Don't Care Words with an Application to the Automata-based Approach for Real Addition*

(Extended Abstract)**

Jochen Eisinger[1] and Felix Klaedtke[2]

[1] Albert-Ludwigs-Universität Freiburg, Faculty of Applied Sciences, Germany
[2] ETH Zurich, Department of Computer Science, Switzerland

Abstract. Automata are a useful tool in infinite-state model checking, since they can represent infinite sets of integers and reals. However, analogous to the use of BDDs to represent finite sets, the sizes of the automata are an obstacle in the automata-based set representation. In this paper, we generalize the notion of "don't cares" for BDDs to word languages as a means to reduce the automata sizes. We show that the minimal weak deterministic Büchi automaton (WDBA) with respect to a given don't care set, under certain restrictions, is uniquely determined and can be efficiently constructed. We apply don't cares to improve the efficiency of a decision procedure for the first-order logic over the mixed linear arithmetic over the integers and the reals based on WDBAs.

1 Introduction

As Büchi observed almost 50 years ago [8, 9], automata can be used to decide arithmetical theories, like Presburger arithmetic. Roughly speaking, a Presburger arithmetic formula defines a regular language, for which one can build the automaton recursively over the structure of the formula. So, automata are used to represent sets of integers that are definable in Presburger arithmetic. More recently, model checkers for systems with unbounded integers, like FAST [1] and ALV [19] have been developed that use such an automata-based set representation. The use of automata in these model checkers can be compared to the use of BDDs in model checkers for finite state systems, like SMV [17]: automata describe sets of system states. Moreover, automata constructions can be used for computing or overapproximating the set of all reachable states.

Sets of reals can be represented by ω-automata. Boigelot, Jodogne, and Wolper [5] have shown recently that even weak deterministic Büchi automata (WDBAs) suffice to represent the first-order definable sets in $(\mathbb{R}, Z, +, <)$, where Z is the unary predicate stating whether a number is an integer. This result paves the way for a more effective automata-based decision procedure for the first-order logic over $(\mathbb{R}, Z, +, <)$. WDBAs can be handled algorithmically almost

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

** Due to space limitations, proofs are omitted. Details are in the technical report [10].

as efficiently as automata over finite words. For instance, in contrast to Büchi automata, they can be efficiently minimized [16] and they are easy to complement. WDBAs and this logic have a wide range of applications, such as the symbolic verification of linear hybrid automata [3,4]. The automata library LASH [15] provides implementations of all the needed operations for implementing a decision procedure for the first-order logic over $(\mathbb{R}, Z, +, <)$ based on WDBAs.

However, analogous to BDDs, it turns out that a limiting factor in the automata-based representation of potential infinite sets of integers or reals is the size of the automata. In fact, our first results of an automata-based decision procedure for the first-order theory over $(\mathbb{R}, Z, +, <)$ were rather discouraging; even for medium sized formulas the minimal WDBAs were often huge. An analysis of the constructed automata lead to the results presented in this article.

For BDDs, many algorithms and methods have been developed to reduce the BDD sizes, which have improved the performance BDD-based model-checkers. One of these techniques is the use of *don't cares* [12]. Roughly speaking, don't cares are inputs of a combinational circuit for which the circuit output is not specified or irrelevant. The BDD representation of a circuit can be reduced by choosing appropriate output values for the don't care inputs. In this paper, we generalize the notion of don't cares for BDDs to languages. In the most general sense, a don't care set is a language over some alphabet. The set chosen depends on the application domain. The intuition of a don't care word is that it is irrelevant whether this word belongs to a language or not. Adding or removing don't care words to languages can result in smaller automata. A trivial example is where the don't care set consists of all words. In this case we can either add or remove all words and obtain an automaton with a single state. However, usually a don't care set is a proper subset of all words and it is not obvious which of these words must be added or removed to obtain smaller automata. Furthermore, the order in which we add and remove words might lead to different (minimal) automata accepting the same language *modulo* the don't care set. We prove that under certain restrictions on the don't care set, the minimal WDBA is uniquely determined and can be efficiently constructed.

To demonstrate the effectiveness of don't cares for automata, we apply it to the approach for representing and manipulating sets of integers and reals by WDBAs. First, we define a straightforward don't care set when encoding reals by ω-words. Second, we present an automata construction for handling the existential quantification, which becomes more complicated when using don't cares. Third, we show by experiments that introducing don't care sets can reduce the automata sizes significantly in computing and representing sets of integers and reals.

We proceed as follows. In 2, we give preliminaries. In §3, we introduce don't care words and present our general results about don't care sets. In §4, we present an automata construction for projecting sets of reals that are represented by WDBAs modulo a specific set of don't cares. In §5, we report on experimental results. Finally, in §6, we draw conclusions.

2 Preliminaries

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 Deterministic Automata

Let Σ be an alphabet. We denote the set of all finite words over Σ by Σ^* and Σ^+ denotes the set $\Sigma^* \setminus \{\varepsilon\}$, where ε is the empty word. Σ^ω is the set of all ω-words over Σ. The *concatenation* of words is written as juxtaposition. 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 an ω-word $\alpha \in \Sigma^\omega$ as $\alpha(0)\alpha(1)\alpha(2)\dots$, where $w(i)$ and $\alpha(i)$ denote the ith letter of w and α, respectively.

A *deterministic finite automaton* (DFA) $\mathcal{A}$ is a tuple $(Q, \Sigma, \delta, q_{\mathrm{I}}, F)$, where Q is a finite set of states, Σ is an alphabet, $\delta : Q \times \Sigma \to Q$ is the transition function, $q_{\mathrm{I}} \in Q$ is the initial state, and $F \subseteq Q$ is the set of accepting states. A state not in F is a *rejecting* state. The *size* of $\mathcal{A}$ is the cardinality of Q. We write $\mathcal{A}_q$ for the DFA that is identical to $\mathcal{A}$ except that $q \in Q$ is the initial state. We extend δ to the function $\hat{\delta} : Q \times \Sigma^* \to Q$ defined as $\hat{\delta}(q, \varepsilon) := q$ and $\hat{\delta}(q, bu) := \hat{\delta}(\delta(q, b), u)$, where $q \in Q$, $b \in \Sigma$, and $u \in \Sigma^*$. The DFA $\mathcal{A}$ defines the language $L_*(\mathcal{A}) := \{w \in \Sigma^* : \hat{\delta}(q_{\mathrm{I}}, w) \in F\}$.

