediate deterministic Streett [27] or parity [8, 23] automaton [13]. Thus, starting with an NBW with n states, we end up with a DCW with $2^{O(n \log n)}$ states. Not less problematic, the determinization construction is awfully complex and is not amenable to optimizations and a symbolic implementation. Note that the NBW to NCW problem does not require us to end up in a deterministic automaton. Yet, it is not known

*School of Engineering and Computer Science, Hebrew University, Jerusalem 91904, Israel. Email: {udiboker,orna}@cs.huji.ac.il

[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 [4]; whereas in *co-Büchi* automata, a run is accepting iff it visits only the accepting set from some position.

[2] When applied to universal Büchi automata, the translation in [20], of alternating Büchi automata into NBW, results in DBW. By dualizing it, one gets a translation of NCW to DCW.

how to take advantage of the allowed nondeterminism, and it is not known 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 the best known lower bound is linear. 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. Only in [11], was it shown that there is an NBW whose equivalent NCW requires a different structure, and only recently a non-trivial lower bound, of $1.5n$, was proven [2].

Let us now turn to the problem of translating NBW to DCW (when possible). As noted above, the best known upper bound for the translation is $2^{O(n \log n)}$, and a $2^{\Omega(n)}$ lower bound follows from determinization of automata on finite words. For general ω-regular languages, Michel's $2^{\Omega(n \log n)}$ lower bound implies we cannot hope for a subset-construction based procedure for NBW determinization [17, 19]. Michel's language, however, is not NCW-recognizable, and the problem of translating NBW to DCW, when possible, is open. As in the case of going to an NCW, the question is not only whether we can come up with a $2^{O(n)}$ construction, but also whether the construction can avoid a complex intermediate determinization construction. Recall that NCWs can be determinized via the breakpoint construction of [20], which is based on the simple subset construction: starting with an NCW with n states, one can generate an equivalent DCW with 3^n states. Thus, a linear NBW to NCW construction would immediately imply a $3^{O(n)}$ NBW to DCW construction. Such a linear construction, however, is not known.

Beyond the theoretical challenge in tightening the gaps, and the fact they are related to other gaps in our knowledge [10], the translations of NBW to NCW and DCW has immediate important applications in formal methods. The first class of applications uses the NCW directly. 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 quadratic [26]. It is shown in [13] 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.

The second class of applications, which we find more appealing, is in procedures that involve determinization. The acceptance by the industry of automata-based procedures that involve determinization of automata on infinite words has been very partial. Examples include complementation, synthesis, game solving, temporal logic decidability, reasoning about Markov decision processes, monitoring-based run-time verification, and more. This has to do with the intricacy of optimal determinization constructions [8, 23, 27], the fact that the state space that results from determinization is awfully complex and is not amenable to optimizations and a symbolic implementation, and the fact that determinization requires the introduction of acceptance conditions that are more complex than the Büchi acceptance condition. A simple translation of NBW to DCW would significantly extend the scope of such procedures, many of which are now restricted to safety properties.[4] This class of applications is particularly appealing if we keep in mind the fact that DCW and DBW are dual, and that NBWs are usually obtained from LTL formulas, whose complementation is straightforward. Thus, an improved NBW to DCW translation implies an improved LTL to DBW translation (when possible).

In this paper we solve both problems. Let us start with the translation of NBW to NCW. In the upper-bound front, we point to useful observations on the structure of automata whose languages are NCW-recognizable, and we use these observations in order to translate 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, our translation yields an equivalent NCW with $n2^n$ states, and it has a simple symbolic implementation [21]. In the lower-bound front, we show that the ability of NBWs to abstract precise counting by counting to infinity with two states leads to exponential succinctness. Formally, we show that for every integer $k \geq 2$, there is a language L_k over a four-letter alphabet, such that L_k can be recognized by an NBW with $O(k)$ states, whereas the minimal NCW that recognizes L_k has $k2^k$ states. This leads to an asymptoti-

[3] A general technique for proving lower bounds on the size of automata on infinite words is suggested in [33]. 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] For some settings of these applications, procedures that avoid determinization have already been suggested [6, 9, 12, 14]. Our goal here is not to avoid determinization, but to suggest a simple subset-construction based determinization procedure for the fragment of NBW that is DCW-recognizable.

cally tight $2^{\Theta(n)}$ bound to the longstanding open problem of the state blow up in the translation of NBW to NCW.

