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 [10,9]. It was shown in [10] (for universality checking) that a simple subsumption technique, which avoids exploration of 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 [10].

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 one is generally interested in non-halting computations, automata are used 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,

* This work was supported in part by the Royal Society grant JP080268, the Czech Science Foundation (projects P103/10/0306 and 102/09/H042), the Czech Ministry of Education (projects COST OC10009 and MSM 0021630528), the internal BUT FIT grant FIT-10-1, the UPMARC project, the CONNECT project, National Science Council of Taiwan project no. 99-2218-E-001-002-MY3, and the ESF project Games for Design and Verification.

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, *Rank-based* and *Ramsey-based* methods have recently gained interest. The former uses a rank-based analysis of rejecting runs [14], 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 [4]. 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, [5] described a simple set-inclusion-based subsumption technique that speeds up the rank-based method for both universality and language inclusion checking. Using this technique, [5] is capable of handling automata several orders of magnitude larger than previously possible. Similarly, [10] 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 paragraph in Section 3).

Our contribution. We improve the Ramsey-based approach in a twofold way. First, we show how to employ simulation preorder to generalize the simple subsumption technique of [10] for Ramsey-based universality checking. Second, 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. Note that the proposed use of simulations is significantly different from just using simulations to reduce Büchi automata (quotienting), followed by an application of the original approach. The reason is that, when reducing automata, one has to use simulation equivalences which tend to be much smaller (and hence less helpful) than simulation preorders, which are used in our method. Therefore, our approach takes full advantage of the asymmetry of simulation preorder, making it more general than just quotienting beforehand the automaton w.r.t. the induced equivalence.

Experimental results show that our algorithm based on simulation subsumption significantly and consistently outperforms the algorithm based on the original subsumption of [10]. We perform the evaluation on Büchi automata models of several mutual exclusion algorithms (the largest having 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 [18]. 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 [21], 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) $p \xRightarrow[F]{w} q$ iff there is an accepting path on w from p to q. (2) $p \xRightarrow{w} q$ 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 every $p, q \in Q$. 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) $\mathcal{L}(p, 1, q) = \{w \in \Sigma^+ \mid p \xRightarrow[F]{w} q\}$, (2) $\mathcal{L}(p, 0, q) = \{w \in \Sigma^+ \mid p \xRightarrow{w} q \wedge \neg\left(p \xRightarrow[F]{w} q\right)\}$, (3) $\mathcal{L}(p, -1, q) = \{w \in \Sigma^+ \mid p \overset{w}{\not\Longrightarrow} q\}$. As in [10], the language of a graph $g \in G_{\mathcal{A}}$ is defined as the intersection of the languages of arcs in g, i.e., $\mathcal{L}(g) = \bigcap_{\langle p,a,q\rangle \in g} \mathcal{L}(p, a, q)$. For each word $w \in \Sigma^+$ and each pair $p, q \in Q$, there exists exactly one arc $\langle p, a, q\rangle$ such that $w \in \mathcal{L}(p, a, q)$. Therefore, the languages of the graphs in $G_{\mathcal{A}}$ form a partition of Σ^+, since they are the intersections of the languages of the arcs. Let Y_{gh} 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 [16,9,10]. A proof can be found in [1].

3 Ramsey-Based Universality Testing

Based on Lemma 1, one can construct an algorithm for checking universality of BA [9]. 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 [9,10,13], 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 $g_a = \{\langle p,-1,q\rangle \mid q \notin \delta(p,a)\} \cup \{\langle p,0,q\rangle \mid p \in (Q \backslash F) \wedge q \in (\delta(p,a) \backslash F)\} \cup \{\langle p,1,q\rangle \mid q \in \delta(p,a) \wedge \{p,q\} \cap F \neq \emptyset\}$. Let $G^1_{\mathcal{A}} = \{g_a \mid a \in \Sigma\}$. As shown in [8] (Lemma 3.1.1), 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 [10], can be used for this purpose. The lasso-finding test of a pair of graphs $\langle g,h\rangle$ checks the existence of a *lasso*, i.e., a path in g from some state $p \in I$ to some state $q \in F$ and a path in h from q to itself. Equivalently, we consider a pair of graphs $\langle g,h\rangle$ to pass the *lasso-finding test* (denoted by $LFT(g,h)$) 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 $i \geq 0$, and $a_j = 1$ for infinitely many $j \in \mathbb{N}$. The following lemma was proved in [10] (we provide a considerably simplified proof in [1]).

