Random Models for Evaluating Efficient Büchi Universality Checking

Corey Fisher[1](✉), Seth Fogarty[2], and Moshe Vardi[1]

[1] Rice University, Houston, USA
corey.s.fisher@gmail.com
[2] Trinity University, San Antonio, USA

Abstract. Automata-theoretic formal verification approaches the problem of guaranteeing that a program conforms to its specification by reducing conformance to language containment. We can prove conformance by representing both programs and specifications as automata and proving that the specification contains the program. This connection to the theory of automata on infinite words motivated an extensive research program into the algorithmic theory of automata on infinite words, with a focus on algorithms that perform well in practice. The focus on practical performance is important because of the large gap between worst-case complexity and practice for many automata-theoretic algorithms. Unfortunately, there are few benchmark instances of automata in industrial verification. To overcome this challenge, Tabakov and Vardi proposed a model for generating random automata as test cases.

The Tabakov-Vardi (T-V) model, however, is just one random model, based on a specific, rather simple model of random graphs. Other models of random graphs have been studied over the years. While the T-V model has the advantage of simplicity, it is not clear that performance analysis conducted on this model is robust, and an analogous analysis over other random models might yield different conclusions. To address this problem, we introduce three novel models of random automata, yielding automata that are richer in structure than the automata generated by the T-V model. By generating large corpora of random automata and using them to evaluate the performance of universality-checking algorithms, we show that the T-V model is a robust random model for evaluating performance of universality-checking algorithms.

1 Introduction

Automata-theoretic formal verification is an approach to the problem of guaranteeing that a program (in software or hardware) conforms to its specification, in which conformance is reduced to the problem to language containment. By representing both programs and specifications as automata and proving that the specification contains the program, we can prove conformance [19]. This connection to automata theory, and, in particular, to the theory of automata on infinite

We recommend viewing the plots in this paper online. For a longer technical report, see http://www.cs.rice.edu/~vardi.

S. Ghosh and S. Prasad (Eds.): ICLA 2017, LNCS 10119, pp. 91–105, 2017.
DOI: 10.1007/978-3-662-54069-5_8

words [21], motivated an extensive research program into the algorithmic theory of automata on infinite words, cf. [20], and the focus of this program is often on algorithms that perform well in practice, cf. [12].

We focus here on the Büchi *universality-checking* problem, which is a simplified case of containment checking, the canonical verification problem [19]. An automaton A is *universal* if it accepts all input words; equivalently A is universal if its complement $\overline{A}$ is *empty*, that is it accepts no input words. A simplistic way to check universality of A is to check emptiness of $\overline{A}$, which can be reduced to reachability analysis of $\overline{A}$'s state-transition graphs. Such an approach would have to deal with the blow-up of Büchi complementation, so extant algorithms for universality use a variety of heuristics to check emptiness of $\overline{A}$ without constructing it in full, cf. [4].

The focus on performance in practice is important because of the large gap between worst-case complexity and performance in practice for many automata-theoretic algorithms. For example, the best upper bound for the complementation of Büchi automata is $2^{O(n \log n)}$ [15] (realized, for example, by the *rank-based* construction in [9]), which matches the known lower bound [13]. This bound is significantly lower that the earlier upper bound of $2^{O(n^2)}$ [16], which uses Büchi 's *Ramsey-based* construction [1]. Yet a comparison of the rank-based construction with the Ramsey-based construction on real-life instances showed that the Ramsey-based construction can be quite competitive in practice with the rank-based construction – each outperforms the other on different problem instances [5].

Nevertheless, the quest for automata-theoretic algorithms that perform well in practice is hampered by the fact that there is a shortage of benchmark instances of automata that arise in industrial verification (see discussion below). To overcome this challenge, Tabakov and Vardi proposed a model for generating random automata on which different algorithms can be evaluated and compared [17,18]. The model has three parameters: (1) the size (number of states) of the automaton, (2) the *density* of transitions (ratio of transitions to states), and (3) The *density* of accepting states (ratio of accepting to total number of states). Subject to these parameters, the model generates automata randomly. The Tabakov-Vardi (T-V, for short) model is attractive for two reasons [17]: First, the model gives rise to an interesting *universality terrain*, which describes the relationship between the probability of automaton universality (which means that all input words are accepted) and the density parameters. Second, the model gives rise to an interesting *performance terrain*, which describes the relationship between algorithmic performance and the density parameters. (We discuss these two terrains in detail in the body of the paper.) In subsequent years, this model has become the standard model for the evaluation of Büchi -complementation tools, cf. [2,4,11,14].

The T-V model, however, is just one specific random model, based on a specific, and quite simple model of random graphs [8]. As we show in this paper, several other models of random graphs have been studied over the years. While the T-V model has the advantage of simplicity, it is not a priori clear that performance analyses conducted on this model are robust, as it is entirely possible that analogous analyses over other random models would yield different conclusions. Since performance analyses over random models are used in this context as a substitute to such analyses over a benchmark suite of real-life problem instances, it is desirable at least to know whether analyses over random models yield robust conclusions.

