Simulation Subsumption in Ramsey-based Büchi Automata Universality and Inclusion Testing

Parosh Aziz Abdulla[1], Yu-Fang Chen[2], Lorenzo Clemente[3], Lukáš Holík[4], Chih-Duo Hong[2], Richard Mayr[3], and Tomáš Vojnar[4]

[1]Uppsala University [2]Academia Sinica [3]University of Edinburgh
[4]Brno University of Technology

Abstract. There are two main classes of methods for checking universality and language inclusion of Büchi-automata: Rank-based methods and Ramsey-based methods. While rank-based methods have a better worst-case complexity, Ramsey-based methods have been shown to be quite competitive in practice [9, 8]. It was shown in [9] (for universality checking) that a simple subsumption technique, which avoids exploring certain cases, greatly improves the performance of the Ramsey-based method. Here, we present a much more general subsumption technique for the Ramsey-based method, which is based on using simulation preorder on the states of the Büchi-automata. This technique applies to both universality and inclusion checking, yielding a substantial performance gain over the previous simple subsumption approach of [9].

1 Introduction

Universality and language-inclusion checking are important problems in the theory of automata, with significant applications, e.g., in model-checking. More precisely, the problem of checking whether an implementation meets a specification can be formulated as a language inclusion problem. The behavior of the implementation is represented by an automaton $\mathcal{A}$, the specification is given by an automaton $\mathcal{B}$, and one checks whether $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$. As we are generally interested in non-halting computations, we use automata as acceptors of languages over *infinite words*. In this paper, we concentrate on *Büchi automata*, where accepting runs are those containing some accepting state infinitely often.

A naïve inclusion-checking algorithm involves complementation: One has that $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$ iff $\mathcal{L}(\mathcal{A}) \cap \overline{\mathcal{L}(\mathcal{B})} = \emptyset$. However, the complementary automaton $\overline{\mathcal{B}}$ is, in the worst case, exponentially bigger than the original automaton $\mathcal{B}$. Hence, direct complementation should be avoided.

Among methods that keep the complementation step implicit, the following two have recently gained interest: *Rank-based* and *Ramsey-based* methods. The former uses a rank-based analysis of rejecting runs [13], leading to a simplified complementation procedure. The latter is based on Büchi's original combinatorial Ramsey-based argument for showing closure of ω-regular languages under complementation [3], which has been improved and shown to be implementable in [15]. Notice that a high worst-case complexity is unavoidable, since both universality and language-inclusion testing are PSPACE-complete problems.

However, in practice, subsumption techniques can often greatly speed up universality/inclusion checking by avoiding the exploration of certain cases that are subsumed by other cases. Recently, [4] described a simple set-inclusion-based subsumption technique for speeding up the rank-based technique, for both universality and language inclusion checking, capable of handling automata of several order of magnitude larger than previously possible. Similarly, [9] improved the Ramsey-based method (but only for universality checking) by a simple subsumption technique that compares finite labeled graphs (using set-inclusion on the set of arcs, plus an order on the labels; see the last par. in Section 3).

We improve the Ramsey-based approach. The contribution of this paper is twofold. First, we show how to employ simulation preorder to generalize the simple subsumption technique of [9] for Ramsey-based universality checking. Furthermore, we introduce a simulation-based subsumption relation for Ramsey-based language inclusion checking, thus extending the theory of subsumption to the realm of Ramsey-based inclusion checking.

Experimental results show that our algorithm based on simulation subsumption significantly and consistently outperforms the algorithm based on the original subsumption of [9]. We perform the evaluation on Büchi automata models of several mutual exclusion algorithms (the largest case has several thousands of states and tens of thousands of transitions), random Büchi automata generated from LTL formulae, and Büchi automata generated from the random model of Tabakov and Vardi [17]. In many cases, the difference between the two approaches is very significant. For example, our approach finishes an experiment on the Bakery algorithm in minutes, while the original approach cannot handle it in hours. In the largest examples generated from LTL formulae, our approach is on average 20 times faster than the original one when testing universality and more than 1900 times faster when testing language inclusion. All relevant information is provided online [20] enabling interested readers to reproduce our experiments.

2 Preliminaries

A *Büchi Automaton (BA)* $\mathcal{A}$ is a tuple $(\Sigma, Q, I, F, \delta)$ where Σ is a finite alphabet, Q is a finite set of states, $I \subseteq Q$ is a non-empty set of *initial* states, $F \subseteq Q$ is a set of *accepting* states, and $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation. For convenience, we write $p \xrightarrow{a} q$ instead of $(p, a, q) \in \delta$.

A *run* of $\mathcal{A}$ on a word $w = a_1 a_2 \ldots \in \Sigma^\omega$ *starting* in a state $q_0 \in Q$ is an infinite sequence $q_0 q_1 q_2 \ldots$ such that $q_{j-1} \xrightarrow{a_j} q_j$ for all $j > 0$. The run is *accepting* iff $q_i \in F$ for infinitely many i. The *language of* $\mathcal{A}$ is the set $\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^\omega \mid \mathcal{A}$ has an accepting run on w starting from some $q_0 \in I\}$.

A *path* in $\mathcal{A}$ on a finite word $w = a_1 \ldots a_n \in \Sigma^+$ is a finite sequence $q_0 q_1 \ldots q_n$ where $q_{j-1} \xrightarrow{a_j} q_j$ for all $0 < j \leq n$. The path is *accepting* iff $q_i \in F$ for some $0 \leq i \leq n$. We define the following predicates for $p, q \in Q$: (1) $p \overset{w}{\underset{F}{\Longrightarrow}} q$ iff there is an accepting path on w from p to q. (2) $p \overset{w}{\Longrightarrow} q$ iff there is a path (not necessarily accepting) on w from p to q. (3) $p \overset{w}{\not\Longrightarrow} q$ iff there is no path on w from p to q.

Define $E = Q \times \{0, 1, -1\} \times Q$ and let $G_\mathcal{A}$ be the largest subset of 2^E whose elements contain exactly one member of $\{\langle p, 0, q\rangle, \langle p, 1, q\rangle, \langle p, -1, q\rangle\}$ for any $p, q \in Q$. Each element in $G_\mathcal{A}$ is a $\{0, 1, -1\}$-arc-labeled graph on Q.

For each pair of states $p, q \in Q$, we define the following three sets of languages: (1) $\mathcal{L}(p,1,q) = \{w \in \Sigma^+ \mid p \overset{w}{\underset{F}{\Longrightarrow}} q\}$, (2) $\mathcal{L}(p,0,q) = \{w \in \Sigma^+ \mid p \overset{w}{\Longrightarrow} q \wedge \neg\left(p \overset{w}{\underset{F}{\Longrightarrow}} q\right)\}$, (3) $\mathcal{L}(p,-1,q) = \{w \in \Sigma^+ \mid p \overset{w}{\not\Longrightarrow} q\}$. As in [9], the language of a graph $g \in G_{\mathcal{A}}$ is defined as the intersection of the languages of arcs in g, i.e., $\mathcal{L}(g) = \bigcap_{\langle p,a,q\rangle \in g} \mathcal{L}(p,a,q)$. Notice that the languages of the graphs in $G_{\mathcal{A}}$ form a partition of Σ^+ since they are intersections of languages of the arcs. Define Y_{gh} to be the ω-regular language $\mathcal{L}(g) \cdot \mathcal{L}(h)^\omega$.

Lemma 1. *(1) $\Sigma^\omega = \bigcup_{g,h \in G_{\mathcal{A}}} Y_{gh}$. (2) For $g, h \in G_{\mathcal{A}}$ s.t. $\mathcal{L}(g), \mathcal{L}(h) \neq \emptyset$, either $Y_{gh} \cap \mathcal{L}(\mathcal{A}) = \emptyset$ or $Y_{gh} \subseteq \mathcal{L}(\mathcal{A})$. (3) $\overline{\mathcal{L}(\mathcal{A})} = \bigcup_{g,h \in G_{\mathcal{A}} \wedge Y_{gh} \cap \mathcal{L}(\mathcal{A}) = \emptyset} Y_{gh}$.*

In fact, Lemma 1 is a relaxed version of the lemma proved by a Ramsey-based argument described in [15, 8, 9]. A proof can be found in Appendix A.

3 Ramsey-based Universality Testing

Based on Lemma 1, one can construct an algorithm for checking universality of BA [8]. This type of algorithm is said to be Ramsey-based since the proof of Lemma 1 relies on the infinite Ramsey theorem. Lemma 1 implies that $\mathcal{L}(\mathcal{A})$ is universal iff $\forall g, h \in G_{\mathcal{A}} : \ Y_{gh} \subseteq \mathcal{L}(\mathcal{A})$. Since $\mathcal{L}(g) = \emptyset$ or $\mathcal{L}(h) = \emptyset$ implies $Y_{gh} \subseteq \mathcal{L}(\mathcal{A})$, it suffices to build and check graphs with nonempty languages in $G_{\mathcal{A}}$ when testing universality.

As proposed in [8, 9, 12], the set $G^f_{\mathcal{A}} = \{g \in G_{\mathcal{A}} \mid \mathcal{L}(g) \neq \emptyset\}$ can be generated iteratively as follows. First, given $g, h \in G_{\mathcal{A}}$, their composition $g;h$ is defined as

$$
\begin{aligned}
&\{\langle p,-1,q\rangle \mid \forall t \in Q : (\langle p,a,t\rangle \in g \wedge \langle t,b,q\rangle \in h) \rightarrow (a = -1 \vee b = -1)\} \cup \\
&\{\langle p,0,q\rangle \mid \exists r \in Q : \langle p,0,r\rangle \in g \wedge \langle r,0,q\rangle \in h \wedge \\
&\qquad \wedge\ \forall t \in Q : (\langle p,a,t\rangle \in g \wedge \langle t,b,q\rangle \in h) \rightarrow (a \neq 1 \wedge b \neq 1)\} \cup \\
&\{\langle p,1,q\rangle \mid \exists r \in Q : \langle p,a,r\rangle \in g \wedge \langle r,b,q\rangle \in h \wedge \neg(a \neq 1 \wedge b \neq 1)\}.
\end{aligned}
$$

For all $a \in \Sigma$, define the single-character graph $g_a = \{\langle p,-1,q\rangle \mid q \notin \delta(p,a)\} \cup \{\langle p,0,q\rangle \mid p \in (Q \setminus F) \wedge q \in (\delta(p,a) \setminus F)\} \cup \{\langle p,1,q\rangle \mid q \in \delta(p,a) \wedge \{p,q\} \cap F \neq \emptyset\}$. Let $G^1_{\mathcal{A}} = \{g_a \mid a \in \Sigma\}$. As stated in Lemma 2, one can obtain $G^f_{\mathcal{A}}$ by repeatedly composing graphs in $G^1_{\mathcal{A}}$ until a fixpoint is reached.

Lemma 2. *A graph g is in $G^f_{\mathcal{A}}$ iff $\exists g_1, \ldots, g_n \in G^1_{\mathcal{A}} : \ g = g_1; \ldots; g_n$.*

It remains to sketch how to check that no pair $\langle g,h\rangle$ of graphs $g, h \in G^f_{\mathcal{A}}$ is a counterexample to universality, which, by Point 3 of Lemma 1, reduces to testing $Y_{gh} \cap \mathcal{L}(\mathcal{A}) \neq \emptyset$. The so called *lasso-finding test*, proposed in [9], can be used for this purpose. A pair of graphs $\langle g,h\rangle$ passes the lasso-finding test (denoted $LFT(g,h)$) iff there is an arc $\langle p,a_0,q_0\rangle$ in g and an infinite sequence of arcs $\langle q_0,a_1,q_1\rangle, \langle q_1,a_2,q_2\rangle, \ldots$ in h s.t. $p \in I$, $a_i \in \{0,1\}$ for all $i \in \mathbb{N}$, and $a_j = 1$ for infinitely many $j \in \mathbb{N}$. The following lemma is proved in Appendix B.

Algorithm 1: *Ramsey-based Universality Checking*

Input: A BA $\mathcal{A} = (\Sigma, Q, I, F, \delta)$, the set of all single-character graphs $G^1_{\mathcal{A}}$
Output: TRUE if $\mathcal{A}$ is universal. Otherwise, FALSE.

$Next := G^1_{\mathcal{A}}$; $Processed := \emptyset$;
while $Next \neq \emptyset$ **do**
  Pick and remove a graph g from $Next$;
  foreach $h \in Processed$ **do**
    if $\neg LFT(g,h) \vee \neg LFT(h,g) \vee \neg LFT(g,g)$ **then return** *FALSE*;
  Add g to $Processed$;
  foreach $h \in G^1_{\mathcal{A}}$ **do if** $g;h \notin Processed$ **then** Add $g;h$ to $Next$;
return *TRUE*;

Lemma 3. *$\mathcal{L}(\mathcal{A})$ is universal iff $LFT(g,h)$ for all $g,h \in G^f_{\mathcal{A}}$.*

To be more specific, the procedure for the lasso-finding test works as follows. It (1) finds all 1-SCCs (strongly connected components that contain only $\{0,1\}$-labeled arcs and at least one of the arcs is 1-labeled) in h, (2) records the set of states T_h from which there is an $\{0,1\}$-labeled path to some state in some 1-SCCs, (3) records the set of states H_g such that for all $q \in H_g$, there exists an arc $\langle p,a,q\rangle \in g$ for some $p \in I$ and $a \geq 0$, and then (4) checks if $H_g \cap T_h \neq \emptyset$. We have $LFT(g,h)$ iff $H_g \cap T_h \neq \emptyset$. This procedure is *polynomial* in the number of $\{0,1\}$-labeled arcs in g and h.

Finally, Algorithm 1 gives a naïve universality test obtained by combining the above principles for generating $G^f_{\mathcal{A}}$ and using LFT. A more efficient version of the algorithm is given in [9], using the following idea. For $f,g,h \in G_{\mathcal{A}}$, we say that $g \sqsubseteq h$ iff for each arc $\langle p,a,q\rangle \in g$, there is an arc $\langle p,a',q\rangle \in h$ such that $a \leq a'$. If $g \sqsubseteq h$, we have that (1) $LFT(f,g) \implies LFT(f,h)$ and (2) $LFT(g,f) \implies LFT(h,f)$ for all $f \in G_{\mathcal{A}}$. Since the algorithm searches for counterexamples to universality, the tests on h are subsumed by the tests on g, and thus h can be discarded. We refer to this method, which is based on the relation $\sqsubseteq$, as *subsumption*, in contrast to our more general *simulation subsumption* which is described in the next section.

