On Semantically-Deterministic Automata

Bader Abu Radi
School of Computer Science and Engineering, Hebrew University, Jerusalem, Israel

Orna Kupferman
School of Computer Science and Engineering, Hebrew University, Jerusalem, Israel

Abstract

Nondeterminism is a fundamental notion in Theoretical Computer Science. A nondeterministic automaton is *semantically deterministic* (SD) if different nondeterministic choices in the automaton lead to equivalent states. Semantic determinism is interesting as it is a natural relaxation of determinism, and as some applications of automata in formal methods require deterministic automata, yet in fact can use automata with some level of nondeterminism, tightly related to semantic determinism.

In the context of finite words, semantic determinism coincides with determinism, in the sense that every pruning of an SD automaton to a deterministic one results in an equivalent automaton. We study SD automata on infinite words, focusing on Büchi, co-Büchi, and weak automata. We show that there, while semantic determinism does not increase the expressive power, the combinatorial and computational properties of SD automata are very different from these of deterministic automata. In particular, SD Büchi and co-Büchi automata are exponentially more succinct than deterministic ones (in fact, also exponentially more succinct than history-deterministic automata), their complementation involves an exponential blow up, and decision procedures for them like universality and minimization are PSPACE-complete. For weak automata, we show that while an SD weak automaton need not be pruned to an equivalent deterministic one, it can be determinized to an equivalent deterministic weak automaton with the same state space, implying also efficient complementation and decision procedures for SD weak automata.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases Automata on infinite words, Nondeterminism, Succinctness, Decision procedures

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.109

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: https://arxiv.org/abs/2305.15489 [2]

Funding This research is supported by the Israel Science Foundation, Grant 2357/19, the European Research Council, Advanced Grant ADVANSYNT, and the Neubauer Foundation of Ph.D Fellowship.

1 Introduction

Automata are among the most studied computation models in theoretical computer science. Their simple structure has made them a basic formalism for the study of fundamental notions, such as *determinism* and *nondeterminism* [35]. While a deterministic computing machine examines a single action at each step of its computation, nondeterministic machines are allowed to examine several possible actions simultaneously. Understanding the power of nondeterminism is at the core of open fundamental questions in theoretical computer science (most notably, the P vs. NP problem).

A prime application of *automata on infinite words* is specification, verification, and synthesis of nonterminating systems. The automata-theoretic approach reduces questions about systems and their specifications to questions about automata [28, 41], and is at the heart of many algorithms and tools. A run of an automaton on infinite words is an infinite

sequence of states, and acceptance is determined with respect to the set of states that the run visits infinitely often. For example, in *Büchi* automata, some of the states are designated as accepting states, and a run is accepting iff it visits states from the set α of accepting states infinitely often [9]. Dually, in *co-Büchi* automata, a run is accepting if it visits the set α only finitely often. Then, *weak* automata are a special case of both Büchi and co-Büchi automata in which every strongly connected component in the graph induced by the automaton is either contained in α or is disjoint from α. We use DBW and NBW to denote determinisitic and nondeterministic Büchi word automata, respectively, and similarly for D/NCW, D/NWW, and D/NFW, for co-Büchi, weak, and automata on finite words, respectively.

For automata on infinite words, nondeterminism not only leads to exponential succinctness, but may also increase the expressive power. This is the case, for example, in Büchi and weak automata, thus NBWs are strictly more expressive than DBWs [29], and NWWs are strictly more expressive than DWWs [6]. On the other hand, NCWs are as expressive as DCWs [32], and in fact, also as NWWs [27]. In some applications of the automata-theoretic approach, such as model checking, algorithms can be based on nondeterministic automata, whereas in other applications, such as synthesis and reasoning about probabilistic systems, they cannot. There, the advantages of nondeterminism are lost, and algorithms involve a complicated determinization construction [36] or acrobatics for circumventing determinization [26, 21].

In a deterministic automaton, the transition function maps each state and letter to a single successor state. In recent years there is growing research on weaker types of determinism. This includes, for example, *unambiguous automata*, which may have many runs on each word, yet only one accepting run [10, 12], automata that are *deterministic in the limit*, where each accepting run should eventually reach a deterministic sub-automaton [40], and automata that are *determinizable by pruning* (DBP), thus embody an equivalent deterministic automaton [4].

In terms of applications, some weaker types of determinism have been defined and studied with specific applications in mind. Most notable are *history-deterministic* automata (HD), which can resolve their nondeterministic choices based on the history of the run [18, 8, 23][1], and can therefore replace deterministic automata in algorithms for synthesis and control, and *good-for-MDPs* automata (GFM), whose product with Markov decision processes maintains the probability of acceptance, and can therefore replace deterministic automata when reasoning about stochastic behaviors [16, 39].

The different levels of determinism induce classes of automata that differ in their succinctness and in the complexity of operations and decision problems on them. Also, some classes are subclasses of others. For example, it follows quite easily from the definitions that every automaton that is deterministic in the limit is GFM, and every automaton that is DBP is HD.

In this paper we study the class of *semantically deterministic* automata (SD). An automaton $\mathcal{A}$ is SD if its nondeterministic choices lead to equivalent states. Formally, if $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, with a transition function $\delta : Q \times \Sigma \to 2^Q$, then $\mathcal{A}$ is SD if for every state $q \in Q$, letter $\sigma \in \Sigma$, and states $q_1, q_2 \in \delta(q, \sigma)$, the set of words accepted from $\mathcal{A}$ with initial state q_1 is equal to the set of words accepted from $\mathcal{A}$ with initial state q_2. Since all nondeterministic choices lead to equivalent states, one may be tempted to think that SD automata are DBP or at least have similar properties to deterministic automata. This is

[1] The notion used in [18] is *good for games* (GFG) automata, as they address the difficulty of playing games on top of a nondeterministic automaton. As it turns out, the property of being good for games varies in different settings and HD is good for applications beyond games. Therefore, we use the term *history determinism*, introduced by Colcombet in the setting of quantitative automata with cost functions [11].

indeed the case for SD-NFWs, namely when one considers automata on finite words. There, it is not hard to prove that any pruning of an SD-NFW to a DFW results in an equivalent DFW. Thus, SD-NFWs are not more succinct than DFWs, and operations on them are not more complex than operations on DFWs.

Once, however, one considers automata on infinite words, the simplicity of SD automata is lost. In order to understand the picture in the setting of infinite words, let us elaborate some more on HD automata, which are strongly related to SD automata. Formally, a nondeterministic automaton $\mathcal{A}$ is HD if there is a strategy g that maps each finite word $u \in \Sigma^*$ to a transition to be taken after u is read, and following g results in accepting all the words in the language of $\mathcal{A}$. Obviously, every DBP automaton is HD – the strategy g can suggest the same transition in all its visits in a state. On the other hand, while HD-NWWs are always DBP [25, 33], this is not the case for HD-NBWs and HD-NCWs [7]. There, the HD strategy may need to suggest different transitions in visits (with different histories) to the same state.

It is easy to see that a strategy g as above cannot choose a transition to states whose language is strictly contained in the language of states that are reachable by other transitions. Thus, all HD automata can be pruned in polynomial time to SD automata [20, 5]. The other direction, however, does not hold: it is shown in [3] that an SD-NWW need not be HD (hence, SD-NBWs and SD-NCWs need not be HD too). Moreover, while all all HD-NWWs are DBP, this is not the case for all SD-NWWs.

In this work we study the succinctness of SD automata with respect to deterministic ones, as well as the complexity of operations on them and decision problems about them. Our goal is to understand the difference between determinism and semantic determinism, and to understand how this difference varies among different acceptance conditions. Our study is further motivated by the applications of automata with different levels of nondeterminism in algorithms for synthesis and for reasoning in a stochastic setting. In particular, beyond the connection to HD automata discussed above, as runs of an SD-NBW on words in the language are accepting with probability 1, all SD-NBWs are GFM [3].

We study semantic determinism for Büchi, co–Büchi, and weak automata. We consider automata with both *state-based* acceptance conditions, as defined above, and *transition-based* acceptance conditions. In the latter, the acceptance condition is given by a subset α of transitions, and a run is required to traverse transitions in α infinitely often (in Büchi automata, termed tD/tNBW) or finitely often (in co-Büchi automata, termed tD/tNCW). As it turns out, especially in the context of HD automata, automata with transition-based acceptance conditions may differ in their properties from automata with traditional state-based acceptance conditions. For example, while HD-tNCWs can be minimized in PTIME [1], minimization of HD-NCWs is NP-complete [38]. In addition, there is recently growing use of transition-based automata in practical applications, with evidences they offer a simpler translation of LTL formulas to automata and enable simpler constructions and decision procedures [14, 15, 13, 40, 30]. Our results for all types of acceptance conditions are summarized in Table 1 below, where we also compare them with known results about deterministic and HD automata.

Let us highlight the results we find the most interesting and surprising. While all three types of SD automata are not DBP, we are able to determinize SD-NWWs in polynomial time, and end up with a DWW whose state space is a subset of the state space of the original SD-NWW. Essentially, rather than pruning transitions, the construction redirects transitions to equivalent states in deep strongly connected components of the SD-NWW, which we prove

Table 1 Succinctness (determinization blow-up), complementation (blow-up in going from an automaton to a complementing one), universality (deciding whether an automaton accepts all words), and minimization (deciding whether an equivalent automaton of a given size exists, and the described results apply also for the case the given automaton is deterministic). All blow-ups are tights, except for HD-NBW determinization, where the quadratic bound has no matching lower bound; all NL, NP, and PSPACE bounds are complete.

	Deterministic	HD	SD
Succinctness	n (B,C,W)	n^2 (B), 2^n (C), n (W) [20, 25, 33]	2^n (B,C), n (W) Theorems 5, 14, 22
Complementation	n (B,C,W) [22]	n (B), 2^n (C), n (W) [20]	2^n (B,C), n (W) Theorems 7, 16, 23
Universality	NL (B,C,W) [22]	P (B,C,W) [17, 20]	PSPACE (B,C), P (W) Theorems 9, 17, 24
Minimization (state based)	NP (B,C), P (W) [37, 31]	NP (B,C), P (W) [38, 25, 31]	PSPACE (B,C), P (W) Theorems 10, 18, 24
Minimization (transition based)	open (B,C), P (W) [31]	open (B), P (C,W) [1, 25, 31]	PSPACE (B,C), P (W) Theorems 10, 18, 24

to result in an equivalent deterministic DWW[2]. This suggests that despite the "not DBP anomaly" of SD-NWWs, they are very similar in their properties to DWWs. On the other hand, except for their expressive power, SD-NBWs and SD-NCWs are not at all similar to DBWs and DCWs: while HD-NBWs are only quadratically more succinct than DBWs, succinctness jumps to exponential for SD-NBWs. This also shows that SD-NBWs may be exponentially more succinct than HD-NBWs, and we show this succinctness gap also for co-Büchi automata, where exponential succinctness with respect to DCWs holds already in the HD level.