Our exponential lower bound suggests that an attempt to improve the translation of NBW to DCW by going through an intermediate NCW is doomed to result in an automaton with doubly-exponentially many states. Fortunately, we show that the NCW constructed by our upper-bound translation is transparent to additional applications of the subset construction! That is, applying the subset construction on top of $\mathcal{C}$ does not increase its state space. Using this property, we can translate the NBW $\mathcal{B}$ to an equivalent DCW with 3^n states. The construction is based on the simple breakpoint construction of [20], and it can be implemented symbolically [21]. This answers to the positive the longstanding open problem of whether an NBW that is DCW-recognizable can be translated to an equivalent DCW with only a $2^{O(n)}$ blow-up. It also implies that an LTL formula of length n that is DBW-recognizable can be translated, using the breakpoint construction, to a DBW with $2^{2^{O(n)}}$ states. For both translations, a matching lower bound exists [13].

Our improved upper bound immediately implies the applications discussed above. We elaborate on the applications further in Section 5. An important and useful property of our construction is the fact it has only a one-sided error when applied to automata whose language is not NCW-recognizable. Thus, given an NBW $\mathcal{B}$, the NCW $\mathcal{C}$ and the DCW $\mathcal{D}$ we construct are such that $L(\mathcal{B}) \subseteq L(\mathcal{C}) = L(\mathcal{D})$ (and $L(\mathcal{B}) = L(\mathcal{C}) = L(\mathcal{D})$ in case $\mathcal{B}$ is NCW-recognizable). Likewise, given an LTL formula ψ, the DBW $\mathcal{D}_\psi$ we construct is such that $L(\mathcal{D}_\psi) \subseteq L(\psi)$ (and $L(\mathcal{D}_\psi) = L(\psi)$ in case ψ is DBW-recognizable). As we show in Section 5, this enables us to extend the scope of the applications also to specifications not in NCW or DBW.

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 $inf(r)$ of states that the run r visits infinitely often. Formally, $inf(r) = \{q \in Q \mid \text{for infinitely many } i \in \mathbb{N}, \text{ we have } r_i = q\}$. As Q is finite, it is guaranteed that $inf(r) \neq \emptyset$. The run r is *accepting* iff the set $inf(r)$ 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 $\{\mathrm{D}, \mathrm{N}\} \times \{\mathrm{B}, \mathrm{C}\} \times \{\mathrm{W}\}$. 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 [18], DBWs are strictly less expressive than NBWs, and so are DCWs [16]. 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 [20].

3 From NBW to NCW

In this section we present our improved upper and lower bounds for the translation of NBW to NCW (when possible). 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 $inf(r) \subseteq \alpha$.

3.1 Upper Bound

We start with the upper bound and provide a constructive proof, showing that for every NBW $\mathcal{B}$ with k states whose language is NCW-recognizable, there is 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!

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 3.1 *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, $L(\mathcal{D}^{d_0}) = L(\mathcal{D}^{d_1})$.*

Proof: We prove that $L(\mathcal{D}^{d_0}) \subseteq L(\mathcal{D}^{d_1})$. The other direction is analogous. Consider a word $w \in L(\mathcal{D}^{d_0})$. Since $\delta_\mathcal{D}(x_0) = d_0$, we have that $x_0 \cdot w \in L(\mathcal{B})$. Let r be the accepting run of $\mathcal{B}$ on $x_0 \cdot w$. Since $\delta_\mathcal{B}(x_0) = \delta_\mathcal{B}(x_1)$, there is a run r' of $\mathcal{B}$ on $x_1 \cdot w$ such that $r'(x_1) = r(x_0)$ and for all prefixes y of w such that $x \prec y$, it holds $r'(y) = r(y)$. Clearly, $inf(r') = inf(r)$, thus r' is accepting, and $x_1 \cdot w \in L(\mathcal{B})$. Hence, $x_1 \cdot w \in L(\mathcal{D})$. Since $\delta_\mathcal{D}(x_1) = d_1$, it follows that $w \in L(\mathcal{D}^{d_1})$, and we are done. □

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 3.2 *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 $L(\mathcal{D}^{d'}) = L(\mathcal{D}^{d_0})$.*

Proof: We prove that $L(\mathcal{D}^{d'}) = L(\mathcal{D}^{d_0})$ for states $d' \in \delta(d_0, w)$. The proof for states $d' \in \delta(d_1, w)$ is analogous.