4 Improving Universality Testing via Simulation

In this section, we describe our technique to use simulation-based subsumption in order to accelerate the Ramsey-based universality test [9] for Büchi automata.

A *simulation* on a BA $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ is a relation $R \subseteq Q \times Q$ such that pRr only if (1) $p \in F \implies r \in F$, and (2) for every transition $p \xrightarrow{a} p'$, there is a transition $r \xrightarrow{a} r'$ such that $p'Rr'$. It can be shown that there exists a unique maximal simulation, which is a preorder (called *simulation preorder* and denoted by $\preceq_{\mathcal{A}}$ or just $\preceq$ when $\mathcal{A}$ is clear from the context), computable in time $\mathcal{O}(|\Sigma||\delta|)$ [10, 11]. The relation $\simeq \; = \; \preceq \cap \succeq$ is called *simulation equivalence*.

If $\mathcal{A}$ is interpreted as an automaton over finite words, $\preceq$ implies language containment, and quotienting w.r.t. $\simeq$ preserves the regular language. If $\mathcal{A}$ is interpreted as a BA, then the particular type of simulation defined above is called *direct simulation*. It implies ω-language containment, and (unlike for fair simulation [6]) quotienting w.r.t. $\simeq$ preserves the ω-regular language of $\mathcal{A}$.

Our method for accelerating the Ramsey-based universality test [9] of $\mathcal{A}$ is based on two optimizations which we describe below. We will also show correctness of the optimizations through Lemmas 4–8 (see below).

Optimization 1. First, we show that it is sufficient to generate graphs in $G^f_{\mathcal{A}}$ that are minimal not only w.r.t. $\sqsubseteq$, but w.r.t. a weaker relation, referred to as the *simulation-subsumption-based relation* $\sqsubseteq^{\forall\exists}$.

Definition 1. *For any $g, h \in G_{\mathcal{A}}$, we say that $g \sqsubseteq^{\forall\exists} h$ iff for every arc $\langle p, a, q\rangle \in g$, there exists an arc $\langle p, a', q'\rangle \in h$ such that $a \leq a'$ and $q \preceq q'$.*

Indeed, by Lemma 7, for any graphs $g, h \in G^f_{\mathcal{A}}$ such that $g \sqsubseteq^{\forall\exists} h$, we can ignore all lasso-finding tests related to h. More precisely, the lemma implies that, for any $f \in G^f_{\mathcal{A}}$, $LFT(g, f) \implies LFT(h, f)$ and $LFT(f, g) \implies LFT(f, h)$. Moreover, by Lemma 6, graph composition is monotonic w.r.t. $\sqsubseteq^{\forall\exists}$: the composition of smaller graphs will yield smaller graphs, and hence we can also ignore all lasso-finding tests related to any extension $h; f$ of h by some $f \in G^f_{\mathcal{A}}$.

Optimization 2. Next, we show that even the structure of the particular graphs in $G_{\mathcal{A}}$ can be simplified via simulation-subsumption allowing us to replace some $\{0, 1\}$-labeled arcs by negative arcs. Since the time complexity of the lasso-finding test is polynomial in the number $\{0, 1\}$-labeled arcs, such a simplification can make the test more time efficient.

For the purpose above, we define a (possibly non-deterministic) operation *Min* that maps each graph $g \in G_{\mathcal{A}}$ to a graph $Min(g) \in G_{\mathcal{A}}$ such that $Min(g) \leqslant^* g$, where $\leqslant^*$ is defined as follows.

Definition 2. *For any graphs $g, h \in G_{\mathcal{A}}$, we write $g \leqslant h$ iff there exist arcs $\langle p, a, q\rangle \in h$ and $\langle p, a', q'\rangle \in g \cap h$ s.t. $a \leq a'$, $q \preceq q'$, and $g = (h \setminus \{\langle p, a, q\rangle\}) \cup \{\langle p, a'', q\rangle\}$ where $a'' \leq a$. The relation $\leqslant^*$ is the transitive closure of $\leqslant$.*

Here, $g \leqslant^* h$ means that g is either equal to h or it is a reduced version of h that can be derived from h by weakening some of the arcs that are anyway "simulation covered" by other arcs present both in g and h. We write $G^m_{\mathcal{A}} = \{g \in G_{\mathcal{A}} \mid \exists h \in G^f_{\mathcal{A}} : \ g \leqslant^* h\}$ to denote the set of reduced versions of graphs with nonempty languages.

In practice, *Min* can be implemented such that it returns a graph which is as $\leqslant^*$-small as possible (meaning that as many arcs as possible will be restricted down to -1).

Concerning the correctness of Optimization 2, note that the relation $\leqslant^*$ does not preserve the language of graphs (and often for $g \leqslant^* h$, $\mathcal{L}(g) = \emptyset$ when $\mathcal{L}(h) \neq \emptyset$). However, by Lemma 4 below, graphs related by $\leqslant^*$ are equivalent w.r.t. the $\sqsubseteq^{\forall\exists}$ relation. That is why, by Lemma 7, we can replace lasso-finding tests on graphs from $G^f_{\mathcal{A}}$ by graphs from $G^m_{\mathcal{A}}$. Moreover, by Lemma 8, $G^m_{\mathcal{A}}$ is closed under composition, and composition of graphs in $G^m_{\mathcal{A}}$ is monotone w.r.t. $\sqsubseteq^{\forall\exists}$ in the sense of Lemma 6. Thus, it suffices to consider just graphs in $G^m_{\mathcal{A}}$.

Correctness of the Optimizations. Let $\simeq^{\forall\exists} = \sqsubseteq^{\forall\exists} \cap (\sqsubseteq^{\forall\exists})^{-1}$. The definition of $\leqslant$ and $\leqslant^*$ together with transitivity of $\simeq^{\forall\exists}$ imply the following basic lemma.

Lemma 4. *For any $g, h \in G_{\mathcal{A}}$, if $g \leqslant^* h$, then $g \simeq^{\forall\exists} h$.*

Next, we prove an auxiliary lemma, which is subsequently used to prove Lemma 6, expressing the needed monotonicity of composition of graphs from $G^m_{\mathcal{A}}$ (in fact, the lemma is even a bit more general than that).

Lemma 5. *Let g be a graph in $G^m_{\mathcal{A}}$. We have that $\langle p, a, q\rangle \in g \ \wedge\ p \preceq p'$ implies $\exists \langle p', a', q'\rangle \in g :\ a \le a' \ \wedge\ q \preceq q'$.*

Proof. If $a = -1$, the lemma trivially holds (e.g., by taking $q' = q$). Assume therefore $a \in \{0, 1\}$. From $g \in G^m_{\mathcal{A}}$, there is some $g' \in G^f_{\mathcal{A}}$ such that $g \leqslant^* g'$. Since $\mathcal{L}(g') \neq \emptyset$ and $a \in \{0, 1\}$, there is some word $w \in \mathcal{L}(g')$ such that $p \overset{w}{\Longrightarrow} q$. Since $p \preceq p'$, there is some q'' such that $p' \overset{w}{\Longrightarrow} q''$, $q \preceq q''$, and if $p \underset{F}{\overset{w}{\Longrightarrow}} q$, then $p' \underset{F}{\overset{w}{\Longrightarrow}} q''$. Since $w \in \mathcal{L}(g')$, $\langle p', a'', q''\rangle \in g'$ for $a \le a''$. From Lemma 4, we get that there is an arc $\langle p', a', q'\rangle \in g$ such that $a \le a'' \le a'$ and $q \preceq q'' \preceq q'$. □

Lemma 6. *Let $f, g, f' \in G_{\mathcal{A}}$ and $g' \in G^m_{\mathcal{A}}$ be graphs s.t. $f \sqsubseteq^{\forall\exists} f'$ and $g \sqsubseteq^{\forall\exists} g'$. Then $f; g \sqsubseteq^{\forall\exists} f'; g'$.*

Proof. We consider an arc $\langle p, c, r\rangle$ in $f; g$ and show that $f'; g'$ must contain a larger arc w.r.t. $\sqsubseteq^{\forall\exists}$. The case $c = -1$ is trivial. For $c \in \{0, 1\}$ there must be arcs $\langle p, a, q\rangle \in f$ and $\langle q, b, r\rangle \in g$ where $a, b \in \{0, 1\}$ and $c = \max(\{a, b\})$. Since $f \sqsubseteq^{\forall\exists} f'$, there is an arc $\langle p, a', q'\rangle \in f'$ with $a \le a'$ and $q \preceq q'$. Since $g \sqsubseteq^{\forall\exists} g'$, there is an arc $\langle q, b', r'\rangle \in g'$ with $b \le b'$ and $r \preceq r'$. Since $g' \in G^m_{\mathcal{A}}$, Lemma 5 implies that there is an arc $\langle q', b'', r''\rangle \in g'$ s.t. $b \le b' \le b''$ and $r \preceq r' \preceq r''$. Thus $\langle p, c'', r''\rangle \in f'; g'$ where $c = \max(\{a, b\}) \le \max(\{a', b''\}) \le c''$ and $r \preceq r''$. □

Below, we prove a lemma allowing us to replace lasso-finding tests on graphs by lasso-finding tests on (minimized versions of) smaller graphs.

Lemma 7. *Let e, f, g, h be graphs in $G_{\mathcal{A}}$ such that $\{f, h\} \cap G^m_{\mathcal{A}} \neq \emptyset$, $e \sqsubseteq^{\forall\exists} g$, and $f \sqsubseteq^{\forall\exists} h$. Then $LFT(e, f) \implies LFT(g, h)$.*

Proof. If $LFT(e, f)$, there exist an arc $\langle p, a_0, q_0\rangle \in e$ and an infinite sequence of arcs $\langle q_0, a_1, q_1\rangle$, $\langle q_1, a_2, q_2\rangle$, $\ldots$ in f s.t. $p \in I$, $a_i \in \{0, 1\}$ for all i, and $a_j = 1$ for infinitely many j. By the premise $e \sqsubseteq^{\forall\exists} g$, there is $\langle p, a'_0, q'_0\rangle \in g$ s.t. $a_0 \le a'_0$ and $q_0 \preceq q'_0$ (Property 1). We now show how to construct an infinite sequence $q'_0 a'_1 q'_1 a'_2 q'_2 \cdots$ that satisfies the following (Property 2): $\langle q'_n, a'_{n+1}, q'_{n+1}\rangle \in h$, $a_{n+1} \le a'_{n+1}$, and $q_n \preceq q'_n$ for all $n \ge 0$. We do this by proving that every finite sequence $q'_0 a'_1 q'_1 \dots q'_{k-1} a'_k q'_k$ satisfying Property 2 can be extended by one step to length $k+1$ while preserving Property 2. Moreover, such a sequence can be started (case $k=0$) since for $k=0$, Property 1 implies Property 2 as q'_1 is not in the sequence then. For the extension, we distinguish two (non-exclusive) cases:

1. $f \in G^m_{\mathcal{A}}$. Since $\langle q_k, a_{k+1}, q_{k+1}\rangle \in f$ and $q_k \preceq q'_k$ (by Property 2), Lemma 5 implies that there exists an arc $\langle q'_k, a, q\rangle \in f$ such that $a_{k+1} \leq a$ and $q_{k+1} \preceq q$. Since $f \sqsubseteq^{\forall\exists} h$, there must be some arc $\langle q'_k, a'_{k+1}, q'_{k+1}\rangle \in h$ such that $a_{k+1} \leq a \leq a'_{k+1}$ and $q_{k+1} \preceq q \preceq q'_{k+1}$.
2. $h \in G^m_{\mathcal{A}}$. Since $\langle q_k, a_{k+1}, q_{k+1}\rangle \in f$ and $f \sqsubseteq^{\forall\exists} h$, there is some arc $\langle q_k, a, q\rangle \in h$ s.t. $a_{k+1} \leq a$ and $q_{k+1} \preceq q$. Since $q_k \preceq q'_k$ (by Property 2) and $\langle q_k, a, q\rangle \in h$, Lemma 5 implies that there is an arc $\langle q'_k, a'_{k+1} q'_{k+1}\rangle \in h$ such that $a_{k+1} \leq a \leq a'_{k+1}$ and $q_{k+1} \preceq q \preceq q'_{k+1}$.

To conclude, there exist an arc $\langle p, a'_0, q'_0\rangle \in g$ and an infinite sequence of arcs $\langle q'_0, a'_1, q'_1\rangle$, $\langle q'_1, a'_2, q'_2\rangle$, $\dots$ in h such that $p \in I$ and $a'_i \in \{0,1\}$ for all i and $a'_j = 1$ for infinitely many j. Hence, $LFT(g,h)$ holds. □

Finally, we show that the set $G^m_{\mathcal{A}}$ is closed under composition.

Lemma 8. *$G^m_{\mathcal{A}}$ is closed under composition. That is, $\forall e, f \in G^m_{\mathcal{A}} : e; f \in G^m_{\mathcal{A}}$.*

Proof. As $e, f \in G^m_{\mathcal{A}}$, there are $g, h \in G^f_{\mathcal{A}}$ with $e \leqslant^* g$ and $f \leqslant^* h$. We will show that $e; f \leqslant^* g; h$. Since by Lemma 2, $g; h \in G^f_{\mathcal{A}}$, this will give $e; f \in G^m_{\mathcal{A}}$. By the definition of $\leqslant^*$, there are $g_0, h_0, g_1, h_1, \dots, g_n, h_n \in G^m_{\mathcal{A}}$ s.t. $g_0 = g, h_0 = h, g_n = e, h_n = f$ and for each $i : 1 \leq i \leq n$, $g_i \leqslant g_{i-1}$ and $h_i \leqslant h_{i-1}$. We will show that for any $i : 1 \leq i \leq n$, $g_i; h_i \leqslant^* g_{i-1}; h_{i-1}$ which implies that $e; f \leqslant^* g; h$.