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-SCC, (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 [10], 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.

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

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 [10] 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 $\mathcal{O}(|\Sigma||Q||\delta|)$ [11,12]. 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 [7]) quotienting w.r.t. $\simeq$ preserves the ω-regular language of $\mathcal{A}$.

Our method for accelerating the Ramsey-based universality test [10] of $\mathcal{A}$ is based on two optimizations which we describe below together with some intuition underlying their correctness. We formally prove the correctness of these optimizations in Lemmas 4–8, presented afterwards.

Optimization 1. The first optimization is based on the observation that the subsumption relation $\sqsubseteq$ from [10] can be weakened by exploiting simulation preorder. We call our weaker notion the *simulation-subsumption-based relation*, written $\sqsubseteq^{\forall\exists}$. The idea is as follows: While in $\sqsubseteq$ one requires that arcs of the form $\langle p, a, q\rangle$ can only be subsumed by arcs of the form $\langle p, a', q'\rangle$ with $a \leq a' \wedge q' = q$, we generalize this notion by replacing the last equality with simulation. This gives rise to the definition of $\sqsubseteq^{\forall\exists}$ below.

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 $q \preceq q'$.*

Optimization 2. The second optimization builds on the fact 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 complexity of the lasso-finding test, subsumption-checking, and graph-composition is polynomial in the number of $\{0,1\}$-arcs, having smaller graphs reduces the cost of these operations.

For the purpose of reducing graphs, we define a (possibly non-deterministic) operation *Min* that maps each graph $f \in G_{\mathcal{A}}$ to a graph $Min(f) \in G_{\mathcal{A}}$ such that $Min(f) \leqslant^* f$. 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 subsumed (simulation-smaller) by other arcs present both in g and h. We define $\leqslant^*$ below.

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 $\leqslant$.*

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).

Correctness of the Optimizations. We now prove the correctness of our optimizations in a formal way. The correctness of the second optimization follows directly from 1) the observation that $\leqslant$ implies $\sqsubseteq^{\forall\exists}$-equivalence (Lemma 4), and 2) the fact that $G^m_{\mathcal{A}}$ is closed under composition (Lemma 8).

Let $\simeq^{\forall\exists} = \sqsubseteq^{\forall\exists} \cap (\sqsubseteq^{\forall\exists})^{-1}$. The lemma below follows by transitivity.

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

In particular, minimized graphs are $\sqsubseteq^{\forall\exists}$-equivalent to their original version. Notice that the relation $\leqslant^*$ does not preserve the language of graphs (and often for $g \leqslant^* h$, $\mathcal{L}(g) = \emptyset$ when $\mathcal{L}(h) \neq \emptyset$).

The correctness of the first optimization follows from the following observations. First, the lasso-finding test is $\sqsubseteq^{\forall\exists}$-monotonic, i.e., if $\sqsubseteq^{\forall\exists}$-smaller (pairs of) graphs pass the test, then so do $\sqsubseteq^{\forall\exists}$-bigger (pairs of) graphs (Lemma 7). In particular, for graphs $f, g, h \in G^f_{\mathcal{A}}$ such that $g \sqsubseteq^{\forall\exists} h$, $LFT(g, f) \implies LFT(h, f)$ and $LFT(f, g) \implies LFT(f, h)$. Therefore, we can ignore all lasso-finding tests related to the bigger h. Second, graph-composition is also $\sqsubseteq^{\forall\exists}$-monotonic (Lemma 6): Composing $\sqsubseteq^{\forall\exists}$-smaller graphs always yields $\sqsubseteq^{\forall\exists}$-smaller graphs. Thus, we can also ignore all lasso-finding tests related to any extension $h; f$ of h, for some $f \in G^f_{\mathcal{A}}$.

