Translating to Co-Büchi Made Tight, Unified, and Useful

UDI BOKER, Hebrew University and IST Austria
ORNA KUPFERMAN, Hebrew University

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 continue and solve the open problems of translating nondeterministic Streett, Rabin, Muller, and parity word automata to NCW and to DCW. Going via an intermediate NBW is not optimal and we describe direct, simple, and asymptotically tight constructions, involving a $2^{\Theta(n)}$ blow-up. The constructions are variants of the subset construction, providing a unified approach for translating all common classes of automata to NCW and DCW.

Beyond the theoretical importance of the results, 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.

Categories and Subject Descriptors: F.1.1 [**Computation by Abstract Devices**]: Models of Computation—*Automata*; F.1.2 [**Computation by Abstract Devices**]: Models of Computation—*Alternation and nondeterminism*; F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic—*Temporal logic*; F.4.3 [**Mathematical Logic and Formal Languages**]: Formal Languages

General Terms: Verification, Theory, Algorithms

Additional Key Words and Phrases: Formal verification, Model checking, Nondeterminism, Büchi automata, co-Büchi 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 [Büchi 1962; McNaughton 1966; Rabin 1969]. 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 [Kurshan 1994; Vardi and Wolper 1994]. Recent industrial-strength property-specification languages such as Sugar, ForSpec, and the recent standard PSL 1.01 include regular expressions and/or automata, mak-

Authors' address: School of Engineering and Computer Science, Hebrew University, Jerusalem 91904, Israel; IST Austria, Klosterneuburg, Austria.
The present article combines and extends [Boker and Kupferman 2009; 2011].
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© YYYY ACM 1529-3785/YYYY/01-ARTA $10.00
DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

ing specification and verification tools that are based on automata even more essential and popular [Accellera 2006].

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 [Wolper et al. 1983; Street and Emerson 1984; Vardi and Wolper 1986; Emerson and Jutla 1988; Safra 1988; Pnueli and Rosner 1989]. 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 [Safra 1988; Muller and Schupp 1995] or parity [Piterman 2006; Kähler and Wilke 2008] automaton [Kupferman and Vardi 2005b], and involves a super-exponential blow-up. 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 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 an NCW.[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 [Kupferman et al.

[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 [Büchi 1962]; 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 [Miyano and Hayashi 1984], of alternating Büchi automata into NBW, results in DBW. By dualizing it, one gets a translation of NCW to DCW.

[3] A general technique for proving lower bounds on the size of automata on infinite words is suggested in [Yan 2006]. 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.

2004], 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 [Aminof et al. 2008].

Let us now turn to the problem of translating NBW to DCW. 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 [Michel 1988; Löding 1999]. 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 [Miyano and Hayashi 1984], 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 [Kupferman 2007], the translations of NBW to NCW and DCW have immediate important applications in formal methods. The first class of applications uses the NCW directly. The premier example in this class is of symbolic linear temporal logic (LTL) model checking. LTL model checking reduces to a search for bad-cycles, whose symbolic implementation involves nested fixed-points, and is typically quadratic [Ravi et al. 2000]. An alternative approach is to translate the LTL formula to the alternation free μ-calculus (AFMC). Evaluating AFMC formulas can be done with linearly many symbolic steps. It is shown in [Kupferman and Vardi 2005b] that given an LTL formula ψ, there is an AFMC formula equivalent to $\forall\psi$ iff ψ can be recognized by a DBW. Moreover, 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 [Safra 1988; Muller and Schupp 1995; Piterman 2006; Kähler and Wilke 2008], 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.

[4]For some settings of these applications, procedures that avoid determinization have already been suggested [Kupferman and Vardi 2001; 2005c; Henzinger and Piterman 2006; Kupferman 2006; Schewe and Finkbeiner 2007]. 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.

In the first part of 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 [Morgenstern and Schneider 2008]. This construction, named *the augmented subset construction*, turns out to have interesting properties and additional applications, as will be detailed in this sequel.

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 $n \geq 2$, there is a language L_n over a four-letter alphabet, such that L_n can be recognized by an NBW with $O(n)$ states, whereas the minimal NCW that recognizes L_n has $n2^n$ states. This leads to an asymptotically 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 $\mathcal{C}$ yields the same state space as if applying it directly on the NBW $\mathcal{B}$. 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 [Miyano and Hayashi 1984], and it can be implemented symbolically [Morgenstern and Schneider 2008]. 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 [Kupferman and Vardi 2005b].

Having solved the problem of translating NBWs to NCWs and DCWs, we turn on to study stronger acceptance conditions, and consider the translation of nondeterministic Streett (NSW), Rabin (NRW), parity (NPW), and Muller (NMW) word automata to NCW and to DCW. A straightforward approach is to first translate the stronger automata to NBWs, and then use our new translations. This approach, however, is not optimal. For example, translating an NRW with n states and index k to the intermediate NBW results in an NBW with nk states, thus the NCW would have $nk2^{nk}$ states, with no matching lower bound to the exponential dependency in k. Even more wasteful is the case of NSWs: staring with an NSW with n states and index k, the intermediate NBW has $n2^k$ states, thus the NCW would have $n2^{k+n2^k}$ states, making the dependency in k doubly-exponential. Hence, we seek a direct translation of these stronger classes of automata to NCW and DCW.

We show that for NSW, an equivalent NCW can be defined on top of the augmented subset construction. The definition of the co-Büchi acceptance condition is more involved than that in the case of NBW, but the blow-up stays the same. This immediately provides $n2^n$ and 3^n upper bound for the translation of NSW to NCW and DCW, respectively, when exists. Clearly, the same construction is valid for special cases of the Streett condition, like the parity or the generalized Büchi conditions.

For NRW and NMW, the situation is more complicated. Unfortunately, an equivalent NCW cannot in general be defined on top of the augmented subset construction. Moreover, even though the results on NSW imply a translation of NRW[1] (that is,

a nondeterministic Rabin automaton with a single pair) to NCW, one cannot hope to proceed via a decomposition of an NRW with index k to k NRW[1]s. Indeed, the underlying NRW[1]s may not be NCW-recognizable, even when the NRW is, and the same for NMWs. We show that still, the NCW can be defined on top of k copies of the augmented subset construction, giving rise to a $kn2^n$ upper bound for the translation to NCW. Moreover, we show that when translating to an equivalent DCW, the k copies can be determinized separately, while connected in a round-robin fashion, which gives rise to a $k3^n$ blow-up. As with the other cases, the blow-up involved in the translations is asymptotically tight. The state blow-up involved in the various translations is summarized in Table I of the Section 7.

Our improved upper bounds immediately imply the applications discussed above. We elaborate on the applications further in Section 6. An important and useful property of our constructions is the fact they have only a one-sided error when applied to automata whose language is not NCW-recognizable. Thus, given an automaton $\mathcal{A}$, the NCW $\mathcal{C}$ and the DCW $\mathcal{D}$ we construct are such that $L(\mathcal{A}) \subseteq L(\mathcal{C}) = L(\mathcal{D})$ (and $L(\mathcal{A}) = L(\mathcal{C}) = L(\mathcal{D})$ in case $\mathcal{A}$ 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 6, this enables us to extend the scope of the applications also to specifications that are not NCW-recognizable.

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 α 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 for a set of states S and a finite word x, we have that $\delta(S, x)$ is the set of states that $\mathcal{A}$ may move into when it is in a state in S and it reads 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 q of $\mathcal{A}$, we denote by $\mathcal{A}^q$ the automaton that is identical to $\mathcal{A}$, except for having $\{q\}$ as its set of initial states. An automaton without an acceptance condition is called a *semi-automaton*.

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, $inf(r) = \{q \in Q \mid \text{ for infinitely many } i \geq 0, \text{ we have } r_i = q\}$. 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.

Several acceptance conditions are studied in the literature. We consider here six:

— *Büchi*, where $\alpha \subseteq Q$, and r is accepting iff $inf(r) \cap \alpha \neq \emptyset$.

— *co-Büchi*, where $\alpha \subseteq Q$, and r is accepting iff $inf(r) \subseteq \alpha$. Note that the definition we use is less standard than the $inf(r) \cap \alpha = \emptyset$ definition; clearly, $inf(r) \subseteq \alpha$ iff $inf(r) \cap (Q \setminus \alpha) = \emptyset$, thus the definitions are equivalent. We chose to go with this variant as it better conveys the intuition that, as with the Büchi condition, a visit in α is a "good event".
— *parity*, where $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_{2k}\}$ with $\alpha_1 \subset \alpha_2 \subset \cdots \subset \alpha_{2k} = Q$, and r is accepting if the minimal index i for which $inf(r) \cap \alpha_i \neq \emptyset$ is even.
— *Rabin*, where $\alpha = \{\langle \alpha_1, \beta_1 \rangle, \langle \alpha_2, \beta_2 \rangle, \ldots, \langle \alpha_k, \beta_k \rangle\}$, with $\alpha_i, \beta_i \subseteq Q$ and r is accepting iff for some $1 \leq i \leq k$, we have that $inf(r) \cap \alpha_i \neq \emptyset$ and $inf(r) \cap \beta_i = \emptyset$.
— *Streett*, where $\alpha = \{\langle \beta_1, \alpha_1 \rangle, \langle \beta_2, \alpha_2 \rangle, \ldots, \langle \beta_k, \alpha_k \rangle\}$, with $\beta_i, \alpha_i \subseteq Q$ and r is accepting iff for all $1 \leq i \leq k$, we have that $inf(r) \cap \beta_i = \emptyset$ or $inf(r) \cap \alpha_i \neq \emptyset$.
— *Muller*, where $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$, with $\alpha_i \subseteq Q$ and r is accepting iff for some $1 \leq i \leq k$, we have that $inf(r) = \alpha_i$.