For every word $w \in (u + v)^*$ there is an index $k \geq 0$ such that $w \in (u + v)^k$. We prove by induction on k that $L(\mathcal{D}^{d_0}) = L(\mathcal{D}^{d'})$ for every $d' \in \delta(d_0, w)$. The base case, in which $k = 0$, is obvious, as $d' = d_0$. Assume that the claim holds for k and consider a word $w \in (u + v)^{k+1}$. Assume that $w = x \cdot u$, for $x \in (u + v)^k$. By the induction hypothesis, the state $d'' = \delta(d_0, x)$ is such that $L(\mathcal{D}^{d_0}) = L(\mathcal{D}^{d''})$.

Consider a word $y \in L(\mathcal{D}^{d_0})$. We claim that $y \in L(\mathcal{D}^{d'})$. Indeed, since $\delta(d_0, u) = d_0$, we have that $u \cdot y \in L(\mathcal{D}^{d_0})$. Thus, $u \cdot y \in L(\mathcal{D}^{d''})$. Since $d' = \delta(d'', u)$, it follows that $y \in L(\mathcal{D}^{d'})$. For the other direction, consider a word $y \in L(\mathcal{D}^{d'})$. We claim that $y \in L(\mathcal{D}^{d_0})$. Indeed, since $d' = \delta(d'', u)$, we have that $u \cdot y \in L(\mathcal{D}^{d''})$. Thus, by the induction hypothesis, $u \cdot y \in L(\mathcal{D}^{d_0})$. Therefore, since $\delta(d_0, u) = d_0$, we get that $y \in L(\mathcal{D}^{d_0})$, as required.

The other case, in which $w = x \cdot v$, is proved analogously, having d_1 in the role of 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 3.3 *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 $v^\omega \in L$.*

Proof: Let $\mathcal{D} = \langle \Sigma, D, \delta, D_0, \alpha \rangle$ be a DCW recognizing L. Assume by way of contradiction that $v^\omega \notin L$. Then, there is $i_1 \geq 1$ such that the run of $\mathcal{D}$ on v^{i_1} visits a state not in α. Let $\delta(v^{i_1} \cdot u) = s_1$. By the first condition, $L(\mathcal{D}^{s_1}) = L$, thus, by our assumption, $v^\omega \notin L(\mathcal{D}^{s_1})$. Then, there is $i_2 \geq 1$ such that the run of $\mathcal{D}^{s_1}$ on v^{i_2} visits a state not in α. Let $\delta(s_1, v^{i_2} \cdot u) = s_2$. In a similar way, we can continue and define an infinite sequence of states $s_1, s_2, s_3, \ldots$ such that for all $j \geq 1$, there is $i_{j+1} \geq 1$ such that the run of $\mathcal{D}^{s_j}$ on $v^{i_{j+1}}$ visits a state not in α and $s_{j+1} = \delta(s_j, v^{i_{j+1}} \cdot u)$. Since D is finite, there are $1 \leq l_1 < l_2$ such that $s_{l_1} = s_{l_2}$.

Consider the word $w = v^{i_1} \cdot u \cdot v^{i_2} \cdot u \cdots v^{i_{l_1}} \cdot u \cdot (v^{i_{l_1+1}} \cdot u \cdots v^{i_{l_2}} \cdot u)^\omega$. On the one hand, the run of $\mathcal{D}$ on w visits infinitely many states not in α. On the other hand, $w \in (v^* \cdot u^+)^\omega$. Therefore, by the second condition, $w \in L$. Thus, we have reached a contradiction, implying that $v^\omega \in L$. □

By considering the language of a specific state of the DCW, Lemma 3.3 implies the following.

Corollary 3.4 *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 our construction together with its acceptance condition.

Theorem 3.5 *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 $\langle b, E \rangle \in inf(r')$. We prove that $\langle b, E \rangle \in \alpha_C$. Since $\langle b, E \rangle \in inf(r')$, then $b \in inf(r)$. 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, $inf(r) \subseteq \alpha_C$, 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 $inf(r')$, and let x be a prefix of w such that $r'(x) = \langle b, E \rangle$. Since r' is accepting, $inf(r') \subseteq \alpha_\mathcal{C}$, 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 3.1 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 3.1, 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 3.4 with u being $z^{i_2 - i_1}$ and v being t_i. The first condition is satisfied by Lemma 3.2. 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 3.4, 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. □

3.2 Lower Bound

In this section we prove an exponential lower bound for the state blow-up in the translation of NBW to NCW. For that, we 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.