We begin by proving an auxiliary lemma—used to prove Lemma 6—which says that minimized graphs are in some sense complete w.r.t. simulation-bigger states.

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 $\exists \langle p', a', q'\rangle \in g :\ a \leq a' \ \wedge\ q \preceq q'$.*

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 $a \leq a''$. From Lemma 4, we get that there is an arc $\langle p', a', q'\rangle \in g$ such that $a \leq a'' \leq a'$ and $q \preceq q'' \preceq q'$. □

The lemma below states that composing minimized graphs is a $\sqsubseteq^{\forall\exists}$-monotonic operation. We actually prove a slightly stronger property, since we do not require that all graphs are minimal.

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 $a \leq a'$ and $q \preceq q'$. Since $g \sqsubseteq^{\forall\exists} g'$, there is an arc $\langle q, b', r'\rangle \in g'$ with $b \leq b'$ 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. $b \leq b' \leq b''$ and $r \preceq r' \preceq r''$. Thus $\langle p, c'', r''\rangle \in f'; g'$ where $c = \max(\{a, b\}) \leq \max(\{a', b''\}) \leq c''$ 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. For the sake of generality, we prove a somewhat stronger property.

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 $\langle q_0, a_1, q_1\rangle$, $\langle q_1, a_2, q_2\rangle$, ... 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. $a_0 \leq a'_0$ 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): $\langle q'_n, a'_{n+1}, q'_{n+1}\rangle \in h$, $a_{n+1} \leq a'_{n+1}$, and $q_n \preceq q'_n$ for all $n \geq 0$. We do this by proving that every finite sequence $q'_0 a'_1 q'_1 \ldots q'_{k-1} a'_k q'_k$ 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 $\langle q'_0, a'_1, q'_1\rangle$, $\langle q'_1, a'_2, q'_2\rangle$, ... 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 $e; f \leqslant^* g; h$ (notice that this does not directly follow from Lemma 6, since $\sqsubseteq^{\forall\exists}$ does not imply $\leqslant^*$). 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 $g_0, h_0, g_1, h_1, \ldots, g_n, h_n \in G^m_{\mathcal{A}}$ 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 gives a complete description of our approach to universality testing of BA. 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 [1] using the closure of $G^m_{\mathcal{A}}$ under composition stated in Lemma 8, the monotonicity from Lemma 6, and the preservation of lasso-finding tests from Lemma 7.

Theorem 1. *Algorithm 2 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 [9] before explaining how to extend our approach from universality to language inclusion checking. Define the set $E_{\mathcal{A}} = Q_{\mathcal{A}} \times Q_{\mathcal{A}}$.