Since $g_i \leqslant g_{i-1}$, for every arc $\langle p, a, q\rangle \in g_i$, $\langle p, a', q\rangle \in g_{i-1}$ with $a \leq a'$. Since $h_i \leqslant h_{i-1}$, for every arc $\langle q, b, r\rangle \in h_i$, $\langle q, b', r\rangle \in h_{i-1}$ with $b \leq b'$. Therefore, by the definition of composition, for each $\langle p, c, r\rangle \in g_i; h_i$, we have $\langle p, c', r\rangle \in g_{i-1}; h_{i-1}$ with $c \leq c'$. To prove that $g_i; h_i \leqslant^* g_{i-1}; h_{i-1}$, it remains to show that there is also $\langle p, \bar{c}, \bar{r}\rangle \in g_i; h_i \cap g_{i-1}; h_{i-1}$ with $c' \leq \bar{c}$ and $r \preceq \bar{r}$. The case when $c = c'$ is trivial. If $c < c'$, then $0 \leq c'$ and thus there are $\langle p, a, q\rangle \in g_{i-1}$ and $\langle q, b, r\rangle \in h_{i-1}$ s.t. $c' = \max(\{a, b\})$. Since $g_i \leqslant g_{i-1}$, there is $\langle p, \bar{a}, \bar{q}\rangle \in g_i \cap g_{i-1}$ with $a \leq \bar{a}$ and $q \preceq \bar{q}$. By Lemma 5 and as $h_i \in G^m_{\mathcal{A}}$, there is also $\langle \bar{q}, b', r'\rangle \in h_i$ with $b \leq b'$ and $r \preceq r'$. Since $h_i \leqslant h_{i-1}$, there is $\langle \bar{q}, \bar{b}, \bar{r}\rangle \in h_i \cap h_{i-1}$ where $b' \leq \bar{b}$ and $r' \preceq \bar{r}$. Together with $\langle p, \bar{a}, \bar{q}\rangle \in g_i \cap g_{i-1}$, this implies that there is $\langle p, \bar{c}, \bar{r}\rangle \in g_i; h_i \cap g_{i-1}; h_{i-1}$ with $\max(\{\bar{a}, \bar{b}\}) \leq \bar{c}$ and $r' \preceq \bar{r}$. Since $c' = \max(\{a, b\}) \leq \max(\{\bar{a}, \bar{b}\}) \leq \bar{c}$ and $r \preceq r' \preceq \bar{r}$, $\langle p, \bar{c}, \bar{r}\rangle$ is the wanted arc. □

The Algorithm. Algorithm 2 describes our approach in pseudo-code. In this algorithm, Lines 4, 5, 14, and 15 implement Optimization 1; Lines 1 and 13 implement Optimization 2. Overall, the algorithm works such that for each graph in $G^f_{\mathcal{A}}$, a minimization of some $\sqsubseteq^{\forall\exists}$-smaller graph is generated and used in the lasso-finding tests (and only minimizations of graphs $\sqsubseteq^{\forall\exists}$-smaller than those in $G^f_{\mathcal{A}}$ are generated and used). The correctness of the algorithm is stated in Theorem 1, which is proved in Appendix C using the closure of $G^m_{\mathcal{A}}$ under composition

Algorithm 2: *Simulation-optimized Ramsey-based Universality Checking*

Input: A BA $\mathcal{A} = (\Sigma, Q, I, F, \delta)$, the set of all single-character graphs $G^1_{\mathcal{A}}$.
Output: TRUE if $\mathcal{A}$ is universal. Otherwise, FALSE.

$Next := \{Min(g) \mid g \in G^1_{\mathcal{A}}\}$; $Init := \emptyset$;
while $Next \neq \emptyset$ **do**
  Pick and remove a graph g from $Next$;
  if $\exists f \in Init : f \sqsubseteq^{\forall\exists} g$ **then** Discard g and **continue**;
  Remove all graphs f from $Init$ s.t. $g \sqsubseteq^{\forall\exists} f$;
  Add g into $Init$;
$Next := Init$; $Processed := \emptyset$;
while $Next \neq \emptyset$ **do**
  Pick a graph g from $Next$;
  if $\exists h \in Processed : \neg LFT(h, g) \vee \neg LFT(g, h) \vee \neg LFT(g, g)$ **then**
    return *FALSE*;
  Remove g from $Next$ and add it to $Processed$;
  foreach $h \in Init$ **do**
    $f = Min(g; h)$;
    if $\exists k \in Processed \cup Next : k \sqsubseteq^{\forall\exists} f$ **then** Discard f and **continue**;
    Remove all graphs k from $Processed \cup Next$ s.t. $f \sqsubseteq^{\forall\exists} k$;
    Add f into $Next$;
return *TRUE*;

stated in Lemma 8, the monotonicity from Lemma 6, and the preservation of lasso-finding tests from Lemma 7.

Theorem 1. *Alg. 2 eventually terminates. It returns TRUE iff $\mathcal{A}$ is universal.*

5 Language Inclusion of BA

Let $\mathcal{A} = (\Sigma, Q_{\mathcal{A}}, I_{\mathcal{A}}, F_{\mathcal{A}}, \delta_{\mathcal{A}})$ and $\mathcal{B} = (\Sigma, Q_{\mathcal{B}}, I_{\mathcal{B}}, F_{\mathcal{B}}, \delta_{\mathcal{B}})$ be two BA. Let $\preceq_{\mathcal{A}}$ and $\preceq_{\mathcal{B}}$ be the maximal simulations on $\mathcal{A}$ and $\mathcal{B}$, respectively. We first introduce some further notations from [8] before explaining how to extend our approach from universality to language inclusion checking. Define the set $E_{\mathcal{A}} = Q_{\mathcal{A}} \times Q_{\mathcal{A}}$. Each element in $E_{\mathcal{A}}$ is an edge $\langle p, q \rangle$ asserting that there is a path from p to q in $\mathcal{A}$. Define the language of an edge $\langle p, q \rangle$ as $\mathcal{L}(p, q) = \{w \in \Sigma^+ \mid p \stackrel{w}{\Longrightarrow} q\}$.

Define $S_{\mathcal{A},\mathcal{B}} = E_{\mathcal{A}} \times G_{\mathcal{B}}$. We call $\mathbf{g} = \langle \overline{g}, g \rangle$ a *supergraph* in $S_{\mathcal{A},\mathcal{B}}$. For any supergraph $\mathbf{g} \in S_{\mathcal{A},\mathcal{B}}$, its language $\mathcal{L}(\mathbf{g})$ is defined as $\mathcal{L}(\overline{g}) \cap \mathcal{L}(g)$. Let $Z_{\mathbf{gh}}$ be the ω-regular language $\mathcal{L}(\mathbf{g}) \cdot \mathcal{L}(\mathbf{h})^{\omega}$. We say $Z_{\mathbf{gh}}$ is *proper* if $\overline{g} = \langle p, q \rangle$ and $\overline{h} = \langle q, q \rangle$ where $p \in I_{\mathcal{A}}$ and $q \in F_{\mathcal{A}}$. Notice that, by the definition of properness, every proper $Z_{\mathbf{gh}}$ is contained in $\mathcal{L}(\mathcal{A})$. The following is a relaxed version of Lemma 4 in [8] (the constraints of being a proper $Z_{\mathbf{gh}}$ are weaker than those in [8]).

Lemma 9. *(1) $\mathcal{L}(\mathcal{A}) = \bigcup \{Z_{\mathbf{gh}} \mid Z_{\mathbf{gh}}$ is proper$\}$. (2) For all non-empty proper $Z_{\mathbf{gh}}$, either $Z_{\mathbf{gh}} \cap \mathcal{L}(\mathcal{B}) = \emptyset$ or $Z_{\mathbf{gh}} \subseteq \mathcal{L}(\mathcal{B})$. (3) $\mathcal{L}(\mathcal{A}) \cap \overline{\mathcal{L}(\mathcal{B})} = \bigcup \{Z_{\mathbf{gh}} \mid Z_{\mathbf{gh}}$ is proper and $Z_{\mathbf{gh}} \cap \mathcal{L}(\mathcal{B}) = \emptyset\}$.*

The above lemma implies that $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$ iff $\forall \mathbf{g}, \mathbf{h} \in S_{\mathcal{A},\mathcal{B}}$ either $Z_{\mathbf{gh}}$ is not proper or $Z_{\mathbf{gh}} \subseteq \mathcal{L}(\mathcal{B})$. Since $\mathcal{L}(\mathbf{g}) = \emptyset$ or $\mathcal{L}(\mathbf{h}) = \emptyset$ implies $Z_{\mathbf{gh}} \subseteq \mathcal{L}(\mathcal{B})$, for language inclusion checking it is sufficient to build and check only supergraphs with nonempty languages.

Supergraphs in $S^f_{\mathcal{A},\mathcal{B}} = \{\mathbf{g} \in S_{\mathcal{A},\mathcal{B}} \mid \mathcal{L}(\mathbf{g}) \neq \emptyset\}$ can be built as follows. First, supergraphs $\mathbf{g} = \langle\langle p_\mathbf{g}, q_\mathbf{g}\rangle, g\rangle$ and $\mathbf{h} = \langle\langle p_\mathbf{h}, q_\mathbf{h}\rangle, h\rangle$ in $S_{\mathcal{A},\mathcal{B}}$ are *composable* iff $q_\mathbf{g} = p_\mathbf{h}$, and their composition $\mathbf{g}; \mathbf{h} = \langle\langle p_\mathbf{g}, q_\mathbf{h}\rangle, g; h\rangle$. For all $a \in \Sigma$, define the set of single-character supergraphs $S^a = \{\langle\langle p, q\rangle, g_a\rangle \mid q \in \delta_\mathcal{A}(p, a)\}$. Let $S^1_{\mathcal{A},\mathcal{B}} := \bigcup_{a\in\Sigma} S^a$. As in universality checking, one can obtain $S^f_{\mathcal{A},\mathcal{B}}$ by repeatedly composing composable supergraphs in $S^1_{\mathcal{A},\mathcal{B}}$ until a fixpoint is reached.

A method to check whether a pair of supergraphs $\langle \mathbf{g}, \mathbf{h}\rangle$ is a counterexample to $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$, i.e., a test whether $Z_{\mathbf{gh}}$ is both proper and disjoint from $\mathcal{L}(\mathcal{B})$, was proposed in [8]. A pair of supergraphs $\langle \mathbf{g} = \langle \overline{g}, g\rangle, \mathbf{h} = \langle \overline{h}, h\rangle\rangle$ passes the double-graph test (denoted $DGT(\mathbf{g}, \mathbf{h})$) iff $Z_{\mathbf{gh}}$ is not proper or $LFT(g, h)$.

Lemma 10. *$\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$ iff $DGT(\mathbf{g}, \mathbf{h})$ for all $\mathbf{g}, \mathbf{h} \in S^f_{\mathcal{A},\mathcal{B}}$.*

Analogously to the universality checking algorithm in Section 3, a language inclusion checking algorithm can be obtained by combining the above principles for generating supergraphs in $S^f_{\mathcal{A},\mathcal{B}}$ and using the double-graph test (cf. Appendix D).

6 Improving Language Inclusion Testing via Simulation

Here we describe our approach of utilizing simulation-based subsumption techniques to improve the Ramsey-based language inclusion test.

In order to be able to use simulation-based subsumption as in Section 4, we lift the subsumption relation $\sqsubseteq^{\forall\exists}$ to supergraphs as follows: Let $\mathbf{g} = \langle\langle p_\mathbf{g}, q_\mathbf{g}\rangle, g\rangle$ and $\mathbf{h} = \langle\langle p_\mathbf{h}, q_\mathbf{h}\rangle, h\rangle$ be two supergraphs in $S_{\mathcal{A},\mathcal{B}}$. Let $\mathbf{g} \sqsubseteq^{\forall\exists}_S \mathbf{h}$ iff $p_\mathbf{g} = p_\mathbf{h}$, $q_\mathbf{g} \succeq_\mathcal{A} q_\mathbf{h}$ and $g \sqsubseteq^{\forall\exists} h$. Define $\simeq^{\forall\exists}_S$ as $\sqsubseteq^{\forall\exists}_S \cap (\sqsubseteq^{\forall\exists}_S)^{-1}$.

Since we want to work with supergraphs that are minimal w.r.t. $\sqsubseteq^{\forall\exists}_S$, we need to change the definition of properness and the respective double-graph test. We say that $Z_{\mathbf{gh}}$ is *weakly proper* iff $\overline{g} = \langle p, q\rangle$ and $\overline{h} = \langle q_1, q_2\rangle$, where $p \in I_\mathcal{A}$, $q_2 \in F_\mathcal{A}$, $q \succeq_\mathcal{A} q_1$ and $q_2 \succeq_\mathcal{A} q_1$.

Definition 3. *Supergraphs* $\mathbf{g}, \mathbf{h} \in S_{\mathcal{A},\mathcal{B}}$ pass the relaxed double-graph test*, written $RDGT(\mathbf{g}, \mathbf{h})$, iff either (1) $Z_{\mathbf{gh}}$ is not weakly proper, or (2) $LFT(g, h)$.*

Notice that, every weakly proper $Z_{\mathbf{gh}}$ is still contained in $\mathcal{L}(\mathcal{A})$, allowing us to prove the following theorem.

Theorem 2. *$\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$ iff $RDGT(\mathbf{g}, \mathbf{h})$ for all $\mathbf{g}, \mathbf{h} \in S^f_{\mathcal{A},\mathcal{B}}$.*

Furthermore, we lift the notions of $\leqslant^*$ and *Min* from Section 4 from graphs to supergraphs. For any two supergraphs $\mathbf{g} = \langle \overline{g}, g\rangle, \mathbf{h} = \langle \overline{h}, h\rangle$ from $S_{\mathcal{A},\mathcal{B}}$ we write $\mathbf{g} \leqslant^*_S \mathbf{h}$ iff $\overline{g} = \overline{h}$ and $g \leqslant^* h$. Then $S^m_{\mathcal{A},\mathcal{B}} = \{\mathbf{g} \in S_{\mathcal{A},\mathcal{B}} \mid \exists \mathbf{h} \in S^f_{\mathcal{A},\mathcal{B}} : \mathbf{g} \leqslant^*_S \mathbf{h}\}$.

