# Promptness in $\omega$-regular Automata

Shaull Almagor[1], Yoram Hirshfeld[2], and Orna Kupferman[1]

[1] Hebrew University, School of Engineering and Computer Science, Jerusalem, Israel.
[2] Tel Aviv University, School of Mathematical Science, Tel Aviv, Israel.

**Abstract.** Liveness properties of on-going reactive systems assert that something good will happen eventually. In satisfying liveness properties, there is no bound on the "wait time", namely the time that may elapse until an eventuality is fulfilled. The traditional "unbounded" semantics of liveness properties nicely corresponds to the classical semantics of automata on infinite objects. Indeed, acceptance is defined with respect to the set of states the run visits infinitely often, with no bound on the number of transitions taken between successive visits.
In many applications, it is important to bound the wait time in liveness properties. Bounding the wait time by a constant is not always possible, as the bound may not be known in advance. It may also be very large, resulting in large specifications. Researchers have studied *prompt eventualities*, where the wait time is bounded, but the bound is not known in advance. We study the automata-theoretic counterpart of prompt eventually. In a *prompt-Büchi* automaton, a run $r$ is accepting if there exists a bound $k$ such that $r$ visits an accepting state every at most $k$ transitions. We study the expressive power of nondeterministic and deterministic prompt-Büchi automata, their properties, and decision problems for them. In particular, we show that regular nondeterministic prompt Büchi automata are exactly as expressive as nondeterministic co-Büchi automata.

## 1 Introduction

A specification of a reactive system describes the required on-going behaviors of the system. Specifications can be viewed as $\omega$-regular languages and are traditionally classified into *safety* and *liveness* properties [1]. Intuitively, safety properties assert that nothing bad will ever happen during the execution of the system, and liveness properties assert that something good will happen eventually. In satisfying liveness properties, there is no bound on the "wait time", namely the time that may elapse until an eventuality is fulfilled.

In many applications, it is important to bound the wait time in liveness properties. Bounding the wait time by a constant changes the specification from a liveness property to a safety property. For example, if we bound the wait time in the specification "every request is eventually granted" and replace it by the specification "every request is granted within $k$ transitions" for some fixed $k$, then we end up with a safety property – we never want to come across a request that is not granted within the next $k$ transitions. While the safety property

is much stronger, it involves two drawbacks. First, the bound $k$ needs to be known in advance, which is not the case in many applications. For example, it may depend on the system, which may not yet be known, or it may change, if the system changes. Second, the bound may be large, causing the state-based description of the specification (e.g., an automaton for it) to be large too. Thus, the common practice is to use liveness properties as an abstraction of such safety properties.

Several earlier work suggested and studied an alternative semantics to eventually properties. The semantics, termed *finitary fairness* in [3], *bounded fairness* in [9], and *prompt eventually* in [15], does not suffer from the above two drawbacks and is still more restrictive than the standard semantics. In the alternative semantics, the wait time is bounded, but the bound is not known in advance. Consider, for example, the computation $\pi = req.grant.req.\neg grant.grant.req.(\neg grant)^2.grant.\ req.(\neg grant)^3.grant\ldots$, in which the wait time to a grant increases in an unbounded (yet still finite) manner. While $\pi$ satisfies the liveness property "every request is eventually granted", it does not satisfy its prompt variant. Indeed, there is no bound $k$ such that $\pi$ satisfies the property "every request is granted within $k$ transitions".

The traditional "unbounded" semantics of liveness properties nicely corresponds to the classical semantics of automata on infinite objects. Indeed, acceptance is defined with respect to the set of states the run visits infinitely often, with no bound on the number of transitions taken between successive visits. The correspondence in the semantics is essential in the translation of temporal-logic formulas to automata, and in the automata-theoretic approach to *specification, verification*, and *synthesis* of nonterminating systems [17, 19]. The automata-theoretic approach views questions about systems and their specifications as questions about languages, and reduces them to automata-theoretic problems like containment and emptiness.

In this paper we introduce and study a prompt semantics for automata on infinite words, by means of *prompt-Büchi* automata. In a Büchi automaton, some of the states are designated as accepting states, and a run is accepting iff it visits states from the accepting set infinitely often [7]. Dually, in a *co-Büchi* automaton, a run is accepting iff it visits states outside the accepting set only finitely often. In a prompt-Büchi automaton, a run is accepting iff there exists a bound $k$ such that the number of transitions between successive visits to accepting states is at most $k$. More formally, if $\mathcal{A}$ is a prompt-Büchi automaton with a set $\alpha$ of accepting states, then an infinite run $r = r_1, r_2, ...$ of $\mathcal{A}$ is accepting iff there exists a bound $k \in \mathbb{N}$ such that for all $i \in \mathbb{N}$ it holds that $\{r_i, ..., r_{i+k-1}\} \cap \alpha \neq \emptyset$. We consider both nondeterministic (NPBWs, for short) and deterministic (DPBW, for short) prompt-Büchi automata.

It is not hard to see that if $\mathcal{A}$ is a Büchi automaton, then the automaton obtained by viewing $\mathcal{A}$ as a prompt Büchi automaton accepts the union of all the safety languages contained in the language of $\mathcal{A}$. As stated in [3], this union need not be $\omega$-regular. In fact, Büchi and prompt-Büchi automata are incomparable in terms of expressive power. Indeed, no NPBW can recognize the $\omega$-regular

language $(a^*b)^\omega$ (infinitely many occurrences of $b$), whereas no nondeterministic Büchi automaton (NBW, for short) can recognize the language $L_b$, where $w \in \{a,b\}^\omega$ is in $L_b$ if there exists a bound $k \in \mathbb{N}$ such that all the subwords of $w$ of length $k$ have at least one occurrence of $b$. Note that $L_b$ is recognized by a two-state DPBW that goes to an accepting state whenever $b$ is read. Note that when an NPBW runs on a word, it guesses and verifies the bound $k$ with which the run is going to be accepting. The bound $k$ may be bigger than the number of states, and still the NPBW has to somehow count and check whether an accepting state appears at least once every $k$ transitions. We would like to understand how this ability of NPBWs influences their expressive power and succinctness.

We start by investigating prompt Büchi automata in their full generality and show that while both NPBWs and DPBWs are closed under intersection and not closed under complementation, only NPBWs are closed under union. Also, NPBWs are strictly more expressive than DPBWs. We then focus on *regular*-NPBWs, namely languages that can be recognized by both an NPBW and an NBW. We first show that NPBWs are NBW-type: if the language of a given NPBW $\mathcal{A}$ is regular, then $\mathcal{A}$ is also an NBW for the language. From a theoretical point of view, our result implies that if a union of safety languages is $\omega$-regular, then the promptness requirement can be removed from every NPBW that recognizes it. From a practical point of view, our result implies that there is no state blow-up in translating NPBWs to NBWs, when possible. On the other hand, we show that there are NBWs that recognize languages that can be recognized by an NPBW, but an NPBW for them requires an automaton with a different structure. Thus, if we add the promptness requirement to an NBW, we may restrict its language, even if this language is a union of safety properties.

Our main result shows that regular-NPBWs are as expressive as nondeterministic co-Büchi automata (NCWs). To show this, we first prove that counting to unbounded bounds is not needed, and that the distance between successive visits in accepting states can be bounded by $3^{n^2}$, where $n$ is the number of states of the automaton. Technically, the bound follows from an analysis of equivalence classes on $\Sigma^*$ and an understanding that increasingly long subwords that skip visits in accepting states must contain equivalent prefixes. It is easy to show that the existence of the global $3^{n^2}$ bound implies that the language is NCW-recognizable. The global bound suggests a translation to NCW that is not optimal. We use results on the translation of NBWs to NCWs [6] in order to show an alternative translation, with a blow up of only $n2^n$. We also describe a matching lower bound. It follows that regular-NPBW are exponentially more succinct than NCWs. The equivalence with NCWs also gives immediate results about the closure properties of regular-NPBWs.

Finally, we study decision problems for prompt automata. We show that the problem of deciding whether a prompt automaton is regular is PSPACE-complete for NPBW and is NLOGSPACE-complete for DPBW. The same bounds hold for the universality and the containment problems. The main challenge in these results is the need to complement the NPBW. We show how we can circum-

vent the complementation, work, instead, with an automaton that approximates the complementing one, in the sense it accepts only words with a fixed bound, and still solve the regularity, universality, and containment problems.

**Related work** The work in [9, 15] studies the prompt semantics from a temporal-logic prospective. In [9], the authors study an eventuality operator parameterized by a bound (see also [2]), and the possibility of quantifying the bound existentially. In [15], the authors study the logic PROMPT-LTL, which extends LTL by a prompt-eventuality operator $\mathbf{F_p}$. A system $S$ satisfies a PROMPT-LTL formula $\psi$ if there is a bound $k \in \mathbb{N}$ such that $S$ satisfies the LTL formula obtained from $\psi$ by replacing all $\mathbf{F_p}$ by $\mathbf{F}^{\leq k}$. Thus, there should exist a bound, which may depend on the system, such that prompt eventualities are satisfied within this bounded wait time. It is shown in [15] that the realizability and the model-checking problems for PROMPT-LTL have the same complexity as the corresponding problems for LTL, though the algorithms for achieving the bounds are technically more complicated. Note that the definition of prompt Büchi automata corresponds to the semantics of the $\mathbf{F_p}$ operator of PROMPT-LTL. Thus, given an LTL formula $\psi$, we can apply to $\psi$ the standard translation of LTL to NBW [19], and end up with an NPBW for the PROMPT-LTL formula obtained from $\psi$ by replacing its eventualities by prompt ones.

The work in [8, 10] studies the prompt semantics in $\omega$-regular games. The games studied are finitary parity and finitary Streett games. It is shown in [8] that these games are determined and that the player whose objective is to generate a computation that satisfies the finitary parity or Streett condition has a memoryless strategy. In contrast, the second player may need infinite memory. In [10] it is shown that the problem of determining the winner in a finitary parity game can be solved in polynomial time.[3]

The closest to our work here is [4, 5], which introduced the notion of promptness to Monadic Second Order Logic (MSOL) and $\omega$-regular expressions. In [5], the authors introduced $\omega$BS-regular expressions, which extend $\omega$-regular expressions with two new operators $B$ and $S$ – variants of the Kleene star operator. The semantics of the new operators is that $(r^B)^\omega$, for a regular expression $r$, states that words in $L(r)$ occur globally with a bounded length, and $(r^S)^\omega$ states that words in $L(r)$ occur with a strictly increasing length. Thus, $\omega$BS-regular expressions are clearly more expressive than NPBWs. In [4], the author studies the properties of a prompt extension to MSOL. The contribution in [4, 5] is orthogonal to our contribution here. From a theoretical point of view, [5, 4] offer an excellent and exhaustive study of the logical aspects of promptness, in terms of closure properties and the decidability of the satisfiability problem for the formalisms and their fragments, where the goal is to give a robust formalism and then study its computational properties. Indeed, the algorithmic aspects of the studied formalisms are not appealing: the complexity of the decidability problem is much higher than that of NPBWs, and the model of automata that is needed

[3] The computations of games with a single player correspond to runs of a nondeterministic automaton. Games, however, do not refer to languages, and indeed the problems we study are very different from these studied in [8, 10].

in order to solve these problems is much more complex than NPBW (the automata are equipped with counters, and the translation of expressions to them is complicated). Our contribution, on the other hand, focuses on the prompt variant of the Büchi acceptance condition. As such, it does not attempt to present a robust promptness formalism but rather to study NPBWs in depth. As our results show, NPBWs are indeed much simpler than the robust formalisms, making them appealing also from a practical point of view.

Due to the lack of space, most proofs are described in the appendix.

## 2 Preliminaries

Given an alphabet $\Sigma$, a word over $\Sigma$ is a (possibly infinite) sequence $w = \sigma_1 \cdot \sigma_2 \cdot \sigma_3 \cdots$ of letters in $\Sigma$. For $x \in \Sigma^*$ and $y \in \Sigma^* \cup \Sigma^\omega$, we say that $x$ is a *prefix* of $y$, denoted $x \preceq y$, if there is $z \in \Sigma^* \cup \Sigma^\omega$ such that $y = x \cdot z$. If $z \neq \epsilon$ then $x$ is a *strict prefix* of $y$, denoted $x \prec y$. For an infinite word $w$ and indices $0 \leq k \leq l$, let $w[k..l] = \sigma_k \cdots \sigma_l$ be the infix of $w$ between positions $k$ and $l$.