The succinctness results are carried over to the blow-up needed for complementation, and to the complexity of decision procedures. Note that this is not the case for HD automata. There, for example, complementation of HD-NBWs results in an automaton with the same number of states [20]. Moreover, even though HD-NCWs are exponentially more succinct than DCWs, language-containment for HD-NCWs can be solved in PTIME [17, 20], and HD-tNCWs can be minimized in PTIME [1]. For SD automata, we show that complementation involves an exponential blow-up. Also, universality and minimization of SD Büchi and co-Büchi automata, with either state-based or transition-based acceptance conditions, is PSPACE-complete, as it is for NBWs and NCWs. We also study the *D-to-SD minimization problem*, where we are given a deterministic automaton $\mathcal{A}$ and a bound $k \geq 1$, and need to decide whether $\mathcal{A}$ has an equivalent SD automaton with at most k states. Thus, the given automaton is deterministic, rather than SD. By [19], the D-to-N minimization problem for automata on finite words (that is, given a DFW, minimize it to an NFW) is PSPACE-complete. It is easy to see that the D-to-SD minimization problem for for automata on finite words can be solved in PTIME. We show that while this is the case also for weak automata, D-to-SD minimization for Büchi and co-Büchi automata is PSPACE-complete.

[2] Our result implies that checking language-equivalence between states in an SD-NWW can be done in PTIME, as the check can be performed on the equivalent DWWs. We cannot, however, use this complexity result in our algorithm, as this involves a circular dependency. Consequently, our construction of the equivalent DWWs involves a language-approximation argument.

Our results show that in terms of combinatorial and computational properties, semantic determinism is very similar to determinism in weak automata, whereas for Büchi and co-Büchi automata, semantic determinism is very similar to nondeterminism.

Due to the lack of space, some proofs are missing, and can be found in the full version [2].

2 Preliminaries

2.1 Languages and Automata

For a finite nonempty alphabet Σ, an infinite *word* $w = \sigma_1 \cdot \sigma_2 \cdots \in \Sigma^\omega$ is an infinite sequence of letters from Σ. A *language* $L \subseteq \Sigma^\omega$ is a set of words. For $1 \leq i \leq j$, we use $w[i,j]$ to denote the infix $\sigma_i \cdot \sigma_{i+1} \cdots \sigma_j$ of w, use $w[i]$ to denote the letter σ_i, and use $w[i,\infty]$ to denote the infinite suffix $\sigma_i \cdot \sigma_{i+1} \cdots$ of w. We also consider languages $R \subseteq \Sigma^*$ of finite words, denote the empty word by ϵ, and denote the set of nonempty words over Σ by Σ^+; thus $\Sigma^+ = \Sigma^* \setminus \{\epsilon\}$. For a set S, we denote its complement by $\overline{S}$. In particular, for languages $R \subseteq \Sigma^*$ and $L \subseteq \Sigma^\omega$, we have $\overline{R} = \Sigma^* \setminus R$ and $\overline{L} = \Sigma^\omega \setminus L$.

A *nondeterministic automaton* is a tuple $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$, where Σ is an alphabet, Q is a finite set of *states*, Q_0 is a set of *initial states*, $\delta : Q \times \Sigma \to 2^Q \setminus \emptyset$ is a *transition function*, and α is an *acceptance condition*, to be defined below. For states q and s and a letter $\sigma \in \Sigma$, we say that s is a *σ-successor* of q if $s \in \delta(q, \sigma)$. Note that the transition function of $\mathcal{A}$ is total, thus for all states $q \in Q$ and letters $\sigma \in \Sigma$, q has at least one σ-successor. If $|Q_0| = 1$ and every state q has a single σ-successor, for all letters σ, then $\mathcal{A}$ is *deterministic*. The transition function δ can be viewed as a transition relation $\Delta \subseteq Q \times \Sigma \times Q$, where for every two states $q, s \in Q$ and letter $\sigma \in \Sigma$, we have that $\langle q, \sigma, s \rangle \in \Delta$ iff $s \in \delta(q, \sigma)$. We define the *size* of $\mathcal{A}$, denoted $|\mathcal{A}|$, as its number of states, thus, $|\mathcal{A}| = |Q|$.

Given an input word $w = \sigma_1 \cdot \sigma_2 \cdots$, a *run* of $\mathcal{A}$ on w is a sequence of states $r = r_0, r_1, r_2, \ldots$, such that $r_0 \in Q_0$, and for all $i \geq 0$, we have that $r_{i+1} \in \delta(r_i, \sigma_{i+1})$, i.e, the run starts in some initial state and proceeds according to the transition function. If the word in the input is infinite, then so is the run. We sometimes view the run $r = r_0, r_1, r_2, \ldots$ on $w = \sigma_1 \cdot \sigma_2 \cdots$ as a sequence of successive transitions $\langle r_0, \sigma_1, r_1 \rangle, \langle r_1, \sigma_2, r_2 \rangle, \ldots$. Note that a deterministic automaton has a single run on an input w. We sometimes extend δ to sets of states and finite words. Then, $\delta : 2^Q \times \Sigma^* \to 2^Q$ is such that for every $S \in 2^Q$, finite word $u \in \Sigma^*$, and letter $\sigma \in \Sigma$, we have that $\delta(S, \epsilon) = S$, $\delta(S, \sigma) = \bigcup_{s \in S} \delta(s, \sigma)$, and $\delta(S, u \cdot \sigma) = \delta(\delta(S, u), \sigma)$. Thus, $\delta(S, u)$ is the set of states that $\mathcal{A}$ may reach when it reads u from some state in S.

The acceptance condition α determines which runs are "good". For automata on finite words, $\alpha \subseteq Q$, and a run is *accepting* if it ends in a state in α. For automata on infinite words, we consider *state-based* and *transition-based* acceptance conditions. Let us start with *state-based* conditions. Here, $\alpha \subseteq Q$, and we use the terms *α-states* and *$\overline{\alpha}$-states* to refer to states in α and in $Q \setminus \alpha$, respectively. For a run $r \in Q^\omega$, let $\mathit{sinf}(r) \subseteq Q$ be the set of states that r visits infinitely often. Thus, $\mathit{sinf}(r) = \{q : q = r_i \text{ for infinitely many } i\text{'s}\}$. In *Büchi* automata, r is *accepting* iff $\mathit{sinf}(r) \cap \alpha \neq \emptyset$, thus if r visits states in α infinitely often. Dually, in *co-Büchi* automata, r is *accepting* iff $\mathit{sinf}(r) \cap \alpha = \emptyset$, thus if r visits states in α only finitely often.

We proceed to *transition-based* conditions. There, $\alpha \subseteq \Delta$ and acceptance depends on the set of transitions that are traversed infinitely often during the run. We use the terms *α-transitions* and *$\bar{\alpha}$-transitions* to refer to transitions in α and in $\Delta \setminus \alpha$, respectively. For a run $r \in \Delta^\omega$, we define $\mathit{tinf}(r) = \{\langle q, \sigma, s \rangle \in \Delta : q = r_i, \sigma = \sigma_{i+1}, \text{ and } s = r_{i+1}, \text{ for infinitely many } i\text{'s}\}$. As expected, in transition-based Büchi automata, r is accepting iff $\mathit{tinf}(r) \cap \alpha \neq \emptyset$, and in transition-based co-Büchi automata, r is accepting iff $\mathit{tinf}(r) \cap \alpha = \emptyset$.

Consider an automaton $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$. In all automata classes, a run of $\mathcal{A}$ that is not accepting is *rejecting*. A word w is accepted by an automaton $\mathcal{A}$ if there is an accepting run of $\mathcal{A}$ on w. The language of $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of words that $\mathcal{A}$ accepts.

Consider a directed graph $G = \langle V, E \rangle$. A *strongly connected set* in G (SCS, for short) is a set $C \subseteq V$ such that for every two vertices $v, v' \in C$, there is a path from v to v'. An SCS is *maximal* if for every non-empty set $C' \subseteq V \setminus C$, it holds that $C \cup C'$ is not an SCS. The *maximal strongly connected sets* are also termed *strongly connected components* (SCCs, for short). An automaton $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$ induces a directed graph $G_{\mathcal{A}} = \langle Q, E \rangle$, where $\langle q, q' \rangle \in E$ iff there is a letter $\sigma \in \Sigma$ such that $\langle q, \sigma, q' \rangle \in \Delta$. The SCSs and SCCs of $\mathcal{A}$ are those of $G_{\mathcal{A}}$.

An automaton $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$ with a state-based acceptance condition $\alpha \subseteq Q$ is *weak* [34] if for each SCC C of $\mathcal{A}$, either $C \subseteq \alpha$ in which case C is *accepting*, or $C \cap \alpha = \emptyset$ in which case C is *rejecting*. We view $\mathcal{A}$ as a Büchi automaton, yet note that a weak automaton can be viewed as both a Büchi and a co-Büchi automaton. Indeed, a run of $\mathcal{A}$ visits α infinitely often iff it gets trapped in an SCC that is contained in α iff it visits states in $Q \setminus \alpha$ only finitely often. Note also that when $\mathcal{A}$ uses a transition-based acceptance condition, we can ignore the membership in α of transitions between SCCs (indeed, such transitions are traversed only finitely often), and say that $\mathcal{A}$ is weak if the transitions in each SCC are all in α or all disjoint from α.

Consider two automata $\mathcal{A}_1$ and $\mathcal{A}_2$. We say that $\mathcal{A}_1$ is *contained in* $\mathcal{A}_2$ if $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$. Then, $\mathcal{A}_1$ and $\mathcal{A}_2$ are *equivalent* if $L(\mathcal{A}_1) = L(\mathcal{A}_2)$, and $\mathcal{A}_1$ is *universal* if $L(\mathcal{A}_1) = \Sigma^\omega$ (or $L(\mathcal{A}_1) = \Sigma^*$, in case it runs on finite words). Finally, $\mathcal{A}_1$ is *minimal* (with respect to a class of automata, say state-based deterministic Büchi automata) if for all automata $\mathcal{A}_2$ equivalent to $\mathcal{A}_1$, we have that $|\mathcal{A}_1| \leq |\mathcal{A}_2|$.

2.2 SD and HD Automata

Consider an automaton $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$. For a state $q \in Q$ of $\mathcal{A}$, we define $\mathcal{A}^q = \langle \Sigma, Q, \{q\}, \delta, \alpha \rangle$, as the automaton obtained from $\mathcal{A}$ by setting the set of initial states to be $\{q\}$. We say that two states $q, s \in Q$ are *equivalent*, denoted $q \sim_{\mathcal{A}} s$, if $L(\mathcal{A}^q) = L(\mathcal{A}^s)$. We say that q *is reachable* if there is a finite word $x \in \Sigma^*$ with $q \in \delta(Q_0, x)$, and say that q *is reachable from* s if q is reachable in $\mathcal{A}^s$. We say that $\mathcal{A}$ is *semantically deterministic* (SD, for short) if different nondeterministic choices lead to equivalent states. Thus, all initial states are equivalent, and for every state $q \in Q$ and letter $\sigma \in \Sigma$, all the σ-successors of q are equivalent. Formally, if $s, s' \in \delta(q, \sigma)$, then $s \sim_{\mathcal{A}} s'$. The following proposition, termed the *SDness property*, follows immediately from the definitions and implies that in SD automata, for all finite words x, all the states in $\delta(Q_0, x)$ are equivalent. Intuitively, it means that a run of an SD automaton can take also bad nondeterministic choices, as long as it does so only finitely many times.

