Improved Ramsey-Based Büchi Complementation

Stefan Breuers, Christof Löding, and Jörg Olschewski

RWTH Aachen University

Abstract. We consider complementing Büchi automata by applying the Ramsey-based approach, which is the original approach already used by Büchi and later improved by Sistla et al. We present several heuristics to reduce the state space of the resulting complement automaton and provide experimental data that shows that our improved construction can compete (in terms of finished complementation tasks) also in practice with alternative constructions like rank-based complementation. Furthermore, we show how our techniques can be used to improve the Ramsey-based complementation such that the asymptotic upper bound for the resulting complement automaton is $2^{\mathcal{O}(n \log n)}$ instead of $2^{\mathcal{O}(n^2)}$.

1 Introduction

The aim of this paper is to present several techniques to improve the Ramsey-based approach to complementation of Büchi automata, which was originally used by Büchi when he introduced this model of automata on infinite words to show the decidability of monadic second order logic over the successor structure of the natural numbers [2]. The method is called Ramsey-based because its correctness relies on a combinatorial result by Ramsey [10] to obtain a periodic decomposition of the possible behaviors of a Büchi automaton on an infinite word. Starting from a Büchi automaton with n states the construction yields a complement automaton with $2^{\mathcal{O}(n^2)}$ states [13]. A non-trivial lower bound of $n!$ for the complementation of Büchi automata was shown by Michel in [8]. The gap between the lower and the upper bound was made tighter by a determinization construction presented by Safra [11] from which a complementation construction with upper bound $2^{\mathcal{O}(n \log n)}$ could be derived. An elegant and simple complementation construction achieving this upper bound was presented by Klarlund in [7] using progress measures, now also referred to as ranking functions. Based on Klarlund's construction the gap between upper and lower bound has been tightened by Friedgut, Kupferman and Vardi [5] and later even further by Schewe [12], leaving only a polynomial gap compared to the improved lower bound that was obtained by Yan [16]. Besides the optimizations of the rank-based approach, a different construction has been developed by Kähler and Wilke in [6], usually called slice-based complementation, and the determinization construction of Safra has been optimized by Piterman [9].

Given all these different constructions, there has recently been an increased interest in experimental evaluations of the different complementation methods.

L. Birkedal (Ed.): FOSSACS 2012, LNCS 7213, pp. 150–164, 2012.

In experimental studies conducted by Tsai et al. [15], it turned out that the Ramsey-based approach was not only inferior to the more modern approaches (determinization-based, rank-based, and slice-based) in terms of the size of the resulting complement automata, but that in most cases its implementation did not even terminate within the imposed time and memory bounds.

Though performing inadequately for complementation, the Ramsey-based approach is used in two other fields, namely universality checking and inclusion checking [4,1]. For these two purposes specific optimization techniques have been developed.

Since the Ramsey-based approach to complementation has got an appealingly simple structure and is nice to teach, our aim is to improve it on both the practical level by using heuristics to reduce the size of the complement automata, and the theoretical level by using the ideas underlying our heuristics to obtain a general optimization of the method that meets the upper bound of $2^{\mathcal{O}(n \log n)}$.

We have implemented these heuristics [17] and conduct experiments on a large set of example automata, showing that the improved construction can compete with other methods for complementation.

The remainder of the paper is structured as follows. In Section 2 we start with some basic definitions and in Section 3 we repeat the original Ramsey-based complementation method. In Section 4 we present our heuristics that are used in the implementation, and we discuss our experimental results. Finally, in Section 5 we show how the ideas from Section 4 can be used to obtain an improvement of the Ramsey-based construction also on a theoretical level.

2 Preliminaries

The set of infinite words over an alphabet Σ is denoted by Σ^ω. For an infinite word $\alpha = a_0 a_1 \cdots$, with $\alpha[i]$ we denote the letter a_i at position i.

An *automaton* is a tuple $\mathcal{A} = \langle Q, \Sigma, q_0, \delta, F \rangle$ where Q is a finite set of states, Σ is a finite alphabet, $q_0 \in Q$ is the initial state, $\delta \colon Q \times \Sigma \to 2^Q$ is the transition function and $F \subseteq Q$ is the set of accepting states. The transition function δ can be extended to a function $\delta^* \colon 2^Q \times \Sigma^* \to 2^Q$ on subsets of the state set and on words in the natural way. By RSC denote the set of all subsets of states that are reachable by the subset construction, i.e. $\mathrm{RSC} := \{P \subseteq Q \mid \exists w \in \Sigma^* \colon \delta^*(\{q_0\}, w) = P\}$. $\mathcal{A}$ is called *deterministic* if $|\delta(q, a)| \leq 1$ for all $q \in Q$ and $a \in \Sigma$.

For automata we consider two different semantics: the usual NFA semantics, in which the automaton accepts a language of finite words, and the Büchi semantics, in which the automaton accepts a language of infinite words. This is made precise in the following definitions.

A *run* of $\mathcal{A}$ on a word $\alpha \in \Sigma^\omega$ is an infinite sequence of states $\rho = p_0, p_1, \ldots$ such that $p_0 = q_0$, and $p_{i+1} \in \delta(p_i, \alpha[i])$ for all $i \geq 0$. A run ρ is *Büchi-accepting* if $\rho(i) \in F$ for infinitely many $i \geq 0$. A *path* from p to q in $\mathcal{A}$ of a word $u = a_1 \cdots a_n \in \Sigma^*$ is a finite sequence of states $p_0, \ldots, p_n$ such that $p = p_0$, $q = p_n$, and $p_i \in \delta(p_{i-1}, a_i)$ for all $1 \leq i \leq n$. The path is *NFA-accepting* if $p_0 = q_0$ and $p_n \in F$.

The languages recognized by an automaton $\mathcal{A}$ are

$$\mathrm{L}_*(\mathcal{A}) \coloneqq \{w \in \Sigma^* \mid \text{there is an NFA-accepting path of } w \text{ in } \mathcal{A}\} \text{ and}$$
$$\mathrm{L}_\omega(\mathcal{A}) \coloneqq \{\alpha \in \Sigma^\omega \mid \text{there is a Büchi-accepting run of } \mathcal{A} \text{ on } \alpha\}.$$

If there is a path from p to q of u in $\mathcal{A}$, then we denote this fact by $p \xrightarrow{u} q$. If furthermore there is a path from p to q of u in $\mathcal{A}$ that visits a final state, then we additionally denote this by $p \xrightarrow[F]{u} q$ (in particular, this is the case if p or q is final).

3 The Ramsey-Based Approach

Throughout this paper, $\mathcal{A} = \langle Q, \Sigma, q_0, \delta, F\rangle$ is a fixed automaton. Our goal is to complement this automaton regarding the Büchi acceptance condition, i.e. we want to obtain an automaton $\mathcal{A}'$ with $\mathrm{L}_\omega(\mathcal{A}') = \Sigma^\omega \setminus \mathrm{L}_\omega(\mathcal{A})$. In the Ramsey-based approach the complement automaton basically guesses a decomposition of the input word into finite segments such that the automaton $\mathcal{A}$ has a specific behavior on these segments. These behaviors are captured by transition profiles. Presentations of this complementation proof can be found in [13] and in [14]. In this section we repeat the essential parts of the method that are required to describe our improvements.