To address this problem, we introduce three[1] novel models of structured random automata, based on existing random graph models – the *vertex-copying* model [7], the *Frank-Strauss* model [6], and the *co-accessible* model [10]. These models are based on different models that have been proposed for random graphs. While the T-V model is uniformly random, generating unstructured automata, these new models constrain randomness in some way to provide structural guarantees about the resulting automata: The vertex-copying model guarantees a power-law degree distribution, the Frank-Strauss model restricts which transitions are valid, and the co-accessible model guarantees that each state in the resulting automaton can reach an accepting state.

These structural properties help the models represent a wide variety of possible types of problem instances that might be encountered in the real world. Furthermore, these model generate problem instances that are quite unlikely to be generated by the T-V model. Our goal is to compare performance analysis on the T-V model against performance analysis on the three new models. If performance analyses on the a variety of different models all reach similar conclusions, then we can conclude that these conclusions are likely robust. If, on the other hand, performance analyses on different models reach different conclusions, then we would gain a deeper understanding of how structure affects algorithmic performance and learn that the choice of algorithm should depend on the structure of the problem instance being solved.

By generating large corpora of random automata and using them to evaluate the performance of universality-checking algorithms we first show that the new models possess the same useful properties for universality as the T-V model. We then replicate results of Fogarty and Vardi [4] for universality checking, using all four random models. We show that the finding reached in [4], concluding that the two tools compared are competitive, is robust across the four models. Finally, we compare Fogarty and Vardi's Rank tool [4], the most recent implementation of the rank-based algorithm, with a modern Ramsey-based tool, RABIT 2.3[2], and show that the Ramsey-based tool strongly outperforms the rank-based tool, again over all four models. We conclude, therefore, that the T-V model, in spite of its simplicity, is an adequate random model for evaluating performance of universality-checking algorithms.

[1] The full version of the paper, with more models, can be found in the technical report [3].

[2] http://languageinclusion.org/doku.php?id=tools.

2 Background

Automata Theory. A *Büchi automaton* is a tuple $A = (\Sigma, Q, Q_0, \delta, F)$, where Σ is a finite alphabet, Q is the finite set of states, $Q_0 \subseteq Q$ is the set of initial states, $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation, and $F \subseteq Q$ is the set of accepting states. Büchi automata take infinite words from Σ^ω as input. A *run* of a Büchi automaton on a word $w_0, w_1, ... \in \Sigma^\omega$ is any infinite sequence $q_0, q_1, ... \in Q^\omega S$ such that $q_0 \in Q_0$, and $(q_i, w_i, q_{i+1}) \in \delta$. A run is *accepting* if some accepting state $q_i \in F$ occurs infinitely often in the run. The Büchi automaton accepts a word w if there is some run of w that is accepting. The set of all words an automaton A accepts is called the *language of* A, or $L(A)$. A *complement* $\overline{A}$ of an automaton A is an automaton whose language is $\Sigma^\omega \setminus L(A)$. Finding the complement of an automaton is called *complementation.*

An automaton A is *contained* in an automaton B when $L(A) \subseteq L(B)$. In automata-theoretic verification [19], we prove that a program satisfies a specification by modeling the program as a Büchi automaton A and the specification as a Büchi automaton B, and then proving that A is contained in B. To check this containment, we check that the intersection of $L(A)$ with $L\left(\overline{B}\right)$ is empty. If it is not empty, then a word in the intersection is a trace of A that violates the specification B. In practice, efficient containment algorithms do not explicitly construct the complement $\overline{B}$, using instead various strategies for on-the-fly complementation and symbolic construction, cf. [4]. Nevertheless, because these strategies are still fundamentally based on complementation, there is a close link between the efficiency of complementation and the efficiency of containment. The two complementation constructions that have been studied in the context of containment checking are the Ramsey-based construction of [16] and the *rank-based* construction of [9]. While the rank-based construction has a better worst-case complexity, the Ramsey-based approach is quite competitive in the context of containment checking [4]. Since the hard step in containment checking is the need to construct (at least implicitly) $\overline{B}$, papers on the subject, e.g. [4,17,18], usually focus on *universality checking*, where $L(A) = \Sigma^\omega$ – that is, checking if $L(B)$ contains the set of all words.

Evaluating Automata-Theoretic Algorithms. The quest for automata-theoretic algorithms that perform well in practice is hampered by a shortage of benchmark instances of automata that arise in industrial verification. The automaton B above corresponds to a formal specification of intended design functionality. Industrial specifications are typically proprietary and not openly available. To overcome this challenge, Tabakov and Vardi (T-V) proposed a model for generating random automata on which different algorithms can be evaluated and compared [17,18]. In subsequent years, this model has become the standard model for the evaluation of automata-theoretic tools, cf. [2,4,11,14]. Specifically, the T-V model was used in [4] to show that despite the worst-case-complexity gap between the Ramsey-based and the rank-based approaches, the two approaches are co-competitive in practice – that is, they each can outperform the other in non-trivial cases, depending on the properties of the automata being checked.