▶ **Proposition 1.** *Consider an SD automaton $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$, and states $q, s \in Q$. If $q \sim_{\mathcal{A}} s$, then for every $\sigma \in \Sigma$, $q' \in \delta(q, \sigma)$, and $s' \in \delta(s, \sigma)$, we have that $q' \sim_{\mathcal{A}} s'$.*

An automaton $\mathcal{A}$ is *history deterministic* (*HD*, for short) if its nondeterminism can be resolved based on the past, thus on the prefix of the input word read so far. Formally, $\mathcal{A}$ is *HD* if there exists a *strategy* $f : \Sigma^* \to Q$ such that the following hold:

1. The strategy f is consistent with the transition function. That is, $f(\epsilon) \in Q_0$, and for every finite word $u \in \Sigma^*$ and letter $\sigma \in \Sigma$, we have that $f(u \cdot \sigma) \in \delta(f(u), \sigma)$.
2. Following f causes $\mathcal{A}$ to accept all the words in its language. That is, for every word $w = \sigma_1 \cdot \sigma_2 \cdots$, if $w \in L(\mathcal{A})$, then the run $f(\epsilon), f(w[1,1]), f(w[1,2]), \ldots$ is an accepting run of $\mathcal{A}$ on w.

We say that the strategy f *witnesses* $\mathcal{A}$'s HDness. Note that, by definition, we can assume that every SD and HD automaton $\mathcal{A}$ has a single initial state. Thus, we sometimes abuse notation and write $\mathcal{A}$ as $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, where q_0 is the single initial state of the SD (or HD) automaton $\mathcal{A}$.

For an automaton $\mathcal{A}$, we say that a state q of $\mathcal{A}$ is *HD*, if $\mathcal{A}^q$ is HD. Note that every deterministic automaton is HD. Also, while not all HD automata can be pruned to deterministic ones [7], removing of transitions that are not used by a strategy f that witnesses $\mathcal{A}$'s HDness does not reduce the language of $\mathcal{A}$ and results in an SD automaton. Moreover, since every state that is used by f is HD, the removal of non-HD states does not affect $\mathcal{A}$'s language nor its HDness. Accordingly, we have the following [20, 5].

▶ **Proposition 2.** *Every HD automaton $\mathcal{A}$ can be pruned to an equivalent SD automaton all whose states are HD.*

We use three-letter acronyms in $\{\mathrm{D}, \mathrm{N}\} \times \{\mathrm{B}, \mathrm{C}, \mathrm{W}, \mathrm{F}\} \times \{\mathrm{W}\}$ to denote the different automata classes. The first letter stands for the branching mode of the automaton (deterministic or nondeterministic); the second for the acceptance condition type (Büchi, co-Büchi, weak, or an automaton that runs over finite inputs); and the third indicates that we consider automata on words. For transition-based automata, we start the acronyms with the letter "t", and for HD or SD automata, we add an HD or SD prefix, respectively. For example, an HD-tNBW is a transition-based HD nondeterministic Büchi automaton, a DFW is a state-based deterministic automaton on finite words, and an SD-NWW is a state-based weak SD automaton.

3 Semantically Deterministic Büchi Automata

In this section we examine SD-tNBWs and SD-NBWs. Our results use the following definitions and constructions: For a language $R \subseteq \Sigma^*$ of finite words, we use ∞R to denote the language of infinite words that contain infinitely many disjoint infixes in R. Thus, $w \in \infty R$ iff $\epsilon \in R$ or there are infinitely many indices $i_1 \leq i'_1 < i_2 \leq i'_2 < \cdots$ such that $w[i_j, i'_j] \in R$, for all $j \geq 1$. For example, taking $\Sigma = \{a, b\}$, we have that $\infty\{ab\}$ is the language of words with infinitely many ab infixes, namely all words with infinitely many a's and infinitely many b's. We say that a finite word $x \in \Sigma^*$ is a *good prefix* for a language $R \subseteq \Sigma^*$ if for all finite words $y \in \Sigma^*$, we have that $x \cdot y \in R$. For example, while the language $(a + b)^* \cdot a$ does not have a good prefix, the word a is a good prefix for the language $a \cdot (a + b)^*$.

Theorem 3 below suggests that one can encode an NFW-recognizable language R in an SD-tNBW $\mathcal{A}$ for ∞R, and Theorem 4 suggests that one can decode an NFW for R from an SD-tNBW for $\infty(\$ \cdot R \cdot \$)$, where $\$ \notin \Sigma$. The blow-up in the sizes of the automata is constant, and both theorems play a major rule in the rest of this section.

▶ **Theorem 3.** *Given an NFW $\mathcal{N}$, one can obtain, in polynomial time, an SD-tNBW $\mathcal{A}$ such that $L(\mathcal{A}) = \infty L(\mathcal{N})$ and $|\mathcal{A}| = |\mathcal{N}|$.*

Proof. Let $\mathcal{N} = \langle \Sigma, Q, Q_0, \delta, F \rangle$. Then, $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta', \alpha \rangle$ is obtained from $\mathcal{N}$ by adding transitions to Q_0 from all states with all letters. A new σ-transition is in α if $Q_0 \cap F \neq \emptyset$ or when $\mathcal{N}$ could transit with σ to a state in F. Formally, for all $s \in Q$ and $\sigma \in \Sigma$, we have that $\delta'(s, \sigma) = \delta(s, \sigma) \cup Q_0$, and $\alpha = \{\langle s, \sigma, q \rangle : q \in Q_0 \text{ and } (Q_0 \cap F \neq \emptyset \text{ or } \delta(s, \sigma) \cap F \neq \emptyset)\}$.

It is easy to see that $|\mathcal{A}| = |\mathcal{N}|$. In order to prove that $\mathcal{A}$ is SD and $L(\mathcal{A}) = \infty L(\mathcal{N})$, we prove in the full version that for every state $q \in Q$, it holds that $L(\mathcal{A}^q) = \infty L(\mathcal{N})$. ◀

▶ **Theorem 4.** *Consider a language $R \subseteq \Sigma^*$ and a letter $\$ \notin \Sigma$. For every SD-tNBW $\mathcal{A}$ such that $L(\mathcal{A}) = \infty(\$ \cdot R \cdot \$)$, there exists an NFW $\mathcal{N}$ such that $L(\mathcal{N}) = R$ and $|\mathcal{N}| \leq |\mathcal{A}| + 1$. In addition, if R has no good prefixes, then $|\mathcal{N}| \leq |\mathcal{A}|$.*

Proof. If R is trivial, then one can choose $\mathcal{N}$ to be a one-state NFW. Assume that R is nontrivial. Let $\mathcal{A} = \langle \Sigma \cup \{\$\}, Q, q_0, \delta, \alpha \rangle$ be an SD-tNBW for $\infty(\$ \cdot R \cdot \$)$, W.l.o.g we assume that all the states of $\mathcal{A}$ are reachable. For a nonempty set of states $S \in 2^Q \setminus \emptyset$, we define the *universal $\overline{\alpha}$ language* of S as

$$L_{u\overline{\alpha}}(S) = \{w \in (\Sigma \cup \{\$\})^\omega : \text{for all } q \in S, \text{ all the runs of } \mathcal{A}^q \text{ on } w \text{ do not traverse } \alpha\}.$$

We say that S is *hopeful* when $(\$ \cdot \overline{R})^\omega \subseteq L_{u\overline{\alpha}}$. Note that S is hopeful iff for every state $q \in S$, it holds that $\{q\}$ is hopeful. Also, if S is hopeful, $x \in \Sigma^*$ is a finite word, and there is a run from S on $\$ \cdot x$ that traverses α, then $x \in R$. Then, we say that S is *good* when for all words $x \in \Sigma^*$, it holds that $x \in \overline{R}$ iff all the runs from S on $\$ \cdot x$ do not traverse α, and the set $\delta(S, \$ \cdot x)$ is hopeful. Note that as R is nontrivial, there exists a word x in $\overline{R}$, and thus by definition, all the \$-labled transitions going out from a good set S are in $\overline{\alpha}$.

Good sets in $\mathcal{A}$ characterize the language R, and as we argue below, their existence induces an NFW for R. In the full version, we prove that a good set exists. We show now that a good set in $\mathcal{A}$ induces an NFW $\mathcal{N}$ for R with the required properties. Let $S \in 2^Q \setminus \emptyset$ be a good set. We define the NFW $\mathcal{N} = \langle \Sigma, Q \cup \{q_{acc}\}, Q_0^S, \delta_S, F_S \rangle$, where $Q_0^S = \delta(S, \$)$, $F_S = \{q_{acc}\} \cup \{q \in Q : \text{the set } \{q\} \text{ is not hopeful}\}$, and the transition function δ_S is defined as follows. For every two states $q, s \in Q$ and letter $\sigma \in \Sigma$, it holds that $s \in \delta_S(q, \sigma)$ iff $\langle q, \sigma, s \rangle \in \overline{\alpha}$. Also, $q_{acc} \in \delta_S(q, \sigma)$ iff there is a σ-labeled α-transition going out from q in $\mathcal{A}$. Also, for all letters $\sigma \in \Sigma$, it holds that $\delta(q_{acc}, \sigma) = \{q_{acc}\}$; that is, q_{acc} is an accepting sink. Thus, $\mathcal{N}$ behaves as the states in $\delta(S, \$)$ as long as it reads $\overline{\alpha}$ transitions of $\mathcal{A}$, moves to the accepting sink q_{acc} whenever an α-transition is encountered, and accepts also whenever it reaches a state in Q that is not hopeful.

In the full version, we prove that $L(\mathcal{N}) = R$. Essentially, this follows from the fact that if we consider a word $x \in \Sigma^*$ such that all the runs from Q_0^S on x in $\mathcal{A}$ do not traverse α, then S being a good set implies that $x \in R$ iff $\delta(S, \$ \cdot x) \cap F_S \neq \emptyset$.

Since the state space of $\mathcal{N}$ is $Q \cup \{q_{acc}\}$, then $|\mathcal{N}| = |\mathcal{A}| + 1$. Moreover, as q_{acc} is an accepting sink, a word $x \in L(\mathcal{N})$ that has a run that ends in q_{acc} is a good prefix for $L(\mathcal{N})$. Hence, as $L(\mathcal{N}) = R$, if R has no good prefixes, then q_{acc} is not reachable in $\mathcal{N}$ and thus can be removed without affecting $\mathcal{N}$'s language. Thus, in this case, we get an NFW $\mathcal{N}$ for R whose size is at most $|\mathcal{A}|$. ◀

3.1 Succinctness and Complementation

In this section we study the succinctness of SD Büchi automata with respect to deterministic ones, and the blow-up involved in their complementation. We show that SD-tNBWs are exponentially more succinct than tDBWs, matching the known upper bound [3], and in fact, also from HD-tNBWs. We also prove an exponential lower bound for complementation. Similar results for SD-NBWs follow, as the transition between the two types of acceptance conditions is linear.