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.
1  $Next := \{Min(g) \mid g \in G^1_{\mathcal{A}}\}$; $Init := \emptyset$;
2  while $Next \neq \emptyset$ do
3      Pick and remove a graph $g$ from $Next$;
4      if $\exists f \in Init : f \sqsubseteq^{\forall\exists} g$ then Discard $g$ and continue;
5      Remove all graphs $f$ from $Init$ s.t. $g \sqsubseteq^{\forall\exists} f$;
6      Add $g$ into $Init$;
7  $Next := Init$; $Processed := \emptyset$;
8  while $Next \neq \emptyset$ do
9      Pick a graph $g$ from $Next$;
10     if $\neg LFT(g, g) \vee \exists h \in Processed : \neg LFT(h, g) \vee \neg LFT(g, h)$ then
           return FALSE;
11     Remove $g$ from $Next$ and add it to $Processed$;
12     foreach $h \in Init$ do
13         $f = Min(g; h)$;
14         if $\exists k \in Processed \cup Next : k \sqsubseteq^{\forall\exists} f$ then Discard $f$ and continue;
15         Remove all graphs $k$ from $Processed \cup Next$ s.t. $f \sqsubseteq^{\forall\exists} k$;
16         Add $f$ into $Next$;
17 return TRUE;
```

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 $\mathcal{L}(p, q) = \{w \in \Sigma^+ \mid p \overset{w}{\Longrightarrow} q\}$.

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 $\overline{h} = \langle q, q\rangle$ for some $p \in I_{\mathcal{A}}$ 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 [9] (the constraints of being a proper $Z_{\mathbf{gh}}$ are weaker than those in [9]).

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 for all $\mathbf{g}, \mathbf{h} \in S_{\mathcal{A},\mathcal{B}}$, 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 $Z_{\mathbf{gh}} \subseteq \mathcal{L}(\mathcal{B})$, it is sufficient to build and check only supergraphs with nonempty languages (whose set we denote $S^f_{\mathcal{A},\mathcal{B}}$) when checking language inclusion.

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 $q_{\mathbf{g}} = p_{\mathbf{h}}$. Then, their composition $\mathbf{g}; \mathbf{h}$ is defined as $\langle\langle p_{\mathbf{g}}, q_{\mathbf{h}}\rangle, g; h\rangle$. 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 [9]. 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 $LFT(g, h)$. The following lemma is due to [9] .

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. [1]).

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) $LFT(g, h)$.*

The following Lemma 11 is needed to prove the central Theorem 2.

Lemma 11. *Every weakly proper $Z_{\mathbf{gh}}$ is contained in $\mathcal{L}(\mathcal{A})$.*

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 $S^m_{\mathcal{A},\mathcal{B}} = \{\mathbf{g} \in S_{\mathcal{A},\mathcal{B}} \mid \exists \mathbf{h} \in S^f_{\mathcal{A},\mathcal{B}} : \mathbf{g} \leqslant^*_S \mathbf{h}\}$. $Min_S(\mathbf{g})$ 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 as well as preservation of the double-graph test on $\sqsubseteq^{\forall\exists}_S$-larger supergraphs (cf. [1]). What slightly differs is the monotonicity of the composition, which is caused by the additional composability requirement. To cope with it, we define a new relation $\trianglelefteq^{\forall\exists}$, weakening $\sqsubseteq^{\forall\exists}_S$: 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 $\sqsubseteq^{\forall\exists}_S$ indeed implies $\trianglelefteq^{\forall\exists}$. From the definitions of $\sqsubseteq^{\forall\exists}_S$, $\trianglelefteq^{\forall\exists}$, and Lemma 6, we then easily get the following relaxed monotonicity lemma.

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.
1  $Next := \{Min_S(\mathbf{g}) \mid \mathbf{g} \in S^1_{\mathcal{A},\mathcal{B}}\}$; $Init := \emptyset$;
2  while $Next \neq \emptyset$ do
3      Pick and remove a supergraph $\mathbf{g}$ from $Next$;
4      if $\exists \mathbf{f} \in Init : \mathbf{f} \sqsubseteq_S^{\forall\exists} \mathbf{g}$ then Discard $\mathbf{g}$ and continue;
5      Remove all supergraphs $\mathbf{f}$ from $Init$ s.t. $\mathbf{g} \sqsubseteq_S^{\forall\exists} \mathbf{f}$;
6      Add $\mathbf{g}$ into $Init$;
7  $Next := Init$; $Processed := \emptyset$;
8  while $Next \neq \emptyset$ do
9      Pick a supergraph $\mathbf{g}$ from $Next$;
10     if $\neg RDGT(\mathbf{g},\mathbf{g}) \vee \exists \mathbf{h} \in Processed : \neg RDGT(\mathbf{h},\mathbf{g}) \vee \neg RDGT(\mathbf{g},\mathbf{h})$ then
           return FALSE;
11     Remove $\mathbf{g}$ from $Next$ and add it to $Processed$;
12     foreach $\mathbf{h} \in Init$ where $\langle \mathbf{g},\mathbf{h} \rangle$ are composable do
13         $\mathbf{f} := Min_S(\mathbf{g};\mathbf{h})$;
14         if $\exists \mathbf{k} \in Processed \cup Next : \mathbf{k} \sqsubseteq_S^{\forall\exists} \mathbf{f}$ then Discard $\mathbf{f}$ and continue;
15         Remove all supergraphs $\mathbf{k}$ from $Processed \cup Next$ s.t. $\mathbf{f} \sqsubseteq_S^{\forall\exists} \mathbf{k}$;
16         Add $\mathbf{f}$ into $Next$;
17 return TRUE;
```