Algorithm 3: *Optimized Ramsey-based Language Inclusion Checking*

Input: BA $\mathcal{A} = (\Sigma, Q_\mathcal{A}, I_\mathcal{A}, F_\mathcal{A}, \delta_\mathcal{A})$, $\mathcal{B} = (\Sigma, Q_\mathcal{B}, I_\mathcal{B}, F_\mathcal{B}, \delta_\mathcal{B})$, and the set $S^1_{\mathcal{A},\mathcal{B}}$.
Output: TRUE if $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$. Otherwise, FALSE.

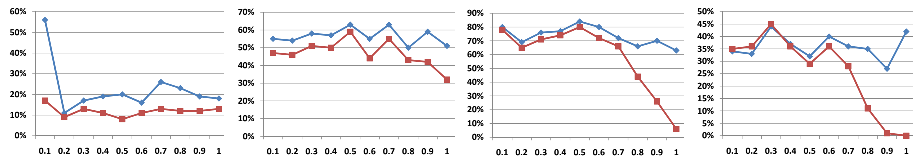
$Next := \{Min_S(\mathbf{g}) \mid \mathbf{g} \in S^1_{\mathcal{A},\mathcal{B}}\}$; $Init := \emptyset$;
while $Next \neq \emptyset$ **do**
    Pick and remove a supergraph $\mathbf{g}$ from $Next$;
    if $\exists \mathbf{f} \in Init : \mathbf{f} \sqsubseteq_S^{\forall\exists} \mathbf{g}$ **then** Discard $\mathbf{g}$ and **continue**;
    Remove all supergraphs $\mathbf{f}$ from $Init$ s.t. $\mathbf{g} \sqsubseteq_S^{\forall\exists} \mathbf{f}$;
    Add $\mathbf{g}$ into $Init$;
$Next := Init$; $Processed := \emptyset$;
while $Next \neq \emptyset$ **do**
    Pick a supergraph $\mathbf{g}$ from $Next$;
    if $\exists \mathbf{h} \in Processed : \neg RDGT(\mathbf{h}, \mathbf{g}) \vee \neg RDGT(\mathbf{g}, \mathbf{h}) \vee \neg RDGT(\mathbf{g}, \mathbf{g})$ **then**
        return *FALSE*;
    Remove $\mathbf{g}$ from $Next$ and add it to $Processed$;
    foreach $\mathbf{h} \in Init$ *where* $\langle \mathbf{g}, \mathbf{h} \rangle$ *are composable* **do**
        $\mathbf{f} := Min_S(\mathbf{g}; \mathbf{h})$;
        if $\exists \mathbf{k} \in Processed \cup Next : \mathbf{k} \sqsubseteq_S^{\forall\exists} \mathbf{f}$ **then** Discard $\mathbf{f}$ and **continue**;
        Remove all supergraphs $\mathbf{k}$ from $Processed \cup Next$ s.t. $\mathbf{f} \sqsubseteq_S^{\forall\exists} \mathbf{k}$;
        Add $\mathbf{f}$ into $Next$;
return *TRUE*;

$Min_S(\mathbf{g})$ again computes a graph that is $\leqslant_S^*$ smaller than $\mathbf{g}$. It is a possibly non-deterministic operation such that $Min_S(\overline{g}, g) = \langle \overline{g}, Min(g) \rangle$.

Like in Section 4, it is now possible to prove a closure of $S^m_{\mathcal{A},\mathcal{B}}$ under composition and a preservation of the double-graph test on $\sqsubseteq_S^{\forall\exists}$-larger supergraphs (cf. Appendix E). What slightly differs is the monotonicity of the composition, caused by the additional composability requirement. To cope with it, we define a new relation $\trianglelefteq^{\forall\exists}$, weakening $\sqsubseteq_S^{\forall\exists}$: For $\mathbf{g} = \langle \langle p, q \rangle, g \rangle, \mathbf{h} = \langle \langle p', q' \rangle, h \rangle \in S_{\mathcal{A},\mathcal{B}}$, $\mathbf{g} \trianglelefteq^{\forall\exists} \mathbf{h}$ iff $p' \preceq p$, $q' \preceq q$, and $g \sqsubseteq^{\forall\exists} h$. Notice that $\sqsubseteq_S^{\forall\exists}$ implies $\trianglelefteq^{\forall\exists}$. From the definitions of $\sqsubseteq_S^{\forall\exists}$, $\trianglelefteq^{\forall\exists}$, and Lemma 6, we then easily get the following relaxed monotonicity lemma.

Lemma 11. *For any* $\mathbf{e}, \mathbf{f}, \mathbf{g} \in S_{\mathcal{A},\mathcal{B}}$ *and* $\mathbf{h} \in S^m_{\mathcal{A},\mathcal{B}}$ *with* $\mathbf{e} \sqsubseteq_S^{\forall\exists} \mathbf{g}$*, and* $\mathbf{f} \trianglelefteq^{\forall\exists} \mathbf{h}$ *where* $\mathbf{e}, \mathbf{f}$ *and* $\mathbf{g}, \mathbf{h}$ *are composable,* $\mathbf{e}; \mathbf{f} \sqsubseteq_S^{\forall\exists} \mathbf{g}; \mathbf{h}$.

Now, to be able to say that it is safe to work with $\sqsubseteq_S^{\forall\exists}$-smaller supergraphs in the incremental supergraph construction, it remains to show that given supergraphs $\mathbf{e}$, $\mathbf{g}$, $\mathbf{h}$ s.t. $\mathbf{e} \sqsubseteq_S^{\forall\exists} \mathbf{g}$ and $\mathbf{g}$, $\mathbf{h}$ are composable, one can always find an $\mathbf{f}$ satisfying the preconditions of Lemma 11—excluding a situation of only the bigger supergraphs $\mathbf{g}$, $\mathbf{h}$ being composable. Fortunately, it is possible to show that the needed supergraph $\mathbf{f}$ always exists in $S^m_{\mathcal{A},\mathcal{B}}$. Moreover, since only 1-letter supergraphs are used on the right of the composition, all supergraphs needed as right operands in the compositional construction can always be found in the minimization of $S^1_{\mathcal{A},\mathcal{B}}$, which can easily be generated.



(a) td=1.5 (b) td=2 (c) td=2.5 (d) td=3

Fig. 1. Timeout Cases of Universality Checking on Tabakov and Vardi's random model. The vertical axis is the percentage of tasks that cannot be finished within the timeout period and the horizontal axis is the acceptance density (*ad*).

Algorithm 3 is our simulation-optimized algorithm for BA inclusion-checking. Its correctness is stated below and proved in Appendix E.

Theorem 3. *Alg. 3 eventually terminates. It returns* TRUE *iff* $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$.

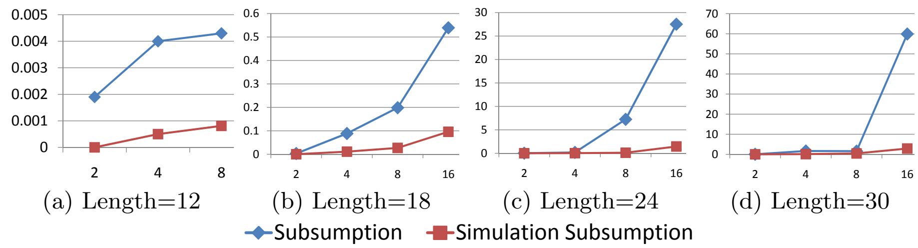
7 Experimental Results

We have implemented both our simulation subsumption algorithms and the original ones of Fogarty and Vardi [8, 9] in Java. For universality testing, we compared our algorithm to the one in [9].[1] For language inclusion testing, we compared our simulation subsumption algorithm to a restricted version that uses only the simple subsumption relation of [9]. (The language inclusion checking algorithm described in [8] does not use any subsumption at all and performed much worse.) We refer interested readers to [20] for all relevant materials needed to reproduce the results. A description of the machines that we used is given in Appendix F.

Universality Testing: We have two sets of experiments. In Experiment 1, we use Tabakov and Vardi's random model. This model fixes the alphabet size to 2 and uses two parameters: *t*ransition *d*ensity (avg. number of outgoing transitions per alphabet symbol) and *a*cceptance *d*ensity (percentage of accepting states). We use this approach with $td = 0.5, 1, \ldots, 4$ and $ad = 0.1, 0.2, \ldots, 1.0$ to generate 100 random BA with 100 states for each combination of *td* and *ad*. We set a timeout period of 3500 sec. In Figure 1, we compare how many timeouts appear when using the two considered approaches. We only list cases with $td = 1.5, 2.0, 2.5, 3.0$, since in the other cases both methods can complete most of the tasks within the timeout period. The average time for both methods is given in Figure 4 in Appendix F. Our approach has a better performance in almost all configurations, and the difference gets larger as *td* and *ad* increase.

Experiment 2 uses BA from random LTL formulae. We generate only valid formulae (in the form $f \vee \neg f$), thus ensuring that the corresponding BA are

[1] The algorithm in [9] uses the subsumption relation $\sqsubseteq$, but a different search strategy than our algorithm. So we also compared two versions of our algorithm with either simulation subsumption $\sqsubseteq^{\forall\exists}$ or just the simple subsumption $\sqsubseteq$ of [9]. While our search strategy with simple subsumption $\sqsubseteq$ is already about 20% faster than [9] on average, the main improvement is achieved by using the simulation subsumpt. $\sqsubseteq^{\forall\exists}$.



(a) Length=12 (b) Length=18 (c) Length=24 (d) Length=30

Fig. 2. Universality checking on automata generated from random LTL. Each point in the figure is the average result of 100 different instances. The horizontal axis is the size of the alphabet. The vertical axis is the average execution time in seconds.

universal. We only focus on valid formulae since most BA generated from random LTL formulae can be recognized as non-universal in almost no time. Thus, the results would have been not interesting. We generate LTL formulae with lengths 12, 18, 24, and 30 and 1–4 propositions (which corresponds to automata with alphabet sizes 2, 4, 8, and 16). For each configuration, we generate 100 BA[2]. The results are shown in Figure 2. The difference between the two approaches is quite large in this set of experiments. With formulae of length 30 and 4 propositions, our approach is 20 times faster than the original subsumption based approach.

Language Inclusion Testing: We again have two sets of experiments. In Experiment 1, we inject (artificial) errors into models of several mutual exclusion algorithms from [14][3], translate both the original and the modified version into BA, and test whether the control flow (w.r.t. the occupation of the critical section) of the two versions is the same. In these models, a state p is accepting iff the critical section is occupied by some process in p. The results are shown in Table 1. The results of a slightly different version, where all states are accepting is given in Table 3 in Appendix F. Our approach significantly and consistently has a better performance than the basic subsumption approach in both versions.

In Experiment 2, we translate two randomly generated LTL formulae to BA $\mathcal{A}$, $\mathcal{B}$ and then check whether $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$. We use formulae of length 10, 15, and 20 with 1–3 propositions (which corresponds to BA of alphabet sizes 2, 4, and 8). For each length of formula and number of propositions, we generate 10 pairs of BA. The relative results are shown in Figure 3. The difference here is significant. For the largest cases (with formula length 20 and 3 propositions), our approach is 1914 times faster than the basic subsumption approach.

8 Conclusions and Extensions

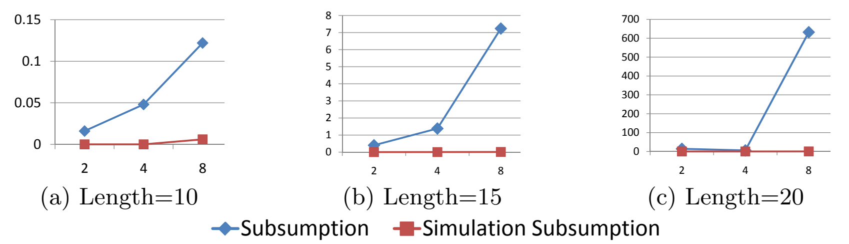
We have introduced simulation-based subsumption techniques for Ramsey-based universality/language-inclusion checking for nondeterministic BA. We evalu-

[2] We do not have formulae having length 12 and 4 propositions because our generator [18] requires that $(length\ of\ formula/3) > number\ of\ propositions$.

[3] The models in [14] are based on guarded commands. We modify the models by randomly weakening or strengthening the guard of some commands.

	Original		Error		Subsumption		Simulation Sub.	
	Tr.	St.	Tr.	St.	$\mathcal{L}(E)\subseteq\mathcal{L}(O)$	$\mathcal{L}(O)\subseteq\mathcal{L}(E)$	$\mathcal{L}(E)\subseteq\mathcal{L}(O)$	$\mathcal{L}(O)\subseteq\mathcal{L}(E)$
Bakery	2697	1506	2085	1146	>1day	>1day	57m55s(F)	>1day
Peterson	34	20	33	20	2.7s(T)	1.4s(F)	0.9s(T)	0.2s(F)
Dining phil. (Ver.1)	212	80	464	161	>1day	>1day	52s(F)	>1day
Dining phil. (Ver.2)	212	80	482	161	>1day	>1day	4m50s(F)	>1day
Fisher	1395	634	3850	1532	>1day	>1day	>1day	>1day
MCS queue lock	21503	7963	3222	1408	oom	2m35s(F)	33m38s(T)	1m11s(F)

Table 1. Language inclusion checking on mutual exclusion algorithms. The columns "Original" and "Error" refer to original, resp., erroneous, model. We test inclusion for both directions. ">1day" = the task cannot be finished within one day. "oom" = required memory > 8GB. If a task completes successfully, we record the run time and whether language inclusion holds "T" or not "F".



