On the Relative Succinctness of Nondeterministic Büchi and co-Büchi Word Automata

Benjamin Aminof, Orna Kupferman, and Omer Lev

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

Abstract. The practical importance of automata on infinite objects has motivated a re-examination of the complexity of automata-theoretic constructions. One such construction is the translation, when possible, of nondeterministic Büchi word automata (NBW) to nondeterministic co-Büchi word automata (NCW). Among other applications, it is used in the translation (when possible) of LTL to the alternation-free μ-calculus. The best known upper bound for the translation of NBW to NCW is exponential (given an NBW with n states, the best translation yields an equivalent NCW with $2^{O(n \log n)}$ states). On the other hand, the best known lower bound is trivial (no NBW with n states whose equivalent NCW requires even $n+1$ states is known). In fact, only recently was it shown that there is an NBW whose equivalent NCW requires a different structure.
In this paper we improve the lower bound by showing that for every integer $k \geq 1$ there is a language L_k over a two-letter alphabet, such that L_k can be recognized by an NBW with $2k+1$ states, whereas the minimal NCW that recognizes L_k has $3k$ states. Even though this gap is not asymptotically very significant, it nonetheless demonstrates for the first time that NBWs are more succinct than NCWs. In addition, our proof points to a conceptual advantage of the Büchi condition: an NBW can abstract precise counting by counting to infinity with two states. To complete the picture, we consider also the reverse NCW to NBW translation, and show that the known upper bound, which duplicates the state space, is tight.

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 [3, 13, 16]. Today, automata on infinite objects are used for *specification* and *verification* 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 [11, 21]. Recent industrial-strength property-specification languages such as Sugar [2], ForSpec [1], and the recent standard PSL 1.01 [5] include regular expressions and/or automata, making specification and verification tools that are based on automata even more essential and popular.

There are many ways to classify an automaton on infinite words. One is the type of its acceptance condition. For example, 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 [3]. Dually, in *co-Büchi* automata, a run is accepting iff it visits states from the accepting set only finitely often. Another way to classify an automaton is by the type of its branching mode. In a *deterministic* automaton, the transition function maps the current state and input letter to a single successor state. When the branching mode is *nondeterministic*, the transition function maps the current state and letter to a set of possible successor states. Thus, 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, and the word is accepted by the automaton if at least one of the runs is accepting.

Early automata-based algorithms aimed at showing decidability. The complexity of the algorithm was not of much interest. Things have changed in the early 80's, when decidability of highly expressive logics became of practical importance in areas such as artificial intelligence and formal reasoning about systems. The change was reflected in the development of two research directions: (1) direct and efficient translations of logics to automata [23, 19, 20], and (2) improved algorithms and constructions for automata on infinite objects [18, 4, 15]. 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 we may be using algorithms that can be significantly improved, but also that something is missing in our understanding of automata on infinite objects.

One such problem, which this article studies, is the problem of translating, when possible, a nondeterministic Büchi word automaton (NBW) to an equivalent nondeterministic co-Büchi word automaton (NCW). 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. The best translation of an NBW to an NCW (when possible) that is currently known actually results in a deterministic co-Büchi automaton (DCW), and it goes via an intermediate deterministic Streett automaton. The determinization step involves an exponential blowup in the number of states [18]. Hence, starting with an NBW with n states, we end up with a DCW with $2^{O(n \log n)}$ states.

The exponential upper bound is particularly annoying, since the best known lower bound is trivial. That is, no NBW with n states whose equivalent NCW requires even $n + 1$ states is known. In fact, only recently was it shown that there is an NBW whose equivalent NCW requires a different structure [8]. Beyond the theoretical challenge in closing the exponential gap, and the fact it is related to other exponential gaps in our knowledge [7], the translation of NBW to NCW has immediate applications in symbolic LTL model checking. We elaborate on this point below.

It is shown in [9] that given an LTL formula ψ, there is an alternation-free μ-calculus (AFMC) formula equivalent to $\forall\psi$ iff ψ can be recognized by a deterministic Büchi automaton (DBW). Evaluating specifications in the alternation-free fragment of μ-calculus can be done with linearly many symbolic steps. In contrast, direct LTL model checking reduces to a search for bad-cycles, whose symbolic implementation involves nested fixed-points, and is typically quadratic [17]. The best known translations of LTL to AFMC first translates the LTL formula ψ to a DBW, which is then linearly translated to an AFMC formula for $\forall\psi$. The translation of LTL to DBW, however, is doubly-exponential, thus the overall translation is doubly-exponential, with only an exponential matching lower bound [9]. A promising direction for coping with this situation was suggested in [9]: Instead of translating the LTL formula ψ to a DBW, one can translate $\neg\psi$ to an NCW. This can be done either directly, or by translating the NBW for $\neg\psi$ to an equivalent NCW. Then, the NCW can be linearly translated to an AFMC formula for $\exists\neg\psi$, whose negation is equivalent to $\forall\psi$. Thus, a polynomial translation of NBW to NCW would imply a singly-exponential translation of LTL to AFMC.[1]

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 language that is a candidate for proving a lower bound for this translation has to strike a delicate balance: the languages has to somehow take advantage of the Büchi acceptance condition, and still be recognizable by a co-Büchi automaton. In particular, attempts to use the main feature of the Büchi condition, namely its ability to easily track infinitely many occurrences of an event, are almost guaranteed to fail, as a co-Büchi automaton cannot recognize languages that are based on such a tracking. Thus, a candidate language has to use the ability of the Büchi condition to easily track the infinity in some subtle way.

In this paper we point to such a subtle way and provide the first non-trivial lower bound for the translation of NBW to NCW. We show that for every integer $k \geq 1$, there is a language L_k over a two-letter alphabet, such that L_k can be recognized by an NBW with $2k + 1$ states, whereas the minimal NCW that recognizes L_k has $3k$ states. Even though this gap is not asymptotically very significant, it demonstrates for the first time that NBWs are more succinct than NCWs. In addition, our proof points to a conceptual advantage of the Büchi condition: an NBW can abstract precise counting by counting to infinity with two states. To complete the picture, we also study the reverse translation, of NCWs to NBWs. We show that the known upper bound for this translation, which doubles the state space of the NCW, is tight.