An *automaton* is a tuple $\mathcal{A} = \langle \Sigma, Q, \delta, Q_0, \alpha \rangle$, where $\Sigma$ is the input alphabet, $Q$ is a finite set of states, $\delta : Q \to 2^Q$ is a transition function, $Q_0 \subseteq Q$ is a set of initial states, and $\alpha \subseteq Q$ is an acceptance condition. We define several acceptance conditions below. Intuitively, $\delta(q, \sigma)$ is the set of states that $\mathcal{A}$ may move into when it is in the state $q$ and it reads the letter $\sigma$. The automaton $\mathcal{A}$ may have several initial states and the transition function may specify many possible transitions for each state and letter, and hence we say that $\mathcal{A}$ is *nondeterministic*. In the case where $|Q_0| = 1$ and for every $q \in Q$ and $\sigma \in \Sigma$, we have that $|\delta(q, \sigma)| = 1$, we say that $\mathcal{A}$ is *deterministic*. The transition function extends to sets of states and to finite words in the expected way, thus for $x \in \Sigma^*$, the set $\delta(S, x)$ is the set of states that $\mathcal{A}$ may move into when it is in a state in $S$ and it reads the finite word $x$. Formally, $\delta(S, \epsilon) = S$ and $\delta(S, x \cdot \sigma) = \bigcup_{q \in \delta(S,x)} \delta(q, \sigma)$. We abbreviate $\delta(Q_0, x)$ by $\delta(x)$, thus $\delta(x)$ is the set of states that $\mathcal{A}$ may visit after reading $x$. A *run* $r = r_1, r_2, r_3, ...$ of $\mathcal{A}$ on an infinite word $w = \sigma_1 \cdot \sigma_2 \cdots \in \Sigma^\omega$ is an infinite sequence of states such that $r_1 \in Q_0$, and for every $i \geq 1$, we have that $r_{i+1} \in \delta(r_i, \sigma_i)$. Note that while a deterministic automaton has a single run on an input word, a nondeterministic automaton may have several runs on an input word. We sometimes refer to $r$ as a word in $Q^\omega$ or as a function from the set of prefixes of $w$ to the states of $\mathcal{A}$ (that is, for a prefix $x \preceq w$, we have $r(x) = r_{|x|+1}$). Acceptance is defined with respect to the set $inf(r)$ of states that the run $r$ visits infinitely often. Formally, $inf(r) = \{q \in Q \mid \text{for infinitely many } i \geq 1, \text{ we have } r_i = q\}$.

As $Q$ is finite, it is guaranteed that $inf(r) \neq \emptyset$. The run $r$ is *accepting* if it satisfies the acceptance condition $\alpha$. A run $r$ satisfies a *Büchi* acceptance condition $\alpha$ if $inf(r) \cap \alpha \neq \emptyset$. That is, $r$ visits $\alpha$ infinitely often. Dually, $r$ satisfies a *co-Büchi* acceptance condition $\alpha$ if $\inf(r) \subseteq \alpha$. That is, $r$ visits $\alpha$ almost always. Note that the latter is equivalent to $inf(r) \cap (Q \setminus \alpha) = \emptyset$

We now define a new acceptance condition, *prompt-Büchi*, as follows: A run $r = r_1, r_2, \ldots$ satisfies *prompt-Büchi* acceptance condition $\alpha$ if there is $k \geq 1$ such that for all $i \geq 1$ there is $j \in \{i, i+1, \ldots, i+k-1\}$ such that $r_j \in \alpha$. That

is, in each block of $k$ successive states, the run $r$ visits $\alpha$ at least once. We say that $r$ is *accepting with a bound* $k$. Observe that if a run satisfies a prompt-Büchi condition $\alpha$, then it also satisfies Büchi $\alpha$. The converse, however, does not hold, as a run can satisfy Büchi $\alpha$ but not prompt-Büchi $\alpha$.

It is easy to see that requiring the bound to apply eventually (instead of always) provides an equivalent definition. Thus, equivalently, a run $r$ satisfies prompt-Büchi acceptance condition $\alpha$ if there is $k \geq 1$ and $n_0 \in \mathbb{N}$ such that for all $i \geq n_0$ there is $j \in \{i, i+1, \ldots, i+k-1\}$ such that $r_j \in \alpha$. We say that $r$ is *accepting with eventual bound* $k$. Given a prompt-Büchi accepting run with bound $k$ and eventual bound $k'$, note that it always holds that $k' \leq k$, and that a strict inequality is possible.

An automaton *accepts* a word if it has an accepting run on it. The *language* of an automaton $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of words that $\mathcal{A}$ accepts. We also say that $\mathcal{A}$ *recognizes* the language $L(\mathcal{A})$. For two automata $\mathcal{A}$ and $\mathcal{A}'$ we say that $\mathcal{A}$ and $\mathcal{A}'$ are *equivalent* if $L(\mathcal{A}) = L(\mathcal{A}')$.

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

Given two classes $\gamma$ and $\eta$ we say that $\gamma$ is *more expressive* than $\eta$ if every $\eta$-recognizable language is also $\gamma$-recognizable. If $\gamma$ is not more expressive than $\eta$ and $\eta$ is not more expressive than $\gamma$, we say that $\gamma$ and $\eta$ are *incomparable*. Different classes of automata have different expressive power. In particular, NBWs recognize all $\omega$-regular languages, while NCWs are strictly less expressive [18].

## 3 Properties of Prompt Languages

In this section we study properties of the prompt-Büchi acceptance condition. As discussed in Section 1, the class of NPBW-recognizable languages is incomparable with that of $\omega$-regular languages. Similar results were shown in [3], in the context of finitary fairness in and safety languages.

**Theorem 1.** NPBWs *and* NBWs *are incomparable.*

The fact that NPBWs and NBWs are incomparable implies we cannot borrow the known closure properties of $\omega$-regular languages to the classes NPBW and DPBW. In Theorem 2 below, we study closure properties. As detailed in the proof, for the cases of NPBW union as well as NPBW and DPBW intersection, the known constructions for NBW and DBW are valid also for the prompt setting. To show non-closure, we define, the following language. Let $\Sigma = \{a, b\}$. For a letter $\sigma \in \Sigma$, let $L_\sigma = \{w \in \{a, b\}^\omega : \text{ there exists } k \in \mathbb{N} \text{ such that for all} i \in \mathbb{N} \text{ we have } \sigma \in \{w_i, w_{i+1}, ..., w_{i+k}\}\}$. Recall that the language $L_b$ is in NPBW but is not $\omega$-regular. It is easy to see that $L_\sigma$ can be recognized by a DPBW

with two states (that goes to an accepting state whenever $\sigma$ is read). We show, however, that the union of $L_a$ and $L_b$ cannot be recognized by a DPBW and that no NPBW exists for its complementation. These results also imply that NPBWs are strictly more expressive than DPBWs.

**Theorem 2.** *1.* NPBWs *are, but* DPBWs *are not closed under union.*
*2.* NPBWs *and* DPBWs *are closed under intersection.*
*3.* NPBWs *and* DPBWs *are not closed under complementation.*
*4.* NPBWs *are strictly more expressive than* DPBWs.

## 4 Regular NPBW

We have seen in Section 3 that NPBWs and NBWs are incomparable. In this section we study regular prompt languages, namely languages that are both NPBW and NBW recognizable. We use *reg*-NPBW and *reg*-DPBW to denote the classes NBW $\cap$ NPBW and NBW $\cap$ DPBW, respectively. For an automaton $\mathcal{A}$, we denote by $\mathcal{A}_B, \mathcal{A}_P$, and $\mathcal{A}_C$ the automaton $\mathcal{A}$ when referred to as an NBW, NPBW, and NCW, respectively.

### 4.1 Typeness

Given two types of automata $\gamma_1$ and $\gamma_2$, we say that a $\gamma_1$ is $\gamma_2$-type if for every automaton $\mathcal{A}$ in the class $\gamma_1$, if $\mathcal{A}$ has an equivalent automaton in the class $\gamma_2$, then there exists an equivalent automaton in the class $\gamma_2$ on the same structure as $\mathcal{A}$ (that is, only changing the acceptance condition). Typeness was studied in [12, 13], and is useful, as it implies that a translation between the two classes does not involve a blowup and is very simple. In this section we study typeness for NPBWs and NBWs.

**Theorem 3.** NPBW *are* NBW*-type: if $\mathcal{A}$ is an automaton such that $L(\mathcal{A}_P)$ is $\omega$-regular, then $L(\mathcal{A}_P) = L(\mathcal{A}_B)$.*

**Proof:** Since both $L(\mathcal{A}_P)$ and $L(\mathcal{A}_B)$ are $\omega$-regular, so is their difference, and thus there is an NBW $\mathcal{A}'$ such that $L(\mathcal{A}') = L(\mathcal{A}_B) \setminus L(\mathcal{A}_P)$. We prove that $L(\mathcal{A}') = \emptyset$. Assume by way of contradiction that $L(\mathcal{A}') \neq \emptyset$. Then $\mathcal{A}'$ accepts also a periodic word, thus there are $u, v \in \Sigma^*$ such that $u \cdot v^\omega \in L(\mathcal{A}_B) \setminus L(\mathcal{A}_P)$. Let $r$ be an accepting run of $\mathcal{A}_B$ on $u \cdot v^\omega$. Thus, $inf(r) \cap \alpha \neq \emptyset$. Hence, there are indices $i < j$ such that $r$ visits $\alpha$ between reading $u \cdot v^i$ and $u \cdot v^j$, and such that $r(uv^i) = r(uv^j)$. Then, however, we can pump $r$ to an accepting run of $\mathcal{A}_P$ on $u \cdot (v^i v^{i+1} \cdots v^{j-1})^\omega = u \cdot v^\omega$, contradicting the fact that $u \cdot v^\omega \notin L(\mathcal{A}_P)$. $\square$

**Theorem 4.** NBWs *are not* NPBW*-type: there exists an automaton $\mathcal{A}$ such that $L(\mathcal{A}_B)$ is* NPBW*-recognizable, but there is no* NPBW *$\mathcal{A}'$ with the same structure as $\mathcal{A}$ such that $L(\mathcal{A}') = L(\mathcal{A}_B)$.*

**Proof:** Consider the automaton $\mathcal{A}$ in Figure 1. Note that $L(\mathcal{A}_B) = \{a, b\}^\omega \setminus \{a^\omega, b^\omega\}$. It is easy to construct a four-state NPBW for $L(\mathcal{A}_B)$. On the other

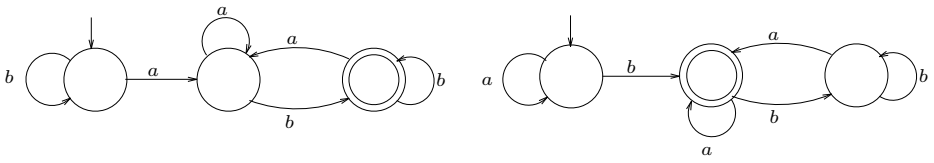


**Fig. 1.** An automaton $\mathcal{A}$ such that $L(\mathcal{A}_B)$ is NPBW recognizable, but no equivalent NPBW can be defined on top of its structure.

hand, it is not hard to see that all possibilities to define an acceptance condition on top of the structure of $\mathcal{A}$ results in an NPBW whose language is different. □

### 4.2 reg-NPBW=NCW

In this section we prove that reg-NPBWs are as expressive as NCWs. We show that while in the deterministic case, an equivalent DCW can always be defined on the same structure, the nondeterministic case is much more complicated and reg-NPBW are exponentially more succinct than NCWs.

