Alternation Removal in Büchi Automata

Udi Boker, Orna Kupferman and Adin Rosenberg

School of Computer Science and Engineering, Hebrew University, Israel.

Abstract. Alternating automata play a key role in the automata-theoretic approach to specification, verification, and synthesis of reactive systems. Many algorithms on alternating automata, and in particular, their nonemptiness test, involve removal of alternation: a translation of the alternating automaton to an equivalent nondeterministic one. For alternating Büchi automata, the best known translation uses the "breakpoint construction" and involves an $O(3^n)$ state blow-up. The translation was described by Miyano and Hayashi in 1984, and is widely used since, in both theory and practice. Yet, the best known lower bound is only 2^n.

In this paper we develop and present a complete picture of the problem of alternation removal in alternating Büchi automata. In the lower bound front, we show that the breakpoint construction captures the accurate essence of alternation removal, and provide a matching $\Omega(3^n)$ lower bound. Our lower bound holds already for universal (rather than alternating) automata with an alphabet of a constant size. In the upper-bound front, we point to a class of alternating Büchi automata for which the breakpoint construction can be replaced by a simpler $n2^n$ construction. Our class, of ordered alternating Büchi automata, strictly contains the class of very-weak alternating automata, for which an $n2^n$ construction is known.

1 Introduction

The automata-theoretic approach to formal verification uses automata on infinite words and trees in order to model systems and their specifications. By translating specifications to automata, we can reduce problems like satisfiability and model checking to the nonemptiness and containment problems of automata. The complexity of the automata-based algorithms is induced by both the blow-up involved in the translation of specifications to automata, and the complexity of the nonemptiness and containment problems for them. The automata-theoretic approach has proven to be extremely useful and popular in practice [1, 22].

Early translations of temporal-logic formulas to automata use nondeterministic automata. The transition function of a nondeterministic word automaton suggests several successor states to each state and letter, and an input word is accepted by the automaton if some run on it is accepting. The translation of LTL to nondeterministic Büchi automata (NBW, for short) is exponential [14, 23]. Since the nonemptiness problem for NBWs can be solved in NLOGSPACE, the translation suggested a PSPACE upper bound for the model-checking and satisfiability problems of LTL [14, 23].

In the early 90s, researchers started to base the automata-theoretic approach on *alternating automata* [19, 20]. In an alternating automaton, the transition function maps a state and a letter to a formula over the set of states, indicating by which states the suffix of the word should be accepted. For example, if $\delta(q_0, a) = q_1 \wedge (q_2 \vee q_3)$, then when the automaton is in state q_0 and reads the letter a, then the suffix of the word should be accepted both from the state q_1 and from either q_2 or q_3. Thus, several copies of the automaton run on the input word. As shown in [4, 13], the translation of temporal logic to alternating automata is simple and involves no blow-up. Accordingly, the complexity is shifted to the nonemptiness problem, which is harder for alternating automata, and involves removal of alternation; that is, a translation to an equivalent nondeterministic automaton. For alternating Büchi automata (ABWs, for short), such a translation involves an exponential blow-up [16], leading to a PSPACE nonemptiness algorithm, which is tight.

It turns out that the use of intermediate alternating automata has many advantages. In some cases, such as branching-time model checking, one can reason about the alternating automaton without removing alternation [13]. In LTL, the use of intermediate alternating automata enables further optimizations on the translation of LTL to NBW [8, 9, 21], and has led to improved minimization algorithms for NBWs [5, 6]. In addition, postponing the removal of alternation to later stages of the algorithms has led to simplified decision and synthesis procedures [7, 12].

Consider an alternating automaton $\mathcal{A}$ with state space Q, transition function δ, and set α of accepting states. Removal of alternation in $\mathcal{A}$ has the flavor of removal of nondeterminism in nondeterministic automata. As there, the constructed automaton follows the subset construction applied to $\mathcal{A}$. Here, however, when the constructed automaton is in a state associated with a subset $S \subseteq Q$, the input word should be accepted from all the states in S, and there may be several successors to the state associated with S. For example, if $\delta(q_0, a) = q_1 \wedge (q_2 \vee q_3)$, then in an equivalent nondeterministic automaton, the transition function would map a state associated with the set $\{q_0\}$ and the letter a to a nondeterministic choice between the two states associated with $\{q_1, q_2\}$ or $\{q_1, q_3\}$. In the case of finite words, it is easy to see that defining the set α' of accepting states to be these associated with sets contained in α results in an equivalent nondeterministic automaton.

The case of infinite words is more difficult. Defining α' as above does not work, as it forces the different copies of $\mathcal{A}$ to visit α simultaneously. Also, it is not clear whether a "round-robin" examination of the copies (as done in the case of NBW intersection) is possible, as the number of copies is not bounded. A procedure for alternation removal in ABWs was suggested in 1984 by Miyano and Hayashi [16]. The idea behind the procedure, known as the *breakpoint construction*, is that the states of the equivalent NBW maintain, in addition to the set S associated with the subset construction, also a set $O \subseteq S \setminus \alpha$ of states along runs that "owe" a visit to the set of accepting states. [1] Thus, starting with an ABW with n states, the breakpoint construction ends up in an NBW with at most 3^n states. While the construction is only exponential (one could have expected a $2^{O(n \log n)}$ blow-up, as is the case of complementation or determinization of NBWs [15]), it is conceptually different from the simple subset construction. In particular, it is annoying that the construction does not make use of the fact that the

[1] The direct translations of LTL to NBW, which do not go via ABWs, implement a similar breakpoint construction, by means of an "eventuality automaton" [23].