The T-V model generates automata using the *uniformly random* choice of elements from a set. The T-V model takes three parameters - an integral *size* n, a positive real *transition density* r, and a real *accepting-state density* f between 0 and 1. The transition density is the average out-degree of each state in the result automaton per input symbol. The accepting-state density is the percentage of states in the result automaton that are accepting states. Formally, a (n, r, f) T-V random automaton is defined as follows. Each random automaton $A = (\Sigma, Q, Q_0, \delta, F)$ has the alphabet $\Sigma = \{1, 0\}$ and set of states $Q = \{0, \ldots, n-1\}$. The set Q_0 of initial states is $\{0\}$. For each $\sigma \in \Sigma$, the model generates a digraph (directed graph) D_σ over the nodes $\{0, \ldots, n-1\}$ with $n * r$ edges chosen uniformly at random from the set of all possible edges $(u, v) \in Q \times Q$. The transition relation δ is then defined as $\{(u, \sigma, v) \mid (u, v) \in D_\sigma\}$. The accepting states F comprise $\lfloor n * f \rfloor$ states selected uniformly at random from Q. Note that each element of D_σ is a random digraph - specifically, a Karp [8] random digraph. Thus, we say that the T-V model *lifts* the Karp model of random digraphs into automata.

The T-V model is attractive for performance evaluation for two reasons [17,18]: First, the useful properties of its *universality terrain*, which describes the relationship between the probability of automaton universality (which means that all input words are accepted) and the density parameters. When transition and accepting-state densities are low, the probability for universality is low, while at higher densities the probability steadily increases. Thus, the model provides a way to evaluate the performance of universality-checking algorithms on both universal and non-universal automata. We call a model "*interesting*" when its universality probabilities vary with the input parameters and increase from low to high probability. Second, the model gives rise to an interesting *performance terrain*, which describes the relationship between algorithmic performance and the density parameters. Specifically, at low and high densities universality checking is easier than at intermediate densities. Thus, the model provides a way to evaluate the performance of universality-checking tools on both easy and hard problems. We take these two features, universality terrain and performance terrain to be desiderata that we expect to have in other models of random automata.

3 Random Models

Our goal in this work is to compare the T-V model to other models of random automata as a framework for evaluating the performance of universality-checking algorithms. We take advantage of the fact that the Tabakov-Vardi technique of *lifting* digraphs into automata is not limited to Karp random digraphs. By substituting other random-digraph models, we can generate new models of random automata.

The Tabakov-Vardi lifting is as follows. A random automata model that lifts a random digraph model has all of the parameters of the digraph model, plus an accepting-state density parameter f. Each random automaton is a tuple $(\Sigma, Q, Q_0, \delta, F)$, with the elements defined as follows. We take the alphabet

$\Sigma = \{0, 1\}$ for all models. For each character $\sigma \in \Sigma$, create a random digraph D_σ using the digraph parameter values of the automaton model. The set Q of states of the random automaton is equivalent to the set N of D_σ's nodes, usually $N = \{0, \ldots, n-1\}$, where n is the size parameter. The initial state set $Q_0 \subseteq Q$ is a singleton set containing one state from Q, usually 0. The transition relation δ is the union of all sets $\{(q, \sigma, r) \mid (q, r) \in D_\sigma\}$ for $\sigma \in \Sigma$. Finally, the set $F \subseteq Q$ of accepting states consists of $\lfloor |N| * f \rfloor$ elements of Q chosen uniformly at random (without repetition). Not all models we study use the Tabakov-Vardi lifting; see details below.

In the rest of this section, we introduce three[3] new models based on this lifting - the *vertex-copying* model, the *Frank-Strauss* model, and the *co-accessible* model. The first two models are based on existing models of structured random digraphs which have found common use in other disciplines, and the co-accessible model guarantees a particular automaton property. While the lack of existing benchmarks makes it difficult to compare these models directly to industrial problem instances, we can use a variety of structured random models to more fully explore the problem space. If these models disagree with the Tabakov-Vardi model, then the T-V model is not rich enough to fully represent the space on its own – if they agree, then it is likely that the conclusions of the T-V model are quite robust.

We show each of the models to have a universality terrain that is somewhat similar but not identical to that of the T-V model, using experiments run on the DAVinCI cluster[4] at Rice University. To show that each model has an interesting universality terrain, we present with each model a terrain plot showing how likely the automata generated by the model are to be universal when made with certain parameters. We generated and tested 100 automata using the parameters at each point on the plot. The universality terrains show that the random models we introduce generate automata whose likelihood of being universal ranges from 0 to 1, just as in the T-V model.

Vertex-Copying Automata. The random vertex-copying model presented here is a simplification of the model defined by Kleinberg *et al.* [7]. A vertex-copying digraph starts out as an empty set of nodes, and adds edges over time. By sometimes choosing edges at random, and at other times copying edges from one node to another, it creates a heavy-tailed distribution – a "rich get richer" effect as nodes with many edges steadily gain more and more edges. This copying is intended to model hyperlinks on the Web – links are often created when someone discovers a link to a site they're interested in on another site, then adds a link to it on their own website, thus "copying" the link from one site to another. This approach may also model code reuse - when a code block is reused, then calls to functions are duplicated.

An (n, b, r) vertex-copying random digraph takes as parameters the *size* n, the *copying probability* b, and the *transition density* r. The vertices are $\{0, \ldots, n-1\}$. The model begins with no edges and adds edges (u, v) to the

[3] Other models can be found in the technical report [3].