2 Preliminaries

2.1 Automata on Infinite Words

Given an alphabet Σ, a *word* over Σ is an infinite sequence $w = \sigma_1 \cdot \sigma_2 \cdots$ of letters in Σ. 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, $\delta : Q \times \Sigma \rightarrow 2^Q$ is a transition function, $Q_0 \subseteq Q$ is a set of initial states, and $\alpha \subseteq Q$ is an acceptance condition. We define several acceptance conditions below. Intuitively, $\delta(q, \sigma)$ is the set of states that $\mathcal{A}$ may move into when it is in the state q and it reads the letter σ. The automaton $\mathcal{A}$ may have several initial states and

[1] Wilke [22] proved an exponential lower-bound for the translation of an NBW for an LTL formula ψ to and 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 is our goal.

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*.

Given two states $p, q \in Q$, a *path* of *length* m from p to q is a finite sequence of states $\pi = \pi_0, \pi_1, \cdots, \pi_{m-1}$ such that $\pi_0 = p$, $\pi_{m-1} = q$, and for every $0 \leq i < m-1$, we have that $\pi_{i+1} \in \bigcup_{\sigma \in \Sigma} \delta(\pi_i, \sigma)$. If $\pi_0 \in \bigcup_{\sigma \in \Sigma} \delta(\pi_{m-1}, \sigma)$ then π is a *cycle*. We say that π is *simple* if all the states of π are different. I.e., if for every $1 \leq i < j < m$, we have that $\pi_i \neq \pi_j$. Let $\pi = \pi_0, \pi_1, \cdots, \pi_{m-1}$ be a simple path of length $m \geq k$. The k*-tail* of π is the set $\{\pi_{m-k}, \ldots, \pi_{m-1}\}$ of the last k states of π. Note that since π is simple the size of its k-tail is k.

A run $r = r_0, r_1, \cdots$ of $\mathcal{A}$ on $w = \sigma_1 \cdot \sigma_2 \cdots \in \Sigma^\omega$ is an infinite sequence of states such that $r_0 \in Q_0$, and for every $i \geq 0$, we have that $r_{i+1} \in \delta(r_i, \sigma_{i+1})$. We sometimes refer to runs as words in Q^ω. 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. Acceptance is defined with respect to the set of states $\mathit{inf}(r)$ that the run r visits infinitely often. Formally, $\mathit{inf}(r) = \{q \in Q \mid \text{ for infinitely many } i \in \mathbb{N}, \text{ we have } r_i = q\}$. As Q is finite, it is guaranteed that $\mathit{inf}(r) \neq \emptyset$. The run r is *accepting* iff the set $\mathit{inf}(r)$ satisfies the acceptance condition α. We consider here the *Büchi* and the *co-Büchi* acceptance conditions. A set $S \subseteq Q$ satisfies a Büchi acceptance condition $\alpha \subseteq Q$ if and only if $S \cap \alpha \neq \emptyset$. Dually, S satisfies a *co-Büchi* acceptance condition $\alpha \subseteq Q$ if and only if $S \cap \alpha = \emptyset$. We say that S is α-free if $S \cap \alpha = \emptyset$. An automaton accepts a word iff it has an accepting run on it. The language of an automaton $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of words that $\mathcal{A}$ accepts. We also say that $\mathcal{A}$ *recognizes* the language $L(\mathcal{A})$. For two automata $\mathcal{A}$ and $\mathcal{A}'$, we say that $\mathcal{A}$ and $\mathcal{A}'$ are *equivalent* if $L(\mathcal{A}) = L(\mathcal{A}')$.

We denote the different classes of automata by three letter acronyms in $\{\mathrm{D}, \mathrm{N}\} \times \{\mathrm{B}, \mathrm{C}\} \times \{\mathrm{W}\}$. The first letter stands for the branching mode of the automaton (deterministic or nondeterministic); the second letter stands for the acceptance-condition type (Büchi, or co-Büchi); the third letter indicates that the automaton runs on words.

Different classes of automata have different expressive power. In particular, while NBWs recognize all ω-regular language [13], DBWs are strictly less expressive than NBWs, and so are DCWs [12]. In fact, a language L can be recognized by a DBW iff its complement can be recognized by a DCW. Indeed, by viewing a DBW as a DCW, we get an automaton for the complementing language, and vice versa. The expressiveness superiority of the nondeterministic model over the deterministic one does not apply to the co-Büchi acceptance condition. There, every NCW has an equivalent DCW.[2]

3 From NBW to NCW

In this section we describe our main result and point to a family of languages $L_1, L_2, \ldots$ such that for all $k \geq 2$, an NBW for L_k requires strictly fewer states than an NCW for L_k.

[2] When applied to universal Büchi automata, the translation in [14], of alternating Büchi automata into NBW, results in DBW. By dualizing it, one gets a translation of NCW to DCW.

3.1 The Languages L_k

We define an infinite family of languages $L_1, L_2, \ldots$ over the alphabet $\Sigma = \{a, b\}$. For every $k \geq 1$, the language L_k is defined as follows:

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

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

Observe that the reason we were able to come up with a very succinct NBW for L_k is that NBW can abstract precise counting by "counting to infinity" with two states. The fact that NCW do not share this ability [12] is what ultimately allows us to prove that NBW are more succinct than NCW. However, it is interesting to note that also NCW for L_k can do much better than $O(k^2)$ states. Even though an NCW cannot validate a guess that a certain letter appears infinitely many times, it does not mean that such a guess is useless. If an NCW guesses that a certain letter appears infinitely many times, then it can postpone counting occurrences of that letter until after it finishes counting k occurrences of the other letter. In other words, $w \in L_k$ iff w has either at least k b's *after* the first k a's, or k a's *after* the first k b's. Following this characterization yields an NCW with two components (corresponding to the two possible guesses) each with two k-counters running sequentially. Since the counters are independent of each other, the resulting NCW has about $4k$ states instead of $O(k^2)$ states. But this is not the end of the story; a more careful look reveals that L_k can also be characterized as follows: $w \in L_k$ iff w has at least k b's *after* the first k a's (this characterizes words in L_k with infinitely many b's), or a finite number of b's that is not smaller than k (this characterizes words in L_k with finitely many b's). Obviously the roles of a and b can also be reversed. As we later show, implementing this idea results in an NCW with $3k + 1$ states. We also show that up to one state this is indeed the best one can do.