The number of sets in the parity and Muller acceptance conditions or pairs in the Rabin and Streett acceptance conditions is called the *index* of the automaton. An automaton accepts a word if 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 $\{\mathrm{D}, \mathrm{N}\} \times \{\mathrm{B}, \mathrm{C}, \mathrm{P}, \mathrm{R}, \mathrm{S}, \mathrm{M}\} \times \{\mathrm{W}\}$. The first letter stands for the branching mode of the automaton (deterministic or nondeterministic); the second letter stands for the acceptance-condition type (Büchi, co-Büchi, parity, Rabin, Streett, or Muller); and the third letter indicates that the automaton runs on words. We say that a language L is γ-*recognizable* if 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 languages [McNaughton 1966], DBWs are strictly less expressive than NBWs, and so are DCWs [Landweber 1969]. In fact, a language L is in DBW iff its complement is in DCW. Indeed, by viewing a DBW as a DCW and switching between accepting and non-accepting states, 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 [Miyano and Hayashi 1984]. As for parity, Rabin, Streett and Muller automata, both the deterministic and nondeterministic models recognize all ω-regular languages [Thomas 1990].

3. FROM NBW TO NCW

In this section we present our improved upper and lower bounds for the translation, when possible, of NBWs to NCWs. 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 n states whose language is NCW-recognizable, there is an equivalent NCW $\mathcal{C}$ with at most $n2^n$ 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!

Definition 3.1 (*Augmented subset construction*). Let $\mathcal{A} = \langle \Sigma, A, \delta_{\mathcal{A}}, A_0 \rangle$ be a semi-automaton (an automaton without an acceptance condition). We define its *augmented subset construction* $\mathcal{A}'$ as the product of $\mathcal{A}$ with its subset construction. Formally, $\mathcal{A}' = \langle \Sigma, A', \delta_{\mathcal{A}'}, A'_0 \rangle$, where

— $A' = A \times 2^A$. That is, the states of $\mathcal{A}'$ are all the pairs $\langle a, E \rangle$ where $a \in A$ and $E \subseteq A$.
— For all $\langle a, E \rangle \in A'$ and $\sigma \in \Sigma$, we have $\delta_{\mathcal{A}'}(\langle a, E \rangle, \sigma) = \delta_{\mathcal{A}}(a, \sigma) \times \{\delta_{\mathcal{A}}(E, \sigma)\}$. That is, $\mathcal{A}'$ nondeterministically follows $\mathcal{A}$ on its A-components and deterministically follows the subset construction of $\mathcal{A}$ on its 2^A-component.
— $A'_0 = A_0 \times \{A_0\}$.

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 automaton.

LEMMA 3.2. *Consider an automaton $\mathcal{A}$ with a transition function $\delta_{\mathcal{A}}$ and a DCW $\mathcal{D}$ with a transition function $\delta_{\mathcal{D}}$ such that $L(\mathcal{A}) = L(\mathcal{D})$. Let d_1 and d_2 be states of $\mathcal{D}$ such that there are two finite words x_1 and x_2 such that $\delta_{\mathcal{D}}(x_1) = d_1$, $\delta_{\mathcal{D}}(x_2) = d_2$, and $\delta_{\mathcal{A}}(x_1) = \delta_{\mathcal{A}}(x_2)$. Then, $L(\mathcal{D}^{d_1}) = L(\mathcal{D}^{d_2})$.*

PROOF. We prove that $L(\mathcal{D}^{d_1}) \subseteq L(\mathcal{D}^{d_2})$. The other direction is analogous. Consider a word $w \in L(\mathcal{D}^{d_1})$. Since $\delta_{\mathcal{D}}(x_1) = d_1$, we have that $x_1 \cdot w \in L(\mathcal{A})$. Let r be the accepting run of $\mathcal{A}$ on $x_1 \cdot w$. Since $\delta_{\mathcal{A}}(x_1) = \delta_{\mathcal{A}}(x_2)$, there is a run r' of $\mathcal{A}$ on $x_2 \cdot w$ such that $r'(x_2) = r(x_1)$ 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 $x_2 \cdot w \in L(\mathcal{A})$. Hence, $x_2 \cdot w \in L(\mathcal{D})$. Since $\delta_{\mathcal{D}}(x_2) = d_2$, it follows that $w \in L(\mathcal{D}^{d_2})$, 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. We show it for DCW, yet it holds for every deterministic automaton.

LEMMA 3.3. *Consider a DCW $\mathcal{D} = \langle \Sigma, D, \delta, D_0, \alpha \rangle$. Let d_1 and d_2 be states in D such that $L(\mathcal{D}^{d_1}) = L(\mathcal{D}^{d_2})$. For all finite words u and v, if $\delta(d_1, u) = d_1$ and $\delta(d_2, v) = d_2$ then for all words $w \in (u + v)^*$ and states $d' \in \delta(d_1, w) \cup \delta(d_2, w)$, we have $L(\mathcal{D}^{d'}) = L(\mathcal{D}^{d_1})$.*

PROOF. We prove that $L(\mathcal{D}^{d'}) = L(\mathcal{D}^{d_1})$ for states $d' \in \delta(d_1, w)$. The proof for states $d' \in \delta(d_2, w)$ is analogous.

For every word $w \in (u+v)^*$ there is an index $n \geq 0$ such that $w \in (u+v)^n$. We prove by induction on n that $L(\mathcal{D}^{d_1}) = L(\mathcal{D}^{d'})$ for every $d' \in \delta(d_1, w)$. The base case, in which $n = 0$, is obvious, as $d' = d_1$. Assume that the claim holds for n and consider a word $w \in (u+v)^{n+1}$. Assume that $w = x \cdot u$, for $x \in (u+v)^n$. By the induction hypothesis, the state $d'' = \delta(d_1, x)$ is such that $L(\mathcal{D}^{d_1}) = L(\mathcal{D}^{d''})$.

Consider a word $y \in L(\mathcal{D}^{d_1})$. We claim that $y \in L(\mathcal{D}^{d'})$. Indeed, since $\delta(d_1, u) = d_1$, we have that $u \cdot y \in L(\mathcal{D}^{d_1})$. 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 $y \in L(\mathcal{D}^{d_1})$. Indeed, since $d' = \delta(d'', u)$, we have that $u \cdot y \in L(\mathcal{D}^{d''})$. Thus, by the induction hypothesis, $u \cdot y \in L(\mathcal{D}^{d_1})$. Therefore, since $\delta(d_1, u) = d_1$, we get that $y \in L(\mathcal{D}^{d_1})$, as required.

The other case, in which $w = x \cdot v$, is proved analogously, having d_2 in the role of d_1. □

Our next observation is the key to the definition of the acceptance condition on top of the augmented subset construction. Intuitively, it shows that if an NCW-recognizable 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 3.4. *Consider an NCW-recognizable language L. For all finite words u and v, if $(v^* \cdot u^+)^\omega \subseteq L$ and for every finite word $x \in (u+v)^*$ and infinite word w we have that $w \in L$ iff $x \cdot w \in L$, 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 3.4 implies the following.

COROLLARY 3.5. *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 $(v^* \cdot u^+)^\omega \subseteq L(\mathcal{D}^d)$ and for all words $w \in (v+u)^*$ and states $d' \in \delta(d, w)$, we have $L(\mathcal{D}^{d'}) = L(\mathcal{D}^d)$, then $v^\omega \in L(\mathcal{D}^d)$.*

We can now present our construction together with its acceptance condition. An example of the construction is provided in Figure 1.

THEOREM 3.6. *For every NBW $\mathcal{B}$ with n states that is NCW-recognizable there is an equivalent NCW $\mathcal{C}$ with at most $n2^n$ states.*

PROOF. Let $\mathcal{B} = \langle \Sigma, B, \delta_\mathcal{B}, B_0, \alpha_\mathcal{B} \rangle$. We define the NCW $\mathcal{C} = \langle \Sigma, C, \delta_\mathcal{C}, C_0, \alpha_\mathcal{C} \rangle$ as the augmented subset construction of $\mathcal{B}$ with the following acceptance condition: 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 $d_2 = \rho(x \cdot z^{i_1})$.

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 $d_1 \in D$ 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 $\eta(p_i) = d_1$.

We claim that the states d_1 and d_2 satisfy the conditions of Lemma 3.2 with x_1 being p_1 and x_2 being $x \cdot z^{i_1}$. Indeed, $\delta_{\mathcal{D}}(p_1) = d_1$, $\delta_{\mathcal{D}}(x \cdot z^{i_1}) = d_2$, 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 3.2, we have $L(\mathcal{D}^{d_1}) = L(\mathcal{D}^{d_2})$.

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_1 satisfies the conditions of Corollary 3.5 with u being $z^{i_2 - i_1}$ and v being t_i. The second condition is satisfied by Lemma 3.3. For the first condition, consider a word $w' \in (v^* \cdot u^+)^\omega$. We prove that $w' \in L(\mathcal{D}^{d_1})$. 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 $\eta(p_1) = d_1$. Hence, $p_1 \cdot w' \in L(\mathcal{B})$. Since $L(\mathcal{B}) = L(\mathcal{D})$, we have that $p_1 \cdot w' \in L(\mathcal{D})$. Therefore, $w' \in L(\mathcal{D}^{d_1})$.

Thus, by Corollary 3.5, for all $i \geq 1$ we have that $t_i^\omega \in L(\mathcal{D}^{d_1})$. Since $\delta_{\mathcal{D}}(d_1, t_i) = d_1$, it follows that all the states that $\mathcal{D}$ visits when it reads t_i from d_1 are in $\alpha_{\mathcal{D}}$. Note that $w = p_1 \cdot t_1 \cdot t_2 \cdots$. Hence, since $\delta_{\mathcal{D}}(p_1) = d_1$, 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. □

Remark 3.7. The best known translation of an NBW to an NCW for the complementing language (when exists) is super-exponential and results in a DCW. Indeed, the translation goes via an intermediate deterministic Streett [Safra 1988] or parity [Piterman 2006; Kähler and Wilke 2008] automaton for the complementing language. Thus, starting with an NBW with n states, we end up with a DCW with $2^{O(n \log n)}$ states. One may wonder whether the ideas in this section could be used in order to come up with a $2^{O(n)}$ translation of an NBW to a complementing NCW (when exists).