[4] http://www.rcsg.rice.edu/sharecore/davinci/.

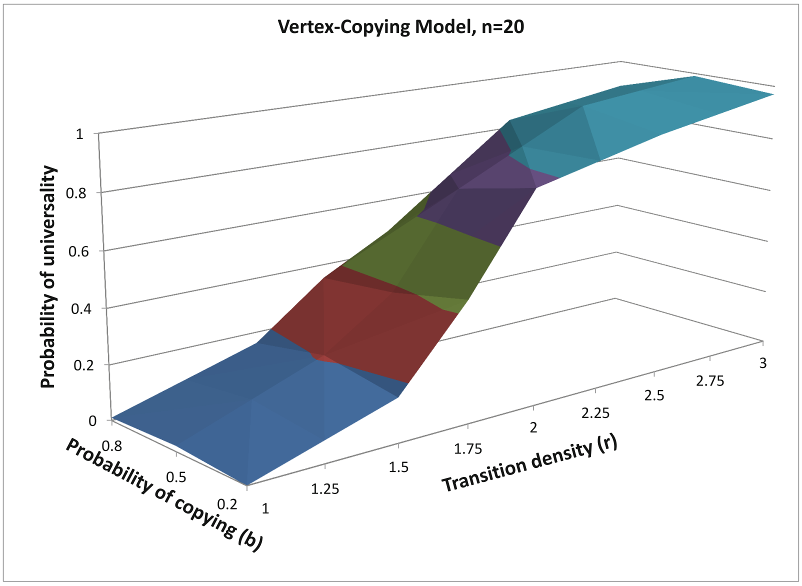

Fig. 1. A vertex-copying universality terrain for $n = 20$. The transition density r ranges from 1 to 3, and the copying probability b ranges from 0.2 to 0.8. The accepting-state density f was set to 0.3. The universality probability is comparable to that of the T-V model for most values of r. Note that increasing b does not monotonically increase universality probability – after a certain point it actually reduces it. This may be because all transitions go to a small number of states, with few transitions leaving them, increasing the likelihood of rejection.

graph one at a time until there are $\lfloor n * r \rfloor$ edges. Each time it does so, it has a probability b of copying an edge from one node to another, and a probability $1 - b$ of simply generating an edge uniformly at random. If it copies, then it chooses an edge $(u, v) \in E$ and a node $u' \in V \backslash u$ uniformly at random. It then adds (u', v) to E. If it generates the edge at random, it acts as in the T-V model. This digraph model extends to automata by directly using the standard lifting. Its universality terrain is given in Fig. 1.

Frank-Strauss Automata. The Frank-Strauss random graph model, based on an approach by Frank and Strauss[5] [6], limits the space of possible edges. Instead of the vertices being integers, vertices are unordered pairs of integers. The Frank-Strauss model permits edges only between vertices that share an element – the vertex $(0, 1)$ can connect to $(0, 3)$ and $(1, 3)$, but not to $(2, 3)$. Within this space, edges are generated uniformly at random. The Frank-Strauss model can represent systems that require some relationship between actors. For example, it can be used to represent binary relationships between individuals in a social setting. Alternatively, we may have a program such that if one module calls another,

[5] Referred to in their paper as a "Markov graph".

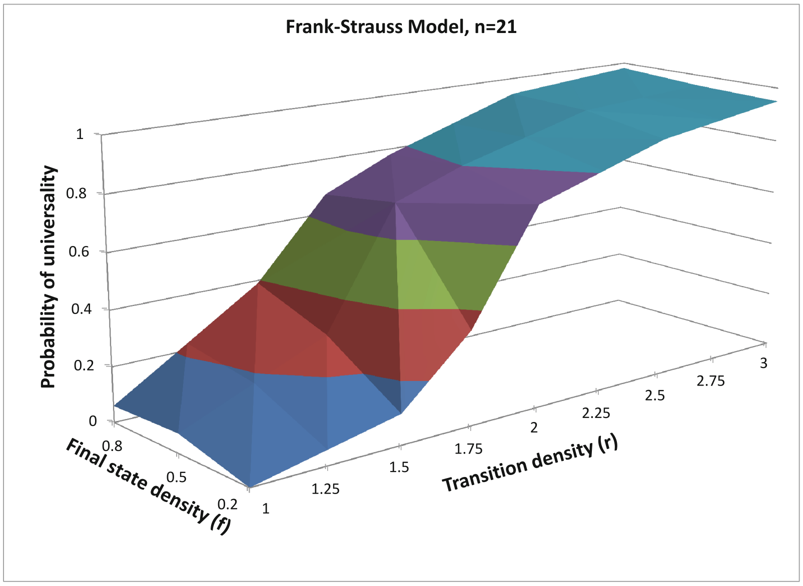

Fig. 2. A Frank-Strauss universality terrain for $l = 21$. r ranges from 1 to 3 and f ranges from 0.2 to 0.8. While the universality probably scales more quickly with r than in the T-V model, there are still a number of points where universality is neither nearly guaranteed nor always absent.

then there must be some relation between them – for example, operating on shared data.