A transition profile of a word u essentially describes the behavior of all states of the automaton when reading u with respect to states that are reachable and states that are reachable via a final state.

Definition 1. *A pair $t = \langle \to_t, \mathrel{\text{-}\!\!\Rightarrow_t}\rangle$ with $\to_t \subseteq Q \times Q$ and $\mathrel{\text{-}\!\!\Rightarrow_t} \subseteq \to_t$ is called a* transition profile *over $\mathcal{A}$. The* transition profile $\tau(u)$ of the word $u \in \Sigma^*$ *over $\mathcal{A}$ is the pair $\langle \to, \mathrel{\text{-}\!\!\Rightarrow}\rangle$ with $p \to q$ iff $p \xrightarrow{u} q$ and $p \mathrel{\text{-}\!\!\Rightarrow} q$ iff $p \xrightarrow[F]{u} q$.*

One can visualize a transition profile as a directed graph. The nodes of the graph represent the states of the automaton, and there is an edge ($\to$) between two nodes, if the word u permits a transition from one state to the other. Additionally this edge is marked ($\mathrel{\text{-}\!\!\Rightarrow}$) if it permits a transition via a final state. A transition profile $\tau(u)$ contains information about the behavior of $\mathcal{A}$ on u, relevant to a Büchi automaton.

We define TP to be the set of all transition profiles over $\mathcal{A}$. There is a natural concatenation operation on TP. For $s_1, s_2 \in \mathrm{TP}$, the transition profile $t = s_1 \cdot s_2$ is defined by

- $p \to_t q$ iff $\exists r \in Q$ such that $p \to_{s_1} r \wedge r \to_{s_2} q$, and
- $p \mathrel{\text{-}\!\!\Rightarrow_t} q$ iff $\exists r \in Q$ such that $(p \mathrel{\text{-}\!\!\Rightarrow_{s_1}} r \wedge r \to_{s_2} q) \vee (p \to_{s_1} r \wedge r \mathrel{\text{-}\!\!\Rightarrow_{s_2}} q)$.

It is easy to see that the concatenation of transition profiles is associative, so $\langle \mathrm{TP}, \cdot\rangle$, the set of all transition profiles together with concatenation, forms a semigroup. With $\tau(\epsilon)$ as the neutral element, it even is a monoid — the *transition monoid* of $\mathcal{A}$. Furthermore, by induction on the length of the words one can show $\tau(u \cdot v) = \tau(u) \cdot \tau(v)$, so τ is a monoid homomorphism.

For an automaton with $|Q| = n$ states, there are n^2 distinct pairs of states, and for each pair, there are only three possibilities (either $p \not\to q$, or $p \to q$ but $p \not\Rightarrow q$, or $p \Rightarrow q$). So we have exactly 3^{n^2} distinct transition profiles in the transition monoid. However, not for all of them there is a word which induces this profile. By RTP we denote the set of all *reachable* transition profiles, i.e., those $t \in \mathrm{TP}$ for which there is a word $u \in \Sigma^*$ with $\tau(u) = t$.

For a transition profile t, define the language

$$\mathrm{L}(t) \coloneqq \tau^{-1}(t) = \{u \in \Sigma^* \mid \tau(u) = t\}.$$

The nonempty languages $\mathrm{L}(t)$ form a partition of the set Σ^* of all words into finitely many distinct classes, for there are only finitely many transition profiles.

It is not difficult to see that for each t the language $\mathrm{L}(t)$ is a regular language of finite words. In fact, the transition monoid of $\mathcal{A}$ can be represented by a deterministic finite automaton that we call the *transition monoid automaton*. Each state of this automaton corresponds to a reachable transition profile, the initial state is $\tau(\epsilon)$, and the a-successor of a state t is computed by $t \cdot \tau(a)$. Constructing this automaton can be done by starting with the initial state and spawning new states doing a breadth-first search, until the automaton is complete.

We define $\mathrm{TMA} = \langle \tilde{Q}, \Sigma, \tilde{q_0}, \tilde{\delta} \rangle$ with $\tilde{Q} = \mathrm{RTP}$, and $\tilde{q_0} = \tau(\epsilon)$, and $\tilde{\delta}(\tilde{q_0}, a) = \tau(a)$ and $\tilde{\delta}(t, a) = t \cdot \tau(a)$ for every $a \in \Sigma$ and $t \in \mathrm{RTP}$. Note that TMA does not have final states. The reason is that we instantiate TMA with different sets of final states, depending on the language we want to accept: By induction on the length of words, one can show that $\tilde{\delta}^*(\tilde{q_0}, w) = \tau(w)$ and from that it follows that the language recognized by TMA with final state set $F = \{t\}$ is exactly $\mathrm{L}(t)$.

The key observation in Büchi's proof is the following lemma. It states that each infinite word can be decomposed into finite segments such that the induced sequence of transition profiles is of the form st^ω. This is an almost direct consequence of Ramsey's theorem which states that for each coloring of (unordered) pairs over an infinite set with finitely many colors there is an infinite monochromatic subset. One can even restrict the transition profiles to those which satisfy the equations $t \cdot t = t$ (so t is idempotent) and $s \cdot t = s$.

Lemma 2 (Sequential Lemma). *For every $\alpha \in \Sigma^\omega$ there is a decomposition of α as $\alpha = uv_1v_2\cdots$ with reachable transition profiles s, t such that $\tau(u) = s$, $\tau(v_i) = t$ for every $i \geq 1$, $t \cdot t = t$, and $s \cdot t = s$.*

Define $\mathrm{L}(\langle s, t \rangle) \coloneqq \mathrm{L}(s) \cdot \mathrm{L}(t)^\omega$ and let

$$\text{s-t-Pairs} \coloneqq \{\langle s, t \rangle \in \mathrm{RTP}^2 \mid s \cdot t = s,\ t \cdot t = t\}.$$

The Sequential Lemma states that the languages $\mathrm{L}(\langle s, t \rangle)$ cover the set of all infinite words:

$$\Sigma^\omega = \bigcup \{\mathrm{L}(\langle s, t \rangle) \mid \langle s, t \rangle \in \text{s-t-Pairs}\}.$$

Given an arbitrary decomposition of an infinite word α into finite segments, the corresponding sequence of transition profiles contains all the relevant information