Lemma 12. *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 we show that it is safe to work with $\sqsubseteq_S^{\forall\exists}$-smaller supergraphs in the incremental supergraph construction. 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 a supergraph $\mathbf{f}$ satisfying the preconditions of Lemma 12—excluding the 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.

Algorithm 3 is our simulation-optimized algorithm for BA inclusion-checking. Its correctness is stated below and proved in [1].

Theorem 3. *Algorithm 3 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 [9,10] in Java. For universality testing, we

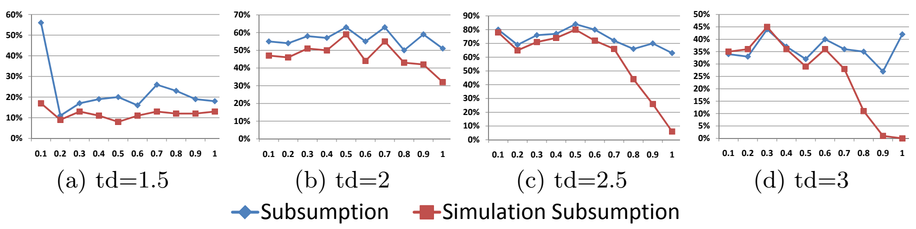

Fig. 1. Timeout cases of universality checking on Tabakov and Vardi's random model. The vertical axis is the percentage of tasks that cannot be finished within the timeout period and the horizontal axis is the acceptance density (ad).

compared our algorithm to the one in [10].[1] For language inclusion testing, we compared our simulation subsumption algorithm to a restricted version that uses the simple subsumption relation of [10]. (The language inclusion checking algorithm described in [9] does not use any subsumption at all and performed much worse.) We refer interested readers to [21] for all relevant materials needed to reproduce the results. A description of the machines that we used is given in [1].

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 (i.e. the average 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 $ad = 0.1, 0.2, \ldots, 1.0$, and generate 100 random BA for each combination of td and ad. In Figure 1, we compare the number of timeouts between the two approaches when the number of states is 100 and the timeout period is 3500 seconds[2]. 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. For the two most difficult configurations ($td = 2.5, 3.0$), the difference between the two approaches gets larger as ad increases. The trend shown in Figure 1 stayed the same when the number of states increases. We refer the interested readers to the Tech. Report [1] for results of automata with the number of states ranging from 10 to 200.

[1] The algorithm in [10] uses the simple subsumption relation $\sqsubseteq$, and also a different search strategy than our algorithm. To take into account the latter, we have also included a combination of our search strategy with the simple subsumption into our experiments. We evaluated the new search strategy with random automata of size 20. While our search strategy with the simple subsumption $\sqsubseteq$ is already on average about 20% faster than [10], the main improvement we witness in our experiments is achieved by using the simulation subsumption $\sqsubseteq^{\forall\exists}$.

[2] Our approach can be slightly slower than [9] in some rare cases, due to the overhead of computing simulation. For those cases, the simulation relation is sparse and does not yield any speedup. However, since there are efficient algorithms for computing simulation, the relative overhead is quite small on difficult instances.

