Efficient Büchi Universality Checking [⋆]

Seth Fogarty and Moshe Y. Vardi

Department of Computer Science, Rice University, Houston, TX
{sfogarty,vardi}@cs.rice.edu

Abstract. The complementation of Büchi automata, required for checking automata universality, remains one of the outstanding automata-theoretic challenges in formal verification. Early constructions using a Ramsey-based argument have been supplanted by rank-based constructions with exponentially better bounds. The best rank-based algorithm for Büchi universality, by Doyen and Raskin, employs a subsumption technique to minimize the size of the working set. Separately, in the context of program termination, Lee et al. have specialized the Ramsey-based approach to size-change termination (SCT) problems. In this context, Ramsey-based algorithms have proven to be surprisingly competitive. The strongest tool, from Ben-Amram and Lee, also uses a subsumption technique, although only for the special case of SCT problems.

We extend the subsumption technique of Ben-Amram and Lee to the general case of Büchi universality problems, and experimentally demonstrate the necessity of subsumption for the scalability of the Ramsey-based approach. We then empirically compare the Ramsey-based tool to the rank-based tool of Doyen and Raskin over a terrain of random Büchi universality problems. We discover that the two algorithms exhibit distinct behavior over this problem terrain. As expected, on many of the most difficult areas the rank-based approach provides the superior tool. Surprisingly, there also exist several areas, including the area most difficult for rank-based tools, on which the Ramsey-based solver scales better than the rank-based solver. This result demonstrates the pitfalls of using worst-case complexity to evaluate algorithms. We suggest that a portfolio approach may be the best approach to checking the universality of Büchi automata.

1 Introduction

The complementation problem for nondeterministic automata over infinite words is a vital step in the automata-theoretic approach to formal verification. The automata-theoretic approach reduces questions about program adherence to a specification to questions about language containment [19]. Representing liveness, fairness, or termination properties requires finite automata that operate on infinite words. One automaton, $\mathcal{A}$, encodes the behavior of the program, while another automaton, $\mathcal{B}$, encodes the formal specification. To ensure adherence, verify that the intersection of $\mathcal{A}$ with the complement of $\mathcal{B}$ is empty. The most difficult step is constructing the complementary automaton $\overline{\mathcal{B}}$. When addressing this problem, the formal verifications community has focused on universality testing [6, 18, 20]. This is the simplest case of containment

[⋆] Work supported in part by NSF grants CCF-0613889, ANI-0216467, OISE-0913807, and CCF-0728882, by BSF grant 9800096, and by gift from Intel.

checking: checking if the universal language is contained in the language of the automaton. Finite automata on infinite words are classified by their acceptance condition and transition structure. We consider here nondeterministic Büchi automata, in which a run is accepting when it visits at least one accepting state infinitely often [2].

The first complementation constructions for nondeterministic Büchi automata employed a Ramsey-based combinatorial argument to partition infinite words into a finite set of regular languages. Proposed by Büchi in 1962 [2], this construction was shown in 1987 by Sistla, Vardi, and Wolper to be implementable with a blow-up of $2^{O(n^2)}$ [17]. This brought the complementation problem into singly-exponential blow-up, but left a gap with the $2^{\Omega(n \log n)}$ lower bound proved by Michel [13].

The gap was tightened one year later in 1988, when Safra described a $2^{O(n \log n)}$ construction [15]. Because of this, the Ramsey-based approach has never been implemented. Work since then has focused on improving the practicality of $2^{O(n \log n)}$ constructions, either by providing simpler constructions, further tightening the bound [16], or improving the derived algorithms. In 2001, Kupferman and Vardi employed a rank-based analysis of Büchi automata to simplify complementation [12]. Recently, Doyen and Raskin have demonstrated the necessity of using a subsumption technique in the rank-based approach, providing a direct universality checker that scales to automata several orders of magnitude larger than previous tools [6].

Separately, in the context of of program termination analysis, Lee, Jones, and Ben-Amram presented the size-change termination (SCT) principle in 2001 [5]. Lee et al. describe a method of size-change termination analysis and reduce this problem to the containment of two Büchi automata. Stating the lack of efficient Büchi containment solvers, they also propose a direct Ramsey-based combinatorial solution. The Lee, Jones, and Ben-Amram (LJB) algorithm was provided as a practical alternative to reducing the SCT problems to Büchi containment, but bears a striking resemblance to the 1987 Ramsey-based complementation construction. In a previous paper, we showed that the LJB algorithm for deciding SCT is a specialized realization of the Ramsey-based construction [9]. When examined empirically, Ramsey-based tools proved to be surprisingly competitive to their rank-based counterparts. The best Ramsey-based tool employs a subsumption technique for the specific case of SCT problems [1].

This paper extends the subsumption technique of Ben-Amram and Lee to the general case of Büchi universality. By doing so we provide a direct algorithm, derived from the Ramsey-based complementation construction, for checking the universality of Büchi automata. We note that subsumption is a heuristic technique and, even with this improvement, there is still an exponential gap between the $2^{O(n^2)}$ Ramsey-based approach and the $2^{O(n \log n)}$ rank-based approach. Motivated by the Ramsey-based approach's strong performance on the domain of SCT problems, we investigate the empirical performance of these two algorithms. Due to a paucity of real-world universality problems, we compare the algorithms over a terrain of random universality problems [6, 18] characterized by transition density, acceptance density, and size.

Our empirical results first demonstrate that, as with rank-based algorithms, subsumption is necessary for scalability in Ramsey-based tools. Further, we observe that the two algorithms exhibit significantly different behavior. The terrain points that pose difficulty for each algorithm, while overlapping, are distinct. In terms of scalability, we show that in many areas the rank-based universality tool performs exponentially better than the Ramsey-based universality tool. However, there also exist several areas where the Ramsey-based tool is more scalable than the rank-based tool, despite the massive difference between $2^{O(n \log n)}$ and $2^{O(n^2)}$. Finally, we discover that the Ramsey-based tool is better at demonstrating non-universality by finding a counterexample, while the rank-based tool is superior when proving universality. This final difference can be attributed to the manner in which each approaches explores the state space of the complemented automaton, but does not explain the other behaviors of the two approaches. We are thus forced to conclude that worst-case complexity is a poor predictor of an algorithms performance, and no substitute for empirical analysis. We suggest that a portfolio approach [10, 14] may be employed when checking the universality of Büchi automata. Failing that, run both algorithms in parallel, and see which terminates first.

2 Preliminaries

In this section we review the relevant details of Büchi automata, introducing along the way the notation used throughout this paper. An *nondeterministic Büchi automaton on infinite words* is a tuple $\mathcal{B} = \langle \Sigma, Q, Q^{in}, \rho, F \rangle$, where Σ is a finite nonempty alphabet, Q a finite nonempty set of states, $Q^{in} \subseteq Q$ a set of initial states, $F \subseteq Q$ a set of accepting states, and $\rho : Q \times \Sigma \to 2^Q$ a nondeterministic transition function. We lift the ρ function to sets of states and words of arbitrary length in the usual fashion.

A *run* of a Büchi automaton $\mathcal{B}$ on a word $w \in \Sigma^\omega$ is a infinite sequence of states $q_0 q_1 ... \in Q^\omega$ such that $q_0 \in Q^{in}$ and, for every $i \geq 0$, we have $q_{i+1} \in \rho(q_i, w_i)$. A run is *accepting* iff $q_i \in F$ for infinitely many $i \in I\!N$. A word $w \in \Sigma^\omega$ is accepted by $\mathcal{B}$ if there is an accepting run of $\mathcal{B}$ on w. The words accepted by $\mathcal{B}$ form the *language* of $\mathcal{B}$, denoted by $L(\mathcal{B})$. A *path* in $\mathcal{B}$ from q to r is a finite subsequence of a run beginning in q and ending in r. A path is *accepting* if some state in the path is in F.

A Büchi automaton $\mathcal{A}$ is contained in a Büchi automaton $\mathcal{B}$ iff $L(\mathcal{A}) \subseteq L(\mathcal{B})$, which can be checked by verifying that the intersection of $\mathcal{A}$ with the complement $\overline{B}$ of $\mathcal{B}$ is empty: $L(\mathcal{A}) \cap L(\overline{\mathcal{B}}) = \emptyset$. We know that the language of an automaton is non-empty iff there are states $q \in Q^{in},\ r \in F$ such that there is a path from q to r and a path from r to itself. The initial path is called the prefix, and the combination of the prefix and cycle is called a *lasso* [19]. Further, the intersection of two automata can be constructed, having a number of states proportional to the product of the number states of the original automata [3]. Thus the most computationally demanding step is constructing the complement of $\mathcal{B}$. In the formal verification field, existing empirical work has focused on the simplest form of containment testing, *universality* testing, where $\mathcal{A}$ is the universal automaton [6, 18].

For algorithms that compute sets of states, a subsumption technique can sometimes be employed to limit the size of working sets. This technique ignores certain states when their behavior is subsumed by other states. A *subsumption* relation is a partial order over the state space of an automaton, such that if a state q subsumes a state r, then r can be removed from any set containing q.

2.1 Ramsey-Based Universality

When Büchi introduced these automata in 1962, he described a complementation construction involving a Ramsey-based combinatorial argument. We describe a universality testing algorithm based on an improved implementation presented in 1987. To construct the complement of $\mathcal{B}$, where $Q = \{q_0, ..., q_{n-1}\}$, we construct a set $\widetilde{Q}_{\mathcal{B}}$ whose elements capture the essential behavior of $\mathcal{B}$. Each element corresponds to an answer to the following question. Given a finite nonempty word w, for every two states $q, r \in Q$: is there a path in $\mathcal{B}$ from q to r over w, and is some such path accepting?

Define $Q' = Q \times \{0,1\} \times Q$, and $\widetilde{Q}_{\mathcal{B}}$ to be the subset of $2^{Q'}$ whose elements do not contain both $\langle q, 0, r\rangle$ and $\langle q, 1, r\rangle$ for any q and r. Each element of $\widetilde{Q}_{\mathcal{B}}$ is a $\{0,1\}$-arc-labeled graph on Q. An arc represents a path in $\mathcal{B}$, and the label is 1 if the path is accepting. Note that there are 3^{n^2} such graphs. With each graph $\widetilde{g} \in \widetilde{Q}_{\mathcal{B}}$ we associate a language $L(\widetilde{g})$, the set of words for which the answer to the posed question is the graph encoded by $\widetilde{g}$.