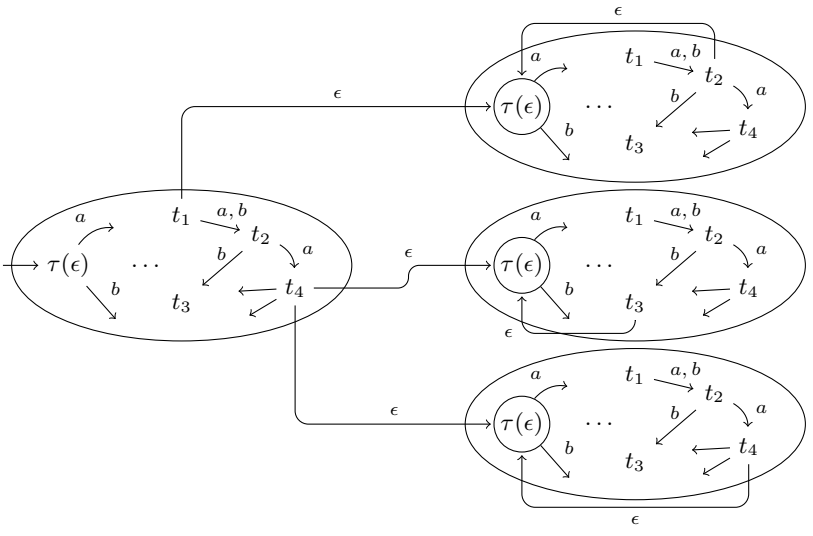

Fig. 1. A layout of a complement automaton with $\mathrm{Reject}_{\mathrm{s,t}} = \{\langle t_1, t_2\rangle, \langle t_4, t_3\rangle, \langle t_4, t_4\rangle\}$

on possible runs of $\mathcal{A}$ on α. This means that if two infinite words are both in $\mathrm{L}(\langle s, t\rangle)$, they both are accepted or both are rejected.

Lemma 3. *Let s and t be transition profiles. Then either* $\mathrm{L}(\langle s, t\rangle) \cap \mathrm{L}_\omega(\mathcal{A}) = \emptyset$, *or* $\mathrm{L}(\langle s, t\rangle) \subseteq \mathrm{L}_\omega(\mathcal{A})$.

This allows us to separate accepting and rejecting s-t-pairs into disjoint sets:

$$\mathrm{Accept}_{\mathrm{s,t}} := \{\langle s, t\rangle \in \text{s-t-Pairs} \mid \mathrm{L}(\langle s, t\rangle) \subseteq \mathrm{L}_\omega(\mathcal{A})\} \text{ and}$$
$$\mathrm{Reject}_{\mathrm{s,t}} := \{\langle s, t\rangle \in \text{s-t-Pairs} \mid \mathrm{L}(\langle s, t\rangle) \cap \mathrm{L}_\omega(\mathcal{A}) = \emptyset\}.$$

As a consequence we obtain the following characterization of the complement of $\mathrm{L}_\omega(\mathcal{A})$:

$$\Sigma^\omega \setminus L = \bigcup \{\mathrm{L}(\langle s, t\rangle) \mid \langle s, t\rangle \in \mathrm{Reject}_{\mathrm{s,t}}\}.$$

Using the transition monoid automaton as a basic building block one can construct a Büchi automaton for the complement. There are automata for each of the languages $\mathrm{L}(s)$ and $\mathrm{L}(t)$ for every rejecting s-t-pair. These can be combined to obtain an automaton for $\mathrm{L}(\langle s, t\rangle) = \mathrm{L}(s) \cdot \mathrm{L}(t)^\omega$ by connecting the final states of the first automaton (for $\mathrm{L}(s)$) to the initial state of the second one (for $\mathrm{L}(t)$), and by allowing the second automaton to loop back to its initial state from its final state with an ϵ-transition, and making the initial state the only final state. This construction assumes that the initial state of the automaton for $\mathrm{L}(t)$ does not have incoming transitions. Otherwise one can easily obtain this property by adding a new copy of the initial state.

In the construction it is even possible to reuse the same automaton for each of the different languages $\mathrm{L}(s)$, as it is done in [13]. This construction is depicted in Figure 1. The ϵ-transitions used in the illustration can be eliminated by standard techniques.

We refer to the first part of the automaton that guesses the initial segment of the decomposition as the *initial part* (on the left-hand side of Figure 1), and to the remaining part of the automaton that guesses the periodic part of the decomposition as the *looping part*.

The complement automaton consists of at most 3^{n^2} Büchi automata in the looping part, each with at most $3^{n^2}+1$ states (since we might need to add a new copy of the initial state), plus an additional initial automaton component of 3^{n^2} states. This results in a complement automaton with at most $2 \cdot 3^{n^2} + 9^{n^2}$ states.

4 Reducing State Space

In this section, we discuss several ideas to reduce the state space of the complement automaton.

4.1 Subset Construction

The first observation is that for a decomposition st^ω of an infinite word, the precise transition profile s is not required to decide whether the pair s,t is in $\mathrm{Reject}_{\mathrm{s,t}}$. The only information that is required on s is, which states are reachable from the initial state in s. Hence, we will replace the copy of TMA that takes care of the initial segment $\mathrm{L}(s)$ of the decomposition by a standard subset automaton, as detailed in the following.

For $P \subseteq Q$ define $\mathrm{L}(P) \coloneqq \{u \in \Sigma^* \mid \delta^*(\{q_0\}, u) = P\}$ to be the set of all words with which from q_0 one can reach exactly the states in P. For a transition profile t, define $t(P) \coloneqq \{q \in Q \mid \exists p \in P\colon p \to_t q\}$ to be the set of all states which are reachable from P via t. It is clear that $t(s(P)) = (s \cdot t)(P)$ for all $s,t \in \mathrm{TP}$ and $P \subseteq Q$.

The Sequential Lemma can easily be adapted to the new setting.

Lemma 4. *For every $\alpha \in \Sigma^\omega$ there is a decomposition of α as $\alpha = uv_1v_2\cdots$ with a set $P \subseteq Q$ and a reachable transition profile t such that $u \in \mathrm{L}(P)$, $\tau(v_i) = t$ for every $i \geq 1$, $t \cdot t = t$, and $t(P) = P$.*

Proof. We pick a decomposition according to Lemma 2 for transition profiles s and t. Setting $P = s(\{q_0\})$ one easily verifies the claimed properties. □

So instead of considering s-t-pairs, from now on we work with P-t-pairs

$$\text{P-t-Pairs} \coloneqq \{\langle P,t\rangle \in \mathrm{RSC} \times \mathrm{RTP} \mid t(P) = P,\ t \cdot t = t\},$$

and we define $\mathrm{L}(\langle P,t\rangle) \coloneqq \mathrm{L}(P) \cdot \mathrm{L}(t)^\omega$. Then Theorem 4 shows that the P-t-pairs again cover the set of all infinite words:

$$\Sigma^\omega = \bigcup \{\mathrm{L}(\langle P,t\rangle) \mid \langle P,t\rangle \in \text{P-t-Pairs}\}\,;$$

and we can divide the set of P-t-pairs into accepting and rejecting ones.

Lemma 5. *Let $P \subseteq Q$ and let t be a transition profile. Then either* $\mathrm{L}(\langle P, t\rangle) \cap \mathrm{L}_\omega(\mathcal{A}) = \emptyset$, *or* $\mathrm{L}(\langle P, t\rangle) \subseteq \mathrm{L}_\omega(\mathcal{A})$.