The state $q \in Q$ is *reachable* from $p \in Q$ if there is a word $w \in \Sigma^*$ such that $\hat{\delta}(p, w) = q$. In the remainder of the text, we assume that every state in an automaton is reachable from its initial state. A *strongly connected component* (SCC) of $\mathcal{A}$ is a set $S \subseteq Q$ such that every $p \in S$ is reachable from every $q \in S$ and S is maximal. For $q \in Q$, $\mathrm{SCC}(q)$ denotes the SCC $S \subseteq Q$ with $q \in S$. We call an SCC S *accepting* if $S \subseteq F$, and *rejecting* if $S \cap F = \emptyset$.

We can view a DFA as a *deterministic Büchi automaton* (DBA). A *run* of the DBA $\mathcal{A}$ on the ω-word $\alpha \in \Sigma^\omega$ is an ω-word $\vartheta \in Q^\omega$ such that $\vartheta(0) = q_{\mathrm{I}}$ and $\vartheta(i+1) = \delta(\vartheta(i), \alpha(i))$, for all $i \in \mathbb{N}$. The run ϑ is *accepting* if $\mathrm{Inf}(\vartheta) \cap F \neq \emptyset$, where $\mathrm{Inf}(\vartheta)$ is the set of states that occur infinitely often in ϑ. The DBA $\mathcal{A}$ defines the ω-language $L_\omega(\mathcal{A}) := \{\alpha \in \Sigma^\omega : \text{the run of } \mathcal{A} \text{ on } \alpha \text{ is accepting}\}$. The DBA $\mathcal{A}$ is *weak* if every SCC of $\mathcal{A}$ is either accepting or rejecting. We use the initialism WDBA for "weak deterministic Büchi automaton." Similarly, we can view a DFA as a *deterministic co-Büchi automaton* (co-DBA). Runs of co-DBAs are defined as for DBAs. A run ϑ of a co-DBA $\mathcal{C}$ is *accepting* if $\mathrm{Inf}(\vartheta) \cap F = \emptyset$, where F is the set of "accepting" states of $\mathcal{C}$. We define $\overline{L}_\omega(\mathcal{C}) := \{\alpha \in \Sigma^\omega : \text{the run of } \mathcal{C} \text{ on } \alpha \text{ is accepting (in the co-Büchi sense)}\}$.

2.2 Representing Sets of Reals with Automata

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

Boigelot, Jodogne, and Wolper have shown in [5] 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 have shown 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 ω-languages from [5]. In the remainder of the text, let $\varrho > 1$ and $\Sigma := \{0, \ldots, \varrho - 1\}$ be fixed. ϱ is called the *base*.

Definition 1. *Let $r \geq 1$.*

1. *V_r denotes the set of all ω-words over the alphabet $\Sigma^r \cup \{\star\}$ of the form $v \star \gamma$, where $v \in (\Sigma^r)^+$ with $v(0) \in \{0, \varrho - 1\}^r$ and $\gamma \in (\Sigma^r)^\omega$.*
2. *An ω-word $v \star \gamma \in \mathsf{V}_r$ represents the vector of reals with r components*

$$\langle\!\langle v \star \gamma \rangle\!\rangle := \sum_{0<i<|v|} \varrho^{|v|-i-1} \cdot v(i) + \sum_{i \geq 0} \varrho^{-i-1} \cdot \gamma(i) + \begin{cases} 0 & \text{if } v(0) = 0, \\ -\varrho^{|v|-1} & \text{if } v(0) = \varrho - 1, \end{cases}$$

   where vector addition and scalar multiplication are componentwise.[3]
3. *For a formula $\varphi(x_1, \ldots, x_r)$, we define $L(\varphi) := \{\alpha \in \mathsf{V}_r : \mathfrak{R} \models \varphi[\langle\!\langle \alpha \rangle\!\rangle]\}$.*

Note that the encoding $v \star \gamma \in \mathsf{V}_1$ of a real is based on the ϱ's complement representation. The symbol $\star$ plays the role of a decimal point, separating the integer part v from the fractional part γ. Moreover, note that every vector in $\mathbb{R}^r$ can be represented by an ω-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 prime factors that are also factors of the base ϱ, has distinct representations, e.g., in base $\varrho = 2$, $\langle\!\langle 0 \star 10^\omega \rangle\!\rangle = \langle\!\langle 0 \star 01^\omega \rangle\!\rangle = \frac{1}{2}$, where b^ω denotes the infinite repetition of the letter b.

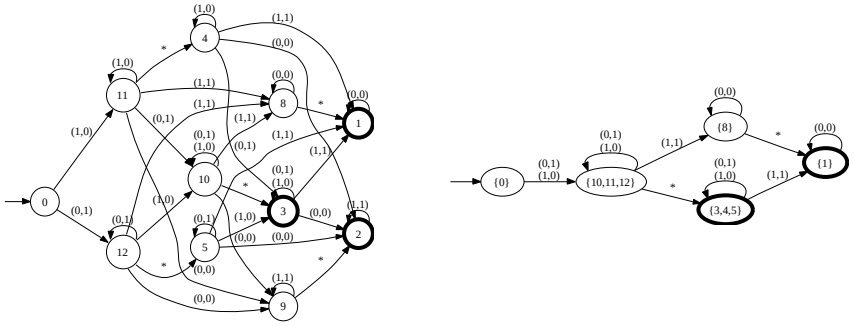
Additional notation. Let $r \geq 1$ and $s, t \in \{1, \ldots, r\}$ with $s \leq t$. We denote the tth coordinate of $b \in \Sigma^r$ by $b_{\restriction t}$ and $b_{\restriction s,t} := (b_{\restriction s}, b_{\restriction s+1}, \ldots, b_{\restriction t})$. We write $\alpha_{\restriction t}$ for the tth *track* of $\alpha \in (\Sigma^r \cup \{\star\})^\omega$, i.e., $\alpha_{\restriction t}$ is the ω-word $\gamma \in (\Sigma \cup \{\star\})^\omega$ defined as $\gamma(i) := \star$ if $\alpha(i) = \star$, and $\gamma(i) := \alpha(i)_{\restriction t}$ otherwise, for $i \in \mathbb{N}$. Analogously, $\alpha_{\restriction s,t}$ denotes the ω-word consisting of the tracks $s, s+1, \ldots, t$ of α. For $m, n \geq 1$ and ω-words $\alpha \in (\Sigma^m \cup \{\star\})^\omega$ and $\beta \in (\Sigma^n \cup \{\star\})^\omega$, we write (α, β) for the ω-word $\gamma \in (\Sigma^{m+n} \cup \{\star\})^\omega$ with $\gamma_{\restriction 1,m} = \alpha$ and $\gamma_{\restriction m+1,m+n} = \beta$. Here, we make the assumption that $\alpha(i) = \star$ iff $\beta(i) = \star$, for all $i \in \mathbb{N}$. We use the same notation for finite words, which is defined analogously.