Büchi condition is memoryless, which suggests that we do not have to run more than n copies. In addition, from a practical point of view, the need to maintain two subsets makes the state space exponentially bigger and makes the implementation of the breakpoint construction difficult and complex [2, 5, 10, 17].

These drawbacks of the breakpoint construction, and its performance in practice for some natural specifications have led Gastin and Oddoux to develop an alternative translation of LTL to NBW [10]. The new translation is based on the fact that the ABWs that correspond to LTL formulas are *very weak*, in the sense that all the cycles in them are of size one (in other words, the only cycles are self-loops). It is shown in [10] that for very weak ABWs, one can replace the breakpoint construction by a simpler construction, with only an $n2^n$ blow-up.

In this paper we develop and present a complete picture of the problem of alternation removal in ABWs. In the lower bound front, we show that the breakpoint construction of [16] and its $\Omega(3^n)$ blow-up cannot be avoided. In the upper-bound front, we point to a class of ABWs that is strictly more expressive than very-weak ABW and for which the breakpoint construction can be replaced by a simpler $n2^n$ construction. Below we elaborate on the two contributions.

First, we show that the concept of the breakpoint construction captures the accurate essence of alternation removal in ABWs. Thus, there is a need to associate the states of the equivalent NBW with two sets, and the $\Omega(3^n)$ blow-up cannot be avoided. Technically, we describe a family of languages L_n such that L_n can be recognized by an alternating (in fact, a universal) Büchi automaton with n states, whereas an equivalent NBW requires at least $\frac{1}{6} \cdot 3^n$ states.[2] This solves negatively the long-standing open problem of improving the breakpoint construction to one with an $O(2^n)$ blow-up. As in [24], our lower-bound proof starts with automata with an exponential alphabet, which we then encode using a fixed-size alphabet. We show that the $\Omega(3^n)$ lower bound applies also to the determinization of nondeterministic co-Büchi word automata and for alternation removal in alternating Büchi tree automata [18].

Second, we introduce *ordered automata* and show that alternation removal in ordered ABWs can avoid the breakpoint construction and involves only an $n2^n$ blow-up. Essentially, an automaton is ordered if the only rejecting cycles induced by its transition function are self loops. Note that all very weak ABWs are ordered, but not vice versa. Indeed, in ordered automata we have no restrictions on cycles that contain accepting states. Ordered automata are strictly more expressive than very weak ABWs. For example, the specifications "p holds in all even positions" and "whenever there is *request*, then *try* and *ack* alternate until *grant* is valid" can be specified by an ordered ABW but not by a very weak ABW. As the above specifications demonstrate, ordered ABWs can handle regular specifications, which are strictly more expressive than LTL and are indeed very popular in modern specification formalisms [3]. Thus, our results extend the fragment of automata for which the breakpoint construction can be avoided. The order condition enables the equivalent NBW to examine the states of the ABW that are not in α in a round-robin fashion: whenever the NBW is in a state associated with a set S of states, it examines a single state $p \in S \setminus \alpha$ and makes sure that no path in the run of

[2] The $\frac{1}{6}$ constant can be reduced and probably also eliminated by some more technical work, which we do not find interesting enough.

the ABW gets trapped in p: as long as p is a successor of itself, it keeps examining p. Only when a chain of p's ends, the NBW changes the examined state. The acceptance condition then makes sure that the NBW does not get trapped in a rejecting state.

We study the expressive power of ordered automata and argue that the order condition defines a fragment of automata for which the breakpoint construction can be avoided. We also show that the $n2^n$ upper bound for the translation of ordered ABWs to NBWs is tight, thus even for ordered automata one needs to augment the subset construction with additional information. Finally, we show that for ordered universal Büchi automata, we can replace the examined state by a subset of letters that are examined, resulting in an alternative construction with blow-up 2^{n+m}, where m is the size of the alphabet. This is in contrast with many translations in automata-theory (c.f. [24], as well as our lower bound proof here), where moving to an alphabet of a constant size does not change the state blow-up.

2 Preliminaries

Given an alphabet Σ, an *infinite word over Σ* is an infinite sequence $w = \sigma_0 \cdot \sigma_1 \cdots \sigma_2 \cdots$ of letters in Σ. For a word w and two indices $t_1, t_2 \geq 0$, we denote by $w[t_1, t_2]$ its subword $\sigma_{t_1} \cdot \sigma_{t_1+1} \cdots \sigma_{t_2}$. In particular, $w[0, t_1]$ is the prefix $\sigma_0 \cdot \sigma_1 \cdots \sigma_{t_1}$ of w, and $w[t_2, \infty]$ is its suffix $\sigma_{t_2} \cdot \sigma_{t_2+1} \cdots$.

For a given set X, let $\mathcal{B}^+(X)$ be the set of positive Boolean formulas over X (i.e., Boolean formulas built from elements in X using $\wedge$ and $\vee$), where we also allow the formulas $\mathbf{true}$ and $\mathbf{false}$. For $Y \subseteq X$, we say that Y *satisfies* a formula $\theta \in \mathcal{B}^+(X)$ iff the truth assignment that assigns *true* to the members of Y and assigns *false* to the members of $X \setminus Y$ satisfies θ. An *alternating Büchi automaton on infinite words* is a tuple $\mathcal{A} = \langle \Sigma, Q, q_{in}, \delta, \alpha \rangle$, where Σ is the input alphabet, Q is a finite set of states, $q_{in} \in Q$ is an initial state, $\delta : Q \times \Sigma \to \mathcal{B}^+(Q)$ is a transition function, and $\alpha \subseteq Q$ is a set of accepting states. We define runs of $\mathcal{A}$ by means of infinite DAGs (directed acyclic graphs).[3] A run of $\mathcal{A}$ on a word $w = \sigma_0 \cdot \sigma_1 \cdots$ is an infinite DAG $\mathcal{G} = \langle V, E \rangle$ satisfying the following (note that there may be several runs of $\mathcal{A}$ on w).

