Mechanizing the Powerset Construction for Restricted Classes of ω-Automata*

Christian Dax[1], Jochen Eisinger[2], and Felix Klaedtke[1]

[1] ETH Zurich, Switzerland
[2] Albert-Ludwigs-Universität Freiburg, Germany

Abstract. Automata over infinite words provide a powerful framework, which we can use to solve various decision problems. However, the automatized reasoning with restricted classes of automata over infinite words is often simpler and more efficient. For instance, weak deterministic Büchi automata, which recognize the ω-regular languages in the Borel class $F_\sigma \cap G_\delta$, can be handled algorithmically almost as efficient as deterministic automata over finite words. In this paper, we show how and when we can determinize automata over infinite words by the standard powerset construction for finite words. The presented construction is more efficient than all-purpose constructions for automata that recognize languages in $F_\sigma \cap G_\delta$. Further, based on the powerset construction, we present an improved automata construction that handles the quantification in the automata-based approach for $\mathrm{FO}(\mathbb{R}, \mathbb{Z}, +, <)$ much more efficiently.

1 Introduction

Around 45 years ago, Büchi and others [3, 7, 18, 19] discovered that automata over infinite words and trees are a useful mathematical tool to understand the decidability of various logics. Nowadays, automata over finite and infinite objects have also emerged as a powerful tool for specification and verification of nonterminating programs [14,22] and for implementing decisions procedures for logical theories [1,6,8]. For instance, the automata-theoretic approach to model checking is easy to understand, automatic, and thus attractive to practitioners. However, its effectiveness is sensitive to the automata model and automata sizes.

Reasoning about or with restricted classes of automata over infinite words is often simpler and more efficient. A prominent example are weak deterministic Büchi automata (WDBAs), which can be handled algorithmically almost as efficient as deterministic automata over finite words. In contrast to Büchi automata, WDBAs can be minimized efficiently [16] and are easy to complement. WDBAs can be used to represent and manipulate sets definable in the mixed first-order logic over the reals and the integers,

* This work was supported by the German Research Council (DFG) and the Swiss National Science Foundation (SNF).

FO($\mathbb{R}, \mathbb{Z}, +, <$) [1]. Further, languages of temporal properties like safety and guarantee properties can be recognized by WDBAs [4]. More general, the languages recognized by WDBAs are the ω-regular languages in the Borel class $F_\sigma \cap G_\delta$ [15,17]. However, it is not obvious how we can benefit from the algorithms for WDBAs if a given automaton is, e.g., a nondeterministic parity automaton that accepts a language in $F_\sigma \cap G_\delta$. In [11], Kupferman et al. observed that the standard powerset construction for automata over finite words can be used to obtain an equivalent WDBA from a given automaton when it accepts a language in $F_\sigma \cap G_\delta$. However, no concrete algorithm is given. In particular, the crucial point how to efficiently determine the accepting states of the WDBA is not discussed.

In this paper, we show how and when we can use the powerset construction to construct from some automaton an equivalent deterministic Büchi automaton. We present an algorithm for the general case and a more efficient one for the special case, where we require that the given automaton accepts a language in $F_\sigma \cap G_\delta$. Generally speaking, instead of using a complicated all-purpose construction for determinizing automata, like Safra's construction [20], we suggest to use specialized, more efficient constructions for subclasses of ω-regular languages. An application of our specialized algorithm is to handle the quantification in the automata-based approach of FO($\mathbb{R}, \mathbb{Z}, +, <$) more efficiently. In [1], Boigelot, Jodogne, and Wolper used the breakpoint construction [13,21] to determinize weak Büchi automata that describe sets definable in FO($\mathbb{R}, \mathbb{Z}, +, <$). Using the powerset construction has the following practical advantages over the breakpoint construction: (1) The powerset construction builds automata that usually have fewer states than the automata obtained by the breakpoint construction. The worst case of the powerset construction is slightly better than the worst case of the breakpoint construction. (2) The powerset construction is easier to implement. For instance, the breakpoint construction builds an automaton, where the states are pairs of sets of states of a given co-Büchi automaton; in the powerset construction, we only have to deal with sets of states.

Furthermore, we present another improvement for the automata-based approach to decide FO($\mathbb{R}, \mathbb{Z}, +, <$). In [6], we introduced so-called don't care words as a means to reduce the automata sizes. Experimental results demonstrated that the savings can be significant; on many practical examples, the number of states reduces up to several orders of magnitude [6]. However, it turned out that the automata construction in [6] that handles the quantification when using don't care words was the bottleneck of our implementation. We present a new construction, which is more efficient than the previous one. Our new implementation[3] that uses the construction presented in this paper usually outperforms our old implementation by a factor of over 50.

We proceed as follows. In §2, we give preliminaries. In §3, we show how and when we can use the powerset construction for automata over infinite words. In §4, we present our new construction to handle the quantification in the automata-based approach for $\mathrm{FO}(\mathbb{R}, \mathbb{Z}, +, <)$.

2 Background

We assume that the reader is familiar with the basics of automata theory and first-order logic. The purpose of this section is to recall some background in these areas, and fix the notation and terminology used in the remainder of the text.

2.1 Languages and Automata

Let Σ be an alphabet. We denote the set of all (finite) words over Σ by Σ^*. We define $\Sigma^+ := \Sigma^* \setminus \{\varepsilon\}$, where ε is the empty word. Σ^ω is the set of all (infinite) words over Σ. We write $|w|$ for the *length* of $w \in \Sigma^*$. We often write a word $w \in \Sigma^*$ of length $\ell \geq 0$ as $w_0 \dots w_{\ell-1}$ and $\alpha \in \Sigma^\omega$ as $\alpha_0\alpha_1 \dots$, where w_i and α_i denote the ith letter of w and α, respectively.