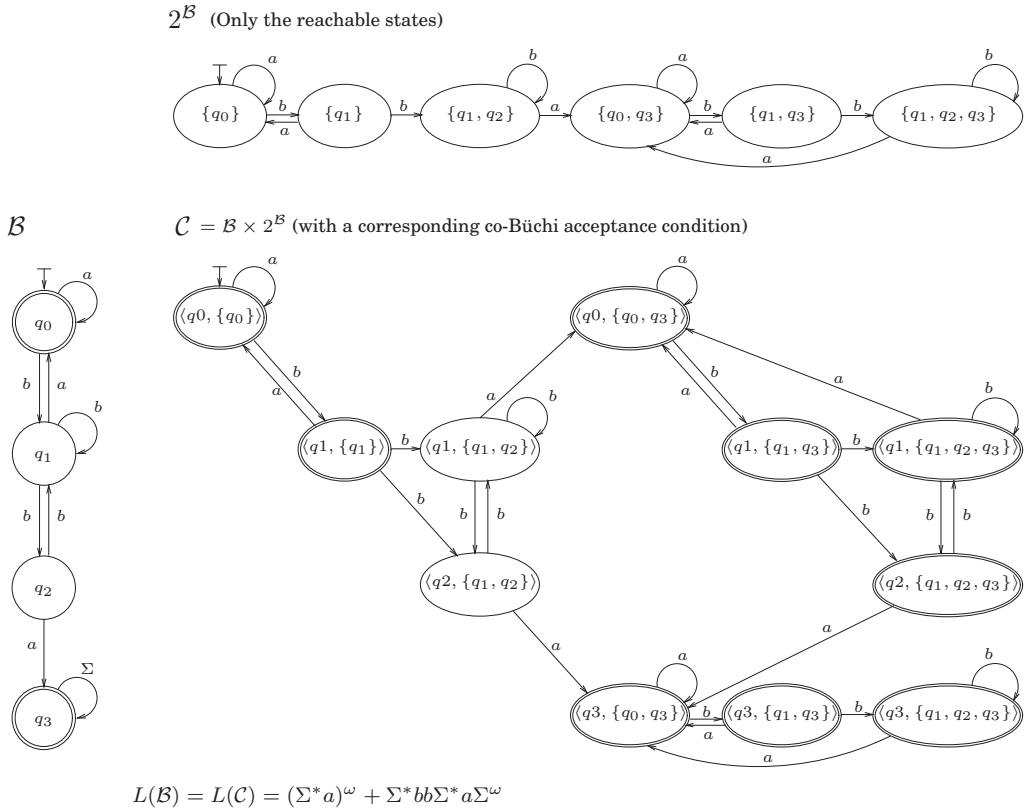

Fig. 1. Translating the NBW $\mathcal{B}$ to the NCW $\mathcal{C}$ via the augmented subset construction.

In fact, a $2^{O(n \log n)}$ lower bound for this problem is quite straightforward: It is not hard to see that the languages used by Michel in order to prove a $2^{O(n \log n)}$ lower bound for NBW complementation are DBW-recognizable [Michel 1988], thus their complementing languages are NCW-recognizable. In addition, every NCW can be translated to an equivalent NBW with at most twice as many states. Hence, if we assume by way of contradiction that there is a $2^{O(n)}$ translation of an NBW to a complementing NCW (when exists), then applying it to the languages used by Michel, and then translating the obtained NCWs to equivalent NBWs, would result in complementing NBWs with only $2^{O(n)}$ states, contradicting Michel's lower bound.

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 $n \geq 2$, an NBW for L_n has $O(n)$ states whereas an NCW for L_n requires at least $n2^n$ states.

Let $\Sigma = \{0, 1, \$, \#\}$. The language L_n is going to be the union of a language L'_n with the language $(\Sigma^* \cdot \#)^\omega$. Before we define L'_n formally, we describe the intuition behind it. Our alert readers are probably bothered by the fact the $(\Sigma^* \cdot \#)^\omega$ component of L_n is not NCW-recognizable. Thus, one task of L'_n is to neutralize the non NCW-recognizability of this component. The way to do this would be to have a finite bound $th(n)$ (the *threshold* of n) such that L'_n contains all words in $(\Sigma^* \cdot \#)^\omega$ that have a

subword $(0+1+\$)^h$, for $h > th(n)$. Accordingly, the language L_n is also the union of L'_n with the language $(\Sigma^{\leq th(n)} \cdot \#)^\omega$, 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_n can use its $L'_n \cup (\Sigma^* \cdot \#)^\omega$ definition, an NCW for L_n must use its equivalent $L'_n \cup (\Sigma^{\leq th(n)} \cdot \#)^\omega$ definition, and must therefore count to $th(n)$. Thus, the second task of L'_n is to fulfill the first task with a threshold that is exponential in n. This way, the ability of the NBW to avoid the counting gives it the exponential advantage we are after.

The language L'_n 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 n-bit cyclic counter in which two adjacent values are separated by $\$$. For example, when $n = 3$, a successful attempt might be $100\$101\$110\$111\$000$. Each subword in $(0+1+\$)^*$ of length $(n+1)2^n$ must reach the value 1^n 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 n. The second type is an "improper-increase error", namely a subword $v_i \cdot \$ \cdot v_{i+1} \in (0+1)^n \cdot \$ \cdot (0+1)^n$ such that v_{i+1} is not the successor of v_i in a correct binary encoding of a cyclic n-bit counter. The language L'_n consists of all words that contain the value 1^n or an error, eventually followed by $\#$.

We now define L'_n formally. For $v, v' \in (0+1)^*$, we use $not\_succ_n(v, v')$ to indicate that v and v' are in $(0+1)^n$ but v' is not the successor of v in the binary encoding of a n-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.

— $S_n = \{\$ \cdot (0+1)^m \cdot \$: m < n\} \cup \{(0+1)^m : m > n\}$,
— $I_n = \{v \cdot \$ \cdot v' \mid not\_succ_n(v, v')\}$, and
— $L'_n = \Sigma^* \cdot (S_n \cup I_n \cup \{1^n\}) \cdot \Sigma^* \cdot \# \cdot \Sigma^\omega$.

Now, $L_n = L'_n \cup (\Sigma^* \cdot \#)^\omega$. For example, for $n = 3$, 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 3.8. *For every $n \geq 1$, the language L'_n can be recognized by an NCW with $O(n)$ states and by an NBW with $O(n)$ states.*

PROOF. We show that there is a nondeterministic automaton over finite words (NFW) with $O(n)$ states recognizing $S_n \cup I_n \cup \{1^n\}$. Completing the NFW to an NBW or an NCW for L'_n is straightforward. It is easy to construct NFWs with $O(n)$ states for S_n and for $\{1^n\}$. An NFW with $O(n)$ states for I_n is fairly standard too, nevertheless we define it in detail in Figure 2, for the sake of completeness. A proper binary counting $\bmod 2^n$ changes all the rightmost bits in a binary block until, and including, the first 0. The other bits remain unchanged. An improper counting implies that at least one bit is improperly set. Thus, the NFW can guess which bit is improperly set, and check it. There are exactly four possibilities for the improper bit: 0 changed to 1 (denoted $0 \rightsquigarrow 1$) instead of $0 \rightsquigarrow 0$, $1 \rightsquigarrow 0$ instead of $1 \rightsquigarrow 1$, $0 \rightsquigarrow 0$ instead of $0 \rightsquigarrow 1$, and $1 \rightsquigarrow 1$ instead of $1 \rightsquigarrow 0$. Since the two relevant bits are exactly $n+1$ positions apart, the NFW should only count to $n+1$ and check whether there is a 0 prior to the $\$$ sign. □

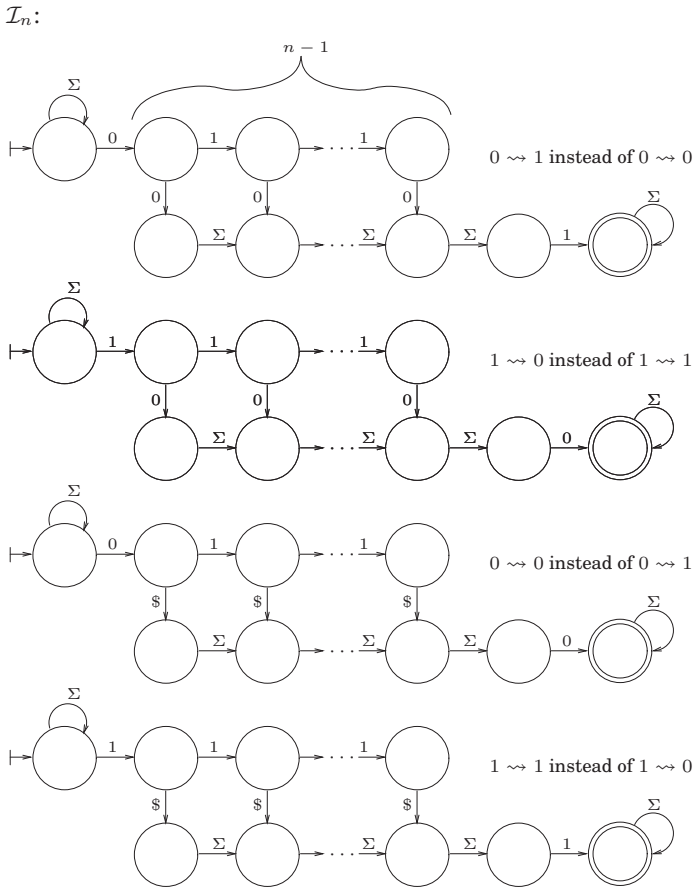

Fig. 2. The finite automaton $\mathcal{I}_n$, recognizing an improper binary counting

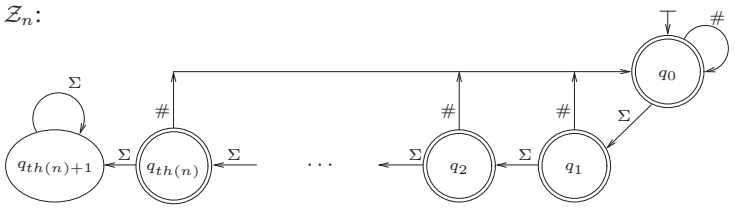

Fig. 3. The NCW $\mathcal{Z}_n$ recognizing infinitely many #'s within bounded distances

An immediate corollary of Lemma 3.8 is that L_n can be recognized by an NBW with $O(n)$ states. Next, we show that while L_n is NCW-recognizable, an NCW for it must be exponentially larger.