Definition 1. [2, 17] *Let* $\widetilde{g} \in \widetilde{Q}_{\mathcal{B}}$ *and* $w \in \Sigma^+$. *Say* $w \in L(\widetilde{g})$ *iff for all* $q, r \in Q$*:*
(1) $\langle q, a, r\rangle \in \widetilde{g},\ a \in \{0,1\}$, *iff there is a path in* $\mathcal{B}$ *from* q *to* r *over* w
(2) $\langle q, 1, r\rangle \in \widetilde{g}$ *iff there is an accepting path in* $\mathcal{B}$ *from* q *to* r *over* w

The languages $L(\widetilde{g})$ for the graphs $\widetilde{g} \in \widetilde{Q}_{\mathcal{B}}$, form a partition of Σ^+. With this partition of Σ^+ we can devise a finite family of ω-languages that cover Σ^ω. For every $\widetilde{g},\ \widetilde{h} \in \widetilde{Q}_{\mathcal{B}}$, let Y_{gh} be the ω-language $L(\widetilde{g}) \cdot L(\widetilde{h})^\omega$. We say that a language Y_{gh} is *proper* if Y_{gh} is non-empty, $L(\widetilde{g}) \cdot L(\widetilde{h}) \subseteq L(\widetilde{g})$, and $L(\widetilde{h}) \cdot L(\widetilde{h}) \subseteq L(\widetilde{h})$. There are a finite, if exponential, number of such languages. A Ramsey-based argument shows that every infinite string belongs to a language of this form, and that $\overline{L(\mathcal{B})}$ can be expressed as the union of languages of this form.

Lemma 1. [2, 17]
(1) $\Sigma^\omega = \bigcup\{Y_{gh} \mid Y_{gh}$ *is proper*$\}$
(2) For $\widetilde{g}, \widetilde{h} \in \widetilde{Q}_{\mathcal{B}}$*, either* $Y_{gh} \cap L(\mathcal{B}) = \emptyset$ *or* $Y_{gh} \subseteq L(\mathcal{B})$
(3) $\overline{L(\mathcal{B})} = \bigcup\{Y_{gh} \mid Y_{gh}$ *is proper and* $Y_{gh} \cap L(\mathcal{B}) = \emptyset\}$

To obtain the complementary Büchi automaton $\overline{\mathcal{B}}$, Sistla et al. construct, for each $\widetilde{g} \in \widetilde{Q}_{\mathcal{B}}$, a deterministic automata on finite words, $\mathcal{B}_g$, that accepts exactly $L(\widetilde{g})$ [17]. Using the automata $\mathcal{B}_g$, one could construct the complementary automaton $\overline{\mathcal{B}}$ and use a lasso-finding algorithm to prove the emptiness of $\overline{\mathcal{B}}$, and thus the universality of $\mathcal{B}$. However, we can avoid an explicit lasso search by employing the rich structure of the graphs in $\widetilde{Q}_{\mathcal{B}}$. For every two graphs $\widetilde{g}, \widetilde{h} \in \widetilde{Q}_{\mathcal{B}}$, determine if Y_{gh} is proper. If Y_{gh} is proper, test if it is contained in $L(\mathcal{B})$ by looking for a lasso with a prefix in $\widetilde{g}$ and a cycle in $\widetilde{h}$. In order to test if a proper language Y_{gh} is contained in $L(\mathcal{B})$, search for a $q \in Q^{in},\ r \in Q,\ a \in \{0,1\}$ such that the arc $\langle q, a, r\rangle \in \widetilde{g}$ and the arc $\langle r, 1, r\rangle \in \widetilde{h}$. We call this test of a pair of graphs the *two-arc test*.

Lemma 2. [17] *A Büchi automaton* $\mathcal{B}$ *is universal iff every proper pair* $\langle \widetilde{g}, \widetilde{h}\rangle$ *of graphs from* $\widetilde{Q}_{\mathcal{B}}$ *passes the two-arc test.*

Lemma 2 yields a PSPACE algorithm to determine universality [17]. Simply check each $\widetilde{g}, \widetilde{h} \in \widetilde{Q}_{\mathcal{B}}$. If Y_{gh} is both proper and not contained in $L(\mathcal{B})$, then the pair $\langle \widetilde{g}, \widetilde{h} \rangle$ provide a counterexample to the universality of $\mathcal{B}$. If no such pair exists, the automaton must be universal. This algorithm faces difficulty on two fronts. First, the number of graphs is 3^{n^2}. Second, checking language nonemptiness is an exponentially difficult problem. To address these problems we construct only graphs with non-empty languages. We borrow the notion of composition from [5], allowing us to use exponential space to compute exactly the needed graphs. Given a graph $\widetilde{g}$ whose language contains the word w_1 and a graph $\widetilde{h}$ whose language contains the word w_2, their composition $\widetilde{g}; \widetilde{h}$ can be defined such that $w_1 w_2 \in L(\widetilde{g}; \widetilde{h})$.

Definition 2. [5] *Given two graphs $\widetilde{g}, \widetilde{h} \in \widetilde{Q}_{\mathcal{B}}$, define their composition $\widetilde{g}; \widetilde{h}$ to be:*

$$\begin{aligned}
&\{\langle q, 1, r\rangle \mid q, r, s \in Q,\ \langle q, b, s\rangle \in \widetilde{g},\ \langle s, c, r\rangle \in \widetilde{h},\ b = 1 \textit{ or } c = 1\} \\
\cup &\{\langle q, 0, r\rangle \mid q, r, s \in Q,\ \langle q, 0, s\rangle \in \widetilde{g},\ \langle s, 0, r\rangle \in \widetilde{h},\ \textit{and} \\
&\quad \forall t \in Q,\ b, c \in \{0,1\}\ .\ \langle q, a, t\rangle \in \widetilde{g} \wedge \langle t, b, r\rangle \in \widetilde{h} \textit{ implies } a = b = 0\}
\end{aligned}$$

Using composition, we can define a concrete algorithm that explores the space of graphs on-the-fly, searching for a counterexample. Given a Büchi automaton $\mathcal{B}$, for every $\sigma \in \Sigma$, define $\widetilde{g}_\sigma$ to be $\{\langle q, 0, r\rangle \mid q \in Q \setminus F,\ r \in \rho(q, \sigma) \setminus F\} \cup \{\langle q, 1, r\rangle \mid q \in Q,\ r \in \rho(q, \sigma),\ q \text{ or } r \in F\}$. Let $\widetilde{Q}^1_{\mathcal{B}}$ be the set $\{\widetilde{g}_\sigma \mid \sigma \in \Sigma\}$. To generate the non-empty graphs, compose graphs from $\widetilde{Q}^1_{\mathcal{B}}$ until we reach closure. The resulting subset of $\widetilde{Q}_{\mathcal{B}}$, written $\widetilde{Q}^f_{\mathcal{B}}$, contains exactly the graphs with non-empty languages. In addition to non-emptiness, properness requires testing language containment. Recall that a pair of graphs $\langle \widetilde{g}, \widetilde{h} \rangle$ with non-empty languages is proper when both $L(\widetilde{g}) \cdot L(\widetilde{h}) \subseteq L(\widetilde{g})$, and $L(\widetilde{h}) \cdot L(\widetilde{h}) \subseteq L(\widetilde{h})$. We employ composition to provide a novel polynomial time test for the containment of graph languages.

Lemma 3. *For any $\widetilde{g}, \widetilde{h}, \widetilde{k} \in \widetilde{Q}^f_{\mathcal{B}}$, it holds that $L(\widetilde{g}) \cdot L(\widetilde{h}) \subseteq L(\widetilde{k})$ iff $\widetilde{g}; \widetilde{h} = \widetilde{k}$*

Algorithm 1 employs composition to search for proper pairs of graphs and check the universality of a Büchi automaton $\mathcal{B}$. On non-universal automaton, this algorithm can terminate as soon as it finds a counterexample, and thus sometimes avoid computing the entire set of graphs.

Algorithm 1: `RamseyUniversality(`$\mathcal{B}$`)`

```
Initialize $\widetilde{Q}^f_{\mathcal{B}} \Leftarrow \widetilde{Q}^1_{\mathcal{B}}$
repeat
    Take two graphs $\widetilde{g}, \widetilde{h} \in \widetilde{Q}^f_{\mathcal{B}}$
    Include $\widetilde{g}; \widetilde{h}$ in $\widetilde{Q}^f_{\mathcal{B}}$
    if $\widetilde{g}; \widetilde{h} = \widetilde{g}$ and $\widetilde{h}; \widetilde{h} = \widetilde{h}$ then
        if $\langle \widetilde{g}, \widetilde{h} \rangle$ fails the two-arc test then return Not Universal
until $\widetilde{Q}^f_{\mathcal{B}}$ reaches fixpoint
return Universal
```

2.2 Rank-Based Complementation

If a Büchi automaton $\mathcal{B}$ does not accept a word w, then every run of $\mathcal{B}$ on w must eventually cease visiting accepting states. The rank-based construction uses a notion of ranks to track the progress of each possible run towards this fair termination. The rank-based construction accepts w precisely if all runs cease visiting accepting states, and so defines a automaton for the complement of $L(\mathcal{B})$. For a definition of this construction, see [12].

An algorithm seeking to refute the universality of $\mathcal{B}$ can look for a lasso in the state-space of the rank-based complement of $\mathcal{B}$. A classical approach is Emerson-Lei backward-traversal nested fixpoint $\nu Y.\mu X.(X \cup (Y \cap F))$ [8]. This nested fixpoint employs the observation that a state in a lasso can reach an arbitrary number of accepting states. The outer fixpoint iteratively computes sets $Y_0, Y_1, ...$ such that Y_i contains all states with a path visiting i accepting states. Universality is checked by testing if Y_∞, the set of all states with a path visiting arbitrarily many accepting states, intersects Q^{in}. In contrast to the Ramsey-based approach, this rank-based approach can terminate early on some *universal* automaton, when some Y_i is already disjoint from Q^{in}. If no initial state has a path to i accepting states, then no initial state can lead to a lasso. In this case we already know the complemented automaton is empty, and the original automaton is universal. In consequence, extracting a counter-example from the Emerson-Lei algorithm is non-trivial, and requires that the algorithm fully terminates. Doyen and Raskin implemented this algorithm using a subsumption relation, providing a universality checker that scales to automata an orders of magnitude larger than previous approaches [6].

3 Subsumption in the Ramsey-based algorithm

Subsumption has proven to be very effective in the rank-based approach [6] and in the Ramsey-based approach specialized to SCT problems [1]. To use subsumption in the special case of SCT problems, Ben-Amram and Lee replaced a test for an arc in idempotent graphs with a test for strongly-connected components in all graphs. To use subsumption in the general Ramsey-based approach, we need to replace the two-arc test over proper pairs of graphs. We simplify Algorithm 1 by removing the requirement that pairs of graphs should be proper. Instead of examining only pairs $\langle \widetilde{g}, \widetilde{h} \rangle$ where $\widetilde{g}; \widetilde{h} = \widetilde{g}$ and $\widetilde{h}; \widetilde{h} = \widetilde{h}$, we examine every pair $\langle \widetilde{g}, \widetilde{h} \rangle$ of non-empty graphs. When examining a proper pair of graphs, we used the two-arc test: search for a $q \in Q^{in},\ r \in Q,\ a \in \{0, 1\}$ such that $\langle q, a, r \rangle \in \widetilde{g}$ and $\langle r, 1, r \rangle \in \widetilde{h}$. When examining a pair of graphs that may not be proper, we cannot limit our search to single arcs. We must test for a path from q to r, and a path from r to itself. We test for this path by computing the strongly connected components of $\widetilde{h}$, and testing if some strongly connected component of $\widetilde{h}$ both contains a 1-labeled arc and is reachable from a start state in $\widetilde{g}$.