3.2 Upper Bounds for L_k

In this section we describe, for every $k \geq 1$, an NBW with $2k + 1$ states and an NCW with $3k + 1$ states that recognize L_k.

Theorem 1. *There is an NBW with $2k + 1$ states that recognizes the language L_k.*

Proof: Consider the automaton in Figure 1. Recall that $w \in L_k$ iff w has at least k b's and infinitely many a's, or at least k a's and infinitely many b's. The lower branch of the automaton checks the first option, and the upper branch checks the second option. Let's focus on the upper branch (a symmetric analysis works for the lower branch). The automaton can reach the state marked t_{k-1} iff it can read $k-1$ a's. From the state t_{k-1} the automaton can continue and accept w, iff w has at least one more a (for a total of at least k a's) and infinitely many b's. Note that from t_k the automaton can only read b. Hence, it moves from t_k to t_{k-1} when it guesses that the current b it reads is the last b in a block of consecutive b's (and thus the next letter in the input is a). Similarly, from t_{k-1} the automaton moves to t_k if it reads an a and guesses that it is the last a in a block of consecutive a's.

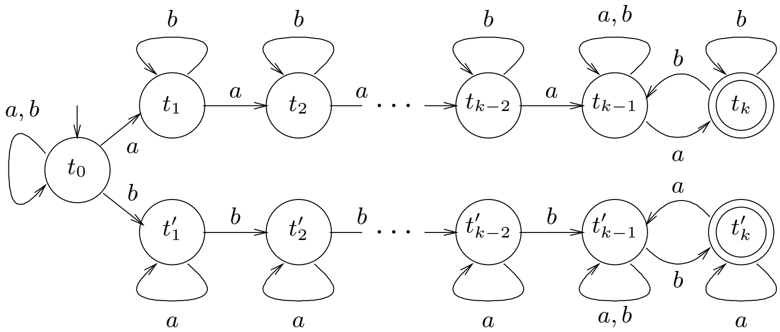

Fig. 1. An NBW for L_k with $2k + 1$ states.

□

Theorem 2. *There is an NCW with* $3k + 1$ *states that recognizes the language* L_k.

Proof: Consider the automaton in Figure 2. Recall that $w \in L_k$ iff w contains at least k b's *after* the first k a's, or a finite number of b's not smaller than k. The upper branch of the automaton checks the first option, and the lower branch checks the second option. It is easy to see that an accepting run using the upper branch first counts k a's, then counts k b's, and finally enters an accepting sink. To see that the lower branch accepts the set of words that have at least k b's, but only finitely many b's, observe that every accepting run using the lower branch proceeds as follows: It stays in the initial state until it guesses that only k b's remain in the input, and then it validates this guess by counting k b's and entering a state from which only a^ω can be accepted. □

Before we turn to study lower bounds for the language L_k, let us note that the strategies used in the NBW and NCW in Figures 1 and 2 are very different. Indeed, the first uses the ability of the Büchi condition to track that an event occurs infinitely often, and the second uses the ability of the co-Büchi condition to track that an event

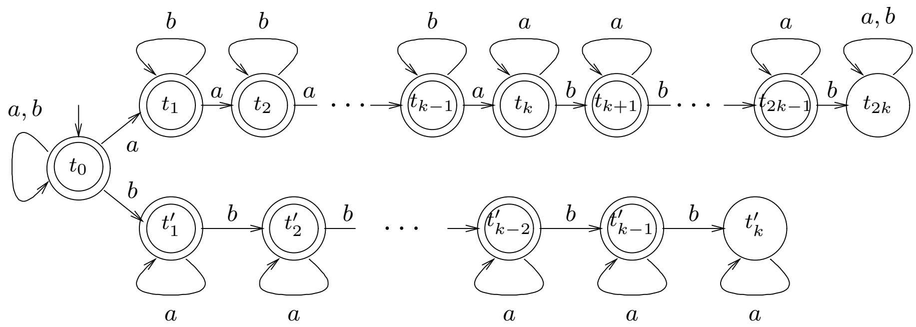

Fig. 2. An NCW for L_k with $3k + 1$ states.

occurs only finitely often. Thus, it is not going to be easy to come up with a general linear translation of NBWs to NCWs that given the NBW in Figure 1 would generate the NCW in Figure 2.

3.3 Lower Bounds for L_k

In this section we prove that the constructions in Section 3.2 are optimal. In particular, this section contains our main technical contribution – a lower bound on the number of states of an NCW that recognizes L_k.

Let $\mathcal{A} = \langle \Sigma, Q, \delta, Q_0, \alpha \rangle$ be an NBW or an NCW that recognizes the language L_k. Let $q_0^a q_1^a q_2^a \cdots$ be an accepting run of $\mathcal{A}$ on the word $a^k b^\omega$, and let $q_0^b q_1^b q_2^b \cdots$ be an accepting run of $\mathcal{A}$ on the word $b^k a^\omega$. Also, let $Q_a = \{q_1^a, q_2^a, \ldots, q_k^a\}$, and $Q_b = \{q_1^b, q_2^b, \ldots, q_k^b\}$. Note that $\mathcal{A}$ may have several accepting runs on $a^k b^\omega$ and $b^k a^\omega$, thus there may be several possible choices of Q_a and Q_b. The analysis below is independent of this choice. Observe that for every $1 \leq i \leq k$, the state q_i^a can be reached from Q_0 by reading a^i, and from it the automaton can accept the word $a^{k-i} b^\omega$. Similarly, the state q_i^b can be reached from Q_0 by reading b^i, and from it the automaton can accept the word $b^{k-i} a^\omega$. A consequence of the above observation is the following lemma.

Lemma 1. *The sets Q_a and Q_b are disjoint, of size k each, and do not intersect Q_0.*

Proof: In order to see that $|Q_a| = k$, observe that if $q_i^a = q_j^a$ for some $1 \leq i < j \leq k$, then $\mathcal{A}$ accepts the word $a^i a^{k-j} b^\omega$, which is impossible since it has less than k a's. A symmetric argument shows that $|Q_b| = k$. In order to see that $Q_a \cap Q_b = \emptyset$, note that if $q_i^a = q_j^b$ for some $1 \leq i, j \leq k$, then $\mathcal{A}$ accepts the word $a^i b^{k-j} a^\omega$, which is impossible since it has less than k b's. Finally, if $q_i^a \in Q_0$ for some $1 \leq i \leq k$, then $\mathcal{A}$ accepts the word $a^{k-i} b^\omega$, which is impossible since it has less than k a's. A symmetric argument shows that $Q_b \cap Q_0 = \emptyset$. $\square$

Since obviously $|Q_0| \geq 1$, we have the following.