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. Our alert readers are probably bothered by the fact 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. The way to do this would be to have a finite bound $th(k)$ (the *threshold* of k) such that L'_k contains all words in $(\Sigma^* \cdot \#)^\omega$ that have a subword $(0+1+\$)^h$, for $h > th(k)$. Accordingly, the language L_k is also the union of L'_k with the language $(\Sigma^{\leq th(k)} \cdot \#)^\omega$, in which the problematic $(\Sigma^* \cdot \#)^\omega$ component is replaced by a component that is NCW-recognizable. The point is that while an NBW that recognizes L_k can use its $L'_k \cup (\Sigma^* \cdot \#)^\omega$ definition, an NCW for L_k must use its equivalent $L'_k \cup (\Sigma^{\leq th(k)} \cdot \#)^\omega$ definition, and must therefore count to $th(k)$. Thus, the second task of L'_k is to fulfill the first task with a threshold that is exponential in k. This way, the ability of the NBW to avoid the counting gives it the exponential advantage we are after.

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.

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

Now, $L_k = L'_k \cup (\Sigma^* \cdot \#)^\omega$. 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 3.6 *For every $k \geq 1$, the language L'_k can be recognized by an NCW and by an NBW 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, [7]). 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 3.6 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 3.7 *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 3.6, 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 $w = (00\cdots0\$00\cdots01\$\cdots\$11\cdots10\#)^\omega$, 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 $r_{t_1} = r_{t_2}$.

Let $w' = w_1 \cdot w_2 \cdots w_{t_1} \cdot (w_{t_1+1} \cdots w_{t_2})^\omega$. 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. □

Lemmas 3.6 and 3.7 imply the desired exponential lower bound:

Theorem 3.8 *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 3.5, we can conclude with the following.[5]

[5]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.

Theorem 3.9 *The asymptotically tight bound for the state blow up in the translation, when possible, of an NBW to an equivalent NCW is $2^{\Theta(n)}$.*

Remark 3.10 While NCW = DCW, nondeterminism does add power to *weak automata* [22]. Let NWW and DWW denote nondeterministic and deterministic weak automata on infinite words. While NWW = NCW, it is known that DWW = DBW ∩ DCW [3]. Since the translation of an NCW to an equivalent NWW is always possible and only doubles the state space, Theorem 3.5 implies an $n2^n$ translation of an NBW to an NWW, when possible. We now show that the languages L_k we used in the lower-bound proof are actually DWW-recognizable, implying that the asymptotically tight bound for the state blow up in the translation, when possible, of an NBW to an equivalent NWW is $2^{\Theta(n)}$. Thus, Theorem 3.9 applies already to the special case of weak automata.

We prove that L_k is DBW-recognizable. Since DWW = DBW ∩ DCW, and L_k is DCW-recognizable (Lemma 3.7), it follows that L_k is DWW-recognizable. Clearly, $(\Sigma^* \cdot \#)^\omega$ is DBW-recognizable. In addition, L'_k is a co-safety language (every word in L'_k has a *good prefix*, namely a prefix all whose extensions are in L'_k), As such, L'_k is DBW-recognizable. Since $L_k = L'_k \cup (\Sigma^* \cdot \#)^\omega$ and DBWs are closed under finite union, we are done. □

4 From NBW to DCW

In Section 3.1, we presented the augmented subset construction and translated an NBW $\mathcal{B}$ to an NCW $\mathcal{C}$. In this section we use the special structure of $\mathcal{C}$ in order to determinize it without an additional exponential blow-up. The key to our construction is the observation that the augmented subset construction is transparent to additional applications of the subset construction. Indeed, applying the subset construction on $\mathcal{C}$ with state space $B \times 2^B$, one ends up in a deterministic automaton with state space $\{\langle E, E\rangle : E \in 2^B\}$, which is clearly isomorphic to 2^B.

It is well known that the subset construction can be used in order to translate an NCW with state space C to a DCW with state space 3^C [20]. Thus, the observation above suggests that, when applied to $\mathcal{C}$, the translation of [20] would not involve an additional exponential blow-up on top of the one involved in going from $\mathcal{B}$ to $\mathcal{C}$. As we show in Theorem 4.1 below, this is indeed the case.

Theorem 4.1 *For every NBW $\mathcal{B}$ with k states that is NCW-recognizable there is an equivalent DCW $\mathcal{D}$ with at most 3^k states.*

Proof: The DBW $\mathcal{D}$ follows all the runs of the NCW $\mathcal{C}$ constructed in Theorem 3.5. Let $\alpha_\mathcal{C} \subseteq B \times 2^B$ be the acceptance condition of $\mathcal{C}$. The DCW $\mathcal{D}$ accepts a word if

some run of $\mathcal{C}$ remains in $\alpha_{\mathcal{C}}$ from some position.[6] At each state, $\mathcal{D}$ keeps the corresponding subset of the states of $\mathcal{C}$, and it updates it deterministically whenever an input letter is read. In order to check that some run of $\mathcal{C}$ remains in $\alpha_{\mathcal{C}}$ from some position, the DCW $\mathcal{D}$ keeps track of runs that do not leave $\alpha_{\mathcal{C}}$. The key observation in [20] is that keeping track of such runs can be done by maintaining the subset of states that belong to these runs.