LEMMA 3.9. *For every $n \geq 2$, the language L_n is NCW-recognizable, and every NCW recognizing L_n must have at least $n2^n$ states.*

PROOF. We first prove that L_n is NCW-recognizable. Let $th(n) = (n+1)2^n$. Consider the language $Z_n = (\Sigma^{\leq th(n)} \cdot \#)^\omega$. It can be easily verified that Z_n is recognized by the NCW $\mathcal{Z}_n$, defined in Figure 3.

We prove that $L_n = L'_n \cup Z_n$. Since, by Lemma 3.8, the language L'_n is NCW-recognizable, it would follow that L_n is NCW-recognizable. Clearly, $Z_n \subseteq (\Sigma^* \cdot \#)^\omega$. Thus, $L'_n \cup Z_n \subseteq L_n$, and we have to prove that $L_n \subseteq L'_n \cup Z_n$. For that, we prove that $(\Sigma^* \cdot \#)^\omega \subseteq L'_n \cup Z_n$. Consider a word $w \in (\Sigma^* \cdot \#)^\omega$. If $w \in Z_n$, then we are done. Otherwise, w contains a subword $u \in (0+1+\$)^h$, for $h > th(n)$. Thus, either u does not properly encode a n-bit cyclic counter (that is, it contains a syntactic or an improper-increase error) or u has the subword 1^n. Hence, $u \in \Sigma^* \cdot (S_n \cup I_n \cup \{1^n\}) \cdot \Sigma^*$. 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, $w \in L'_n$, and we are done.

We now turn to prove the lower bound. Assume by way of contradiction that there is an NCW $\mathcal{C}_n$ with acceptance set α and at most $n2^n - 1$ states that recognizes L_n. Consider the word $w = (00\cdots0\$00\cdots01\$\cdots\$11\cdots10\#)^\omega$, in which the distance between two consequent $\#$'s is $d = (n+1)(2^n-1)$. Note that for all $n \geq 2$, we have that $d > n2^n$. The word w has infinitely many $\#$'s and it therefore belongs to L_n. Thus, there is an accepting run r of $\mathcal{C}_n$ 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 $\mathcal{C}_n$ has at most $n2^n - 1$ states, there are two positions t_1 and t_2, with $t_0 < t_1 < t_2 \leq t_0 + n2^n$, 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 $\mathcal{C}_n$ 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 $w' \notin L_n$.

Since $t_2 \leq t_0 + n2^n$ and $n2^n < d$, 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 $L_n = L'_n \cup (\Sigma^* \cdot \#)^\omega$. By the above, $w' \notin (\Sigma^* \cdot \#)^\omega$. Furthermore, since $L'_n = \Sigma^* \cdot (S_n \cup I_n \cup \{1^n\}) \cdot \Sigma^* \cdot \# \cdot \Sigma^\omega$, the fact w' has no occurrences of $\#$ after position t_0 implies that the only chance of w' to be in L_n is to have a prefix of $w_1 \cdots w_{t_0}$ in $\Sigma^* \cdot (S_n \cup I_n \cup \{1^n\}) \cdot \Sigma^* \cdot \#$. 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 n-bit counter, nor they reach the value 1^n. It follows that $w \notin L_n$, and we are done. □

Lemmas 3.8 and 3.9 imply the desired exponential lower bound:

THEOREM 3.10. *There is a family of languages $L_2, L_3, \ldots$, over an alphabet of size 4, such that for every $n \geq 2$, the language L_n is NCW-recognizable, it can be recognized by an NBW with $O(n)$ states, and every NCW that recognizes it has at least $n2^n$ states.*

Combining the above lower bound with the upper bound in Theorem 3.6, we can conclude with the following.[5]

THEOREM 3.11. *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 3.12. While NCW = DCW, nondeterminism does add power to *weak automata* [Muller et al. 1986]. Let NWW and DWW denote nondeterministic and deterministic weak automata on infinite words. While NWW = NCW, it is known that DWW = DBW ∩ DCW [Landweber 1969; Maler and Staiger 1997; Boigelot et al. 2001]. Since the translation of an NCW to an equivalent NWW is always possible and only doubles the state space, Theorem 3.6 implies an $n2^n$ translation of an NBW to an NWW, when possible. We now show that the languages L_n we used in the lower-bound proof are actually DWW-recognizable, implying that the asymptotically tight bound for the state

[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_n requires $O(n)$ states and not strictly n states.

blow up in the translation, when possible, of an NBW to an equivalent NWW is $2^{\Theta(n)}$. Thus, Theorem 3.11 applies already to the special case of weak automata.

We prove that L_n is DBW-recognizable. Since DWW = DBW $\cap$ DCW, and L_n is DCW-recognizable (Lemma 3.9), it follows that L_n is DWW-recognizable. Clearly, $(\Sigma^* \cdot \#)^\omega$ is DBW-recognizable. In addition, L'_n is a co-safety language (every word in L'_n has a *good prefix*, namely a prefix all whose extensions are in L'_n), As such, L'_n is DBW-recognizable. Since $L_n = L'_n \cup (\Sigma^* \cdot \#)^\omega$ 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 $\{\{\langle q, E\rangle \mid q \in E\} : E \subseteq B\}$, which is isomorphic to 2^B. This transparency also applies to a union of some identical copies of the augmented subset construction.

PROPOSITION 4.1. *Consider a semi-automaton $\mathcal{A}$, let $\mathcal{C}$ be its augmented subset construction, and let $\mathcal{C}'$ be a union of some copies of $\mathcal{C}$. Then the subset construction of $\mathcal{C}'$ is isomorphic to the subset construction of $\mathcal{A}$.*

PROOF. Applying the subset construction on $\mathcal{C}$ with state space $A \times 2^A$, one ends up in a deterministic automaton with state space $\{\{\langle q, E\rangle \mid q \in E\} : E \subseteq A\}$, which is isomorphic to 2^A. Since $\mathcal{C}'$ is the union of some identical copies of $\mathcal{C}$, say $\mathcal{C}_1, \ldots, \mathcal{C}_m$, its subset construction is $\{\{\langle\langle q_1, E\rangle, \langle q_2, E\rangle, \ldots, \langle q_m, E\rangle\rangle \mid q_1, \ldots, q_m \in E\} : E \subseteq A\}$, which is again isomorphic to 2^A. (Note that each of $q_1, \ldots, q_m$ above is not a name of a specific state, but a state-variable.) □

It is well known that the subset construction can be used as an intermediate layer in translating an NCW with state space C to a DCW with state space 3^C [Miyano and Hayashi 1984]. Thus, Proposition 4.1 above suggests that, when applied to $\mathcal{C}$, the translation of [Miyano and Hayashi 1984] 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 4.2 below, this is indeed the case.

THEOREM 4.2. *For every NBW $\mathcal{B}$ with n states that is NCW-recognizable there is an equivalent DCW $\mathcal{D}$ with at most 3^n states.*

PROOF. The DCW $\mathcal{D}$ follows all the runs of the NCW $\mathcal{C}$ constructed in Theorem 3.6. 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 [Miyano and Hayashi 1984] 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 $f(E) = \{b \mid \langle b, E\rangle \in \alpha_\mathcal{C}\}$. Thus, when the subset component of $\mathcal{D}$ is in state E, it should continue

[6]Readers familiar with the construction of [Miyano and Hayashi 1984] 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 3.6.

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.

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

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 $3^{|B|}$. □

5. STRONGER ACCEPTANCE CONDITIONS

The Büchi acceptance condition can be viewed as a special case of the Rabin, Streett, and Muller acceptance conditions. While nondeterministic Rabin, Streett, and Muller automata are not more expressive than nondeterministic Büchi automata, they are more succinct: translating an NRW, NSW, and NMW with n states and index k to an NBW, results in an NBW with $O(nk)$, $O(n2^k)$, and $O(n^2k)$ states, respectively [Safra and Vardi 1989; Seidl and Niwiński 1999]. In this section we study the translation, when possible, of NRWs, NSWs, and NMWs to NCWs and DCWs. Since the Büchi acceptance condition is a special case of these stronger conditions, the $2^{\Omega(n)}$ lower bound from Theorem 3.10 applies, and the challenge is to come up with matching upper bounds.

A first attempt to translate NRWs, NSWs, and NMWs to NCWs is to go via intermediate NBWs and then apply the augmented subset construction. This, however, results in NCWs that are not optimal. A second attempt is then to apply the augmented subset construction directly on the input automaton, and check the possibility of defining on top of it a suitable co-Büchi acceptance condition. It is not hard to see that this second attempt does not work for all automata. Consider for example the Rabin acceptance condition. Note that the augmented subset construction does not alter a deterministic automaton. Also, DRWs are not DCW-type [Kupferman et al. 2004] (that is, there is a DRW $\mathcal{A}$ whose language is DCW-recognizable, but still no DCW equivalent to $\mathcal{A}$ can be defined on top of the structure of $\mathcal{A}$). It follows that there are NRWs whose language is NCW-recognizable, but still no NCW recognizing them can be defined on top of the automaton obtained by applying the augmented subset construction on them (see Theorem 5.2 for a concrete example).

With this in mind, this section is a collection of good news. First, we show that NSWs can be translated to NCWs on top of the augmented subset construction. Second, while this is not valid for NRWs and NMWs, we show that they can be translated to NCWs on top of a union of copies of the augmented subset construction. Moreover, when translating to an equivalent DCW, the different copies can be determinized separately.