– $V \subseteq Q \times \mathbb{N}$ is as follows. Let $Q_l \subseteq Q$ denote all states in level l. Thus, $Q_l = \{q : \langle q, l \rangle \in V\}$. Then, $Q_0 = \{q_{in}\}$, and Q_{l+1} satisfies $\bigwedge_{q \in Q_l} \delta(q, \sigma_l)$.
– $E \subseteq \bigcup_{l \geq 0} (Q_l \times \{l\}) \times (Q_{l+1} \times \{l+1\})$ is such that $E(\langle q, l \rangle, \langle q', l+1 \rangle)$ iff $Q_{l+1} \setminus \{q'\}$ does not satisfy $\delta(q, \sigma_l)$.

Thus, the root of the DAG contains the initial state of the automaton, and the states associated with nodes in level $l+1$ satisfy the transitions from states corresponding to nodes in level l. For a set $S \subseteq Q$, a node $\langle q, i \rangle \in V$ is an S-node if $q \in S$. The run $\mathcal{G}$ accepts the word w if all its infinite paths satisfy the acceptance condition α. Thus, in the case of Büchi automata, all the infinite paths have infinitely many α-nodes. We sometimes refer also to co-Büchi automata, where a run is accepting iff all its paths

[3] In general, runs of alternating automata are defined by means of infinite trees. Since we are going to deal only with acceptance conditions that have memoryless runs, we can work instead with DAGs [4, 11].

have only finitely many α-nodes. A word w is accepted by $\mathcal{A}$ if there a run that accepts it. The language of $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of infinite words that $\mathcal{A}$ accepts.

We sometimes refer to automata in which the acceptance condition is defined with respect to the transitions. Thus, such an automaton is a tuple $\mathcal{A} = \langle \Sigma, Q, q_{in}, \delta \rangle$, where the transition function is $\delta : Q \times \Sigma \to \mathcal{B}^+(Q \times \{\bot, \top\})$, and a run is accepting if all its paths contain infinitely many transitions with $\top$.

When the formulas in the transition function of $\mathcal{A}$ contain only conjunctions, then $\mathcal{A}$ is universal. When they contain only disjunctions, then $\mathcal{A}$ is nondeterministic, and its runs are DAGs of width 1, where at each level there is a single node. Accordingly, we sometimes refer to the transition function of a nondeterministic automaton as $\delta : Q \times \Sigma \to 2^Q$, and refer to its runs as sequences $r = q_0, q_1, \ldots$ of states. We extend δ to sets of states, by letting $\delta(S, a) = \bigcup_{q \in S} \delta(q, a)$, and recursively to words in Σ^*, by letting $\delta(S, \epsilon) = S$, and $\delta(S, w \cdot \sigma) = \delta(\delta(S, w), \sigma)$, for every $w \in \Sigma^*$ and $\sigma \in \Sigma$. As with words, we denote the subrun of r between positions t_1 and t_2 by $r[t_1, t_2]$. The set of states that a run or a subrun r visits is denoted by $states(r)$.

Finally, we denote the different classes of automata by three letter acronyms in $\{\mathrm{D}, \mathrm{N}, \mathrm{U}, \mathrm{A}\} \times \{\mathrm{B}, \mathrm{C}\} \times \{\mathrm{W}\}$. The first letter stands for the branching mode of the automaton (deterministic, nondeterministic, universal or alternating); the second letter stands for the acceptance-condition type (Büchi or co-Büchi); and the third letter indicates that the automaton runs on words. We add the prefix TR to denote automata with acceptance on transitions. For example, TR-UBW stands for a universal Büchi word automaton with acceptance on transitions.

3 The Lower Bound

In this section we show that the breakpoint construction is accurate, in the sense that it keeps the exact data required for translating an ABW to an NBW. Starting with an ABW with state space Q and acceptance set α (in fact, we even start with a UBW), the NBW generated by the breakpoint construction has a state for each pair $\langle S, O \rangle$, where $S \subseteq Q$ and $O \subseteq S \setminus \alpha$. We show that the construction is optimal, as an equivalent NBW must, essentially, have a different state corresponding to each pair $\langle S, O \rangle$. Our proof basically shows that the NBW must have a state corresponding to every two such pairs, while for simplicity reasons we ignore some cases, getting a constant factor. Formally, we prove the following.

Theorem 1. *There is a family of UBWs $\mathcal{U}_4, \mathcal{U}_5, \ldots$ over an alphabet of 8 letters, such that for every $n \geq 4$, the UBW $\mathcal{U}_n$ has n states, and every NBW equivalent to $\mathcal{U}_n$ has at least $\frac{1}{6}3^n$ states.*

In [24], Yan presents the "full automata approach", suggesting to seek for lower bounds on automata with unbounded alphabets, allowing every possible transition. Only then, should one try to implement the required rich transitions via finite words over a fixed alphabet. We adopt this approach, and further extend it. Not only do we assume an alphabet letter for every possible transition, but we also choose whether the transition visits the accepting states. For that reason, we start with TR-UBWs $\mathcal{A}_n$, having the acceptance condition on transitions rather than on states. Afterwards, we transform $\mathcal{A}_n$ to the required UBW $\mathcal{U}_n$, which is over a fixed alphabet and has acceptance on states.