Proof. If $\mathrm{L}(P) \cdot \mathrm{L}(t)^\omega \cap \mathrm{L}_\omega(\mathcal{A})$ is not empty, then there is a word α which lies in both sets. Then α can be decomposed as $\alpha = uv_1v_2\cdots$ with $u \in \mathrm{L}(P)$ and all v_i being in $\mathrm{L}(t)$. Because the word α is in L, there must be an accepting run of $\mathcal{A}$ on α. Consider the form of this run to be $q_0 \xrightarrow{u} q_1 \xrightarrow{v_1} q_2 \xrightarrow{v_2} q_3 \cdots$. Note that $q_1 \in P$. This run visits infinitely often an accepting state. So for infinitely many i it holds $q_i \xrightarrow[F]{v_i} q_{i+1}$.

Now let β be any word in $\mathrm{L}(P) \cdot \mathrm{L}(t)^\omega$. Then β can also be decomposed as $\beta = u'v'_1v'_2\cdots$ with $u' \in \mathrm{L}(P)$ and all v'_i being in $\mathrm{L}(t)$. We have $u' \in \mathrm{L}(P)$ and $\tau(v_i) = \tau(v'_i)$ for all $i \geq 1$. Then there is a run ρ' of $\mathcal{A}$ on β of the form $q_0 \xrightarrow{u'} q_1 \xrightarrow{v'_1} q_2 \xrightarrow{v'_2} q_3 \cdots$. It holds $q_i \xrightarrow[F]{v'_i} q_{i+1}$ for the very same i as above. So ρ' is an accepting run and $\beta \in \mathrm{L}_\omega(\mathcal{A})$. □

We adapt the definition of the set of accepting and rejecting pairs:

$$\mathrm{Accept}_{\mathrm{P,t}} := \{\langle P, t\rangle \in \text{P-t-Pairs} \mid \mathrm{L}(\langle P, t\rangle) \subseteq \mathrm{L}_\omega(\mathcal{A})\}$$
$$\mathrm{Reject}_{\mathrm{P,t}} := \{\langle P, t\rangle \in \text{P-t-Pairs} \mid \mathrm{L}(\langle P, t\rangle) \cap \mathrm{L}_\omega(\mathcal{A}) = \emptyset\}$$

Summarizing the above observations, we can improve the construction from Section 3 by replacing the initial copy TMA (on the left-hand side in Figure 1) by a copy of a simple automaton keeping track of the set of reachable states, denoted by PA. A set P is connected to the copy of the TMA for the transition profile t if $\langle P, t\rangle \in \mathrm{Reject}_{\mathrm{P,t}}$.

Note that according to the above description all pairs of the form $\langle \emptyset, t\rangle$ and $\langle P, t_\emptyset\rangle$ with the empty transition profile $t_\emptyset = \langle \emptyset, \emptyset\rangle$ are in $\mathrm{Reject}_{\mathrm{P,t}}$. However, all these pairs correspond to words on which no run exists at all and thus have a prefix leading from $\{q_0\}$ to $\emptyset$ in the initial part PA of the complement automaton. In our implementation we hence make $\emptyset$ an accepting state and do not further consider the pairs of the above form in the construction of the automaton.

4.2 Merging Transition Profiles

In this section, we show how to merge different copies of TMA in the looping part of the complement automaton. Our hope is to obtain, for the looping part, a smaller number of automata and/or automata that are smaller in terms of their state space.

As we have seen in Section 3, the automata for the right-hand part are generated by setting the acceptance component of the transition monoid automaton TMA to a singleton set. Let us denote the resulting automata by *singleton automata*. A natural way to extend this practice is to allow arbitrary subsets of the state space to be set as the acceptance component. Then the resulting automaton (considered as a $*$-automaton) accepts the union of the languages formerly accepted by the singleton automata, and (in the best case) the singleton automata are not needed anymore and can be removed from the right-hand part.

Obviously, this merging cannot be done in an arbitrary fashion. Below we state criteria that are sufficient for merging several of the singleton automata in the looping part.

In the following, a set of P-t-pairs is called a *bucket*. For a bucket $B = \{\langle P_1, t_1\rangle, \ldots, \langle P_n, t_n\rangle\}$, we define the language $\mathrm{L}(B) \coloneqq \mathrm{L}^{\mathrm{P}}(B) \cdot \big(\mathrm{L}^{\mathrm{t}}(B)\big)^{\omega}$, where $\mathrm{L}^{\mathrm{P}}(B) \coloneqq \bigcup_{i=1}^{n} \mathrm{L}(P_i)$ and $\mathrm{L}^{\mathrm{t}}(B) \coloneqq \bigcup_{i=1}^{n} \mathrm{L}(t_i)$. Obviously, for a bucket B, it holds $\mathrm{L}(B) \supseteq \mathrm{L}(\langle P, t\rangle)$ for each $\langle P, t\rangle \in B$ (simply by definition of $\mathrm{L}(B)$).

Now we are interested in a condition that allows to merge several rejecting pairs into a bucket B such that $\mathrm{L}(B)$ has an empty intersection with $\mathrm{L}(\mathcal{A})$.

Definition 6. *For a bucket $B = \{\langle P_1, t_1\rangle, \ldots, \langle P_n, t_n\rangle\}$, we define its* join *as the pair $\langle P, t\rangle$ with $P = \bigcup_i P_i$ and $p \to_t q$ iff $\exists i\colon p \to_{t_i} q$, and $p \twoheadrightarrow_t q$ iff $\exists i\colon p \twoheadrightarrow_{t_i} q$. We say that such a pair $\langle P, t\rangle$* has a lasso *if there is a sequence of states $p_1, \ldots, p_k, \ldots, p_n$, $1 \leq k < n$ such that*

- *$p_1 \in P$,*
- *$p_i \to_t p_{i+1}$ for all $1 \leq i < n$,*
- *$p_k \twoheadrightarrow_t p_{k+1}$, and*
- *$p_n \to_t p_k$.*

We say that a bucket is mergeable *if its join does not have a lasso.*

For the join $\langle P, t\rangle$ of a bucket, note that P is not necessarily a reachable subset and that t is not necessarily a reachable transition profile.

The following lemma expresses, that by aggregating several mergeable P-t-pairs into one bucket, the language accepted by the resulting automaton is still inside the complement of $\mathrm{L}(\mathcal{A})$.

Lemma 7. *Let B be a mergeable bucket. Then $\mathrm{L}(B) \cap \mathrm{L}_{\omega}(\mathcal{A}) = \emptyset$.*

Proof. Let $B = \{\langle P_1, t_1\rangle, \ldots, \langle P_n, t_n\rangle\}$ and let $\langle P, t\rangle$ be the join of B and let α be a word in $\mathrm{L}(B)$. Then $\alpha \in \bigcup_{i=1}^{n} \mathrm{L}(P_i) \cdot (\bigcup_{i=1}^{n} \mathrm{L}(t_i))^{\omega}$ and there is an infinite sequence $i_0, i_1, i_2, \ldots$ with $1 \leq i_j \leq n$ such that α can be decomposed as $\alpha = uv_1v_2\cdots$ with $u \in \mathrm{L}(P_{i_0})$ and $v_j \in \mathrm{L}(t_{i_j})$ for each $j \geq 1$.