We start with the deterministic case. As detailed in the proof, a DPBW $\mathcal{A}_P$ that recognizes a regular language can be translated to an equivalent DCW by making the set $\alpha$ of accepting states maximal. That is, by adding to $\alpha$ all states that do not increase the language of the DPBW. Intuitively, it follows from the fact that if a run of the obtained DCW does not get trapped in $\alpha$, and $\alpha$ is maximal, then there must exists a state $q \notin \alpha$ that is visited infinitely often along the run. The word that witnesses the fact that $q$ cannot be added to $\alpha$ can then be pumped to a word showing that $L(\mathcal{A}_P) \neq L(\mathcal{A}_B)$, contradicting the regularity of $\mathcal{A}_P$.

**Theorem 5.** *Let $\mathcal{A}$ be a reg-*DPBW*, then there exists a* DCW *$\mathcal{B}$ on the same structure as $\mathcal{A}$ such that $L(\mathcal{A}) = L(\mathcal{B})$.*

**Proof:** Consider the DPBW $\mathcal{A}_P$. For simplicity, we assume that $\alpha$ is maximal (that is, no states can be added to $\alpha$ without increasing the language. Clearly, we can turn a non-maximal accepting condition to an equivalent maximal one). We claim that $L(\mathcal{A}_C) = L(\mathcal{A}_P)$.

It is easy to see that $L(\mathcal{A}_C) \subseteq L(\mathcal{A}_P)$. Indeed, if $w$ is in $L(\mathcal{A}_C)$, then the run of $\mathcal{A}$ on $w$ eventually gets stuck in $\alpha$, and so $w \in L(\mathcal{A}_P)$. For the other direction, consider a word $w \in L(\mathcal{A}_P)$ and assume by way of contradiction that $w \notin L(\mathcal{A}_C)$. Let $r$ be the accepting run of $\mathcal{A}$ on $w$, and let $q \in \alpha$ and $q' \notin \alpha$ be states that are visited infinitely often in $r$. Since $r$ is prompt-accepting and not co-Büchi accepting, such $q$ and $q'$, which are reachable from each other, exist. Let $v \in \Sigma^*$ be a word leading from the initial state of $\mathcal{A}$ to $q'$ and let $u \in \Sigma^*$ be a word leading from $q'$ back to itself via $q$. Note that since $q \in \alpha$, then $v \cdot u^\omega$ is accepted by $\mathcal{A}$ (with a $\max\{|v|, |u|\}$ bound).

Recall that $\alpha$ is maximal. Thus, adding $q'$ to $\alpha$ would result in an automaton whose language strictly contains $L(\mathcal{A}_P)$. Hence, there exists $z \in \Sigma^*$, such that $z$ leads from $q'$ to itself and $v \cdot z^\omega$ is not in $L(\mathcal{A}_P)$. Thus, the run on $z$ from $q'$ is a cycle back to $q'$ that visits no states in $\alpha$.

For all $k \geq 1$, the word $w_k = v \cdot (z^k \cdot u)^\omega$ is accepted by $\mathcal{A}$. Indeed, the run $r$ of $\mathcal{A}$ on $w_k$ first gets to $q'$ after reading $v$. Then, whenever the run reads a $z^k u$ subword, it traverses a loop from $q'$ to itself $k$ times while reading $z^k$, and traverses a loop that visits $\alpha$ while reading u. Since the cycle traversed while reading $u$ occurs every $|z^k u|$ letters (that is, every fixed number of letters), this run is accepting.

Denote the number of states in $\mathcal{A}$ by $n$ and let $k > n$. In every $z$-block there are $l_1$ and $l_2$ such that $r$ visits the same state after it reads $z^{l_1}$ and $z^{l_2}$ in this block. Formally, for all $i \geq 0$ there are $0 \leq l_1 < l_2 \leq k$ such that $r(v \cdot (z^k \cdot u)^i \cdot z^{l_1}) = r(v \cdot (z^k \cdot u)^i \cdot z^{l_2})$. This means we can pump the $z$-blocks of $w_k$ to a word $w = v \cdot z^{i_1} \cdot u \cdot z^{i_2} \cdot u \cdot z^{i_3} \cdot u \cdots$, with $i_1 < i_2 < i_3 < \cdots$, such that $w \in L(\mathcal{A}_B)$. By Theorem 3, we know that $L(\mathcal{A}_B) = L(\mathcal{A}_P)$. Hence, also $w \in L(\mathcal{A}_P)$. Let $k$ be the bound with which $w$ is accepted by $\mathcal{A}_P$. Since there is $j \geq 1$ such that $i_j > \max\{k, n\}$, we can conclude, as in the proof of Theorem 1, that the word $w' = v \cdot z^{i_1} \cdot u \cdot z^{i_2} \cdots u \cdot z^{i_j} \cdot u \cdot z^\omega$ is accepted by $\mathcal{A}_P$. However, the run $r'$ of $\mathcal{A}$ on $w'$ has $r'(v \cdot z^{i_1} \cdot u \cdot z^{i_2} \cdots u \cdot z^{i_j} \cdot u) = q'$. Then, reading the $z^\omega$ suffix, the run $r'$ does not visit $\alpha$, implying that $\mathcal{A}_P$ does not accept $w'$, and we have reached a contradiction. □

We now proceed to study the (much harder) nondeterministic case. Consider an automaton $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$. For $u \in \Sigma^*$ and $q, q' \in Q$, we say that $q \to_u^+ q'$ if $q' \in \delta(q, u)$ and there is a run of $\mathcal{A}$ from $q$ to $q'$ that reads $u$ and visits $\alpha$. Similarly, we say that $q \to_u^- q'$ if $q' \in \delta(q, u)$ but all runs of $\mathcal{A}$ from $q$ to $q'$ that read $u$ do not visit $\alpha$.

We define a relation $\equiv_{\mathcal{A}} \subseteq \Sigma^* \times \Sigma^*$, where $u \equiv_{\mathcal{A}} v$ if for all $q, q' \in Q$, we have that $q \to_u^+ q'$ iff $q \to_v^+ q'$ and $q \to_u^- q'$ iff $q \to_v^- q'$. Intuitively, $u \equiv_{\mathcal{A}} v$ if for all states $q$ of $\mathcal{A}$, reading $u$ from $q$ has exactly the same effect (by means of reachable states and visits to $\alpha$) as reading $v$ from $q$.

It is easy to see that $\equiv_{\mathcal{A}}$ is reflexive, symmetric, and transitive. We can characterize each equivalence class $[u]$ of $\equiv_{\mathcal{A}}$ by a function $f_u : Q \to \{+, -, \bot\}^Q$, where for all $q, q' \in Q$, we have $f_u(q)(q') = +$ iff $q \to_u^+ q'$, $f_u(q)(q') = -$ iff $q \to_u^- q'$ and $f_u(q)(q') = \bot$ iff $q' \notin \delta(q, u)$. Since there are only $3^{|Q|^2}$ such functions, we have the following.

**Lemma 1.** *The relation $\equiv_{\mathcal{A}}$ is an equivalence relation with at most $3^{(|Q|^2)}$ equivalence classes.*

**Lemma 2.** *Consider an NPBW $\mathcal{A}$. Let $m$ denote the number of equivalence classes of $\equiv_{\mathcal{A}}$. Consider a word $w \in \Sigma^m$. There exist $s \in \Sigma^*$ and $z \in \Sigma^+$ such that $s \prec z \preceq w$, and $s \equiv_{\mathcal{A}} z$.*

**Proof:** Since $\equiv_{\mathcal{A}}$ has $m$ equivalence classes, every set of $m+1$ words must contain at least two $\equiv_{\mathcal{A}}$-equivalent words. In particular, this holds for the set $\{\epsilon, w[0..1], w[0..2], \ldots, w[0..m]\}$. □

**Theorem 6.** *Consider an automaton $\mathcal{A}$. Let $m$ denote the number of equivalence classes of $\equiv_{\mathcal{A}}$. If $L(\mathcal{A}_P)$ is $\omega$-regular, then every word in $L(\mathcal{A}_P)$ has an accepting run with an eventual bound of $2m$.*

**Proof:** Since $\mathcal{A}_P$ is $\omega$-regular, then, by Theorem 3, we have that $L(\mathcal{A}_P) = L(\mathcal{A}_B)$. Consider a word $w \in L(\mathcal{A}_P)$. We prove that there is a run that accepts $w$ with bound at most $2m$. Let $w = b_1 \cdot b_2 \cdot b_3 \cdots$ be a partition of $w$ to blocks of length $m$; thus, for all $i \geq 1$, we have that $b_i \in \Sigma^m$. We define $w'$ by replacing each block $b_i$ by a new block $b'_i$, of length strictly greater than $m$, defined as follows. For $i \geq 1$, let $h_i = b_1 \cdots b_{i-1}$ and $h'_i = b'_1 \cdots b'_{i-1}$. Thus, $h_i$ and $h'_i$ are the prefixes of $w$ and $w'$, respectively, that consist of the first $i-1$ blocks. Note that $h_1 = h'_1 = \epsilon$.

Assume we have already defined $h'_i$. We define $b'_i$ as follows: By Lemma 2, for every $i \geq 1$ there exist $s_i \in \Sigma^*$ and $z_i \in \Sigma^+$ such that $s_i \prec s_i \cdot z_i \preceq b_i$ and $s_i \equiv_{\mathcal{A}} s_i \cdot z_i$. Let $t_i$ in $\Sigma^*$ be such that $b_i = s_i \cdot z_i \cdot t_i$. Now, we define $b'_i = s_i \cdot (z_i)^i \cdot t_i$. Thus, $b'_i$ is obtained from $b_i$ by pumping an infix of it that lies between two prefixes that are $\equiv_{\mathcal{A}}$-equivalent.

For runs $r$ and $r'$ of $\mathcal{A}$ on $w$ and $w'$, respectively, we say that $r$ and $r'$ are matching if for all $i \geq 0$, the run $r$ visits $\alpha$ when it reads the block $b_i$ iff the run $r'$ visits $\alpha$ when it reads the block $b'_i$. We prove that every run $r$ of $\mathcal{A}$ on $w$ induces a matching run $r'$ of $\mathcal{A}$ on $w'$, and, dually, every run $r'$ of $\mathcal{A}$ on $w'$ induces a matching run $r$ of $\mathcal{A}$ on $w$.

Consider a run $r$ of $\mathcal{A}$ on $w$. For all $i \geq 1$, recall that $b_i = s_i \cdot z_i \cdot t_i$, where $s_i \equiv_{\mathcal{A}} s_i \cdot z_i$. It is easy to see (by induction on $j$) that for all $j \geq 1$, the latter implies that $s_i \cdot z_i \equiv_{\mathcal{A}} s_i \cdot (z_i)^j$. In particular, $s_i \cdot z_i \equiv_{\mathcal{A}} s_i \cdot (z_i)^i$. We define the behavior of $r'$ on $b'_i$ as follows. By the definition so far, $r'(h'_i) = r(h_i)$. We rely on the fact that $s_i \cdot z_i \equiv_{\mathcal{A}} h'_i s_i \cdot (z_i)^i$ and define $r'$ so that $r'(h'_i \cdot s_i \cdot (z_i)^i) = r(h_i \cdot s_i \cdot z_i)$. Also, $r'$ visits $\alpha$ when it reads $s_i \cdot (z_i)^i$ iff $r$ visits $\alpha$ when it reads $s_i \cdot z_i$. On $t_i$ we define $r'$ to be identical to $r$. It is easy to see that $r$ and $r'$ are matching. In a similar way, every run $r'$ of $\mathcal{A}$ on $w'$ induces a matching run $r$ of $\mathcal{A}$ on $w$.

Recall that $w \in L(\mathcal{A}_P)$. Therefore, there is a run $r$ of $\mathcal{A}$ on $w$ that visits $\alpha$ infinitely often, or, equivalently, visits $\alpha$ in infinitely many blocks. Hence, the matching run $r'$ of $\mathcal{A}$ on $w'$ also visits $\alpha$ in infinitely many blocks, and $w' \in L(\mathcal{A}_B) = L(\mathcal{A}_P)$.