A *strongly connected component* (SCC) of a graph $\widetilde{g}$ is a maximal set S of nodes, so that for every $q, r \in S$ there is a path from q to r, and a path from r to q. Computing the strongly connected components of a graph can be done in linear time with a depth-first search [4]. An SCC S in a graph $\widetilde{g}$ is 1-labeled when there are $q, r \in S$ with an arc $\langle q, 1, r \rangle \in \widetilde{g}$. We say there is a path from a state q to an SCC S when there is a path from q to an element of S. Once we partition the nodes into strongly connected components, we can simply search for a reachable 1-labeled SCC.

Definition 3. *A pair $\langle \widetilde{g}, \widetilde{h} \rangle$ of graphs passes the* lasso-finding test *when there exists: $q \in Q^{in},\ r \in Q,\ a \in \{0,1\}$ and $S \subseteq Q$ such that, $\langle q, a, r \rangle \in \widetilde{g}$, there is a path from r to S in $\widetilde{h}$, and S is a 1-labeled SCC of $\widetilde{h}$.*

Lemma 4. *$\widetilde{Q}^{f}_{\mathcal{B}}$ contains a pair $\langle \widetilde{g}, \widetilde{h} \rangle$ that fails the lasso-finding test iff $\widetilde{Q}^{f}_{\mathcal{B}}$ contains a pair of graphs $\langle \widetilde{g}', \widetilde{h}' \rangle$ that fails the two-arc test.*

In [9], we demonstrated that the Lee, Jones, and Ben-Amram algorithm for size-change termination is a specialized realization of the Ramsey-based containment test. In [1], Ben-Amram and Lee optimize this specialized algorithm, removing certain graphs when computing the closure under composition. Using the lasso-finding test, we now show how to employ Ben-Amram and Lee's subsumption relation for the general case of Büchi universality. Doing so allows us to ignore graphs when they are approximated by other graphs.

Intuitively, a graph $\widetilde{g}$ approximates another graph $\widetilde{h}$ when the arcs of $\widetilde{g}$ are a subset of, or less strict than, the arcs of $\widetilde{h}$. In this case, finding an arc or SCC in $\widetilde{g}$ is strictly harder than finding one in than $\widetilde{h}$. When the right arc can be found in $\widetilde{g}$, then it also occurs in $\widetilde{h}$. When $\widetilde{g}$ does not have a satisfying arc, then we already have a counterexample. Thus we need not consider $\widetilde{h}$.

Formally, given two graphs $\widetilde{g}, \widetilde{h} \in \widetilde{Q}_{\mathcal{B}}$, we say that $\widetilde{g}$ *approximates* $\widetilde{h}$, written $\widetilde{g} \preceq \widetilde{h}$, when for every arc $\langle q, a, r \rangle \in \widetilde{g}$ there is an arc $\langle q, a', r \rangle \in \widetilde{h},\ a \leq a'$. Note that approximation is a transitive relation. Using the notion of approximation, we present an algorithm that computes a subset of $\widetilde{Q}^{f}_{\mathcal{B}}$, called $\widetilde{Q}^{\preceq}_{\mathcal{B}}$. A set of graphs $\widetilde{Q}$ is *$\preceq$-closed under composition* when for every $\widetilde{g}, \widetilde{h} \in \widetilde{Q}$, there exists $\widetilde{k} \in \widetilde{Q}$ such that $\widetilde{k} \preceq \widetilde{g}; \widetilde{h}$. Given a set $\widetilde{Q}^{1}_{\mathcal{B}}$ of graphs, Algorithm 2 computes a set $\widetilde{Q}^{\preceq}_{\mathcal{B}}$ by keeping only the minimal elements under the $\preceq$ relation. $\widetilde{Q}^{\preceq}_{\mathcal{B}}$ will be $\preceq$-closed under composition, but not closed under composition in the normal sense.

Note that the lasso-finding test is required to safely limit our search to graphs in $\widetilde{Q}^{\preceq}_{\mathcal{B}}$. Since we are now removing elements from $\widetilde{Q}^{\preceq}_{\mathcal{B}}$, it is possibly that the proper pair of graphs in $\widetilde{Q}^{f}_{\mathcal{B}}$ that fails the two-arc test may never be computed: a graph in the pair may be approximated by another graph, one that does not satisfy the conditions of properness. When using the lasso-finding test, on the other hand, we examine all pairs of graphs. As an example, consider the set containing the single graph $\widetilde{g} = \{\langle q, 0, q \rangle, \langle q, 0, r \rangle, \langle r, 0, q \rangle\}$. We leave it to the reader to verify that this set is $\preceq$-closed under composition, and that $\langle \widetilde{g}, \widetilde{g} \rangle$ fails the lasso-finding test, but that $\langle \widetilde{g}, \widetilde{g} \rangle$ is not proper. Similarly, if we consider the graph $\widetilde{g} = \{\langle q, 1, r \rangle, \langle r, 1, q \rangle\}$, we find that $\langle \widetilde{g}, \widetilde{g} \rangle$ fails the two-arc test, but passes the lasso-finding test.

Theorem 1. *Given an initial set $\widetilde{Q}^{1}_{\mathcal{B}}$ of graphs, the set $\widetilde{Q}^{f}_{\mathcal{B}}$ contains a proper pair $\langle \widetilde{g}, \widetilde{h} \rangle$ of graphs that fails the two-arc test if and only if the set $\widetilde{Q}^{\preceq}_{\mathcal{B}}$ computed in Algorithm 2 contains a pair $\langle \widetilde{g}', \widetilde{h}' \rangle$ of graphs that fails the lasso-finding test.*

Algorithm 2: `RamseyUniversality(`$\mathcal{B}$`)`

```
Construct the set $\widetilde{Q}^1_{\mathcal{B}}$ of all single-character graphs
Initialize the worklist $\widetilde{W} \Leftarrow \widetilde{Q}^1_{\mathcal{B}}$
Initialize the set $\widetilde{Q}^{\preceq}_{\mathcal{B}} \Leftarrow \emptyset$
while $\widetilde{W} \neq \emptyset$ do
    Remove an element $\widetilde{g}$ from $\widetilde{W}$
    for $\widetilde{h} \in \widetilde{Q}^{\preceq}_{\mathcal{B}}$ do
        if $\widetilde{h} \preceq \widetilde{g}$ then
            Discard $\widetilde{g}$ and exit for
        else if $\widetilde{g} \preceq \widetilde{h}$ then
            Remove $\widetilde{h}$ from $\widetilde{Q}^{\preceq}_{\mathcal{B}}$
        else if $\langle \widetilde{g}, \widetilde{h} \rangle$ or $\langle \widetilde{h}, \widetilde{g} \rangle$ fails the lasso-finding test then
            return Not Universal
    if $\widetilde{g}$ has not been discarded then
        Add $\widetilde{g}$ to $\widetilde{Q}^{\preceq}_{\mathcal{B}}$
        for $\widetilde{h} \in \widetilde{Q}^1_{\mathcal{B}}$ do Add $\widetilde{g}; \widetilde{h}$ to $\widetilde{W}$
return Universal
```

Based on the algorithm used by Ben-Amram and Lee, Algorithm 2 extends Algorithm 1 to exploit subsumption and avoid computing the entirety of $\widetilde{Q}^f_{\mathcal{B}}$[1]. To make the algorithm more concrete, a worklist is used to keep track of which graphs have yet to be considered. Further, instead of composing arbitrary pairs of graphs, we compose each graph only with graphs from $\widetilde{Q}^1_{\mathcal{B}}$. Since any composition can be phrased as a sequence of compositions of graphs from $\widetilde{Q}^1_{\mathcal{B}}$, this is sufficient to generate the entirety of $\widetilde{Q}^{\preceq}_{\mathcal{B}}$ while reducing the size of the worklist considerably. To achieve reasonable performance, our implementation memoizes the strongly connected components of graphs and implements the lasso-finding test as an intersection test over two sets of states.

4 Empirical Analysis

The subsumption technique employed in Algorithm 2 is purely a heuristic improvement: the worst-case complexity of the algorithm does not change. Thus the Ramsey-based algorithm has a worst-case running time exponentially slower than that of the rank-based algorithm. Motivated by the strong performance of Ramsey-based algorithms on SCT problems [9], we compare Ramsey and rank based solvers on a terrain of random automata.

To evaluate the performance of various tools on Büchi universality problems, we employ the random model proposed by Tabakov and Vardi and later used by Doyen and Raskin [6, 18]. This model fixes the input alphabet as $\Sigma = \{0, 1\}$ and considers the containment of Σ^ω in, and thus the *universality* of, the language of a random automata. Each automaton $\mathcal{B} = \langle \Sigma, Q, Q^{in}, \rho, F \rangle$ is constructed with a given *size* n, *transition*

[1] This algorithm does not prune $\widetilde{Q}^1_{\mathcal{B}}$ for subsumed graphs. As our alphabet consists of two characters, and $\widetilde{Q}^1_{\mathcal{B}}$ contains two elements, this is acceptable for our use. For larger alphabets, $\widetilde{Q}^1_{\mathcal{B}}$ could be checked for subsumed graphs.

density r, and *acceptance density* f. Q is simply the set $\{0...n-1\}$, and $Q^{in} = \{0\}$. For each letter $\sigma \in \Sigma$, we choose $\lceil n * r \rceil$ pairs of states $(s, s') \in Q^2$ uniformly at random and the transitions $\langle s, \sigma, s' \rangle$ are included in ρ. We impose one exception to avoid trivial cases of non-universality: the initial node must have at least one outgoing transition for each letter of the alphabet. The set F of accepting states comprises $\lceil n * f \rceil$ states, likewise chosen uniformly at random.

Data points are derived from 100 or more[2] random automata with the given n, r, and f. Each tool is given one hour to solve each problem. When possible, we compute the median running time [6, 18]. This allows us to plot the data on a logarithmic scale and easily judge exponential behavior. However, in many cases interesting behavior emerges after a significant percentage of the runs time out. In these cases we measure the timeout percentage instead of median running time.

Our rank-based tool, simply called RANK, is a slightly modified version of the **Mh** tool developed by Doyen and Raskin [6]. Our Ramsey-based tool, called RAMSEY, is based on the **sct/scp** program– an optimized C implementation of the SCT algorithm from Ben-Amram and Lee [1]. We have modified the RAMSEY tool to solve arbitrary Büchi universality problems by implementing Algorithm 2. Both tools can be configured to not employ their subsumption techniques. In this case, we append (ns) to the program name.