5.1. From NSW to NCW

The translation of an NSW to an NCW, when exists, can be done on top of the augmented subset construction, generalizing the acceptance condition used for translating an NBW to an NCW.

In the translation of an NBW to an NCW, we start with an NBW $\mathcal{B}$ and define a state $\langle b, E\rangle$ of the augmented subset construction to be co-Büchi accepting if there is some path p in $\mathcal{B}$, taking $\langle b, E\rangle$ back to itself via a Büchi accepting state. The correctness

of the construction follows from the fact that an NCW-recognizable language is closed under pumping such cycles. Thus, if $\mathcal{B}$ accepts a word that includes a subword along which p is read, then $\mathcal{B}$ also accepts words obtained by pumping the subword along which p is read. In turns out that this intuition is valid also when we start with an NSW $\mathcal{S}$: a state $\langle s, E\rangle$ of the augmented subset construction is co-Büchi accepting if there is some path p in $\mathcal{S}$, taking $\langle s, E\rangle$ back to itself, such that p visits α_i or avoid β_i for every pair i in the Streett acceptance condition. This guarantees that pumping p infinitely often results in a run that satisfies the Streett condition, which in turn implies that an NCW-recongnizable language is closed under such pumping.

We formalize and prove this idea below.

THEOREM 5.1. *For every NSW $\mathcal{S}$ with n states that is NCW-recognizable, there is an equivalent NCW $\mathcal{C}$ with at most $n2^n$ states.*

PROOF. Let $\mathcal{S} = \langle \Sigma, S, \delta_{\mathcal{S}}, S_0, \langle \beta_1, \alpha_1\rangle, \ldots \langle \beta_k, \alpha_k\rangle\rangle$. We define the NCW $\mathcal{C} = \langle \Sigma, C, \delta_{\mathcal{C}}, C_0, \alpha_{\mathcal{C}}\rangle$ as the augmented subset construction of $\mathcal{S}$ with the following acceptance condition: a state is a member of $\alpha_{\mathcal{C}}$ if it is reachable from itself along a path whose projection on S visits α_i or avoids β_i for every $1 \leq i \leq k$.

Formally, $\langle s, E\rangle \in \alpha_{\mathcal{C}}$ if there is a finite word $z = z_1 z_2 \cdots z_m$ of length m and a sequence of $m+1$ states $\langle s_0, E_0\rangle \ldots \langle s_m, E_m\rangle$ such that $\langle s_0, E_0\rangle = \langle s_m, E_m\rangle = \langle s, E\rangle$, and for all $0 \leq l < m$ we have $\langle s_{l+1}, E_{l+1}\rangle \in \delta_C(\langle s_l, E_l\rangle, z_{l+1})$, and for every $1 \leq i \leq k$, either there is $0 \leq l < m$ such that $s_l \in \alpha_i$ or $s_l \notin \beta_i$ for all $0 \leq l < m$. We refer to z as the *witness* for $\langle s, E\rangle$. Note that z may be the empty word.

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

We first prove that $L(\mathcal{S}) \subseteq L(\mathcal{C})$. Note that this direction is always valid, even if $\mathcal{S}$ is not NCW-recognizable. Consider a word $w \in L(\mathcal{S})$. Let r be an accepting run of $\mathcal{S}$ on w. We prove that the run r' induced by r is accepting. Let $J \subseteq \{1, \ldots, k\}$ denote the set of indices of acceptance-pairs whose β-element is visited infinitely often by r. That is, $J = \{j \mid \beta_j \cap \mathit{inf}(r) \neq \emptyset\}$. Consider a state $\langle s, E\rangle \in \mathit{inf}(r')$. We prove that $\langle s, E\rangle \in \alpha_{\mathcal{C}}$. Since $\langle s, E\rangle$ appears infinitely often in r' and r is accepting, it follows that there are two (not necessarily adjacent) occurrences of $\langle s, E\rangle$, between which r visits α_j for all $j \in J$ and avoids β_i for all $i \notin J$. Hence, we have the required witness for $\langle s, E\rangle$, and we are done.

We now prove that $L(\mathcal{C}) \subseteq L(\mathcal{S})$. Consider a word $w \in L(\mathcal{C})$, and let r be an accepting run of $\mathcal{C}$ on w. Let $J \subseteq \{1, \ldots, k\}$ denote the set of indices of acceptance-pairs whose β-element is visited infinitely often by r. That is, $J = \{j \mid (\beta_j \times 2^S) \cap \mathit{inf}(r) \neq \emptyset\}$. If J is empty then the projection of r on its $\mathcal{S}$-component is accepting, and we are done. Otherwise, we proceed as follows. For every $j \in J$, let $\langle s_j, E_j\rangle$ be a state in $(\beta_j \times 2^S) \cap \mathit{inf}(r)$.

By the definition of J, all the states $\langle s_j, E_j\rangle$, with $j \in J$, are visited infinitely often in r, whereas states whose $\mathcal{S}$-component is in β_i, for $i \notin J$, are visited only finitely often in r. Accordingly, the states $\langle s_j, E_j\rangle$, with $j \in J$, are strongly connected via a path that does not visit β_i, for $i \notin J$. In addition, for every $\langle s_j, E_j\rangle$, with $j \in J$, there is a witness z_j for the membership of $\langle s_j, E_j\rangle$ in $\alpha_{\mathcal{C}}$, going from $\langle s_j, E_j\rangle$ back to itself via α_j and either avoiding β_i or visiting α_i, for every $1 \leq i \leq k$. Let $\langle s, E\rangle$ be one of these $\langle s_j, E_j\rangle$ states, and let x be a prefix of w such that $r(x) = \langle s, E\rangle$. Then, there is a finite word z along which there is a path from $\langle s, E\rangle$ back to itself, visiting all α_j for $j \in J$ and either avoiding β_i or visiting α_i for every $1 \leq i \leq k$. Therefore, $x \cdot z^\omega \in L(\mathcal{S})$.

Recall that the language of $\mathcal{S}$ is NCW-recognizable. Let $\mathcal{D} = \langle \Sigma, D, \delta_D, D_0, \alpha_{\mathcal{D}}\rangle$ be a DCW equivalent to $\mathcal{S}$. Since $L(\mathcal{S}) = L(\mathcal{D})$ and $x \cdot z^\omega \in L(\mathcal{S})$, it follows that the run ρ

of $\mathcal{D}$ on $x \cdot z^\omega$ is accepting. Since D is finite, there are two indices, l and m, such that $l < m$, $\rho(x \cdot z^l) = \rho(x \cdot z^m)$, and for all prefixes y of $x \cdot z^\omega$ such that $x \cdot z^l \preceq y$, we have $\rho(y) \in \alpha_{\mathcal{D}}$. Let q be the state of $\mathcal{D}$ such that $q = \rho(x \cdot z^l)$.

Consider the run η of $\mathcal{D}$ on w. Since r visits $\langle s, E\rangle$ infinitely often and D is finite, there must be a state $d \in D$ and infinitely many prefixes $p_1, p_2, \ldots$ of w such that for all $i \geq 1$, we have $r(p_i) = \langle s, E\rangle$ and $\eta(p_i) = d$. We claim that the states q and d of $\mathcal{D}$ satisfy the conditions of Lemma 3.2 with x_1 being p_1 and x_2 being $x \cdot z^l$. Indeed, $\delta_{\mathcal{D}}(p_1) = d$, $\delta_{\mathcal{D}}(x \cdot z^l) = q$, and $\delta_{\mathcal{S}}(p_1) = \delta_{\mathcal{S}}(x \cdot z^l) = E$. For the latter equivalence, recall that $\delta_{\mathcal{S}}(x) = E$ and $\delta_{\mathcal{S}}(E, z) = E$. Hence, by Lemma 3.2, we have that $L(\mathcal{D}^q) = L(\mathcal{D}^d)$.

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 satisfies the conditions of Corollary 3.5 with u being z^{m-l} and v being t_i. The second condition is satisfied by Lemma 3.3. For the first condition, consider a word $w' \in (v^* \cdot u^+)^\omega$. We prove that $w' \in L(\mathcal{D}^d)$. Recall that there is a run of $\mathcal{S}^s$ on v that goes back to s while avoiding β_i for all $i \notin J$ and there is a run of $\mathcal{S}^s$ on u that goes back to s while visiting α_j for all $j \in J$ and either visiting α_i or avoiding β_i for all $i \notin J$. (Informally, u "fixes" all the problems of v, by visiting α_j for every β_j that v might visit.) Recall also that for the word p_1, we have that $r(p_1) = \langle s, E\rangle$ and $\eta(p_1) = d$. Hence, $p_1 \cdot w' \in L(\mathcal{S})$. Since $L(\mathcal{S}) = L(\mathcal{D})$, we have that $p_1 \cdot w' \in L(\mathcal{S})$. Therefore, $w' \in L(\mathcal{D}^d)$.

Thus, by Corollary 3.5, for all $i \geq 1$ we have that $t_i^\omega \in L(\mathcal{D}^d)$. Since $\delta_{\mathcal{D}}(d, t_i) = d$, it follows that all the states that $\mathcal{D}$ visits when it reads t_i from d are in $\alpha_{\mathcal{D}}$. Note that $w = p_1 \cdot t_1 \cdot t_2 \cdots$. Hence, since $\delta_{\mathcal{D}}(p_1) = d$, the run of $\mathcal{D}$ on w is accepting, thus $w \in L(\mathcal{D})$. Since $L(\mathcal{D}) = L(\mathcal{S})$, it follows that $w \in L(\mathcal{S})$, and we are done. □

Two common special cases of the Streett acceptance condition are the parity and the *generalized Büchi* acceptance conditions. In a generalized Büchi automaton with states Q, the acceptance condition is $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$ with $\alpha_i \subseteq Q$, and a run r is accepting if $inf(r) \cap \alpha_i \neq \emptyset$ for all $1 \leq i \leq k$. Theorem 5.1 implies that an NCW-recognizable nondeterministic parity or generalized Büchi automaton with n states can be translated to an NCW with $n2^n$ states, which can be defined on top of the augmented subset construction.