A (finite, labeled, pointed) *transition system* (TS) T is a tuple $(Q, \Sigma, \delta, q_\mathrm{I})$, where Q is a finite set of states, Σ is an alphabet, $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ is the transition function, and $q_\mathrm{I} \in Q$ is the initial state. For $q \in Q$, we define $T_q := (Q, \Sigma, \delta, q)$. We extend δ to the function $\hat{\delta} : Q \times \Sigma^* \to \mathcal{P}(Q)$ defined as $\hat{\delta}(q, \varepsilon) := \{q\}$ and $\hat{\delta}(q, bu) := \bigcup_{p \in \delta(q,b)} \hat{\delta}(p, u)$, where $q \in Q$, $b \in \Sigma$, and $u \in \Sigma^*$. T is *deterministic* if $|\delta(p, b)| = 1$, for all $p \in Q$ and $b \in \Sigma$. In this case, we write $\delta(p, b) = q$ and $\hat{\delta}(p, w) = q$ instead of $\delta(p, b) = \{q\}$ and $\hat{\delta}(p, w) = \{q\}$, respectively. A state $q \in Q$ is *reachable* from $p \in Q$ if there is a word $w \in \Sigma^*$ such that $q \in \hat{\delta}(p, w)$. In the remainder of the text, we assume that every state in a TS is reachable from its initial state. A *strongly connected component* (SCC) of T is a set $S \subseteq Q$ such that every $p \in S$ is reachable from every $q \in S$ and S is maximal. A *run* ϱ of T on $\alpha \in \Sigma^\omega$ is a word $\varrho \in Q^\omega$ such that $\varrho_0 = q_\mathrm{I}$ and $\varrho_{i+1} \in \delta(\varrho_i, \alpha_i)$, for all $i \geq 0$. $\mathrm{Inf}(\varrho)$ is the set of states that occur infinitely often in ϱ.

An *automaton* $\mathcal{A}$ is a tuple (T, C), where T is a TS and C is an acceptance condition. In the following, we mainly use the Büchi and co-Büchi conditions, which are defined as follows.[4]

[3] Our implementation is publicly available at `http://lira.gforge.avacs.org/`.

[4] Other standard acceptance conditions like Rabin or parity to which we sometimes refer in the remainder of the text are given in Appendix A.

– $S \subseteq Q$ satisfies the *Büchi condition* $C \subseteq Q$ if $S \cap C \neq \emptyset$.
– $S \subseteq Q$ satisfies the *co-Büchi condition* $C \subseteq Q$ if $S \cap C = \emptyset$.

A run ϱ is *accepting* if $\mathrm{Inf}(\varrho)$ satisfies the acceptance condition C; it is *rejecting*, otherwise. We define $L(\mathcal{A}) := \{\alpha \in \Sigma^\omega : \text{there is an accepting run } \varrho \text{ on } \alpha\}$.

We type an automaton $\mathcal{A} = (T, C)$ according to its acceptance condition C. For instance, if C is the Büchi condition, $\mathcal{A}$ is a *Büchi automaton* (BA) and if C is the co-Büchi condition, we call it a *co-Büchi automaton* (co-BA). Further, if T is *deterministic*, $\mathcal{A}$ is a *deterministic* BA (DBA) or *deterministic co*-BA (co-DBAs), respectively. A BA (T, C) is *weak* if $S \cap C = \emptyset$ or $S \subseteq C$, for every SCC $S \subseteq Q$. We use the initialisms WBA for "weak Büchi automaton" and WDBA for "weak deterministic Büchi automaton."

For $L \subseteq \Sigma^\omega$, we define the congruence relation $\approx_L \subseteq \Sigma^* \times \Sigma^*$ as $u \approx_L v$ iff $u\alpha \in L \Leftrightarrow v\alpha \in L$, for all $\alpha \in \Sigma^\omega$. If $\approx_L$ has finite index, we define the TS $\mathcal{C}_L$ as $\mathcal{C}_L := (\{[v] : v \in \Sigma^*\}, \Sigma, \delta, [\varepsilon])$ with $\delta([v], b) := [vb]$, where $[u]$ denotes the equivalence class of $u \in \Sigma^*$. Note that δ is well-defined.

2.2 Representing Sets of Reals with Automata

Let $\mathfrak{R}$ be the structure $(\mathbb{R}, \mathbb{Z}, +, <)$, where $+$ and $<$ are as expected and $\mathbb{Z}$ is the unary predicate such that $\mathbb{Z}(x)$ is true iff x is an integer. For a formula $\varphi(x_1, \ldots, x_r)$ and $a_1, \ldots, a_r \in \mathbb{R}$, we write $\mathfrak{R} \models \varphi[a_1, \ldots, a_r]$ if φ is true in $\mathfrak{R}$ when the variable x_i is interpreted as a_i, for $1 \leq i \leq r$.

In [1], Boigelot, Jodogne, and Wolper show that for every first-order definable set $X \subseteq \mathbb{R}^r$ in $\mathfrak{R}$, there is a WDBA $\mathcal{A}$ that describes X. Moreover, they show that $\mathcal{A}$ can be effectively constructed from a formula $\varphi(x_1, \ldots, x_r)$ that defines X, i.e., $X = \{\overline{a} \in \mathbb{R}^r : \mathfrak{R} \models \varphi[\overline{a}]\}$. We recall the precise correspondence between subsets of $\mathbb{R}^r$ and languages from [1]. Let Σ be the alphabet $\{0, 1\}$.[5] For $r \geq 1$, we define:

1. V_r is the set of all words over the alphabet $\Sigma^r \cup \{\star\}$ of the form $v \star \gamma$, where $v \in (\Sigma^r)^+$ and $\gamma \in (\Sigma^r)^\omega$.
2. A word $v \star \gamma \in \mathsf{V}_r$ represents the vector of reals with r components
$$\langle\!\langle v \star \gamma \rangle\!\rangle := -2^{|v|-1} + \textstyle\sum_{0<i<|v|} 2^{|v|-i-1} \cdot v_i + \sum_{i \geq 0} 2^{-i-1} \cdot \gamma_i$$
where vector addition and scalar multiplication are componentwise.
3. For a formula $\varphi(x_1, \ldots, x_r)$, we define $L(\varphi) := \{\alpha \in \mathsf{V}_r : \mathfrak{R} \models \varphi[\langle\!\langle \alpha \rangle\!\rangle]\}$.