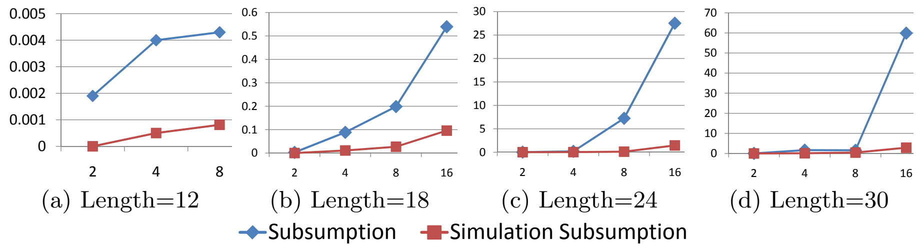

Fig. 2. Universality checking on automata generated from random LTL. Each point in the figure is the average result of 100 different instances. The horizontal axis is the size of the alphabet. The vertical axis is the average execution time in seconds.

Table 1. Language inclusion checking on mutual exclusion algorithms. The columns "Original" and "Error" refer to the 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 the fact whether language inclusion holds or not ("T" or "F", respectively).

	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	32m15s(F)	49m57s
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	11m52s(F)	>1day
Dining phil. Ver.2	212	80	482	161	>1day	>1day	14m40s(F)	>1day
Fisher Ver.1	1395	634	3850	1532	>1day	oom	1m23s(F)	>1day
Fisher Ver.2	1395	634	1420	643	>1day	>1day	8s(T)	8s(T)
MCS queue lock	21503	7963	3222	1408	>1day	>1day	1h36m(T)	6m22s(F)

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 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 not have been 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[3]. 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 [15][4], translate both the original and the modified version into

[3] We do not have formulae having length 12 and 4 propositions because our generator [19] requires that (*length of formula*/3) > *number of propositions*.

[4] The models in [15] are based on guarded commands. We modify the models by randomly weakening or strengthening the guard of some commands.

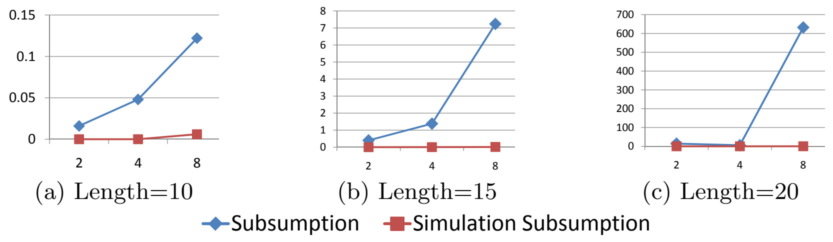

Fig. 3. Language inclusion checking on automata generated from random LTL. Each point in the figure is the average result of 10 different instances. The horizontal axis is the size of the alphabet. The vertical axis is the average execution time in seconds.

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 [1]. In both versions, our approach has significantly and consistently better performance than the basic subsumption approach.

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 having length 10, 15, and 20 and 1–3 propositions (which corresponds to BA of alphabet sizes 2, 4, and 8). For each length of formula and each number of propositions, we generate 10 pairs of BA. The relative results are shown in Figure 3. The difference in performance of using the basic subsumption and the simulation subsumption is again quite significant here. For the largest cases (with formulae having 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 evaluated our approach by a wide set of experiments, showing that the simulation-subsumption approach consistently outperforms the basic subsumption of [10].

Our techniques can be extended in several ways. Weaker simulations for BA have been defined in the literature, e.g., delayed/fair simulation [7], or their multipebble extensions [6]. 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 being *weakly proper* in Section 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 [3].