An (l, r) Frank-Strauss random graph takes as parameters a *label size* l and a *transition density* r. The set V of vertices is the set $\{(i, j) \mid i, j \in 0, \ldots, l-1\}$ of unordered pairs of elements. Since we allow the case where $i = j$, there are $\binom{l+1}{2} = \frac{l(l+1)}{2}$ such vertices. We generate $\lfloor |V| * r \rfloor$ edges. To generate each edge, first choose a vertex (u_1, u_2) uniformly at random as the source, and then choose a vertex $(v_1, v_2) \in \{u_1, u_2\} \times \{0, \ldots, l\}$ uniformly at random as the destination. This digraph model extends to automata directly by using the standard lifting. The universality terrain is presented in Fig. 2.

Co-accessible Automata. The co-accessible model of random automata is so named because it guarantees that the resulting automata are co-accessible, where an automaton is *co-accessible* if all states $q \in Q$ are co-accessible, that is, can reach an accepting state. Because this property is meaningful only for automata, the co-accessible model cannot be based on lifting a model of random digraphs. It is loosely based on Leslie's generation of connected automata [10]. Automata possessing this property correspond to useful program properties – for example, a co-accessible automaton may specify that the program can recover and perform its intended function from every state.

The co-accessible model takes as parameters a *size* n, a *transition density* r, and an *accepting state density* f. The co-accessible model does not define the transition relation based on an underlying digraph. Instead, we start with a set $Q = \{0, \ldots, n-1\}$ of states and initial and accepting state sets Q_0 and F as in the T-V model. The transition relation δ is initially empty.

To fill in δ, we construct a random spanning inverted forest over Q. This is a set of trees over the automaton which contains every state, each rooted at an accepting state, and where edges go from children to parents instead of parents to children. A forest can be found as follows: make a set of co-accessible states $C = F$ and states that are not yet co-accessible $U = Q \backslash F$, then select some $u \in U$, $c \in C$ and $\sigma \in \Sigma$ uniformly at random. Add (u, σ, c) to δ, then remove u from U and add it to C, repeating until U is empty.

Once the spanning forest has been constructed, the model must fill in the rest of the transition relation. It then ensures that each character $\sigma \in \Sigma$ is associated with exactly $\lfloor n * r \rfloor$ edges. If some σ_0 has more than $\lfloor n * r \rfloor$ transitions, replace random transitions (u, σ_0, v) with (u, σ_1, v) for $\sigma_0 \neq \sigma_1$ and $\sigma_1 \in \Sigma$. Then generate new edges uniformly at random, as in the T-V model, for each character with fewer than $\lfloor n * r \rfloor$ transitions. We assume $r \geq 1$. The universality terrain is given in Fig. 3.

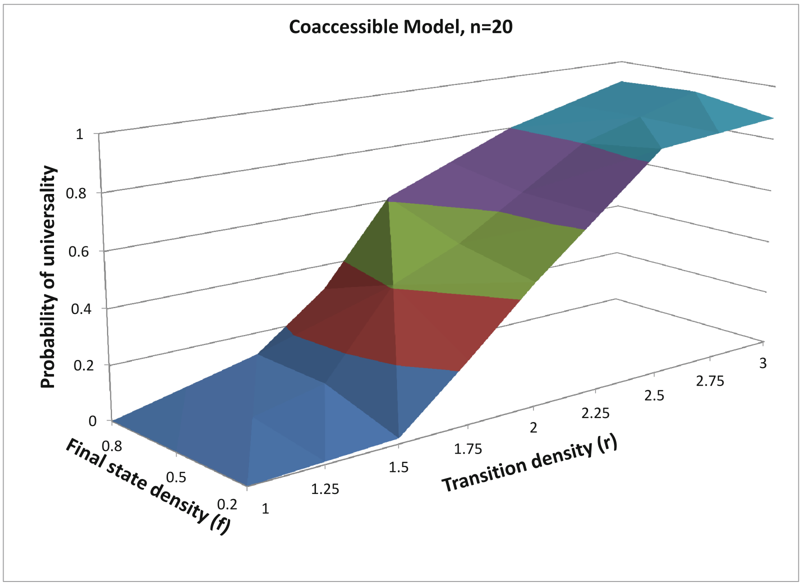

Fig. 3. A co-accessible universality terrain for $n = 20$. The transition density r ranges from 1 to 3, and f ranges from 0.2 to 0.8. Notice that the slope is much shallower than in previous models. This gives us an extremely wide range of useful configurations for testing.

4 Experiments

Methodology. Having defined three new random models and, via universality testing, proven them to be interesting for performance evaluation, we then used these models to run timing experiments for three universality checkers. We first compared the Rank and Ramsey tools[6] from [4]. To acquire a more recent picture of the comparison between algorithms, we also compared these tools with the RABIT 2.3 tool[7], a more recent Ramsey-based containment checker. As in the previous section, experiments were run on the DAVinCI cluster at Rice University, which consists of many Westmere nodes with 2.83 GHz processors and 48 GB of memory per node. We limited each job to 30 GB of memory and one hour of time. Jobs that did not finish were marked as timeouts.