(a) Length=10 (b) Length=15 (c) Length=20

Fig. 3. Language inclusion checking on automata generated from random LTL. Each point in the figure is the average result of 10 different instances. The horizontal axis is the size of the alphabet. The vertical axis is the average execution time in seconds.

ated our approach by a wide set of experiments, showing that the simulation-subsumption approach consistently outperforms the basic subsumption of [9].

Our techniques can be extended in several ways. Weaker simulations for BA have been defined in the literature, e.g., delayed/fair simulation [6], or their multipebble extensions [5]. One possibility is to quotient the BA w.r.t. (multipebble) delayed simulation, which (unlike quotienting w.r.t. fair simulation) preserves the language. Furthermore, in our language inclusion checking algorithm, the subsumption w.r.t. direct simulation $\preceq_{\mathcal{A}}$ on $\mathcal{A}$ can be replaced by the weaker delayed simulation (but not by fair simulation). Moreover, in the definition of *weakly proper* in Sect. 6, in the condition $q \succeq_{\mathcal{A}} q_1$, the $\succeq_{\mathcal{A}}$ can be replaced by any other relation that implies ω-language containment, e.g., fair simulation. On the other hand, delayed/fair simulation cannot be used for subsumption in the automaton $\mathcal{B}$ in inclusion checking (nor in universality checking), since Lemma 6 does not carry over.

Next, our language-inclusion algorithm does not currently exploit any simulation preorders *between* $\mathcal{A}$ and $\mathcal{B}$. Of course, direct/delayed/fair simulation between the respective initial states of $\mathcal{A}$ and $\mathcal{B}$ is sufficient (but not necessary) for language inclusion. However, it is more challenging to exploit simulation preorders between internal states of $\mathcal{A}$ and internal states of $\mathcal{B}$.

Finally, it is easy to see that the proposed simulation subsumption technique can be built over *backward simulation preorder* too. It would, however, be interesting to evaluate such an approach experimentally. Further, one could then also try to extend the framework to use some form of *mediated preorders* combining forward and backward simulation preorders like in the approach of [2].

References

1. P. A. Abdulla, Y.-F. Chen, L. Holík, R. Mayr, and T. Vojnar. When Simulation Meets Antichains (On Checking Language Inclusion of NFAs). To appear in *Proc. of TACAS'10*.
2. P. A. Abdulla, Y.-F. Chen, L. Holík, and T. Vojnar. Mediating for Reduction (On Minimizing Alternating Büchi Automata). In *Proc. of FSTTCS'09*, 2009.
3. J.R. Büchi. On a Decision Method in Restricted Second Order Arithmetic. In *Proc. of Int. Con. on Logic, Method, and Phil. of Science*. 1962.
4. L. Doyen and J.-F. Raskin. Improved Algorithms for the Automata-based Approach to Model Checking. In *Proc. of TACAS'07*, LNCS 4424. Springer, 2007.
5. K. Etessami. A Hierarchy of Polynomial-Time Computable Simulations for Automata. In *Proc. of CONCUR'02*, LNCS 2421. Springer, 2002.
6. K. Etessami, T. Wilke, and R.A. Schuller. Fair Simulation Relations, Parity Games, and State Space Reduction for Büchi Automata. *SIAM J. Comp.*, 34(5), 2005.
7. S. Fogarty. Büchi Containment and Size-Change Termination. Master's Thesis, Rice University, 2008.
8. S. Fogarty and M. Y. Vardi. Büchi Complementation and Size-Change Termination. In *Proc. of TACAS'09*, LNCS 5505, 2009.
9. S. Fogarty and M.Y. Vardi. Efficient Büchi Universality Checking. To appear in *Proc. of TACAS'10*. `http://www.cs.rice.edu/~vardi/papers/tacas10rj.pdf`
10. M.R. Henzinger and T.A. Henzinger and P.W. Kopke. Computing Simulations on Finite and Infinite Graphs. In *Proc. FOCS'95*. IEEE CS, 1995.
11. L. Holík and J. Šimáček. Optimizing an LTS-Simulation Algorithm. In *Proc. of MEMICS'09*, 2009.
12. N. D. Jones, C. S. Lee, and A. M. Ben-Amram. The Size-Change Principle for Program Termination. In *Proc. of POPL'01*. ACM SIGPLAN, 2001.
13. O. Kupferman and M.Y. Vardi. Weak Alternating Automata Are Not That Weak. *ACM Transactions on Computational Logic*, 2(2):408-29, 2001.
14. R. Pelánek. BEEM: Benchmarks for Explicit Model Checkers. In *Proc. of SPIN'07*, LNCS 4595. Springer, 2007.
15. A. P. Sistla, M. Y. Vardi, and P. Wolper. The Complementation Problem for Büchi Automata with Applications to Temporal Logic. In *Proc. of ICALP'85*, LNCS 194. Springer, 1985.
16. F. Somenzi and R. Bloem. Efficient Büchi Automata from LTL Formulae. In *Proc. of CAV'00*, LNCS 1855. Springer, 2000.
17. D. Tabakov and M.Y. Vardi. Model Checking Büchi Specifications. In *Proc. of LATA'07*, 2007.
18. Y.-K. Tsay, Y.-F. Chen, M.-H. Tsai, K.-N. Wu, and W.-C. Chan. GOAL: A Graphical Tool for Manipulating Büchi Automata and Temporal Formulae. In *Proc. of TACAS'07*, LNCS 4424. Springer, 2007.
19. M. D. Wulf, L. Doyen, T. A. Henzinger, and J.-F. Raskin. Antichains: A New Algorithm for Checking Universality of Finite Automata. In *Proc. of CAV'06*, LNCS 4144. Springer, 2006.
20. `http://iis.sinica.edu.tw/FMLAB/CAV2010` (capitalize "FMLAB" and "CAV2010").

A Proof of Lemma 1 from Section 2

Define the function $\mathcal{H}(A) = \{\{x, y\} \mid x, y \in A \wedge x \neq y\}$, where A is an countably infinite set. The following is a suitable version of the infinite Ramsey theorem.

Lemma 12. *Let A be an countably infinite set, and let $B = \mathcal{H}(A)$. For any partitioning of B into finitely many classes $B_1, \dots, B_m$, there exists an infinite subset A' of A s.t. $\mathcal{H}(A') \subseteq B_k$, for some k.*

Proof (Lemma 1). We first show Point 1, i.e., $\Sigma^\omega = \bigcup\{Y_{gh} \mid g, h \in G_\mathcal{A}\}$. Let $w = a_0 a_1 \dots$ be any ω-word. We have to show that there exist graphs $g, h \in G_\mathcal{A}$ s.t. $w \in Y_{gh}$.

Consider prefixes $w_i = a_0 \dots a_i$ of w. Each w_i belongs to the language of some graph in $G_\mathcal{A}$, and, being the number of such graphs finite, there exists a graph $g \in G_\mathcal{A}$ s.t. $w_i \in \mathcal{L}(g)$ for infinitely many i's. Let $A = \mathcal{L}(g)$.

For any $w_i, w_j \in \mathcal{L}(g)$, $i \leq j$, let $w_j \ominus w_i = a_{i+1} \dots a_j$ (and define $w_j \ominus w_i$ as $w_i \ominus w_j$ if $i > j$). Let $B = \mathcal{H}(A)$ be the set of unordered pairs of strings from $\mathcal{L}(g)$. Consider the partitioning of $B = \bigcup_{h \in G_\mathcal{A}} B_h$ into finitely many classes, where each class B_h is defined as: $\{w_i, w_j\} \in B_h$ iff $w_j \ominus w_i \in \mathcal{L}(h)$.

By the infinite-Ramsey theorem, there exists a graph h in $G_\mathcal{A}$ and an infinite subset A' of $\mathcal{L}(g)$ s.t. $\mathcal{H}(A') \subseteq B_h$, i.e., for every w_i, w_j in A', $w_j \ominus w_i$ belongs to $\mathcal{L}(h)$. Thus, it is possible to split the word w as follows:

$$w = a_0 \dots a_{i_0 - 1} \quad a_{i_0} \dots a_{i_1 - 1} \quad a_{i_1} \dots a_{i_2 - 1} \quad a_{i_2} \dots$$

where $a_0 \dots a_{i_0-1} \in \mathcal{L}(g)$, and $a_{i_k} \dots a_{i_{k+1}-1} \in \mathcal{L}(h)$ for $k \geq 0$. (Here, i_0 is the least index i s.t. $w_i \in A'$, and, inductively, i_{k+1} is the least index $i > i_k$ s.t. $w_i \in A'$.) Hence, $w \in Y_{gh}$.

We now show Point 2, i.e., $\forall g, h \in G_\mathcal{A}$ with $\mathcal{L}(g), \mathcal{L}(h) \neq \emptyset$, either $Y_{gh} \cap \mathcal{L}(\mathcal{A}) = \emptyset$ or $Y_{gh} \subseteq \mathcal{L}(\mathcal{A})$. We prove that if a word $w \in Y_{gh}$ is in $\mathcal{L}(\mathcal{A})$, then *every* word $w' \in Y_{gh}$ is in $\mathcal{L}(\mathcal{A})$ as well.

The crucial observation is that, from the definition of $\mathcal{L}(g)$, all words in $\mathcal{L}(g)$ share the same reachability properties in $\mathcal{A}$. More precisely, $w \in \mathcal{L}(g)$ implies that, for any pair of states $p, q \in Q$, $w \in \mathcal{L}(p, a, q)$ with $\langle p, a, q\rangle \in g$. This implies that, for any two $w, w' \in \mathcal{L}(g)$, and for any two $p, q \in Q$, 1) w induces a path between p and q iff w' does so, i.e., $p \overset{w}{\Longrightarrow} q$ iff $p \overset{w'}{\Longrightarrow} q$, and 2) w induces an accepting path between p and q iff w' does so, i.e., $p \overset{w}{\underset{F}{\Longrightarrow}} q$ iff $p \overset{w'}{\underset{F}{\Longrightarrow}} q$.

Thus, assume that $w \in Y_{gh}$ is in $\mathcal{L}(\mathcal{A})$ and let w' be any other word in Y_{gh}. We have to show $w' \in \mathcal{L}(\mathcal{A})$. Let $\pi = q_0 q_1 \dots$ be an accepting run of $\mathcal{A}$ over w, thus witnessing w being accepted. We can decompose w in $w = w_0 w_1 w_2 \dots$, with $w_0 \in \mathcal{L}(g)$ and $w_i \in \mathcal{L}(h)$, for all $i > 0$. Moreover, the above decomposition of w induces a decomposition of π in $q_{i_0} \dots q_{i_1} \dots q_{i_2} \dots$ (where $i_0 = 0$) s.t. $q_{i_k} \overset{w_k}{\Longrightarrow} q_{i_{k+1}}$ for any $k \geq 0$, and $q_{i_k} \overset{w_k}{\underset{F}{\Longrightarrow}} q_{i_{k+1}}$ for infinitely many $k > 0$.

We now decompose $w' = w'_0 w'_1 w'_2 \dots$ as we did with w above, i.e., $w'_0 \in \mathcal{L}(g)$ and $w'_i \in \mathcal{L}(h)$, for $i > 0$, and we show how to build an accepting run $\pi' = q'_0 q'_1 \dots$

over w'. We take $q'_0 := q_0$. By $w'_0 \in \mathcal{L}(g)$ and the crucial observation 1) above, $q_0 \stackrel{w'_0}{\Longrightarrow} q_{i_1}$, hence we make the two runs π and π' synchronize at i_1, i.e., $q'_{i_1} := q_{i_1}$. Similarly, for any $k > 0$, by $w'_k \in \mathcal{L}(h)$ and the crucial observation 1), then $q_{i_k} \stackrel{w'_k}{\Longrightarrow} q_{i_{k+1}}$, thus $q'_{i_{k+1}} := q_{i_{k+1}}$, for $k > 0$. Hence, the two paths π and π' synchronize at indices i_k, for $k \geq 0$. Moreover, if $q_{i_k} \underset{F}{\stackrel{w_k}{\Longrightarrow}} q_{i_{k+1}}$ for infinitely many positive k's, then, by the crucial observation 2), $q'_{i_k} \underset{F}{\stackrel{w'_k}{\Longrightarrow}} q'_{i_{k+1}}$ for infinitely many positive k's as well. Hence, if π is an accepting run, then π' is an accepting run. Thus, $w' \in \mathcal{L}(\mathcal{A})$.

Finally, Point 3 follows directly from Points 1 and 2.

□

B Proofs of Lemmas from Section 3

The following auxiliary lemma has been proved in [7].

Lemma 13. *(Lemma 3.1.1 in [7])* $\forall g, h \in G_{\mathcal{A}} : \mathcal{L}(g) \cdot \mathcal{L}(h) \subseteq \mathcal{L}(g; h)$.

We now prove Lemma 2, which states the relationship between graphs in $G^f_{\mathcal{A}}$ and $G^1_{\mathcal{A}}$.

Proof (Lemma 2). Since the languages of the graphs in $G_{\mathcal{A}}$ form a partition of Σ^+, we have that $w \in \mathcal{L}(g)$ and $w \in \mathcal{L}(h)$ iff $\mathcal{L}(g) = \mathcal{L}(h)$ iff $g = h$. Let h be a graph with a non-empty language and let $w = a_1 a_2 a_3 \dots a_n$ be a word in $\mathcal{L}(h)$. The word w is also in the language of the graph $g_{a_1}; g_{a_2}; \dots; g_{a_n}$. Therefore, $h = g_{a_1}; g_{a_2}; \dots; g_{a_n}$. Conversely, assume $h = g_{a_1}; g_{a_2}; \dots; g_{a_n}$ with $a_i \in \Sigma$ for $1 \leq i \leq n$. By Lemma 13, the word $a_1 a_2 a_3 \dots a_n$ is in $\mathcal{L}(h)$. Therefore, $h \in G^f_{\mathcal{A}}$.
□

Before proving Lemma 3, we first prove the lemma below.