[5] To simplify matters, we use the 2's complement representation. A generalization to the p's complement representation for any $p > 1$ is straightforward.

Note that in a word $v \star \gamma \in \mathsf{V}_r$ the symbol $\star$ plays the role of a decimal point, separating the integer part v from the fractional part γ. Further, note that every vector in $\mathbb{R}^r$ can be represented by a word in V_r. However, the representation is not unique. First, we can repeat the first letter arbitrary often without changing the represented vector. Second, a vector that contains in a component a rational whose denominator has only the prime factor 2 has distinct representations, e.g., $\langle\!\langle 0 \star 10^\omega \rangle\!\rangle = \langle\!\langle 0 \star 01^\omega \rangle\!\rangle = \frac{1}{2}$. For $\alpha \in \mathsf{V}_r$, let $L(\alpha) := \{\beta \in \mathsf{V}_r : \langle\!\langle \alpha \rangle\!\rangle = \langle\!\langle \beta \rangle\!\rangle\}$.

3 Determinization with the Powerset Construction

In this section, we investigate when and how we can use the powerset construction to determinize automata over infinite words. The *powerset transition system* of a TS $T = (Q, \Sigma, \delta, q_\mathrm{I})$ is $\mathcal{P}(T) := (\mathcal{P}(Q), \Sigma, \delta_\mathcal{P}, \{q_\mathrm{I}\})$ with $\delta_\mathcal{P}(R, b) := \bigcup_{q \in R} \delta(q, b)$, for $R \subseteq Q$ and $b \in \Sigma$.

Lemma 1. *Let $\mathcal{A} = (T, C)$ be an automaton. If the* DBA $(\mathcal{C}_{L(\mathcal{A})}, E)$ *accepts $L(\mathcal{A})$, for some E then there is an F such that the* DBA $(\mathcal{P}(T), F)$ *accepts $L(\mathcal{A})$.*

Proof. Assume that $T = (Q, \Sigma, \delta, q_\mathrm{I})$. We define $F := \{P \subseteq Q : \hat{\delta}(q_\mathrm{I}, u) = P \text{ and } [u] \in E, \text{ for some } u \in \Sigma^*\}$. For $\alpha \in \Sigma^\omega$, let ϱ be the run of $\mathcal{C}_{L(\mathcal{A})}$ and ϱ' be the run of $\mathcal{P}(T)$. We show that $\varrho_i \in E$ iff $\varrho'_i \in F$, for all $i \geq 0$. Let $v := \alpha_0 \dots \alpha_{i-1}$. Note that $\varrho_i = [v]$. The direction from left to right holds by the definition of F. For the other direction, assume that $\varrho'_i \in F$, i.e., there is a word $u \in \Sigma^*$ with $\hat{\delta}(q_\mathrm{I}, u) = \varrho'_i$ and $[u] \in E$. Since $\varrho'_i = \hat{\delta}(q_\mathrm{I}, u) = \hat{\delta}(q_\mathrm{I}, v)$, we have $u \approx_{L(\mathcal{A})} v$ and hence, $[u] = [v] = \varrho_i$. $\square$

The converse direction of Lemma 1 does not hold in general. To see this, let L be the language $\{\alpha \in \{0,1\}^\omega : 1 \text{ occurs infinitely often in } \alpha\}$. Since $\approx_L$ has only one equivalence class, it is straightforward to see that there is no set E such that $(\mathcal{C}_L, E)$ accepts L. However, there is a DBA $\mathcal{A} = (T, C)$ that accepts L and since T is deterministic, there is obviously a set F such that the DBA $(\mathcal{P}(T), F)$ accepts L.

An application of Lemma 1 is to use the powerset construction to determinize an automaton $\mathcal{A}$ whenever the DBA consisting of the TS $\mathcal{C}_{L(\mathcal{A})}$ with an appropriate set of accepting states E accepts the language $L(\mathcal{A})$. Unfortunately, checking for an automaton $\mathcal{A}$ whether there is a set E such that $L(\mathcal{C}_{L(\mathcal{A})}, E) = L(\mathcal{A})$ is PSPACE-hard. This can be shown by a similar argumentation as in the proof of Theorem 4.2 in [12]. However, there are important classes of ω-regular languages L for which there is always a set E such that $L(\mathcal{C}_L, E) = L$. For instance, the ω-regular languages in

```
if S has no loop then return REJECTING
Let R be some state in S.
Let w be some word in Σ+ such that δ̂_P(R, w) = R.
for each q ∈ R do
    if w^ω ∈ L(T_q, C) then return ACCEPTING
return REJECTING
```

Fig. 1. Algorithm to determine whether an SCC S of $\mathcal{P}(T)$ is accepting or rejecting.

the Borel class $F_\sigma \cap G_\delta$ satisfy this condition. Relevant classes of temporal properties define languages in $F_\sigma \cap G_\delta$, e.g., safety and guarantee properties [4] and languages definable in the first-order logic over $\mathfrak{R}$ are also in $F_\sigma \cap G_\delta$ [1]. In the next subsection, we present an algorithm that determinizes automata for such languages more efficiently than by using other constructions, like the breakpoint construction or Safra's construction.

3.1 Powerset Construction for Languages in $F_\sigma \cap G_\delta$

For this subsection, assume that the automaton $\mathcal{A} = (T, C)$ accepts a language in the Borel class $F_\sigma \cap G_\delta$, where $T = (Q, \Sigma, \delta, q_I)$. We present an algorithm that determinizes $\mathcal{A}$ by using the powerset construction.