Since $w \in L(\mathcal{A}_P)$, there is a run $r'$ on $w'$ that is accepting with some bound $k \geq 1$. For every $i > k$, the block $b'_i$ contains the infix $(z_i)^i$, for $z_i \neq \epsilon$, and is therefore of length at least $k$. Hence, the run $r'$ visits $\alpha$ when it reads the the block $b'_i$. Let $r$ be a run of $\mathcal{A}$ on $w$ that matches $r'$. Since $r$ and $r'$ are matching, the run $r$ visits $\alpha$ when it reads the block $b_i$, for all $i > k$. Since $|b_i| = m$, the longest $\alpha$-less window in $r$ after block $i$ is of length $2m-1$, thus $r$ is accepting with an eventual bound of $2m$. Hence, $w$ is accepted by a run that has a $2m$ eventual bound. □

Theorem 6, together with Lemma 1, induce a translation of reg-NPBW to NCW: the NCW can guess the location in which the eventual bound $k$ becomes valid and then stays in an accepting region as long as an accepting state of the NPBW is visited at least once every $k$ transition. Since an NCW can be translated to an NPBW with eventual bound 1, we can conclude with the following.

**Theorem 7.** *reg-*NPBWs *are as expressive as* NCWs.

The construction used in the proof of Theorem 7 involves a blow-up that depends on the number $3^{(|Q|^2)}$ of equivalence classes, and is thus super-exponential. We now show that while we can do better, an exponential blow-up can not be avoided.

**Theorem 8.** *The tight blow-up in the translation of reg-*NPBW *to* NCW *is* $n2^n$*, where* $n$ *is the number of states of the* NPBW*.*

**Proof:** Consider a reg-NPBW $\mathcal{A}$ with $n$ states. From Theorem 7, we know that $L(\mathcal{A}_P)$ is NCW-recognizable. From Theorem 3, we know that $L(\mathcal{A}_P) = L(\mathcal{A}_B)$. Thus, $\mathcal{A}_B$ is an NBW whose language is NCW recognizable. Hence, by [6], there exists an NCW $\mathcal{B}$ with at most $n2^n$ states equivalent to $\mathcal{A}_B$, and hence also to $\mathcal{A}_P$. Moreover, it can be shown that the family of languages with which the $n2^n$ lower bound was proven in [6] can be defined by means of reg-NPBWs, rather than NBWs, implying a matching lower bound. □

### 4.3 Properties of reg-NPBW and reg-DPBW

The fact that reg-NPBW =NCW immediately implies that closure properties known for NCWs can be applied to reg-NPBW. For reg-DPBW, we present the corresponding constructions. The fact that only reg-DPBW are closed under complementation also implies that reg-NPBW are more expressive than reg-DPBW.

**Theorem 9.** *1. reg-*NPBW *are closed under finite union and intersection, and are not closed under complementation.*
*2. reg-*DPBW *are closed under finite union, intersection and complementation.*
*3. reg-*NPBW *are strictly more expressive than reg-*DPBW*.*

## 5 Decision problems

In this section we study three basic decision problems for prompt automata: regularity (given $\mathcal{A}_P$, deciding whether $L(\mathcal{A}_P)$ is $\omega$-regular) universality (given $\mathcal{A}_P$, deciding whether $L(\mathcal{A}_P) = \Sigma^\omega$), and containment (given $\mathcal{A}_P$ and an NBW $\mathcal{A}$, deciding whether $L(\mathcal{A}) \subseteq L(\mathcal{A}_P)$). Note that the nonemptiness problem for an NPBW $\mathcal{A}_P$ can ignore the promptness and solve the nonemptiness of $\mathcal{A}_B$ instead. The other problems, however, require new techniques. The main challenge solving them has to do with the fact that we cannot adopt solutions from

the regular setting, as these involve complementation. Instead, we use *approximated determinization* of NPBW: determinization that restricts attention to words accepted with some fixed eventual bound. We show that we can approximately determinize NPBWs and that we can use the approximating automaton in order to solve the three problems.

We first show that in the deterministic setting, the three problems can be solved by analyzing the structure of the automaton. To see the idea behind the algorithms, consider a reachable state $q \in Q$ such that $q$ is reachable from itself both by a cycle that intersects $\alpha$ and by a cycle that does not intersect $\alpha$. The existence of such a state implies the existence of words $x, u, v \in \Sigma^*$ such that the word $x(uv^i)^\omega$ is accepted by $\mathcal{A}_P$ for all $i \in \mathbb{N}$, but $w = xuvuv^2uv^3 \cdots$ satisfies $w \in L(\mathcal{A}_B) \setminus L(\mathcal{A}_P)$. Thus, $\mathcal{A}_P$ is regular iff no such state exists, which can be checked in NLOGSPACE. As detailed in the proof, the algorithms for universality and containment follow similar arguments.

**Theorem 10.** *The regularity, universality, and containment problems for* DPBW *are NLOGSPACE-complete.*

We continue to the nondeterministic setting and start with the construction of the deterministic co-Büchi approximating automaton. The DCW $\mathcal{D}$ that approximates the NPBW $\mathcal{A}$ has a parameter $k$ and it accepts exactly all words that are accepted in $\mathcal{A}$ with eventual bound $k$. Essentially, $\mathcal{D}$ follows the subset construction of $\mathcal{A}$, while keeping, for each reachable state $q$, the minimal number of transitions along which $q$ is reachable from a state in $\alpha$ in some run of $\mathcal{A}$. If this number exceeds $k$, then $\mathcal{D}$ regards it as $\infty$, meaning that this state cannot be part of a run that has already reached the suffix in which the eventual bound applies. A run of $\mathcal{D}$ is accepting if eventually it visits only subsets that contain at least one state with a finite bound. Indeed, a run of $\mathcal{D}$ can get stuck in such states iff there is a run of $\mathcal{A}$ that visits $\alpha$ every at most $k$ transitions.

**Theorem 11.** *Let $\mathcal{A}$ be an* NPBW *with $n$ states. For each $k \in \mathbb{N}$ there exists a* DCW *$\mathcal{D}$ with at most $(k+1)^n$ states such that $L(\mathcal{D}) = \{w : w$ is accepted by $\mathcal{A}$ with eventual bound $k\}$.*

**Proof:** Let $\mathcal{A} = \langle \Sigma, Q, \delta, Q_0, \alpha \rangle$ and let $k \in \mathbb{N}$. We define $\mathcal{D} = \langle \Sigma, Q', \delta', Q'_0, \alpha' \rangle$ as follows. The state space $Q'$ is the set of partial functions $d : Q \to \{0, ..., k-1, \infty\}$. For a state $d \in Q'$, we say that a state $q \in Q$ is *$d$-safe* if $d(q) \in \{0, 1, ..., k-1\}$. Let $safe(d) = \{q : q$ is $d$-safe$\}$ be the set of all states that are $d$-safe. When $\mathcal{A}$ reads a word $w$, a state $d \in Q'$ keeps track of all the runs of $\mathcal{A}$ on $w$ in the following manner. For each state $q \in Q$, if $d(q)$ is undefined, then $q$ is not reachable in all runs of $\mathcal{A}$ on the prefix of the input word read so far. Otherwise, $d(q)$ is the minimal number of transition that some run of $\mathcal{A}$ on $w$ that visits $q$ has made since visiting $\alpha$. If this number is greater than $k$, that is, no run of $\mathcal{A}$ that visits $q$ has visited $\alpha$ in the past $k$ transitions, then $d(q) = \infty$. Intuitively, this means that runs that visit $q$ are still not in the suffix in which $\alpha$ is visited promptly. As we shall prove below, some run has reached a "good" suffix, iff $\mathcal{D}$ can get trapped in states that contain at least one safe state.

We now define $\mathcal{D}$ formally. The initial state is $Q'_0 = \{d_0\}$ such that for all $q \in Q_0$, we have $d_0(q) = 0$. The set of accepting states is $\alpha' = \{d : \mathit{safe}(d) \neq \emptyset\}$. Before we define the transition function, we define an addition operator on $\{0, 1, ..., k-1, \infty\}$ as follows. For all $i \in \{1, ..., k-1, \infty\}$, we define $i + 1 = i + 1$ if $i < k - 1$ and $i + 1 = \infty$ if $i \in \{k - 1, \infty\}$.

For technical convenience, we associate with a state $d$ the set $S_d$ of states on which $d$ is defined. We say that $d$ is *lost* if for all $q \in S_d$, we have $d(q) = \infty$. For $d \in Q'$ and $\sigma \in \Sigma$ we define $\delta'$ as follows. If $d$ is lost, then $\delta'(d, \sigma)(q') = 0$ for all $q' \in \delta(S_d, \sigma)$. Otherwise we define $\delta'(d, \sigma) = d'$ as follows. For $q' \in \delta(\mathit{safe}(d), \sigma) \cap \alpha$, we have $d'(q') = 0$. For $q' \in \delta(S_d, \sigma) \cap (Q \setminus \alpha)$ we have $d'(q') = \min\{d(q) + 1 : q' \in \delta(q, \sigma) \text{ and } q \in S_d\}$. Finally, for $q' \in (\delta(S_d, \sigma) \setminus \delta(\mathit{safe}(d), w_i)) \cap \alpha$ we have $d'(q') = \infty$.

Intuitively, $\mathit{safe}(d)$ is the set of states that are still within the $k$ bound in some run of $\mathcal{A}$. Thus, $d'$ checks which of these states indeed visit $\alpha$ after reading $\sigma$, and resets them to 0. Otherwise it increases the counter. If all states reached $\infty$, then the eventual bound $k$ was not yet in effect. We "take note" of this by not visiting $\alpha$, and then $d'$ resets to 0, giving the eventual bound a new chance. One may notice that the sets $S_d$ are exactly the subset construction of $\mathcal{A}$. Thus, essentially, $d$ is a labeling function of the subset construction.

The correctness of $\mathcal{D}$ is proven in the appendix. □

We refer to $\mathcal{D}$ as the *k-approximating DCW of* $\mathcal{A}$. While the deterministic DCW constructed in Theorem 11 accepts only words that are accepted in the original NPBW with a fixed eventual bound, Theorem 6 enables us to use such a DCW in the process of deciding properties of the NPBW. In particular, for a regular-NPBW $\mathcal{A}$, Theorem 6 implies that a DCW constructed with bound $2 \cdot 3^{n^2}$ is equivalent to $\mathcal{A}$. Recall that NPBWs are NBW-type. By [6], an NBW whose language is DCW-recognizable can be translated to a DCW with $3^n$ states. Hence we have the following.

**Theorem 12.** *Let $\mathcal{A}$ be a regular* NPBW*. There exists a* DCW *with at most $3^n$ states such that $L(\mathcal{D}) = L(\mathcal{A})$.*

In fact, as we show below, the deterministic approximating automaton is helpful for deciding all three problems, even when applied to non-regular NPBWs.

**Theorem 13.** *The regularity, universality, and containment problems for* NPBWs *are PSPACE-complete.*

**Proof:** We prove here the upper bounds. For the lower bounds (detailed in the appendix), we describe a reduction from NFW universality to NPBW universality, and a reduction from NPBW universality to NPBW regularity. Since universality can be reduced to containment, PSPACE-hardness for all the three problems follow.

We first prove that the universality problem is in PSPACE. Let $\mathcal{A}$ be an NPBW. Let $\mathcal{D}$ be the DCW defined in Theorem 11 with bound $2 \cdot 3^{n^2}$. We claim that $L(\mathcal{A}_P) = \Sigma^\omega$ iff $L(\mathcal{D}) = \Sigma^\omega$. For the first direction, if $L(\mathcal{A}_P) = \Sigma^\omega$, then in

particular $L(\mathcal{A}_P)$ is in reg-NPBW. From Theorem 12 we get that $L(\mathcal{D}) = L(\mathcal{A}_P)$, so $L(\mathcal{D}) = \Sigma^\omega$. For the other direction, observe that it is always true that $L(\mathcal{D}) \subseteq L(\mathcal{A}_P)$. Thus, if $L(\mathcal{D}) = \Sigma^\omega$, then $L(\mathcal{A}_P) = \Sigma^\omega$.

Finally, observe that we can check the universality of $\mathcal{D}$ in PSPACE. This is because the size of $\mathcal{D}$ is $(2 \cdot 3^{n^2} + 1)^n$, its constructing can proceed on-the-fly, and the universality problem for DCWs is in NLOGSPACE.

