Advanced Ramsey-based Büchi Automata Inclusion Testing

Parosh Aziz Abdulla[1], Yu-Fang Chen[2], Lorenzo Clemente[3], Lukáš Holík[1,4], 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. Checking language inclusion between two nondeterministic Büchi automata $\mathcal{A}$ and $\mathcal{B}$ is computationally hard (PSPACE-complete). However, several approaches which are efficient in many practical cases have been proposed. We build on one of these, which is known as the *Ramsey-based approach*. It has recently been shown that the basic Ramsey-based approach can be drastically optimized by using powerful subsumption techniques, which allow one to prune the search-space when looking for counterexamples to inclusion. While previous works only used subsumption based on set inclusion or forward simulation on $\mathcal{A}$ and $\mathcal{B}$, we propose the following new techniques: (1) A larger subsumption relation based on a combination of backward and forward simulations on $\mathcal{A}$ and $\mathcal{B}$. (2) A method to additionally use forward simulation *between* $\mathcal{A}$ and $\mathcal{B}$. (3) Abstraction techniques that can speed up the computation and lead to early detection of counterexamples. The new algorithm was implemented and tested on automata derived from real-world model checking benchmarks, and on the Tabakov-Vardi random model, thus showing the usefulness of the proposed techniques.

1 Introduction

Checking inclusion between finite-state models is a central problem in automata theory. First, it is an intriguing theoretical problem. Second, it has many practical applications. For example, in the automata-based approach to model-checking [18], both the system and the specification are represented as finite-state automata, and the model-checking problem reduces to testing whether any behavior of the system is allowed by the specification, i.e., to a language inclusion problem.

We consider language inclusion for Büchi automata (BA), i.e., automata over infinite words. While checking language inclusion between nondeterministic BA is computationally hard (PSPACE-complete [12]), much effort has been devoted to devising approaches that can solve as many practical cases as possible. A naïve approach to language inclusion between BA $\mathcal{A}$ and $\mathcal{B}$ would first complement the latter into a BA $\mathcal{B}^c$, and then check emptiness of $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B}^c)$. The problem is that $\mathcal{B}^c$ is in general exponentially larger than $\mathcal{B}$. Yet, one can determine whether $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B}^c) \neq \emptyset$ by only looking at some "small" portion of $\mathcal{B}^c$. The *Ramsey-based approach* [15, 8, 9] gives a recipe for doing this. It is a descendant of Büchi's original BA complementation procedure, which uses the infinite Ramsey theorem in its correctness proof.

The essence of the Ramsey-based approach for checking language inclusion between $\mathcal{A}$ and $\mathcal{B}$ lies in the notion of *supergraph*, which is a data-structure representing

a class of finite words sharing similar behavior in the two automata. Ramsey-based algorithms contain (i) an initialization phase where a set of supergraph seeds are identified, (ii) a search loop in which supergraphs are iteratively generated by composition with seeds, and (iii) a test operation where pairs of supergraphs are inspected for the existence of a counterexample. Intuitively, this counterexample has the form of an infinite ultimately periodic word $w_1(w_2)^\omega \in \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B}^c)$, where one supergraph witnesses the prefix and the other the loop. While supergraphs themselves are small, and the test in (iii) can be done efficiently, the limiting factor in the basic algorithm lies in the exponential number of supergraphs that need to be generated. Therefore, a crucial challenge in the design of Ramsey-based algorithms is to limit the supergraphs explosion problem. This can be achieved by carefully designing certain *subsumption relations* [9, 1], which allow one to safely discard subsumed supergraphs, thus reducing the search space. Moreover, methods based on minimizing supergraphs [1] by pruning their structure can further reduce the search space, and improve the complexity of (iii) above.

This paper contributes to the Ramsey-based approach to language inclusion in several ways. (1) We define a new subsumption relation based on both *forward* and *backward simulation* within the two automata. Our notion generalizes the subset-based subsumption of [9] and the forward simulation-based subsumption of [1]. (2) On a similar vein, we improve minimization of supergraphs by employing forward and backward simulation for minimizing supergraphs. (3) We introduce a method of exploiting forward simulation *between* the two automata, while previously only simulations internal to each automaton have been considered. (4) Finally, we provide a method to speed up the tests performed on supergraphs by grouping similar supergraphs together in a combined representation and extracting more abstract test-relevant information from it.