3 Don't Cares for Optimizing the Real Representation

In this section, we define our optimized representation of the reals as ω-words, which leads us to the general concept of don't care words for ω-languages. We first give a motivating example.

Example 2. Consider the formula $\varphi(x, y) := x \neq 0 \wedge x + y = 0$. The minimal WDBA accepting $L(\varphi)$ in base $\varrho = 2$ is shown in Figure 1(a). This WDBA is rather

[3] Note that we do not distinguish between vectors and tuples.



(a) straightforward encoding (b) optimized encoding

Fig. 1. Minimal WDBAs for the formula $x \neq 0 \wedge x+y=0$. For the sake of readability, we have omitted the rejecting sink states and their incoming transitions.

complex as it must either accept or reject all ω-words that represent the same pair of reals. For instance, the ω-words $\alpha := (1,0)\star(1,0)^\omega$ and $\beta := (0,1)\star(0,1)^\omega$ represent the pair $(0,0)$ of reals, which does not satisfy φ and thus, the WDBA must reject them. In the optimized encoding we exploit that already the ω-word $\gamma := (0,0) \star (0,0)^\omega$ takes care of the fact that the pair of reals $(0,0)$ is not in the represented set. That means, we can add α and β to the ω-language. More general, an ω-word that has a suffix in which at least one of its tracks is of the form 1^ω is treated as a *don't care*, i.e., we can freely chose whether the automaton should accept or reject this ω-word. Observe that for every don't care representing the pair (x, y) of reals, there is an ω-word that also represents (x, y) and is not a don't care.

Consider again the ω-words α and β, which are don't cares. When reading these ω-words, we eventually loop in the states 4 and 5, respectively. Note that all runs that eventually stay in one of these states are don't cares. Making the states 4 and 5 accepting clearly alters the ω-language of the WDBA. However, we only add ω-words that are don't cares, like α and β. If the states 4 and 5 are accepting we can merge them with state 3. Analogously, we can make state 2 rejecting. Then, we can merge the states 2 and 9 with the rejecting sink state. We could also make the states 11 or 12 accepting. However, this would not be beneficial since it will prevent us from merging the states 10, 11, and 12. The resulting minimized automaton is depicted in Figure 1(b).

In the context of encoding reals by ω-words we use the following don't cares.

Definition 3. *Let $r \geq 1$. An ω-word $\alpha \in (\Sigma^r \cup \{\star\})^\omega$ is a* don't care word *if there are $t \in \{1, \ldots, r\}$ and $k \in \mathbb{N}$ such that $\alpha(i) \in \Sigma^r$ and $\alpha(i)_{\restriction t} = \varrho - 1$, for all $i \geq k$. DC_r denotes the set of all don't care words in $(\Sigma^r \cup \{\star\})^\omega$.*

Instead of constructing a WDBA that accepts the ω-language $L(\varphi)$ for a formula φ, we are interested in constructing a WDBA that accepts an ω-language that coincide on all the ω-words in $L(\varphi)$ that are not don't care words. Note that removing or adding *all* don't care words to $L(\varphi)$ does not necessarily result in a smaller automaton. Also note that by removing or adding all don't care words we can obtain ω-languages that are not recognizable by WDBAs.

The following definition generalizes the concept of ω-words for which we "do not care" if they belong to an ω-language or not.

Definition 4. *A* don't care set D *is an* ω*-language over some alphabet* Γ*, and an* ω*-word in* D *is a* don't care word. *For* ω*-languages* $L, L' \subseteq \Gamma^\omega$*, we write* $L \equiv_D L'$ *if* $L \setminus D = L' \setminus D$.

We want to remark that the so-called don't care sets will usually depend on the application context. In our case, the don't care sets DC_r naturally arise from the encoding of the reals in Definition 1.

In the remainder of this section, we present general results about ω-languages with respect to a don't care set D. We focus on ω-languages that can be described by Büchi automata, in particular by WDBAs. In §3.1 and §3.2, we establish some straightforward facts. Namely, in §3.1, we observe that standard automata constructions carry over to handle the Boolean operations when using don't care sets, and in §3.2, we show how to solve the emptiness problem for Büchi automata with respect to an ω-regular don't care set $D \subseteq \Gamma^\omega$. In §3.3, we describe minimization of WDBAs with respect to a don't care set $D \subseteq \Gamma^\omega$, where we assume that D fulfills the two properties: (1) $D \neq \Gamma^\omega$ and (2) $\alpha \in D \Leftrightarrow u\alpha \in D$, for all $u \in \Gamma^*$ and $\alpha \in \Gamma^\omega$. In particular, we show that the minimal WDBA is uniquely determined (up to isomorphism) and we give an efficient algorithm for constructing it under the assumption that D is ω-regular.

3.1 Boolean Operations

The automata construction for Boolean operations, like union and complementation of ω-languages, need not to be changed when using a don't care set $D \subseteq \Gamma^\omega$. For instance, for complementation, if we have that $L \equiv_D L'$, for ω-languages $L, L' \subseteq \Gamma^\omega$, then we have that $\Gamma^\omega \setminus L \equiv_D \Gamma^\omega \setminus L'$. Note that it is irrelevant whether L and L' differ on D, i.e., $L \cap D \neq L' \cap D$.

For WDBAs, we can use the standard product construction for the intersection and union. Let $\mathcal{A} = (Q, \Gamma, \delta, q_\mathrm{I}, F)$ and $\mathcal{B} = (Q', \Gamma, \delta', q'_\mathrm{I}, F')$ be WDBAs. For the intersection, we define $\mathcal{D} := (Q \times Q', \Gamma, \eta, (q_\mathrm{I}, q'_\mathrm{I}), F \times F')$, where $\eta((q, q'), b) := (\delta(q, b), \delta'(q', b))$, for $q \in Q$, $q' \in Q'$, and $b \in \Gamma$. The construction for the union is similar. Complementing WDBAs is done by flipping accepting and rejecting states of a WDBA. We define $\mathcal{C} := (Q, \Gamma, \delta, q_\mathrm{I}, Q \setminus F)$.

Proposition 5. *(a) For the* WDBA $\mathcal{D}$*, it holds that* $L_\omega(\mathcal{D}) \equiv_D L_\omega(\mathcal{A}) \cap L_\omega(\mathcal{B})$. *(b) For the* WDBA $\mathcal{C}$*, it holds that* $L_\omega(\mathcal{C}) \equiv_D \Gamma^\omega \setminus L_\omega(\mathcal{A})$.