Formally, let $\mathcal{B} = \langle \Sigma, B, \delta_{\mathcal{B}}, B_0, \alpha_{\mathcal{B}} \rangle$. We define a function $f : 2^B \to 2^B$ by $f(E) = \{b : \langle b, E \rangle \in \alpha_{\mathcal{C}}\}$. Thus, when the subset component of $\mathcal{D}$ is in state E, it should continue and check the membership in $\alpha_{\mathcal{C}}$ only for states in $f(E)$. We define the DCW $\mathcal{D} = \langle \Sigma, D, \delta_{\mathcal{D}}, D_0, \alpha_{\mathcal{D}} \rangle$ as follows.

- $D = \{\langle S, O \rangle : S \subseteq B \text{ and } O \subseteq S \cap f(S)\}$.
- For all $\langle S, O \rangle \in D$ and $\sigma \in \Sigma$, the transition function is defined as follows.
  - If $O \neq \emptyset$, then $\delta_D(\langle S, O \rangle, \sigma) = \{\langle \delta_{\mathcal{B}}(S, \sigma), \delta_{\mathcal{B}}(O, \sigma) \cap f(S) \rangle\}$.
  - If $O = \emptyset$, then $\delta_D(\langle S, O \rangle, \sigma) = \{\langle \delta_{\mathcal{B}}(S, \sigma), \delta_{\mathcal{B}}(S, \sigma) \cap f(S) \rangle\}$.
- $D_0 = \{\langle B_0, \emptyset \rangle\}$.
- $\alpha_{\mathcal{D}} = \{\langle S, O \rangle : O \neq \emptyset\}$.

Thus, the run of $\mathcal{D}$ on a word w has to visit states in $2^B \times \{\emptyset\}$ only finitely often, which holds iff some run of $\mathcal{C}$ on w eventually always visits $\alpha_{\mathcal{C}}$. Since each state of D corresponds to a function from B to the set of size three $\{$ "in $S \cap O$", "in $S \setminus O$", "not in S"$\}$, its number of states is at most 3^B. □

We refer to the construction in Theorem 4.1 as the *breakpoint construction*. We note that the exponential lower bound for determinization of automata on finite words implies that our upper bound is asymptotically tight. On the other hand, as with the breakpoint construction in [20], the problem of whether the 3^n bound can be improved to a 2^n bound is left open.

5 Applications

In this section we describe immediate applications of our NBW to NCW (Theorem 3.5) and NBW to DBW (Theorem 4.1) constructions. The goal of these applications is to simplify procedures that currently involve determinization, either by using an NCW instead of a deterministic Büchi or parity automaton, or by using a DBW instead of a deterministic parity automaton. Note that both constructions are based on the subset constructions, have a simple state space, and amenable to optimizations, and can be implemented symbolically [21].

Theorems 3.5 and 4.1 guarantee that if the given NBW is in NCW, then the constructions result in equivalent automata. We first show that if this is not the case, then the constructions have only a one-sided error. As we shall see below, this would become handy in many of the applications.

Lemma 5.1 *For an NBW $\mathcal{B}$, let $\mathcal{C}$ be the NCW obtained by the augmented subset construction, and let $\mathcal{D}$ be the DBW obtained from $\mathcal{B}$ by applying the breakpoint construction. Then, $L(\mathcal{B}) \subseteq L(\mathcal{C})$ and $L(\mathcal{B}) \subseteq L(\mathcal{D})$.*

Proof: It is easy to see that the proof of the $L(\mathcal{B}) \subseteq L(\mathcal{C})$ direction in Theorem 3.5, as well as the equivalence of $\mathcal{C}$ and $\mathcal{D}$ in Theorem 4.1, do not rely on the assumption that $\mathcal{B}$ is in NCW. □

We now turn to describe the applications.

5.1 Deciding membership in NCW and DBW

As discussed in Section 1, NCWs are less expressive than NBWs, and they recognize exactly all languages that complement languages in DBW. Motivated by the connections among DBW, alternation-free μ-calculus (AFMC), and alternating weak tree automata, [13] studies the problem of deciding whether an NBW or an LTL formula is in DBW. The solution in [13] involves determinization: given an NBW $\mathcal{B}$, translates $\mathcal{B}$ to an equivalent deterministic Rabin automaton $\mathcal{R}$, and then check whether $\mathcal{R}$ is in DBW. The latter check can be done in linear time. Deciding whether an LTL formula is in DBW has applications beyond deciding its membership in AFMC. Indeed, there is no reason to work with complicated deterministic automata in cases a DBW is sufficiently strong.