The correctness of the combined use of forward and backward simulation turns out to be far from trivial, requiring suitable generalizations of the basic notions of composition and test. Technically, we consider generalized composition and test operations where *jumps* are allowed—a jump occurring between states related by backward simulation. The proofs justifying the use of jumping composition and test are much more involved than in previous works.

We have implemented our techniques and tested them on BA derived from a set of real-world model checking benchmarks [14], and from the Tabakov-Vardi random model [17]. The new technique is able to finish many of the difficult problem instances in minutes, while the algorithm of [1] cannot finish them even in one day. The concrete numbers of our experimental results can be found in Section 8 and also in Appendix H. All the benchmarks we used, the source code, and the executable of our implementation are available at `http://www.languageinclusion.org/CONCUR2011`.

Related work. An alternative approach to language inclusion for BA is given by *rank-based* methods [13], which provide a different complementation procedure based on a rank-based analysis of rejecting runs. This approach is orthogonal to Ramsey-based algorithms. In fact, while rank-based approaches have a better worst-case complexity, Ramsey-based approaches can still perform better on many examples [9]. A subsumption-based algorithm for the rank-based approach has been given in [4]. Subsumption techniques have recently been considered also for automata over *finite words* [19, 2].

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. A *run* of $\mathcal{A}$ on a word $w = \sigma_1 \sigma_2 \ldots \in \Sigma^\omega$ *starting* in a state $q_0 \in Q$ is an infinite sequence $q_0 q_1 \ldots$ s.t. $(q_{j-1}, \sigma_j, q_j) \in \delta$ for all $j > 0$. The run is *accepting* iff $q_i \in F$ for infinitely many i. The *language of* $\mathcal{A}$ is $\mathcal{L}(\mathcal{A}) = \{w \mid \mathcal{A} \text{ has an accepting run on } w \text{ starting from some } q_0 \in I\}$.

A *path* in $\mathcal{A}$ on a finite word $w = \sigma_1 \ldots \sigma_n \in \Sigma^+$ is a finite sequence $q_0 q_1 \ldots q_n$ s.t. $\forall 0 < j \leq n : \ (q_{j-1}, \sigma_j, q_j) \in \delta$. The path is *accepting* iff $\exists 0 \leq i \leq n : \ q_i \in F$. For any $p, q \in Q$, let $p \overset{w}{\rightsquigarrow}_F q$ iff there is an accepting path on w from p to q, and $p \overset{w}{\rightsquigarrow} q$ iff there is a (not necessarily accepting) path on w from p to q.

A *forward simulation* [3] on $\mathcal{A}$ is a relation $R \subseteq Q \times Q$ such that pRr only if $p \in F \implies r \in F$, and for every transition $(p, \sigma, p') \in \delta$, there exists a transition $(r, \sigma, r') \in \delta$ s.t. $p'Rr'$. A *backward simulation* on $\mathcal{A}$ ([16], where it is called *reverse simulation*) is a relation $R \subseteq Q \times Q$ s.t. $p'Rr'$ only if $p' \in F \implies r' \in F$, $p' \in I \implies r' \in I$, and for every $(p, \sigma, p') \in \delta$, there exists $(r, \sigma, r') \in \delta$ s.t. pRr. Note that this notion of backward simulation is stronger than the usual finite-word automata version, as we require not only compatibility w.r.t. initial states, but also w.r.t. final states. It can be shown that there exists a unique maximal forward simulation denoted by $\preceq_{\mathsf{f}}^{\mathcal{A}}$ and also a unique maximal backward simulation denoted by $\preceq_{\mathsf{b}}^{\mathcal{A}}$, which are both polynomial-time computable preorders [10]. We drop the superscripts when no confusion can arise.

In the rest of the paper, we fix two BA $\mathcal{A} = (\Sigma, Q_{\mathcal{A}}, I_{\mathcal{A}}, F_{\mathcal{A}}, \delta_{\mathcal{A}})$ and $\mathcal{B} = (\Sigma, Q_{\mathcal{B}}, I_{\mathcal{B}}, F_{\mathcal{B}}, \delta_{\mathcal{B}})$. The *language inclusion problem* consists in deciding whether $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$. It is well known that deciding language inclusion is PSPACE-complete [12], and that forward simulations [3] can be used as an underapproximation thereof. Here, we focus on deciding language inclusion precisely, by giving a complete algorithm.