3.2 Emptiness Check

The emptiness problem for Büchi automata modulo a don't care set D is to check whether a Büchi automaton $\mathcal{A}$ accepts an ω-word that is not in D. If D is ω-regular, then we can solve this problem by constructing the Büchi automaton accepting $L_\omega(\mathcal{A}) \setminus D$ and check whether the resulting Büchi automaton accepts an ω-word. The complexity is in $\mathrm{O}(n)$, where n is the number of states of $\mathcal{A}$. Note that D is fixed and hence, the size of the Büchi automaton for D is a constant.

3.3 Minimizing WDBAs with Don't Cares

Löding showed in [16] that the minimal WDBA can be constructed in two steps. In the first steps, the WDBA is put in linear time into a normal form by determining a suitable set of accepting states. This step does not change the accepted ω-language, since it only alters the acceptance types of states (rejecting or accepting) that cannot occur infinitely often in a run. In the second step, the WDBA in normal form is minimized by a standard DFA minimization algorithm, like that of Hopcroft [13]. We extend Löding's algorithm to WDBAs such that it takes a don't care set D over the alphabet Γ into account, where we require that (1) $D \neq \Gamma^\omega$ and (2) $\alpha \in D \Leftrightarrow u\alpha \in D$, for all $u \in \Gamma^*$ and $\alpha \in \Gamma^\omega$.

Definition 6. *Let $\mathcal{A} = (Q, \Gamma, \delta, q_\mathrm{I}, F)$ be a* WDBA.
1. *$\mathcal{A}$ is* D-minimal *if there is no smaller* WDBA $\mathcal{B}$ *such that $L_\omega(\mathcal{A}) \equiv_D L_\omega(\mathcal{B})$.*
2. *A state $q \in Q$ is* D-recurrent *if $L_\omega(\mathcal{A}') \setminus D \neq \emptyset$, where $\mathcal{A}'$ is the* WDBA *$(Q, \Gamma, \delta, q, \mathrm{SCC}(q))$. A state is* D-transient *if it is not D-recurrent. An* SCC *is* D-recurrent *if it contains a D-recurrent state, otherwise, it is* D-transient.

Note that an SCC without loops is D-transient. Moreover, note that for the ω-words not in D, it is irrelevant whether a D-transient SCC is accepting or rejecting. Thus, we can make D-transient SCCs accepting or rejecting without altering the accepted ω-language modulo the don't care set D.

Similar to Löding's algorithm, we construct first a suitable set of accepting states by determining the acceptance types of D-transient states optimal in the sense that applying a minimization algorithm for DFAs yields the minimal WDBA with respect to the don't care set D. We need the following definitions.

Definition 7. *Let $\mathcal{A} = (Q, \Gamma, \delta, q_\mathrm{I}, F)$ be a* WDBA.
1. *A mapping $c : Q \to \mathbb{N}$ is a* D-coloring *for $\mathcal{A}$ if the two conditions hold:*
   - *$c(q)$ is even $\Leftrightarrow q \in F$, for every D-recurrent state $q \in Q$, and*
   - *$c(p) \leq c(q)$, for all $p, q \in Q$ and $b \in \Gamma$ with $\delta(p, b) = q$.*

   The D-coloring c is k-maximal, *where $k \in \mathbb{N}$, if $c(q) \leq k$ and $c'(q) \leq c(q)$, for every $q \in Q$ and every D-coloring $c' : Q \to \mathbb{N}$ for $\mathcal{A}$.*
2. *$\mathcal{A}$ is in* D-normal form *if for some even $k \in \mathbb{N}$, there is a k-maximal D-coloring $c : Q \to \mathbb{N}$ such that $F = F_c$, where $F_c := \{q \in Q : c(q)$ is even$\}$.*[4]

The algorithm in Figure 2 computes the D-normal form of a given WDBA $\mathcal{A} = (Q, \Gamma, \delta, q_\mathrm{I}, F)$. The main task of the algorithm is to compute a k-maximal coloring for $\mathcal{A}$, where k is even and large enough. This is done by looking at the acyclic SCC graph of $\mathcal{A}$, which the algorithm traverses in a reversed topological ordering (lines 4–19). The SCC graph and the topological ordering can be computed in linear time. Observe that the states in an SCC have the same color in a D-coloring. In the ith traversal of the for-loop (lines 4–19), we color the states in the ith SCC with respect to the reversed topological ordering, where the states in the successor SCCs are already colored. If there are no successor SCCs,

[4] Alternatively, we could require that k has to be odd. But we must fix some parity in order to obtain a canonical form for D-minimal WDBAs in D-normal form.

1: Compute the SCC graph G of $\mathcal{A}$.
2: Compute a topological ordering $v_1, \ldots, v_m$ on the vertices of G. To simplify notation, we identify a vertex v_i with its corresponding SCC, i.e., a set of states.
3: Let $k \geq m$ be an even number.
4: **for** $i = m$ **downto** 1 **do** /* *Compute a k-maximal D-coloring $c : Q \to \mathbb{N}$* */
5: **if** v_i has no successors **and** v_i is accepting **then**
6: Define $c(q) := k$, for all $q \in v_i$.
7: **else if** v_i has no successors **and** v_i is rejecting **then**
8: Define $c(q) := k - 1$, for all $q \in v_i$.
9: **else**
10: Let $\ell := \min\{c(q) : v_j \text{ is a successor of } v_i \text{ and } q \in v_j\}$.
11: **if** v_i is D-transient **then**
12: Define $c(q) := \ell$, for all $q \in v_i$.
13: **else if** (ℓ is even **and** v_i is accepting) **or** (ℓ is odd **and** v_i is rejecting) **then**
14: Define $c(q) := \ell$, for all $q \in v_i$.
15: **else**
16: Define $c(q) := \ell - 1$, for all $q \in v_i$.
17: **end if**
18: **end if**
19: **end for**
20: Return the WDBA $\mathcal{A}' := (Q, \Gamma, \delta, q_{\mathrm{I}}, F_c)$.

Fig. 2. Algorithm for computing the D-normal form of a WDBA $\mathcal{A} = (Q, \Gamma, \delta, q_{\mathrm{I}}, F)$.

we assign the maximal color to the states depending on k and their acceptance type (lines 5–8). Note that an SCC with no successors cannot be D-transient, since $D \neq \Gamma^\omega$. If the SCC has successors, the maximal color for the states in this SCC depends on the minimal color ℓ of the successor SCCs (line 10). If the SCC is D-transient (lines 11–12) then ℓ is the maximal color we can assign to these states. Depending on ℓ, the states in the SCC will then be either accepting or rejecting in the resulting WDBA. If the SCC is D-recurrent, the coloring has to preserve the acceptance type of the states in the SCC. Depending on ℓ, we assign the maximal possible color to the states in the SCC (lines 13–15).