Theorem 3. *Every NCW or NBW that recognizes L_k has at least $2k + 1$ states.*

Theorem 3 implies that the upper bound in Theorem 1 is tight, thus the case for NBW is closed. In order to show that every NCW $\mathcal{A} = \langle \Sigma, Q, \delta, Q_0, \alpha \rangle$ that recognizes the language L_k has at least $3k$ states, we prove the next two lemmas.

Lemma 2. *If $\mathcal{A} = \langle \Sigma, Q, \delta, Q_0, \alpha \rangle$ is an NCW that recognizes the language L_k, then there are two (not necessarily different) states $q_a, q_b \notin \alpha$, such that q_a and q_b are reachable from each other using α-free paths, and satisfy that $\mathcal{A}$ can accept the word a^ω from q_a, and the word b^ω from q_b.*

Proof: Let n be the number of states in $\mathcal{A}$, and let $r = r_0, r_1, \cdots$ be an accepting run of $\mathcal{A}$ on the word $(a^n b^n)^\omega$. Observe that since r is an accepting run then $\mathit{inf}(r) \cap \alpha = \emptyset$. Since $\mathcal{A}$ has a finite number of states, there exists $l \geq 0$ such that all the states visited after reading $(a^n b^n)^l$ are in $\mathit{inf}(r)$. Furthermore, there must be a state q_a that appears twice among the $n+1$ states $r_{2nl}, \cdots, r_{2nl+n}$ that r visits while reading the $(l+1)-th$ block of a's. It follows that there is $1 \leq m_a \leq n$ such that q_a can be reached from q_a by reading a^{m_a} while going only through states not in α. Similarly, there is a state $q_b \in \mathit{inf}(r)$ and $1 \leq m_b \leq n$ such that q_b can be reached from q_b by reading b^{m_b} while going only through states not in α. Hence, $\mathcal{A}$ can accept the word $(a^{m_a})^\omega = a^\omega$ from q_a, and the word $(b^{m_b})^\omega = b^\omega$ from q_b. Since q_a and q_b appear infinitely often on the α-free tail $r_{2nl}, \cdots$ of r, they are reachable from each other using α-free paths. □

Note that a similar lemma for NBW does not hold. For example, the NBW in Figure 1 is such that there is no state from which a^ω can be accepted, and that can be reached from a state from which b^ω can be accepted. Also note that there may be several possible choices for q_a and q_b, and that our analysis is independent of such a choice.

Lemma 3. *Every simple path from Q_0 to q_a is of length at least $k+1$ and its k-tail is disjoint from $Q_0 \cup Q_a$. Similarly, every simple path from Q_0 to q_b is of length at least $k+1$ and its k-tail is disjoint from $Q_0 \cup Q_b$.*

Proof: We prove the lemma for a path π from Q_0 to q_a (a symmetric argument works for a path to q_b). By Lemma 2, $\mathcal{A}$ can accept the word a^ω from q_a. Hence, $\mathcal{A}$ must read at least k b's before reaching q_a. This not only implies that π is of length at least $k+1$, but also that no state in the k-tail of π can be reached (in zero or more steps) from Q_0 without reading b's. Since all states in $Q_0 \cup Q_a$ violate this requirement, we are done. □

Lemmas 1 and 3 together imply that if there exists a simple path π from Q_0 to q_a whose k-tail is disjoint from Q_b (alternatively, a simple path from Q_0 to q_b whose k-tail is disjoint from Q_a), then $\mathcal{A}$ has at least $3k+1$ states: Q_0, Q_a, Q_b, and the k-tail of π. The NCW used to establish the upper bound in Theorem 2 indeed has such a path. Unfortunately, this is not the case for every NCW recognizing L_k. However, as the next two lemmas show, if the k-tail of π is α-free we can "compensate" for each state (except for q_k^b) common to π and Q_b, which gives us the desired $3k$ lower bound. The proof of the main theorem then proceeds by showing that if we fail to find a simple path from Q_0 to q_a whose k-tail is disjoint from Q_b, and we also fail to find a simple path from Q_0 to q_b whose k-tail is disjoint from Q_a, then we can find a simple path from Q_0 to q_a whose k-tail is α-free.

Lemma 4. *There is a one-to-one function $f_a : Q_a \setminus (\{q_k^a\} \cup \alpha) \to \alpha \setminus (Q_0 \cup Q_a \cup Q_b)$. Similarly, there is a one-to-one function $f_b : Q_b \setminus (\{q_k^b\} \cup \alpha) \to \alpha \setminus (Q_0 \cup Q_a \cup Q_b)$.*

Proof: We prove the lemma for f_b (a symmetric argument works for f_a). Let n be the number of states in $\mathcal{A}$. Consider some $q_i^b \in Q_b \setminus (\{q_k^b\} \cup \alpha)$. In order to define $f_b(q_i^b)$, take an accepting run $r = r_0, r_1, \cdots$ of $\mathcal{A}$ on the word $b^i a^n b^{k-i} a^\omega$. Among the $n+1$ states $r_i, \cdots, r_{i+n}$ that r visits while reading the sub-word a^n there must be two equal states $r_{i+m} = r_{i+m'}$, where $0 \leq m < m' \leq n$. Since the word $b^i a^m (a^{m'-m})^\omega$ has less than k b's it must be rejected. Hence, there has to be a state $s_i \in \alpha$ along the path $r_{i+m}, \cdots, r_{i+m'}$. We define $f_b(q_i^b) = s_i$. Note that s_i can be reached from Q_0 by reading a word with only i b's, and that $\mathcal{A}$ can accept from s_i a word with only $k-i$ b's. We prove that $s_i \notin Q_0 \cup Q_a \cup Q_b$.

- $s_i \notin Q_0 \cup Q_a \cup \{q_1^b, \ldots q_{i-1}^b\}$ because all states in $Q_0 \cup Q_a \cup \{q_1^b, \ldots q_{i-1}^b\}$ can be reached (in zero or more steps) from Q_0 by reading less than i b's, and from s_i the automaton can accept a word with only $k-i$ b's.
- $s_i \neq q_i^b$ since $s_i \in \alpha$ and $q_i^b \notin \alpha$.
- $s_i \notin \{q_{i+1}^b, \ldots q_k^b\}$ because s_i can be reached from Q_0 by reading a word with only i b's, and from all states in $\{q_{i+1}^b, \ldots q_k^b\}$ the automaton can accept a word with less than $k-i$ b's.