Assume that $\alpha \in \mathrm{L}(\mathcal{A})$ and consider an accepting run ρ of $\mathcal{A}$ on α. Consider the form of this run to be $q_0 \xrightarrow{u} q_1 \xrightarrow{v_1} q_2 \xrightarrow{v_2} q_3 \cdots$, and let R be the set of those states which occur infinitely often in this form. Since $\alpha \in \mathrm{L}(\mathcal{A})$, there is a state $q \in F$ such that q is visited infinitely often in ρ. Now we argue that $\langle P, t\rangle$ has a lasso. We have $q_1 \in P_{i_0}$ and therefore $q_1 \in P$. We have $q_i \to_t q_{i+1}$ for all $i \geq 1$. Since there are infinitely many accepting states visited in ρ, there must be a state $q_k \in R$ with $q_k \twoheadrightarrow_{t_{i_k}} q_{k+1}$ and therefore $q_k \twoheadrightarrow_t q_{k+1}$. Finally since q_k occurs infinitely often in the above form, there must be a state q_n with $n > k$ and $q_n \to_{t_{i_n}} q_k$, and therefore $q_n \to_t q_k$. So $\langle P, t\rangle$ has a lasso and this is a contradiction to the premise that B is mergeable. □

Proposition 8. *Let $\{B_1, \ldots, B_n\}$ be a set of buckets such that each bucket B_i is mergeable, and for each $\langle P, t\rangle \in \mathrm{Reject}_{\mathrm{P,t}}$ there is a bucket B_i with $\langle P, t\rangle \in B_i$. Then $\mathrm{L}(B_1) \cup \cdots \cup \mathrm{L}(B_n) = \Sigma^{\omega} \setminus \mathrm{L}(\mathcal{A})$.*

Proof. Let $\alpha \in \mathrm{L}(B_i)$ for some $1 \leq i \leq n$. Since B_i is mergeable, by Theorem 7 it follows that $\alpha \notin \mathrm{L}(\mathcal{A})$.

For the other direction, let $\alpha \in \Sigma^\omega \setminus \mathrm{L}(\mathcal{A})$. By Theorem 4 and the definition of $\mathrm{Reject}_{\mathrm{P,t}}$, we know that there is a $\langle P, t\rangle \in \mathrm{Reject}_{\mathrm{P,t}}$ such that $\alpha \in \mathrm{L}(\langle P, t\rangle)$. Then there is a bucket B_i with $\langle P, t\rangle \in B_i$ and since $\mathrm{L}(B_i) \supseteq \mathrm{L}(\langle P, t\rangle)$ it follows that $\alpha \in \mathrm{L}(B_i)$. □

Given a set $\{B_1, \ldots, B_n\}$ of buckets as in Proposition 8, and automata $\mathcal{A}_i = \langle Q_i, \Sigma, q_0^i, \delta_i, F_i\rangle$ for the languages $\mathrm{L}^{\mathrm{t}}(B_i)$, we can now further modify the construction from Section 3 by connecting for each pair $\langle P, t\rangle \in B_i$ the set P from the initial PA to the automaton $\mathcal{A}_i$ for $\mathrm{L}^{\mathrm{t}}(B_i)$, and applying the looping construction on the $\mathcal{A}_i$. This is done formally in the following definition, where we do not use ϵ-transitions for the connections and loops as in Figure 1 but directly eliminate them.

Definition 9. *Let $\{B_1, \ldots, B_n\}$ be a set of buckets as in Proposition 8 and for each $1 \leq i \leq n$ let $\mathcal{A}_i = \langle Q_i, \Sigma, q_0^i, \delta_i, F_i\rangle$ be an automaton such that $\mathrm{L}_*(\mathcal{A}_i) = \mathrm{L}^{\mathrm{t}}(B_i)$. Furthermore assume that the initial states of the $\mathcal{A}_i$ do not have incoming transitions. Then define the automaton $\mathcal{A}' \coloneqq \langle Q', \Sigma, q_0', \delta', F'\rangle$ with*

- $Q' = \mathrm{RSC} \cup \bigcup_i Q_i$,
- $q_0' = \{q_0\}$,
- $F' = \{\emptyset\} \cup \{q_0^1, \ldots, q_0^n\}$, *and*
- $\delta'(P, a) = \{\delta^*(P, a)\} \cup \{q \in Q' \mid \exists i\big(\exists \langle P, t\rangle \in B_i \text{ for some } t \wedge q \in \delta_i(q_0^i, a)\big)\}$ *for $P \subseteq Q$ and $a \in \Sigma$, and*
- $\delta'(q, a) = \begin{cases} \delta_i(q, a) \text{ if } \delta_i(q, a) \cap F_i = \emptyset \\ \{q_0^i\} \cup \delta_i(q, a) \text{ otherwise} \end{cases}$ *for $q \in Q_i$ and $a \in \Sigma$.*

It is easy to see that $\mathrm{L}_\omega(\mathcal{A}') = \mathrm{L}(B_1) \cup \cdots \cup \mathrm{L}(B_n)$ and by Proposition 8 we obtain that $\mathcal{A}'$ is an automaton for the complement of $\mathrm{L}(\mathcal{A})$.

In the above definition the automata for $\mathrm{L}^{\mathrm{t}}(B_i)$ are parameters and we did not specify how to construct them. Since for a bucket B, it holds $\mathrm{L}^{\mathrm{t}}(B) = \bigcup_{\langle P,t\rangle \in B}^{n} \mathrm{L}(t)$, one can use the transition monoid automaton TMA, setting all the states t to be final for which some pair $\langle P, t\rangle$ is in B. This is how we proceed in our implementation. In Section 5 we work with specific buckets and for those provide an alternative construction that allows us to give a better upper bound on the size of the resulting automata.

Minimizing t-Automata. The automata for the languages $\mathrm{L}^{\mathrm{t}}(B)$ that are obtained from TMA, as described above, are deterministic. As these automata are very large and have only few accepting states, they likely have many redundant states, too. So a natural approach here is to minimize these automata. After generating a bucket automaton $\mathcal{A}^{\mathrm{t}}(B)$, our algorithm immediately computes the minimal equivalent deterministic automaton, which in our experiments often results in much smaller automata for the looping part of the complement automaton.

4.3 Experimental Results

In order to compare our complementation method with existing ones, we tried to reflect the experimental setting of Tsai et al. [15]. They compared four different implementations, named `Ramsey`, `Safra-Piterman`, `Rank`, and `Slice`. Each of these implement one of the four approaches, Ramsey-based, determinization-based, rank-based, and slice-based, respectively. They randomly generated 11,000 input automata, each with 15 states, an alphabet of size 2, and combinations of 11 transition densities and 10 acceptance densities. Then they fed these complementation tasks to a cluster of computers. For each task, they allocated a 2.83 GHz CPU with 1 GB of memory and 10 minutes computation time. Since `Ramsey` performed very poorly in their experiments, they excluded this implementation from the overall comparison. The other three implementations were then improved by various heuristics. In the end, the improved versions of `Safra-Piterman`, `Rank`, and `Slice` had 4 tasks, 3,383 tasks, and 216 tasks that aborted unsuccessfully, respectively.