We ran two types of experiments: terrain experiments and scaling experiments. In terrain experiments, the size of the automata is held constant, and two other parameters are changed to see the effects on running time. In scaling experiments, all parameters are held constant except those affecting the size of the automaton, and we steadily increase the size to see how the implementations respond to larger problems. We conduct scaling experiments with parameters that are particularly difficult for at least one tool to handle, as determined by the terrain experiments, to test practical worst-case performance. We generated 100 automata using each combination of parameter values in both kinds of experiments, and report median running time.

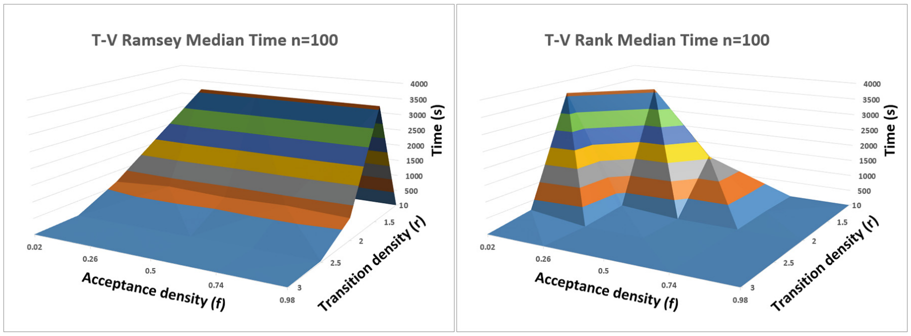

Fig. 4. For terrain experiments on the Tabakov-Vardi model, we tested parameter values of $n = 100$, or $l = 14$, $r \in \{1, 1.5, 2, 2.5, 3\}$, and $f \in \{0.02, 0.26, 0.5, 0.74, 0.98\}$. These graphs show results for the Rank and Ramsey tools. Note that Rank and Ramsey are not directly comparable - Ramsey tends to be slower at points where $r = 1.5$ and $r = 2$, while Rank tends to be slower at $f = 0.02$ and $f = 0.26$. This agrees with previous results [4] using the Tabakov-Vardi model.

[6] https://www.cs.rice.edu/CS/Verification/Software/software.html.
[7] http://www.languageinclusion.org/doku.php?id=tools.

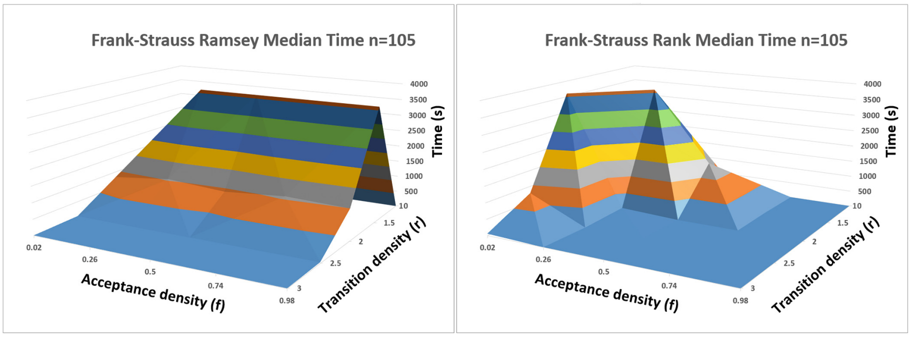

Fig. 5. For terrain experiments on the Frank-Strauss model, we tested parameter values of $n = 105$, or $l = 14$, $r \in \{1, 1.5, 2, 2.5, 3\}$, and $f \in \{0.02, 0.26, 0.5, 0.74, 0.98\}$. These graphs show results for the Rank and Ramsey tools. Again, the Rank model tends to perform the slowest at low f and low r, while Ramsey is slowest at $r = 2$. This agrees with our results on the Tabakov-Vardi model, as do the terrains of other models found in the technical report [3].

Results. We find both that choice of model does not seriously impact tool comparisons, and that RABIT noticeably outperforms Rank and Ramsey.

In both terrain (Figs. 4, 5 and 6) and scaling (Fig. 7) experiments, we find that the relative efficiency of tools is very similar across models. All models show that, as in the Tabakov-Vardi model in [4], the Rank and Ramsey are not directly comparable – which parameters are used to generate an automaton determine which tool solves it most efficiently, as seen in the terrain experiments in Fig. 5. Since all models agree with T-V here, it is reasonable to use the T-V model to compare tools. Nevertheless, while models agree on the comparison between tools, they do not have the same running time. For example, in Fig. 7, we see on a log scale that there is a factor 10 difference between the running time of Ramsey on the Tabakov-Vardi and co-accessible models. Thus, the T-V model should be relied on for relative comparisons, but not for predicting runtimes.

Since there was little difference in comparison between models, Rank and Ramsey compare similarly to their results in [4]. Yet, when we compare Rank to RABIT, we saw a massive speedup at all difficult points – sometimes thousands of times faster. At $n = 100$, the terrain was flat, with most cases terminating in just over a tenth of a second. Therefore, the improved modern Ramsey tools are more suited for practical use than Rank-based ones. However, as seen in Fig. 6, random models can still provide interesting performance terrain on the more efficient tools by scaling up the size of the problems.