In line 11 of the algorithm, we must check whether an SCC S is D-transient. This can be done by checking whether $L_\omega(\mathcal{C}) \subseteq D$ holds, where $\mathcal{C}$ is the WDBA $(Q, \Sigma, \delta, q, S)$ and q is an arbitrarily chosen state in S. Note that $L_\omega(\mathcal{C}) \subseteq D$ iff $L_\omega(\mathcal{C}) \cap (\Gamma^\omega \setminus D) = \emptyset$. Under the assumption that D is ω-regular, it is easy to see that $L_\omega(\mathcal{C}) \cap (\Gamma^\omega \setminus D) = \emptyset$ can be checked in time $\mathrm{O}(|S|)$, since D is fixed and we can construct a Büchi automaton for the ω-language $\Gamma^\omega \setminus D$ in a preprocessing step. In summary, the checks performed in line 11 take time $\mathrm{O}(\sum_{S \text{ SCC of } \mathcal{A}} |S|) = \mathrm{O}(|Q|)$. So, if D is ω-regular, the algorithm in Figure 2 computes a k-maximal coloring in linear time.

Lemma 8. *For a given* WDBA $\mathcal{A} = (Q, \Gamma, \delta, q_{\mathrm{I}}, F)$*, there is a set* $F' \subseteq Q$ *such that the* WDBA $\mathcal{A}' := (Q, \Gamma, \delta, q_{\mathrm{I}}, F')$ *is in D-normal form and* $L_\omega(\mathcal{A}) \equiv_D L_\omega(\mathcal{A}')$*. The set F' can be constructed in time* $\mathrm{O}(|Q|)$ *if D is ω-regular.*

Our minimization algorithm for WDBAs with the don't care set D is as follows: First, we put the given WDBA into D-normal form. Second, we apply to the WDBA in D-normal form the classical DFA minimization algorithm [13]. The

overall complexity is in $O(n \log n)$, where n is the size of $\mathcal{A}$. This algorithm returns the unique minimal WDBA for the don't care set D.

Theorem 9. *For a given* WDBA $\mathcal{A} = (Q, \Gamma, \delta, q_I, F)$*, there is a* D*-minimal* WDBA $\mathcal{A}'$ *with* $L_\omega(\mathcal{A}) \equiv_D L_\omega(\mathcal{A}')$*.* $\mathcal{A}'$ *can be constructed in time* $O(|Q| \log |Q|)$ *if* D *is* ω*-regular. Furthermore, every* D*-minimal* WDBA $\mathcal{B}$ *in* D*-normal form with* $L_\omega(\mathcal{A}) \equiv_D L_\omega(\mathcal{B})$ *is isomorphic to* $\mathcal{A}'$*.*

Remark 10. Similar to Definition 3, we can define for $r \geq 1$, the set I_r that consists of the ω-words over $\Sigma^r \cup \{\star\}$ that are not periodic in at least one track. Note that such a periodic track, if it is also in V_1, corresponds to an irrational number. Obviously, I_r has the properties (1) and (2). The decision procedure for the first-order logic over $\mathfrak{R}$ using WDBAs given in [5] can be understood as an automata-based decision procedure for the first-order logic over $(\mathbb{Q}, Z, +, <)$ using WDBAs with the don't care sets I_r. Note that the ω-languages definable in the first-order logic over $(\mathbb{Q}, Z, +, <)$ are in general not ω-regular using the encoding in Definition 1.2. From this point of view, we see that WDBAs modulo don't care sets can describe non-ω-regular languages and in this case, they even have a canonical minimal form (Theorem 9). Analogously, WDBAs with the don't care sets DC_r can describe ω-regular languages that are not in the Borel class $F_\sigma \cap G_\delta$, which exactly captures the expressive power of WDBAs [18]. Furthermore, by Theorem 9, the ω-words in DC_r that have to be added to or removed from the ω-language are uniquely determined in order to obtain the minimal WDBA for the ω-language modulo the don't care set DC_r.

4 Quantification for the Reals

In this section, we give an automata construction for WDBAs that handles the quantification in the first-order logic over $\mathfrak{R}$ when using the don't care sets DC_r.

Roughly speaking, for the straightforward encoding, the existential quantification is done by eliminating the track of the quantified variable in the transitions of the WDBA.[5] Intuitively, this nondeterministic automaton guesses the digits of the quantified variable. As explained in [5], we can determinize this automaton by using the breakpoint construction for weak co-Büchi automata (see [14]). The construction for handling the existential quantification that we present in this subsection for the optimized encoding is also based on the breakpoint construction. However, the construction is more subtle because of the following problem: Assume that $\mathcal{A}$ is a WDBA for the formula $\varphi(x_1, \ldots, x_r)$, i.e., $L_\omega(\mathcal{A}) \equiv_{\mathsf{DC}_r} L(\varphi)$. Eliminating the track of the variable x_r results in a nondeterministic Büchi automaton that might accept ω-words $\alpha \notin \mathsf{DC}_{r-1}$ for which there is only an ω-word $\gamma \in \mathsf{DC}_1$ such that $(\alpha, \gamma) \in L_\omega(\mathcal{A})$. A WDBA for $\exists x_r \varphi$ must not accept such ω-words α. A concrete instance of this problem is given in the example:

Example 11. Consider again the formula $\varphi(x, y) := x \neq 0 \wedge x + y = 0$ and the WDBA in Figure 1(b) from Example 2. Eliminating the x-track, i.e., the

[5] Some additional work is needed for the sign bit, see, e.g., [5,6] for details.

first track, yields a nondeterministic Büchi automaton that accepts the ω-word $0 \star 0^\omega$, since we can infinitely loop in state $q := \{3, 4, 5\}$ by reading the letter 0. However, $\mathfrak{R} \not\models \exists x \varphi[\langle\!\langle 0 \star 0^\omega \rangle\!\rangle]$. Here, the problem is that the only ω-word γ such that $(\gamma, 0 \star 0^\omega)$ is accepted by the WDBA in Figure 1(b) is the don't care word $1 \star 1^\omega$. On the one hand, for the ω-word $0 \star 0^\omega$ the state q has to be rejecting. On the other hand, for the ω-word $0 \star (10)^\omega$ the state q has to be accepting.