Lemma 14. *For $g, h \in G^f_{\mathcal{A}}$, $LFT(g, h)$ iff $Y_{gh} \cap \mathcal{L}(\mathcal{A}) \neq \emptyset$.*

Proof. For the "if" direction, assume that $Y_{gh} \cap \mathcal{L}(\mathcal{A}) \neq \emptyset$. Let uv^ω be a word in $Y_{gh} \cap \mathcal{L}(\mathcal{A})$ s.t. $u \in \mathcal{L}(g)$ and $v \in \mathcal{L}(h)$. Since $uv^\omega \in \mathcal{L}(\mathcal{A})$, we have that there exists an infinite sequence of states $p, q_0, q_1, \dots$ in Q s.t. (1) $p \stackrel{u}{\Longrightarrow} q_0$, $p \in I$ and (2) $q_i \stackrel{v}{\Longrightarrow} q_{i+1}$ for all $i \in \mathbb{N}$ and $q_j \underset{F}{\stackrel{v}{\Longrightarrow}} q_{j+1}$ for infinitely many $j \in \mathbb{N}$. From (1) and $u \in \mathcal{L}(g)$, there exists an arc $\langle p, a_0, q_0 \rangle$ in g for $a_0 \in \{0, 1\}$, $p \in I$. From (2) and $v \in \mathcal{L}(h)$, there exists an infinite sequence of arcs $\langle q_0, a_1, q_1 \rangle, \langle q_1, a_2, q_2 \rangle \dots \langle q_{k-1}, a_k, q_k \rangle \dots$ in h where $a_i \in \{0, 1\}$ for all $i \in \mathbb{N}$, and $a_j = 1$ for infinitely many $j \in \mathbb{N}$. Therefore, we have $LFT(g, h)$.

For the "only if" direction, assume that $LFT(g, h)$. There exists an infinite sequence of states $p, q_0, q_1, \dots$ in Q s.t. (1) the arc $\langle p, a_0, q_0 \rangle$ in g for $a_0 \in \{0, 1\}$, $p \in I$ and (2) the arcs $\langle q_0, a_1, q_1 \rangle, \langle q_1, a_2, q_2 \rangle \dots \langle q_{k-1}, a_k, q_k \rangle \dots$ in h where $a_i \in \{0, 1\}$ for all $i \in \mathbb{N}$, and $a_j = 1$ for infinitely many $j \in \mathbb{N}$. Let uv^ω be any word in

Y_{gh} s.t. $u \in \mathcal{L}(g)$ and $v \in \mathcal{L}(h)$—such a word must exist as $g, h \in G^f_{\mathcal{A}}$. From (1) and (2), we have $p \overset{u}{\Longrightarrow} q_0$ for $p \in I$, $q_i \overset{v}{\Longrightarrow} q_{i+1}$ for all $i \in \mathbb{N}$ and $q_j \overset{v}{\underset{F}{\Longrightarrow}} q_{j+1}$ for infinitely many $j \in \mathbb{N}$. Hence, $uv^\omega \in \mathcal{L}(\mathcal{A})$, and thus we have $Y_{gh} \cap \mathcal{L}(\mathcal{A}) \neq \emptyset$. □

Now, we can prove Lemma 3.

Proof (Lemma 3). We have to show that $\mathcal{A}$ is universal iff for every pair of graphs g, h in $G^f_{\mathcal{A}}$, $LFT(g, h)$ holds. For the "if" direction, assume that $\mathcal{A}$ is not universal. Hence, by Point 3 of Lemma 1, there must be some $g, h \in G^f_{\mathcal{A}}$ s.t. $Y_{gh} \cap \mathcal{L}(\mathcal{A}) = \emptyset$. By Lemma 14, $\neg LFT(g, h)$. For the "only if" direction, assume $\mathcal{A}$ is universal. By Point 3 in Lemma 1, $\forall g, h \in G^f_{\mathcal{A}} : Y_{gh} \cap \mathcal{L}(\mathcal{A}) \neq \emptyset$. By Lemma 14, we have $LFT(g, h)$ for all possible $g, h \in G^f_{\mathcal{A}}$. □

C Correctness of Algorithm 2

Lemma 15. *For any $g \in G^1_{\mathcal{A}}$, there is some $f \in Init$ on Line 7 of Algorithm 2 such that $f \sqsubseteq^{\forall\exists} g$.*

Proof. First note that due to Lemma 4, $Min(g) \sqsubseteq^{\forall\exists} g$ for each $g \in G^1_{\mathcal{A}}$. The graph $Min(g)$ is put into *Next* on Line 1. Then, $Min(g)$ either

- stays in *Next*,
- is moved to *Init* due to Lines 3 and 6,
- is discarded due to Lines 3 and 4 which can, however, happen only if there is already some $\sqsubseteq^{\forall\exists}$-smaller graph in *Init* (which can then only be replaced by even $\sqsubseteq^{\forall\exists}$-smaller graphs on Lines 5 and 6),
- or is removed from *Init* on Line 5 when some $\sqsubseteq^{\forall\exists}$-smaller graph is put into *Init* on Line 6 (which can then in turn be replaced by even $\sqsubseteq^{\forall\exists}$-smaller graphs on Lines 5 and 6).

Since the loop terminates only when $Next = \emptyset$, the lemma clearly holds. □

Lemma 16. *Once any graph f appears in the set Next in between of Lines 8–16 of Algorithm 2, then from this point in the run of the algorithm onwards there is always some $f' \in Next \cup Processed$ such that $f' \sqsubseteq^{\forall\exists} f$.*

Proof. The lemma holds since f can only stay in *Next*, be moved to *Processed* on Line 11, or be removed from *Next* or *Processed* on Line 15 in which case, however, a $\sqsubseteq^{\forall\exists}$-smaller graph is put into *Next* on Line 16 (which can in turn be replaced by further $\sqsubseteq^{\forall\exists}$-smaller graphs on Lines 15 and 16). □

Lemma 17. *All graphs tested on the lasso-finding test within a run of Algorithm 2 are in $G^m_{\mathcal{A}}$.*

Proof. All graphs that participate in the lasso-finding test on Line 10 are either (1) equal to $Min(g)$ for some $g \in G^1_{\mathcal{A}}$, or (2) they are generated on Line 13. Minimizations of graphs from $G^1_{\mathcal{A}}$ are clearly in $G^m_{\mathcal{A}}$. In Case (2), the newly generated graphs are minimizations of compositions of graphs from *Init* and graphs from *Next*. Provided that $Next \subseteq G^m_{\mathcal{A}}$, any graph generated on Line 13 is again in $G^m_{\mathcal{A}}$ by Lemma 8 and because $G^m_{\mathcal{A}}$ is clearly closed under *Min*. Since the initial value of *Next* is a subset of $G^m_{\mathcal{A}}$ (minimizations of graphs from $G^1_{\mathcal{A}}$) and since after initialization, only graphs generated on Line 13 can be added into *Next*, *Next* will always stay a subset of $G^m_{\mathcal{A}}$. Therefore, all graphs f generated on Line 13 and used in the lasso-finding test on Line 10 are always in $G^m_{\mathcal{A}}$. □

Let $Processed^{end}$ denote the contents of the set *Processed* when Algorithm 2 terminates.

Lemma 18. *If Algorithm 2 returns* TRUE*, then for every $g \in G^f_{\mathcal{A}}$, there exists some $h \in Processed^{end}$ such that $h \sqsubseteq^{\forall\exists} g$.*

Proof. By Lemma 2, the graph g is a finite composition $g_0; g_1; \ldots; g_n$ of $n+1$ graphs from $G^1_{\mathcal{A}}$. We now prove by induction that for each $0 \leq i \leq n$, there exists a graph $h_i \in Processed^{end}$ such that $h_i \sqsubseteq^{\forall\exists} g_0; g_1; \ldots; g_i$.

We start with the base case. Due to Lemma 15 and the code on Line 7, there is some $f_0 \in Next$ such that $f_0 \sqsubseteq^{\forall\exists} g_0$ when the second loop is entered for the first time. Using Lemma 16 and the fact that *Next* is empty when Algorithm 2 returns TRUE, the base case is proved.

For the inductive step, we assume that there exists a graph $h_k \in Processed^{end}$ such that $h_k \sqsubseteq^{\forall\exists} g_0; g_1; \ldots; g_k$. Due to Lemma 15, we know that there is $f_{k+1} \in Init$ such that $f_{k+1} \sqsubseteq^{\forall\exists} g_{k+1}$. By Lemmas 6 and 4, $Min(h_k; f_{k+1}) \sqsubseteq^{\forall\exists} g_0; g_1; \ldots; g_k; g_{k+1}$. The graph $Min(h_k; f_{k+1})$ is added into *Next* on Line 16 right after h_k is moved to *Processed* on Line 11 (unless there is already some $\sqsubseteq^{\forall\exists}$-smaller graph in $Processed \cup Next$). From Lemma 16 and from the fact that *Next* is empty when Algorithm 2 returns TRUE, it is then clear that there is some graph $h_{k+1} \in Processed^{end}$ such that $h_{k+1} \sqsubseteq^{\forall\exists} g_0; g_1; \ldots; g_{k+1}$. □

Lemma 19. *Algorithm 2 eventually terminates.*

Proof. The set $G^1_{\mathcal{A}}$ is finite, and hence the set *Next* constructed on Line 1 is finite too. Each iteration of the loop in Lines 2–6 removes one graph from *Next*, and hence the loop must terminate. Moreover, each of the finitely many iterations of the first loop adds at most one graph to *Init*, and hence the set $Next = Init$ constructed on Line 7 is finite (if Line 7 is reached at all). Now, for the loop on Lines 8–16 not to terminate, it would have to be the case that some graph is repeatedly removed from *Next* on Line 11. Hence, this graph would have to be repeatedly added into *Next* too. However, due to Lemma 16, once some graph g appears in *Next* on Line 11, there will always be some graph $h \in Processed \cup Next$ such that $h \sqsubseteq^{\forall\exists} g$. Hence, due to the test on Line 14, g cannot be added to *Next* for the second time. □

Now we are finally able to prove Theorem 1.

Proof (Theorem 1). By Lemma 19, Algorithm 2 eventually terminates and returns a value.

First, we prove by contradiction that if TRUE is returned, $\mathcal{A}$ is universal. Assume that $\mathcal{A}$ is not universal. Then, by Lemma 3, (1) there is either some pair of graphs $g \neq h \in G^f_{\mathcal{A}}$ such that $\neg LFT(g,h)$, or (2) there is a graph $f \in G^f_{\mathcal{A}}$ such that $\neg LFT(f,f)$.

Let us start with Case (1). By Lemma 18, we have that there exist some $g',h' \in \mathit{Processed}^{end}$ such that $g' \sqsubseteq^{\forall\exists} g$ and $h' \sqsubseteq^{\forall\exists} h$. Assume that g' is added to *Processed* after h' (the other case being symmetrical due to the "or" used in the condition on Line 10). When g' is about to be added to *Processed*, the algorithm performs a lasso-finding test on $\langle g',h' \rangle$ on Line 10. Since $\langle g,h \rangle$ fails the lasso-finding test, $g' \sqsubseteq^{\forall\exists} g$, and $h' \sqsubseteq^{\forall\exists} h$, we know by Lemma 7 that also $\neg LFT(g',h')$, and thus the algorithm returns FALSE, which leads to a contradiction.

In Case (2), by Lemma 18, we have that there exists some $f' \in \mathit{Processed}^{end}$ such that $f' \sqsubseteq^{\forall\exists} f$. A lasso-finding test on $\langle f',f' \rangle$ is performed on Line 10, right before moving $\mathbf{f'}$ to *Processed*, which happens on Line 11. Since $\neg LFT(f,f)$, $f' \sqsubseteq^{\forall\exists} f$, we know by Lemma 7 that also $\neg LFT(f',f')$, and thus the algorithm returns FALSE, which again leads to a contradiction.

On the other hand, if $\mathcal{A}$ is universal, due to Lemma 3, each pair of graphs $\langle g,h \rangle$ for $g,h \in G^f_{\mathcal{A}}$ passes the lasso-finding test. At the same time, Lemma 17 guarantees that for every pair of graphs $g',h' \in G_{\mathcal{A}}$ that are subject to the lasso-finding test in the run of Algorithm 2, $g',h' \in G^m_{\mathcal{A}}$. Therefore, there are some $g,h \in G^f_{\mathcal{A}}$ such that $g' \leqslant^* g$ and $h' \leqslant^* h$ which by Lemma 4 gives $g' \simeq^{\forall\exists} g$ and $h' \simeq^{\forall\exists} h$, and hence, $g \sqsubseteq^{\forall\exists} g'$ and $h \sqsubseteq^{\forall\exists} h'$. Since $h \in G^f_{\mathcal{A}} \subseteq G^m_{\mathcal{A}}$, Lemma 7 implies that $LFT(g',h')$. Therefore, the algorithm can only return TRUE in this case. □

D Basic Ramsey-based Language Inclusion Checking

Below, we provide proofs of Lemmas 9 and 10.

Proof (Lemma 9). For Point 1, we have to prove $\mathcal{L}(\mathcal{A}) = \bigcup\{Z_{\mathbf{gh}} \mid Z_{\mathbf{gh}} \text{ is proper}\}$. For the left-to-right inclusion, we proceed as follows. Let w be a word in $\mathcal{L}(\mathcal{A})$. We show that there exist supergraphs $\mathbf{g},\mathbf{h} \in S_{\mathcal{A},\mathcal{B}}$ s.t. $w \in Z_{\mathbf{gh}}$ and $Z_{\mathbf{gh}}$ is proper. From $w \in \mathcal{L}(\mathcal{A})$, there must be a run $p \ldots q \ldots q \ldots$ in $\mathcal{A}$ over w s.t. $p \in I_{\mathcal{A}}$, $q \in F_{\mathcal{A}}$, and q occurs infinitely often. Hence, we have that $w \in \mathcal{L}(p,q)\mathcal{L}(q,q)^\omega$. By Point 1 of Lemma 1, there must be a pair of graphs $g,h \in G_{\mathcal{B}}$ s.t. $w \in \mathcal{L}(g)\mathcal{L}(h)^\omega$. Hence, the argument can be proved by letting $\mathbf{g} = \langle\langle p,q \rangle, g\rangle$ and $\mathbf{h} = \langle\langle q,q \rangle, h\rangle$. We now prove the right-to-left inclusion. For any two supergraphs $\mathbf{g} = \langle\langle p,q \rangle, g\rangle, \mathbf{h} = \langle\langle q,q \rangle, h\rangle \in S_{\mathcal{A},\mathcal{B}}$ s.t. $Z_{\mathbf{gh}}$ is proper, let w be a word in $Z_{\mathbf{gh}}$. We show that $w \in \mathcal{L}(\mathcal{A})$. From the definition of the language of supergraphs and $w \in Z_{\mathbf{gh}}$, we have that $w \in \mathcal{L}(p,q)\mathcal{L}(q,q)^\omega$. Moreover, by the definition of properness, we have $p \in I_{\mathcal{A}}$, $q \in F_{\mathcal{A}}$. Hence, there must be an accepting run $p \ldots q \ldots q \ldots$ in $\mathcal{A}$ over w, and hence, $w \in \mathcal{L}(\mathcal{A})$.