▶ **Theorem 5.** *There is a family $L_1, L_2, L_3, \ldots$ of languages such that for every $n \geq 1$, there is an SD-tNBW with $3n + 3$ states that recognizes L_n, yet every tDBW or HD-tNBW that recognizes L_n needs at least 2^n states.*

Proof. For $n \geq 1$, let $[n] = \{1, \ldots, n\}$, and let $\Sigma_n = \{1, \ldots, n, \$, \#\}$. We say that a word $z \in \Sigma_n^*$ is *good* if $z = \$ \cdot x \cdot \# \cdot i$, where $x \in [n]^+$ and i appears in x. Let $R_n \subseteq \Sigma_n^*$ be the language of all good words. We define $L_n = \infty R_n$. First, it is not hard to see that R_n can be recognized by an NFW $\mathcal{N}_n$ with $3n + 3$ states. Essentially, $\mathcal{N}_n$ guesses the last letter i in the input word and then checks that the guess is correct (see sketch in Figure 1). By Theorem 3, there is an SD-tNBW for L_n with $3n + 3$ states.

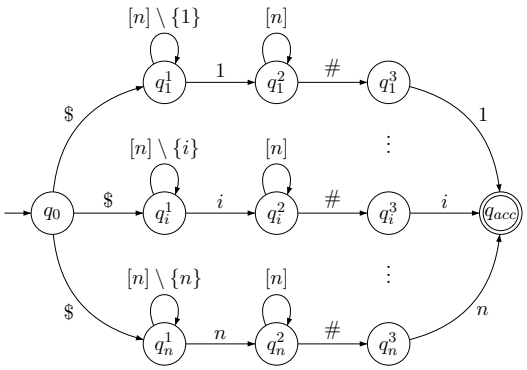

Figure 1 The NFW $\mathcal{N}_n$. Missing transitions lead to a rejecting-sink.

Before we prove that every tDBW or HD-tNBW that recognizes L_n needs at least 2^n states, let us note that it is already known that going from a DFW for a language $R \subseteq \Sigma^*$ to a tDBW for ∞R, may involve a blow-up of $2^{n-2-log_2(n)}$[24]. Thus, Theorem 3 implies an exponential gap between SD-tNBWs and tDBWs. Moreover, as HD-tNBWs are at most quadratically more succinct than tDBWs [20], the above can be extended to a $2^{\frac{n-2-log_2(n)}{2}}$ lower bound for the succinctness of SD-tNBWs with respect to HD-tNBWs. Our example here is tighter.

It is left to prove that every HD-tNBW that recognizes L_n needs at least 2^n states. Assume towards contradiction that $\mathcal{A} = \langle \Sigma_n, Q, q_0, \delta, \alpha \rangle$ is an HD-tNBW for L_n with $|Q| < 2^n$ states. In the full version, we iteratively define infinite sequences of finite words $x_1, x_2, x_3, \ldots$ and states $q_0, q_1, \ldots$ such that for all $k \geq 1$, the word x_k starts with \$, has no good infixes, and there is a run of the form $r_k = q_{k-1} \xrightarrow{x_k} q_k$ in $\mathcal{A}$ on x_k that traverses α. The challenging part in the construction is to make it valid also for HD (and not only deterministic) automata. For this, the definition of the words in the sequence is defined with respect to a strategy that attempts to witness the HDness of $\mathcal{A}$. To see why such sequences imply a contradiction, note that the concatenation of the runs $r_1, r_2, \ldots$ is an accepting run of $\mathcal{A}$ on the word $x = x_1 \cdot x_2 \cdots$. As $\mathcal{A}$ recognizes $L_n = \infty R_n$, it follows that $x \in \infty R_n$. On the other hand, x has no good infixes, and so $x \notin \infty R_n$. Indeed, if there is a good infix in x, then it must contain letters from different x_k's; in particular, it must contain at least two \$'s. ◀

▶ Remark 6. In order to get a slightly tighter bound, one can show that a minimal tDBW for L_n needs at least 2^{n+1} states and that the language L_n is not *HD-helpful*. That is, a minimal HD-tNBW for L_n is not smaller than a minimal tDBW for L_n, and so the 2^{n+1} bound holds also for a minimal HD-tNBW. The proof starts with a tDBW for L_n that has 2^{n+1} states, considers its complement tDCW, and shows that the application of the polynomial minimization algorithm of [1] on it, namely the algorithm that returns an equivalent minimal HD-tNCW, does result in a smaller automaton. The result then follows from the fact that a minimal HD-tNCW for a language is smaller than a minimal HD-tNBW for its complement [20]. The proof involves many notions and observations from [1] about minimal HD-tNCWs.

In the full version, we still describe a tDBW with 2^{n+1} states for L_n, and readers familiar with [1] can observe that the application of the HD-tNCW minimization algorithm on its dual tDCW does not make it smaller.

▶ **Theorem 7.** *There is a family $L_1, L_2, L_3, \ldots$ of languages such that for every $n \geq 1$, there is an SD-tNBW with $O(n)$ states that recognizes L_n, yet every SD-tNCW that recognizes $\overline{L_n}$ needs at least $2^{O(n)}$ states.*

Proof. Let $\Sigma = \{0, 1\}$. For $n \geq 1$, let $R_n = \{w : w \in 0 \cdot (0+1)^{n-1} \cdot 1 + 1 \cdot (0+1)^{n-1} \cdot 0\}$. It is easy to see that R_n can be recognized by an NFW $\mathcal{N}_n$ with $O(n)$ states. We define $L_n = \infty R_n$. First, by Theorem 3, there is an SD-tNBW with $O(n)$ states for L_n. In order to prove that an SD-tNCW for $\overline{L_n}$ needs at least $2^{O(n)}$ states, we prove that in fact every tNCW for $\overline{L_n}$ needs that many states. For this, note that $\overline{L_n}$ consists of all words w for which there is $u \in (0+1)^n$ such that $w \in (0+1)^* \cdot u^\omega$. Indeed, for such words w, the suffix u^ω contains no infix in R_n. Also, if a word contain only finitely many infixes in R_n, then it must have a suffix with no such infixes, namely a suffix of the form u^ω for some $u \in (0+1)^n$. Then, the proof that a tNCW for $\overline{L_n}$ needs exponentially many states is similar to the proof that an NFW for $\{u \cdot u : u \in (0+1)^n\}$ needs exponentially many states. Indeed, it has to remember the last n letters read. ◀

3.2 Decision Problems

We continue to decision problems about SD-tNBWs and SD-NBWs, and show that the exponential succinctness comes with a price: the complexity of all the problems we study coincides with the one known for tNBWs and NBWs. Accordingly, we only prove lower bounds for SD-tNBWs. Matching upper bounds follow from the known complexity for tNBWs, and same bounds for SD-NBWs follow from linear translations between SD-tNBWs and SD-NBWs. The problems we study are *language containment:* given two SD-tNBWs $\mathcal{A}_1$ and $\mathcal{A}_2$, decide whether $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$, *universality:* given an SD-tNBW $\mathcal{A}_1$, decide whether $L(\mathcal{A}_1) = \Sigma^\omega$, and *minimization:* given an SD-tNBW $\mathcal{A}_1$ and an integer $k \geq 1$, decide whether there is an SD-tNBW $\mathcal{A}_2$ such that $L(\mathcal{A}_1) = L(\mathcal{A}_2)$ and $|\mathcal{A}_2| \leq k$.

The exponential succinctness of SD automata motivates also the study of the *D-to-SD minimization problem*. Here, we are given a tDBW $\mathcal{A}_1$ and an integer $k \geq 1$, and we need to decide whether there is an SD-tNBW $\mathcal{A}_2$ such that $L(\mathcal{A}_1) = L(\mathcal{A}_2)$ and $|\mathcal{A}_2| \leq k$. For automata on finite words, the D-to-N minimization problem is known to be PSPACE-complete [19].

Note that a lower bound for universality implies a lower bound also for language containment. We still start with language containment, as it is much simpler.

▶ **Theorem 8.** *The language-containment problem for SD-tNBWs is PSPACE-hard.*

Proof. We describe a reduction from the universality problem for NFWs. Given an NFW $\mathcal{N}$ over Σ, let $\mathcal{N}'$ be an NFW over $\Sigma \cup \{\$\}$ such that $L(\mathcal{N}') = \$ \cdot L(\mathcal{N}) \cdot \$$. Now, let $\mathcal{A}_1$ be a 1-state tDBW over $\Sigma \cup \{\$\}$ such that $L(\mathcal{A}_1) = \infty\$$, and let $\mathcal{A}_2$ be the SD-tNBW obtained by applying the operation from Theorem 3 on $\mathcal{N}'$. Note that $L(\mathcal{A}_2) = \infty(\$ \cdot L(\mathcal{N}) \cdot \$)$ and $|\mathcal{A}_2| = |\mathcal{N}| + 3$.

We claim that $\mathcal{N}$ is universal iff $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$. First, if $L(\mathcal{N}) = \Sigma^*$, then $L(\mathcal{A}_2) = \infty(\$ \cdot \Sigma^* \cdot \$) = \infty\$$ and so $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$. Conversely, if there is a word $x \in \Sigma^* \setminus L(\mathcal{N})$, then the word $w = (\$ \cdot x)^\omega$ is in $L(\mathcal{A}_1) \setminus L(\mathcal{A}_2)$. Indeed, w has infinitely many \$'s, yet for every infix $\$ \cdot y \cdot \$$ of w, we have that $y \notin L(\mathcal{N})$, and so $w \notin L(\mathcal{A}_2)$. ◀

The proof in Theorem 8 uses $\infty\$$ as the "contained language". For the universality problem, where we cannot relay on hints from words in the contained language, we have to work much harder and generate such hints from runs of Turing machines. Specifically, we prove PSPACE hardness by a generic reduction from polynomial space Turing machines. Such reductions associate with a Turing machine T an automaton $\mathcal{A}$ that recognizes the language R of words that do not encode legal rejecting computations of T, and so $R = \Sigma^*$ iff the machine has no rejecting computations. The automaton $\mathcal{A}$ is nondeterministic, as it has to guess violations of attempts to encode legal accepting computations. In order to replace $\mathcal{A}$ by an SD automaton, we manipulate the Turing machine so that the language of the generated automaton is of the form ∞R, for which we can construct an SD-tNBW.

▶ **Theorem 9.** *The universality problem for SD-tNBWs is PSPACE-hard.*

Proof. We do a reduction from polynomial-space Turing machines. Given a Turing machine T with space complexity $s : \mathbb{N} \to \mathbb{N}$, we construct in time polynomial in $|T|$ and $s(0)$, an SD-tNBW $\mathcal{A}$ of size polynomial in T and $s(0)$, such that $\mathcal{A}$ is universal iff T accepts the empty tape[3]. Let $n_0 = s(0)$. Thus, each configuration in the computation of T on the empty tape uses at most n_0 cells. We assume that T halts from all configurations (that is, not just from these reachable from an initial configuration of T); Indeed, by adding a polynomial-space counter to T, one can transform a polynomial-space Turing machine that need not halt from all configurations to one that does halt. We also assume, without loss of generality, that once T reaches a final (accepting or rejecting) state, it erases the tape, moves with its reading head to the leftmost cell, and moves to the initial state. Thus, all computations of T are infinite and after visiting a final configuration for the first time, they eventually consists of repeating the same finite computation on the empty tape that uses at most n_0 tape cells.