Number of outs in the experiment by Tsai et a. of the best versions of these construction including pre-minimisation of input atuoamt

URL to Alekto website

Alekto

We have implemented our ideas in a Java program. It can be obtained from [17], including its Java source code. Because there is no unique distribution of P-t-pairs to buckets, we had to agree on a concrete way to fill the buckets. We have chosen the following greedy algorithm. Maintain a list $(B_1, \ldots, B_n)$ of buckets, which can grow if needed. For each pair $\langle P, t\rangle \in \text{Reject}_{\text{P,t}}$, add $\langle P, t\rangle$ to the first bucket B_i such that $B_i \cup \{\langle P, t\rangle\}$ is mergeable. If no such bucket exists, then create a new empty bucket B_{n+1} and start over again. The algorithm that uses all of the above heuristics, the subset construction for the initial part, the merging of transition profiles for the looping part, and the minimization of the bucket automata, together with the greedy bucket filling algorithm, is called `improved-Ramsey`.

Our complementation jobs were conducted in sequence on a single machine with a 2.83 GHz CPU, a timeout of 10 minutes, and 4 GB of memory from which only 1 GB was used for the Java heap space. We used the same 11,000 complementation tasks as Tsai et al. Out of these, 10,839 finished successfully, 152 ran out of memory, and 9 violated the time limit. In terms of successfully finished tasks, this puts `improved-Ramsey` second place, between `Safra-Piterman` and `Slice`.

161 outs

This ranking is based solely on the number of outs (timeouts and memory outs). This is however not reliable, because the two experiments have been executed on different machines.

The size of the complement automata computed by `improved-Ramsey` range from 0 to 337,464 states with an average size of 361.09 states (328.97 after removing dead states). We were not able to adequately compare these sizes with the results of [15] for the following reason. Tsai et al. provide average sizes only for 7,593 of the initial 11,000 automata, namely for those tasks that finished successfully by all of their implementations. Our numbers, on the other hand, base upon all 10,839 finished tasks of our implementation.

Cannot compare results, because they don't know which are the effective samples

Number of outs:

Safra-Piterman:	4	Tsai et al (2011) Table 3
imprvd-Ramsey:	161	This paper
Slice:	216	Tsai et al (2011) Table 3
Rank:	3383	Tsai et al (2011) Table 3

5 Preorder-Based Optimization

The aim of this section is to improve the Ramsey-based construction in such a way that the resulting complement automaton is of size $2^{\mathcal{O}(n \log n)}$. There are

two things we have to take care of: The complement automaton is composed of the initial subset automaton and the part for the ω-iteration, in which for each transition profile there is one copy of the transition monoid automaton. To reduce the number of states we have to reduce (1) the number of copies, and (2) the size of these automata.

In Section 4.2 we have seen that we can merge transition profiles in the looping part of the complement automaton, as long as the combination of the merged transition profiles does not introduce an accepting cycle. To obtain an automaton for the complement it is sufficient to cover $\mathrm{Reject}_{\mathrm{P,t}}$ by mergeable buckets of transition profiles.

In this section we show how to systematically do this merging such that the number of buckets, and the respective size of the automata for the buckets are of order $2^{\mathcal{O}(n \log n)}$. The key idea is that we can merge all transition profiles that can be embedded into the same preorder in such a way that $\not\Rightarrow$ edges are strictly increasing in the order and $\rightarrow$ edges are not decreasing, thus avoiding accepting cycles. This is made precise in the following definitions.

A *total preorder* (or *weak order*) on $P \subseteq Q$ is a binary relation $\lesssim$ on P that is total (for all $p, q \in P$ it holds $p \lesssim q$ or $q \lesssim p$), and transitive (for all $p, q, r \in P$ with $p \lesssim q$ and $q \lesssim r$ it holds $p \lesssim r$). With Pre denote the set of all pairs $\langle P, \lesssim \rangle$ such that $P \subseteq Q$ and $\lesssim$ is a total preorder on P.

For any total preorder $\lesssim$, one can define its corresponding equivalence relation by $p \simeq q \Leftrightarrow p \lesssim q \wedge q \lesssim p$. The resulting equivalence classes are linearly ordered by $[p] \leq [q] \Leftrightarrow p \lesssim q$. As usual one defines $[p] < [q]$ if $[p] \leq [q] \wedge [q] \not\leq [p]$.

Note that the number of preorders on a set with n elements is bounded by n^n because each preorder can be characterized by a mapping that assigns to each element of the set a number from 1 to n corresponding to its position in the order (equivalent elements are mapped to the same number). So the number of pairs $\langle P, \lesssim \rangle$ is bounded by $2^n n^n$.

Definition 10. *Let* $\langle P, \lesssim \rangle \in \mathrm{Pre}$. *We say that a transition profile* $t = \langle \rightarrow, \not\Rightarrow \rangle$ *is* compatible with $\langle P, \lesssim \rangle$ *if*
- $t(P) \subseteq P$, *and*
- *for all* $p, q \in P$ *with* $p \rightarrow q$ *it holds* $[p] \leq [q]$, *and*
- *for all* $p, q \in P$ *with* $p \not\Rightarrow q$ *it holds* $[p] < [q]$.

For any preordered set $\langle P, \lesssim \rangle$, let us define the bucket

$$B_{\langle P, \lesssim \rangle} := \{\langle P, t \rangle \in \mathrm{RSC} \times \mathrm{RTP} \mid t \text{ is compatible with } \langle P, \lesssim \rangle\}.$$

It is not difficult to see that these buckets are mergeable in the sense of Section 4.2 and that each pair in $\mathrm{Reject}_{\mathrm{P,t}}$ is compatible with a suitable preorder.

Lemma 11. $B_{\langle P, \lesssim \rangle}$ *is mergeable for each* $\langle P, \lesssim \rangle \in \mathrm{Pre}$ *and for each* $\langle P, t \rangle \in \mathrm{Reject}_{\mathrm{P,t}}$, *there is a total preorder* $\lesssim$ *on* P *with* $\langle P, t \rangle \in B_{\langle P, \lesssim \rangle}$.

Proof. We start with the first claim. Assume the join of $B_{\langle P, \lesssim \rangle}$ has a lasso. Then there is a sequence of states $p_1, \ldots, p_k, \ldots, p_n \in Q$, $1 \leq k < n$ with

$p_1 \in P$, $p_i \to p_{i+1}$ for all $i < n$, $p_k \twoheadrightarrow p_{k+1}$, and $p_n \to p_k$. Since $t(P) \subseteq P$ for all t compatible with $\langle P, \lesssim \rangle$, it holds $p_i \in P$ for all $1 \leq i \leq n$ by induction on i. Then we have $[p_{k+1}] \leq [p_{k+2}] \leq \cdots \leq [p_n] \leq [p_k]$, and $[p_k] < [p_{k+1}]$, which is a contradiction.