All experiments were performed on the Shared University Grid at Rice (SUG@R)[3], a cluster of Sunfire x4150 nodes, each with two 2.83GHz Intel Xeon quad-core processors and 16GB of RAM. Each run is given a dedicated node.

4.1 Subsumption

We know that subsumption is vital to the performance of rank-based solvers [6]. Further, we have observed subsumption's utility on the domain of SCT problems [9]. This motivates us to extend the subsumption technique of [1] to the case of general Büchi universality, resulting in Algorithm 2. Employing observations from Section 4.2 below, we check the practical utility of subsumption on the most difficult terrain point for RAMSEY, where transition density $r = 1.5$ and acceptance density $f = 0.5$. Figure 1 displays RAMSEY's performance as size increases, on a logarithmic scale. If more than 50% of the problems timed out, the median is displayed at 3600 seconds, which flattens the RAMSEY (ns) line at the last data point. We observe that the RAMSEY (ns) line has a higher slope than the RAMSEY line. As this graph uses a logarithmic scale, this difference in the slope indicates an exponential improvement in scalability when subsumption is used. Similar results held for every terrain point we measured, demonstrating that although a heuristic technique, subsumption is required for the scalability of our Ramsey-based approach. We also note that the curves appear to be linear on the logarithmic scale, suggesting that the median running time for this terrain point is $2^{O(n)}$, rather than the $2^{O(n^2)}$ of the worst-case complexity bound.

[2] When the results from 100 automata appear anomalous, additional automata are generated and tested to improve the fidelity of the results. No data are ever excluded.

[3] `http://rcsg.rice.edu/sugar/`

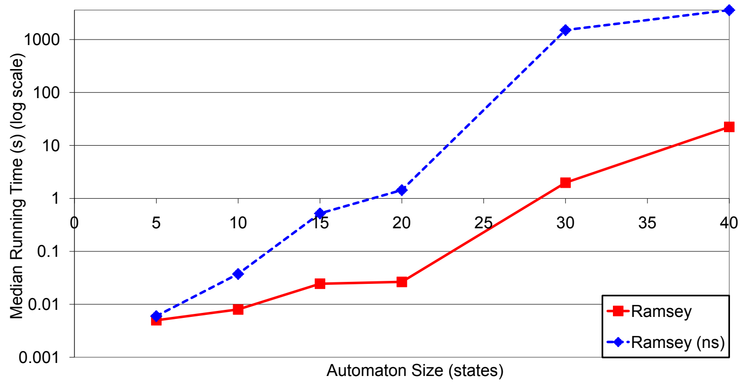

Fig. 1: Subsumption exponentially improves RAMSEY median running times ($r = 1.5, f = 0.5$)

4.2 Behavior over the terrain

As stated above, randomly generated automata of a given size can vary in two parameters. By changing the transitions and acceptance density, we can observe the behavior of each tool over a variety of terrain points. Automata with a high transition density tend to be universal, while automata with low transition density tend to be non-universal. Acceptance density has a smaller, but still noticeable, affect on universality [6, 18]. To map out the behavior of the two tools over this terrain, we hold size constant at $n = 100$, and examined a variety of terrain points. We generate data points for each combination of transition density $r \in \{0.02, 0.26, 0.50, 0.74, 0.98\}$ and acceptance density $f \in \{0.5, 1.5, 2.0, 2.5, 3.0\}$.

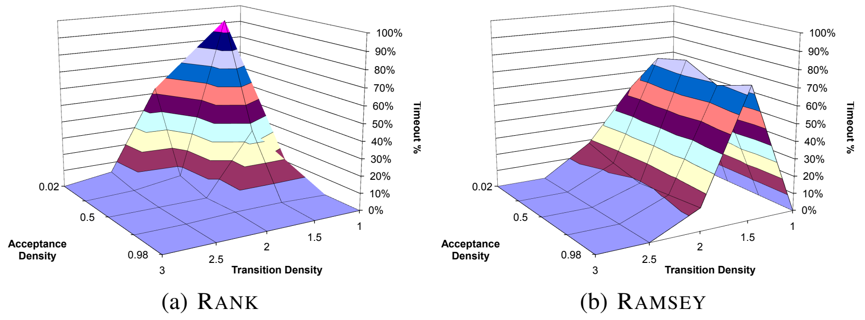

Fig. 2: Differences in behavior between RANK and RAMSEY over problem terrain, measured as percentage of problems that timeout when size $n = 100$

Figure 2(a) displays the percentage of cases in which the RANK tool timed out in each terrain point. As observed in [6], there is a sharp spike in timeouts at transitions density $r = 1.5$, acceptance density of 0.26. This spike trails off quickly as transition density changes, and only slightly more gradually as acceptance density changes. There is a subtler high point at $r = 2.0$, $f = 0.02$, where the timeouts rise to 50%. This is

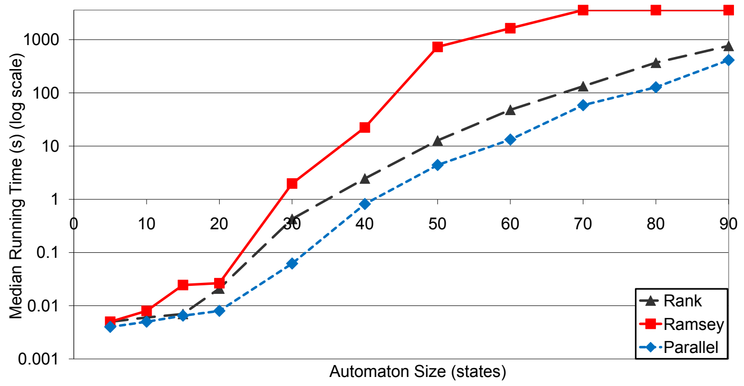

Fig. 3: RANK scales exponentially better than RAMSEY when $r = 1.5$ and $f = 0.5$ (log scale)

consistent with other rank-based tools, even those using different algorithms [18]. Figure 2(b) displays the percentage of cases in which the RAMSEY tool timed out in each terrain point. Like RANK, $r = 1.5$, $f = 0.26$ is a difficult terrain point for RAMSEY. However, RAMSEY continues to time out frequently along all terrain points with transition density $r = 1.5$, and has no significant timeouts at any other terrain points.

Simply glancing at the terrain graphs, it appears that RANK may perform better than RAMSEY in most terrain points. On the other hand, RAMSEY does not exhibit a second high point at $r = 2.0$, $f = 0.02$, and at least for this size of automata RAMSEY beats RANK at the hardest point for RANK. What these graphs clearly show is that those attributes that make a problem hard for RANK to handle are not necessarily the same as those attributes of a problem that cause difficulty for RAMSEY.

4.3 Scalability

We explore some interesting terrain points by measuring the scalability of each algorithm: we hold the transition and acceptance densities constant, and increase size. We choose to investigate three terrain points: a point $r = 1.5$, $f = 0.5$, where RANK seems to perform better than RAMSEY; the main spike $r = 1.5$, $f = 0.26$, where both tools exhibited difficulty solving problems; and a final point $r = 2.0$, $f = 0.05$ near RANK's second high point, where RAMSEY seems to perform better.

Figure 3 displays the median running time for problems with the transition density at $r = 1.5$ and the acceptance density at $f = 0.5$, on a logarithmic scale. If more than 50% of the problems timed out, the median is displayed at 3600 seconds, cutting off RAMSEY's line. As the scale is logarithmic, the difference in the slope between RANK's line and RAMSEY's indicates that, on this terrain point, RANK clearly scales exponentially better than RAMSEY. The third line, labeled "Parallel", displays the behavior of running both tools in parallel on separate machines, and terminating as soon as either tool gives an answer. Is is notable that this line, while having the same slope as RANK's, is lower; indicating there are a number of cases even at this terrain point where RAMSEY terminates before RANK.

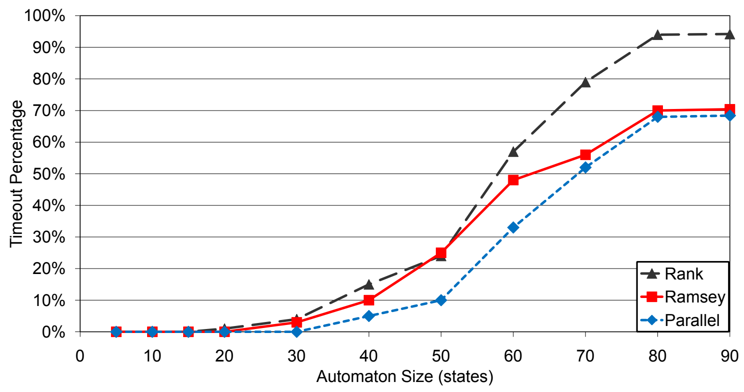

Fig. 4: RAMSEY scales better than RANK at the most difficult terrain point ($r = 1.5$, $f = 0.26$)

The most difficult terrain points for both tools lie near $r = 1.5$, $f = 0.26$. Up to $n = 50$, median running times (see addendum) indicate that RAMSEY performs better than RANK only by a constant factor. Past this this size, the percentage of timeouts is too high for median measurements to be meaningful. However, a gap in the timeout percentage appears as the automata grow larger than 50 states. Figure 4 displays the percentage of runs that timed out for each size n at this terrain point. It does appear that, past $n = 50$, RAMSEY begins to scale significantly better than RANK. We again display the behavior of running both tools in parallel on separate machines using the third line, labeled "Parallel." We again find that even at a terrain point that favors one tool, RAMSEY, we benefit from running both tools simultaneously.

At size $n = 100$, RANK exhibited difficulty when the transition density was 2.0 and the acceptance density was low. We measured the scalability of RAMSEY and RANK on problems with $r = 2.0$ and $f = 0.05$. At this terrain point the median running times do not increase exponentially for either RANK or RAMSEY. As a large number of problems still did not complete, Figure 5 displays the timeout percentages as size grows. At this terrain point, RAMSEY does appear to scale better than RANK. However the gap is not the exponential improvement we observed when RANK performed better than RAMSEY. At this configuration, running the tools in parallel was only a slight improvement over running RAMSEY alone.

4.4 Universal vs. non-universal automata

Section 2 reviews the algorithms used by RANK and RAMSEY to explore the state space of the complemented automaton. Of note is that in certain cases each tools can terminate before computing the entirety of their fixpoints: RANK on universal automata, and RAMSEY on non-universal automata. This suggests that RANK may perform better on universal automata, and RAMSEY may perform better on non-universal automata.