The family of TR-UBWs. For every $n \geq 4$, we define the TR-UBW $\mathcal{A}_n = \langle \Gamma, Q, \delta, q_{in} \rangle$, where $Q = \{q_1, q_2, \ldots, q_n\}$, $q_{in} = q_1$, and $\Gamma = \{\mathit{reach}(S), \mathit{award}(S,O), \mathit{unify}(S)$ and $\mathit{connect}(S,O,O') : S \subseteq Q$ and $\emptyset \neq O, O' \subsetneq S\}$ is an alphabet consisting of four types of letters. The transition function $\delta : Q \times \Gamma \to 2^{Q \times \{\top, \bot\}}$ is defined as follows (see Figure 1):

– $\mathit{reach}(S)$: reaching a subset $S \subseteq Q$ from q_1, without a visit in an accepting transition. Formally,

$$\delta(q, \mathit{reach}(S)) = \begin{cases} S \times \{\bot\} & \text{if } q = q_1 \\ \emptyset & \text{otherwise.} \end{cases}$$

– $\mathit{award}(S,O)$: continuing the paths currently in S and awarding those in O with a visit in an accepting transition. Formally,

$$\delta(q, \mathit{award}(S,O)) = \begin{cases} \langle q, \top \rangle & \text{if } q \in O \\ \langle q, \bot \rangle & \text{if } q \in S \setminus O \\ \emptyset & \text{otherwise.} \end{cases}$$

We also refer to $\mathit{award}(S, \emptyset)$, defined in the same way.

– $\mathit{unify}(S)$: connecting, without a visit in an accepting transition, all states in S to all states in S. Formally,

$$\delta(q, \mathit{unify}(S)) = \begin{cases} S \times \{\bot\} & \text{if } q \in S \\ \emptyset & \text{otherwise.} \end{cases}$$

– $\mathit{connect}(S,O,O')$: connecting, without a visit in an accepting transition, all states in O to all states in O' and all states in $S \setminus O$ to all states in S. Formally,

$$\delta(q, \mathit{connect}(S,O,O')) = \begin{cases} O' \times \{\bot\} & \text{if } q \in O \\ S \times \{\bot\} & \text{if } q \in S \setminus O \\ \emptyset & \text{otherwise.} \end{cases}$$

Consider an NBW $\mathcal{B}_n$ with state space U and acceptance set β equivalent to $\mathcal{A}_n$. For showing the correspondence between the states of $\mathcal{B}_n$ and all possible pairs $\langle S, O \rangle$, we present a set of words in $L(\mathcal{A}_n)$ that will be shown to fully utilize the required state space of $\mathcal{B}_n$.

The words. For every $n \geq 4$, consider the TR-UBW $\mathcal{A}_n$ defined above. We say that a a triple $\langle S, O, O' \rangle \in 2^Q \times 2^Q \times 2^Q$ is *relevant* if $\emptyset \neq O, O' \subsetneq S$. For every relevant triple $\langle S, O, O' \rangle$, we define the infinite word $w_{S,O,O'} = \mathit{reach}(S) \cdot \mathit{reward}(S,O,O')^\omega$, where $\mathit{reward}(S,O,O') = \mathit{unify}(S) \cdot \mathit{award}(S, S \setminus O) \cdot \mathit{connect}(S,O,O') \cdot \mathit{award}(S,O')$.

Lemma 1. *For all relevant triples $\langle S, O, O' \rangle$, the word $w_{S,O,O'}$ is in $L(\mathcal{A}_n)$.*

Since the words are in $L(\mathcal{A}_n)$, each has an accepting run $r_{S,O,O'}$ of the equivalent NBW $\mathcal{B}_n$ on it. We first show that these runs are distinct for different Ss.

Lemma 2. *Let r_1 and r_2 be accepting runs of $\mathcal{B}_n$ on $w_1 = w_{S_1,O_1,O_1'}$ and $w_2 = w_{S_2,O_2,O_2'}$, respectively. If $S_1 \neq S_2$, then $\mathit{states}(r_1[1,\infty]) \cap \mathit{states}(r_2[1,\infty]) = \emptyset$.*

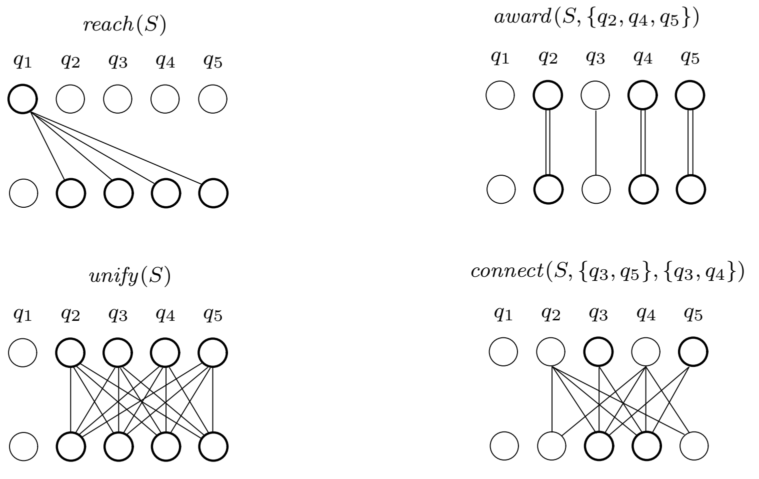

Fig. 1. An illustration of the required actions, for $S = \{q_2, q_3, q_4, q_5\}$. The doubled transitions are accepting.