For Point 2, we have to show for all non-empty proper $Z_{\mathbf{gh}}$ that either $Z_{\mathbf{gh}} \cap \mathcal{L}(\mathcal{B}) = \emptyset$ or $Z_{\mathbf{gh}} \subseteq \mathcal{L}(\mathcal{B})$ holds. Let $\mathbf{g} = \langle\langle p,q \rangle, g\rangle, \mathbf{h} = \langle\langle q,q \rangle, h\rangle \in S_{\mathcal{A},\mathcal{B}}$ be two supergraphs s.t. $Z_{\mathbf{gh}}$ is non-empty and proper. From the definition of the languages of supergraphs, we have $Z_{\mathbf{gh}} \subseteq Y_{gh}$. From Point 2 of Lemma 1, either $Y_{gh} \cap \mathcal{L}(\mathcal{B}) = \emptyset$ or $Y_{gh} \subseteq \mathcal{L}(\mathcal{B})$. Therefore, either $Z_{\mathbf{gh}} \cap \mathcal{L}(\mathcal{B}) = \emptyset$ or $Z_{\mathbf{gh}} \subseteq \mathcal{L}(\mathcal{B})$.

Point 3 follows directly from Point 1 and Point 2. □

Proof (Lemma 10). We have to show that $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$ iff $DGT(\mathbf{g}, \mathbf{h})$ for all $\mathbf{g}, \mathbf{h} \in S^f_{\mathcal{A},\mathcal{B}}$. For the "if" direction, assume $\mathcal{L}(\mathcal{A}) \not\subseteq \mathcal{L}(\mathcal{B})$. Hence, by Point 3 of Lemma 9, there must be some $\mathbf{g}$ and $\mathbf{h}$ s.t. $Z_{\mathbf{gh}}$ is proper and $Z_{\mathbf{gh}} \cap \mathcal{L}(\mathcal{B}) = \emptyset$. By the definition of the double-graph test and Lemma 14, we have $\neg DGT(\mathbf{g}, \mathbf{h})$. For the "only if" direction, assume $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$. By Point 3 of Lemma 9, there are no $\mathbf{g}$ and $\mathbf{h}$ s.t. $Z_{\mathbf{gh}}$ is proper, $Z_{\mathbf{gh}} \cap \mathcal{L}(\mathcal{B}) = \emptyset$, and $Z_{\mathbf{gh}} \neq \emptyset$. Therefore, by the definition of the double-graph test and Lemma 14, $DGT(\mathbf{g}, \mathbf{h})$ holds for all possible $\mathbf{g}, \mathbf{h} \in G^f_{\mathcal{A}}$. □

The next lemma is an analogy of Lemma 2, allowing the incremental construction of the set $S^f_{\mathcal{A},\mathcal{B}}$.

Lemma 20. *A supergraph* $\mathbf{g}$ *is in* $S^f_{\mathcal{A},\mathcal{B}}$ *iff* $\exists \mathbf{g}_1, \ldots, \mathbf{g}_n \in S^1_{\mathcal{A},\mathcal{B}} : \ \mathbf{g} = \mathbf{g}_1; \ldots; \mathbf{g}_n$.

Proof. The lemma is a direct consequence of Lemmas 3.4.1 and 3.4.3 stated in [7] (using a slightly different notation). □

A basic language inclusion test algorithm (see Algorithm 4) can be obtained by combining Lemmas 9, 10, and 20.

Algorithm 4: *Ramsey-based Language Inclusion Checking (A Modified Version of The Approach in [8])*

```
Input: A = (Σ, Q_A, I_A, F_A, δ_A) and B = (Σ, Q_B, I_B, F_B, δ_B), the set S^1_{A,B}
Output: TRUE if L(A) ⊆ L(B). Otherwise, FALSE.
Next := G^1_{A,B}; Processed := ∅;
while Next ≠ ∅ do
    Pick and remove a supergraph g from Next;
    if ∃h ∈ Processed : ¬DGT(h, g) ∨ ¬DGT(g, h) ∨ ¬DGT(g, g) then
        return FALSE;
    Add g to Processed;
    foreach h ∈ S^1_{A,B} such that ⟨g, h⟩ is composable and g; h ∉ Processed do
        Add g; h to Next;
return TRUE;
```

E Correctness of Algorithm 3

We first provide a proof of Theorem 2.

Proof (Theorem 2). We show instead that there is a pair of supergraphs in $S^f_{\mathcal{A},\mathcal{B}}$ that fails the relaxed double-graph test iff $\mathcal{L}(\mathcal{A}) \nsubseteq \mathcal{L}(\mathcal{B})$.

First, assume that $\langle \mathbf{f}, \mathbf{g} \rangle$ fails the relaxed double-graph test for some $\mathbf{f}, \mathbf{g} \in S^f_{\mathcal{A},\mathcal{B}}$. Let $Z_{\mathbf{fg}}$ be the ω-regular language $\mathcal{L}(\mathbf{f}) \cdot \mathcal{L}(\mathbf{g})^\omega$. $Z_{\mathbf{fg}}$ is non-empty because $\mathbf{f}, \mathbf{g} \in S^f_{\mathcal{A},\mathcal{B}}$. Furthermore, $Z_{\mathbf{fg}} \subseteq \mathcal{L}(\mathcal{A})$ because $q \succeq_{\mathcal{A}} q_1$, $q_2 \succeq_{\mathcal{A}} q_1$, and $q_2 \in F_{\mathcal{A}}$. Finally, $Z_{\mathbf{fg}} \cap \mathcal{L}(\mathcal{B}) = \emptyset$ because $\langle f, g \rangle$ fails the lasso-finding test (Lemma 14). Thus $\mathcal{L}(\mathcal{A}) \nsubseteq \mathcal{L}(\mathcal{B})$.

If $\mathcal{L}(\mathcal{A}) \nsubseteq \mathcal{L}(\mathcal{B})$, then by Lemma 9 and Lemma 10, there is a pair of supergraphs $\langle \mathbf{f}, \mathbf{g} \rangle \in S^f_{\mathcal{A},\mathcal{B}}$ that fail the (stronger) double graph test, i.e., the double graph test in the sense of Section 5. Therefore, $\langle \mathbf{f}, \mathbf{g} \rangle$ also fails the relaxed double graph test in the sense of the new weaker version of Definition 3. □

The following lemma states the closure of $S^m_{\mathcal{A},\mathcal{B}}$ under composition.

Lemma 21. *For any supergraphs* $\mathbf{e}, \mathbf{f} \in S^m_{\mathcal{A},\mathcal{B}}$ *that are composable,* $\mathbf{e}; \mathbf{f} \in S^m_{\mathcal{A},\mathcal{B}}$.

Proof. Let $\mathbf{e} = \langle \langle p, q \rangle, e \rangle$ and $\mathbf{f} = \langle \langle q, r \rangle, f \rangle$. By the definition of $S^m_{\mathcal{A},\mathcal{B}}$, there are supergraphs $\mathbf{g}, \mathbf{h} \in S^f_{\mathcal{A},\mathcal{B}}$ such that $\mathbf{g} = \langle \langle p, q \rangle, g \rangle$, $\mathbf{h} = \langle \langle q, r \rangle, h \rangle$ where $g, h \in G^f_{\mathcal{B}}$ and $e \leqslant^* g$ and $f \leqslant^* h$. As is shown in the proof of Lemma 8, $e; f \leqslant^* g; h$. By the definition of $S^f_{\mathcal{A},\mathcal{B}}$ and as $\mathbf{g}, \mathbf{h} \in S^f_{\mathcal{A},\mathcal{B}}$, there are words $w_1 \in \mathcal{L}(p, q) \cap \mathcal{L}(g)$ and $w_2 \in \mathcal{L}(q, r) \cap \mathcal{L}(h)$. The word $w_1.w_2$ must also be in $\mathcal{L}(p, r)$ and by Lemma 13 $w_1.w_2 \in \mathcal{L}(g; h)$. This means that $\mathcal{L}(g; h) \cap \mathcal{L}(p, r) \neq \emptyset$ and also that $g; h \in G^f_{\mathcal{B}}$. We can conclude that $\mathbf{g}; \mathbf{h} \in S^f_{\mathcal{A},\mathcal{B}} \subseteq S^m_{\mathcal{A},\mathcal{B}}$. □

Further, we state and prove a lemma about the preservation of the relaxed double-graph test on $\sqsubseteq^{\forall\exists}_S$-bigger supergraphs.

Lemma 22. *Let* $\mathbf{e}$, $\mathbf{f}$, $\mathbf{g}$, $\mathbf{h}$ *be graphs in* $S_{\mathcal{A},\mathcal{B}}$ *such that* $\{\mathbf{f}, \mathbf{h}\} \cap S^m_{\mathcal{A},\mathcal{B}} \neq \emptyset$, $\mathbf{e} \sqsubseteq^{\forall\exists}_S \mathbf{g}$, *and* $\mathbf{f} \sqsubseteq^{\forall\exists}_S \mathbf{h}$. *If* $\langle \mathbf{e}, \mathbf{f} \rangle$ *passes the relaxed double-graph test, then* $\langle \mathbf{g}, \mathbf{h} \rangle$ *passes the relaxed double-graph test too.*

Proof. We may equivalently show that if $\langle \mathbf{g}, \mathbf{h} \rangle$ fails the relaxed double-graph test, then $\langle \mathbf{e}, \mathbf{f} \rangle$ fails the relaxed double-graph test too. Assume that $\langle \mathbf{g}, \mathbf{h} \rangle$ fails the relaxed double-graph test. Then, $\mathbf{g} = \langle \langle p, q_1 \rangle, g \rangle$ and $\mathbf{h} = \langle \langle q_2, q_3 \rangle, h \rangle$ such that $q_1 \succeq_{\mathcal{A}} q_2$, $q_3 \succeq_{\mathcal{A}} q_2$, $p \in I_{\mathcal{A}}$, $q_3 \in F_{\mathcal{A}}$, and $\langle g, h \rangle$ fails the lasso-finding test. Let $\mathbf{e} = \langle \langle p', q'_1 \rangle, e \rangle$ and $\mathbf{f} = \langle \langle q'_2, q'_3 \rangle, f \rangle$. From $\mathbf{e} \sqsubseteq^{\forall\exists}_S \mathbf{g}$ and $\mathbf{f} \sqsubseteq^{\forall\exists}_S \mathbf{h}$, we have that $p' = p$, $q'_1 \succeq_{\mathcal{A}} q_1$, $q'_2 = q_2$, $q'_3 \succeq_{\mathcal{A}} q_3$. Thus $p' \in I_{\mathcal{A}}$ and $q'_3 \in F_{\mathcal{A}}$. Furthermore, $q'_1 \succeq_{\mathcal{A}} q_1 \succeq_{\mathcal{A}} q_2 = q'_2$ and $q'_3 \succeq_{\mathcal{A}} q_3 \succeq_{\mathcal{A}} q_2 = q'_2$. Moreover, due to Lemma 7, $\langle e, f \rangle$ fails the lasso-finding test. Hence, $\langle \mathbf{e}, \mathbf{f} \rangle$ fails the relaxed double-graph test. □

Below, we provide several auxiliary lemmas used in the proof of the correctness of Algorithm 3.

Lemma 23. *For any* $\mathbf{g}, \mathbf{h} \in S_{\mathcal{A},\mathcal{B}}$, *if* $\mathbf{g} \leqslant^*_S \mathbf{h}$, *then* $\mathbf{h} \simeq^{\forall\exists}_S \mathbf{g}$.

Proof. Immediately by Lemma 4 and the definitions of $\sqsubseteq^{\forall\exists}_S$ and $\leqslant^*_S$. □

Lemma 24. *For any* $\mathbf{g} \in S^1_{\mathcal{A},\mathcal{B}}$*, there is some* $\mathbf{f} \in \mathit{Init}$ *on Line 7 of Algorithm 3 such that* $\mathbf{f} \sqsubseteq^{\forall\exists}_S \mathbf{g}$*.*

Proof. The proof is analogous to the proof of Lemma 15, it only uses Lemma 23 instead of Lemma 4. □

Lemma 25. *Once any graph* $\mathbf{f}$ *appears in the set Next in between of Lines 8–16 of Algorithm 3, then from this point in the run of the algorithm onwards there is always some* $\mathbf{f}' \in \mathit{Next} \cup \mathit{Processed}$ *such that* $\mathbf{f}' \sqsubseteq^{\forall\exists}_S \mathbf{f}$*.*

Proof. Analogous to the proof of Lemma 16. □

Lemma 26. *Given a supergraph* $\mathbf{g} = \langle\langle p, q\rangle, g\rangle \in S^1_{\mathcal{A},\mathcal{B}}$ *and a state* $p' \in Q_\mathcal{A}$ *such that* $p' \succeq_\mathcal{A} p$*, then there exists* $\langle\langle p', q'\rangle, g\rangle \in S^1_{\mathcal{A},\mathcal{B}}$ *such that* $q' \succeq_\mathcal{A} q$*.*

Proof. Since $\mathbf{g} \in S^1_{\mathcal{A},\mathcal{B}}$, there is some $a \in \Sigma_\mathcal{A} \cap \mathcal{L}(g)$ such that $p \xrightarrow{a} q$. Since $p' \succeq_\mathcal{A} p$, there must be some $q' \in Q_\mathcal{A}$ such that $p' \xrightarrow{a} q'$ and $q' \succeq_\mathcal{A} q$. We have $\langle p', q'\rangle \in E^1_\mathcal{A}$ and $\langle\langle p', q'\rangle, g\rangle \in S^1_{\mathcal{A},\mathcal{B}}$. □

Let $\mathit{Processed}^{end}$ denote the contents of the set *Processed* when Algorithm 3 terminates.