Below we describe a decision procedure that avoids the determinization. The procedure is based on the following lemma, which is an easy corollary of Theorem 3.5 and Lemma 5.1.

Lemma 5.2 *Consider an NBW $\mathcal{B}$. Let $\mathcal{C}$ be its augmented-subset construction. Then, $\mathcal{B}$ is in NCW iff $L(\mathcal{C}) \subseteq L(\mathcal{B})$.*

We start with a procedure for deciding whether a given NBW $\mathcal{B}$ is in NCW. By Lemma 5.2, the latter can be reduced to checking whether $L(\mathcal{C}) \subseteq L(\mathcal{B})$, for the augmented subset construction $\mathcal{C}$ of $\mathcal{B}$. This involves the generation of $\mathcal{C}$ and of the complement $\tilde{\mathcal{B}}$ of $\mathcal{B}$, and checking the emptiness of the product $C \times \tilde{\mathcal{B}}$. Since complementation can be done without determinization [12], we are done. For

[6] Readers familiar with the construction of [20] may find it easier to view the construction here as one that dualizes a translation of universal co-Büchi automata to deterministic Büchi automata, going through universal Büchi word automata – these constructed by dualizing Theorem 3.5.

$\mathcal{B}$ with n states, the complexity is $2^{O(n \log n)}$. We note that in addition to avoiding determinization, the constants in the $O()$ notation are much better than these in the procedure that involves determinization [12].

Now, in order to decide whether a given LTL formula is in NCW or in DBW, we proceed as follows. Given an LTL formula ψ, let $\mathcal{B}_\psi$ be an NBW for ψ, and let $\mathcal{C}_\psi$ be its augmented subset construction. By Lemma 5.2, the formula ψ is in NCW iff $L(\mathcal{C}_\psi) \subseteq L(\mathcal{B}_\psi)$. The latter can be reduced to checking the emptiness of the intersection of $\mathcal{C}_\psi$ with the complement of $\mathcal{B}_\psi$. Fortunately, ψ is given, so rather than complementing $\mathcal{B}_\psi$, we can generate and use $\mathcal{B}_{\neg\psi}$ instead. Thus, our procedure not only avoids determinization, but also it does not involve complementation.

Now, since ψ is in DBW iff $\neg\psi$ is in NCW, we can use the above procedure also for checking whether a given LTL formula is in DBW. As above, complementation can be avoided.[7]

5.2 Translating LTL to AFMC

Consider an LTL formula ψ. By [13], there is an AFMC formula equivalent to $\forall\psi$ iff ψ is in DBW. As described in Section 5.1, we can first check whether ψ can be translated to AFMC. Let $\mathcal{C}_{\neg\psi}$ be the augmented subset construction of the NBW $\mathcal{B}_{\neg\psi}$ for $\neg\psi$ [30]. By Lemma 5.1, we have that $L(\mathcal{C}_{\neg\psi})$ accepts exactly all computations that violate ψ. By [13], we can expand $\mathcal{C}_{\neg\psi}$ existentially to an AFMC formula θ that is satisfied in exactly all systems that have a computation that violates ψ. The AFMC formula $\neg\theta$ is then satisfied in exactly all systems all of whose computations satisfy ψ. Thus, using the augmented subset construction, we have generated the AFMC formula without determinization.

Note that while the translation avoids determinization, it is still doubly-exponential. In model checking, the system is much bigger than the specification, and its size is typically the computational bottleneck. Therefore, the fact that evaluation of AFMC formulas can be done in linearly many symbolic steps (in both the system and the specification) makes the translation appealing in practice. We note that no matching lower bound is known, and the blow-up in the translation of LTL to AFMC is still open. [8]

[7] A different Safraless procedure for deciding whether a specification can be translated to a DBW is described in [14]. The procedure there, however, requires NBWs of both the specification and its negation, and using it in order to check whether a given NBW is in NCW requires complementation. In addition, the procedure in [14] involves tree automata and is more complicated.

[8] Wilke [31] proved an exponential lower-bound for the translation of an NBW for an LTL formula ψ to an AFMC formula equivalent to $\forall\psi$. This lower-bound does not preclude a polynomial upper-bound for the translation of an NBW for $\neg\psi$ to an AFMC formula equivalent to $\exists\neg\psi$, which would imply an exponential upper bound for the translation of LTL to AFMC.