Replacing a letter $\mathit{connect}(S, O, O')$ in the word $w_{S,O,O'}$ with a letter $\mathit{connect}(S, P, P')$ (of another tuple) may result in a word out of $L(\mathcal{A}_n)$. We say that a tuple $\langle S, P, P'\rangle$ is *humbler than* a tuple $\langle S, O, O'\rangle$ if the run of $\mathcal{A}_n$ on $\mathit{award}(S, S\backslash O)\cdot \mathit{connect}(S, P, P')\cdot \mathit{award}(S, O')$ visits an accepting transition along every path that starts in a state in S.

Lemma 3. *If $\langle S, P, P'\rangle$ is humbler than $\langle S, O, O'\rangle$ then $O \subseteq P$ and $P' \subseteq O'$.*

Let r be a specific accepting run $r_{S,O,O'}$ of $\mathcal{B}_n$ on $w_{S,O,O'}$. Since r goes infinitely often along the subword $\mathit{reward}(S, O, O')$, there is some state q visited infinitely often at the starting positions of the subword $\mathit{reward}(S, O, O')$. Since r is accepting, there are cases in which r visits β between two such visits of q. That is, there are positions t_1 and t_2 such that $r(t_1) = r(t_2) = q$ and $\mathit{states}(r[t_1, t_2]) \cap \beta \neq \emptyset$. We shall refer to the subrun of r between positions t_1 and t_2 as the loop $l_{S,O,O'}$. Such a loop contains at least one transition corresponding to $\mathit{connect}(S, O, O')$, going from some state u to some state v. We refer to u and v as a *bridge* for $\langle S, O, O'\rangle$.

Assigning designated bridges to relevant triples. A *bridge assignment* is a function $f : 2^Q \times 2^Q \times 2^Q \to U \times U$. We say that a bridge assignment f is *good* if for every relevant triple $\langle S, O, O'\rangle$, the bridge $\langle u, v\rangle = f(\langle S, O, O'\rangle)$ satisfies one of the following.

1. There is a transition from u to v on $\mathit{connect}(S, O, O')$ along $l_{S,O,O'}$, and for all relevant triples $\langle S, P, P'\rangle$, if there is a transition from u to v on $\mathit{connect}(S, P, P')$, then $\langle S, P, P'\rangle$ is humbler than $\langle S, O, O'\rangle$, or
2. (Intuitively, we cannot choose u and v that satisfy the condition above, in which case we choose a transition that visits an accepting state). For all pairs $\langle u', v'\rangle \in U \times U$, if there is a transition from u' to v' on $\mathit{connect}(S, O, O')$ along $l_{S,O,O'}$, then there is a tuple $\langle S, P, P'\rangle$ such that there is a transition from u' to v' on

$connect(S,P,P')$ and $\langle S,P,P'\rangle$ is not humbler than $\langle S,O,O'\rangle$, in which case there is a transition from u to v on $connect(S,O,O')$ along $l_{S,O,O'}$ that visits β. [4]

Consider a relevant tuple $\langle S,O,O'\rangle$. If we cannot assign to $\langle S,O,O'\rangle$ a pair $\langle u,v\rangle$ that satisfies Condition (1) above, then all transitions from u to v on $connect(S,O,O')$ along $l_{S,O,O'}$ are also transitions along loops that are not accepting. Since the loop $l_{S,O,O'}$ does visit β, one of these transitions should visit β, and f can assign it. Hence we have the following.

Lemma 4. *There is a good bridge assignment.*

Next, we show that every pair of states can serve as the assigned bridge of at most two relevant triples. Intuitively, since there are "many" relevant triples, this would imply that "many bridges are needed". Intuitively, it follows from the fact that, by Lemma 3, if $\langle S,P,P'\rangle$ is humbler than $\langle S,O,O'\rangle$ and $\langle S,O,O'\rangle$ is humbler than $\langle S,P,P'\rangle$, then $O=P$ and $O'=P'$.

Lemma 5. *For every good bridge assignment f and pair $\langle u,v\rangle \in U \times U$, we have $|f^{-1}(\{\langle u,v\rangle\})| \leq 2$.*

Fixed alphabet. The size of the alphabet $\varGamma$ of $\mathcal{A}_n$ is exponential in n. From now on, let us refer to the alphabet of $\mathcal{A}_n$ as $\varGamma_n$. The UBWs $\mathcal{U}_n$ we are after have an alphabet $\varSigma$ of 8 letters, and a single additional accepting state. Using the 8 letters it is possible to simulate each of the letters $\gamma \in \varGamma_n$ by a finite sequence of letters (whose length depends on γ) in $\varSigma$. In particular, the set of states visited when γ is simulated includes an accepting state iff the transition taken when γ is read is accepting.

Lemma 6. *There is a set $\varSigma$ of size 8 such that for every $n \geq 4$, there are functions $\tau : \varGamma_n \to \varSigma^*$ and $\rho : (Q \cup \{q_{acc}\}) \times \varSigma \to 2^{Q\cup\{q_{acc}\}}$ such that for all $q \in Q$ and $\gamma \in \varGamma_n$, if $\delta(q,\gamma) = \{\langle q_1,b_1\rangle,\ldots,\langle q_m,b_m\rangle\}$, then the following hold.*

- *$\rho(q,\tau(\gamma)) = \{q_1,\ldots,q_m\}$, and*
- *Let $\tau(\gamma) = \sigma_1,\ldots,\sigma_l$. For all $1 \leq i \leq m$, and sequences $r_0,\ldots,r_l$ such that $r_0 = q$, $r_{j+1} \in \rho(r_j,\sigma_{j+1})$ for all $1 \leq j < l$, and $r_l = s_i$, there is $0 \leq j \leq l$ such that $r_i = q_{acc}$ iff $b_i = \top$.*