We make the following observation. From [17], we know that some DBA $(\mathcal{C}_{L(\mathcal{A})}, E)$ accepts $L(\mathcal{A})$. It follows from Lemma 1 that some DBA $(\mathcal{P}(T), F)$ accepts $L(\mathcal{A})$. According to Theorem 5.2 in [1], all SCCs of $(\mathcal{P}(T), F)$ contain only accepting or only rejecting loops. Define G as the union of all states of SCCs with at least one accepting loop. Note that $(\mathcal{P}(T), G)$ is a WDBA that accepts $L(\mathcal{A})$.

To determinize $\mathcal{A}$ we proceed in two steps. First, we construct $\mathcal{P}(T)$. Then, we use the algorithm in Figure 1 to compute the set G.[6] If an SCC S has no loop then its states are not in G by definition. Otherwise, let $R \in S$ and $w \in \Sigma^+$ such that $\hat{\delta}_{\mathcal{P}}(R, w) = R$. Lemma 2 states that S is accepting, i.e., $S \subseteq G$ iff $w^\omega \in L(T_q, C)$, for some $q \in R$.

Lemma 2. *Let R be a state in $\mathcal{P}(T)$ and $w \in \Sigma^+$ such that $\hat{\delta}_{\mathcal{P}}(R, w) = R$. Then, (T_q, C) accepts w^ω, for some $q \in R$ iff the SCC of R is accepting.*

Proof. Let u in Σ^* such that $\hat{\delta}_{\mathcal{P}}(\{q_{\mathrm{I}}\}, u) = R$. "If case:" If (T_q, C) accepts w^ω, for some $q \in R$ then $\mathcal{A}$ accepts uw^ω. Since $L(\mathcal{P}(T), G) = L(\mathcal{A})$, $(\mathcal{P}(T), G)$ also accepts uw^ω. Since $(\mathcal{P}(T), G)$ is a WDBA and R occurs infinitely often in the run on uw^ω, the SCC of R is accepting. "Only if case:" If the SCC of R is accepting, $(\mathcal{P}(T), G)$ accepts uw^ω. So, $\mathcal{A}$ accepts uw^ω. Since $\hat{\delta}(q_{\mathrm{I}}, u) = R$, (T_q, C) accepts w^ω, for some $q \in R$. □

[6] In [11], it is claimed that for a BA $\mathcal{A} = (T, C)$ for which there is a WDBA that accepts $L(\mathcal{A})$, the Büchi condition for $\mathcal{P}(T)$ can be chosen as $\{P : P \cap C \neq \emptyset\}$. A counterexample for this claim is the TS $(\{r, s, t\}, \{0\}, \delta, r)$ with $\delta(r, 0) = \{r, s\}$ and $\delta(s, 0) = \delta(t, 0) = \{t\}$ and the Büchi condition $\{s\}$.

$R \leftarrow \emptyset$
$A \leftarrow \emptyset$
Let G be the graph (V, E) with $V := S$ and $E := \{(p, q) : \delta(p, b) = q$, for some $b \in \Sigma\}$.
while there is a loop $\pi = v_0 \dots v_\ell$ in G with $0 < \ell \leq |S|$ and $v_0 \in V \setminus R$ and there is no $X \in A$ such that $X \subseteq \{v_0, \dots, v_\ell\}$ **do**
  Let $u \in \Sigma^*$ be a word with $\hat{\delta}(q_{\mathrm{I}}, u) = v_0$.
  Let $w \in \Sigma^+$ be a word of length $\ell - 1$ with $\delta(v_i, w_i) = v_{i+1}$, for all $0 \leq i < \ell$.
  if $uw^\omega \notin L(\mathcal{A})$ **then**
    $R \leftarrow R \cup \{v_0, \dots, v_\ell\}$
    Update A, i.e., remove the v_is in every $X \in A$.
  else
    $A \leftarrow A \cup \big\{\{v_i : 0 \leq i \leq \ell \text{ and } v_i \notin R\}\big\}$
  end if
  while there is a vertex $v \in V$ with $\{v\} \in A$ **do**
    Delete vertex v in G.
    Update A, i.e., remove $X \in A$ whenever $v \in X$.
  end while
end while
return $S \setminus R$

Fig. 2. Algorithm to determine the set of accepting states for an SCC S for T.

It remains to answer the question whether (T_q, C) accepts w^ω or equivalently whether $\{w^\omega\} \cap L(T_q, C) = \emptyset$. We can easily construct an automaton that accepts $\{w^\omega\} \cap L(T_q, C)$ and check its emptiness efficiently according to the acceptance condition (see [5, 9, 10] for details).

3.2 The General Case

In this subsection, we consider a more general case: A problem instance consists of an automaton $\mathcal{A}$ and a deterministic TS T. We want to compute an F such that the DBA (T, F) accepts $L(\mathcal{A})$. We require that there is at least one F' such that the DBA (T, F') accepts $L(\mathcal{A})$. In the remainder of this subsection, assume that $T = (Q, \Sigma, \delta, q_{\mathrm{I}})$.

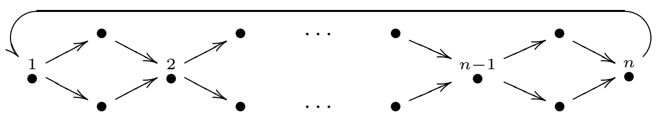
Observe that we can consider each SCC of T separately, i.e., for each SCC S, we can compute a set $F_S \subseteq Q$ without taking into account the other SCCs of T. However, note that such a set F_S is not uniquely determined and there might be dependencies on the states in S that we have to take care off. F is then the union of the sets F_S, for all SCCs S of T. The algorithm in Figure 2 returns such a set F_S, for an SCC S of T.