To confirm this hypothesis, we compare RANK and RAMSEY on a corpus of universal and non-universal automata. Our corpus is derived from 1000 automata with size $n = 50$, transition density $r = 1.8$, and acceptance density $f = 0.2$. This point was chosen because of the relatively equal proportion of universal and non-universal au-

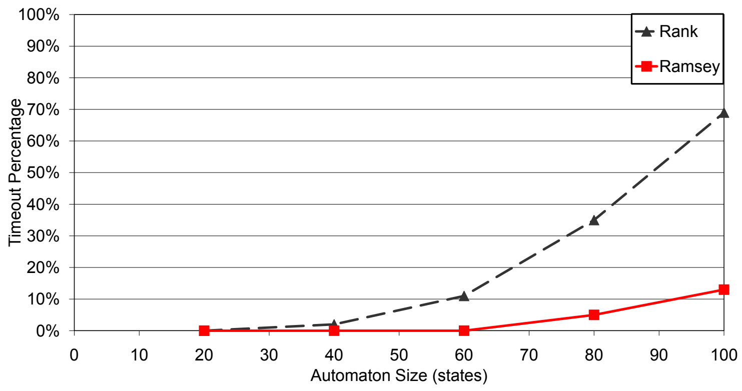

Fig. 5: RAMSEY scales much better than RANK when $r = 2$ and $f = 0.05$

tomata. Table 1 summarizes the results. RANK does indeed perform better on universal automata. Universal automata were solved in a median time of 108.3 seconds, while on non-universal automata, the median running time was 177.8 seconds. We observe the inverse behavior in RAMSEY: on non-universal automata RAMSEY had a median running time of only 33.1 seconds, while on universal automata the median running time was 253.4 seconds. The universality or non-universality of a problem does affect the performance of each approach.

	Count	RANK	RAMSEY
Universal	460	108.3	253.4
Non-Universal	527	177.8	33.1
Unknown	13		

Table 1: RANK performs better on universal problems, RAMSEY on non-universal problems, measured by median running time ($n = 50, r = 1.8, f = 0.2$)

The question naturally arises: does the difference in performance on universal vs. non-universal automata fully explain the different behaviors of RAMSEY and RANK. This is not the case. As previously noted in Figure 3, RANK performs exponentially better than RAMSEY on automata with a transition density of 1.5 and an acceptance density of 0.5. More than 80% of the solved automata at this terrain point are non-universal: a distribution that should favor RAMSEY. Further, Figure 5 shows a terrain point where RAMSEY scales significantly better than RANK. At this terrain point, more than two-thirds of solved automata with $n > 50$ were universal, and should have favored RANK. Therefore we cannot conclude that the difference in behavior between RANK and RAMSEY is truly attributed to the gap in performance between universal and non-universal automata.

5 Conclusion

This paper tells two stories. The first story is about subsumption. In general, subsumption is a trade off: there is a benefit to reducing the working sets of the algorithms, but checking for subsumed states can be computationally expensive. In the domain of CNF satisfiability solvers, subsumption is generally regarded as an ineffective technique: the overhead of checking for subsumed clauses outweighs any benefit gained from removing them. For checking Büchi automata universality, it has previously been shown that subsumption is not only useful, but vital for the scalability of the rank-based approach [6]. In this paper, we demonstrate that this also holds for the Ramsey-based approach, which use not only a different construction but also a different algorithm to explore the state space of this construction. These results suggest the use of subsumption relations in other constructions, such as the slice-based construction of Kähler and Wilke [11].

The second story is that neither the rank-based approach nor the Ramsey-based approach to Büchi universality testing is clearly superior. This is true despite the massive gap in worst-case complexity between the two approaches. Each approach exhibits distinct behavior on the terrain of random universality problems. Due to these differences, we do not believe a winner takes all approach is best for universality checking. The current best approach is to run both tools in parallel, and see which terminates first. Doing so improves performance by a constant factor, relative to the best tool for any given terrain point.

Preferable to running the algorithms in parallel would be to employ a *portfolio approach*. A portfolio approach attempts to predict which algorithm would perform better on a given problem [10]. To do this, we would have to examine the space of universality problems and discover significant attributes of problems. Transition and acceptance density are not the only observable attributes of an automaton, or even necessarily the most important ones. While they are significant for randomly generated problems, there is no reason to expect that transition and acceptance density are good indicators of difficulty for real-world problems. In the case of SAT solvers, over ninety pertinent attributes were found [7]. Machine-learning techniques were used to identify which features suggest which approach to SAT solving. The challenge that now faces us is discovering a similar body of features with which to characterize Büchi automata, and to create a corpus of automata to characterize. In addition to transition and acceptance density, attributes could include the density of initial states, the number of strongly connected components in the automata, and the density of strongly connecting components containing an accepting state[4]. One point that is well demonstrated in our investigation is that theoretical worst-case analysis often yields little information on actual algorithmic performance; an algorithm running in $2^{O(n^2)}$ can perform better in practice than an algorithm running in $2^{O(n \log n)}$. We do note RAMSEY, the program running in $2^{O(n^2)}$ time and space, sometimes consumed on the order of 20 GB of memory, where RANK rarely consumed more than 300 megabytes.

Finally, in this paper we focus on universality as a special case of Büchi containment that encapsulates its algorithmically difficult aspects. To actually verify that an implementation $\mathcal{A}$ adheres to a specification $\mathcal{B}$, we need to lift our universality testing

[4] We thank the reviewers for suggestions on possible criteria.

algorithms to the general case of containment testing. Computing the intersection of two automata uses the product of the state spaces. For the rank-based approach, this results in pairing a state $\mathcal{A}$ with a state in in $\texttt{KV}(\mathcal{B})$. The theory of rank-based containment testing with subsumption is described in [6] and implemented in RANK. Ramsey-based universality, however, avoids directly exploring the state space of the automata. A theory of Ramsey-based containment was developed for [9], but without subsumption. To add containment testing to RAMSEY requires the extension of the theory developed in this paper for universality testing.

References

1. A.M. Ben-Amram and C.S. Lee. Program termination analysis in polynomial time. In *TOPLAS*, 29(1) (2007)
2. J.R. Büchi. On a decision method in restricted second order arithmetic. In *ICLMPS* (1962)
3. Y. Choueka. Theories of automata on ω-tapes: A simplified approach. In *JCSS* (1974)
4. T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. MIT Press and McGraw-Hill (1990)
5. N.D. Jones C.S. Lee and A.M. Ben-Amram. The size-change principle for program termination. In *POPL*, pp. 81–92 (2001)
6. L. Doyen and J.-F. Raskin. Antichains for the automata-based approach to model-checking. In *LMCS*, 1(5) pp. 1–20 (2009)
7. A Devkar Y. Shoham E. Nudelman, K. Leyton-Brown and H. Hoos. Understanding random SAT: Beyond the clauses-to-variables ratio. In *CP*, pp. 438–452 (2004)
8. E.A. Emerson and C.-L. Lei. Efficient model checking in fragments of the propositional μ-calculus. In *LICS*, pp. 267–278 (1986)
9. S. Fogarty and M.Y. Vardi. Büchi complementation and size-change termination. In *TACAS*, pp. 16–30 (2009)
10. G Andrew J McFadden K. Leyton-Brown, E. Nudelman and Y. Shoham. A portfolio approach to algorithm selection. In *IJCAI*, pp. 1542–1543 (2003)
11. D. Kähler and T. Wilke. Complementation, disambiguation, and determinization of Büchi automata unified. In *ICALP*, pp. 724–735 (2008)
12. O. Kupferman and M.Y. Vardi. Weak alternating automata are not that weak. *ACM Transactions on Computational Logic*, 2(2):408–429 (2001)
13. M. Michel. Complementation is more difficult with automata on infinite words. CNET, Paris (1988)
14. K.Y. Rozier and M.Y. Vardi. LTL satisfiability checking. In *SPIN*, pp. 149–167 (2007)
15. S. Safra. On the complexity of ω-automata. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pp. 319–327 (1988)
16. S. Schewe. Büchi complementation made tight. In *STACS*, pp. 661–672 (2009)
17. A.P. Sistla, M.Y. Vardi, and P. Wolper. The complementation problem for Büchi automata with applications to temporal logic. In *ICALP*, volume 194, pp. 465–474. Springer, (1985)
18. D. Tabakov and M.Y. Vardi. Model checking Büchi specifications. In *LATA* (2007)
19. M.Y. Vardi. Automata-theoretic model checking revisited. In *VMCAI*, volume 4349 of *Lecture Notes in Computer Science*, pp. 137–150. Springer (2007)
20. M.D. Wulf, L. Doyen, T. A. Henzinger, and J.-F. Raskin. Antichains: A new algorithm for checking universality of finite automata. In *CAV*, pp. 17–30 (2006)

A Proof of Lemma 4

To prove Lemma 4 correct, we proceed in two steps. Recall that the original algorithm tested only pairs of graphs $\langle \widetilde{g}, \widetilde{h} \rangle$ where $\widetilde{g}; \widetilde{h} = \widetilde{g}$ and $\widetilde{h}; \widetilde{h} = \widetilde{h}$. In the first step, we remove the requirement that $\widetilde{g}; \widetilde{h} = \widetilde{g}$. To do so, we replace the two-arc test with a three-arc test. This allows us to test every pair of a graph $\widetilde{g}$ and an idempotentgraph $\widetilde{h}$. In the second step, we remove the requirement that $\widetilde{h}; \widetilde{h} = \widetilde{h}$. This is accomplished by altering the three-arc test to search for paths through $\widetilde{h}$ instead of single arcs, resulting in the lasso-finding test. This allows us to test every pair of graphs.

Algorithm 1 performs the two-arc test on every pair $\langle \widetilde{g}, \widetilde{h} \rangle$ of graphs where $\widetilde{g}; \widetilde{h} = \widetilde{g}$ and $\widetilde{h}; \widetilde{h} = \widetilde{h}$. The first requirement can be eliminated by modifying the two-arc test. Say that a pair $\langle \widetilde{g}, \widetilde{h} \rangle$ of graphs passes the *three-arc test* when there exists a $q \in Q^{in}$, $r, s \in Q$ and $a, a' \in \{0, 1\}$ such that $\langle q, a, r \rangle \in \widetilde{g}$, $\langle r, a', s \rangle \in \widetilde{h}$, and $\langle s, 1, s \rangle \in \widetilde{h}$. Note that a pair of graphs passing the two-arc test also passes the three-arc test: take r to be s and a' to be 1.