3 Ramsey-Based Language Inclusion Testing

Abstractly, the Ramsey-based approach for checking $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$ consists in building a *finite* set $\mathcal{X} \subseteq 2^{\mathcal{L}(\mathcal{A})}$ of fragments of $\mathcal{L}(\mathcal{A})$ satisfying the following two properties:

α (covering) $\bigcup \mathcal{X} = \mathcal{L}(\mathcal{A})$.
β (dichotomy) For all $X \in \mathcal{X}$, either $X \subseteq \mathcal{L}(\mathcal{B})$ or $X \cap \mathcal{L}(\mathcal{B}) = \emptyset$.

The covering property ensures that the considered fragments cover $\mathcal{L}(\mathcal{A})$, and the dichotomy property states that the fragments are either entirely in $\mathcal{L}(\mathcal{B})$ or disjoint from $\mathcal{L}(\mathcal{B})$. Moreover, the fragments are chosen such that they can be effectively generated and such that their inclusion in $\mathcal{L}(B)$ is easy to test. During the generation of the fragments, it then suffices to test each of them for inclusion in $\mathcal{L}(\mathcal{B})$. If this is the case, the inclusion $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$ holds. Otherwise, there is a fragment $X \subseteq \mathcal{L}(\mathcal{A}) \setminus \mathcal{L}(\mathcal{B})$ s.t. every ω-word $w \in X$ is a counterexample to the inclusion of $\mathcal{L}(\mathcal{A})$ in $\mathcal{L}(\mathcal{B})$.

We now instantiate the above described abstract algorithm by giving primitives for representing fragments of $\mathcal{L}(\mathcal{A})$ satisfying the conditions of covering and dichotomy. Much like in [8], we introduce the notion of *arcs* for satisfying Condition α, the notion of *graphs* for Condition β, and then we put them together in the notion of *supergraphs* as to satisfy $\alpha + \beta$. Then, we explain that supergraphs can be effectively generated and that the fragment languages they represent can be easily tested for inclusion in $\mathcal{L}(B)$.

Condition α*: Edges and properness.* An *edge* $\langle p,a,q\rangle$ is an element of $E_{\mathcal{A}} = Q_{\mathcal{A}} \times \{0,1\} \times Q_{\mathcal{A}}$. Its language $\mathcal{L}\langle p,a,q\rangle \subseteq \Sigma^+$ contains a word $w \in \Sigma^+$ iff either (1) $a = 1$ and $p \overset{w}{\rightsquigarrow}_F q$, or (2) $a = 0$, $p \overset{w}{\rightsquigarrow} q$, but not $p \overset{w}{\rightsquigarrow}_F q$. A pair of edges $(\langle q_1,a,q_2\rangle, \langle q_3,b,q_4\rangle)$ is *proper* iff $q_1 \in I_{\mathcal{A}}$, $q_2 = q_3 = q_4$, and $b = 1$. A pair of edges (x,y) can be used to encode the ω-language $Y_{xy} = \mathcal{L}(x) \cdot (\mathcal{L}(y))^{\omega}$. Clearly, if the pair of edges is proper, $Y_{xy} \subseteq \mathcal{L}(\mathcal{A})$. Intuitively, the language of a proper pair of edges contains words accepted by lasso-shaped accepting runs starting from q_1 and looping through q_2. Furthermore, it is clearly the case that one can completely cover $\mathcal{L}(\mathcal{A})$ by languages Y_{xy}. Thus, the set $\mathcal{X}_{\mathsf{edges}} = \{Y_{xy} \mid (x,y) \text{ is proper }\}$ satisfies Condition α.

Condition β*: Graphs.* A *graph* g is a subset of edges from $E_{\mathcal{B}} = Q_{\mathcal{B}} \times \{0,1\} \times Q_{\mathcal{B}}$ containing at most one edge for every pair of states. Its language is defined as the set of words over Σ^+ that are consistent with all the edges of the graph. Namely, $w \in \mathcal{L}(g)$ iff, for any pair of states $p,q \in Q_{\mathcal{B}}$, either (1) $p \overset{w}{\rightsquigarrow}_F q$ and $\langle p,1,q\rangle \in g$, (2) $p \overset{w}{\rightsquigarrow} q$, $\neg(p \overset{w}{\rightsquigarrow}_F q)$, and $\langle p,0,q\rangle \in g$, or (3) $\neg(p \overset{w}{\rightsquigarrow} q)$ and there is no edge in g of the form $\langle p,a,q\rangle$. Intuitively, the language of a graph consists of words that all connect any chosen pair of states in the same way (i.e., possibly through an accepting state, through non-accepting states only, or not at all). Let G be the set of all graphs. Not all graphs, however, contain meaningful information, e.g., a graph may contain an edge between states not reachable from each other. Such contradictory information makes the language of a graph empty. Define $G^f = \{g \in G \mid \mathcal{L}(g) \neq \emptyset\}$ as the set of graphs with non-empty languages.