Before we present our construction, we remark that removing all don't care words from the ω-language of the given WDBA before applying the construction in [5] for handling the existential quantification does not work. The reason is that the resulting DBA is not necessarily *weak* and hence, we cannot longer apply the breakpoint construction after eliminating the track of the quantified variable.

Assume that $\mathcal{A} = (Q, \Sigma^r \cup \{\star\}, \delta, q_\mathrm{I}, F)$ is a WDBA for the formula φ with r free variables, i.e., $L_\omega(\mathcal{A}) \equiv_{\mathsf{DC}_r} L(\varphi)$. We divide the construction of the WDBA for $\exists x_i \varphi$ into two steps. First, we construct from $\mathcal{A}$ a co-DBA $\mathcal{B}$ that accepts an ω-language for $\exists x_i \varphi$, i.e., $\overline{L}_\omega(\mathcal{B}) \equiv_{\mathsf{DC}_{r-1}} L(\exists x_i \varphi)$. Second, we show that $\mathcal{B}$ can be easily turned into a WDBA. To simplify notation, we assume without loss of generality that $i = r$ and $L_\omega(\mathcal{A}) \subseteq \mathsf{V}_r$.

To define $\mathcal{B}$'s transition function, we need the following definitions. For $u \in \Sigma^+$ with $u(0) \in \{0, \varrho - 1\}$, we define

$$\overline{u} := \begin{cases} 0^n & \text{if } u = (\varrho-1)^n \text{ with } n > 0, \\ 010^n & \text{if } u = 0(\varrho-1)^n \text{ with } n \geq 0, \\ v(c+1)0^n & \text{if } u = vc(\varrho-1)^n \text{ with } v \in \Sigma^+,\ c \in \Sigma \setminus \{\varrho-1\}, \text{ and } n \geq 0. \end{cases}$$

Note that $\langle\!\langle u(\varrho-1)^n \star (\varrho-1)^\omega \rangle\!\rangle = \langle\!\langle \overline{u}0^n \star 0^\omega \rangle\!\rangle$, for all $n \geq 0$ and $u \in \Sigma^+$ with $u(0) \in \{0, \varrho - 1\}$. We define the relation $M \subseteq Q \times Q$ by pMq iff $p \in F$ and for every $\alpha \in (\Sigma^{r-1})^\omega \setminus \mathsf{DC}_{r-1}$, it holds that $(\alpha, (\varrho-1)^\omega) \in L_\omega(\mathcal{A}') \Rightarrow (\alpha, 0^\omega) \in L_\omega(\mathcal{A}_q)$, where $\mathcal{A}'$ is the WDBA $(Q, \Sigma^r \cup \{\star\}, \delta, p, \mathrm{SCC}(p))$.

Intuitively, the construction works as follows. As in the breakpoint construction, $\mathcal{B}$ has states of the form (R, S). Roughly speaking, in the first component we collect $\mathcal{A}$'s states that are reached by guessing the digits of the variable x_r. The second component checks whether we eventually stay in an accepting SCC of $\mathcal{A}$. In contrast to the breakpoint construction, R and S are not only subsets of Q but sets of pairs of states of $\mathcal{A}$. The reason for using pairs of states is the following. Assume that we reach the pair (R, S) from $\mathcal{B}$'s initial state by reading a finite prefix of an ω-word $\gamma \in \mathsf{V}_{r-1} \setminus \mathsf{DC}_{r-1}$. For $(p, q) \in R$, we have that p is reached by guessing a finite prefix of the digits of a real number for the quantified variable x_r. However, the guessed digits u could be a finite prefix of a don't care word $\alpha \in \mathsf{DC}_1 \cap \mathsf{V}_1$. Suppose that we visit p infinitely often when reading (γ, α). If p is accepting, $\mathcal{A}$ accepts (γ, α). However, since (γ, α) is a don't care word, $\langle\!\langle \alpha \rangle\!\rangle$ is not necessarily a real number such that $\mathfrak{R} \models \varphi[\langle\!\langle \gamma \rangle\!\rangle, \langle\!\langle \alpha \rangle\!\rangle]$. In order to detect such a case, we use the state q and the relation M. The state q is the state that is reached when guessing the corresponding digits for u of the ω-word $\beta \in \mathsf{V}_1 \setminus \mathsf{DC}_1$ such that $\langle\!\langle \alpha \rangle\!\rangle = \langle\!\langle \beta \rangle\!\rangle$. If pMq holds, then we know that $\mathfrak{R} \models \varphi[\langle\!\langle \gamma \rangle\!\rangle, \langle\!\langle \alpha \rangle\!\rangle]$, since $\langle\!\langle \alpha \rangle\!\rangle = \langle\!\langle \beta \rangle\!\rangle$ and $\mathcal{A}$ also accepts β. Hence, p is rightly an

accepting state for the prefix of γ we have read so far. In the case where pMq does not hold, we have to treat p as a rejecting state.

Formally, $\mathcal{B}$ is the co-DBA $(\{q'_{\mathrm{I}}\} \cup (K \times K), \Sigma^{r-1} \cup \{\star\}, \eta, q'_{\mathrm{I}}, K \times \{\emptyset\})$, where $K := \mathcal{P}(Q \times Q)$, q'_{I} is a fresh state and η is defined as follows. For the initial state, we define $\eta(q'_{\mathrm{I}}, b) := (\emptyset, \emptyset)$, for $b \notin \{0, \varrho-1\}^{r-1}$, and for $b \in \{0, \varrho-1\}^{r-1}$, we define $\eta(q'_{\mathrm{I}}, b) := (I(b), \emptyset)$, where

$$I(b) := \left\{ \left(\hat{\delta}(q_{\mathrm{I}}, (b^{|u|}, u)), \hat{\delta}(q_{\mathrm{I}}, (b^{|\overline{u}|}, \overline{u}))\right) \,:\, u \in \Sigma^+ \text{ with } u(0) \in \{0,1\} \right\}.$$

For a state $(R, S) \in K \times K$ and $b \in \Sigma^{r-1}$, we define

$$\eta\big((R,S), b\big) := \begin{cases} (R', R' \cap M) & \text{if } S = \emptyset, \\ (R', S' \cap M) & \text{if } S \neq \emptyset, \end{cases}$$

where

$$\begin{aligned} R' := {} & \left\{ \big(\delta(p, (b, \varrho-1)), \delta(q, (b, 0))\big) \,:\, (p,q) \in R \right\} \cup \\ & \left\{ \big(\delta(p, (b, c)), \delta(p, (b, c+1))\big) \,:\, (p,q) \in R \text{ and } c \in \Sigma \setminus \{\varrho-1\} \right\}, \text{ and} \\ S' := {} & \left\{ \big(\delta(p, (b, \varrho-1)), \delta(q, (b, 0))\big) \,:\, (p,q) \in S \right\} \cup \\ & \left\{ \big(\delta(p, (b, c)), \delta(p, (b, c+1))\big) \,:\, (p,q) \in S \text{ and } c \in \Sigma \setminus \{\varrho-1\} \right\}. \end{aligned}$$

Finally, $\eta((R,S), \star) := (R', R' \cap M)$, where $R' := \{(\delta(p, \star), \delta(q, \star)) \,:\, (p,q) \in R\}$.