Due to space limitation we only sketch the algorithm. We iteratively investigate loops π in the SCC S from which we gain additional information about which of the states in S have to be accepting or rejecting. Note that a loop is a sequence of vertices $v_0 \dots v_\ell$ with $\ell > 0$, $v_0 = v_\ell$, and for all $0 \leq i < \ell$, there is an edge from v_i to v_{i+1}. For a loop $\pi = v_0 \dots v_\ell$, there is a word $w \in \Sigma^+$ that visits the states in π in the same order. Moreover, there is a word $u \in \Sigma^*$ with $\hat{\delta}(q_{\mathrm{I}}, u) = v_0$. We check if $\mathcal{A}$ rejects

uw^ω. If this is the case, we know that the states $v_0, \ldots, v_\ell$ are rejecting. If $\mathcal{A}$ accepts uw^ω, we know that at least one of the states $v_0, \ldots, v_\ell$ is accepting. The algorithm maintains a set R, where R contains the states that have to be rejecting. The algorithm also maintains a set A of sets of states, where $X \in A$ means that at least one of the states in X has to be accepting. Initially, R and A are empty. For example, if we derive the fact that a state $p \in S$ is rejecting, we put p in R and delete p in every $X \in A$. Further, if A contains a singleton $\{q\}$, we know that q has to be accepting and we remove the sets X from A that contain q. Moreover, the algorithm maintains a graph G. Intuitively speaking, G together with A describe the loops of the SCC S that we still need to investigate. Initially, G is the transition graph of the SCC S.

Note that we need not to investigate loops in G that visit a state for which we already know that it is accepting. Thus, as soon as we conclude that a state p is accepting, we delete p in G (and all its in-going and out-going edges). That means, that no loop in the updated graph will visit p. Further, a loop π has to visit at least a state for which we do not know whether it is accepting or rejecting. With out loss of generality, we assume that the first state in π is such a state. Moreover, we can restrict ourselves to loops π for which the set of visited states is not a superset of any $X \in A$. The reason for this is that at least one state in X has to be accepting and thus, $\mathcal{A}$ also accepts the word corresponding to the loop π. Therefore, we do not obtain any new information by investigating π. Finally, note that it suffices to check loops of length at most $|S| + 1$.

The algorithm in Figure 2 terminates since it only checks finitely many loops. However, in the worst case it checks exponentially many loops. For instance, the SCC graph



has 2^{n-1} loops of length $2n$. If all words corresponding to these loops are accepted, the algorithm checks all loops of length $2n$. We remark that from smaller loops we obtain more information. In particular, from a self-loop we immediately see if the state in it has to be accepting or rejecting. So, a heuristic is to check loops ordered increasingly by their lengths.

We want to remark that the algorithm in Figure 2 can be easily adapted such that we can use it to obtain a set $F \subseteq Q$ for the co-Büchi condition, i.e., that the co-DBA (T, F) accepts $L(\mathcal{A})$. However, note that if $\mathcal{A}$ is a co-BA we can also already the breakpoint construction to obtain an equivalent co-DBA. The problem for other acceptance conditions is left open.

4 Quantification with Don't Care Words

In this section, we optimize the automata-based approach to decide the first-order logic over $\mathfrak{R}$. In §4.1, we recall previous work on don't care words and in §4.2, we present a new construction to handle the quantification in the first-order logic over $\mathfrak{R}$ more efficiently.

4.1 Don't Care Words

In [6], so-called don't care words are used as a means to reduce the automata sizes. The intuition of a don't care word is that it is irrelevant whether it belongs to a language or not. More formally: for languages $L, L', D \subseteq \Sigma^\omega$, we write $L \equiv_D L'$ if $L \setminus D = L' \setminus D$. That means, L and L' are equivalent modulo the language D. We call D a *don't care set*. Observe that a don't care set D gives us the flexibility to add and remove words in D from a language L such that the automata representation of L can be made smaller.

In the following, we focus on the following don't care sets. Fix $r \geq 0$ and $\Sigma := \{0, 1\}$. A word $\alpha \in (\Sigma^r \cup \{\star\})^\omega$ is a *don't care word* if there are $t \in \{1, \dots, r\}$ and $k \in \mathbb{N}$ such that $\alpha_i \in \Sigma^r$ and $(\alpha_i)_{\restriction t} = 1$, for all $i \geq k$, where $b_{\restriction t}$ denotes the tth coordinate of $b \in \Sigma^r$. DC_r is the set of all don't care words in $(\Sigma^r \cup \{\star\})^\omega$.

In [6], we extend many automata constructions so that they take a don't care set with certain properties into account. For instance, the unique minimal WDBA with respect to the don't care set DC_r exists and we can compute it efficiently. Unfortunately, handling the quantification in the first-order logic over $\mathfrak{R}$ becomes more involved when using don't care words. Note that when we delete a "track" in a word $\alpha \in \mathsf{DC}_{r+1}$, we might obtain a word $\alpha' \notin \mathsf{DC}_r$. Hence, by just deleting the projected "track" in the words of a language, we may obtain a language that contains too many words. It turned out that the given construction in [6] is the bottleneck of our implementation. In the following, we present a new construction, which is much more efficient.

4.2 Optimized Construction

For this subsection, let $\varphi(x_1, \dots, x_r)$ be a formula and $\mathcal{A}$ a WDBA such that $L(\varphi) \equiv_{\mathsf{DC}_r} L(\mathcal{A})$. Our construction of a WDBA $\mathcal{B}$ with $L(\exists x_1 \varphi) \equiv_{\mathsf{DC}_{r-1}} L(\mathcal{B})$ consists of two separate steps. The first step takes care of the words $\alpha \in \mathsf{DC}_r \cap L(\neg\varphi)$ that are accepted by the given WDBA and for which the deletion of the first "track" yields a word not in DC_{r-1}. The second step uses the powerset construction presented in §3.1 with minor modifications.