Lemma 27. *If Algorithm 3 returns* TRUE*, then for every* $\mathbf{g} \in S^f_{\mathcal{A},\mathcal{B}}$*, there exists some* $\mathbf{h} \in \mathit{Processed}^{end}$ *such that* $\mathbf{h} \sqsubseteq^{\forall\exists}_S \mathbf{g}$*.*

Proof. By Lemma 20, the supergraph $\mathbf{g}$ is a finite composition $\mathbf{g}_0; \mathbf{g}_1; \ldots; \mathbf{g}_n$ of $n+1$ supergraphs from $S^1_{\mathcal{A},\mathcal{B}}$. We now prove by induction that for each $0 \leq i \leq n$, there exists a supergraph $\mathbf{h}_i \in \mathit{Processed}^{end}$ such that $\mathbf{h}_i \sqsubseteq^{\forall\exists}_S \mathbf{g}_0; \mathbf{g}_1; \ldots; \mathbf{g}_i$.

We start with the base case. Due to Lemma 24 and the code on Line 7, there is some $\mathbf{f}_0 \in \mathit{Next}$ such that $\mathbf{f}_0 \sqsubseteq^{\forall\exists}_S \mathbf{g}_0$ when the second loop is entered for the first time. Using Lemma 25 and the fact that *Next* is empty when Algorithm 3 returns TRUE, the base case is proved.

For the inductive step, we assume that there exists a graph $\mathbf{h}_k \in \mathit{Processed}^{end}$ such that $\mathbf{h}_k = \langle \overline{h}_k, h_k\rangle = \langle\langle p, q_{\overline{h}_k}\rangle, h_k\rangle \sqsubseteq^{\forall\exists}_S \mathbf{g}_0; \mathbf{g}_1; \ldots; \mathbf{g}_k = \mathbf{g} = \langle \overline{g}, g\rangle = \langle\langle p, q_{\overline{g}}\rangle, g\rangle$ where $q_{\overline{h}_k} \succeq_\mathcal{A} q_{\overline{g}}$ and $h_k \sqsubseteq^{\forall\exists} g$. We further have $\mathbf{g}_{k+1} = \langle \overline{g}_{k+1}, g_{k+1}\rangle = \langle\langle q_{\overline{g}}, q\rangle, g_{k+1}\rangle \in S^1_{\mathcal{A},\mathcal{B}}$. Since $\mathbf{g}_{k+1} \in S^1_{\mathcal{A},\mathcal{B}}$ and $q_{\overline{h}_k} \succeq_\mathcal{A} q_{\overline{g}}$, due to Lemma 26, there must be some $q' \in Q_\mathcal{A}$ such that $q' \succeq_\mathcal{A} q$ and $\mathbf{e}_{k+1} = \langle\langle q_{\overline{h}_k}, q'\rangle, g_{k+1}\rangle \in S^1_{\mathcal{A},\mathcal{B}}$. We have that $\mathbf{e}_{k+1} \trianglelefteq^{\forall\exists} \mathbf{g}_{k+1}$. Due to Lemma 24, we know that there is $\mathbf{f}_{k+1} \in \mathit{Init}$ from Line 7 onwards such that $\mathbf{f}_{k+1} \sqsubseteq^{\forall\exists}_S \mathbf{e}_{k+1}$. $\mathbf{f}_{k+1} \sqsubseteq^{\forall\exists}_S \mathbf{e}_{k+1} \trianglelefteq^{\forall\exists} \mathbf{g}_{k+1}$ implies that $\mathbf{f}_{k+1} \trianglelefteq^{\forall\exists} \mathbf{g}_{k+1}$ (since $\sqsubseteq^{\forall\exists}_S \subseteq \trianglelefteq^{\forall\exists}$ and $\trianglelefteq^{\forall\exists}$ is obviously transitive). Therefore, by Lemma 11, we get $\mathbf{h}_k; \mathbf{f}_{k+1} \sqsubseteq^{\forall\exists}_S \mathbf{g}; \mathbf{g}_{k+1}$. By Lemma 23, we have $\mathit{Min}_S(\mathbf{h}_k; \mathbf{f}_{k+1}) \sqsubseteq^{\forall\exists}_S \mathbf{g}; \mathbf{g}_{k+1}$. The supergraph $\mathit{Min}_S(\mathbf{h}_k; \mathbf{f}_{k+1})$ is added into *Next* on Line 16 right after $\mathbf{h}_k$ is moved to *Processed* on Line 11 (unless there is already some $\sqsubseteq^{\forall\exists}_S$-smaller graph in $\mathit{Processed} \cup \mathit{Next}$). From Lemma 25 and from the fact that *Next* is empty when Algorithm 3 returns TRUE, it is then clear that there is some graph $\mathbf{h}_{k+1} \in \mathit{Processed}^{end}$ such that $\mathbf{h}_{k+1} \sqsubseteq^{\forall\exists}_S \mathbf{g}_0; \mathbf{g}_1; \ldots; \mathbf{g}_{k+1}$. □

Lemma 28. *All supergraphs tested on the relaxed double-graph test within a run of Algorithm 3 are in $S^m_{\mathcal{A},\mathcal{B}}$.*

Proof. The proof is an analogous to the proof of Lemma 17. The class $S^m_{\mathcal{A},\mathcal{B}}$ is closed under composition by Lemma 21 and it is obviously closed under Min_S. The algorithm starts with graphs from $S^m_{\mathcal{A},\mathcal{B}}$ and creates new supergraphs using these two operations only. $\square$

Lemma 29. *Algorithm 3 eventually terminates.*

Proof. Analogous to the proof of Lemma 19. $\square$

Now we are finally able to prove Theorem 3.

Proof (Theorem 3). By Lemma 29, Algorithm 3 eventually terminates and returns a value.

First, we prove by contradiction that if TRUE is returned, $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$ holds. Assume that TRUE is returned and $\mathcal{L}(\mathcal{A}) \not\subseteq \mathcal{L}(\mathcal{B})$. Since $\mathcal{L}(\mathcal{A}) \not\subseteq \mathcal{L}(\mathcal{B})$, Theorem 2 implies that either (1) there is some pair of supergraphs $\mathbf{g} \neq \mathbf{h} \in S^f_{\mathcal{A},\mathcal{B}}$ such that $\langle \mathbf{g}, \mathbf{h} \rangle$ fails the relaxed double-graph test, or (2) there is a supergraph $\mathbf{f} \in S^f_{\mathcal{A},\mathcal{B}}$ such that $\langle \mathbf{f}, \mathbf{f} \rangle$ fails the relaxed double-graph test.

Let us start with Case (1). By Lemma 27, we have that there exist some $\mathbf{g}', \mathbf{h}' \in \mathit{Processed}^{end}$ such that $\mathbf{g}' \sqsubseteq^{\forall\exists}_S \mathbf{g}$ and $\mathbf{h}' \sqsubseteq^{\forall\exists}_S \mathbf{h}$. Assume that $\mathbf{g}'$ is added to *Processed* after $\mathbf{h}'$ (the other case being symmetrical due to the "or" used in the condition on Line 10). When $\mathbf{g}'$ is about to be added to *Processed*, the algorithm performs a relaxed double-graph test on $\langle \mathbf{g}', \mathbf{h}' \rangle$ on Line 10. Since $\langle \mathbf{g}, \mathbf{h} \rangle$ fails the relaxed double-graph test, $\mathbf{g}' \sqsubseteq^{\forall\exists}_S \mathbf{g}$, and $\mathbf{h}' \sqsubseteq^{\forall\exists}_S \mathbf{h}$, we know by Lemma 22 that $\langle \mathbf{g}', \mathbf{h}' \rangle$ also fails the relaxed double-graph test, and thus the algorithm returns FALSE, which leads to a contradiction.

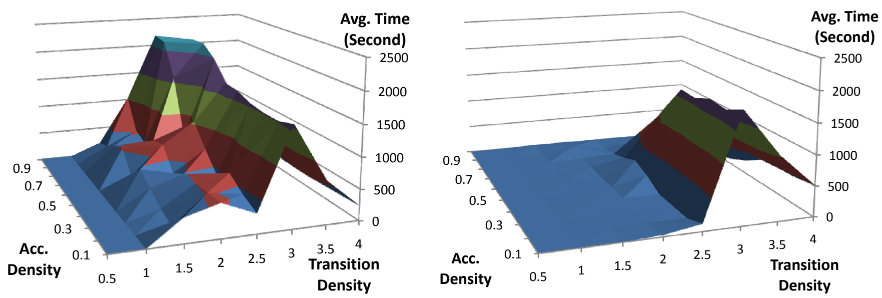
In Case (2), by Lemma 27, we have that there exists some $\mathbf{f}' \in \mathit{Processed}^{end}$ such that $\mathbf{f}' \sqsubseteq^{\forall\exists}_S \mathbf{f}$. A relaxed double-graph test on $\langle \mathbf{f}', \mathbf{f}' \rangle$ is performed on Line 10, right before moving $\mathbf{f}'$ to *Processed*, which happens on Line 11. Since $\langle \mathbf{f}, \mathbf{f} \rangle$ fails the relaxed double-graph test, $\mathbf{f}' \sqsubseteq^{\forall\exists}_S \mathbf{f}$, we know by Lemma 22 that $\langle \mathbf{f}', \mathbf{f}' \rangle$ also fails the relaxed double-graph test, and thus the algorithm returns FALSE, which again leads to a contradiction.

On the other hand, if $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$, then due to Theorem 2, each pair of supergraphs $\langle \mathbf{g}, \mathbf{h} \rangle$ for any $\mathbf{g}, \mathbf{h} \in S^f_{\mathcal{A},\mathcal{B}}$ passes the relaxed double-graph test. At the same time, Lemma 28 guarantees that any two supergraphs $\mathbf{g}', \mathbf{h}'$ that are are subject to the relaxed double-graph test in the run of Algorithm 3 are from $S^m_{\mathcal{A},\mathcal{B}}$. By the definition of $S^m_{\mathcal{A},\mathcal{B}}$, there are $\mathbf{g}, \mathbf{h} \in S^f_{\mathcal{A},\mathcal{B}}$ with $\mathbf{g}' \leqslant^*_S \mathbf{g}$ and $\mathbf{h}' \leqslant^*_S \mathbf{h}$, which by Lemma 23 gives $\mathbf{g} \sqsubseteq^{\forall\exists}_S \mathbf{g}'$ and $\mathbf{h} \sqsubseteq^{\forall\exists}_S \mathbf{h}'$. Lemma 22 implies that $\langle \mathbf{g}', \mathbf{h}' \rangle$ passes the relaxed double-graph test. Therefore, the algorithm can only return TRUE in this case. $\square$

F Additional Results on the Experiments

Experiment	Description of Machines Used
Tabakov and Vardi's (Universality)	A machine with 8 Intel(R) Xeon(R) 2.0GHz processors. Each instance of the experiment has a dedicated CPU core with 2GB memory.
Random LTL (Universality)	A machine with 2 Intel(R) Xeon(R) 2.66GHz processors. Each instance of the experiment has a dedicated CPU core with 2GB memory.
Mutual Exclusion Algorithms (Inclusion)	A machine with 8 Intel(R) Xeon(R) 2.0GHz processors (MCS queue lock, Dining phil. Ver.1) and a machine with 8 AMD Opteron(tm) 2.8GHz processors (other experiments). Each instance of the experiment has a dedicated CPU core with 8GB memory.
Random LTL (Inclusion)	A machine with 2 Intel(R) Xeon(R) 2.66GHz processors. Each instance of the experiment has a dedicated CPU core with 2GB memory.

Table 2. Machines we used in the experiments.



(a) Subsumption (b) Simulation Subsumption

Fig. 4. Average time for universality checking on Tabakov and Vardi's random model. Each point in the figure is the average result of 100 Büchi automata. If both methods cannot finish the task within the timeout period (3500 seconds), we simply ignore this test while computing the average result. If only one method finishes the task within the timeout period while the other does not, we use 3500 seconds as the execution time of the failed task. From the figure, one can see that our simulation subsumption technique outperforms the basic subsumption in most of the cases.

	Original		Error		Subsumption		Simulation Sub.	
	Tr.	St.	Tr.	St.	$\mathcal{L}(E)\subseteq\mathcal{L}(O)$	$\mathcal{L}(O)\subseteq\mathcal{L}(E)$	$\mathcal{L}(E)\subseteq\mathcal{L}(O)$	$\mathcal{L}(O)\subseteq\mathcal{L}(E)$
Bakery	2697	1506	2085	1146	>1day	>1day	2m47s(F)	20m(F)
Peterson	34	20	33	20	2.1s(T)	0.17s(F)	0.43s(T)	0.08s(F)
Dining phil. (Ver.1)	212	80	464	161	1m18s(F)	>1day	3.7s(F)	>1day
Dining phil. (Ver.2)	212	80	482	161	3m1s(F)	>1day	5s(F)	>1day
Fisher	1395	634	3850	1532	>1day	>1day	14h4m(F)	>1day
MCS queue lock	21503	7963	3222	1408	oom	3m10s(F)	25m55s(T)	1m39s(F)

Table 3. Language inclusion checking on models of mutual exclusion algorithms (*where all states are accepting*). The rows "Original" and "Error" are the statistical data of the BA of the original model and the erroneous model, respectively. We test both $\mathcal{L}(E) \subseteq \mathcal{L}(O)$ and $\mathcal{L}(O) \subseteq \mathcal{L}(E)$. The result ">1day" indicates that the task cannot be finished within one day, and "oom" that the task required memory > 8GB. If a task completes successfully, we record the run time and whether language inclusion holds ("T") or not ("F").