Lemma 12. *It holds that* $\overline{L}_\omega(\mathcal{B}) \equiv_{\mathsf{DC}_{r-1}} L(\exists x_r \varphi)$.

An SCC of $\mathcal{B}$ might contain accepting and rejecting states. The next lemma shows that if an SCC of $\mathcal{B}$ contains accepting and rejecting states then we can make all states in this SCC accepting. Given this, it is easy to turn the co-DBA $\mathcal{B}$ into a WDBA $\mathcal{A}'$ for $L(\exists x_r \varphi)$, i.e., $L_\omega(\mathcal{A}') \equiv_{\mathsf{DC}_{r-1}} L(\exists x_r \varphi)$.

Lemma 13. *Let $\psi(y_1, \ldots, y_s)$ be a formula and let $\mathcal{C} = (P, \Sigma^s \cup \{\star\}, \mu, p_{\mathrm{I}}, E)$ be a co-DBA with $\overline{L}_\omega(\mathcal{C}) \equiv_{\mathsf{DC}_s} L(\psi)$. If $S \subseteq P$ is an SCC with $S \cap E \neq \emptyset$ then $\overline{L}_\omega(\mathcal{C}') \equiv_{\mathsf{DC}_s} L(\psi)$, where $\mathcal{C}'$ is the co-DBA $(P, \Sigma^s \cup \{\star\}, \mu, p_{\mathrm{I}}, E \cup S)$.*

The above given construction yields a WDBA that has $1 + 2^{2 \cdot |Q|^2}$ states. However, some of the states are not reachable from the initial state q'_{I}, e.g., the states $(R, S) \in K \times K$ with $S \not\subseteq R$ are never reachable from q'_{I}. Next, we briefly discuss the auxiliary computations involved in the construction.

For the transitions from the initial state q'_{I}, we need to compute the sets $I(b)$, for every $b \in \Sigma^{r-1}$. Computing $I(b)$ separately for each $b \in \Sigma^{r-1}$, yields an algorithm that is exponential in r and is not practical. The algorithm described in [6] for determining the initial transitions of DFAs for quantifying Presburger arithmetic formulas, can be adopted to our construction and it works well in practice, although it has exponential worst case complexity in r.

For computing the relation M, we define the WDBAs $\mathcal{G} := (Q, \Sigma^{r-1}, \delta_1, q_{\mathrm{I}}, F)$ and $\mathcal{H} := (Q, \Sigma^{r-1}, \delta_2, q_{\mathrm{I}}, F)$, where $\delta_1(q, b) := \delta(p, (b, \varrho-1))$ and $\delta_2(q, b) := \delta(p, (b, 0))$, for $q \in Q$ and $b \in \Sigma^{r-1}$. For states $p, q \in Q$, we have that pMq iff (1) $p \in F$ and (2) $L_\omega(\mathcal{G}') \cap L_\omega(\mathcal{H}_q)$ contains an ω-word not in DC_{r-1}, where $\mathcal{G}'$ is the WDBA $(Q, \Sigma^{r-1}, \delta_1, p, \mathrm{SCC}(p))$. Since the SCC of p consists of at most $|F|$ states, condition (2) can be checked in time $\mathrm{O}(|Q| \cdot |F|)$, see §3.1 and §3.2. An upper bound for computing M is $\mathrm{O}(|Q|^2 \cdot |F|^2)$, since the first component in M has to be a state in F.

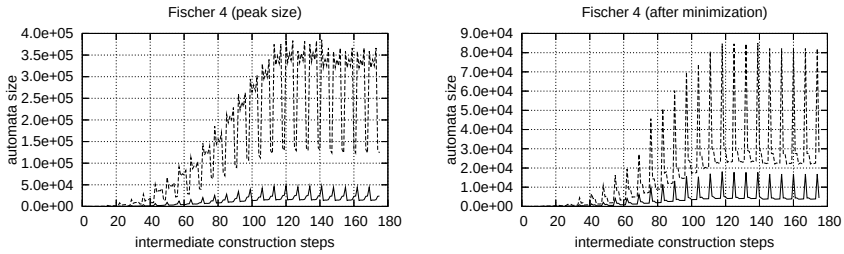


Fig. 3. Automata sizes encountered during the computation for Fischer's protocol with 4 processes. The solid (dashed) lines correspond to the optimized (straightforward) encoding. The intermediate construction steps correspond to the flows and jumps of the processes in Fischer's protocol. We obtain similar results for the other protocols.

5 Experimental Results

In this section, we report on experimental results obtained from our prototype implementation of an automata-based decision procedure for the first-order logic over $\mathfrak{R}$.[6] We want to point out that in our implementation we only used the don't care sets DC_r (Definition 3). We have carried out tests on two different classes of problems: (1) randomly generated formulas and (2) the iterative computation of the reachable states of infinite-state systems. In the later case, we mainly focus on the sizes of the automata, as our prototype is not intended to compete with optimized tools for solving the reachability problem.

Random Formulas. We have applied our prototype to randomly generated formulas. For a test set of 100 formulas with 4 variables with about 10 disjunctions and conjunctions each, the savings in terms of automata sizes encountered during the construction are observable (on average 8.4%), although moderate. Our new construction for the quantification generates larger automata (on average 40.1%), however, after normalization and minimization the resulting automata with don't care sets are smaller (on average 7.7%). Our prototype requires up to one order of magnitude more runtime for the quantification when using don't care words. When restricting the 4 variables to the integer domain, the savings due to the don't care set become more substantial (on average 48.5%), as every integer has encodings that are in the don't care set. In comparison to an implementation based on LASH [15] without don't cares, our prototype is faster. The marginal difference in performance on small quantifier free formulas grows rapidly when the formulas contain quantifiers or have more variables.