For the first step, we use the following automata construction. Let $\mathcal{D} = (U, G)$ and $\mathcal{D}' = (U', G')$ be WDBAs with $U = (Q, \Sigma^r \cup \{\star\}, \delta, q_\mathrm{I})$ and $U' = (Q', \Sigma^r \cup \{\star\}, \delta', q'_\mathrm{I})$. We define the WDBA $\mathcal{D} \otimes \mathcal{D}' := (U \otimes U', E)$ as $U \otimes U' := (Q \times Q', \Sigma^r \cup \{\star\}, \eta, (q_\mathrm{I}, q'_\mathrm{I}))$, where η and E are as follows.

- For $(p, p') \in Q \times Q'$, we define $\eta((p, p'), \star) := \big(\delta(p, \star), \delta'(p', \star)\big)$.
- For the initial state $(q_\mathrm{I}, q'_\mathrm{I})$, $c \in \{0, 1\}$, and $b \in \{0, 1\}^{r-1}$, we define

$$\eta((q_\mathrm{I}, q'_\mathrm{I}), (c, b)) := \begin{cases} \big(\delta(q_\mathrm{I}, (0, b)), \hat{\delta}(q_\mathrm{I}, (01, bb))\big) & \text{if } c = 0, \\ \big(\delta(q_\mathrm{I}, (1, b)), \delta'(q'_\mathrm{I}, (0, b))\big) & \text{if } c = 1. \end{cases}$$

- For a state $(p, p') \neq (q_\mathrm{I}, q'_\mathrm{I})$, $c \in \{0, 1\}$, and $b \in \{0, 1\}^{r-1}$, we define

$$\eta((p, p'), (c, b)) := \begin{cases} \big(\delta(p, (0, b)), \delta(p, (1, b))\big) & \text{if } c = 0, \\ \big(\delta(p, (1, b)), \delta'(p', (0, b))\big) & \text{if } c = 1. \end{cases}$$

- The states of an SCC S of $U \otimes U'$ are in E iff there is a state $(p, q) \in S$ such that $p \in F$ and $\eta((p, q), (0, b)) \in S$, for some $b \in \{0, 1\}^{r-1}$.

Lemma 3. *The* WDBA $\mathcal{A} \otimes \mathcal{A}$ *accepts only words in* V_r*. Moreover, for* $\alpha \in \mathsf{V}_r$ *the following properties hold.*

1. *If* $\alpha \in L(\varphi)$ *then* $L(\mathcal{A} \otimes \mathcal{A}) \cap L(\alpha) \neq \emptyset$.
2. *If* $\alpha \notin L(\varphi)$ *then* $(L(\mathcal{A} \otimes \mathcal{A}) \setminus D) \cap L(\alpha) = \emptyset$*, where* $D := \{v \star \gamma \in \mathsf{V}_r :$ *there are* $k \in \mathbb{N}$ *and* $t \in \{2, \dots, r\}$ *such that* $(\gamma_i)_{\restriction t} = 1$*, for all* $i \geq k\}$.

Intuitively, the first component of the states of the WDBA $\mathcal{A} \otimes \mathcal{A}$ are used to simulate the runs of the WDBA $\mathcal{A}$. $\mathcal{A} \otimes \mathcal{A}$ will not accept a word $\alpha \in \mathsf{DC}_r$ if there exists a $k \geq 0$ such that $(\alpha_i)_{\restriction 1} = 1$, for all $i \geq k$. If, however, $\alpha \in L(\varphi)$ then there is a $\beta \in L(\alpha)$ such that $(\beta_i)_{\restriction 1} = 0$, for all $i \geq k$ and $\beta \in L(\mathcal{A} \otimes \mathcal{A})$. The second component of the states are used to detect the case whether the first "track" of α eventually becomes 1 or not.

In the second step, we use the WDBA $\mathcal{A} \otimes \mathcal{A}$ to construct an auxiliary WBA $\mathcal{C} = (U, G)$ such that $L(\mathcal{C}) \equiv_{\mathsf{DC}_{r-1}} L(\exists x_1 \varphi)$. This construction step is straightforward and is mainly done by ignoring the first component of a letter in the transition function of the WDBA $\mathcal{A} \otimes \mathcal{A}$. Intuitively speaking, $\mathcal{C}$ guesses the bits of x_1.[7] Now, we define the WDBA $\mathcal{B}$ as $(\mathcal{P}(U), G)$, where we adapt the algorithm described in §3.1 to determine G. For determining whether an SCC S of $\mathcal{P}(U)$ has to be accepting or rejecting, we pick in line 3 of the algorithm in Figure 1 a word $w \in (\Sigma^{r-1})^+$ such that $w^\omega \notin \mathsf{DC}_{r-1}$. That means, for every $t \in \{1, \dots, r-1\}$, there is an integer i such that $0 \leq i < |w|$ and $(w_i)_{\restriction t} \neq 1$. If no such word exists, it is irrelevant whether S is accepting or rejecting, since by eventually staying in S when reading a word we accept or reject a don't care word.

[7] Some additional work is needed for the sign bit, see, e.g., [1, 2] for details.