It is left to prove that f_b is one-to-one. To see that, observe that if for some $1 \leq i < j \leq k$ we have that $s_i = s_j$, then the automaton would accept a word with only $i + (k-j)$ b's, which is impossible since $i + (k-j) < k$. $\square$

The following lemma formalizes our counting argument.

Lemma 5. *If there is a simple path π from Q_0 to q_a, or from Q_0 to q_b, such that the k-tail of π is α-free, then $\mathcal{A}$ has at least $3k$ states.*

Proof: We prove the lemma for a path π from Q_0 to q_a (a symmetric argument works for a path to q_b). By Lemma 1, it is enough to find $k-1$ states disjoint from $Q_0 \cup Q_a \cup Q_b$. Let $P \subseteq Q_b$ be the subset of states of Q_b that appear on the k-tail of π, and let R be the remaining $k - |P|$ states of this k-tail. By Lemma 3 we have that R is disjoint from $Q_0 \cup Q_a$, and by definition it is disjoint from Q_b. We have thus found $k - |P|$ states disjoint from $Q_0 \cup Q_a \cup Q_b$. It remains to find a set of states S which is disjoint from $Q_0 \cup Q_a \cup Q_b \cup R$, and is of size at least $|P| - 1$. Since the k-tail of π is α-free, it follows from Lemma 4 that for every state q_i^b in P, except maybe q_k^b, there is a "compensating" state $f_b(q_i^b) \in \alpha \setminus (Q_0 \cup Q_a \cup Q_b)$. We define S to be the set $S = \bigcup_{\{q_i^b \in P, q_i^b \neq q_k^b\}} \{f_b(q_i^b)\}$ of all these compensating states. Since f_b is one-to-one S is of size at least $|P| - 1$. Since R is α-free and $S \subseteq \alpha$ it must be that S is also disjoint from R, and we are done. $\square$

We are now ready to prove our main theorem.

Theorem 4. *Every NCW that recognizes the language L_k has at least $3k$ states.*

Proof: As noted earlier, by Lemmas 1 and 3, if there exists a simple path from Q_0 to q_a whose k-tail is disjoint from Q_b, or if there exists a simple path from Q_0 to q_b whose k-tail is disjoint from Q_a, then $\mathcal{A}$ has at least $3k+1$ states: Q_0, Q_a, Q_b, and the k-tail of this path. We thus assume that on the k-tail of every simple path from Q_0 to q_a there is a state from Q_b, and that on the k-tail of every simple path from Q_0 to q_b there is a state from Q_a. Note that since by Lemma 3 the k-tail of every simple path from Q_0 to q_b is disjoint from Q_b, it follows from our assumption that $q_a \neq q_b$.

Another consequence of our assumption is that q_a is reachable from Q_b. Take an arbitrary simple path from Q_b to q_a, let q_i^b be the last state in Q_b on this path, and let $q_i^b = v_0, \cdots, v_h = q_a$ be the tail of this path starting at q_i^b. Note that if $q_a \in Q_b$ then $h = 0$. Define π^a to be the path $q_0^b, \cdots, q_i^b, v_1, \cdots, v_h$. Observe that by Lemma 1, and the fact that $v_1, ..., v_h$ are not in Q_b, the path π^a is simple. Hence, by our assumption, the k-tail of π^a intersects Q_b. Since $v_1, ..., v_h$ are not in Q_b, it must be that $h < k$.

By Lemma 2, q_b is reachable from q_a without using states in α. Thus, there exists a simple α-free path $q_a = u_0, ..., u_m = q_b$. Since $u_0 = q_a \in \pi^a$, we can take $0 \leq j \leq m$ to be the maximal index such that u_j appears on π^a. Define the path π^b, from Q_0 to q_b, to be the prefix of π^a until (but not including) u_j, followed by the path $u_j, ..., u_m$. Note that π^b is a simple path since by our choice of u_j it is the concatenation of two disjoint simple paths. Hence, by our assumption, there is some state $q_j^a \in Q_a$ on the k-tail of π^b. We claim that q_j^a must be on the α-free tail $u_j, ..., u_m$ of π^b. Recall that all the states in π^b before u_j are also in π^a, so it is enough to prove that q_j^a is not in π^a. By Lemma 1, q_j^a cannot be equal to any of the first $i+1$ states of π^a. By Lemma 3, and the fact that $h < k$, it cannot be equal to any of the remaining h states of π^a. We can thus conclude that the tail of π^b starting at q_j^a is α-free.

We are now in a position to build a new simple path π from Q_0 to q_a, whose k-tail is α-free. By Lemma 5, this completes the proof. We first define a path π' from Q_0 to q_a by concatenating to the path $q_0^a, q_1^a, \cdots q_{j-1}^a$ the tail of π^b starting at q_j^a, followed by some α-free path from q_b to q_a (by Lemma 2 such a path exists). Since π' may have repeating states, we derive from it the required simple path π by eliminating repetitions in an arbitrary way. Observe that the only states in π' (and thus also in π) that may be in α are the states $\{q_0^a, q_1^a, \ldots q_{j-1}^a\}$. By Lemma 3, the k-tail of π is disjoint from $Q_0 \cup Q_a$. Hence, it must be α-free. □

Combining the upper bound in Theorem 1 with the lower bound in Theorem 4, we get the following corollary.

Corollary 1. *For every integer $k \geq 1$, there is a language L_k over a two-letter alphabet, such that L_k can be recognized by an NBW with $2k+1$ states, whereas the minimal NCW that recognizes L_k has $3k$ states.*

4 From NCW to NBW

As shown in Section 3, NBWs are more succinct than NCWs. In this section we study the translation of NCW to NBW and show that the converse is also true. That is, we show that the known construction that translates an NCW with n states and acceptance condition α, to an equivalent NBW with $2n - |\alpha|$ states, is tight. For reference, we first briefly recall this translation. The translation we present follows [10], which complements deterministic Büchi automata.