It can be shown that the languages of graphs partition Σ^+. Like with edges, a pair of graphs (g,h) can be used to encode the ω-language $Y_{gh} = \mathcal{L}(g) \cdot (\mathcal{L}(h))^{\omega}$. Intuitively, the pair of graphs g, h encodes *all* runs in $\mathcal{B}$ over the ω-words in Y_{gh}. These runs can be obtained by selecting an edge from g and possibly multiple edges from h that can be connected by their entry/exit states to form a lasso. Since the words in the language of graphs have the same power for connecting states, accepting runs exist for all elements of Y_{gh} or for none of them. The following lemma [15, 8, 9] shows that the set $\mathcal{X}_{\mathsf{graphs}} = \{Y_{gh} \mid g,h \in G^f\}$ satisfies Condition β.

Lemma 1. *For graphs* g,h*, either* $Y_{gh} \subseteq \mathcal{L}(\mathcal{B})$ *or* $Y_{gh} \cap \mathcal{L}(\mathcal{B}) = \emptyset$*.*

Condition $\alpha + \beta$*: Supergraphs.* We combine edges and graphs to build more complex objects satisfying, at the same time, Conditions α and β. A *supergraph* is a pair $\mathbf{g} = \langle x,g\rangle \in E_{\mathcal{A}} \times G$.[1] A supergraph is only meaningful if the information in the edge-part is consistent with that in the graph-part. To this end, let $\mathcal{L}(\mathbf{g}) = \mathcal{L}(x) \cap \mathcal{L}(g)$ and let $S^f = \{\mathbf{g} \mid \mathcal{L}(\mathbf{g}) \neq \emptyset\}$ be the set of supergraphs with non-empty language. For two supergraphs $\mathbf{g} = \langle x,g\rangle$ and $\mathbf{h} = \langle y,h\rangle$, the pair $(\mathbf{g},\mathbf{h})$ is *proper* if the edge-pair (x,y) is proper. Let $Y_{\mathbf{gh}} = \mathcal{L}(\mathbf{g}) \cdot (\mathcal{L}(\mathbf{h}))^{\omega}$. Notice that $Y_{\mathbf{gh}} \subseteq Y_{xy} \cap Y_{gh}$. Therefore, since Y_{gh} satisfies Condition β, so does $Y_{\mathbf{gh}} \subseteq Y_{gh}$. For Condition α, we show that Y_{xy} can be covered by a family of languages of the form $Y_{\langle x,g\rangle\langle y,h\rangle}$. This is sound since $Y_{\langle x,g\rangle\langle y,h\rangle} \subseteq Y_{xy}$ for *any* g,h. Completeness follows from the lemma below, stating that every word $w \in Y_{xy}$ lies in a set of the form $Y_{\langle x,g\rangle\langle y,h\rangle}$. It is proved by a Ramsey-based argument.

[1] The definition of supergraph given here is slightly different from [8, 1], where the edge-part is just a pair of states (p,q). Having labels allows us to give a notion of properness which does not require to have $q \in F$.

Lemma 2. *For proper edges (x,y) and $w \in Y_{xy}$, there exist graphs g,h s.t. $w \in Y_{\langle x,g\rangle\langle y,h\rangle}$.*

Thus, Y_{xy} can be covered by $\mathcal{X}_{xy} = \{Y_{\mathbf{gh}} \mid g,h \in G^f, \mathbf{g} = \langle x,g\rangle, \mathbf{h} = \langle y,h\rangle\}$. Since $\mathcal{X}_{\mathsf{edges}}$ covers $\mathcal{L}(\mathcal{A})$, and each $Y_{xy} \in \mathcal{X}_{\mathsf{edges}}$ can be covered by $\mathcal{X}_{xy}$, it follows that $\mathcal{X} = \{Y_{\mathbf{gh}} \mid \mathbf{g},\mathbf{h} \in S^f, (\mathbf{g},\mathbf{h}) \text{ is proper}\}$ covers $\mathcal{L}(\mathcal{A})$. Thus, $\mathcal{X}$ fulfills $\alpha+\beta$.