References

1. B. Boigelot, S. Jodogne, and P. Wolper, *An effective decision procedure for linear arithmetic over the integers and reals*, ACM Trans. Comput. Log., 6 (2005), pp. 614–633.
2. B. Boigelot and L. Latour, *Counting the solutions of Presburger equations without enumerating them*, Theoret. Comput. Sci., 313 (2004), pp. 17–29.
3. J. Büchi, *On a decision method in restricted second order arithmetic*, in Logic, Methodology and Philosophy of Science (Proc. 1960 Int. Congr.), Stanford University Press, 1962, pp. 1–11.
4. E. Chang, Z. Manna, and A. Pnueli, *The safety-progress classification*, in Logic and Algebra of Specifications, F. Bauer, W. Brauer, and H. Schwichtenberg, eds., NATO Advanced Science Institutes Series, Springer-Verlag, 1991, pp. 143–202.
5. E. M. Clarke, E. A. Emerson, and A. P. Sistla, *Automatic verification of finite-state concurrent systems using temporal logic specifications*, ACM Trans. Program. Lang. Syst., 8 (1986), pp. 244–263.
6. J. Eisinger and F. Klaedtke, *Don't care words with an application to the automata-based approach for real addition*, in Proc. of the 18th Int. Conf. on Computer Aided Verification, vol. 4144 of Lect. Notes Comput. Sci., 2006, pp. 67–80.
7. C. C. Elgot and M. O. Rabin, *Decidability and undecidability of extensions of second (first) order theory of (generalized) successor*, J. Symbolic Logic, 31 (1966), pp. 169–181.
8. J. G. Henriksen, J. L. Jensen, M. E. Jørgensen, N. Klarlund, R. Paige, T. Rauhe, and A. Sandholm, *Mona: Monadic second-order logic in practice*, in Proc. of the 1st Int. Workshop on Tools and Algorithms for Construction and Analysis of Systems, vol. 1019 of Lect. Notes Comput. Sci., 1996, pp. 89–110.
9. M. R. Henzinger and J. A. Telle, *Faster algorithms for the nonemptiness of Streett automata and for communication protocol pruning*, in Scandinavian Workshop on Algorithm Theory, 1996, pp. 16–27.
10. V. King, O. Kupferman, and M. Y. Vardi, *On the complexity of parity word automata*, in Proc. of the 4th Int. Conf. on Foundations of Software Science and Computation Structures (FoSSaCS), Lect. Notes Comput. Sci., 2001, pp. 276–286.
11. O. Kupferman, G. Morgenstern, and A. Murano, *Typeness for ω-regular automata*, Int. J. Found. Comput. Sci., 17 (2006), pp. 869–884.
12. O. Kupferman and M. Vardi, *Freedom, weakness, and determinism: From linear-time to branching-time*, in Proc. of the 13th Annual IEEE Symp. on Logic in Computer Science, IEEE Computer Society Press, 1998, pp. 81–92.
13. ———, *Weak alternating automata are not that weak*, ACM Trans. Comput. Log., 2 (2001), pp. 408–429.
14. R. P. Kurshan, *Computer Aided Verification of Coordinating Processes*, Princeton University Press, 1994.
15. L. H. Landweber, *Decision problems for ω-automata*, Math. Syst. Theory, 3 (1969), pp. 376–384.
16. C. Löding, *Efficient minimization of deterministic weak ω-automata*, Inform. Process. Lett., 79 (2001), pp. 105–109.
17. O. Maler and L. Staiger, *On syntactic congruences for omega-languages*, Theoret. Comput. Sci., 181 (1997), pp. 93–112.
18. M. O. Rabin, *Decidability of second-order theories and automata over infinite trees*, Trans. Amer. Math. Soc., 141 (1969), pp. 1–35.

19. ———, *Decidable theories*, in Handbook of Mathematical Logic, J. Barwise, ed., vol. 70 of Studies in Logic, North-Holland, 1977, pp. 595–629.
20. S. Safra, *Complexity of Automata on Infinite Objects*, PhD thesis, The Weizman Institute of Science, Rehovot, Israel, 1989.
21. T. H. Satoru Miyano, *Alternating finite automata on ω-words*, Theoret. Comput. Sci., 32 (1984), pp. 321–330.
22. M. Vardi and P. Wolper, *An automata-theoretic approach to automatic program verification*, in Proc. of the 1st Symp. on Logic in Computer Science, IEEE Computer Society Press, 1986, pp. 322–331.

A Acceptance Conditions

We recall other acceptance conditions for automata over infinite words. Let $\mathcal{A} = (Q, \Sigma, \delta, q_\mathrm{I}, C)$ be an automaton. For $S \subseteq Q$, we define the following acceptance conditions:

- S satisfies the *Muller condition* $C \subseteq \mathcal{P}(Q)$ if $S \in C$.
- S satisfies the *parity condition* $C : Q \to \mathbb{N}$ if $\min\{C(q) : q \in S\}$ is even.
- S satisfies the *Rabin condition* $C \subseteq \mathcal{P}(Q) \times \mathcal{P}(Q)$ if there is a pair $(E, F) \in C$ such that $S \cap E \neq \emptyset$ and $S \cap F = \emptyset$.
- S satisfies the *Streett condition* $C \subseteq \mathcal{P}(Q) \times \mathcal{P}(Q)$ if for all pairs $(E, F) \in C$, it holds that $S \cap E = \emptyset$ or $S \cap F \neq \emptyset$.

Note that the Büchi condition is dual to the co-Büchi condition: S satisfies the Büchi condition iff S does not satisfy the co-Büchi condition. Similarly, the Rabin condition is dual to the Streett condition. Further note that the Rabin condition is a disjunction of conjunctions of Büchi and co-Büchi conditions, and a Streett condition is a conjunction of disjunctions of Büchi and co-Büchi conditions.

B Additional Proof Details

B.1 Proof of Lemma 3

Proof. The proof that $L(\mathcal{A} \otimes \mathcal{A}) \subseteq \mathsf{V}_r$ is straightforward. For the remainder of the proof, assume that $\alpha \in \mathsf{V}_r$.

1. Without loss of generality, we assume that $\alpha \notin \mathsf{DC}_r$. Let $\varrho = q_0 q_1 \ldots \in Q^\omega$ be the run of $\mathcal{A} = (T, C)$ on α. Note that ϱ is accepting. There is an integer $k \geq 0$, such that $q_i \in C$, for all $i \geq k$. Let ϱ' be the run of $\mathcal{A} \otimes \mathcal{A} = (T \otimes T, E)$ on α. By definition, the run ϱ' has the form $\varrho' = (q_0, q'_0)(q_1, q'_1) \ldots \in (Q \times Q)^\omega$, for some $q'_0, q'_1, \ldots \in Q$. We have to show that ϱ' is accepting.