To prove the second claim, note that by definition of $\mathrm{Reject}_{\mathrm{P,t}}$ it holds $t(P) \subseteq P$. Consider the directed graph G with vertex set P and edge relation $\to_t$. The SCCs of G are preordered by $C_1 \; R \; C_2$ iff there is a path from a state $p \in C_1$ to a state $q \in C_2$ in G. We make this preorder total by ordering incomparable SCCs of G in an arbitrary way. We obtain a total preorder R' on the set of SCCs. This induces a total preorder $\lesssim$ on P by $p \lesssim q$ iff $C_p \; R' \; C_q$ for $p \in C_p$ and $q \in C_q$. Clearly $p \to_t q$ implies $[p] \leq [q]$. Let $C \subseteq P$ be an SCC of G. Then for states $p_1, p_2 \in C$, it cannot hold $p_1 \twoheadrightarrow_t p_2$, as otherwise we can construct an accepting run $q_0 \xrightarrow{u} p_1 \xrightarrow[F]{v_1} p_2 \xrightarrow{v_2} p_3 \to \cdots p_n \xrightarrow{v_n} p_1 \cdots$ of $\mathcal{A}$ on a word $u(v_1 \cdots v_n)^\omega$ with $u \in \mathrm{L}(P)$ and $v_i \in \mathrm{L}(t)$. So $p_1 \twoheadrightarrow_t p_2$ implies $[p_1] < [p_2]$. Altogether, t is compatible with $\langle P, \lesssim \rangle$, and thus $\langle P, t \rangle \in B_{\langle P, \lesssim \rangle}$. □

According to the above lemma we have already found a covering of $\mathrm{Reject}_{\mathrm{P,t}}$ by a number of buckets that is bounded by $2^n n^n$. It remains to show that for a given preorder $\langle P, \lesssim \rangle$ the language $\mathrm{L}^{\mathrm{t}}(B_{\langle P, \lesssim \rangle})$, that is, those words whose transition profile is compatible with $\langle P, \lesssim \rangle$, can be recognized by a "small" automaton.

We realize this as follows. When reading a word v, the automaton keeps track for each $q \in Q$ which are the maximal equivalence classes $[p']$ and $[p'']$ of $\langle P, \lesssim \rangle$ such that in $\tau(v)$ there is an $\to$ edge from an element of $[p']$ to q and there is an $\twoheadrightarrow$ edge from and element of $[p'']$ to q. This information can easily be updated when appending a new letter to v, and Lemma 13 shows that it suffices to deduce whether $\tau(v)$ is compatible with $\langle P, \lesssim \rangle$.

To formalize this idea let $\mathcal{M}_{\langle P, \lesssim \rangle}$ be the set of all functions $f \colon Q \to P_\perp$, where $P_\perp = \{\perp\} \cup (P/\simeq)$ and the linear order $\leq$ on $P/\simeq$ is extended to $P_\perp$ by setting $\perp < [p]$ for all $[p] \in P/\simeq$.

The states of the automaton are pairs of such mappings. The initial state is the pair $\langle \phi_\epsilon, \psi_\epsilon \rangle \in \mathcal{M}_{\langle P, \lesssim \rangle} \times \mathcal{M}_{\langle P, \lesssim \rangle}$ with

$$\phi_\epsilon(q) \coloneqq \begin{cases} [q] & \text{if } q \in P, \\ \perp & \text{else;} \end{cases} \quad \text{and} \quad \psi_\epsilon(q) \coloneqq \begin{cases} [q] & \text{if } q \in P \cap F, \\ \perp & \text{else.} \end{cases}$$

For two functions $\phi, \psi \in \mathcal{M}_{\langle P, \lesssim \rangle}$ and for a letter $a \in \Sigma$, we define the update of ϕ and ψ by letter a to be $\langle \phi, \psi \rangle \cdot a \coloneqq \langle \phi', \psi' \rangle$ with

$$\phi'(q) = \max\{\phi(r) \mid r \in Q, r \xrightarrow{a} q\}, \text{ and}$$
$$\psi'(q) = \begin{cases} \max\{\phi(r) \mid r \in Q, r \xrightarrow{a} q\} & \text{if } q \in F, \\ \max\{\psi(r) \mid r \in Q, r \xrightarrow{a} q\} & \text{else.} \end{cases}$$

We use the convention $\max \emptyset = \perp$. Note that the definition depends on the context $\langle P, \lesssim \rangle$ in which it is used.

We write $\langle \phi_a, \psi_a \rangle$ for $\langle \phi_\epsilon, \psi_\epsilon \rangle \cdot a$, and inductively we write $\langle \phi_{va}, \psi_{va} \rangle$ for $\langle \phi_v, \psi_v \rangle \cdot a$. The function ϕ_v maps every state q to the maximal class in

$P/\simeq$ from which one can reach q by reading v; it maps to $\max \emptyset = \bot$ if no such class exists. Accordingly, the function ψ_v maps every state q to the maximal class from which one can reach q passing an accepting state by reading v; it maps to $\bot$ if no such class exists. We formalize this in the following lemma.

Lemma 12. *Let* $\langle P, \lesssim \rangle \in \text{Pre}$ *and* $v \in \Sigma^*$. *Then* $\phi_v(q) = \max\{[p] \mid p \in P, p \xrightarrow{v} q\}$ *and* $\psi_v(q) = \max\{[p] \mid p \in P, p \xrightarrow[F]{v} q\}$ *for each* $q \in Q$.

To define the set of final states of the automaton we have to identify those pairs of functions that indicate whether the transition profile of the current word is compatible with $\langle P, \lesssim \rangle$.

Let $\phi, \psi \in \mathcal{M}_{\langle P, \lesssim \rangle}$. We say that the pair $\langle \phi, \psi \rangle$ is *consistent with* $\langle P, \lesssim \rangle$ if
- for all $q \in Q \setminus P$ it holds $\phi(q) = \bot$, and
- for all $q \in P$ it holds $\phi(q) \leq [q]$, and
- for all $q \in P$ it holds $\psi(q) < [q]$.

Lemma 13. *Let* $\langle P, \lesssim \rangle \in \text{Pre}$ *and* $v \in \Sigma^*$. *Then the transition profile* $\tau(v)$ *is compatible with* $\langle P, \lesssim \rangle$ *iff* $\langle \phi_v, \psi_v \rangle$ *is consistent with* $\langle P, \lesssim \rangle$.