Generating and Testing Supergraphs. While supergraphs in S^f are a convenient syntactic object for manipulating languages in $\mathcal{X}$, testing that a given supergraph has non-empty language is expensive (PSPACE-complete). In [11], this problem is elegantly solved by introducing a natural notion of *composition* of supergraphs, which preserves non-emptiness: The idea is to start with a (small) set of supergraphs which have non-empty language by construction, and then to obtain S^f by composing supergraphs until no more supergraphs can be generated.

For a BA $\mathcal{C}$ and a symbol $\sigma \in \Sigma$, let $E^\sigma_{\mathcal{C}} = \{\langle p,a,q\rangle \mid (p,\sigma,q) \in \delta_{\mathcal{C}}, (a=1 \iff p \in F \vee q \in F)\}$ be the set of edges induced by σ. The initial seed for the procedure is given by *one-letter supergraphs* in $S^1 = \bigcup_{\sigma\in\Sigma}\{(x, E^\sigma_{\mathcal{B}}) \mid x \in E^\sigma_{\mathcal{A}}\}$. Notice that $S^1 \subseteq S^f$ by construction. Next, two edges $x = \langle p,a,q\rangle$ and $y = \langle q',b,r\rangle$ are *composable* iff $q = q'$. For composable edges x and y, let $x;y = \langle p, \max(a,b), r\rangle$. Further, the *composition* $g;h$ of graphs g and h is defined as follows: $\langle p,c,r\rangle \in g;h$ iff there is a state q s.t. $\langle p,a,q\rangle \in g$ and $\langle q,b,r\rangle \in h$, and $c = \max_{q\in Q}\{\max(a,b) \mid \langle p,a,q\rangle \in g, \langle q,b,r\rangle \in h\}$. Then, supergraphs $\mathbf{g} = \langle x,g\rangle$ and $\mathbf{h} = \langle y,h\rangle$ are *composable* iff $\langle x,y\rangle$ are composable, and their *composition* is the supergraph $\mathbf{g};\mathbf{h} = \langle x;y,g;h\rangle$. Notice that S^f is closed under composition, i.e., $\mathbf{g},\mathbf{h} \in S^f \implies \mathbf{g};\mathbf{h} \in S^f$. Composition is also *complete* for generating S^f:

Lemma 3. *[1] A supergraph $\mathbf{g}$ is in S^f iff $\exists \mathbf{g}_1,\ldots,\mathbf{g}_n \in S^1$ such that $\mathbf{g} = \mathbf{g}_1;\ldots;\mathbf{g}_n$.*

Now that we have a method for generating all relevant supergraphs, we need a way of checking inclusion of (supregraphs representing) fragments of $\mathcal{L}(\mathcal{A})$ in $\mathcal{L}(\mathcal{B})$. Let $(\mathbf{g},\mathbf{h})$ be a (proper) pair of supergraphs. By the dichotomy property, $Y_{\mathbf{gh}} \subseteq \mathcal{L}(\mathcal{B})$ iff $Y_{\mathbf{gh}} \cap \mathcal{L}(\mathcal{B}) \neq \emptyset$. We test the latter condition by the so-called *double graph test*: For a pair of supergraphs $(\mathbf{g},\mathbf{h})$, $DGT(\mathbf{g},\mathbf{h})$ iff, whenever $(\mathbf{g},\mathbf{h})$ is proper, then $LFT(g,h)$. Here, LFT is the so-called *lasso-finding test*: Intuitively, LFT checks for a lasso with a handle in g and an accepting loop in h. Formally, $LFT(g,h)$ iff there is an edge $\langle p,a_0,q_0\rangle \in g$ and an infinite sequence of edges $\langle q_0,a_1,q_1\rangle$, $\langle q_1,a_2,q_2\rangle,\ldots \in h$ s.t. $p \in I$ and $a_j = 1$ for infinitely many j's.