We define $\mathcal{A}$ so that it accepts a word w iff (C1) no suffix of w is an encoding of a legal computation of T that uses at most n_0 tape cells, or (C2) w has infinitely many infixes that encode the accepting state of T.

It is not hard to see that T accepts the empty tape iff $\mathcal{A}$ is universal. Indeed, if T accepts the empty tape, and there is a word w that does not satisfy (C1), thus w has a suffix that is an encoding of a legal computation of T that uses at most n_0 cells, then the encoded computation eventually reaches a final configuration, from which it eventually repeats the accepting computation of T on the empty tape infinitely many times, and so w satisfies (C2). Conversely, If T rejects the empty tape, then the word w that encodes the computation of T on the empty tape does not satisfy (C1) nor (C2), and so $\mathcal{A}$ does not accept w.

Finally, the fact that T is a polynomial-space Turing machine enables us to define $\mathcal{A}$ with polynomially many states, as we detail in the full version. ◀

We continue to the minimization problem. Note that here, a PSPACE upper bound does not follow immediately from the known PSPACE upper bound for tNBWs, as the candidate automata need to be SD. Still, as SDness can be checked in PSPACE [3], a PSPACE upper bound follows. Also note that here, the case of SD-NBWs is easy, as a non-empty SD-NBW

[3] This is sufficient, as one can define a generic reduction from every language L in PSPACE as follows. Let T_L be a Turing machine that decides L in polynomial space $f(n)$. On input w for the reduction, the reduction considers the machine T_w that on every input, first erases the tape, writes w on its tape, and then runs as T_L on w. Then, the reduction outputs an automaton $\mathcal{A}$, such that T_w accepts the empty tape iff $\mathcal{A}$ is SD. Note that the space complexity of T_w is $s(n) = \max(n, f(|w|))$, and that w is in L iff T_w accepts the empty tape. Since $\mathcal{A}$ is constructed in time polynomial in $s(0) = f(|w|)$ and $|T_w| = \text{poly}(|w|)$, it follows that the reduction is polynomial in $|w|$.

is universal iff it has an equivalent SD-NBW with one state. For transition-based acceptance, the language of a single-state SD-tNBW need not be trivial, and so we have to examine the specific language used for the universality PSPACE-hardness proof (see proof in the full version):

▶ **Theorem 10.** *The minimization problem for SD-tNBWs is PSPACE-hard.*

As we show below, the minimization problem stays hard even when we start from a deterministic automaton:

▶ **Theorem 11.** *The D-to-SD minimization problem for Büchi automata is PSPACE-hard.*

Proof. We start with Büchi automata with transition-based acceptance, and describe a reduction from the D-to-N minimization problem for automata on finite words: given a DFW $\mathcal{A}_1$, and an integer $k \geq 1$, decide whether there is an NFW $\mathcal{A}_2$ such that $L(\mathcal{A}_1) = L(\mathcal{A}_2)$ and $|\mathcal{A}_2| \leq k$. In [19], the authors prove that the problem is PSPACE-hard, in fact PSPACE-hard already for DFWs that recognize a language that has no good prefixes.

Consider a language $R \subseteq \Sigma^*$. The reduction is based on a construction that turns an NFW for R into an SD-tNBW for $\infty(\$ \cdot R \cdot \$)$, for a letter $\$ \notin \Sigma$. Note that by applying the construction from Theorem 3 on the language $\$ \cdot R \cdot \$$, we can get an SD-tNBW for $\infty(\$ \cdot R \cdot \$)$. The construction there, however, does not preserve determinism. Therefore, we need a modified polynomial construction, which takes advantage of the \$'s. We describe the modified construction below.

Given an NFW $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, F \rangle$, and a letter $\$ \notin \Sigma$, we construct the tNBW $\mathcal{A}' = \langle \Sigma \cup \{\$\}, Q, Q_0, \delta', \alpha \rangle$, where for all states $q \in Q$ and letters $\sigma \in \Sigma$, we have that $\delta'(q, \sigma) = \delta(q, \sigma)$. Also, $\delta'(q, \$) = Q_0$, and $\alpha = \{\langle s, \sigma, q \rangle : q \in Q_0 \text{ and } s \in F\}$. Thus, $\mathcal{A}'$ is obtained from $\mathcal{A}$ by adding \$-transitions from all states to Q_0. A new transition is in α iff its source is an accepting state of $\mathcal{A}$. In the full version, we prove that $\mathcal{A}'$ is an SD-tNBW and $L(\mathcal{A}') = \infty(\$ \cdot L(\mathcal{A}) \cdot \$)$. Also, as we only added transitions labeled \$ to Q_0, it is easy to see that if $\mathcal{A}$ is deterministic, then so is $\mathcal{A}'$.

We now describe the reduction. Given a DFW $\mathcal{A}$ over Σ such that $L(\mathcal{A})$ has no good prefixes, and given an integer k, the reduction returns the polynomial tDBW $\mathcal{A}'$ and the integer k. We prove next that the reduction is correct. Thus, the DFW $\mathcal{A}$ has an equivalent NFW with at most k states iff the tDBW $\mathcal{A}'$ has an equivalent SD-tNBW with at most k states. For the first direction, if $\mathcal{B}$ is an NFW equivalent to $\mathcal{A}$ whose size is at most k, then by the above construction, it holds that $\mathcal{B}'$ is an SD-tNBW whose size is at most k, and $L(\mathcal{B}') = \infty(\$ \cdot L(\mathcal{B}) \cdot \$) = \infty(\$ \cdot L(\mathcal{A}) \cdot \$) = L(\mathcal{A}')$.

Conversely, if $\mathcal{B}'$ is an SD-tNBW for $\infty(\$ \cdot L(\mathcal{A}) \cdot \$)$ whose size is at most k, then as $L(\mathcal{A})$ has no good prefixes, we get by Theorem 4 that there is an NFW $\mathcal{B}$ for $L(\mathcal{A})$ whose size is at most k, and we are done.

Finally, since the transitions between automata with transition-based and state-based acceptance may involve a linear blow-up, here we need to be careful in extending the result to the state-based setting. In the full version, we prove that the arguments in Theorem 11 can be adapted to automata with state-based acceptance, thus PSPACE-completeness holds also for Büchi automata with state-based acceptance. ◀

4 Semantically Deterministic co-Büchi Automata

In this section we study SD co-Büchi automata. Here too, our results are based on constructions that involve encodings of NFWs by SD-tNCWs. Here, however, the constructions are more complicated, and we first need some definitions and notations.

Consider a tNCW $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$. We refer to the SCCs we get by removing $\mathcal{A}$'s α-transitions as the $\overline{\alpha}$-*components* of $\mathcal{A}$; that is, the $\overline{\alpha}$-*components* of $\mathcal{A}$ are the SCCs of the graph $G_{\mathcal{A}^{\overline{\alpha}}} = \langle Q, E^{\overline{\alpha}} \rangle$, where $\langle q, q' \rangle \in E^{\overline{\alpha}}$ iff there is a letter $\sigma \in \Sigma$ such that $\langle q, \sigma, q' \rangle \in \overline{\alpha}$. We say that $\mathcal{A}$ is *normal* if there are no $\overline{\alpha}$-transitions connecting different $\overline{\alpha}$-components. That is, for all states q and s of $\mathcal{A}$, if there is a path of $\overline{\alpha}$-transitions from q to s, then there is also a path of $\overline{\alpha}$-transitions from s to q. Note an accepting run of $\mathcal{A}$ eventually gets trapped in one of $\mathcal{A}$'s $\overline{\alpha}$-components. In particular, accepting runs in $\mathcal{A}$ traverse transitions that leave $\overline{\alpha}$-components only finitely often. Hence, we can add transitions among $\overline{\alpha}$-components to α without changing the language of the automaton. Accordingly, in the sequel we assume that given tNCWs are normal.

We proceed to encoding NFWs by SD-tNCWs. For a language $R \subseteq \Sigma^*$, we define the language $\bowtie_{\$}(R) \subseteq (\Sigma \cup \{\$\})^\omega$, by

$$\bowtie_{\$}(R) = \{w : \text{if } w \text{ has infinitely many } \$, \text{ then it has a suffix in } (\$R)^\omega\}.$$

Also, we say that a finite word $x \in \Sigma^*$ is a *bad infix* for R if for all words $w \in \Sigma^*$ that have x as an infix, it holds that $w \in \overline{R}$.

Note that for every language $R \subseteq \Sigma^*$, we have that $\bowtie_{\$}(R) = \overline{\infty(\$ \cdot \overline{R} \cdot \$)}$. Thus, $\bowtie_{\$}(R)$ complements $\infty(\$ \cdot \overline{R} \cdot \$)$. Yet, unlike tDCWs and tDBWs, which dualize each other, SD-tNBWs and SD-tNCWs are not dual. Hence, adjusting Theorems 3 and 4 to the co-Büchi setting, requires different, in fact more complicated, constructions.

▶ **Theorem 12.** *Given an NFW $\mathcal{N}$, one can obtain, in linear time, an SD-tNCW $\mathcal{A}$ such that $L(\mathcal{A}) = \bowtie_{\$}(L(\mathcal{N}))$ and $|\mathcal{A}| = |\mathcal{N}|$.*

Proof. Given $\mathcal{N} = \langle \Sigma, Q, Q_0, \delta, F \rangle$, we obtain $\mathcal{A}$ by adding \$-transitions from all states to Q_0. The new transitions are in α iff they leave a state in $Q \setminus F$. Formally, $\mathcal{A} = \langle \Sigma \cup \{\$\}, Q, Q_0, \delta', \alpha \rangle$, where for all $s \in Q$ and $\sigma \in \Sigma$, we have that $\delta'(s, \sigma) = \delta(s, \sigma)$, and $\delta'(s, \$) = Q_0$. Then, $\alpha = \{\langle s, \$, q \rangle : q \in Q_0 \text{ and } s \in Q \setminus F\}$. It is easy to see that $|\mathcal{A}| = |\mathcal{N}|$. In order to prove that $\mathcal{A}$ is SD and $L(\mathcal{A}) = \bowtie_{\$}(L(\mathcal{N}))$, we prove in the full version that for every state $q \in Q$, it holds that $L(\mathcal{A}^q) = \bowtie_{\$}(L(\mathcal{N}))$. Essentially, this follows from the fact that $\mathcal{A}$ can avoid traversing α when it reads an input of the form $x \cdot \$$, only when $x \in L(\mathcal{N})$. ◀

▶ **Theorem 13.** *Consider a language $R \subseteq \Sigma^*$ and a letter $\$ \notin \Sigma$. For every tNCW $\mathcal{A}$ such that $L(\mathcal{A}) = \bowtie_{\$}(R)$, there exists an NFW $\mathcal{N}$ such that $L(\mathcal{N}) = R$ and $|\mathcal{N}| \leq |\mathcal{A}| + 1$. In addition, if R has bad infixes, then $|\mathcal{N}| \leq |\mathcal{A}|$.*