References

1. Abdulla, P.A., Chen, Y.-F., Clemente, L., Holík, L., Hong, C.-D., Mayr, R., Vojnar, T.: Simulation Subsumption in Ramsey-based Büchi Automata Universality and Inclusion Testing. Technical report FIT-TR-2010-02, FIT BUT (2010), `http://www.fit.vutbr.cz/~holik/pub/FIT-TR-2010-002.pdf`
2. Abdulla, P.A., Chen, Y.-F., Holík, L., Mayr, R., Vojnar, T.: When Simulation Meets Antichains (On Checking Language Inclusion of Nondeterministic Finite (Tree) Automata). In: Esparza, J., Majumdar, R. (eds.) TACAS 2010. LNCS, vol. 6015, pp. 158–174. Springer, Heidelberg (2010)
3. Abdulla, P.A., Chen, Y.-F., Holík, L., Vojnar, T.: Mediating for Reduction (On Minimizing Alternating Büchi Automata). In: Proc. of FSTTCS'09, Leibniz International Proceedings in Informatics, vol. 4 (2009)
4. Büchi, J.R.: On a Decision Method in Restricted Second Order Arithmetic. In: Proc. of Int. Con. on Logic, Method, and Phil. of Science (1962)
5. Doyen, L., Raskin, J.-F.: Improved Algorithms for the Automata-based Approach to Model Checking. In: Grumberg, O., Huth, M. (eds.) TACAS 2007. LNCS, vol. 4424, pp. 451–465. Springer, Heidelberg (2007)
6. Etessami, K.: A Hierarchy of Polynomial-Time Computable Simulations for Automata. In: Brim, L., Jančar, P., Křetínský, M., Kucera, A. (eds.) CONCUR 2002. LNCS, vol. 2421, p. 131. Springer, Heidelberg (2002)
7. Etessami, K., Wilke, T., Schuller, R.A.: Fair Simulation Relations, Parity Games, and State Space Reduction for Büchi Automata. SIAM J. Comp. 34(5) (2005)
8. Fogarty, S.: Büchi Containment and Size-Change Termination. Master's Thesis, Rice University (2008)
9. Fogarty, S., Vardi, M.Y.: Büchi Complementation and Size-Change Termination. In: Proc. of TACAS'09. LNCS, vol. 5505. Springer, Heidelberg (2009)
10. 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)
11. Henzinger, M.R., Henzinger, T.A., Kopke, P.W.: Computing Simulations on Finite and Infinite Graphs. In: Proc. FOCS'95. IEEE CS, Los Alamitos (1995)
12. Holík, L., Šimáček, J.: Optimizing an LTS-Simulation Algorithm. In: Proc. of MEMICS'09 (2009)
13. Jones, N.D., Lee, C.S., Ben-Amram, A.M.: The Size-Change Principle for Program Termination. In: Proc. of POPL'01. ACM SIGPLAN (2001)
14. Kupferman, O., Vardi, M.Y.: Weak Alternating Automata Are Not That Weak. ACM Transactions on Computational Logic 2(2), 408–429 (2001)

15. Pelánek, R.: BEEM: Benchmarks for Explicit Model Checkers. In: Bošnački, D., Edelkamp, S. (eds.) SPIN 2007. LNCS, vol. 4595, pp. 263–267. Springer, Heidelberg (2007)
16. Sistla, A.P., Vardi, M.Y., Wolper, P.: The Complementation Problem for Büchi Automata with Applications to Temporal Logic. In: Brauer, W. (ed.) ICALP 1985. LNCS, vol. 194. Springer, Heidelberg (1985)
17. Somenzi, F., Bloem, R.: Efficient Büchi Automata from LTL Formulae. In: Emerson, E.A., Sistla, A.P. (eds.) CAV 2000. LNCS, vol. 1855. Springer, Heidelberg (2000)
18. Tabakov, D., Vardi, M.Y.: Model Checking Büchi Specifications. In: Proc. of LATA'07 (2007)
19. Tsay, Y.-K., Chen, Y.-F., Tsai, M.-H., Wu, K.-N., Chan, W.-C.: GOAL: A Graphical Tool for Manipulating Büchi Automata and Temporal Formulae. In: Grumberg, O., Huth, M. (eds.) TACAS 2007. LNCS, vol. 4424, pp. 466–471. Springer, Heidelberg (2007)
20. Wulf, M.D., 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)
21. `http://iis.sinica.edu.tw/FMLAB/CAV2010`
(capitalize "FMLAB" and "CAV 2010")