We proceed to prove that the regularity problem is in PSPACE. Consider the automaton $\mathcal{D}$ constructed in Theorem 12. If $L(\mathcal{A}_P) = L(\mathcal{A}_B)$ then $L(\mathcal{A}_P)$ is $\omega$-regular and thus $L(\mathcal{D}) = L(\mathcal{A}_P) = L(\mathcal{A}_B)$. On the other hand, if $L(\mathcal{D}) = L(\mathcal{A}_P)$ then $L(\mathcal{A}_P)$ is $\omega$-regular and by Theorem 3 we have that $L(\mathcal{A}_P) = L(\mathcal{A}_B)$. Thus, deciding whether $L(\mathcal{A}_P) = L(\mathcal{A}_B)$ is equivalent to deciding whether $L(\mathcal{D}) = L(\mathcal{A}_B)$. Note, however, that by Theorem 11 it is always true that $L(\mathcal{D}) \subseteq L(\mathcal{A}_P) \subseteq L(\mathcal{A}_B)$. Thus, it is sufficient to check whether $L(A_B) \subseteq L(\mathcal{D})$. The latter can be done in PSPACE, by simulating the complement of $\mathcal{D}$ on the fly, similarly to the proof of NPBW universality.

It is left to prove that the containment problem is in PSPACE. We describe a PSPACE algorithm for deciding whether $L(\mathcal{B}) \subseteq L(\mathcal{A}_P)$. Let $m$ and $n$ be the number of states in $\mathcal{B}$ and $\mathcal{A}$, respectively.First, check whether $L(\mathcal{B}) \subseteq L(\mathcal{A}_B)$. If $L(\mathcal{B}) \not\subseteq L(\mathcal{A}_B)$ return no. Otherwise, construct, per Theorem 11, a $2k$-approximating DCW, with $k = (m+1) \cdot 3^{n^2}$. Next, check whether $L(\mathcal{B}) \subseteq L(\mathcal{D})$. If so, return yes. Otherwise, return no.

It is not hard to see that the algorithm can be implemented in PSPACE. We now prove its correctness. Since it is always true that $L(\mathcal{D}) \subseteq L(\mathcal{A}_P) \subseteq L(\mathcal{A}_B)$, then it is easy to see that if $L(\mathcal{B}) \subseteq L(\mathcal{D})$ then $L(\mathcal{B}) \subseteq L(\mathcal{A}_P)$, and if $L(\mathcal{B}) \not\subseteq L(\mathcal{A}_B)$ then $L(\mathcal{B}) \not\subseteq L(\mathcal{A}_P)$. It remains to show that if $L(\mathcal{B}) \subseteq L(\mathcal{A}_B)$ but $L(\mathcal{B}) \not\subseteq L(\mathcal{D})$ then $L(\mathcal{B}) \not\subseteq L(\mathcal{A}_P)$.

Recall that $L(\mathcal{D}) = \{w : w \text{ is accepted by } \mathcal{A}_P \text{ with eventual bound of at most } 2k\}$. We claim that if $L(\mathcal{B}) \subseteq L(\mathcal{A}_P)$ then $L(\mathcal{B}) \subseteq L(\mathcal{D})$. Thus, we show that if $w \in L(\mathcal{B})$ then $w$ is accepted by $\mathcal{A}_P$ with eventual bound of at most $2k$. The proof of this claim follows the reasoning in the proof of Theorem 6.

Let $w \in L(\mathcal{B})$. Since $L(\mathcal{B}) \subseteq L(\mathcal{A}_P)$ then $w \in L(\mathcal{A}_P)$. Assume by way of contradiction that $w$ is accepted by $\mathcal{A}_P$ with an eventual bound greater than $2k$. We divide $w$ into blocks of length $k$, so $w = b_1 \cdot b_2 \cdots$ such that $|b_i| = k$. Let $s$ and $r$ be accepting runs of $\mathcal{B}$ and $\mathcal{A}_P$ on $w$, respectively. Thus, there are infinitely many blocks $b_i$ such that $s$ visits $\alpha_\mathcal{B}$ when reading $b_i$. Since $r$ has an eventual bound greater than $2k$, then there are infinitely many blocks $b_i$ such that $r$ does not visit $\alpha_\mathcal{A}$ when reading $b_i$. Since we took $k = (m+1) \cdot 3^{n^2}$, then in every block $b_i = x_1 \cdots x_k$ there exist two indices $j_1 < j_2$ such that $x_1 \cdots x_{j_1} \equiv_\mathcal{A} x_1 \cdots x_{j_2}$ and $s(x_{j_1}) = s(x_{j_2})$. We can now pump the infix $x_{j_1} \cdots x_{j_2}$ such that both $r$ and $s$ are not affected outside the pumped infix. Now, similarly to the proof of Theorem 6, we can pump infinitely many blocks $b_i$ such that the run $r$ is no longer prompt-accepting and $s$ is still accepting. We end up with a word $w'$ that is accepted by $\mathcal{B}$. By our assumption, $w'$ is accepted by $\mathcal{A}_P$, and thus has a prompt accepting run $r'$ of $\mathcal{A}_P$. We can shrink the pumped blocks of $w'$ back to

$w$ such that the respective shrinking of $s'$ is a prompt accepting run of $w$ with eventual bound of at most $2k$, which leads to a contradiction. □

In [14], the authors describe a PSPACE LTL model-checking algorithm for Prompt-LTL. Our PSPACE containment algorithm completes the picture and implies that all prompt properties given by an NPBW can be model-checked in PSPACE.

## References


1. B. Alpern and F.B. Schneider. Defining liveness. *IPL*, 21:181–185, 1985.
2. R. Alur, K. Etessami, S. La Torre, and D. Peled. Parametric temporal logic for model measuring. *ACM Transactions on Computational Logic*, 2(3):388–407, 2001.
3. R. Alur and T.A. Henzinger. Finitary fairness. In *Proc. 9th IEEE Symp. on Logic in Computer Science*, pages 52–61, 1994.
4. M. Bojanczyk. A bounding quantifier. In *Proc. 13th Annual Conf. of the European Association for Computer Science Logic*, pages 41–55, 2004.
5. M. Bojańczyk and T. Colcombet. Bounds in $\omega$-regularity. In *Proc. 21st IEEE Symp. on Logic in Computer Science*, pages 285–296, 2006.
6. U. Boker and O. Kupferman. Co-ing Büchi made tight and helpful. In *Proc. 24th IEEE Symp. on Logic in Computer Science*, pages 245–254, 2009.
7. J.R. Büchi. On a decision method in restricted second order arithmetic. In *Proc. Int. Congress on Logic, Method, and Philosophy of Science*, pages 1–12, 1962.
8. K. Chatterjee and T.A. Henzinger. Finitary winning in $\omega$-regular games. In *Proc. 12th TACAS*, LNCS 3920, pages 257–271, 2006.
9. N. Derahowitz, D.N. Jayasimha, and S. Park. Bounded fairness. In *Verification: Theory and Practice*, LNCS 2772, pages 304–317, 2003.
10. F. Horn. Faster algorithms for finitary games. In *Proc. 13th TACAS*, LNCS 4424, pages 472–484, 2007.
11. N.D. Jones. Space-bounded reducibility among combinatorial problems. *Journal of Computer and Systems Science*, 11:68–75, 1975.
12. S.C. Krishnan, A. Puri, and R.K. Brayton. Deterministic $\omega$-automata vis-a-vis deterministic Büchi automata. In *Algorithms and Computations*, LNCS 834, pages 378–386, 1994.
13. O. Kupferman, G. Morgenstern, and A. Murano. Typeness for $\omega$-regular automata. *IJFCS*, 17(4):869–884, 2006.
14. O. Kupferman, N. Piterman, and M.Y. Vardi. Safraless compositional synthesis. In *Proc 18th CAV*, LNCS 4144, pages 31–44, 2006.
15. O. Kupferman, N. Piterman, and M.Y. Vardi. From liveness to promptness. In *Proc 19th CAV*, LNCS 4590, pages 406–419, 2007.
16. O. Kupferman and M.Y. Vardi. Verification of fair transition systems. In *Proc 8th CAV*, LNCS 1102, pages 372–382, 1996.
17. R.P. Kurshan. *Computer Aided Verification of Coordinating Processes*. Princeton Univ. Press, 1994.
18. L.H. Landweber. Decision problems for $\omega$–automata. *Mathematical Systems Theory*, 3:376–384, 1969.
19. M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.

# A Proofs

## A.1 Proof of Theorem 1

To show that NPBW are not more expressive than NBW, consider the NBW-recognizable language $(a^*b)^\omega$ ("infinitely many $b$'s). We claim that there is no NPBW for it. Assume by way of contradiction that $\mathcal{A}$ is an NPBW recognizing $(a^*b)^\omega$. Consider the word $w = baba^2ba^3ba^4ba^5\cdots$. Since $w$ has infinitely many $b$'s, it is accepted by $\mathcal{A}$. Let $n$ be the number of states of $\mathcal{A}$ and let $k \in \mathbb{N}$ be such that there is an accepting run $r$ with bound $k$ of $\mathcal{A}$ on $w$. Let $j = n \cdot k$. There must be $i_1, i_2$, and $i_3$ such that $0 \leq i_1 \leq i_2 \leq i_3 \leq j$, $i_1 < i_3$, $r(baba^2 \cdots ba^j ba^{i_1}) = r(baba^2 \cdots ba^j ba^{i_3})$, and $r(baba^2 \cdots ba^j ba^{i_2}) \in \alpha$. Then, however, the run $r$ can be pumped to an accepting run on $baba^2 \cdots ba^k ba^\omega$, which has only finitely many $b$'s.

For the opposite direction, let $L_b = \{w : \text{there is } k \geq 1 \text{ such that all subwords of } w \text{ of length } k \text{ have at least one occurrence of } b\}$. It is easy to see that the two-state DPBW that goes to an accepting state whenever $b$ is read and goes to a non-accepting state whenever $a$ is read recognizes $L_b$. We claim that there is no NBW for $L_b$.

Assume by way of contradiction that There is an NBW $\mathcal{A}'$, with $n$ states, that accepts $L_2$. Consider the word $w = (b \cdot a^{n+1})^\omega$. An accepting run of $\mathcal{A}'$ on $w$ can be pumped to an accepting run on a word $b \cdot a^{i_1} \cdot b \cdot a^{i_2} \cdot b \cdot a^{i_3} \cdot b \cdots$, with $i_1 < i_2 < i_3 < \cdots$. Such a word, however, is not in $L_b$.

## A.2 Proof of Theorem 2

Throughout the proof we are going to refer to the following languages over the alphabet $\Sigma = \{a, b\}$. For $\sigma \in \{a, b\}$, let $L_\sigma = \{w \in \{a, b\}^\omega : \text{ there exists } k \in \mathbb{N} \text{ such that for all} i \in \mathbb{N} \text{ we have } \sigma \in \{w_i, w_{i+1}, ..., w_{i+k}\}\}$. Note that $L_b$ is the language used in the proof of Theorem 1. Thus, there is a DPBW for it and, dually, also for the language $L_a$.

**Union** We start with the nondeterministic case. It is easy to see that the standard union construction works. That is, given NPBWs $\mathcal{A}_1$ and $\mathcal{A}_2$ we can obtain an NPBW for their union by defining the state space, transition function, set of initial states and set of accepting states to be the union of the corresponding components in $A_1$ and $A_2$. Thus, NPBW are closed under finite union.

For the deterministic case, we claim that the language $L = L_a \cup L_b$ is not DPBW-recognizable. Assume by way of contradiction that there is a DPBW $\mathcal{A}$ for $L$. Let $n$ denote the number of states of $\mathcal{A}$. First notice that for a word $x \in \{a, b\}^*$, the words $x \cdot a^\omega$ and $x \cdot b^\omega$ are in $L$.

Consider a reachable state $q \in Q$ and let $y \in \Sigma^*$ such that $\delta(y) = q$. Since $\mathcal{A}$ is deterministic, then for any index $m > n$, the run of $\mathcal{A}$ on $a^m$ from $q$ must contain a cycle with an accepting state. Indeed, otherwise the run of $\mathcal{A}$ on $y \cdot a^\omega$ is not accepting. Since the length of such a cycle is at most $n$, and since this is

true for all $q \in Q$, then for every word $w \in L$ and for every index $l > n$, there exists an index $l' \geq l$ such that we can pump every infix $a^m$ of $w$ to $a^{l'}$, such that the run of $\mathcal{A}$ on the pumped infix visits an accepting state every $n$ states. Obviously this holds for $b$ blocks as well.