Proof. Let $\mathcal{A} = \langle \Sigma \cup \{\$\}, Q, Q_0, \delta, \alpha \rangle$ be a tNCW for $\bowtie_{\$}(R)$. We assume that $\mathcal{A}$ is normal and all of its states are reachable. We define the NFW $\mathcal{N} = \langle \Sigma, Q \cup \{q_{rej}\}, Q_0^{\mathcal{N}}, \delta_{\mathcal{N}}, F_{\mathcal{N}} \rangle$, where

- $Q_0^{\mathcal{N}} = \{q \in Q : \text{there is a state } q' \text{ such that } \langle q', \$, q \rangle \in \overline{\alpha}\}$.
- $F_{\mathcal{N}} = \{q \in Q : \text{there is a state } q' \text{ such that } \langle q, \$, q' \rangle \in \overline{\alpha}\}$.
- The transition function $\delta_{\mathcal{N}}$ is defined as follows: for every two states $q, s \in Q$ and letter $\sigma \in \Sigma$, it holds that $s \in \delta_{\mathcal{N}}(q, \sigma)$ iff $\langle q, \sigma, s \rangle \in \overline{\alpha}$. Also, if $\{q\} \times \{\sigma\} \times \delta(q, \sigma) \subseteq \alpha$, then $\delta_{\mathcal{N}}(q, \sigma) = q_{rej}$. Finally, for all letters $\sigma \in \Sigma$, it holds that $\delta_{\mathcal{N}}(q_{rej}, \sigma) = q_{rej}$; that is, q_{rej} is a rejecting sink.

Thus, $\mathcal{N}$ tries to accept words $x \in \Sigma^*$ for which there is a run in $\mathcal{A}$ that does not traverse α on the word $\$ \cdot x \cdot \$$.

We prove that $L(\mathcal{N}) = R$. We first prove that $R \subseteq L(\mathcal{N})$. Consider a word $x \in R$, and let $r = r_0, r_1, r_2, \ldots$ be an accepting run of $\mathcal{A}$ on $(\$ \cdot x)^\omega$. As r is accepting, there are $i < j$ such that $r_i, r_{i+1}, \ldots, r_j$ is a run that does not traverse α on $\$ \cdot x \cdot \$$. By the definition of $\delta_{\mathcal{N}}$, $r_{i+1} \in Q_0^{\mathcal{N}}, r_{j-1} \in F_{\mathcal{N}}$, and so $r_{i+1}, r_{i+2}, \ldots, r_{j-1}$ is an accepting run of $\mathcal{N}$ on x.

We prove next that $L(\mathcal{N}) \subseteq R$. Consider a word $x = \sigma_1 \cdot \sigma_2 \cdots \sigma_n \in L(\mathcal{N})$ and let $r = r_0, r_1, \ldots r_n$ be an accepting run of $\mathcal{N}$ on x. As r ends an accepting state of $\mathcal{N}$, then it does not visit q_{rej}. Hence, the definition of $\delta_{\mathcal{N}}$ implies that r is a run that does not traverse α in $\mathcal{A}$. Also, as $r_0 \in Q_0^{\mathcal{N}}$ and $r_n \in F_{\mathcal{N}}$, there are states $q_1, q_2 \in Q$ such that $\langle q_1, \$, r_0\rangle, \langle r_n, \$, q_2\rangle \in \overline{\alpha}$. Hence, $q_1, r_0, r_1, \ldots, r_n, q_2$ is a run that does not traverse α in $\mathcal{A}$ on the word $\$ \cdot x \cdot \$$. As $\mathcal{A}$ is normal, there is a word $y \in (\Sigma \cup \{\$\})^*$ such that there is a run that does not traverse α of $\mathcal{A}^{q_2}$ on y that reaches q_1. Therefore, $(\$ \cdot x \cdot \$ \cdot y)^\omega \in L(\mathcal{A}^{q_1})$. As all the states of $\mathcal{A}$ are reachable, it follows that there is a word $z \in (\Sigma \cup \{\$\})^*$ such that $q_1 \in \delta(Q_0, z)$, and thus $z \cdot (\$ \cdot x \cdot \$ \cdot y)^\omega \in L(\mathcal{A})$. Hence, as $L(\mathcal{A}) = \bowtie_{\$}(R)$, we get that $x \in R$.

Since the state space of $\mathcal{N}$ is $Q \cup \{q_{rej}\}$, we have that $|\mathcal{N}| = |\mathcal{A}| + 1$. Moreover, if R has a bad infix x, then $x \in \overline{R}$. Hence, if we consider the word x^ω, then as it has no \$'s, it is accepted by $\mathcal{A}$. Hence, there is a state $q \in Q$ such that $\mathcal{A}^q$ has a run on x^ω that does not leave the $\overline{\alpha}$-component of q. As we detail in the full version, the fact that x is a bad infix of R implies that the $\overline{\alpha}$-component of q does not contain \$-transitions. Hence, since the only states in Q that are reachable in $\mathcal{N}$ lie in $\overline{\alpha}$-components that contain a \$-transition, we can remove the state q_{rej} from $\mathcal{N}$ and make q a rejecting sink instead. Hence, in this case, we have that $|\mathcal{N}| \leq |\mathcal{A}|$, and we are done. ◀

4.1 Succinctness and Complementation

In this section we study the succinctness of SD co-Büchi automata with respect to deterministic ones, and the blow-up involved in their complementation. Recall that unlike Büchi automata, for co-Büchi automata, HD automata are exponentially more succinct than deterministic ones. Accordingly, our results also refer to the succinctness of SD automata with respect to HD ones:

▶ **Theorem 14.** *There is a family $L_1, L_2, L_3, \ldots$ of languages such that for every $n \geq 1$, there is an SD-tNCW with $3n + 3$ states that recognizes L_n, yet every tDCW or HD-tNCW that recognizes L_n needs at least 2^n states.*

Proof. For $n \geq 1$, consider the alphabet $\Sigma_n = [n] \cup \{\#\}$, and the language $R_n = \{x \cdot \# \cdot i : x \in [n]^+ \text{ and } i \text{ appears in } x\} \subseteq \Sigma_n^*$. We define $L_n = \bowtie_{\$}(R_n)$. Recall that a word over $z \in (\Sigma_n \cup \{\$\})^*$ is good if $z = \$ \cdot x \cdot \# \cdot i$, where $x \in [n]^+$ and i appears in x, and note that L_n consists of exactly the words with finitely many \$'s, or that have a suffix that is a concatenation of good words. First, it is not hard to see that R_n can be recognized by an NFW $\mathcal{N}_n$ with $3n + 3$ states, for example, a candidate for $\mathcal{N}_n$ can be obtained from the NFW in Figure 1 by removing the \$-transitions from q_0, and letting q_0 guess to behave as any state in $\bigcup_{i \in [n]} \{q_i^1\}$. By Theorem 12, there is an SD-tNCW for L_n with $3n + 3$ states.

In order to show that an HD-tNCW for L_n needs at least 2^n states, we rely on properties of HD-tNCWs [20, 1] and argue that (1) an HD-tNCW for L_n has a state q such that $L_{\overline{\alpha}}(q) = \{w : \text{there is a run from } q \text{ on } w \text{ that does not traverse } \alpha\}$ is such that $(\$ \cdot R_n)^\omega \subseteq L_{\overline{\alpha}}(q)$, and (2) the $\overline{\alpha}$-component of q is of size at least 2^n. As we detail in the full version, the first claim follows from the fact that HD-tNCWs can be assumed to be $\overline{\alpha}$ *deterministic*, thus all their $\bar{\alpha}$-transitions are deterministic [20, 1]. The second claim follows from the fact that the $\overline{\alpha}$-component of q should detect good subwords, and should remember for this subsets of $[n]$. ◀

▶ Remark 15. As has been the case with Büchi automata, here too, an analysis of the application of the HD-tNCW minimization algorithm on a tDCW for L_n leads to a slightly tighter bound. Specifically, in the full version, we describe a tDCW for L_n with $2^{n+1} + 1$ states, such that the application of the HD-tNCW minimization algorithm on it does not make it smaller.

▶ **Theorem 16.** *There is a family $L_1, L_2, L_3, \ldots$ of languages such that for every $n \geq 1$, there is an SD-tNCW with $O(n)$ states that recognizes L_n, yet every SD-tNBW that recognizes $\overline{L_n}$ needs at least $2^{O(n)}$ states.*

Proof. Let $\Sigma = \{0, 1\}$, and let $R_n = \{w : w \in (0+1)^* \cdot (0 \cdot (0+1)^{n-1} \cdot 1 + 1 \cdot (0+1)^{n-1} \cdot 0) \cdot (0+1)^*\}$. Thus, R_n is the language of all words that contain two letters that are at distance n and are different. We define $L_n = {\bowtie_\$}(R_n)$.

It is easy to see that R_n can be recognized by an NFW with $O(n)$ states. Hence, by Theorem 12, there is an SD-tNCW with $O(n)$ states for L_n. We prove next that every SD-tNBW for $\overline{L_n}$ needs at least $2^{O(n)}$ states. For this, note that $\overline{L_n}$ consists of all words w such that w contains infinitely many \$'s yet has no suffix in $(\$R_n)^\omega$. Thus, w contains infinitely many infixes in $\$\overline{R_n}\$$. Therefore, $\overline{L_n} = \infty(\$ \cdot \overline{R_n} \cdot \$)$. It is not hard to prove that an NFW for $\overline{R_n}$ needs at least $2^{O(n)}$ states. Then, by Theorem 4, this bound is carried over to a $2^{O(n)}$ lower bound on an SD-tNBW for $\overline{L_n}$. ◀

4.2 Decision Problems

We continue to decision problems for SD-tNCWs and SD-NCWs. As in Section 3.2, we state only the lower bounds, and for SD-tNCWs. Matching upper bounds and similar results for SD-NCWs follow from the known upper bounds for tNCWs and the known linear translations between state-based and transition-based automata.

We start with language-containment and universality.

▶ **Theorem 17.** *The language-containment and universality problems for SD-tNCWs are PSPACE-hard.*

Proof. We describe a reduction from universality of NFWs to universality of SD-tNCWs. Given an NFW $\mathcal{N}$ over Σ, the reduction returns the SD-tNCW $\mathcal{A}$ over $\Sigma \cup \{\$\}$ that is obtained by applying the operation from Theorem 12 on $\mathcal{N}$. Thus, $L(\mathcal{A}) = {\bowtie_\$}(L(\mathcal{N}))$.