5.3 Translating LTL to DBW

As discussed in Section 1, numerous applications of automata theory in practice require determinization. The intricacy of optimal determinization constructions have led to a situation in which many of these applications are not implemented in practice, or are restricted to safety properties, for which determinization is easy and is based on the subset construction. Our NBW to DCW construction in Theorem 4.1 immediately extends the scope of these applications to all LTL specifications in DBW. In particular, synthesis, control, game solving, probabilistic model checking, and monitoring-based run-time verification can be now performed for LTL specifications in DBW with a simple determinization construction. The same holds for CTL* satisfiability, for specifications whose path formulas are in DBW.

Given an LTL formula ψ, an input to one of these procedures, one can use the decision procedure described in Section 5.1 in order to check whether ψ is in DBW, in which case our simple determinization construction can be applied to it. Formally, we have the following. For an LTL formula ψ, let $L(\psi)$ denote the set of computations satisfying ψ. The following is an easy corollary of the duality between DBW and DCW.

Lemma 5.3 *For an LTL formula ψ in DBW, let $\mathcal{B}_{\neg\psi}$ be an NBW accepting $L(\neg\psi)$, and let $\mathcal{D}_\psi$ be the DBW obtained by dualizing the breakpoint construction of $\mathcal{B}_{\neg\psi}$. Then, $L(\mathcal{D}_\psi) = L(\psi)$.*

5.4 Using the one-sided error

The applications above have been described for specifications in NCW or DBW. In this section we argue that the one-sided error of the construction enables us to use it also for arbitrary specifications.

Given an LTL formula ψ, let $\mathcal{C}_{\neg\psi}$ be the augmented subset construction of the NBW $\mathcal{B}_{\neg\psi}$ for $\neg\psi$. By Lemma 5.1, we have that $L(\mathcal{C}_{\neg\psi})$ accepts all words that violate ψ. Let θ be an AFMC formula obtained by expanding $\mathcal{C}_{\neg\psi}$ existentially. Thus, θ is satisfied in a system $\mathcal{S}$ iff $\mathcal{S}$ has a computation accepted by $\mathcal{C}_{\neg\psi}$. Given a system $\mathcal{S}$, we can still model check $\mathcal{S}$ with respect to θ. If θ is not satisfied, we can conclude that no computation violates ψ. Indeed, $\mathcal{C}_{\neg\psi}$ overapproximates $\mathcal{B}_{\neg\psi}$. If θ is satisfied, we can check whether the witness to the satisfaction indeed violates ψ. If it does, we found a counterexample and we are done. If not, we can conclude that ψ is not in DBW and either look for another counterexample using θ, or use standard methods.

Now, for the applications that translate LTL formulas to deterministic automata, Lemma 5.1 implies that when applied to arbitrary LTL formulas, the automaton $\mathcal{D}_\psi$ constructed in Lemma 5.3 is such that $L(\mathcal{D}_\psi) \subseteq L(\psi)$. The polarity of the error (that is, $\mathcal{D}_\psi$ underapproximates ψ) is

the helpful one. Indeed, for the purpose of monitoring, a computation that is a member of $L(\mathcal{D}_\psi)$ surely satisfies ψ. Only in case an error is detected, one should check whether ψ is violated along it. For the purpose of CTL* satisfiability, consider a tree automaton for a CTL* formula in positive normal form in which the deterministic automata for the path formulas are approximated by DBWs. Clearly, if the automaton is not empty, then a witness to the satisfiability exists. Finally, we can solve the synthesis problem using $\mathcal{D}_\psi$. If we get a transducer that realizes $\mathcal{D}_\psi$, we know that it also realizes ψ, and we are done. Moreover, as suggested in [14], in case $\mathcal{D}_\psi$ is unrealizable, we can check, again using an approximating DBW, whether $\neg\psi$ is realizable for the environment. Only if both ψ is unrealizable for the system and $\neg\psi$ is unrealizable for the environment, we need precise realizability. Note that then, we can also conclude that ψ is not in DBW.

References

[1] Accellera Organization Inc. http://www.accellera.org, 2006.

[2] B. Aminof, O. Kupferman, and O. Lev. On the relative succinctness of nondeterministic Büchi and co-Büchi word automata. In *Proc. 15th LPAR, LNCS 5330, pages 183–197*, 2008.