There is one noticeable difference between algorithms not shown – both Ramsey-based algorithms used much more memory than Rank did. When provided with 5 GB of memory, the Rank tool performed acceptably, but Ramsey and RABIT crashed regularly. 30 GB of memory provided was necessary to avoid crashes due to running out of memory.

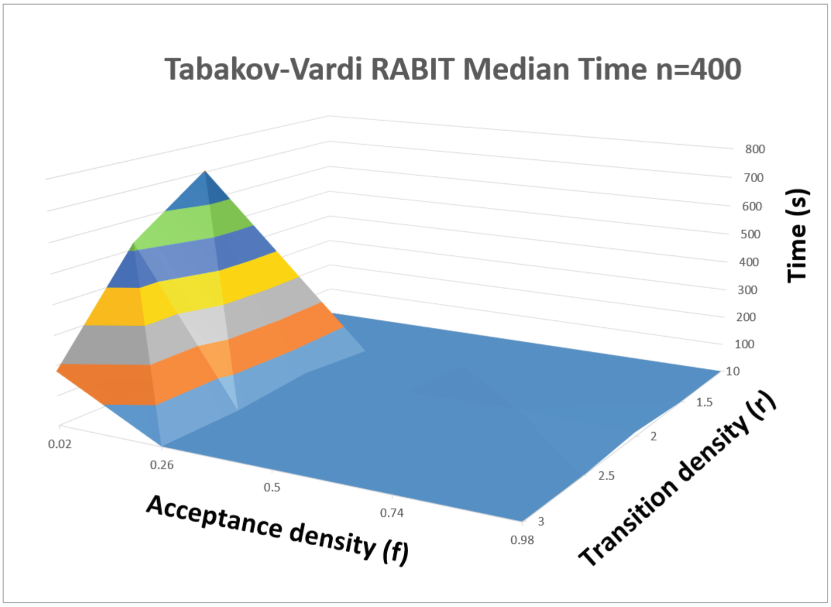

Fig. 6. For all terrain experiments at $n = 100$ for RABIT, we found that the terrain was entirely flat - very few problems took more than one second to terminate. Therefore we show results for RABIT on $n = 400$, instead, with parameter values $r \in \{1, 1.5, 2, 2.5, 3\}$, and $f \in \{0.02, 0.26, 0.5, 0.74, 0.98\}$. Note that the maximum Y-axis value is only 800 seconds, because at no point was the median result a timeout. RABIT has the most difficulty at high transition density and extremely low acceptance densities, with orders of magnitude slower performance on $f = 0.02$. While it does not appear on this graph, we also find that RABIT takes about two orders of magnitude more time at $r = 2.0$ and high f than other areas, and one order of magnitude less than the extremely difficult areas. Also, we find that at $r = 1.5$, we consistently had a small (5 %) chance of timeouts at all values of n tested with few to no timeouts elsewhere, though the median time taken was no higher.

5 Concluding Remarks

While formal verification provides important software tools, it has been unclear whether these tools are efficient enough to be used in practice. Thus, the T-V model is a powerful tool for automata-theoretic formal verification, allowing us to test the efficiency of algorithms for determining conformance to a specification. Due to concerns about whether the model accurately reflected real-world performance, we tested other models to see if the structure of a problem would influence the results; we found that it did not. Future work in the area can proceed to test algorithms and tools on the T-V model, more confident that it is robust and that its results are widely applicable.

This work gives reason to believe that the Tabakov-Vardi model is a robust model with results that are likely to be close to the real-world. Complementation,

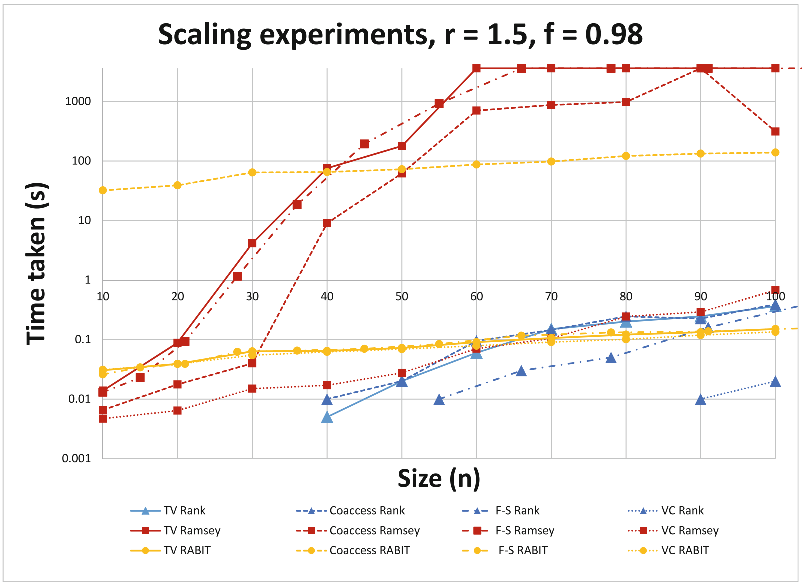

Fig. 7. For this set of scaling experiments, we set $r = 1.5$ and $f = 0.98$, and scale n from 10 to 100. In the Frank-Strauss model, l scales from 4 to 14. This point was chosen for scaling because it is particularly difficult for Ramsey. On this log-scale plot, different tools (indicated by shared color and marker shape) tend to have similar slopes regardless of model (indicated by shared line style). Notably, an obvious exponential gap exists between other models and Ramsey at these parameters for every model except the trivally-easy vertex-copying model. Since f is high, this is an easy point for Rank. The relationship between tools found by T-V is also reflected in the other random models shown here.