5.2. From NRW and NMW to NCW

In this section we study the translation of NRWs and NMWs to NCWs, when exists. Unfortunately, for automata in these classes we cannot define an equivalent NCW on top of the augmented subset construction. Intuitively, the key idea of Subsection 5.1, which is based on the ability to pump paths that satisfy the acceptance condition, is not valid in the Rabin and the Muller acceptance conditions, as in these conditions, visiting some "bad" states infinitely often need not be compensated by visiting some "good" ones infinitely often. We formalize this in the example below, which consists of the fact that DRWs are not DCW-type [Kupferman et al. 2004].

THEOREM 5.2. *There is an NRW and an NMW that are NCW-recognizable but an equivalent NCW for them cannot be defined on top of the augmented subset construction.*

PROOF. Consider the NRW $\mathcal{A}$ appearing in Figure 4. The language of $\mathcal{A}$ consists of all words over the alphabet $\{0, 1\}$ that either have finitely many 0's or have finitely many 1's. This language is clearly NCW-recognizable, as it is the union of two NCW-recognizable languages. Since $\mathcal{A}$ is deterministic and the augmented subset construction does not alter a deterministic automaton, it suffices to show that there is no co-Büchi acceptance condition α' that we can define on the structure of $\mathcal{A}$ and get an equivalent language. Indeed, α' may either be $\emptyset$, $\{q_0\}$, $\{q_1\}$, or $\{q_0, q_1\}$, none of which

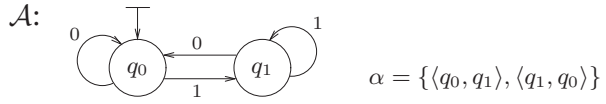

Fig. 4. The NRW $\mathcal{A}$, having no equivalent NCW on top of its augmented subset construction.

provides the language of $\mathcal{A}$. Since every NRW has an equivalent NMW over the same structure, the above result also applies to the NMW case. □

Consider an NRW or an NMW $\mathcal{A}$ with index k. Our approach for translating $\mathcal{A}$ to an NCW is to decompose it to k NSWs over the same structure, and apply the augmented subset construction on each of the components. Note that the components may not be NCW-recognizable even when $\mathcal{A}$ is, thus, we should carefully analyze the proof of Theorem 5.1 and prove that the approach is valid.

We now formalize and prove the above approach. We start with the decomposition of an NRW or an NMW with index k into k NSWs over the same structure.

LEMMA 5.3. *Every NRW or NMW $\mathcal{A}$ with index k is equivalent to the union of k NSWs over the same structure as $\mathcal{A}$.*

PROOF. An NRW $\mathcal{A}$ with states A and index k is the union of k NRWs with index 1 over the same structure as $\mathcal{A}$. Since a single-indexed Rabin acceptance condition $\{\langle \alpha_1, \beta_1\rangle\}$ is equivalent to the Streett acceptance condition $\{\langle \alpha_1, \emptyset\rangle, \langle A, \beta_1\rangle\}$, we are done.

An NMW $\mathcal{A}$ with states A and index k is the union of k NMWs with index 1 over the same structure as $\mathcal{A}$. Since a single-indexed Muller acceptance condition $\{\alpha_1\}$ is equivalent to the Streett acceptance condition $\{\langle A \setminus \alpha_1, \emptyset\rangle\} \cup \bigcup_{s\in\alpha_1}\{\langle A, \{s\}\rangle\}$, we are done. □

Next we show that a union of k NSWs can be translated to a single NSW over their union.

LEMMA 5.4. *Consider k NSWs, $\mathcal{S}_1, \ldots, \mathcal{S}_k$, over the same structure. There is an NSW $\mathcal{S}$ over the disjoint union of their structures, such that $L(\mathcal{S}) = \bigcup_{i=1}^{k} L(\mathcal{S}_i)$.*

PROOF. We obtain the Streett acceptance condition of $\mathcal{S}$ by taking the union of the Streett acceptance conditions of the NSWs $\mathcal{S}_1, \ldots, \mathcal{S}_k$. Note that while the underlying NSWs are interpreted disjunctively (that is, in order for a word to be accepted by the union, there should be an accepting run on it in some $\mathcal{S}_i$), the pairs in the Streett condition are interpreted conjunctively (that is, in order for a run to be accepting, it has to satisfy the constraints by all the pairs in the Streett condition). We prove that still $L(\mathcal{S}) = \bigcup_{i=1}^{k} L(\mathcal{S}_i)$. First, if a run r of $\mathcal{S}$ is an accepting run of an underlying NSW $\mathcal{S}_i$, then the acceptance conditions of the other underlying NSWs are vacuously satisfied. Hence, if a word is accepted by $\mathcal{S}_i$ for some $1 \leq i \leq k$, then $\mathcal{S}$ accepts it too. For the other direction, if a word w is accepted in $\mathcal{S}$, then its accepting run in $\mathcal{S}$ is also an accepting run of one of the underlying NSWs, thus w is in $\bigcup_{i=1}^{k} L(\mathcal{S}_i)$. □

Finally, we combine the translation to Streett automata with the augmented subset construction and get the required upper bound for NRW and NMW.

THEOREM 5.5. *For every NCW-recognizable NRW or NMW with n states and index k, there is an equivalent NCW $\mathcal{C}$ with at most $kn2^n$ states.*

PROOF. Consider an NRW or an NMW $\mathcal{A}$ with n states and index k. By Lemmas 5.3 and 5.4, there is an NSW $\mathcal{S}$ whose structure consists of k copies of the structure of $\mathcal{A}$

such that $L(\mathcal{S}) = L(\mathcal{A})$. Let $\mathcal{C}$ be the NCW equivalent to $\mathcal{S}$, defined over the augmented subset construction of $\mathcal{S}$, as described in Theorem 5.1. Note that $\mathcal{S}$ has nk states, thus a naive application of the augmented subset construction on it results in an NCW with $kn2^{kn}$ states. The key observation, which implies that we get an NCW with only $kn2^n$ states, is that applying the augmented subset construction on $\mathcal{S}$, the deterministic component of all the underlying NCWs is the same, and it coincides with the subset construction applied to $\mathcal{A}$. To see this, assume that $\mathcal{A} = \langle \Sigma, A, A_0, \delta, \alpha \rangle$. Then, $\mathcal{S} = \langle \Sigma, A \times \{1, \ldots, k\}, A_0 \times \{1, \ldots, k\}, \delta', \alpha' \rangle$, where for all $a \in A, 1 \leq j \leq k$, and $\sigma \in \Sigma$, we have that $\delta'(\langle a, j \rangle, \sigma) = \delta(a, \sigma) \times \{j\}$. Applying the augmented subset construction, we get the product of $\mathcal{S}$ and its subset construction, where the latter has a state for every reachable subset of S. That is, a subset $G' \subseteq S$ is a state of the subset construction if there is a finite word u for which $\delta'(u) = G'$. Since for all $a \in A, 1 \leq j \leq k$, and $\sigma \in \Sigma$, we have that $\delta'(\langle a, j \rangle, \sigma) = \delta(a, \sigma) \times \{j\}$, it follows that G' is of the form $G \times \{j\}$ for all $1 \leq j \leq k$ and some $G \subseteq A$. Hence, there are up to $2^{|A|} = 2^n$ states in the subset construction of $\mathcal{S}$. Thus, when we apply the augmented subset construction on $\mathcal{S}$, we end up with an NCW with only $kn2^n$ states, and we are done. □

5.3. To DCW

In Section 4 we showed that if $\mathcal{C}$ is an NCW obtained by the augmented subset construction, then applying the breakpoint construction on $\mathcal{C}$ does not involve an exponential blow up. This implies, by Subsection 5.1, a 3^n blow-up for the translation of an NSW to a DCW.

THEOREM 5.6. *For every DCW-recognizable NSW $\mathcal{A}$ with n states, there is an equivalent DCW $\mathcal{D}$ with at most 3^n states.*

PROOF. By Theorem 5.1, the NSW $\mathcal{A}$ has an equivalent NCW $\mathcal{C}$ with $n2^n$ states, obtained by applying the augmented subset construction on $\mathcal{A}$. As in Theorem 4.2, applying the breakpoint construction on $\mathcal{C}$ we end up with a DCW with at most 3^n states. □

By [Boker et al. 2010], one cannot avoid the 3^n state blow-up for translating an NCW to a DCW. Since this lower bound clearly holds also for the stronger NSW condition, we get a tight bound.

THEOREM 5.7. *The tight bound for the state blow-up in the translation, when possible, of NSW to an equivalent DCW is $\Theta(3^n)$.*

Translating an NRW or an NMW to an NCW is more complicated than translating an NSW. As shown in Subsection 5.2, the translation of an NRW or NMW $\mathcal{A}$ with index k results in an NCW that has k copies of the augmented subset construction of $\mathcal{A}$. When applying the breakpoint construction on this NCW, its k different parts might cause a doubly-exponential blowup. Fortunately, we can avoid it by determinizing each of the k parts separately and connecting them in a round-robin fashion. This implies a $k3^n$ blow-up for the translation of NRWs and NMWs, to DCW.

THEOREM 5.8. *For every DCW-recognizable NMW or NRW $\mathcal{A}$ with n states and index k there is an equivalent DCW $\mathcal{D}$ with at most $k3^n$ states.*

PROOF. by Theorem 5.5, $\mathcal{A}$ has an equivalent NCW $\mathcal{C}$ with $kn2^n$ states. The NCW $\mathcal{C}$ is obtained by applying the augmented subset construction on k copies of $\mathcal{A}$, and thus has k unconnected components, $\mathcal{C}_1, \ldots, \mathcal{C}_k$ that are identical up to their acceptance conditions $\alpha_{\mathcal{C}_1}, \ldots, \alpha_{\mathcal{C}_k}$.