Lemma 5. *Given a pair $\langle \widetilde{g}, \widetilde{h} \rangle$ of graphs in $\widetilde{Q}^f_{\mathcal{B}}$ where $\widetilde{h}; \widetilde{h} = \widetilde{h}$, let $\widetilde{g}' = \widetilde{g}; \widetilde{h}$.*
(1) The pair $\langle \widetilde{g}', \widetilde{h} \rangle$ is proper
(2) $\langle \widetilde{g}, \widetilde{h} \rangle$ passes the three-arc test iff $\langle \widetilde{g}', \widetilde{h} \rangle$ passes the two-arc test

Proof:

(1) As $\widetilde{g}$ and $\widetilde{h}$ are both in $\widetilde{Q}^f_{\mathcal{B}}$, they both have non-empty languages. We are given that $\widetilde{h}; \widetilde{h} = \widetilde{h}$. All that remains is to prove that $\widetilde{g}'; \widetilde{h} = \widetilde{g}'$. By definition, it holds that $\widetilde{g}; \widetilde{h} = \widetilde{g}'$. Since $\widetilde{h}; \widetilde{h} = \widetilde{h}$, this is equivalent to $\widetilde{g}; (\widetilde{h}; \widetilde{h}) = \widetilde{g}'$. Since composition is associative, this can be rewritten as $(\widetilde{g}; \widetilde{h}); \widetilde{h} = \widetilde{g}'$, which is to say $\widetilde{g}'; \widetilde{h} = \widetilde{g}'$.

(2) Presume that $\langle \widetilde{g}, \widetilde{h} \rangle$ passes the three-arc test. This is true when there exists $q \in Q^{in}$, $r, s \in Q$ and $a, a' \in \{0, 1\}$ such that $\langle q, a, r \rangle \in \widetilde{g}$, $\langle r, a', s \rangle \in \widetilde{h}$, and $\langle s, 1, s \rangle \in \widetilde{h}$. By the definition of composition, $\langle q, a, r \rangle \in \widetilde{g}$ and $\langle r, a', s \rangle \in \widetilde{h}$ implies that $\langle q, a'', s \rangle \in \widetilde{g}'$, $a'' = \max(a, a')$. Therefore the two-arc test for $\langle \widetilde{g}', \widetilde{h} \rangle$ is satisfied by the states q and s, and the arcs $\langle q, a'', s \rangle$ and $\langle s, 1, s \rangle$.
Conversely, assume that $\langle \widetilde{g}', \widetilde{h} \rangle$ satisfies the two-arc test. This implies there are $q \in Q^{in}$, $r \in Q, a \in \{0, 1\}$ such that $\langle q, a, r \rangle \in \widetilde{g}'$ and $\langle r, 1, r \rangle \in \widetilde{h}$. By the definition of composition, since $\langle q, a, r \rangle \in \widetilde{g}; \widetilde{h}$, there must be some $t \in Q$ and $a', a'' \in \{0, 1\}$ such that $\langle q, a', t \rangle \in \widetilde{g}$ and $\langle t, a'', r \rangle \in \widetilde{h}$. Therefore the three-arc test for $\langle \widetilde{g}, \widetilde{h} \rangle$ is satisfied by the states q, t and r, and the arcs $\langle q, a', t \rangle$, $\langle t, a'', r \rangle$, and $\langle r, 1, r \rangle$.

□

Lemma 6. *$\widetilde{Q}^f_{\mathcal{B}}$ contains a pair $\langle \widetilde{g}, \widetilde{h} \rangle$ of graphs that is proper and fails the two-arc test if and only if $\widetilde{Q}^f_{\mathcal{B}}$ contains a pair $\langle \widetilde{g}', \widetilde{h}' \rangle$ of graphs where $\widetilde{h}'; \widetilde{h}' = \widetilde{h}'$ that fails the three-arc test.*

Proof: In one direction, assume $\widetilde{Q}^f_{\mathcal{B}}$ contains a pair $\langle \widetilde{g}, \widetilde{h} \rangle$ of graphs where $\widetilde{g}; \widetilde{h} = \widetilde{g}$ and $\widetilde{h}; \widetilde{h} = \widetilde{h}$ that fails the two-arc test. We show by contradiction that $\langle \widetilde{g}, \widetilde{h} \rangle$ fails the three-arc test. If $\langle \widetilde{g}, \widetilde{h} \rangle$ passes the three-arc test, then by Lemma 5 $\langle \widetilde{g}; \widetilde{h}, \widetilde{h} \rangle$ must pass the two-arc test. However, by assumption $\widetilde{g}; \widetilde{h} = \widetilde{g}$, and thus $\langle \widetilde{g}, \widetilde{h} \rangle$ would pass the two-arc test: a contradiction.

In the other direction, assume $\widetilde{Q}^f_{\mathcal{B}}$ contains a pair $\langle \widetilde{g}', \widetilde{h}' \rangle$ of graphs where $\widetilde{h}'; \widetilde{h}' = \widetilde{h}'$ that fails the three-arc test. By Lemma 5, $\langle \widetilde{g}'; \widetilde{h}', \widetilde{h}' \rangle$ is proper and fails the two-arc test. As $\widetilde{Q}^f_{\mathcal{B}}$ is closed under composition, $\widetilde{g}'; \widetilde{h}' \in \widetilde{Q}^f_{\mathcal{B}}$. □

Now we wish to remove the remaining requirement of the pair $\langle \widetilde{g}, \widetilde{h} \rangle$ of graphs; that $\widetilde{h}; \widetilde{h} = \widetilde{h}$. The three-arc test searches for $q \in Q^{in}$, $r, s \in Q$ and $a, a' \in \{0, 1\}$ such that $\langle q, a, r \rangle \in \widetilde{g}$, $\langle r, a', s \rangle \in \widetilde{h}$, and $\langle s, 1, s \rangle \in \widetilde{h}$. In searching for the arc $\langle s, 1, s \rangle$ we are asking if Y_{gh} has an accepting run that cycles through s. By ensuring that $\widetilde{h}$ is idempotent, we ensure that the edge relation of $\widetilde{h}$ is transitive. Thus, we know that every path in $\mathcal{B}$ over a word in $L(\widetilde{h})^+$ is described by an arc in $\widetilde{h}$. Thus, if there is no arc $\langle s, 1, s \rangle \in \widetilde{h}$, there cannot be an accepting cycle in $\mathcal{B}$ from s to s on a word in $L(\widetilde{h})^+$. If no reachable s has such an arc, we know there cannot be an accepting run on a word in $L(\widetilde{h})^\omega$. In order to remove the requirement that $\widetilde{h}; \widetilde{h} = \widetilde{h}$, we can no longer assume that a path in $\mathcal{B}$ from q to r on a word in $L(\widetilde{h})^+$ is matched by an arc $\langle q, a, r \rangle \in \widetilde{h}$. Instead, we must search for a sequence $p_0 p_1 ... p_{n-1} p_n$ of states in Q such that $p_0 = q$, $p_n = r$, and for each $0 \le i < n$ there is an arc $\langle p_i, a_i, p_{i+1} \rangle \in \widetilde{h}$ for some $a_i \in \{0, 1\}$.

To relate sequences of arcs in graphs to single arcs in idempotent graphs, we use exponentiation of graphs. Let $\widetilde{h}^n$ be the composition of n copies of $\widetilde{h}$, $\widetilde{h}; ...; \widetilde{h}$. Let $\widetilde{h}^\star$ be the first $\widetilde{h}^n$ where $\widetilde{h}^n; \widetilde{h}^n = \widetilde{h}^n$. Note that, as with all idempotent graphs, the edge relation of $\widetilde{h}^\star$ must be transitive. We first show that $\widetilde{h}^\star$ is well defined for every graph.

Lemma 7. *Let $\widetilde{h}$ be a graph. There exists an $n \in \mathbb{N}$ such that $\widetilde{h}^n; \widetilde{h}^n = \widetilde{h}^n$*

Proof: Consider the infinite sequence of graphs $\widetilde{h}^1, \widetilde{h}^2, \widetilde{h}^3, ...$ As there are a finite number of graphs, clearly there is a graph $\widetilde{h}'$ that occurs infinitely often in this sequence.

Let a be the index of the first occurrence of $\widetilde{h}'$ such that $\widetilde{h}^a = \widetilde{h}'$. Let b be the first index larger than $2a$ so that $\widetilde{h}^b = \widetilde{h}'$. Let $c = b - 2a$. We prove that $a + c$ is such an n, i.e. $\widetilde{h}^{a+c}; \widetilde{h}^{a+c} = \widetilde{h}^{a+c}$.
(1) As $\widetilde{h}^a = \widetilde{h}^b$, for every $i > 0$ it holds that $\widetilde{h}^{a+i} = \widetilde{h}^{b+i}$
(2) By the definition of exponentiation, $\widetilde{h}^{a+c}; \widetilde{h}^{a+c} = \widetilde{h}^{2a+2c}$
(3) By definition, $2a + c = b$
(4) By (2) and (3), $\widetilde{h}^{a+c}; \widetilde{h}^{a+c} = \widetilde{h}^{b+c}$
(5) By (1), $\widetilde{h}^{a+c} = \widetilde{h}^{b+c}$
(6) By (4) and (5), $\widetilde{h}^{a+c}; \widetilde{h}^{a+c} = \widetilde{h}^{a+c}$

□

We now want to demonstrate that connectivity between two nodes in $\widetilde{h}$ is strongly related to arcs between those two nodes in $\widetilde{h}^\star$. Once we have done so, we go on to prove that a set of graphs $\langle \widetilde{g}, \widetilde{h} \rangle$ passes the lasso-finding test exactly when $\langle \widetilde{g}, \widetilde{h}^\star \rangle$ passes the three-arc test.

Lemma 8. *Let $\widetilde{h}$ be a graph, S an SCC of $\widetilde{h}$, and $q \in Q$ a node with a path to S in $\widetilde{h}$. For all $i > 0$, there is a path from q to S in $\widetilde{h}^i$.*

Proof: Say that the path from q to S is of length n. Because S is an SCC, for every $m > n$ there is also a path from q to S of length m: we can follow arcs between elements of S arbitrarily. Take $x \in \mathbb{N}$ such that $xi > n$. There is a path of length xi from q to S in $\widetilde{h}$. Let $p_0p_1p_2...p_{xi}$ be this path, so that $p_0 = q$ and $p_{xi} \in S$. By the definition of composition, there is an arc in $\widetilde{h}^i$ from p_0 to p_i. Furthermore, for every $0 \leq k \leq xi$ there is an arc in $\widetilde{h}^i$ from p_k to p_{k+i}. Thus, $p_0p_ip_{2i}...p_{(x-1)i}p_{xi}$ forms a path of length x from q to S in $\widetilde{h}^i$. □

Lemma 9. *Let $\widetilde{h}$ be a graph and S a 1-labeled SCC of $\widetilde{h}$:*
(1) S forms a clique in $\widetilde{h}^\star$
(2) All arcs in $\widetilde{h}^\star$ between elements of S are 1-labeled
(3) For all $q \in Q$, $\widetilde{h}$ has a path from q to S iff $\widetilde{h}^\star$ has an arc from q to every element of S