Consider the word $w = (a^{n+1}b^{n+1})^\omega$. Clearly $w \in L$ and therefore we can pump each $a$-block and $b$-block in an increasing sequence, yielding an accepting run on the word $\bar{w} = a^{i_0}b^{i_1}a^{i_2}b^{i_3}\cdots$ such that $n < i_0 < i_1 < i_2 < ....$ This is a contradiction since $\bar{w} \notin L$.

**Intersection** We claim that the intersection construction for Büchi automata is valid also in the case of prompt-Büchi. Given NPBW $\mathcal{A}_1 = \langle \Sigma, Q_1, \delta_1, Q_1^0, \alpha_1 \rangle$ and $\mathcal{A}_2 = \langle \Sigma, Q_2, \delta_2, Q_2^0, \alpha_2 \rangle$ Define $\mathcal{A} = \langle \Sigma, Q, \delta, Q^0, \alpha \rangle$ where $Q = Q_1 \times Q_2 \times \{1, 2\}$, that is, two copies of the product automaton. The transition function is defined by

$$\delta(\langle q_1, q_2, i \rangle, \sigma) = \begin{cases} \langle \delta_1(q_1, \sigma), \delta_2(q_2, \sigma), i \rangle & q_i \notin \alpha_i \\ \langle \delta_1(q_1, \sigma), \delta_2(q_2, \sigma), 3 - i \rangle & q_i \in \alpha_i \end{cases}$$

The set of accepting states is $\alpha = \alpha_1 \times Q_2 \times \{1\}$ and the initial states are $Q^0 = Q_1^0 \times Q_2 \times \{1\}$.

We prove that $L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$. Consider a word $w \in L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$. Then, there exist accepting runs $r_1$ and $r_2$ of $\mathcal{A}_1$ and $\mathcal{A}_2$ on $w$, respectively. Denote the bounds of the runs by $k_1$ and $k_2$. Let $r$ be the run of $\mathcal{A}$ on $w$ induced by the projections of $r_1$ and $r_2$ on the product automaton (the third component of the run is determined by the projections of the runs, according to $\delta$). Observe that by the definition of $\delta$ this is a legal run. Further observe that when in copy 1 of the product automaton, $\mathcal{A}$ visits an accepting state in every interval of length $k_1$, and when in copy 2 $\mathcal{A}$ visits $Q_1 \times \alpha_2 \times 2$ in every interval of length $k_2$. This implies that an accepting state of $\mathcal{A}$ is visited at least once in every interval of length $k_1 + k_2$, which means $\mathcal{A}$ accepts $w$ with bound $k_1 + k_2$.

For the other direction, consider a word $w \in L(\mathcal{A})$. Let $r$ be an accepting run of $\mathcal{A}$ on $w$ with bound $k$. By definition, $r$ visits $\alpha_1 \times Q_2 \times \{1\}$ in interval of length $k$. By the definition of $\delta$, this means that in every such interval, $r$ goes to the second copy and returns, which in turn means that $r$ visits $Q_1 \times \alpha_2 \times \{2\}$ in every interval of length $k$. Thus, the projection of $r$ on the first two components induces accepting runs of $\mathcal{A}_1$ and $\mathcal{A}_2$ on $w$ with bound $k$, and we are done.

Further note that this construction preserves determinism, so NPBW and DPBW are closed under intersection.

**complementation** Consider again the DPBW-recognizable language $L_b$. Let $L = \Sigma^\omega \setminus L_b$. Thus, $L = \{w :$ for all $k \geq 1$, the word $a^k$ is a subword of $w\}$. We claim that there is no NPBW for $L$. Assume by way of contradiction that there is an NPBW with $n$ states for $L$. A simple pumping argument on the word $w = aba^2ba^3ba^4b\cdots$ enables us to find two indices $i_1 < i_2 < ... < i_l$ such that

the word $w' = aba^2ba^3b \cdots a^{k-1}b(a^{i_1}ba^{i_2}b \cdots a^{i_l}b)^\omega$ is also accepted, which is a contradiction, since there is a bound on the length of subwords of the form $a^*$ in $w'$.

**Determinization** This is an easy corollary of the fact that DPBWs are not closed under union whereas NPBWs are. To see this, consider two DPBW-recognizable languages $L_1$ and $L_2$ such that $L_1 \cup L_2$ is not DPBW-recognizable. Since $L_1 \cup L_2$ is NPBW-recognizable, we are done. In particular, this holds for our languages $L_a$ and $L_b$.

### A.3 Proof of Theorem 7

For the first direction, we prove that every NCW-recognizable language is also in reg-NPBW. It is easy to see that the standard construction that translates an NCW to an NBW yields an automaton such that every accepting run gets stuck in $\alpha$. The language of such an automaton is clearly in reg-NPBW.

For the other direction, let $L \in$ reg-NPBW and let $\mathcal{A}$ be an automaton such that $L = L(\mathcal{A}_P)$. By Theorem 3, such an automaton exists. Furthermore, by Theorem 6 we know that there exists $k \in \mathbb{N}$ such that every word $w \in L(\mathcal{A}_P)$ is accepted by a run with an eventual bound of at most $k$.

We define an NCW $\mathcal{A}' = \langle \Sigma, Q', Q'_0, \delta', \alpha' \rangle$ such that $L(\mathcal{A}_P) = L(\mathcal{A}')$. Intuitively, we take $k$ copies of $\mathcal{A}$ that are all accepting, and another copy that "waits". A run advances through these copies and resets to the first accepting copy when getting to $\alpha$. If more than $k$ steps are made without getting to $\alpha$ the run goes into a rejecting state. We now define this formally:

$$Q' = Q \times \{0, 1, ..., k\} \cup \{q_{rej}\}$$

$$Q'_0 = Q_0 \times \{0\}$$

$$\alpha' = Q' \setminus ((Q \times \{0\}) \cup \{q_{rej}\})$$

$\delta'$ is defined as follows: for all $\sigma \in \Sigma$ we have $\delta'(q_{rej}) = \{q_{rej}\}$ and for all $\langle q, i \rangle \in Q' \setminus \{q_{rej}\}$ we have

$$\delta'(\langle q, i \rangle, \sigma) = \begin{cases} \delta(q, \sigma) \times \{0, 1\} & i = 0 \\ \delta(q, \sigma) \times \{i + 1\} & 0 < i < k \text{ and } q \notin \alpha \\ \delta(q, \sigma) \times \{1\} & 0 < i \leq k \text{ and } q \in \alpha \\ q_{rej} & i = k \text{ and } q \notin \alpha \end{cases}$$

We claim that $L(\mathcal{A}') = L(\mathcal{A}_P)$. Consider a word $w \in L(\mathcal{A}_P)$, then $\mathcal{A}$ has an accepting run $r$ on $w$ with an eventual bound of at most $k$. $r$ induces a run $r'$ of $\mathcal{A}'$ on $w$ which waits in copy 0 until the $k$ bound is in effect, and then moves between accepting copies and resets to the first copy every (at most) $k$ transitions. This means that $r'$ never gets to $q_{rej}$ and eventually never visits copy 0, so it is stuck in $\alpha'$ and thus accepting. $L(\mathcal{A}_P) \subseteq L(\mathcal{A}')$.

On the other hand, consider a word $w \in L(\mathcal{A}')$. Let $r'$ be an accepting run of $\mathcal{A}'$ on $w$. The run $r'$ gets stuck in $\alpha'$ eventually. By the construction of $\mathcal{A}'$, this implies that eventually, $r'$ visits $\alpha \times \{i\}$ for $0 < i \leq k$ every (at most) $k$ transitions. $r'$ induces a run $r$ of $\mathcal{A}$ on $w$ that eventually visits $\alpha$ every (at most) $k$ transitions, and is therefore accepted by $\mathcal{A}_P$.

We conclude that $L = L(\mathcal{A}_P) = L(\mathcal{A}')$, and therefore $L$ is NCW-recognizable.

### A.4 Proof of Theorem 9

The properties for reg-NPBWs follow from the known properties of NCWs and Theorem 7.

For the deterministic case, we start with closure to union. As we proved in Theorem 5, for a run $r$ of a reg-DPBW automaton we have $inf(r) \cap \alpha \neq \emptyset$ iff $inf(r) \subseteq \alpha$. Let $\mathcal{A}_1, \mathcal{A}_2$ be reg-NPBW. We use the product construction $\mathcal{A}_1 \times \mathcal{A}_2$ and set the accepting states to be $(\alpha_1 \times Q_2) \cup (Q_1 \times \alpha_2)$. If a word was accepted by (w.l.o.g) $\mathcal{A}_1$, then the induced run in the cross product gets stuck in $\alpha_1 \times Q_2$, and is therefore accepting.

On the other hand, let $r$ be an accepting run of $\mathcal{A}_1 \times \mathcal{A}_2$ on some word $w$, then $r$ visits $\alpha_1 \times Q_2$ or $Q_1 \times \alpha_2$ infinitely often (w.l.o.g assume $\alpha_1 \times Q_2$). Therefore, the projection of $r$ on $\mathcal{A}_1$ visits $\alpha_1$ infinitely often, which means it gets stuck in $\alpha_1$, so $w$ is accepted in $\mathcal{A}_1$ and thus in the union.

Closure for intersection follows from the product construction, which preserves determinism. For complementation, we already proved that for a run $r$ in a DPBW, we have $inf(r) \cap \alpha \neq \emptyset$ iff $inf(r) \subseteq \alpha$. This means that if we dualize $\alpha$ we get a DPBW for the complement language. Finally, in order to show that reg-NPBW are strictly more expressive than reg-DPBW, consider the language $L = (a + b)^*.b^\omega$ (only finitely many $a$'s). The language $L$ is NCW-recognizable and therefore is in reg-NPBW. From the NBW typeness of NPBW, it follows that there cannot be a DPBW for $L$. Indeed, had there been a DPBW for $L$ then we would have a DBW for $L$, which is not DBW-recognizable [18].

### A.5 Proof of Theorem 10

**Regularity is in NLOGSPACE.** Let $\mathcal{A} = \langle \Sigma, Q, \{q_0\}, \delta, \alpha \rangle$ be a deterministic automaton. For a state $q \in Q$, we say that $q$ is *pumpable* if $q$ is reachable from itself via a cycle that does not visit $\alpha$.

We first claim that $L(\mathcal{A}_P) \subsetneq L(\mathcal{A}_B)$ iff there exists a reachable state $q' \in \alpha$ such that $q'$ is reachable from itself via a cycle that passes through a pumpable state.

Indeed, for the first direction, assume that there exists a reachable state $q' \in \alpha$ and a pumpable state $q \in Q$ such that $q'$ is reachable from itself via $q$. Let $x, u_1, u_2, z \in \Sigma^*$ be finite words such that $\delta(q_0, x) = q'$, $\delta(q', u_1) = q$, $\delta(q, u_2) = q'$, and $\delta(q', z) = q'$ such that $z$ "witnesses" that $q'$ is pumpable (that is, the run from $q'$ on $z$ does not visit $\alpha$). Consider the word $w = xu_1zu_2u_1z^2u_2u_1z^3u_2u_1z^4....$ Since $\mathcal{A}$ is deterministic, the single run of $\mathcal{A}$ on $w$ does not visit $\alpha$ in the $z^i$ blocks,

and does visit $\alpha$ at least after reading each $u_2$ (which gets back to $q' \in \alpha$). Thus, $w \in L(\mathcal{A}_B) \setminus L(\mathcal{A}_P)$, so $L(\mathcal{A}_P) \subsetneq L(\mathcal{A}_B)$.