There is an SCC $S \subseteq Q \times Q$ and an integer $\ell \geq k$ such that $(q_i, q'_i) \in S$, for all $i \geq \ell$. Note that $q_i \in C$, for all $i \geq \ell$. Since $\alpha \notin \mathsf{DC}_r$, there is a $j \geq \ell$ such that $(\alpha_j)_{\restriction 1} = 0$ and $\eta((q_j, q'_j), \alpha_j) \in S$. Thus, by definition, S is accepting.

2. Let $\varrho = q_0 q_1 \ldots \in Q^\omega$ be the run of $\mathcal{A}$ on an ω-word $\beta \in L(\alpha)$ and let ϱ' be the run of $\mathcal{A} \otimes \mathcal{A}$ on β. By definition, ϱ' has the form $\varrho' = (q_0, q'_0)(q_1, q'_1) \ldots \in (Q \times Q)^\omega$, for some $q'_0, q'_1, \ldots \in Q$.

First, assume that ϱ is rejecting, i.e., there is an integer $k \geq 0$ such that $q_i \notin C$, for all $i \geq k$. Moreover, there is an SCC $S \subseteq Q \times Q$ of $\mathcal{A} \otimes \mathcal{A}$

and an integer $\ell \geq k$ such that $(q_i, q'_i) \in S$ and $q_i \notin C$ for all $i \geq \ell$. We have to show that S is not accepting. By definition, S can only be accepting if there is a state $(p, p') \in S$ with $p \in C$. This is not possible, since $\mathcal{A}$ is weak and thus p and q_ℓ cannot be in the same SCC of $\mathcal{A}$.

Second, assume that ϱ is accepting. Note that $\beta \in \mathsf{DC}_r$, since $\beta \notin L(\varphi)$ and $\beta \in L(\mathcal{A})$. If $\beta \in D$ then there is nothing to prove. In the following, assume that $\beta \notin D$. We conclude that only the first track of β is a don't care word.

Since ϱ is accepting, there is an integer $k \geq 0$ such that $q_i \in C$, for all $i \geq k$. Moreover, there is an SCC $S \subseteq Q \times Q$ of $\mathcal{A} \otimes \mathcal{A}$ and an integer $\ell \geq k$ such that $(q_i, q'_i) \in S$ and $q_i \in C$, for all $i \geq \ell$. We have to show that S is rejecting.

For the sake of contradiction, assume that S is accepting. By definition, there is a state $(p, p') \in S$ such that $\eta((p, p'), (0, b)) \in S$, for some $b \in \{0, 1\}^r$. It follows that there are words $u, v \in (\{0, 1\}^r)^+$ such that

(1) $\hat{\eta}((q_\ell, q'_\ell), u) = \hat{\eta}((q_\ell, q'_\ell), v) = (q_\ell, q'_\ell)$,
(2) $(u_j)_{\restriction 1} = 0$, for some integer j with $0 \leq j < |u|$,
(3) $(v_j)_{\restriction 1} = 1$, for all integers j with $0 \leq j < |v|$, and
(4) for every $0 \leq j < |v|$, there is an integer i such that $2 \leq i \leq r$ and $(v_j)_{\restriction i} = 0$.

Note that we can require (4), since we assume that $\beta \notin D$. Intuitively speaking, u describes a loop through the accepting state (p, q') and v describes the loop β will take.

Let $\mathcal{B} = (P, \{0, 1\}^r \cup \{\star\}, \mu, p_\mathrm{I}, G)$ be a WDBA with $L(\mathcal{B}) = L(\varphi)$. We use $\mathcal{B}$ to define an infinite sequence of words $w^{(0)}, w^{(1)}, \ldots \in (\{0, 1\}^r \cup \{\star\})^*$. Let $w^{(0)} := \beta_0 \ldots \beta_{\ell-1}$. For $i > 0$, we define $w^{(i)} := w^{(i-1)}w$, where the definition of the word $w \in (\{0, 1\}^r)^+$ depends on the state $\hat{\mu}(p_\mathrm{I}, w^{(i-1)})$.

- Case 1: $\hat{\mu}(p_\mathrm{I}, w^{(i-1)}) \notin G$. We define $w := (uv)^n$, where $n \geq 1$ is some integer such that $\hat{\mu}(p_\mathrm{I}, w^{(i-1)}(uv)^n) \in G$. Such an integer n exists, since $w^{(i-1)}(uv)^\omega \in L(\mathcal{A}) \setminus \mathsf{DC}_r$ and thus, $\mathcal{B}$ has to accept $w^{(i-1)}(uv)^\omega$.
- Case 2: $\hat{\mu}(p_\mathrm{I}, w^{(i-1)}) \in G$. We define $w := v^n$, where $n \geq 1$ is some integer such that $\hat{\mu}(p_\mathrm{I}, w^{(i-1)}v^n) \notin G$. Such an integer n exists, since $\mathcal{A}$ rejects the "normalized word" of $w^{(i-1)}v^\omega$. This fact can be shown by constructing the run on the "normalized word" of $w^{(i-1)}v^\omega$ on $\mathcal{A}$ from the second component of the state pairs in the run ϱ' of $\mathcal{A} \otimes \mathcal{A}$.

Note that for every $i \geq 0$, we have that $\hat{\eta}((q_\mathrm{I}, q_\mathrm{I}), w^{(i)}) = (q_\ell, q'_\ell)$. This follows from the definition of $w^{(0)}$ and from (1).

Let γ be the ω-word that is the limit of this sequence $w^{(0)}, w^{(1)}, \ldots$ of words. The run of $\mathcal{B}$ on γ infinitely alternates between accepting and rejecting states. This contradicts the weakness of $\mathcal{B}$. $\square$