We can now complete the proof of Theorem 1. For every $n \geq 4$, let $\mathcal{B}_n$ be an NBW over the alphabet $\varSigma$ equivalent to $\mathcal{A}_n$. We can partition an input word that simulate the words $w_{S,O,O'}$ to blocks, where each block corresponds to a letter in $\varGamma_n$. We refer to a state of $\mathcal{B}_n$ that appears after reading a block as a "big-state". For every $n \geq 4$, consider the UBW $\mathcal{U}_n$ with state space $Q' = \{q_1,q_2,\ldots,q_n,q_{acc}\}$ that simulates $\mathcal{A}_n$ as described in Lemma 6, and an equivalent NBW $\mathcal{B}_n$. For every subset $S \subseteq Q' \setminus \{q_{acc}\}$ and nonempty subsets $O,O' \subsetneq S$ there is the loop $l_{S,O,O'}$ of big-states in $\mathcal{B}_n$. By Lemma 2, the loops are distinct among the different S's with respect to their big-states. Let X_S be the set of big-states in all the loops corresponding to a specific S. We know that $\mathcal{B}_n$ has at least $\varSigma_{S\subseteq Q'\setminus\{q_{acc}\}}|X_S|$ states.

[4] Thus, $u \in \beta$ or $v \in \beta$; we still describe the condition in terms of the transition as it makes the transformation to an automaton with a fixed alphabet clearer.

Let f be a good bridge assignment. By Lemma 4, such an assignment f exists. Consider a specific subset $S \subseteq Q' \setminus \{q_{acc}\}$. By Lemma 5, every pair of states in X_S can be the assigned bridge of at most two relevant triples in $\{S\} \times (2^S \setminus \{S, \emptyset\}) \times (2^S \setminus \{S, \emptyset\})$. There are $(2^{|S|} - 2)^2$ such relevant triples. Thus, there are at least $(2^{|S|} - 2)^2/2$ pairs of states in X_S. Therefore, there are at least $\frac{2^{|S|}-2}{\sqrt{2}} \geq \frac{2^{|S|}}{2}$ states in X_S. [5] Hence, there are at least $\Sigma_{S \subseteq Q' \setminus \{q_{acc}\}} |X_S| = \Sigma_{S \subseteq Q' \setminus \{q_{acc}\}} \frac{2^{|S|}}{2} = \frac{1}{2} 3^n$ states in $\mathcal{B}_n$. Starting with a UBW with $n + 1$ states, we get a state blow-up of $\frac{1}{2} 3^{n-1} = \frac{1}{6} 3^n$.

Combined with the breakpoint construction, we have a tight bound for the translation of an ABW to an equivalent NBW. Applying the construction in [16] to UBW, one ends up with a DBW. Since we described the lower bound using UBWs, we also get a tight bound for alternation removal of UBW, and, dually, to determinization of nondeterministic co-Büchi automata. Formally, we have the following.

Theorem 2. *The tight bound for translating ABWs or UBWs to NBWs and for determinization of NCWs is* $\Theta(3^n)$.

4 Ordered Automata

In Section 3 we showed that, in general, a blow-up of $\Omega(3^n)$ cannot be avoided when translating an ABW to an NBW. In this section we introduce and explore a subclass of ABWs that can be translated to an equivalent NBW with a blow-up of only $n2^n$.

Definition 1. *An automaton* $\mathcal{A} = \langle \Sigma, Q, \delta, q_{in}, \alpha \rangle$ *is* ordered *if there exists a partial order* $\leq_{\mathcal{A}}$ *on* $Q \setminus \alpha$*, such that for every* $q, q' \in Q \setminus \alpha$ *and* $\sigma \in \Sigma$*, if* $q' \in \delta(q, \sigma)$*, then* $q' \leq_{\mathcal{A}} q$.

Note that, equivalently, $\mathcal{A}$ is ordered if the only cycles consisting solely of states not in α are self loops.

The order property is less restrictive than the very-weak condition of [10]. To demonstrate this extra strength, we describe below the ordered ABW for the property "whenever there is *request*, then *try* and *ack* alternate until *grant* is valid" over the alphabet $\Sigma = 2^{AP}$, where $AP = \{try, ack, req, grant\}$. Since the ABW has a single rejecting state, it is obviously ordered. Note that this property cannot be specified in LTL or in a very weak ABW. Note also how the ordered ABW uses universal branches in order to allow the *try-ack* cycle to be accepting. Indeed, fulfilling the eventuality is taken care by a different copy of the ABW.

The automata used in the lower-bound proof have the property that every two states not in α are reachable from each other without an intermediate visit to α. In a sense, this property is an antipode of the order property presented in Definition 1. We argue that violating the order property is what forces an equivalent NBW to associate its states with two subsets of states of the ABW. Indeed, as we show below, an ABW that has the order property can be translated to an equivalent NBW with an $n2^n$ blow-up. Still, even for ordered automata, the NBW needs to maintain information beyond the subset construction, thus the $n2^n$ translation is tight.

[5] This $\geq$ is not correct for a very small subset S, but since we accumulate over all the subsets, the total sum does satisfy it.