For the other direction, assume that $L(\mathcal{A}_P) \subsetneq L(\mathcal{A}_B)$. This implies the existence of a word $w \in L(\mathcal{A}_B) \setminus L(\mathcal{A}_P)$. Let $r$ be the run of $\mathcal{A}$ on $w$, then for every $k \in \mathbb{N}$ there exists an $\alpha$-less $k$-window in $r$. Let $n_0 \in \mathbb{N}$ be an index such that for all $n > n_0$ it holds that $r_n \in inf(r)$. Let $r_{i_1}, ..., r_{i_{n+1}}$ be an $\alpha$-less window such that $i_1 > n_0$. From the pigeonhole principle, there exists $i_1 \leq j_1 < j_2 \leq i_{n+1}$ such that $r_{j_1} = r_{j_2}$. Let $q = r_{j_1}$, then $q$ is pumpable. Furthermore, since $inf(r) \cap \alpha \neq \emptyset$ (as $r$ is accepted by $\mathcal{A}_B$), then there exists $q' \in \alpha \cap inf(r)$ such that $q$ is reachable from itself via $q'$, since $q' \in inf(r)$. Thus, the condition we defined holds.

We now present an algorithm for deciding whether $L(\mathcal{A}_B) \subseteq L(\mathcal{A}_P)$. Given an automaton $\mathcal{A} = \langle \Sigma, Q, \{q_0\}, \delta, \alpha \rangle$ as input, for each reachable state $q' \in \alpha$, check if $q'$ is reachable from itself via a cycle that passes through a pumpable state. If such a state exists, then $L(\mathcal{A}_P) \subsetneq L(\mathcal{A}_B)$. Otherwise $L(\mathcal{A}_P) = L(\mathcal{A}_B)$.

It is easy to see that this algorithm can be implemented in NLOGSPACE, as it is a polynomial length sequence of reachability problems.

**Regularity is NLOGSPACE-hard.** We show a reduction from the reachability problem, which was shown to be NLOGSPACE-hard in [11]. Let $G = \langle V, E \rangle$ be a directed graph, and let $s, t \in V$. Let $|E| = m$ and let $\{e_1, ..., e_m\}$ be an enumeration of the edges. Let $q \notin V$, we construct the DPBW $\mathcal{D} = \langle \{1, ..., m\} \cup \{a, b\}, V \cup \{q\}, \{s\}, \delta, \{t\} \rangle$. We define $\delta$ as follows. For each $e_i = (u, v) \in E$ we define $\delta(u, i) = v$. Also, $\delta(t, a) = q$, $\delta(q, a) = q$, $\delta(q, b) = t$. Clearly, $\mathcal{D}$ is deterministic. We claim that $t$ is reachable from $s$ iff $L(\mathcal{A}_P) = L(\mathcal{A}_B)$. Intuitively, it follows form the fact that we added a non-regular component at state $t$.

For the first direction, assume that $t$ is not reachable from $s$. Thus, there is no reachable accepting state in $\mathcal{D}$, so $L(\mathcal{A}_P) = L(\mathcal{A}_B) = \emptyset$.

For the other direction, assume $t$ is reachable from $s$. Let $s, u_1, ..., u_k, t$ be a path from $s$ to $t$, and assume the edges along this path are $e_{i_0}, ..., e_{i_k}$. Consider the word $w = i_0 \cdot i_1 \cdots i_k \cdot aba^2ba^3ba^4b \cdots$. It is easy to see that $w \in L(\mathcal{A}_B) \setminus L(\mathcal{A}_P)$. Thus, $L(\mathcal{A}_P) \neq L(\mathcal{A}_B)$, and we are done.

**Universality is in NLOGSPACE.** Let $\mathcal{A}_P$ be a DPBW. Since $\Sigma^\omega$ is regular, then , by Theorem 3, we have that that $L(\mathcal{A}_P) = \Sigma^\omega$ iff $L(\mathcal{A}_P) = L(\mathcal{A}_B)$ and $L(\mathcal{A}_B) = \Sigma^\omega$. since $\mathcal{A}$ is deterministic, both can be checked in NLOGSPACE.

**Universality and containment are NLOGSPACE-hard.** It is not hard to see that the reduction from NFW universality to NPBW universality, shown in the proof of Theorem 13 requires constant space and preserves determinization. Thus, it is also a reduction from DFW universality to DPBW universality. Since DFW universality is NLOGSPACE-hard, and universality is a special case of containment, we get PSPACE-harness for both universality and containment.

**Containment is in NLOGSPACE.** Let $\mathcal{B}$ be an NBW and let $\mathcal{A}_P$ be a DPBW. Consider the automaton $\mathcal{C} = \mathcal{B} \times \mathcal{A}$. We say that a state $\langle s, q \rangle \in \mathcal{C}$ is *joint-pumpable* if $\langle s, q \rangle$ is reachable from itself via a cycle whose projection on $\mathcal{A}$ intersects $\alpha_{\mathcal{A}}$, and via a cycle whose projection on $\mathcal{A}$ does not intersects $\alpha_{\mathcal{A}}$, and the projection of one of these cycles on $\mathcal{B}$ intersects $\alpha_{\mathcal{B}}$. It is not hard to see that $L(\mathcal{B}) \subseteq L(\mathcal{A}_P)$ iff there is no joint-pumpable state in $\mathcal{A} \times \mathcal{B}$. The proof is similar to that of DPBW regularity. Also, deciding the existence of a joint-pumpable state can be done in NLOGSPACE. Since NLOGSPACE is closed under complementation, it follows that deciding whether $L(\mathcal{B}) \subseteq L(\mathcal{A}_P)$ is in NLOGSPACE.

### A.6 Proof of Theorem 11

We prove that $L(\mathcal{D}) = \{w : w \text{ is accepted by } \mathcal{A} \text{ with eventual bound } k\}$. For the first direction, let $w \in L(\mathcal{A}_P)$ be accepted with eventual bound $k$. Let $r = q_0, q_1, ...$ be an accepting run of $\mathcal{A}_P$ on $w$ with eventual bound $k$. Thus, there exists an index $l$ such that for all $i > l$ we have that $\{r_i, r_{i+1}, ..., r_{i+k-1}\} \cap \alpha \neq \emptyset$. Consider the run $s = d_0, d_1, ...$ of $\mathcal{D}$ on $w$. Assume by way of contradiction that $s$ is not accepting. This is equivalent to saying that $inf(s) \cap (Q' \setminus \alpha') \neq \emptyset$. Notice that $Q' \setminus \alpha' = \{d : safe(d) = \emptyset\}$. We present two claims.

1. For all $i \in \mathbb{N}$, $q_i \in S_{d_i}$
2. There exists an index $l' > l$ such that for all $i > l'$, $d_i(q_i) \in safe(d)$

We first show that these claims imply that $s$ is accepting. Indeed, for all $i > l'$ the claims imply that $safe(d_i) \neq \emptyset$. Thus $d_i \in \alpha'$, so $s$ gets stuck in $\alpha'$ and is therefore accepting. We now prove the claims.

1. We prove the claim by induction. For $i = 0$, we know that $q_0 \in Q_0$, so by the definition of $d_0 \in Q'_0$ we have $q_0 \in S_{d_0}$. Assume correctness for $i$. Since $r$ is a legal run, then $q_{i+1} \in \delta(q_i, w_i)$. Observe that by the definition of $\delta'$, $S_{d_{i+1}} = (\delta(safe(d_i), w_i) \cap \alpha) \cup (\delta(S_{d_i}, w_i) \cap (Q \setminus \alpha)) \cup (\delta(S_{d_i}, w_i) \setminus \delta(safe(d_i), w_i) \cap \alpha) = \delta(S_{d_i}, w_i)$. Since $q_{i+1} \in \delta(q_i, w_i)$ and $q_i \in S_{d_i}$, then $q_{i+1} \in S_{d_{i+1}}$.
2. Recall that after $l$, the run $r$ visits $\alpha$ in every $k$ states interval. Let $l'$ be the minimal index greater than $l$ such that $d_{l'} \notin \alpha'$. By our contradictory assumption, such $l'$ exists.
   We now prove by induction over $i$ that $q_{l'+i} \in safe(d_{l'+i})$ and that for $0 \leq j < d_{l'+i}(q_{l'+i})$ we have $r_{l'+i-j} \notin \alpha$. For $i = 1$, from claim 1 we get that $d_{l'}(q_{l'}) \in S_{d_{l'}}$. Also, $d_{l'} \notin \alpha'$, implying that $d_{l'}(q_{l'}) = \infty$. From the definition of $\delta'$, and since $q_{l'+1} \in \delta(q_{l'}, w_{l'})$, then $d_{l'+1}(q_{l'+1}) = 0$. The condition that for all $1 \leq j \leq d_{l'+i}$ trivially holds in this case. Assume correctness for $i$, we prove for $i + 1$. Since $d_{l'+i}(q_{l'+i}) \in \{0, ..., k-1\}$ and since $q_{l'+i+1} \in \delta(q_{l'+i}, w_{l'+i})$, then $q_{l'+i+1} \in S_{d_{l'+i+1}}$. If $q_{l'+i+1} \in \alpha$, then by the definition of $\delta'$ we get that $d_{l'+i+1}(q_{l'+i+1}) = 0$, so the claim holds. Otherwise, let $m = d_{l'+i}(q_{l'+i})$. By the definition of $\delta'$ we have that $d_{l'+i+1}(q_{l'+i+1}) \leq m + 1$. If $m = k - 1$,

then by the induction hypothesis, for $j \in \{1, ..., k-1\}$ we have $q_{l'+i-j} \notin \alpha$. Since the eventual bound $k$ is in effect after $l$, this implies that $q_{l'+i+1} \in \alpha$, otherwise the run violates the bound. Thus, $d_{l'+i+1}(q_{l'+i+1}) = 0$ and the claim holds. So $m < k-1$. Thus, $m+1 \leq k-1$, so $q_{l'+i+1} \in \mathit{safe}(d_{l'+i+1})$. Furthermore, for all $0 < j < d_{l'+i+1}(q_{l'+i+1})$ we know from the induction hypothesis that $q_{l'+i+1-j} \notin \alpha$. For $j = 0$ we know that $q_{l'+i+1} \notin \alpha$. Thus, we are done.

We conclude that $L(\mathcal{A}_P) \subseteq L(\mathcal{D})$.

For the other direction, let $w \in L(\mathcal{D})$. Thus, the single run $r = d_0, d_1, d_2, ...$ of $\mathcal{D}$ on $w$ eventually gets stuck in $\alpha'$. Consider the run DAG of $\mathcal{A}$ on $w$. The run DAG is defined as the graph $G = \langle V, E \rangle$ such that $V = Q \times \mathbb{N}$, and there is an edge $(\langle q, i \rangle, \langle q', j \rangle)$ iff $j = i+1$ and $q' \in \delta(q, w_i)$. Thus, every run of $\mathcal{A}$ on $w$ induces an infinite path in $G$ and vice-versa. The run $r$ induces a $\{0, ..., k-1, \infty, \bot\}$-labeling on the vertices of $G$ by labeling the vertex $\langle q, i \rangle$ with $d_i(q)$ if $q \in S_{d_i}$ and $\bot$ otherwise. Let $l$ be the index such that for all $i > l$ $d_i \in \alpha'$. Thus, for all $i > l$ there exists $q \in Q$ such that $q$ is $d_i$-safe. We now restrict to the subgraph $G'$ of $G$ induced by $V \setminus \{\langle q, i \rangle : d_i(q) = \bot \text{ or } i > l \text{ and } d_i(q) = \infty\}$. Thus, we remove all vertices that are labeled $\bot$ and all vertices that are labeled $\infty$ after level $l$. Recall that on any level $i > l$ there is a vertex $q$ such that $d_i(q) \in \{0, ..., k-1\}$. Thus, $G'$ is infinite.

We now claim that the graph $G'$ contains a path of any finite length starting at level 1. Indeed, let $j > l$, and let $\langle q, j \rangle$ be a vertex such that $q$ is $d_j$-safe. By the definition of $\delta'$, we see that $d_j(q) \in \{0, ..., k-1\}$ only in two cases. Either $q \in \delta(q', w_j)$ and $d_{j-1}(q') \in \{0, ..., k-1\}$, or $q \in \delta(q', w_j)$ and $d_{j-1} \notin \alpha'$. Since $r$ is accepting, the latter condition can only hold for $j \leq l$. We can continue tracking back nodes until level $l$. From level $l$ we can continue in a path to level 1, since if $d_i(q) \neq \bot$ then we can find $q'$ such that $d_{i-1}(q') \neq \bot$ and $q \in \delta(q')$. We conclude the existence of a path $q_0, q_1, ..., q_{j-1}, q$ in $G'$. Thus, we have a path of any finite length starting at level 1 in $G'$.