Since the k components of $\mathcal{C}$ all have the same $A \times 2^A$ structure, applying the standard subset construction on $\mathcal{C}$, one ends up with a deterministic automaton that is

isomorphic to 2^A. Applying the breakpoint construction on $\mathcal{C}$, we could thus hope to obtain a deterministic automaton with only $3^{|A|}$ states. This construction, however, has to consider the different acceptance conditions α_i, maintaining in each state not only a pair $\langle S, O\rangle$, but a tuple $\langle S, O_1, \ldots, O_k\rangle$, where each $O_i \subseteq S$ corresponds to the breakpoint construction with respect to α_i. Such a construction, however, involves a k^n blow-up.

We circumvent this blow-up by determinizing each of the $\mathcal{C}_i$'s separately and connecting the resulting $\mathcal{D}_i$'s in a round-robin fashion, moving from $\mathcal{D}_i$ to $\mathcal{D}_{i \pmod k+1}$ when the set O, which maintains the set of states in paths in which $\mathcal{D}_i$ avoids α_i, becomes empty. Now, there is $1 \leq i \leq k$ such that $\mathcal{C}_i$ has a run that eventually gets stuck in α_i iff there is $1 \leq i \leq k$ such that in the round-robin construction, the run gets stuck in a copy that corresponds to $\mathcal{D}_i$ in states with $O \neq \emptyset$.

Formally, for every $1 \leq i \leq k$, we define a function $f_i : 2^A \to 2^A$ by $f_i(E) = \{a \mid \langle a, E\rangle \in \alpha_{\mathcal{C}_i}\}$. We define the DCW $\mathcal{D} = \langle \Sigma, D, \delta_{\mathcal{D}}, D_0, \alpha_{\mathcal{D}}\rangle$ as follows.

— $D = \{\langle S, O, i\rangle \mid S \subseteq A$, $O \subseteq S \cap f_i(S)$, and $i \in \{1, \ldots k\}\}$.
— For all $\langle S, O, i\rangle \in D$ and $\sigma \in \Sigma$, the transition function is defined as follows.
  — If $O \neq \emptyset$, then $\delta_D(\langle S, O, i\rangle, \sigma) = \{\langle S', O', i'\rangle\}$, where $S' = \delta_{\mathcal{A}}(S, \sigma)$, $O' = \delta_{\mathcal{A}}(O, \sigma) \cap f_i(\delta_{\mathcal{A}}(S, \sigma))$ and $i' = i \pmod k + 1$ if $O' = \emptyset$ and $i' = i$ otherwise.
  — If $O = \emptyset$, then $\delta_D(\langle S, O, i\rangle, \sigma) = \{\langle S', O', i'\rangle\}$, where $S' = \delta_{\mathcal{A}}(S, \sigma)$, $O' = \delta_{\mathcal{A}}(S, \sigma) \cap f_i(\delta_{\mathcal{A}}(S, \sigma))$ and $i' = i \pmod k + 1$ if $O' = \emptyset$ and $i' = i$ otherwise.
— $D_0 = \{\langle A_0 \text{ of } \mathcal{C}_1, \emptyset\rangle\}$.
— $\alpha_{\mathcal{D}} = \{\langle S, O, i\rangle \mid O \neq \emptyset\}$.

A run of $\mathcal{D}$ is accepting if it gets stuck in one of the sets of accepting states. Since the different parts of $\mathcal{C}$ are unconnected, we have that a run of $\mathcal{C}$ is accepting iff it gets stuck in the accepting states of one of the $\mathcal{C}_i$'s. Hence, a word is accepted by $\mathcal{C}$ iff it is accepted by $\mathcal{D}$, and we are done. □

Since the lower bound for NBW clearly holds also for the stronger condition, we can conclude with the following.

THEOREM 5.9. *The asymptotically tight bound for the state blow up in the translation, when possible, of NRW and NMW to an equivalent DCW is* $2^{\Theta(n)}$.

6. APPLICATIONS

In this section we describe immediate applications of our NBW to NCW (Theorem 3.6) and NBW to DCW (Theorem 4.2) 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 construction, have a simple state space, are amenable to optimizations, and can be implemented symbolically [Morgenstern and Schneider 2008]. Also, the constructions described for the richer acceptance conditions (Theorems 5.1, 5.5, 5.6 and 5.8) extend these applications to NSW, NRW, and NMW, and to translations of LTL to automata that use the richer acceptance conditions.

Theorems 3.6, 5.1, 5.5, 5.6 and 5.8 guarantee that if the given automaton is NCW-recognizable, 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 6.1. *For an automaton $\mathcal{A}$, let $\mathcal{C}$ be the NCW obtained by the translations of Theorems 3.6, 5.1 and 5.5, and let $\mathcal{D}$ be the DCW obtained from $\mathcal{A}$ by the constructions of Theorems 4.2, 5.6 and 5.8. Then,* $L(\mathcal{A}) \subseteq L(\mathcal{C}) = L(\mathcal{D})$.

PROOF. It is easy to see that the proof of the $L(\mathcal{A}) \subseteq L(\mathcal{C})$ direction in Theorems 3.6, 5.1 and 5.5, as well as the equivalence of $\mathcal{C}$ and $\mathcal{D}$ in Theorems 5.6 and 5.8, do not rely on the assumption that $\mathcal{A}$ is NCW-recognizable. □

We now turn to describe the applications.

6.1. 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, [Kupferman and Vardi 2005b] studies the problem of deciding whether an NBW or an LTL formula is in DBW. The solution in [Kupferman and Vardi 2005b] 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 3.6 and Lemma 6.1.

LEMMA 6.2. *Consider an automaton $\mathcal{A}$. Let $\mathcal{C}$ be its translation to NCW by the construction of Theorems 3.6, 5.1 and 5.5. Then, $\mathcal{A}$ is NCW-recognizable iff $L(\mathcal{C}) \subseteq L(\mathcal{A})$.*

We start with a procedure for deciding whether a given nondeterministic automaton $\mathcal{B}$ is NCW-recognizable. By Lemma 6.2, the latter can be reduced to checking whether $L(\mathcal{C}) \subseteq L(\mathcal{B})$, for the NCW $\mathcal{C}$ constructed from $\mathcal{B}$ in Theorems 3.6, 5.1, and 5.5. This involves the generation of $\mathcal{C}$ and of the complement $\tilde{\mathcal{B}}$ of $\mathcal{B}$, and checking the emptiness of the product $C \times \tilde{\mathcal{B}}$. For the Büchi, Rabin, and Streett conditions, complementation can be done without determinization [Kupferman and Vardi 2001; 2005a]. For $\mathcal{B}$ with n states and index k, the complexity is $2^{O(n \log n)}$ for NBW, is $2^{O(nk \log n)}$ for NRW, and is $2^{O(nk \log nk)}$ for NSW. 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 [Kupferman and Vardi 2001].

We proceed with the problem of deciding whether a given LTL formula is in NCW or in DBW. Given an LTL formula ψ, let $\mathcal{B}_\psi$ be a nondeterministic automaton for ψ, and let $\mathcal{C}_\psi$ be its NCW translation as in Theorems 3.6, 5.1, and 5.5. By Lemma 6.2, 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. Note that we do not restrict $\mathcal{B}_\psi$ to be an NBW. Indeed, the translation of LTL to NBW goes via nondeterministic generalized Büchi automata. Thus, keeping in mind that the translation of NSW and NRW to NCW results in the same state space that the translation of NBW does, a direct application of the procedures on the richer acceptance conditions yields better complexity. Moreover, in general, the translation of LTL to NSW can result in automata that are exponentially more succinct than equivalent NBWs [Safra and Vardi 1989].

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]

6.2. Translating LTL to AFMC

Consider an LTL formula ψ. By [Kupferman and Vardi 2005b], there is an AFMC formula equivalent to $\forall\psi$ iff ψ is in DBW. As described in Section 6.1, we can first check whether ψ can be translated to AFMC. Let $\mathcal{C}_{\neg\psi}$ be the NCW-translation, as in Theorem 3.6, of the NBW $\mathcal{B}_{\neg\psi}$ for $\neg\psi$ [Vardi and Wolper 1994]. By Lemma 6.1, we have that $L(\mathcal{C}_{\neg\psi})$ accepts exactly all computations that violate ψ. By [Kupferman and Vardi 2005b], 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 that the blow-up in the translation of LTL to AFMC is still open. [8]

6.3. 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 4.2 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 6.1 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 6.3. *Consider an LTL formula ψ that is DBW-recognizable. Let $\mathcal{B}_{\neg\psi}$ be an NBW accepting $L(\neg\psi)$, and let $\mathcal{D}_\psi$ be the DBW obtained by dualizing the DCW obtained by the construction of Theorem 4.2 on $\mathcal{B}_{\neg\psi}$. Then, $L(\mathcal{D}_\psi) = L(\psi)$.*

[7] A different Safraless procedure for deciding whether a specification can be translated to a DBW is described in [Kupferman and Vardi 2005c]. 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 NCW-recognizable requires complementation. In addition, the procedure in [Kupferman and Vardi 2005c] involves tree automata and is more complicated.

[8] Wilke [Wilke 1999] 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.

Note that, also here, one need not translate the LTL formula to an NBW, and can instead translate it to a nondeterministic generalized Büchi or even Streett automaton, which are more succinct.

6.4. 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 NCW-translation of the NBW (or NSW, NRW, or NMW) $\mathcal{B}_{\neg\psi}$ for $\neg\psi$. By Lemma 6.1, 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, $\mathcal{C}_{\neg\psi}$ over-approximates $\mathcal{B}_{\neg\psi}$. 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 an LTL formula to a deterministic automaton, Lemma 6.1 implies that when applied to an arbitrary LTL formula, the automaton $\mathcal{D}_\psi$ constructed in Lemma 6.3 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 [Kupferman and Vardi 2005c], 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.