and thus containment checking, should be practical on real-world problems. We also discovered an improvement of many orders of magnitude in modern containment checkers using a Ramsey-based approach. RABIT outperformed both older Ramsey and rank-based tools significantly, and can scale up much higher. Since little work has been done on rank-based solvers since 2010, current heuristics-driven Ramsey-based approaches are the best available options for containment checking for Büchi automata.

Acknowledgements. Work supported in part by NSF grants CCF-1319459 and IIS-1527668, by NSF Expeditions in Computing project "ExCAPE: Expeditions in Computer Augmented Program Engineering", as well as the Data Analysis and Visualization Cyberinfrastructure funded by NSF grant OCI-0959097 and Rice University.

References

1. Büchi, J.R.: Turing-machines and the Entscheidungsproblem. Math. Ann. **148**(3), 201–213 (1962)
2. Doyen, L., Raskin, J.: Antichains for the automata-based approach to model-checking. arXiv preprint arXiv:0902.3958 (2009)
3. Fisher, C., Fogarty, S., Vardi, M.: Random models for efficient Büchi universality checking. Technical report. Department of Computer Science, Rice University, Houston, TX, October 2016. http://www.cs.rice.edu/~vardi
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). doi:10.1007/978-3-642-12002-2_17
5. Fogarty, S., Vardi, M.Y.: Büchi complementation and size-change termination. In: Kowalewski, S., Philippou, A. (eds.) TACAS 2009. LNCS, vol. 5505, pp. 16–30. Springer, Heidelberg (2009). doi:10.1007/978-3-642-00768-2_2
6. Frank, O., Strauss, D.: Markov graphs. J. Am. Stat. Assoc. **81**(395), 832–842 (1986)
7. Kleinberg, J., Kumar, R., Raghavan, P., Rajagopalan, S., Tomkins, A.: The web as a graph: measurements, models, and methods. In: Asano, T., Imai, H., Lee, D.T., Nakano, S., Tokuyama, T. (eds.) COCOON 1999. LNCS, vol. 1627, pp. 1–17. Springer, Heidelberg (1999). doi:10.1007/3-540-48686-0_1
8. Karp, R.M.: The transitive closure of a random digraph. Random Struct. Alg. **1**(1), 73–93 (1990)
9. Kupferman, O., Vardi, M.Y.: Weak alternating automata are not that weak. ACM Trans. Comput. Logic (TOCL) **2**(3), 408–429 (2001)
10. Leslie, T.: Efficient approaches to subset construction. Technical report. University of Waterloo, Canada (1995)
11. de Wulf, M., Doyen, L., Henzinger, T.A., Raskin, J.-F.: Antichains: a new algorithm for checking universality of finite automata. In: Ball, T., Jones, R.B. (eds.) CAV 2006. LNCS, vol. 4144, pp. 17–30. Springer, Heidelberg (2006). doi:10.1007/11817963_5
12. 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). doi:10.1007/978-3-642-18098-9_28
13. Michel, M.: Complementation is more difficult with automata on infinite words. CNET, Paris (1988). 15
14. 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. LNCS, vol. 6901, pp. 187–202. Springer, Heidelberg (2011). doi:10.1007/978-3-642-23217-6_13
15. Safra, S.: On the complexity of ω-automata. In: 29th Annual Symposium on Foundations of Computer Science, pp. 319–327. IEEE (1988)
16. Sistla, A.P., Vardi, M.Y., Wolper, P.: The complementation problem for Büchi automata with applications to temporal logic. Theor. Comput. Sci. **49**(2), 217–237 (1987)
17. Tabakov, D., Vardi, M.Y.: Experimental evaluation of classical automata constructions. In: Sutcliffe, G., Voronkov, A. (eds.) LPAR 2005. LNCS, vol. 3835, pp. 396–411. Springer, Heidelberg (2005). doi:10.1007/11591191_28
18. Tabakov, D., Vardi, M.Y.: Model checking Büchi specifications. In: Proceedings of 1st International Conference on Language and Automata Theory and Applications, pp. 565–576 (2007)

19. Vardi, M., Wolper, P.: An automata-theoretic approach to automatic program verification. In: Proceedings of the First Symposium on Logic in Computer Science, pp. 322–331. IEEE Computer Society (1986)
20. Vardi, M.Y.: The Büchi complementation saga. In: Thomas, W., Weil, P. (eds.) STACS 2007. LNCS, vol. 4393, pp. 12–22. Springer, Heidelberg (2007). doi:10.1007/978-3-540-70918-3_2
21. Vardi, M.Y., Wolper, P.: Reasoning about infinite computations. Inf. Comput. **115**(1), 1–37 (1994)