Since each vertex has an exit degree of at most $|Q|$, then every vertex has finite degree. Since there is a path of any finite length, we can use König's lemma and conclude there is an infinite path $\pi$ in $G'$. $\pi$ Induces an infinite run $q_0, q_1, ...$ of $\mathcal{A}$ on $w$, such that for $i > l$ $q_i$ is $d_i$-safe. Note that this run is not necessarily accepting. However, if $d_i(q_i) \in \{0, ..., k-1\}$, this implies that there exists $q' \in \alpha$ in level $i' \in \{i, i-1, ..., i-k\}$ such that $\langle q, i \rangle$ is reachable from $zugq', i'$ in $G'$. Since this is true for all $i > l$, we conclude there are infinitely many $\langle q, i \rangle$ nodes in $G'$ such that $q \in \alpha$ and $q$ is $d_i$-safe. We refer to such nodes as *$\alpha$-nodes.*

Now, observe the set $O = \{\langle q, i \rangle \in G' : \langle q, i \rangle \text{ is an } \alpha\text{-node and } l < i \leq l+k\}$. From what we proved, this set is not empty. Clearly, it is also finite. Assume by contradiction that from each $q \in O$ there are only finitely many reachable $\alpha$-nodes. Thus, there are only finitely many $\alpha$ nodes reachable from $O$. Since there are infinitely many $\alpha$-nodes in $G'$, this implies the existence of an $\alpha$-node $\langle q', j \rangle \in G'$ that is not reachable from $O$ such that $j > l+k$. We can trace a path

from $\langle q', j\rangle$ to some state $\langle q'', l\rangle$ such that in every $k$ states there is an $\alpha$-node. However, this means that $\langle q', j\rangle$ is reachable from an $\alpha$ node in $O$- contradiction.

Thus, there exists an $\alpha$-node from which there are infinitely many reachable $\alpha$-nodes. Note that we can also require that any of the reachable nodes will be on a path that passes through $\alpha$ every at most $k$ steps. In an argument similar to König's Lemma, we can conclude that there exists a path through $G'$ with infinitely many $\alpha$ nodes in intervals of at most $k$. Furthermore, if this path starts in $\langle q, i\rangle$ for $i > 1$ then $d_i(q) \in \{1, ..., k, \infty\}$, and we can trace back this path to level 1 only through nodes labeled with $\{1, ..., k, \infty\}$. This path induces an accepting run of $\mathcal{A}_P$ on $w$, and we are done.

### A.7 Proof of Theorem 13

**Universality and containment are PSPACE-hard.** Recall that deciding the universality problem for NFW is PSPACE-hard. We show a reduction from NFW universality to NPBW universality. Since universality is a special case of containment, we get PSPACE-harness for both universality and containment. The reduction is similar to the one described in [16] for universality of fair transition systems. Given an NFW $\mathcal{A} = \langle \Sigma, Q, \delta, Q_0, \alpha\rangle$ as input for the NFW universality problem, we construct an NPBW $\mathcal{A}' = \langle \Sigma \cup \{\#\}, Q \cup \{q_{acc}, q_\Sigma\}, \delta', Q_0 \cup \{q_\Sigma\}, \alpha \cup \{q_\Sigma, q_{acc}\}\rangle$, where

$$\delta'(q,\sigma) = \begin{cases} \delta(q,\sigma) & q \in Q, \sigma \in \Sigma \\ q_{acc} & q = q_{acc}, \sigma \in \Sigma \cup \{\#\} \\ q_\Sigma & q = q_\Sigma, \sigma \in \Sigma \\ q_{acc} & q \in \alpha, \sigma = \# \end{cases}$$

Thus, on letters from $\Sigma$, the NPBW $\mathcal{A}'$ goes to $q_\Sigma$, from which it accepts all words in $\Sigma^\omega$, and can also continues in the NFW $\mathcal{A}$, waiting for a visit in $\alpha$. From the accepting states of $\mathcal{A}$, the NPBW $\mathcal{A}'$ can read the letter # and go to an accepting sink.

We claim that $L(\mathcal{A}) = \Sigma^*$ iff $L(\mathcal{A}'_P) = (\Sigma \cup \{\#\})^\omega$. First observe that for every $w \in \Sigma^\omega$ it holds that $w \in L(\mathcal{A}'_P)$, since the run $(q_\Sigma)^\omega$ is an accepting legal run on $w$. If $L(\mathcal{A}) = \Sigma^*$, then for any word $x \in \Sigma^*$ there exists a run $r$ of $\mathcal{A}$ on $x$ such that $r_{|x|} \in \alpha$. Let $w \in (\Sigma \cup \{\#\})^\omega$, then either $w \in \Sigma^\omega$ or $w \in \Sigma^*\#(\Sigma \cup \{\#\})^\omega$, in the first case, $w \in L(\mathcal{A}'_P)$ be our observation. In the second case, let $w = x\#z$ where $x \in \Sigma^*$ and $z \in (\Sigma \cup \{\#\})^\omega$. $\mathcal{A}$ has an accepting run on $x$, which induces a prefix of a run $r'$ of $\mathcal{A}'$ on $w$, such that $r'_{|x|} \in \alpha$. Thus, when $\mathcal{A}'$ reads the first # in $w$, it can pass to $q_{acc}$, thus there is an accepting run of $\mathcal{A}'$ on $w$, so $L(\mathcal{A}'_P) = \Sigma^\omega$.

For the other direction, if $L(\mathcal{A}) \neq \Sigma^*$ then there exists a word $x \in \Sigma^*$ such that $x \notin L(\mathcal{A})$. Observe the runs of $\mathcal{A}'_P$ on the word $w = x\#^\omega$. There is a single run that starts from $q_\Sigma$, which is not accepting as it reaches and stays in a rejecting state after reading #. All the runs that start from $Q_0$ do not reach an

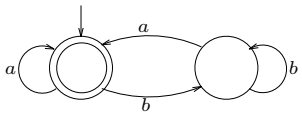


**Fig. 2.** $L(\mathcal{B}_B) = (a \cdot b^*)^\omega$

accepting state after reading $x$, and therefor all the runs go to a rejecting state after reading #. Thus $w \notin L(\mathcal{A}'_P)$, and therefore $L(\mathcal{A}'_P) \neq (\Sigma \cup \#)^\omega$.

**Regularity is PSPACE-hard.** We show a reduction from the universality problem for NPBW, we proved to be PSPACE-hard. Let $\mathcal{B} = \langle \{a, b\}, Q_\mathcal{B}, \delta_\mathcal{B}, Q^0_\mathcal{B}, \alpha_\mathcal{B} \rangle$ be the automaton in Figure 2.

Given two infinite words $w_1 = \sigma_1\sigma_2\sigma_3 \cdots \in \Sigma^\omega$ and $w_2 = \tau_1\tau_2\tau_3 \cdots \in \{a, b\}^\omega$, we define

$$w_1 \oplus w_2 = \langle \sigma_1, \tau_1 \rangle \langle \sigma_2, \tau_2 \rangle \langle \sigma_3, \tau_3 \rangle \cdots \in (\Sigma \times \{a, b\})^\omega.$$

Given an NPBW $\mathcal{A} = \langle \Sigma, Q_\mathcal{A}, \delta_\mathcal{A}, Q^0_\mathcal{A}, \alpha_\mathcal{A} \rangle$, an input for the universality problem, we define the automaton $\mathcal{C} = \mathcal{A} \times \mathcal{B} = \langle \Sigma \times \{a, b\}, Q_\mathcal{A} \times Q_\mathcal{B}, \delta, Q^0_\mathcal{A} \times Q^0_\mathcal{B}, \alpha \rangle$, where $\delta(\langle q_1, q_2 \rangle, \langle \sigma_1, \sigma_2 \rangle) = \delta_\mathcal{A}(q_1, \sigma_1) \times \delta_\mathcal{B}(q_2, \sigma_2)$, and $\alpha = (\alpha_\mathcal{A} \times Q_\mathcal{B}) \cup (Q_\mathcal{A} \times \alpha_\mathcal{B})$. It is not hard to see that $\mathcal{C}$ accepts a word $w_1 \oplus w_2$ if $\mathcal{A}$ accepts $w_1$ or $\mathcal{B}$ accepts $w_2$.

We prove that $\mathcal{A}$ is universal iff $L(\mathcal{C}_P) = L(\mathcal{C}_B)$. For the first direction, if $\mathcal{A}$ is universal, then for every infinite word $w_1 \in \Sigma^\omega$ there is an accepting run of $\mathcal{A}_P$ on $w$. This implies that for every word $w_1 \oplus w_2 \in (\Sigma \times \{a, b\})^\omega$ there is an accepting run of $\mathcal{C}_P$ on $w_1 \oplus w_2$. Indeed, an accepting run of $\mathcal{A}_P$ on $w_1$ induces a run of $\mathcal{C}$ on $w_1 \oplus w_2$ that visits $\alpha_\mathcal{A} \times Q_\mathcal{B} \subseteq \alpha_\mathcal{C}$ promptly. Hence $L(\mathcal{C}_P) = (\Sigma \times \{a, b\})^\omega$. Since $L(\mathcal{C}_P) \subseteq L(\mathcal{C}_B)$ we conclude that $L(\mathcal{C}_P) = L(\mathcal{C}_B) = (\Sigma \times \{a, b\})^\omega$.

For the other direction, assume that $\mathcal{A}$ is not universal. Consider a word $w_1 \notin L(\mathcal{A}_P)$. Since $L(\mathcal{A}) \neq \Sigma^\omega$, such a word $w_1$ exists. We distinguish between two cases. If $w_1 \notin L(\mathcal{A}_B)$, then every run of $\mathcal{A}$ on $w_1$ visits $\alpha_\mathcal{A}$ only a finite number of times. Let $w_2 \in \{a, b\}^\omega$ be an infinite word such that $w_2 \in L(\mathcal{B}_B) \setminus L(\mathcal{B}_P)$. For example, $w_2 = ab^2ab^3ab^4 \cdots$. Consider the word $w_1 \oplus w_2$. Since $w_1 \notin L(\mathcal{A}_P)$, every run of $\mathcal{C}$ on $w_1 \oplus w_2$ visits $\alpha_\mathcal{A} \times Q_\mathcal{B}$ only a finite number of times. Also, since $w_2 \in L(\mathcal{B}_B)$, there exists a run of $\mathcal{C}$ on $w_1 \oplus w_2$ that visits $Q_\mathcal{A} \times \alpha_\mathcal{B}$ infinitely many times. Further notice that since $w_2 \notin L(\mathcal{B}_P)$, there is no run that visits $Q_\mathcal{A} \times \alpha_\mathcal{B}$ promptly. We conclude that $w_1 \oplus w_2 \in L(\mathcal{C}_B) \setminus L(\mathcal{C}_P)$, thus $L(\mathcal{C}_P) \neq L(\mathcal{C}_B)$.

For the second case, assume $w_1 \in L(\mathcal{A}_B)$. Let $w_2 = b^\omega$, and consider the word $w_1 \oplus w_2$. By definition, the single run of $\mathcal{B}$ on $w_2$ visits always the non-accepting state. Recall that $w_1 \notin L(\mathcal{A}_P)$, which means that every run of $\mathcal{C}$ on $w_1 \oplus w_2$ does not visit $\alpha_\mathcal{A} \times Q_\mathcal{B}$ promptly. In addition, every run of $\mathcal{C}$ on $w_1 \oplus w_2$ does not visit $Q_\mathcal{A} \times \alpha_\mathcal{B}$ at all. However, an accepting run of $\mathcal{A}_B$ on $w_1$ induces

an accepting run of $\mathcal{C}_B$ on $w_1 \oplus w_2$, thus $w_1 \oplus w_2 \in L(\mathcal{C}_B) \setminus L(\mathcal{C}_P)$ and again, $L(\mathcal{C}_P) \neq L(\mathcal{C}_B)$.