Proof:
(1) By Lemma 8, for every $q \in S$ and $i \in \mathbb{N}$, there is a path from q to S in $\widetilde{h}^i$. Therefore, for every $i \in \mathbb{N}$, S is a SCC of $\widetilde{h}^i$. In specific, S is a SCC of $\widetilde{h}^\star$. Since $\widetilde{h}^\star$'s edge relation is transitive, this implies S is a clique.
(2) Since S is a clique in $\widetilde{h}^\star$, and since $\widetilde{h}^\star;\widetilde{h}^\star = \widetilde{h}^\star$, it is sufficient to prove there is at least one 1-labeled arc in $\widetilde{h}^\star$ between elements of S. Let n be the smallest number so that $\widetilde{h}^n = \widetilde{h}^\star$ As S is 1-labeled, there are two states $q, r \in S$ such that $\langle q, 1, r\rangle \in \widetilde{h}$. Consider $\widetilde{h}; \widetilde{h}^{n-1}$. By Lemma 8, there is a path from r to S in $\widetilde{h}^{n-1}$. This path starts with an arc $\langle r, a, s\rangle \in \widetilde{h}^{n-1}$, $a \in \{0, 1\}$. By the definition of composition, there is then an arc $\langle q, 1, s\rangle$ in $\widetilde{h}^n$. As there is a path from r to s and a path from s to S, s must be in S. This is precisely a 1-labeled arc in $\widetilde{h}^\star$ between two elements of S.
(3) In one direction, assume there is a path from q to S. By Lemma 8 there is a path from q to S in $\widetilde{h}^\star$. As noted above, the edge relation of every idempotent graph is transitive. As $\widetilde{h}^\star$'s edge relation is transitive, this path implies the existence of an arc from q to every element of S.
In the other direction, assume $\widetilde{h}^n = \widetilde{h}^\star$ has an arc from q to some $r \in S$. By the definition of composition, such an arc implies the existence of a path in $\widetilde{h}$ of length n from q to r.
□

Lemma 10. *Given a pair $\langle\widetilde{g}, \widetilde{h}\rangle$ of graphs in $\widetilde{Q}^f_{\mathcal{B}}$, it holds that $\langle\widetilde{g}, \widetilde{h}\rangle$ passes the lasso-finding test iff $\langle\widetilde{g}, \widetilde{h}^\star\rangle$ passes the three-arc test*

Proof: In one direction, assume that $\langle\widetilde{g}, \widetilde{h}\rangle$ passes the lasso-finding test. This holds when there exists $q \in Q^{in},\ r \in Q,\ a \in \{0, 1\}$ and $S \subseteq Q$ such that the arc $\langle q, a, r\rangle \in \widetilde{g}$, there is a path from r to S in $\widetilde{h}$, and S is a 1-labeled SCC of $\widetilde{h}$. Take any $s \in S$.

By part (3) of Lemma 9, the presence of a path from r to S in $\widetilde{h}$ implies there is an arc $\langle r, a', s\rangle \in \widetilde{h}^\star$, $a \in \{0,1\}$. Further, by parts (1) and (2) of Lemma 9, the arc $\langle s, 1, s\rangle \in \widetilde{h}^\star$. Thus the three-arc test for $\langle \widetilde{g}, \widetilde{h}^\star\rangle$ is satisfied by $\langle q, a, r\rangle \in \widetilde{g}$, $\langle r, a', s\rangle \in \widetilde{h}^\star$, and $\langle s, 1, s\rangle \in \widetilde{h}^\star$.

In the other direction, take that $\langle \widetilde{g}, \widetilde{h}^\star\rangle$ passes the three-arc test. This implies there exists $q \in Q^{in}$, $r, s \in Q$ and $a, a' \in \{0,1\}$ such that $\langle q, a, r\rangle \in \widetilde{g}$, $\langle r, a', s\rangle \in \widetilde{h}^\star$, and $\langle s, 1, s\rangle \in \widetilde{h}^\star$. By part (3) of Lemma 9, the arc $\langle r, a', s\rangle \in \widetilde{h}^\star$ implies a path from r to s in $\widetilde{h}$. Similarly, the arc $\langle s, 1, s\rangle \in \widetilde{h}^\star$ implies a path from s to itself in $\widetilde{h}$. Thus, s is part of an SCC S. Recall that $\widetilde{h}^\star$ is the composition of n copies of $\widetilde{h}$, for some $n > 0$. Since $\langle s, 1, s\rangle \in \widetilde{h}^\star$, by the definition of composition there is a path of length n from s to itself in $\widetilde{h}$ that contains a 1-labeled arc. Thus, S is a 1-labeled SCC of $\widetilde{h}$. The lasso-finding test for $\langle \widetilde{g}, \widetilde{h}\rangle$ is therefore satisfied by the arc $\langle q, a, r\rangle \in \widetilde{g}$, the path from r to $s \in S$ in $\widetilde{h}$, and S, a 1-labeled SCC of $\widetilde{h}$. □

Lemma 11. *A set $\widetilde{Q}^f_{\mathcal{B}}$ of graphs contains a pair $\langle \widetilde{g}, \widetilde{h}\rangle$ of graphs where $\widetilde{h}; \widetilde{h} = \widetilde{h}$ that fails the three-arc test if and only if $\widetilde{Q}^f_{\mathcal{B}}$ contains a pair $\langle \widetilde{g}', \widetilde{h}'\rangle$ of graphs that fails the lasso-finding test.*

Proof: In one direction, assume $\widetilde{Q}^f_{\mathcal{B}}$ contains a pair $\langle \widetilde{g}, \widetilde{h}\rangle$ of graphs where $\widetilde{h}; \widetilde{h} = \widetilde{h}$ that fails the three-arc test. If $\langle \widetilde{g}, \widetilde{h}\rangle$ passes the lasso-finding test, then by Lemma 10 $\langle \widetilde{g}, \widetilde{h}^\star\rangle$ passes the three-arc test. However, as $\widetilde{h}; \widetilde{h} = \widetilde{h}$, it holds that $\widetilde{h}^\star = \widetilde{h}$, and this would imply $\langle \widetilde{g}, \widetilde{h}\rangle$ passes the three-arc test: a contradiction.

In the other direction, assume $\widetilde{Q}^f_{\mathcal{B}}$ contains a pair $\langle \widetilde{g}, \widetilde{h}\rangle$ of graphs that fails the lasso-finding test. By Lemma 10, $\langle \widetilde{g}, \widetilde{h}^\star\rangle$ fails the three-arc test, and by definition $\widetilde{h}^\star; \widetilde{h}^\star = \widetilde{h}^\star$. As $\widetilde{Q}^f_{\mathcal{B}}$ is closed under composition, $\widetilde{h}^\star \in \widetilde{Q}^f_{\mathcal{B}}$. □

Lemma 4 follows directly from Lemmas 6 and 11.

B Proof of Theorem 1

We have now shown that it is sufficient to search $\widetilde{Q}^f_{\mathcal{B}}$ for pairs of graphs that fail the lasso finding test. In the final step of proving Algorithm 2 correct, we show that it is safe to restrict our search to graphs in $\widetilde{Q}^{\preceq}_{\mathcal{B}}$. i.e. that when using the lasso finding test $\widetilde{Q}^f_{\mathcal{B}}$ contains a counterexample if and only if $\widetilde{Q}^{\preceq}_{\mathcal{B}}$ contains a counterexample. We do so by showing that if $\widetilde{Q}^f_{\mathcal{B}}$ contains a pair of graphs that fails the lasso-finding test, then $\widetilde{Q}^{\preceq}_{\mathcal{B}}$ contains a pair of graphs approximating the pair in $\widetilde{Q}^f_{\mathcal{B}}$. This pair of graphs also fails the lasso-finding test. Recall that we say $\widetilde{g} \preceq \widetilde{h}$ when, for every arc $\langle q, a, r\rangle \in \widetilde{g}$, there is an arc $\langle q, a', r\rangle \in \widetilde{h}$, where if $a = 1$ then $a' = 1$. We first state some basic properties of approximation.

Lemma 12. *Let $\widetilde{g}, \widetilde{h}$ be two graphs where $\widetilde{g} \preceq \widetilde{h}$:*
(1) For $q, r \in Q$, if $\widetilde{g}$ has a path from q to r then $\widetilde{h}$ has a path from q to r
(2) If S is a 1-labeled SCC in $\widetilde{g}$, then there is 1-labeled SCC T of $\widetilde{h}$ where $S \subseteq T$

Proof:
(1) Every arc in $\widetilde{g}$ has a corresponding arc in $\widetilde{h}$, so a path in $\widetilde{g}$ remains a path in $\widetilde{h}$.
(2) Let S be a 1-labeled SCC in $\widetilde{g}$. By part (1) above, all nodes in S are in the same SCC T of $\widetilde{g}$. Furthermore, there must be $q, r \in S$ with an arc $\langle q, 1, r\rangle \in \widetilde{g}$. This arc must also appear in $\widetilde{h}$. Therefore T is a 1-labeled SCC of $\widetilde{h}$.

□

Lemma 13. *Let $\widetilde{g}, \widetilde{h}, \widetilde{k}, \widetilde{l}$ be four graphs. If $\widetilde{g} \preceq \widetilde{h}$ and $\widetilde{k} \preceq \widetilde{l}$, then $\widetilde{g}; \widetilde{k} \preceq \widetilde{h}; \widetilde{l}$.*

Proof: Take an arc $\langle q, a, s\rangle \in \widetilde{g}; \widetilde{k}$. First assume a is 0. This arc exists because there is state r, an arc $\langle q, 0, r\rangle \in \widetilde{g}$, and an arc $\langle r, 0, s\rangle \in \widetilde{k}$. As $\widetilde{g} \preceq \widetilde{h}$, there is an arc $\langle q, a_1, r\rangle \in \widetilde{h}$. As $\widetilde{k} \preceq \widetilde{l}$, there is an arc $\langle r, a_2, s\rangle$. Therefore the arc $\langle q, a''', s\rangle \in \widetilde{h}; \widetilde{l}$, $a''' \geq \max(a', a'')$.

If a is 1, there is state r, an arc $\langle q, a_1, r\rangle \in \widetilde{g}$ and an arc $\langle r, a_2, s\rangle \in \widetilde{k}$ where $a_1 = 1$ or $a_2 = 1$. As $\widetilde{g} \preceq \widetilde{h}$, there is an arc $\langle q, a_1', r\rangle \in \widetilde{h}$. As $\widetilde{k} \preceq \widetilde{l}$, there is an arc $\langle r, a_2', s\rangle \in \widetilde{l}$. If $a_1 = 1$, then a_1' must be 1 and $\langle q, 1, s\rangle \in \widetilde{h}; \widetilde{l}$. If $a_2 = 1$, then $a_2' = 1$ and the arc $\langle q, 1, s\rangle \in \widetilde{h}; \widetilde{l}$. We can conclude that $\widetilde{g}; \widetilde{k} \preceq \widetilde{h}; \widetilde{l}$. □

Recall that a set, $\widetilde{Q}_{\mathcal{B}}$, of graphs is *$\preceq$-closed under composition* when, for every two graphs, $\widetilde{g}, \widetilde{h} \in \widetilde{Q}_{\mathcal{B}}$, there is a graph $\widetilde{k} \in \widetilde{Q}_{\mathcal{B}}$ so that $\widetilde{k} \preceq \widetilde{g}; \widetilde{h}$. Given a set $\widetilde{Q}^1_{\mathcal{B}}$ of graphs we show that it is sufficient to test a $\preceq$-closed subset of $\widetilde{Q}^f_{\mathcal{B}}$, its closure under composition. Let $\widetilde{Q}^1_{\mathcal{B}}$ be a set of graphs, $\widetilde{Q}^f_{\mathcal{B}}$ the closure of $\widetilde{Q}^1_{\mathcal{B}}$ under composition, and $\widetilde{Q}^{\preceq}_{\mathcal{B}}$ a set computed during Algorithm 2.