First, if $L(\mathcal{N}) = \Sigma^*$, then $(\Sigma \cup \{\$\})^* \cdot (\$L(\mathcal{N}))^\omega = \infty\$$, and so $L(\mathcal{A}) = (\Sigma \cup \{\$\})^\omega$. Conversely, if there is a word $x \in \Sigma^* \setminus L(\mathcal{N})$, then the word $w = (\$x)^\omega$ is not in $L(\mathcal{A})$. Indeed, w has infinitely many \$'s, yet it has a word not in $L(\mathcal{N})$ between every two consecutive \$'s. ◀

We continue to the minimization problem. Note that while minimization is PSPACE-complete for NCWs, it is NP-complete for DCWs and in PTIME for HD-tNCWs. Thus, our PSPACE lower bound suggests a significant difference between SD and HD automata. Note also that, as has been the case with Büchi automata, the case of SD-NCWs is easy, as a non-empty SD-NCW is universal iff it has an equivalent SD-NCW with one state, which is not the case for SD-tNCWs (see proof in the full version):

▶ **Theorem 18.** *The minimization problem for SD-tNCWs is PSPACE-hard.*

We continue to the D-to-SD minimization problem, showing it stays PSPACE-hard.

▶ **Theorem 19.** *The D-to-SD minimization problem for co-Büchi automata is PSPACE-hard.*

Proof. We reduce from the D-to-N minimization problem for automata on finite words, which is already PSPACE-hard for languages that have bad infixes[19]. We start with co-Büchi automata with transition-based acceptance.

Consider the construction from Theorem 12. Recall it takes an NFW $\mathcal{A}$ as input, returns an SD-tNCW $\mathcal{A}'$ of the same size for ${\bowtie_\$}(L(\mathcal{A}))$, and preserves determinism.

Consider a DFW $\mathcal{A}$ over Σ such that $L(\mathcal{A})$ has a bad infix, and an integer k. The reduction returns the SD-tNCW $\mathcal{A}'$ constructed from $\mathcal{A}$ in Theorem 12, and the integer k. Recall that $L(\mathcal{A}') = \bowtie_{\$}(L(\mathcal{A}))$, and that the construction in Theorem 12 preserves determinism. Thus, the automaton $\mathcal{A}'$ is really a tDCW.

We prove next that the reduction is correct. That is, the DFW $\mathcal{A}$ has an equivalent NFW with at most k states iff the tDCW $\mathcal{A}'$ has an equivalent SD-tNCW with at most k states. For the first direction, if $\mathcal{B}$ is an NFW equivalent to $\mathcal{A}$ whose size is at most k, then, by applying to it the construction from Theorem 12, we get an SD-tNCW $\mathcal{B}'$ whose size is at most k, and $L(\mathcal{B}') = \bowtie_{\$}(L(\mathcal{B})) = \bowtie_{\$}(L(\mathcal{A})) = L(\mathcal{A}')$.

Conversely, if $\mathcal{B}'$ is an SD-tNCW for $\bowtie_{\$}(L(\mathcal{A}))$ whose size is at most k, then as $L(\mathcal{A})$ has bad infixes, we get by Theorem 13 that there is an NFW $\mathcal{B}$ for $L(\mathcal{A})$ whose size is at most k, and we are done.

In the full version, we extend the proof to co-Büchi automata with state-based acceptance. ◀

5 Semantically Deterministic Weak Automata

By [3], SD-NWWs need not be DBP or even HD. For completeness we describe here the example from [3], as it highlights the challenges in SD-NWW determinization. Consider the automaton $\mathcal{A}$ in Figure 2.

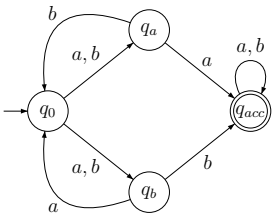

Figure 2 An SD-NWW that is not DBP.

It is easy to see that $\mathcal{A}$ is weak, and all its states are universal, and so it is SD. On the other hand, $\mathcal{A}$ is not HD as every strategy has a word with which it does not reach q_{acc} – a word that forces every visit in q_a and q_b to be followed by a visit in q_0. Below we show that despite not being DBP, SD-NWWs can be determinized in polynomial time. Essentially, our proof is based on redirecting transitions of the SD-NWW to deep components in the automaton. In our example, note that while the SD-NWW $\mathcal{A}$ is not HD, it has a deterministic state q_{acc} that recognizes $L(\mathcal{A})$.

Consider an SD-NWW $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$. We denote the set of $\mathcal{A}$'s SCCs by $\mathcal{C}(\mathcal{A})$, and the SCC containing a state q by $C(q)$. Let $C_1 \leq C_2 \leq \cdots \leq C_m$ be a total order on the SCCs of $\mathcal{A}$, extending the partial order induced by δ. That is, if $q' \in \delta(q, \sigma)$, for some letter σ, then $C(q) \leq C(q')$. When $C \leq C'$, we say that C' is *deeper than* C. Thus, states along runs of a weak automaton proceed from SCCs to deeper ones, and eventually get stuck in some SCC.

If we had an algorithm that checks language equivalence between states in $\mathcal{A}$ in polynomial time, we could have a polynomial determinization algorithm that defines the σ-successor of a state as the deepest state among all states equivalent to its σ-successors (since $\mathcal{A}$ is SD, all these successors agree on their language). Since we still do not have such an algorithm (in fact, it would follow from our construction), we approximate language equivalence by an equivalence that follows from the semantic determinism of $\mathcal{A}$.

For two states $s_1, s_2 \in Q$, we say that s_1 and s_2 are *δ-close*, if there is a state $q \in Q$ and a word $w \in \Sigma^*$ such that $s_1, s_2 \in \delta(q, w)$. Note that the δ-close relation refines equivalence, yet the converse does not hold. Indeed, the SDness property implies that $s_1 \sim_{\mathcal{A}} s_2$ for all δ-close states s_1 and s_2.

▶ **Lemma 20.** *If s_1 and s_2 are δ-close, then there is a state q and word w of length at most $|Q|^2$ such that $s_1, s_2 \in \delta(q, w)$.*

In order to calculate the δ-close relation in polynomial time, we define a sequence $H_0, H_1, H_2, \ldots \subseteq Q \times Q$ of relations, where $\langle s_1, s_2 \rangle \in H_i$ iff there is a state q and word w of length at most i such that $s_1, s_2 \in \delta(q, w)$. By Lemma 20 (see proof in the full version), we are guaranteed to reach a fixed point after at most $|Q|^2$ iterations. The relations H_i are defined as follows.

- $H_0 = \{\langle q, q \rangle : q \in Q\}$.
- For $i \geq 0$, we define $H_{i+1} = H_i \cup \{\langle s_1, s_2 \rangle :$ there is $\langle q_1, q_2 \rangle \in H_i$ and letter $\sigma \in \Sigma$ such that $s_1 \in \delta(q_1, \sigma)$ and $s_2 \in \delta(q_2, \sigma)\}$.

Let $j \geq 0$ be such that $H_{j+1} = H_j$. It is not hard to see that H_j is the δ-close relation. While the δ-close relation is reflexive and symmetric, it is not transitive. Now, let $H \subseteq Q \times Q$ be the closure of H_j under transitivity. That is, $\langle s_1, s_2 \rangle \in H$ iff there is $k \geq 2$ and states $q_1, q_2, \ldots, q_k$ such that $q_1 = s_1$, $q_k = s_2$ and $\langle q_i, q_{i+1} \rangle \in H_j$ for all $1 \leq i < k$.

The following lemma implies that H propagates to successor states (see proof in the full version):

▶ **Lemma 21.** *If $H(s, s')$, then for every letter $\sigma \in \Sigma$ and states $q \in \delta(s, \sigma)$ and $q' \in \delta(s', \sigma)$, we have that $H(q, q')$.*

It is easy to see that H is an equivalence relation. Let $\mathcal{P} = \{P_1, \ldots, P_k\}$ be the set of the equivalence classes of H. For each equivalence class $P \in \mathcal{P}$, we fix the *representative of* P as some state in $P \cap C$, where $C \in \mathcal{C}(\mathcal{A})$ is the deepest SCC that intersects P. Let $p_1, \ldots, p_k$ be the representatives of the sets in $\mathcal{P}$. For a state $q \in Q$, let $\widetilde{q}$ denote the representative of the set $P \in \mathcal{P}$ with $q \in P$. Note that as H refines $\sim_{\mathcal{A}}$, we have that $q \sim_{\mathcal{A}} \widetilde{q}$.

▶ **Theorem 22.** *Given an SD-NWW $\mathcal{A}$ with state space Q, we can construct, in polynomial time, an equivalent DWW $\mathcal{D}$ with state space Q', for $Q' \subseteq Q$.*

Proof. Given $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, we define $\mathcal{D} = \langle \Sigma, Q', q'_0, \delta', \alpha \cap Q' \rangle$, where

- $Q' = \{p_1, \ldots, p_k\}$. Note that indeed $Q' \subseteq Q$.
- $q'_0 = \widetilde{q_0}$.
- For $p \in Q'$ and $\sigma \in \Sigma$, we define $\delta'(p, \sigma) = \tilde{q}$, for some $q \in \delta(p, \sigma)$. Note that all the states in $\delta(p, \sigma)$ are δ-close, and thus belong to the same set in $\mathcal{P}$. Hence, the choice of q is not important, as $\delta'(p, \sigma)$ is the representative of this set.

We prove that $\mathcal{D}$ is a DWW equivalent to $\mathcal{A}$. First, in order to see that $\mathcal{D}$ is weak, consider states p, p' such that $p' \in \delta'(p, \sigma)$. Thus, p' is the representative of a state $q \in \delta(p, \sigma)$. As $\mathcal{A}$ is weak, we have that $C(p) \leq C(q)$. As $p' = \tilde{q}$, we have that $C(q) \leq C(p')$. Hence $C(p) \leq C(p')$, and we are done.

We continue and prove that $L(\mathcal{A}) = L(\mathcal{D})$. We first prove that $L(\mathcal{A}) \subseteq L(\mathcal{D})$. Consider a word w and assume that $w \in L(\mathcal{A})$. Let $r = q_0, q_1, q_2, \ldots$ be an accepting run of $\mathcal{A}$ on w, and let $r' = s_0, s_1, s_2, \ldots$ be the run of $\mathcal{D}$ on w. If r' is accepting, we are done. Otherwise, namely if r' is rejecting, we point to a contradiction. Let $j \geq 0$ be the index in which r'

visits only states in $sinf(r')$. Note that all the states $s_j, s_{j+1}, s_{j+2}, \ldots$ belong to some SCC C of $\mathcal{A}$. Since we assume that r' is rejecting, and acceptance in $\mathcal{D}$ is inherited from $\mathcal{A}$, we get that C is a rejecting SCC of $\mathcal{A}$. We claim that all the runs of $\mathcal{A}^{s_j}$ on the suffix $w[j+1,\infty]$ of w are stuck in C. Thus, $w[j+1,\infty] \notin L(\mathcal{A}^{s_j})$. On the other hand, we claim that for all $i \geq 0$, we have that $q_i \sim_{\mathcal{A}} s_i$. Then, however, $w[j+1,\infty] \notin L(\mathcal{A}^{q_j})$, contradicting the fact that r is accepting:

The easy part is to prove that for all $i \geq 0$, we have that $q_i \sim_{\mathcal{A}} s_i$. Indeed, it follows from the fact that for all $i \geq 0$, we have that $H(q_i, s_i)$ and the fact that H refines $\sim_{\mathcal{A}}$. The proof proceeds by an induction on i. First, as for every state $q \in Q$, we have that $H(q, \widetilde{q})$, then $H(q_0, q'_0)$, and so the claim follows from s_0 being q'_0. Assume now that $H(q_i, s_i)$. Recall that s_{i+1} is $\widetilde{q}$ for some $q \in \delta(s_i, w[i+1])$. Also, $q_{i+1} \in \delta(q_i, w[i+1])$. Then, by Lemma 21, we have that $H(q_{i+1}, q)$. Since, in addition, we have that $H(q, \widetilde{q})$, then the transitivity of H implies that $H(q_{i+1}, s_{i+1})$, and we are done.