Theorem 5. *[10] Given an NCW $\mathcal{A} = \langle \Sigma, Q, \delta, Q_0, \alpha \rangle$ with n states, one can build an equivalent NBW $\mathcal{A}'$ with $2n - |\alpha|$ states.*

Proof: The NBW $\mathcal{A}'$ is built by taking two copies of $\mathcal{A}$, deleting all the states in α from the second copy, and making all the remaining states of the second copy accepting. Transitions are also added to enable the automaton to move from the first copy to the second copy, but not back. The idea is that since an accepting run of $\mathcal{A}$ visits states in α only finitely many times, it can be simulated by a run of $\mathcal{A}'$ that switches to the second copy when states in α are no longer needed. More formally, $\mathcal{A}' = \langle \Sigma, (Q \times \{0\}) \cup ((Q \setminus \alpha) \times \{1\}), \delta', Q_0 \times \{0\}, (Q \setminus \alpha) \times \{1\} \rangle$, where for every $q \in Q$ and $\sigma \in \Sigma$ we have $\delta'(\langle q, 0 \rangle, \sigma) = (\delta(q, \sigma) \times \{0\}) \cup ((\delta(q, \sigma) \setminus \alpha) \times \{1\}$, and for every $q \in Q \setminus \alpha$ and $\sigma \in \Sigma$ we have $\delta'(\langle q, 1 \rangle, \sigma) = (\delta(q, \sigma) \setminus \alpha) \times \{1\}$. $\square$

Observe that if $\alpha = \emptyset$, then $L(\mathcal{A}) = \Sigma^*$, and the translation is trivial. Hence, the maximal possible blowup is when $|\alpha| = 1$. In the remainder of this section we prove that there are NCWs (in fact, DCWs with $|\alpha| = 1$) for which the $2n - |\alpha|$ blowup cannot be avoided.

4.1 The Languages L'_k

We define a family of languages $L'_2, L'_3, \ldots$ over the alphabet $\Sigma = \{a, b\}$. For every $k \geq 2$ we let $L'_k = (a^k b^k + a^k b^{k-1})^* (a^k b^{k-1})^\omega$. Thus, a word $w \in \{a, b\}^\omega$ is in L'_k iff w begins with an a, all the blocks of consecutive a's in w are of length k, all the blocks of consecutive b's in w are of length k or $k - 1$, and only finitely many blocks of consecutive b's in w are of length k. Intuitively, an automaton for L'_k must be able to count finitely many times up to $2k$, and infinitely many times up to $2k - 1$. The key point is that while a co-Büchi automaton can share the states of the $2k - 1$ counter with those of the $2k$ counter, a Büchi automaton cannot.

4.2 Upper bounds for L'_k

We first describe an NCW (in fact, a DCW) with $2k$ states that recognizes the language L'_k. By Theorem 5, one can derive from it an equivalent NBW with $4k - 1$ states.

Theorem 6. *There is a DCW with $2k$ states that recognizes the language L'_k.*

Proof: Consider the automaton in Figure 3. It is obviously deterministic, and it is easy to see that it accepts the language $a^k b^{k-1} (a^k b^{k-1} + b a^k b^{k-1})^* (a^k b^{k-1})^\omega = (a^k b^k + a^k b^{k-1})^* (a^k b^{k-1})^\omega = L'_k$. $\square$

Note that the NCW in Figure 3 is really a DCW, thus the lower bound we are going to prove is for the DCW to NBW translation. It is worth noting that the dual translation, of DBW to NCW (when exists), involves no blowup. Indeed, if a DBW $\mathcal{A}$ recognizes a language that is recognizable by an NCW, then this language is also recognizable by a DCW, and there is a DCW on the same structure as $\mathcal{A}$ for it [6, 8].

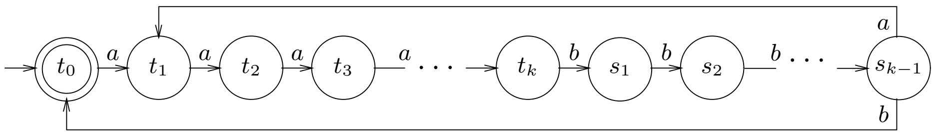

Fig. 3. A DCW for L'_k with $2k$ states.

4.3 Lower bounds for L'_k

In this section we prove that the NBW obtained by applying the construction in Theorem 5 to the automaton in Figure 3, is optimal. Thus, every NBW for L'_k has at least $4k-1$ states. Note that this also implies that the upper bound in Theorem 6 is tight too.

We first show that an automaton for L'_k must have a cycle along which it can count to $2k$, for the purpose of keeping track of occurrences of $a^k b^k$ in the input.

Lemma 6. *If $\mathcal{A} = \langle \Sigma, Q, \delta, Q_0, \alpha \rangle$ is an NCW or an NBW that recognizes the language L'_k, then there is a cycle C, reachable from Q_0, with at least $2k$ different states, along which $\mathcal{A}$ can traverse a finite word containing the substring $a^k b^k$.*

Proof: Let n be the number of states in $\mathcal{A}$, and let $r = r_0, r_1, \cdots$ be an accepting run of $\mathcal{A}$ on the word $w = (a^k b^k)^{n+1}(a^k b^{k-1})^\omega$. Since $\mathcal{A}$ has only n states, there must be $1 \leq i < j \leq n+1$, such that $r_{i2k} = r_{j2k}$. Consider the cycle $C = r_{i2k}, \cdots, r_{j2k-1}$. Note that $r_0 \in Q_0$ and thus C is reachable from Q_0. Also note that $j - i \geq 1$, and that $\mathcal{A}$ can traverse $(a^k b^k)^{j-i}$ along C.