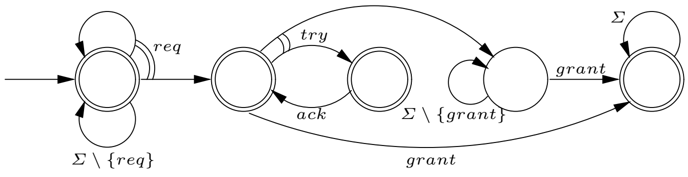

Fig. 2. An ordered ABW specifying "whenever there is *request*, then *try* and *ack* alternate until *grant* is valid". For a propositional assertion θ over AP, a transition labeled θ stands for a transition with all the letters $\sigma \in 2^{AP}$ that satisfy θ.

Theorem 3. *The tight bound for translating an ordered ABW to an NBW is* $\Theta(n2^n)$.

Proof. We start with the upper bound. Let $\mathcal{A} = \langle \Sigma, Q, \delta, q_{in}, \alpha \rangle$ be an ordered ABW, and let $\leq_{\mathcal{A}}$ be an extension of the partial order on $Q \setminus \alpha$ to a total order on Q. Let $|Q| = n$. The order $\leq_{\mathcal{A}}$ allows us to identify Q with $\{1, 2, \ldots, n\}$ while preserving the natural order. We define the equivalent NBW $\mathcal{A}' = \langle \Sigma, Q', \delta', q'_{in}, \alpha' \rangle$ as follows.

- $Q' \subseteq 2^Q \times (Q \setminus \alpha \cup \{0\})$ is such that $\langle S, p \rangle \in Q'$ iff $p \in (S \setminus \alpha) \cup \{0\}$. Intuitively, the set S follows the subset construction applied to $\mathcal{A}$: when $\mathcal{A}$ is in a state in $S \times \{0, \ldots, n\}$, the word in the input should be accepted from all the states in S. Note that since $\mathcal{A}$ is alternating, there may be several sets S' that are possible successors of a set S. Since the input word should be accepted from all the states in S, all the paths that start in states in S should not get trapped in a state not in α. To ensure this, $\mathcal{A}'$ examines the states not in α in a round-robin fashion: at each moment it examines a single state $p \in S \setminus \alpha$ and makes sure that no path in the run of $\mathcal{A}$ gets trapped in p: as long as p is a successor of itself, it keeps examining p. Only when a chain of p's ends (either because p is not in S' or because p is in S' but is not a successor of itself), $\mathcal{A}'$ changes the examined state, to the maximal state in S' that is smaller than p. If no such state exists, $\mathcal{A}'$ sets p to 0. As would be made clear below, this earns $\mathcal{A}'$ a visit in the set of accepting states, and causes it to start a new round of checks.
- $q'_{in} = \langle \{q_{in}\}, 0 \rangle$.
- In order to define the transition function, we first define a function $next : 2^Q \times 2^Q \times \{0, \ldots, n\} \times \Sigma \to \{0, \ldots, n\}$, which returns the next state that should be examined by $\mathcal{A}'$. Formally, $next(S, S', p, \sigma)$ is (we fix $\max(\emptyset) = 0$):

$$\begin{bmatrix} p & \text{if } p \neq 0 \text{ and } S' \setminus \{p\} \not\models \delta(p,\sigma) \\ \max(\{q \mid q \in S' \setminus (\alpha \cup \{p\}) \wedge q \leq p\}) & \text{if } p \neq 0 \text{ and } S' \setminus \{p\} \models \delta(p,\sigma) \\ \max(S' \setminus \alpha) & \text{if } p = 0 \end{bmatrix}$$

  Now, $\delta'(\langle S, p \rangle, \sigma) = \{\langle S', next(S, S', p, \sigma) \rangle \mid S' \models \delta(S, \sigma)\}$.
  Thus, each transition guesses the next set S' and updates the examined new state accordingly.
- $\alpha' = 2^Q \times \{0\}$.

We now turn to the lower bound. The lower bound of Theorem 1 does not hold for ordered UBWs as the UBWs $\mathcal{U}_n$ used there are, obviously, not ordered. In order to prove an $\Omega(n2^n)$ lower bound, we argue that the actions $reach()$ and $award()$ can be simulated by an ordered UBW over an alphabet whose size is linear in n, and that using them we can point to words that force the NBW to have at least $\Omega(n2^n)$ states.

For every $n \geq 4$, consider the TR-UBW $\mathcal{A}_n$ defined in Section 3. Using the actions $reach()$ and $award()$, one can define for every set $S \subseteq Q \setminus \{q_{acc}\}$ the word $w_s = reach(S) \cdot reward(S)^\omega$, where $reward(S) = \bullet_{q \in S}\ award(S, \{q\})$. These words belong to $L(\mathcal{A}_n)$, entailing for every S a distinct loop of states in an equivalent NBW. We show that the restriction of $\mathcal{A}_n$, having only the $reach()$ and $award()$ actions, can be simulated by an ordered UBW $\mathcal{O}_n$ over an alphabet whose size is linear in n, and that each such loop of big states in an NBW equivalent to $\mathcal{O}_n$ has at least $|S|$ states, providing the required lower bound of $\Sigma_{S \subseteq Q}|S| = \Omega(n2^n)$.

4.1 Fixed Alphabet

Usually, the alphabet size does not influence the state blow-up involved in automata translation. This is also the case with the translation of ABWs to NBWs, as shown in Section 3. Yet, ordered UBWs provide an interesting example of a case in which the alphabet size does matter. While Theorem 3 provides an $\Omega(n2^n)$ lower bound for the translation of an ordered UBW to an equivalent NBW, we show below that the translation can be done with only $O(2^n)$ state blow-up over a fixed alphabet.