Reachability Analysis. Infinite-state systems, like systems with unbounded integers or linear hybrid automata can be analyzed symbolically in the first-order logic over $\mathfrak{R}$. We have analyzed the Bakery protocol, Fischer's protocol, and the railroad crossing example [11]. Using don't care words, the automata constructed

[6] Our prototype is publicly available online at `http://www.informatik.uni-freiburg.de/~eisinger/research/rva.html`.

		with don't cares			without don't cares		
	iterations	**peak**	**final**	**runtime**	**peak**	**final**	**runtime**
Fischer 2	9	238	53	43.98s	2,318	182	49.52s
Fischer 3	15	44,631	405	164.75s	90,422	2,045	184.59s
Fischer 4	21	51,676	4,377	2,739.58s	417,649	27,548	4,353.66s
Fischer 5	27	145,629	55,885	20,972.79s	1625,141	430,727	53,940.37s
Railroad	8	152,826	7,735	1,594.32s	365,004	9,411	1,080.24s
Bakery 2	30	107	-	52.42s	557	-	63.64s
Bakery 3	30	314	-	107.74s	2,010	-	121.09s
Bakery 4	30	909	-	201.41s	8,883	-	272.70s

Fig. 4. Iterations required to reach the fixpoint of the reachable state set for several infinite-state systems, construction times, and peak and final automata sizes. Note that the fixpoint for Bakery cannot be reached using our naive fixpoint computation.

during the iterative computation of the reachable states become smaller by an order of magnitude (see Figures 3 and 4). This saving can be explained by the following two observations. First, the formulas that describe the transitions of a system contain many variables (the formulas for Fischer's protocol with 5 processes have 34 variables). Note that the don't care sets contain more words if the formula contains many free variables. Second, the construction of the reachable state set requires a large number of automata constructions. Although the saving in a single automata construction might be small, the overall saving grows with the number of automata constructions.

6 Conclusions

We generalized the concept of *don't cares* for BDDs to automata and demonstrated that don't cares are effective in reducing the automata sizes. On the one hand, we were able to prove rather general results about don't cares sets, like the minimization of WDBAs. On the other hand, we presented an automata construction for the quantification in the first-order logic $\mathfrak{R}$, which depends on the used don't care set. We demonstrated the potential of don't cares by a prototype.

Related to our work is [2] on widening sets of integers that are represented by automata. In order to obtain always an overapproximation of a set, widening an automaton represented set only adds words to the language. In contrast, we allow words to be removed, and adding or removing don't care words still yields an exact automata-based representation of a set. Moreover, for the sets of vectors of reals, we used a don't care set for which the automata-based set representation is still unique. We want to point out that the widening method [2] is complementary to don't care words and hence, they can be combined in infinite-state model checkers that use an automata-based representation for the reachable states of a system. Analogously, don't care words are complementary to acceleration techniques like [7]. However, further work is needed in combining these techniques, since the automata constructions might need some adjustment to work also for don't care words (see, e.g., the automata construction in §4).

Future work also includes improving the mechanization of the automata construction for handling the existential quantification in the first-order logic over $\mathfrak{R}$, which is currently the bottleneck in our prototype. Another direction we want to pursue is to exploit don't cares further. For example, for carrying out the quantification of x in the second disjunct of the formula $\psi(\overline{y}) \vee \exists x \varphi(x, \overline{y})$, we can use the language of the automaton for ψ as a don't care set for making the automaton for φ smaller before we apply the construction for the existential quantification. Overall, we believe that don't care words have a large potential for making automata-based model checking more effective.

References

1. S. Bardin, A. Finkel, J. Leroux, and L. Petrucci, *FAST: Fast acceleration of symbolic transition systems*, in CAV'03, LNCS 2725, pp. 118–121.
2. C. Bartzis and T. Bultan, *Widening arithmetic automata*, in CAV'04, LNCS 3114, pp. 321–333.
3. B. Boigelot, L. Bronne, and S. Rassart, *An improved reachability analysis method for strongly linear hybrid systems*, in CAV'97, LNCS 1254, pp. 167–178.
4. B. Boigelot, F. Herbreteau, and S. Jodogne, *Hybrid acceleration using real vector automata*, in CAV'03, LNCS 2725, pp. 193–205.
5. B. Boigelot, S. Jodogne, and P. Wolper, *An effective decision procedure for linear arithmetic over the integers and reals*, ACM ToCL, 6 (2005), pp. 614–633.
6. B. Boigelot and L. Latour, *Counting the solutions of Presburger equations without enumerating them*, TCS, 313 (2004), pp. 17–29.
7. B. Boigelot, A. Legay, and P. Wolper, *Omega-regular model checking*, in TACAS'04, LNCS 2988, pp. 561–575.
8. J. Büchi, *Weak second-order arithmetic and finite automata*, Zeitschrift der mathematischen Logik und Grundlagen der Mathematik, 6 (1960), pp. 66–92.
9. ———, *On a decision method in restricted second order arithmetic*, in Logic, Methodology and Philosophy of Science, Stanford University Press, 1962, pp. 1–11.
10. J. Eisinger and F. Klaedtke, *Don't care words with an application to the automata-based approach for real addition*, Tech. Rep. 223, Institut für Informatik, Albert-Ludwigs-Universität Freiburg, 2006.
11. T. Henzinger, *The theory of hybrid automata*, in LICS'96, pp. 278–292.
12. Y. Hong, P. A. Beerel, J. R. Burch, and K. L. McMillan, *Safe BDD minimization using don't cares*, in DAC'97, ACM Press, pp. 208–213.
13. J. E. Hopcroft, *An $n \log n$ algorithm for minimizing the states in a finite automaton*, in Theory of Machines and Computations, 1971, pp. 189–196.
14. O. Kupferman and M. Vardi, *Weak alternating automata are not that weak*, ACM ToCL, 2 (2001), pp. 408–429.
15. LASH, *The Liège Automata-based Symbolic Handler*. `http://www.montefiore.ulg.ac.be/~boigelot/research/lash/`.
16. C. Löding, *Efficient minimization of deterministic weak ω-automata*, IPL, 79 (2001), pp. 105–109.
17. K. L. McMillan, *Symbolic Model Checking*, Kluwer Academic Publishers, 1993.
18. L. Staiger and K. Wagner, *Automatentheoretische und automatenfreie Charakterisierungen topologischer Klassen regulärer Folgenmengen*, Elektronische Informationsverarbeitung und Kybernetik, 10 (1974), pp. 379–392.
19. T. Yavuz-Kahveci, C. Bartzis, and T. Bultan, *Action language verifier, extended*, in CAV'05, LNCS 3576, pp. 413–417.