7. DISCUSSION

The simplicity of the co-Büchi condition and its duality to the Büchi condition make it an interesting theoretical object. Its many recent applications in practice motivate further study of it. Translating a Büchi automaton, as well as automata of richer acceptance conditions, to co-Büchi automata is useful in formal verification and synthesis, yet the state blow-up that such translations involve was a long-standing open problem. We solved the problem, and provided asymptotically tight constructions for translating all common classes of automata to nondeterministic and deterministic co-Büchi automata.

The state blow-up involved in the various translations is summarized in Table I.

All the constructions are extensions of the augmented subset construction and breakpoint construction, which are in turn extensions of the basic subset construction. In particular, the set of accepting states is induced by simple reachability queries in the graph of the automaton. Hence, the constructed automata have a simple state space and are amenable to optimizations and to symbolic implementations.

In spite of the exponential lower bound, we do not view the results in the paper as "bad news". We believe that the simple and symbolic translation of LTL to DBW (when exists), which follows from the translation of NBW and NSW to NCW, is a very good news. As discussed in Section 6.3, numerous applications of automata theory in prac-

Table I.

From ╲ To	NCW	DCW
NBW, NPW, NSW	$O(n2^n)$	$O(3^n)$
NRW, NMW	$O(kn2^n)$	$O(k3^n)$

The state blow-up involved in the translation, when possible, of a word automaton with n states and index k to an equivalent NCW and DCW.

tice require determinization. The intricacy of optimal determinization constructions has led to a situation in which many of these applications are not implemented in practice. From the theoretical side, we see work on new algorithms for the applications, ones that avoid the determinization [Kupferman and Vardi 2005a; Kupferman 2006; Henzinger and Piterman 2006; Ehlers 2010]. From the practical side, we see work on restricting the scope of the applications to specifications for which determinization is easy [Jobstmann and Bloem 2006; Piterman et al. 2006]. In particular, many applications are restricted to safety properties, for which determinization is based on the subset construction. Theorems 5.6 and 5.8 and Lemma 6.3 imply that one need not restrict attention to safety properties; the appealing simple and symbolic algorithms that handle them can be applied also to properties that are DBW-recognizable, and in fact, thanks to the one-sided error, to all of LTL.

REFERENCES

ACCELLERA. 2006. Accellera organization inc. http://www.accellera.org.

AMINOF, B., KUPFERMAN, O., AND LEV, O. 2008. 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 Series, vol. 5330. Springer, 183–197.

BOIGELOT, B., JODOGNE, S., AND WOLPER, P. 2001. On the use of weak automata for deciding linear arithmetic with integer and real variables. In *Proc. Int. Joint Conf. on Automated Reasoning*. Lecture Notes in Computer Science Series, vol. 2083. Springer, 611–625.

BOKER, U. AND KUPFERMAN, O. 2009. Co-ing Büchi made tight and helpful. In *Proc. 24th IEEE Symp. on Logic in Computer Science*. 245–254.

BOKER, U. AND KUPFERMAN, O. 2011. Co-Büching them all. In *Proc. 14th Int. Conf. on Foundations of Software Science and Computation Structures*. Lecture Notes in Computer Science Series, vol. 6604. Springer, 184–198.

BOKER, U., KUPFERMAN, O., AND ROSENBERG, A. 2010. Alternation removal in Büchi automata. In *Proc. 37th Int. Colloq. on Automata, Languages, and Programming*. Vol. 6199. 76–87.

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

EHLERS, R. 2010. Symbolic bounded synthesis. In *Proc. 22nd Int. Conf. on Computer Aided Verification*. Lecture Notes in Computer Science Series, vol. 6174. Springer, 365–379.

EMERSON, E. AND JUTLA, C. 1988. The complexity of tree automata and logics of programs. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*. 328–337.

HENZINGER, T. AND PITERMAN, N. 2006. Solving games without determinization. In *Proc. 15th Annual Conf. of the European Association for Computer Science Logic*. Lecture Notes in Computer Science Series, vol. 4207. Springer, 394–410.

JOBSTMANN, B. AND BLOEM, R. 2006. Game-based and simulation-based improvements for LTL synthesis. In *3nd Workshop on Games in Design and Verification*.

KÄHLER, D. AND WILKE, T. 2008. Complementation, disambiguation, and determinization of Büchi automata unified. In *Proc. 35th Int. Colloq. on Automata, Languages, and Programming*. Lecture Notes in Computer Science Series, vol. 525. Springer, 724–735.

KUPFERMAN, O. 2006. Avoiding determinization. In *Proc. 21st IEEE Symp. on Logic in Computer Science*. 243–254.

KUPFERMAN, O. 2007. Tightening the exchange rate between automata. In *Proc. 16th Annual Conf. of the European Association for Computer Science Logic*. Lecture Notes in Computer Science Series, vol. 4646. Springer, 7–22.

KUPFERMAN, O., MORGENSTERN, G., AND MURANO, A. 2004. Typeness for ω-regular automata. In *2nd Int. Symp. on Automated Technology for Verification and Analysis*. Lecture Notes in Computer Science Series, vol. 3299. Springer, 324–338.

KUPFERMAN, O. AND VARDI, M. 2001. Weak alternating automata are not that weak. *ACM Transactions on Computational Logic 2,* 2, 408–429.

KUPFERMAN, O. AND VARDI, M. 2005a. Complementation constructions for nondeterministic automata on infinite words. In *Proc. 11th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*. Lecture Notes in Computer Science Series, vol. 3440. Springer, 206–221.

KUPFERMAN, O. AND VARDI, M. 2005b. From linear time to branching time. *ACM Transactions on Computational Logic 6,* 2, 273–294.

KUPFERMAN, O. AND VARDI, M. 2005c. Safraless decision procedures. In *Proc. 46th IEEE Symp. on Foundations of Computer Science*. 531–540.

KURSHAN, R. 1994. *Computer Aided Verification of Coordinating Processes*. Princeton Univ. Press.

LANDWEBER, L. 1969. Decision problems for ω–automata. *Mathematical Systems Theory 3*, 376–384.

LÖDING, C. 1999. Optimal bounds for the transformation of omega-automata. In *Proc. 19th Conf. on Foundations of Software Technology and Theoretical Computer Science*. Lecture Notes in Computer Science Series, vol. 1738. 97–109.

MALER, O. AND STAIGER, L. 1997. On syntactic congruences for ω-languages. *Theoretical Computer Science 183,* 1, 93–112.

MCNAUGHTON, R. 1966. Testing and generating infinite sequences by a finite automaton. *Information and Control 9*, 521–530.

MICHEL, M. 1988. Complementation is more difficult with automata on infinite words. CNET, Paris.

MIYANO, S. AND HAYASHI, T. 1984. Alternating finite automata on ω-words. *Theoretical Computer Science 32*, 321–330.

MORGENSTERN, A. AND SCHNEIDER, K. 2008. Generating deterministic ω-automata for most LTL formulas by the breakpoint construction. In *Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen*. Freiburg, Germany.

MULLER, D., SAOUDI, A., AND SCHUPP, P. 1986. Alternating automata, the weak monadic theory of the tree and its complexity. In *Proc. 13th Int. Colloq. on Automata, Languages, and Programming*. Lecture Notes in Computer Science Series, vol. 226. Springer, 275 – 283.

MULLER, D. AND SCHUPP, P. 1995. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of theorems of Rabin, McNaughton and Safra. *Theoretical Computer Science 141*, 69–107.

PITERMAN, N. 2006. From nondeterministic Büchi and Streett automata to deterministic parity automata. In *Proc. 21st IEEE Symp. on Logic in Computer Science*. IEEE press, 255–264.

PITERMAN, N., PNUELI, A., AND SAAR, Y. 2006. Synthesis of reactive(1) designs. In *Proc. 7th Int. Conf. on Verification, Model Checking, and Abstract Interpretation*. Lecture Notes in Computer Science Series, vol. 3855. Springer, 364–380.

PNUELI, A. AND ROSNER, R. 1989. On the synthesis of a reactive module. In *Proc. 16th ACM Symp. on Principles of Programming Languages*. 179–190.

RABIN, M. 1969. Decidability of second order theories and automata on infinite trees. *Transaction of the AMS 141*, 1–35.

RAVI, K., BLOEM, R., AND SOMENZI, F. 2000. 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 Series, vol. 1954. Springer, 143–160.

SAFRA, S. 1988. On the complexity of ω-automata. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*. 319–327.

SAFRA, S. AND VARDI, M. 1989. On ω-automata and temporal logic. In *Proc. 21st ACM Symp. on Theory of Computing*. 127–137.

SCHEWE, S. AND FINKBEINER, B. 2007. Bounded synthesis. In *5th Int. Symp. on Automated Technology for Verification and Analysis*. Lecture Notes in Computer Science Series, vol. 4762. Springer, 474–488.

SEIDL, H. AND NIWIŃSKI, D. 1999. On distributive fixed-point expressions. *Theoretical Informatics and Applications 33,* 4–5, 427–446.

STREET, R. AND EMERSON, E. 1984. An elementary decision procedure for the μ-calculus. In *Proc. 11th Int. Colloq. on Automata, Languages, and Programming*. Vol. 172. Springer, 465–472.

THOMAS, W. 1990. Automata on infinite objects. *Handbook of Theoretical Computer Science*, 133–191.

VARDI, M. AND WOLPER, P. 1986. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and Systems Science 32,* 2, 182–221.

VARDI, M. AND WOLPER, P. 1994. Reasoning about infinite computations. *Information and Computation 115,* 1, 1–37.

WILKE, T. 1999. CTL^+ is exponentially more succinct than CTL. In *Proc. 19th Conf. on Foundations of Software Technology and Theoretical Computer Science*. Lecture Notes in Computer Science Series, vol. 1738. Springer, 110–121.

WOLPER, P., VARDI, M., AND SISTLA, A. 1983. Reasoning about infinite computation paths. In *Proc. 24th IEEE Symp. on Foundations of Computer Science*. 185–194.

YAN, Q. 2006. 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 Series, vol. 4052. Springer, 589–600.

Received Month Year; revised Month Year; accepted Month Year