We now prove that the states $r_{i2k}, \cdots, r_{(i+1)2k-1}$ are all different, thus C has at least $2k$ different states. Assume by way of contradiction that this is not the case, and let $0 \leq h < l \leq 2k-1$ be such that $r_{i2k+h} = r_{i2k+l}$. Define $u = a^k b^k$, and let $u = xyz$, where $x = u_1 \cdots u_h$, $y = u_{h+1} \cdots u_l$, and $z = u_{l+1} \cdots u_{2k}$. Observe that x and z may be empty, and that since $0 \leq h < l \leq 2k-1$, it must be that $0 < |y| < 2k$. Also note that $\mathcal{A}$ can traverse x along $r_{i2k} \cdots r_{i2k+h}$, and traverse y along the cycle $\hat{C} = r_{i2k+h}, \cdots, r_{i2k+l-1}$. By adding k more traversals of the cycle $\hat{C}$ we can derive from r a run $r' = r_0 \cdots r_{i2k+h} \cdot (r_{i2k+h+1} \cdots r_{i2k+l})^{k+1} \cdot r_{i2k+l+1} \cdots$ on the word $w' = (a^k b^k)^i x y^{k+1} z (a^k b^k)^{n-i}(a^k b^{k-1})^\omega$. Similarly, by removing from r a traversal of $\hat{C}$, we can derive a run $r'' = r_0 \cdots r_{i2k+h} r_{i2k+l+1} \cdots$ on the word $w'' = (a^k b^k)^i xz(a^k b^k)^{n-i}(a^k b^{k-1})^\omega$. Since $\mathit{inf}(r) = \mathit{inf}(r') = \mathit{inf}(r'')$, and r is accepting, so are r' and r''. Hence, w' and w'' are accepted by $\mathcal{A}$.

To derive a contradiction, we show that $w' \notin L'_k$ or $w'' \notin L'_k$. Recall that $xyz = a^k b^k$ and that $0 < |y| < 2k$. Hence, there are two cases to consider: either $y \in a^+ + b^+$, or $y \in a^+ b^+$. In the first case we get that y^{k+1} contains either a^{k+1} or b^{k+1}, which implies that $w' \notin L'_k$. Consider now the case $y \in a^+ b^+$. Let $y = a^m b^t$. Since $i > 0$, the prefix $(a^k b^k)^i xza^k$ of w'' ends with $b^k a^{k-m} b^{k-t} a^k$. Since all the consecutive blocks of a's in w must be of length k, and $m > 0$, it must be that $k - m = 0$. Hence, w'' contains the substring $b^k b^{k-t}$. Recall that $k = m$, and that $m + t < 2k$. Thus, $k - t > 0$, and

$b^k b^{k-t}$ is a string of more than k consecutive b's. Since no word in L'_k contains such a substring, we are done. $\square$

The following lemma shows that an NBW recognizing L'_k must have a cycle going through an accepting state along which it can count to $2k-1$, for the purpose of recognizing the $(a^k b^{k-1})^\omega$ tail of words in L'_k.

Lemma 7. *If $\mathcal{A} = \langle \Sigma, Q, \delta, Q_0, \alpha \rangle$ is an NBW that recognizes the language L'_k, then $\mathcal{A}$ has a cycle C, with at least $2k-1$ different states, such that $C \cap \alpha \neq \emptyset$.*

Proof: Since $L(\mathcal{A})$ is not empty, there must be a state $c_0 \in \alpha$ that is reachable from Q_0, and a simple cycle $C = c_0, \cdots, c_{m-1}$ going through c_0. Since C is simple, all its states are different. It remains to show that $m \geq 2k-1$. Let $u \in \Sigma^*$ be such that $\mathcal{A}$ can reach c_0 from Q_0 while reading u, and let $v = \sigma_1 \cdots \sigma_m$ be such that $\mathcal{A}$ can traverse v along C. It follows that $w = uv^\omega$ is accepted by $\mathcal{A}$. Since all words in L'_k have infinitely many a's and b's, it follows that a and b both appear in v. We can thus let $1 \leq j < m$ be such that $\sigma_j \neq \sigma_{j+1}$. Let x be the substring $x = \sigma_j \cdots \sigma_m \sigma_1 \cdots \sigma_{j+1}$ of vv. Since $\sigma_j \neq \sigma_{j+1}$, it must be that x contains one block of consecutive letters all equal to σ_{j+1} that *starts* at the second letter of x, and another block of consecutive letters all equal to σ_j that *ends* at the letter before last of x. Since $|x| = m+2$ we have that x contains at least one block of consecutive a's and one block of consecutive b's that start and end within the span of m letters. Recall that since $w \in L'_k$ then all the blocks of consecutive a's in v^ω must be of length k, and all the blocks of consecutive b's in v^ω must be of length at least $k-1$. Hence, $m \geq k + k - 1$. $\square$

Theorem 7. *Every NBW $\mathcal{A} = \langle \Sigma, Q, \delta, Q_0, \alpha \rangle$ that recognizes the language L'_k has at least $4k-1$ states.*

Proof: By Lemma 6, there is a cycle $C = c_0, \cdots, c_{n-1}$, reachable from Q_0, with at least $2k$ different states, along which $\mathcal{A}$ can read some word $z = z_1 \cdots z_n$ containing $a^k b^k$. By Lemma 7, there is a cycle $C' = c'_0, \cdots, c'_{m-1}$, with at least $2k-1$ different states, going through an accepting state $c'_0 \in \alpha$. In order to prove that $\mathcal{A}$ has at least $4k-1$ states, we show that C and C' are disjoint. Assume by way of contradiction that there is a state $q \in C \cap C'$. By using q as a pivot we can construct a run r that alternates infinitely many times between the cycles C and C'. Since C' contains an accepting state, the run r is accepting. To reach a contradiction, we show that r is a run on a word containing infinitely many occurrences of b^k, and thus it must be rejecting. Let $0 \leq l < n$ and $0 \leq h < m$ be such that $q = c_l = c'_h$, and let $q_0, \cdots, q_t$ be a path from Q_0 to q (recall that C is reachable from Q_0). Consider the run $r = q_0 \cdots q_{t-1}(c'_h \cdots c'_{m-1} c'_0 \cdots c'_{h-1} c_l \cdots c_{n-1} c_0 \cdots c_{n-1} c_0 \cdots c_{l-1})^\omega$. Let $x, y \in \Sigma^*$ be such that $\mathcal{A}$ can read x along the path $q_0, \cdots q_t$, and read y while going from c'_h back to itself along the cycle C'. Observe that r is a run of $\mathcal{A}$ on the word $w = x \cdot (y \cdot z_{l+1} \cdots z_n \cdot z \cdot z_1 \cdots z_l)^\omega$. Since $c'_0 \in \alpha$ and r goes through c'_0 infinitely many times, r is an accepting run of $\mathcal{A}$ on w. Since w contains infinitely many occurrences of z it contains infinitely many occurrences of b^k, and thus $w \notin L'_k$, which is a contradiction. $\square$