Lemma 14.
(1) $\widetilde{Q}^{\preceq}_{\mathcal{B}}$ is $\preceq$-closed
(2) $\widetilde{Q}^{\preceq}_{\mathcal{B}} \subseteq \widetilde{Q}^f_{\mathcal{B}}$.
(3) For every $\widetilde{g}_\sigma \in \widetilde{Q}^1_{\mathcal{B}}$, at every point in the computation of Algorithm 2, there is a $\widetilde{h} \in \widetilde{Q}^{\preceq}_{\mathcal{B}} \cup \widetilde{W}$ such that $\widetilde{h} \preceq \widetilde{g}_\sigma$.

Proof:
(1) Note that the composition of two elements is only omitted from $\widetilde{Q}^{\preceq}_{\mathcal{B}}$ on lines (8) or (10), when it is approximated by an element already in $\widetilde{Q}^{\preceq}_{\mathcal{B}}$. Thus $\widetilde{Q}^{\preceq}_{\mathcal{B}}$ is indeed $\preceq$-closed.
(2) As every element in $\widetilde{Q}^{\preceq}_{\mathcal{B}}$ is the composition of a finite number of elements of $\widetilde{Q}^1_{\mathcal{B}}$, $\widetilde{Q}^{\preceq}_{\mathcal{B}}$ is indeed a subset of $\widetilde{Q}^f_{\mathcal{B}}$,
(3) At the beginning of the computation, $\widetilde{g}_\sigma$ is in the worklist $\widetilde{W}$. Since the worklist is empty at the end of the computation, at some point we know $\widetilde{g}_\sigma$ was removed from the worklist. In order for $\widetilde{g}_\sigma$ to be discarded on line (8) or removed on line (10) of Algorithm 2, there must be a $\widetilde{h}_0$ included in $\widetilde{Q}^{\preceq}_{\mathcal{B}}$ such that $\widetilde{h}_0 \prec \widetilde{g}_\sigma$. Similarly, $\widetilde{h}_0$ can only be removed if a graph $\widetilde{h}_1$ that approximates it is included, and so on. This allows us to conclude that at every point in the computation of Algorithm 2 there will be a graph $\widetilde{h}_i \in \widetilde{Q}^{\preceq}_{\mathcal{B}}$ such that:

$$\widetilde{h}_i \preceq \widetilde{h}_{i-1} \preceq \cdots \preceq \widetilde{h}_0 \preceq \widetilde{g}$$

As approximation is transitive, it holds that $\widetilde{h}_i \preceq \widetilde{g}$.

$\square$

Lemma 15. *For every $\widetilde{g} \in \widetilde{Q}^f_{\mathcal{B}}$, there is $\widetilde{g}' \in \widetilde{Q}^{\preceq}_{\mathcal{B}}$ such that $\widetilde{g}' \preceq \widetilde{g}$.*

Proof: The graph $\widetilde{g}$ is the finite composition $\widetilde{g}_0; \widetilde{g}_1; ...; \widetilde{g}_{n-1}$ of n elements of $\widetilde{Q}^1_{\mathcal{B}}$. Let $\widetilde{g}_{0...i}$ be the sequence of compositions $\widetilde{g}_0; ...; \widetilde{g}_i$. We prove by induction that, for each i, there is a graph $\widetilde{h}_i \in \widetilde{Q}^{\preceq}_{\mathcal{B}}$ so that $\widetilde{h}_i \preceq \widetilde{g}_{0...i}$. Since $\widetilde{g} = \widetilde{g}_{1...(n-1)}$, this implies $\widetilde{h}_{n-1} \preceq \widetilde{g}$. The base case follows immediately from the third clause of Lemma 14 and the fact that the worklist is empty at the end of the computation.

Inductively, assume there is a graph $\widetilde{h}_i \in \widetilde{Q}^{\preceq}_{\mathcal{B}}$ so that $\widetilde{h}_i \preceq \widetilde{g}_{0...i}$. First, as $\widetilde{g}_{i+1} \in \widetilde{Q}^1_{\mathcal{B}}$, the third clause of Lemma 14 implies a $\widetilde{g}'_{i+1} \in \widetilde{Q}^{\preceq}_{\mathcal{B}}$ exists so that $\widetilde{g}'_{i+1} \preceq \widetilde{g}_{i+1}$. Second, note that by Lemma 13 it holds that $\widetilde{h}_i; \widetilde{g}'_{i+1} \preceq \widetilde{g}_{0...i}; \widetilde{g}_{i+1}$. Finally, as $\widetilde{Q}^{\preceq}_{\mathcal{B}}$ is $\preceq$-closed under composition, either $\widetilde{h}_i; \widetilde{g}'_{i+1}$ is in $\widetilde{Q}^{\preceq}_{\mathcal{B}}$, or it is discarded/removed on line (8)/(10) when it is approximated by some $\widetilde{k} \in \widetilde{Q}^{\preceq}_{\mathcal{B}}$. In the former case, take $\widetilde{h}_{i+1}$ to be $\widetilde{h}_i; \widetilde{g}'_{i+1}$. In the later case, recall that approximation is transitive and take $\widetilde{h}_{i+1}$ to be $\widetilde{k}$.

$\square$

Lemma 16. *$\widetilde{Q}^f_{\mathcal{B}}$ contains a pair of graphs that fails the lasso-finding test iff $\widetilde{Q}^{\preceq}_{\mathcal{B}}$ contains a pair of graphs that fails the lasso-finding test.*

Proof: In one direction, assume $\widetilde{Q}^{\preceq}_{\mathcal{B}}$ contains a pair $\langle \widetilde{g}, \widetilde{h} \rangle$ of graphs that fails the lasso-finding test. By Lemma 14, $\widetilde{Q}^{\preceq}_{\mathcal{B}} \subseteq \widetilde{Q}^f_{\mathcal{B}}$. Therefore $\langle \widetilde{g}, \widetilde{h} \rangle$ is such a pair of graphs in $\widetilde{Q}^f_{\mathcal{B}}$.

In the other direction, assume $\widetilde{Q}^f_{\mathcal{B}}$ contains a pair $\langle \widetilde{g}, \widetilde{h} \rangle$ of graphs that fails the lasso-finding test. By Lemma 15 there must be a $\widetilde{g}', \widetilde{h}' \in \widetilde{Q}^{\preceq}_{\mathcal{B}}$ such that $\widetilde{g}' \preceq \widetilde{g}$ and $\widetilde{h}' \preceq \widetilde{h}$. We show by way of contradiction that $\langle \widetilde{g}', \widetilde{h}' \rangle$ must fail the lasso-finding test. Presume that $\langle \widetilde{g}', \widetilde{h}' \rangle$ passes the lasso-finding test. This implies the existence of $q \in Q^{in}$, $r \in Q$, $a, b \in \{0, 1\}$ and $S \subseteq Q$ such that $\langle q, a, r \rangle \in \widetilde{g}'$, there is a path from r to S in $\widetilde{h}'$, and S is a 1-labeled SCC of $\widetilde{h}'$. As $\widetilde{g}' \preceq \widetilde{g}$, there must be an arc $\langle q, a', r \rangle \in \widetilde{g}$. Further, by the second half of Lemma 12 there must be a $T \supseteq S$ that is a 1-labeled SCC of $\widetilde{h}$. By the first half of Lemma 12, there must be a path from r to S, and thus to T, in $\widetilde{h}$. However, as $\langle \widetilde{g}, \widetilde{h} \rangle$ fails the lasso-finding test, there can be no such q, r, and T. $\square$

Theorem 1 is now proven by the composition of Lemmas 4 and 16.

C Reference Graphs

Figure 6 displays the median running time of RAMSEY and RANK for $r = 1.5$, $f = 0.26$ on a logarithmic scale, up to size $n = 50$. This is the same terrain point as Figure 4. We observe that the slope of the two lines is roughly the same, indicating that below $n = 50$, RAMSEY scales only a constant factor better than RANK.

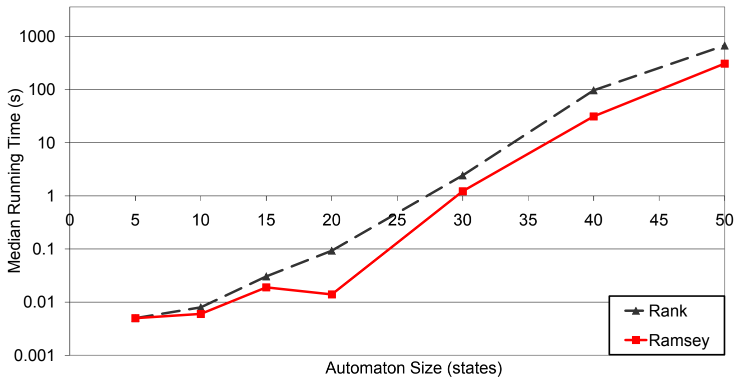

Fig. 6: Median running times when $r = 1.5$ and $f = 0.26$, log scale

Figure 7 displays the median running time for $r = 2$, $f = 0.05$ on a logarithmic scale. The final point for RANK has been flattened, as more than 50% of the cases timed out. This is the same terrain point as Figure 5. We note that both curves are less than linear, indicating a sub-exponential growth in running time.

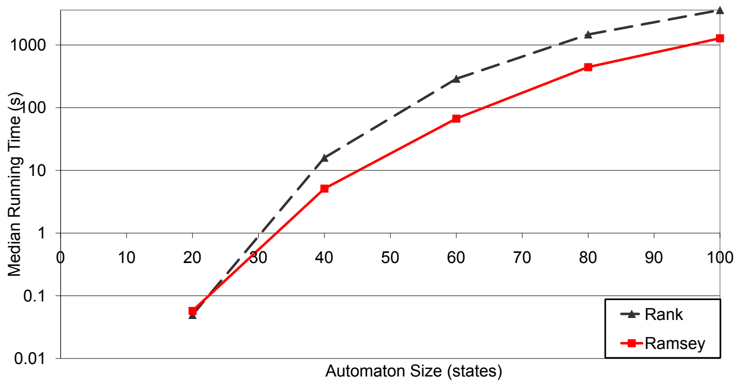

Fig. 7: Median running times when $r = 2$ and $f = 0.05$, log scale