[3] B. Boigelot, S. Jodogne, and P. Wolper. On the use of weak automata for deciding linear arithmetic with integer and real variables. In *Proc. IJCAR*, LNCS 2083, pages 611–625, 2001.

[4] J.R. Büchi. On a decision method in restricted second order arithmetic. In *Proc. Int. Congress on Logic, Method, and Philosophy of Science. 1960*, pages 1–12. Stanford University Press, 1962.

[5] E.A. Emerson and C. Jutla. The complexity of tree automata and logics of programs. In *Proc. 29th FOCS*, pages 328–337, 1988.

[6] T.A. Henzinger and N. Piterman. Solving games without determinization. In *Proc. 15th CSL*, LNCS 4207, pages 394–410, 2006.

[7] O. Kupferman, Y. Lustig, and M.Y. Vardi. On locally checkable properties. In *Proc. 13th LPAR*, LNCS 4246, pages 302–316, 2006.

[8] D. Kähler and T. Wilke. Complementation, disambiguation, and determinization of Büchi automata unified. In *Proc. 35th ICALP*, LNCS 525, pages 724–735, 2008.

[9] O. Kupferman. Avoiding determinization. In *Proc. 21st LICS*, pages 243–254, 2006.

[10] O. Kupferman. Tightening the exchange rate beteen automata. In *Proc. 16th CSL*, LNCS 4646, pages 7–22, 2007.

[11] O. Kupferman, G. Morgenstern, and A. Murano. Typeness for ω-regular automata. In *Proc. 2nd ATVA*, LNCS 3299, pages 324–338, 2004.

[12] O. Kupferman and M.Y. Vardi. Weak alternating automata are not that weak. *ACM ToCL*, 2(2):408–429, 2001.

[13] O. Kupferman and M.Y. Vardi. From linear time to branching time. *ACM ToCL*, 6(2):273–294, 2005.

[14] O. Kupferman and M.Y. Vardi. Safraless decision procedures. In *Proc. 46th FOCS*, pages 531–540, 2005.

[15] R.P. Kurshan. *Computer Aided Verification of Coordinating Processes*. Princeton Univ. Press, 1994.

[16] L.H. Landweber. Decision problems for ω–automata. *Mathematical Systems Theory*, 3:376–384, 1969.

[17] C. Löding. Optimal bounds for the transformation of omega-automata. In *Proc. 19th FST& TCS*, LNCS 1738, pages 97–109, 1999.

[18] R. McNaughton. Testing and generating infinite sequences by a finite automaton. *I& C*, 9:521–530, 1966.

[19] M. Michel. Complementation is more difficult with automata on infinite words. CNET, Paris, 1988.

[20] S. Miyano and T. Hayashi. Alternating finite automata on ω-words. *TCS*, 32:321–330, 1984.

[21] A. Morgenstern and K. Schneider. From LTL to Symbolically Represented Deterministic Automata. In *Proc. 9th VMCAI*, LNCS 4905, pages 279 – 293, 2008.

[22] D.E. Muller, A. Saoudi, and P.E. Schupp. Alternating automata, the weak monadic theory of the tree and its complexity. In *Proc 13th ICALP*, LNCS 226, pages 275–283, 1986.

[23] N. Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. In *Proc. 21st LICS*, pages 255-264, 2006.

[24] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. 16th POPL*, pages 179–190, 1989.

[25] M.O. Rabin. Decidability of second order theories and automata on infinite trees. *Transaction of the AMS*, 141:1–35, 1969.

[26] K. Ravi, R. Bloem, and F. Somenzi. A comparative study of symbolic algorithms for the computation of fair cycles. In *Proc. 3rd FMCAD*, LNCS 1954, pages 143–160, 2000.

[27] S. Safra. On the complexity of ω-automata. In *Proc. 29th FOCS*, pages 319–327, 1988.

[28] R.S. Street and E.A. Emerson. An elementary decision procedure for the μ-calculus. In *Proc. 11th ICALP*, LNCS 172, pages 465–472, 1984.

[29] M.Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *JCSS*, 32(2):182–221, 1986.

[30] M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *I& C*, 115(1):1–37, 1994.

[31] T. Wilke. CTL$^+$ is exponentially more succinct than CTL. In *Proc. 19th FST & TCS*, LNCS 1738, pages 110–121, 1999.

[32] P. Wolper, M.Y. Vardi, and A.P. Sistla. Reasoning about infinite computation paths. In *Proc. 24th FOCS*, pages 185–194, 1983.

[33] Q. Yan. Lower bounds for complementation of ω-automata via the full automata technique. In *Proc. 33rd ICALP*, LNCS 4052, pages 589–600, 2006.