Combining the upper bound in Theorem 6 with the lower bound in Theorem 7 we get the following corollary:

Corollary 2. *For every integer $k \geq 2$, there is a language L'_k over a two-letter alphabet, such that L'_k can be recognized by a DCW with $2k$ states, whereas the minimal NBW that recognizes L'_k has $4k - 1$ states.*

5 Discussion

We have shown that NBWs are more succinct than NCWs. The advantage of NBWs that we used is their ability to save states by counting to infinity with two states instead of counting to k, for some parameter k. The bigger k is, the bigger is the saving. In our lower bound proof, k is linear in the size of the state space. Increasing k to be exponential in the size of the state space would lead to an exponential lower bound for the NBW to NCW translation. Once we realized this advantage of the Büchi condition, we tried to find an NBW that uses a network of nested counters in a way that would enable us to increase the relative size of k. We did not find such an NBW, and we conjecture that the succinctness of the Büchi condition cannot go beyond saving one copy of the state space. Let us elaborate on this.

The best known upper bound for the NBW to NCW translation is still exponential, and the upper bound for the NCW to NBW translation is linear. Still, it was much easier to prove the succinctness of NCWs with respect to NBWs (Section 4) than the succinctness of NBWs with respect to NCWs (Section 3). Likewise, DCWs are more succinct than NBWs (Section 4), whereas DBWs are not more succinct than NCWs [6]. The explanation for this quite counterintuitive "ease of succinctness" of the co-Büchi condition is the expressiveness superiority of the Büchi condition. Since every NCW has an equivalent NBW, all NCWs are candidates for proving the succinctness of NCW. On the other hand, only NBWs that have an equivalent NCW are candidates for proving the succinctness of NBWs. Thus, the candidates have to take an advantage of the strength of the Büchi condition, but at the same time be restricted to the co-Büchi condition. This restriction has caused researchers to believe that NBWs are actually co-Büchi-type (that is, if an NBW has an equivalent NCW, then it also has an equivalent NCW on the same structure). The results in [8] refuted this hope, and our results here show that NBWs can actually use their expressiveness superiority for succinctness. While the results are the first to show such a succinctness, our fruitless efforts to improve the lower bound further have led us to believe that NBWs cannot do much more than abstracting counting up to the size of the state space. Intuitively, as soon as the abstracted counting goes beyond the size of the state space, the language has a real "infinitely often" nature, and it is not recognizable by an NCW. Therefore, our future research focuses on improving the upper bound. The very different structure and strategy behind the NBW and NCW in Figures 1 and 2 hint that this is not going to be an easy journey either.

References

1. R. Armoni, L. Fix, A. Flaisher, R. Gerth, B. Ginsburg, T. Kanza, A. Landver, S. Mador-Haim, E. Singerman, A. Tiemeyer, M.Y. Vardi, and Y. Zbar. The ForSpec temporal logic: A new

temporal property-specification logic. In *Proc. 8th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, LNCS 2280, pages 196–211. Springer, 2002.
2. I. Beer, S. Ben-David, C. Eisner, D. Fisman, A. Gringauze, and Y. Rodeh. The temporal logic Sugar. In *Proc 13th Int. Conf. on Computer Aided Verification*, LNCS 2102, pages 363–367. Springer, 2001.
3. J.R. Büchi. On a decision method in restricted second order arithmetic. In *Proc. Int. Congress on Logic, Method, and Philosophy of Science. 1960*, pages 1–12. Stanford University Press, 1962.
4. E.A. Emerson and C. Jutla. The complexity of tree automata and logics of programs. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 328–337, 1988.
5. Accellera Organization Inc. http://www.accellera.org, 2006.
6. S.C. Krishnan, A. Puri, and R.K. Brayton. Deterministic ω-automata vis-a-vis deterministic Büchi automata. In *Algorithms and Computations*, LNCS 834, pages 378–386, 1994.
7. O. Kupferman. Tightening the exchange rate beteen automata. In *Proc. 16th Annual Conf. of the European Association for Computer Science Logic*, LNCS 4646, pages 7–22, 2007.
8. O. Kupferman, G. Morgenstern, and A. Murano. Typeness for ω-regular automata. In *2nd Int. Symp. on Automated Technology for Verification and Analysis*, LNCS 3299, pages 324–338. Springer, 2004.
9. O. Kupferman and M.Y. Vardi. From linear time to branching time. *ACM Transactions on Computational Logic*, 6(2):273–294, 2005.
10. R.P. Kurshan. Complementing deterministic Büchi automata in polynomial time. *Journal of Computer and Systems Science*, 35:59–71, 1987.
11. R.P. Kurshan. *Computer Aided Verification of Coordinating Processes*. Princeton Univ. Press, 1994.
12. L.H. Landweber. Decision problems for ω–automata. *Mathematical Systems Theory*, 3:376–384, 1969.
13. R. McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9:521–530, 1966.
14. S. Miyano and T. Hayashi. Alternating finite automata on ω-words. *Theoretical Computer Science*, 32:321–330, 1984.
15. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. 16th ACM Symp. on Principles of Programming Languages*, pages 179–190, 1989.
16. M.O. Rabin. Decidability of second order theories and automata on infinite trees. *Transaction of the AMS*, 141:1–35, 1969.
17. K. Ravi, R. Bloem, and F. Somenzi. A comparative study of symbolic algorithms for the computation of fair cycles. In *Proc. 3rd Int. Conf. on Formal Methods in Computer-Aided Design*, LNCS 1954, pages 143–160. Springer, 2000.
18. S. Safra. On the complexity of ω-automata. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 319–327, 1988.
19. R.S. Street and E.A. Emerson. An elementary decision procedure for the μ-calculus. In *Proc. 11th Int. Colloq. on Automata, Languages, and Programming*, LNCS 172, pages 465–472. Springer, 1984.
20. M.Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and Systems Science*, 32(2):182–221, 1986.
21. M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.
22. T. Wilke. CTL^+ is exponentially more succinct than CTL. In *Proc. 19th Conf. on Foundations of Software Technology and Theoretical Computer Science*, LNCS 1738, pages 110–121. Springer, 1999.
23. P. Wolper, M.Y. Vardi, and A.P. Sistla. Reasoning about infinite computation paths. In *Proc. 24th IEEE Symp. on Foundations of Computer Science*, pages 185–194, 1983.