Theorem 4. *An ordered UBW with n states over an alphabet with m letters has an equivalent DBW with 2^{m+n} states.*

Proof. Let $\mathcal{A} = \langle \Sigma, Q, \delta, q_{in}, \alpha \rangle$. We define $\mathcal{A}' = \langle \Sigma, Q', \delta', q'_{in}, \alpha' \rangle$, where

- $Q' = 2^Q \times 2^\Sigma$. Intuitively, the 2^Q component is a simple subset construction. The 2^Σ component has the task of maintaining a set of letters recently read from the input word, with the property that all suffixes consisting entirely of letters from this set are rejected by $\mathcal{A}$.
- For a state $\langle S, P \rangle$, we say that P *detains* S if there is a state $q \in S \setminus \alpha$ such that for every letter $\sigma \in P$, we have $q \in \delta(q, \sigma)$. Now, for all states $\langle S, P \rangle \in Q'$ and $\sigma \in \Sigma$, we define

$$\delta'(\langle S, P \rangle, \sigma) = \begin{bmatrix} \langle \delta(S, \sigma), P \cup \{\sigma\} \rangle & \text{if } P \cup \{\sigma\} \text{ detains } S. \\ \langle \delta(S, \sigma), \emptyset \rangle & \text{otherwise.} \end{bmatrix}$$

  That is, the 2^Q component follows the subset construction, while the current letter is added to the 2^Σ component as long as the required property (which is equivalent to $P \cup \{\sigma\}$ detaining S) is retained. So a path in a run of $\mathcal{A}$ gets trapped in some state q iff the 2^Σ component manages to avoid the empty set thanks to q.
- $q'_{in} = \langle \{q_{in}\}, \emptyset \rangle$.
- $\alpha' = 2^Q \times \{\emptyset\}$.

Remark 1. It is shown in [18] that the breakpoint construction is valid when applied to alternating Büchi tree automata. Our lower bound proof clearly holds also for alternation removal in tree automata. As for the upper bound, it is not hard to see that the definition of ordered automata can be extended to the setting of tree automata, and that both translations in Theorems 3 and 4 stay valid in this setting.

References

1. Accellera. Accellera organization inc. http://www.accellera.org, 2006.
2. N. Daniele, F. Guinchiglia, and M.Y. Vardi. Improved automata generation for linear temporal logic. In *Proc 11th CAV*, LNCS 1633, pages 249–260, 1999.
3. C. Eisner and D. Fisman. *A Practical Introduction to PSL*. Springer, 2006.
4. E.A. Emerson and C. Jutla. Tree automata, μ-calculus and determinacy. In *Proc. 32nd FOCS*, pages 368–377, 1991.
5. K. Etessami and G.J. Holzmann. Optimizing Büchi automata. In *Proc. 11th CONCUR*, LNCS 1877, pages 153–167, 2000.
6. K. Etessami, Th. Wilke, and R. A. Schuller. Fair simulation relations, parity games, and state space reduction for Büchi automata. *Siam J. Comput.*, 34(5):1159–1175, 2005.
7. E. Filiot, N. Jin, and J.-F. Raskin. An antichain algorithm for LTL realizability. In *Proc 21st CAV*, LNCS 5643, pages 263–277, 2009.
8. C. Fritz. Constructing Büchi automata from linear temporal logic using simulation relations for alternating Büchi automata. In *Proc. 8th IAA*, LNCS 2759, pages 35–48, 2003.
9. C. Fritz and T. Wilke. State space reductions for alternating Büchi automata: Quotienting by simulation equivalences. In *Proc. 22nd FST & TCS*, LNCS 2556, pages 157–169, 2002.
10. P. Gastin and D. Oddoux. Fast LTL to Büchi automata translation. In *Proc 13th CAV*, LNCS 2102, pages 53–65, 2001.
11. O. Kupferman and M.Y. Vardi. Model checking of safety properties. *Formal Methods in System Design*, 19(3):291–314, 2001.
12. O. Kupferman and M.Y. Vardi. Complementation constructions for nondeterministic automata on infinite words. In *Proc. 11th TACAS*, LNCS 3440, pages 206–221, 2005.
13. O. Kupferman, M.Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, 2000.
14. O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *Proc. 12th POPL*, pages 97–107, 1985.
15. C. Löding. Optimal bounds for the transformation of omega-automata. In *Proc. 19th FST & TCS*, LNCS 1738, pages 97–109, 1999.
16. S. Miyano and T. Hayashi. Alternating finite automata on ω-words. *Theoretical Computer Science*, 32:321–330, 1984.
17. A. Morgenstern and K. Schneider. From LTL to symbolically represented deterministic automata. In *Proc. 9th VMCAI*, LNCS 4905, pages 279–293, 2008.
18. A.W. Mostowski. Regular expressions for infinite trees and a standard form of automata. In *Computation Theory*, LNCS 208, pages 157–168, 1984.
19. D.E. Muller, A. Saoudi, and P.E. Schupp. Alternating automata, the weak monadic theory of the tree and its complexity. In *Proc. 13th ICALP*, LNCS 226, pages 275 – 283, 1986.
20. D.E. Muller and P.E. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54:267–276, 1987.
21. F. Somenzi and R. Bloem. Efficient Büchi automata from LTL formulae. In *Proc 12th CAV*, LNCS 1855, pages 248–263, 2000.
22. M.Y. Vardi. Nontraditional applications of automata theory. In *Proc. 11th TACAS*, LNCS 789, pages 575–597, 1994.
23. M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.
24. Q. Yan. Lower bounds for complementation of ω-automata via the full automata technique. In *Proc. 33rd ICALP*, LNCS 4052, pages 589–600, 2006.