We proceed to the other part, namely, proving that all the runs of $\mathcal{A}^{s_j}$ on the suffix $w[j+1,\infty]$ of w are stuck in C. Let $u_{j+1}, u_{j+2}, \ldots$ be such that for all $i \geq 1$, it holds that $u_{j+i} \in \delta(s_{j+i-1}, w[j+i])$ and $H(u_{j+i}, s_{j+i})$. Note that such states exist, as $s_{j+i-1} \xrightarrow{w[j+i]} s_{j+i}$ is a transition of $\mathcal{D}$ and so $s_{j+i} = \widetilde{q}$ for all $q \in \delta(s_{j+i-1}, w[j+i])$. Consider a run $r'' = v_0, v_1, v_2, \ldots$ of $\mathcal{A}^{s_j}$ on $w[j+1,\infty]$. Note that $v_0 = s_j$, and so $H(v_0, s_j)$. Therefore, by Lemma 21, we have that $H(v_1, u_{j+1})$, implying $H(v_1, s_{j+1})$. By repeated applications of Lemma 21, we get that $H(v_i, u_{j+i})$, implying $H(v_i, s_{j+i})$, for all $i \geq 1$. Assume now by way of contradiction that the run r'' leaves the SCC C, and so there is $i \geq 1$ such that $C(v_i) \not\leq C(s_{j+i})$. As $H(v_i, s_{j+i})$, the states v_i and s_{j+i} are in the same equivalence class $P \in \mathcal{P}$. Then, the definition of Q' implies that $C(s_{j+i}) \geq C(v_i)$, and we have reached a contradiction.

It is left to prove that $L(\mathcal{D}) \subseteq L(\mathcal{A})$. Consider a word $w \in L(\mathcal{D})$. Let $r' = s_0, s_1, s_2, \ldots$ be the run of $\mathcal{D}$ on w, and let $j \geq 0$ be the index in which r' visits only states in $sinf(r')$; in particular, as argued above, $sinf(r')$ is included in some SCC C of $\mathcal{A}$. Thus, all the states $s_j, s_{j+1}, s_{j+2}, \ldots$ are in C. Since $w \in L(\mathcal{D})$, then r' is accepting, and so C is an accepting SCC of $\mathcal{A}$. As in the previous direction, it holds that s_j is $\mathcal{A}$-equivalent to all the states in $\delta(q_0, w[1,j])$. Hence, to conclude that $w \in L(\mathcal{A})$, we show that there is an accepting run of $\mathcal{A}^{s_j}$ on $w[j+1,\infty]$. Assume by way of contradiction that all the runs of $\mathcal{A}^{s_j}$ on $w[j+1,\infty]$ are rejecting. In particular, all these runs leave the SCC C. As in the previous direction, this contradicts the choice of the states $s_j, s_{j+1}, s_{j+2}, \ldots$ as representatives of equivalence classes in $\mathcal{P}$. ◀

Since DWWs can be complemented by dualization (that is, by switching α and $\bar{\alpha}$), Theorem 22 implies the following.

▶ **Theorem 23.** *Given an SD-NWW $\mathcal{A}$ with n states, we can construct, in polynomial time, an SD-NWW (in fact, a DWW) that complements $\mathcal{A}$.*

Since the language-containment problem for DWW can be solved in NLOGSPACE, and minimization for DWW is similar to minimization of DFWs and can be solved in polynomial time [31], we have the following.

▶ **Theorem 24.** *The language-containment, universality, and minimality problems for SD-NWWs can be solved in polynomial time.*

References

1 B. Abu Radi and O. Kupferman. Minimizing GFG transition-based automata. In *Proc. 46th Int. Colloq. on Automata, Languages, and Programming*, volume 132 of *LIPIcs*, pages 100:1–100:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.

2 B. Abu Radi and O. Kupferman. On semantically-deterministic automata. *CoRR*, abs/2305.15489, 2023. `arXiv:2305.15489`.

3 B. Abu Radi, O. Kupferman, and O. Leshkowitz. A hierarchy of nondeterminism. In *46th Int. Symp. on Mathematical Foundations of Computer Science*, volume 202 of *LIPIcs*, pages 85:1–85:21, 2021.

4 B. Aminof, O. Kupferman, and R. Lampert. Reasoning about online algorithms with weighted automata. *ACM Transactions on Algorithms*, 6(2), 2010.

5 M. Bagnol and D. Kuperberg. Büchi good-for-games automata are efficiently recognizable. In *Proc. 38th Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume 122 of *LIPIcs*, pages 16:1–16:14, 2018.

6 B. Boigelot, S. Jodogne, and P. Wolper. On the use of weak automata for deciding linear arithmetic with integer and real variables. In *Proc. Int. Joint Conf. on Automated Reasoning*, volume 2083 of *Lecture Notes in Computer Science*, pages 611–625. Springer, 2001.

7 U. Boker, D. Kuperberg, O. Kupferman, and M. Skrzypczak. Nondeterminism in the presence of a diverse or unknown future. In *Proc. 40th Int. Colloq. on Automata, Languages, and Programming*, volume 7966 of *Lecture Notes in Computer Science*, pages 89–100, 2013.

8 U. Boker and K. Lehtinen. When a little nondeterminism goes a long way: an introduction to history-determinism. *ACM SIGLOG News*, 10(1):177–196, 2023.

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

10 Olivier Carton and Max Michel. Unambiguous büchi automata. *Theoretical Computer Science*, 297(1):37–81, 2003.

11 T. Colcombet. The theory of stabilisation monoids and regular cost functions. In *Proc. 36th Int. Colloq. on Automata, Languages, and Programming*, volume 5556 of *Lecture Notes in Computer Science*, pages 139–150. Springer, 2009.

12 T. Colcombet, K. Quaas, and M. Skrzypczak. Unambiguity in automata theory (dagstuhl seminar 21452). *Dagstuhl Reports*, 11(10):57–71, 2021.

13 A. Duret-Lutz, A. Lewkowicz, A. Fauchille, Th. Michaud, E. Renault, and L. Xu. Spot 2.0 — a framework for LTL and ω-automata manipulation. In *14th Int. Symp. on Automated Technology for Verification and Analysis*, volume 9938 of *Lecture Notes in Computer Science*, pages 122–129. Springer, 2016.

14 P. Gastin and D. Oddoux. Fast LTL to Büchi automata translation. In *Proc. 13th Int. Conf. on Computer Aided Verification*, volume 2102 of *Lecture Notes in Computer Science*, pages 53–65. Springer, 2001.

15 D. Giannakopoulou and F. Lerda. From states to transitions: Improving translation of LTL formulae to Büchi automata. In *Proc. 22nd International Conference on Formal Techniques for Networked and Distributed Systems*, volume 2529 of *Lecture Notes in Computer Science*, pages 308–326. Springer, 2002.

16 E.M. Hahn, M. Perez, S. Schewe, F. Somenzi, A. Trivedi, and D. Wojtczak. Good-for-MDPs automata for probabilistic analysis and reinforcement learning. In *Proc. 26th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, volume 12078 of *Lecture Notes in Computer Science*, pages 306–323. Springer, 2020.

17 T.A. Henzinger, O. Kupferman, and S. Rajamani. Fair simulation. *Information and Computation*, 173(1):64–81, 2002.

18 T.A. Henzinger and N. Piterman. Solving games without determinization. In *Proc. 15th Annual Conf. of the European Association for Computer Science Logic*, volume 4207 of *Lecture Notes in Computer Science*, pages 394–410. Springer, 2006.

19 T. Jiang and B. Ravikumar. Minimal NFA problems are hard. *SIAM Journal on Computing*, 22(6):1117–1141, 1993.

20 D. Kuperberg and M. Skrzypczak. On determinisation of good-for-games automata. In *Proc. 42nd Int. Colloq. on Automata, Languages, and Programming*, pages 299–310, 2015.

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

22 O. Kupferman. Automata theory and model checking. In *Handbook of Model Checking*, pages 107–151. Springer, 2018.

23 O. Kupferman. Using the past for resolving the future. *Frontiers in Computer Science*, 4, 2023.

24 O. Kupferman and O. Leshkowitz. On repetition languages. In *45th Int. Symp. on Mathematical Foundations of Computer Science*, Leibniz International Proceedings in Informatics (LIPIcs), 2020.

25 O. Kupferman, S. Safra, and M.Y. Vardi. Relating word and tree automata. *Ann. Pure Appl. Logic*, 138(1-3):126–146, 2006.

26 O. Kupferman and M.Y. Vardi. Safraless decision procedures. In *Proc. 46th IEEE Symp. on Foundations of Computer Science*, pages 531–540, 2005.

27 R.P. Kurshan. Complementing deterministic Büchi automata in polynomial time. *Journal of Computer and Systems Science*, 35:59–71, 1987.

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

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

30 W. Li, Sh. Kan, and Z. Huang. A better translation from LTL to transition-based generalized Büchi automata. *IEEE Access*, 5:27081–27090, 2017.

31 C. Löding. Efficient minimization of deterministic weak ω-automata. *Information Processing Letters*, 79(3):105–109, 2001.

32 S. Miyano and T. Hayashi. Alternating finite automata on ω-words. *Theoretical Computer Science*, 32:321–330, 1984.

33 G. Morgenstern. Expressiveness results at the bottom of the ω-regular hierarchy. M.Sc. Thesis, The Hebrew University, 2003.

34 D.E. Muller, A. Saoudi, and P. E. Schupp. Weak alternating automata give a simple explanation of why most temporal and dynamic logics are decidable in exponential time. In *Proc. 3rd IEEE Symp. on Logic in Computer Science*, pages 422–427, 1988.

35 M.O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:115–125, 1959.

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

37 S. Schewe. Beyond Hyper-Minimisation—Minimising DBAs and DPAs is NP-Complete. In *Proc. 30th Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume 8 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 400–411, 2010.

38 S. Schewe. Minimising good-for-games automata is NP-complete. In *Proc. 40th Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume 182 of *LIPIcs*, pages 56:1–56:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.

39 S. Schewe, Q. Tang, and T. Zhanabekova. Deciding what is good-for-MDPs. *CoRR*, abs/2202.07629, 2022. `arXiv:2202.07629`.

40 S. Sickert, J. Esparza, S. Jaax, and J. Křetínský. Limit-deterministic Büchi automata for linear temporal logic. In *Proc. 28th Int. Conf. on Computer Aided Verification*, volume 9780 of *Lecture Notes in Computer Science*, pages 312–332. Springer, 2016.

41 M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.