Proof. Let $t = \tau(v)$. With Theorem 12 we obtain

$$\begin{aligned} t(P) \subseteq P &\iff \forall q \in Q \setminus P \; \neg \exists p \in P \colon p \xrightarrow{v} q \\ &\iff \forall q \in Q \setminus P \colon \phi_v(q) = \bot, \text{ as well as} \end{aligned}$$

$$\begin{aligned} \forall p, q \in P \colon (p \xrightarrow{v} q \Rightarrow [p] \leq [q]) &\iff \forall q \in P \colon \max\{[p] \mid p \in P, p \xrightarrow{v} q\} \leq [q] \\ &\iff \forall q \in P \colon \phi_v(q) \leq [q], \text{ and} \end{aligned}$$

$$\begin{aligned} \forall p, q \in P \colon (p \xrightarrow[F]{v} q \Rightarrow [p] < [q]) &\iff \forall q \in P \colon \max\{[p] \mid p \in P, p \xrightarrow[F]{v} q\} < [q] \\ &\iff \forall q \in P \colon \psi_v(q) < [q]. \end{aligned}$$

□

Combining the above observations we can define the deterministic automaton $\mathcal{A}_{\langle P, \lesssim \rangle} \coloneqq \langle Q'', \Sigma, q_0'', \delta'', F'' \rangle$ as follows:
- $Q'' = \{\langle P, \lesssim, \phi, \psi \rangle \mid \phi, \psi \in \mathcal{M}(P)\}$
- $q_0'' = \langle P, \lesssim, \phi_\epsilon, \psi_\epsilon \rangle$
- $\delta''(\langle P, \lesssim, \phi, \psi \rangle, a) = \{\langle P, \lesssim, \phi', \psi' \rangle\}$ where $\langle \phi', \psi' \rangle = \langle \phi, \psi \rangle \cdot a$
- $F'' = \{\langle P, \lesssim, \phi, \psi \rangle \mid \langle \phi, \psi \rangle \text{ is consistent with } \langle P, \lesssim \rangle\}$

As a consequence of Lemma 13 we obtain the following result.

Lemma 14. *For every* $\langle P, \lesssim \rangle$, *the automaton* $\mathcal{A}_{\langle P, \lesssim \rangle}$ *accepts those words for which the transition profile is compatible with* $\langle P, \lesssim \rangle$, *that is,* $\mathrm{L}_*(\mathcal{A}_{\langle P, \lesssim \rangle}) = \mathrm{L}^{\mathrm{t}}(B_{\langle P, \lesssim \rangle})$.

Now we can plug the automata $\mathcal{A}_{\langle P, \lesssim \rangle}$ into the construction from Definition 9. The results from this section in combination with Proposition 8 imply that the resulting automaton indeed recognizes the complement language of the original automaton.

Theorem 15. *Applied to a Büchi automaton with n states, the Ramsey-based method in combination with preorder merging of transition profiles yields a complement automaton with at most* $2^n + 2^n n^n((n+1)^{2n}+1)$ *states.*

Proof. As mentioned above, the number of pairs $\langle P, \lesssim \rangle$ is bounded by $2^n n^n$.

The number of states in $\mathcal{A}_{\langle P,\lesssim\rangle}$ is not greater than $(n+1)^{2n}$. For applying the ω-iteration to these automata a new initial state has to be introduced.

The initial part of the complement automaton consists of a subset automaton. Altogether this gives the claimed bound. □

6 Conclusion

We proposed several heuristics to improve the Ramsey-based Büchi complementation method. We implemented these heuristics and showed that, in practice, our implementation can compete with implementations of other complementation methods. We introduced a novel optimization of the Ramsey-based method using preorders, with a $2^{\mathcal{O}(n \log n)}$ upper bound. From this and from the fact that the improved Ramsey-based approach yields good experimental results, we conclude that the Ramsey-based approach still has its place, in contrast to the results in [15].

Although the preorder-based optimization results in a complement automaton for which we can prove a better upper bound, our implementation uses a different strategy (the greedy strategy described in Section 4.3) to construct the buckets. The reason is that the greedy strategy easily allows to restrict to reachable transition profiles, while the preorder based approach does not allow this (at least not directly). Therefore, the improved Ramsey-based method still has to compute the entire reachable part of the transition monoid, and we are not aware of any upper bound on its size better than 3^{n^2}. However, the experiments suggest that in many cases the reachable part of the monoid is much smaller than the worst case.

We see our paper also in the broader context of comparing and unifying different complementation methods. Only recently, Fogarty et al. [3] compared the rank-based with the slice-based method. They combined both approaches and obtained, utilizing a total preorder on the nodes in a run DAG, a unified complementation method. We would not be surprised if the improved Ramsey-based method could also be unified with one of the other methods. This is one way, of extending our work. A second direction of future work could investigate whether the heuristics of our work can be used for universality and inclusion checking of Büchi automata.

Acknowledgments. We thank the authors of [15] for providing the 11,000 example automata, and the anonymous referees for their helpful comments.

References

1. Abdulla, P.A., Chen, Y.-F., Clemente, L., Holík, L., Hong, C.-D., Mayr, R., Vojnar, T.: Advanced Ramsey-based Büchi Automata Inclusion Testing. In: Katoen, J.-P., König, B. (eds.) CONCUR 2011 – Concurrency Theory. LNCS, vol. 6901, pp. 187–202. Springer, Heidelberg (2011)
2. Büchi, J.R.: On a decision method in restricted second order arithmetic. In: Logic, Methodology and Philosophy of Science, pp. 1–11. Stanford Univerity Press (1962)
3. Fogarty, S., Kupferman, O., Vardi, M.Y., Wilke, T.: Unifying Büchi complementation constructions. In: CSL (2011)
4. Fogarty, S., Vardi, M.Y.: Efficient Büchi Universality Checking. In: Esparza, J., Majumdar, R. (eds.) TACAS 2010. LNCS, vol. 6015, pp. 205–220. Springer, Heidelberg (2010)
5. Friedgut, E., Kupferman, O., Vardi, M.Y.: Büchi complementation made tighter. International Journal of Foundations of Computer Science 17(4), 851–868 (2006)
6. Kähler, D., Wilke, T.: Complementation, Disambiguation, and Determinization of Büchi Automata Unified. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 724–735. Springer, Heidelberg (2008)
7. Klarlund, N.: Progress measures for complementation of ω-automata with applications to temporal logic. In: FOCS, pp. 358–367. IEEE Computer Society (1991)
8. Michel, M.: Complementation is more difficult with automata on infinite words. Technical report, CNET, Paris (1988)
9. Piterman, N.: From nondeterministic Büchi and Streett automata to deterministic parity automata. Logical Methods in Computer Science 3(3) (2007)
10. Ramsey, F.P.: On a problem of formal logic. Proceedings of the London Mathematical Society 2(1), 264 (1930)
11. Safra, S.: On the complexity of ω-automata. In: FOCS, pp. 319–327. IEEE (1988)
12. Schewe, S.: Büchi complementation made tight. In: STACS. LIPIcs, vol. 3, pp. 661–672. Schloss Dagstuhl (2009)
13. Sistla, A.P., Vardi, M.Y., Wolper, P.: The complementation problem for Büchi automata with applications to temporal logic. Theoretical Computer Science 49, 217–237 (1987)
14. Thomas, W.: Automata on infinite objects. In: Handbook of Theoretical Computer Science. Formal Models and Semantics, vol. B, pp. 133–192. Elsevier Science Publishers, Amsterdam (1990)
15. Tsai, M.-H., Fogarty, S., Vardi, M.Y., Tsay, Y.-K.: State of Büchi Complementation. In: Domaratzki, M., Salomaa, K. (eds.) CIAA 2010. LNCS, vol. 6482, pp. 261–271. Springer, Heidelberg (2011)
16. Yan, Q.: Lower bounds for complementation of ω-automata via the full automata technique. Logical Methods in Computer Science 4(1) (2008)
17. `http://www.automata.rwth-aachen.de/research/Alekto/`