Lemma 4. *[1] $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$ iff for all $\mathbf{g},\mathbf{h} \in S^f$, $DGT(\mathbf{g},\mathbf{h})$.*

Basic Algorithm [8]. The basic algorithm for checking inclusion enumerates all supergraphs from S^f by extending supergraphs on the right by one-letter supergraphs from S^1; that is, a supergraph $\mathbf{g}$ generates new supergraphs by selecting some $\mathbf{h} \in S^1$ and building $\mathbf{g};\mathbf{h}$. Then, $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$ holds iff all the generated pairs pass the DGT.

Intuitively, the algorithm processes all lasso-shaped runs that can be used to accept some words in $\mathcal{A}$. These runs are represented by the edge-parts of proper pairs of generated supergraphs. For each such run of $\mathcal{A}$, the algorithm uses LFT to test whether there is a corresponding accepting run of $\mathcal{B}$ among *all* the possible runs of $\mathcal{B}$ on the words represented by the given pair of supergraphs. These latter runs are encoded by the graph-parts of the respective supergraphs.

4 Optimized Language Inclusion Testing

The basic algorithm of Section 3 is wasteful for two reasons. First, not all edges in the graph component of a supergraph are needed to witness a counterexample to inclusion: Hence, we can reduce a graph by keeping only a certain subset of its edges (Optimization 1). Second, not all supergraphs need to be generated and tested: We show a method which safely allows the algorithm to discard certain supergraphs (Optimization 2). Both optimizations rely on various notions of *subsumption*, which we introduce next.

Given two edges $x = \langle p,a,q\rangle$ and $y = \langle r,b,s\rangle$, we say that y *subsumes* x, written $x \sqsubseteq y$, if $p = r$, $a \leq b$, and $q = s$; that x *forward-subsumes* y, written $x \sqsubseteq_{\mathsf{f}} y$, if $p = r$, $a \leq b$, and $q \preceq_{\mathsf{f}} s$; that x *backward-subsumes* y, written $x \sqsubseteq_{\mathsf{b}} y$, if $p \preceq_{\mathsf{b}} r$, $a \leq b$, and $q = s$; and that x *forward-backward-subsumes* y, written $x \sqsubseteq_{\mathsf{fb}} y$, if $p \preceq_{\mathsf{b}} r$, $a \leq b$, and $q \preceq_{\mathsf{f}} s$. We lift all the notions of subsumption to graphs: For any $z \in \{\mathsf{f},\mathsf{b},\mathsf{fb},\sqcup\}$ and for graphs g and h, let $g \sqsubseteq_z h$ iff, for every edge $x \in g$, there exists an edge $y \in h$ s.t. $x \sqsubseteq_z y$. Since the simulations $\preceq_{\mathsf{f}}$ and $\preceq_{\mathsf{b}}$ are preorders, all subsumptions are preorders. We define backward and forward-backward subsumption equivalence as $\simeq_{\mathsf{b}} = \sqsubseteq_{\mathsf{b}} \cap \sqsubseteq_{\mathsf{b}}^{-1}$ and $\simeq_{\mathsf{fb}} = \sqsubseteq_{\mathsf{fb}} \cap \sqsubseteq_{\mathsf{fb}}^{-1}$, respectively.

4.1 Optimization 1: Minimization of Supergraphs

The first optimization concerns the structure of individual supergraphs. Let $\mathbf{g} = \langle x,g\rangle \in S$ be a supergraph, with g its graph-component. We minimize g by deleting edges therein which are subsumed by $\sqsubseteq_{\mathsf{fb}}$-larger ones. That is, whenever we have $x \sqsubseteq_{\mathsf{fb}} y$ for two edges $x,y \in g$, we remove x and keep y. Intuitively, subsumption-larger arcs contribute more to the capability of representing lassoes since their right and left endpoints are $\preceq_{\mathsf{f}}/\preceq_{\mathsf{b}}$-larger, respectively, and have therefore a richer choice of possible futures and pasts. Subsumption smaller arcs are thus redundant, and removing them does not change the capability of g to represent lassoes in $\mathcal{B}$. Formally, we define a minimization operation *Min* mapping a supergraph $\mathbf{g} = \langle x,g\rangle$ to its minimized version $\mathit{Min}(\mathbf{g}) = \langle x,\mathit{Min}(g)\rangle$ where $\mathit{Min}(g)$ is the minimization applied to the graph-component.[2]

Definition 1. *For two graphs g and h, let $g \leqslant h$ iff (1) $g \sqsubseteq h$ and (2) $h \sqsubseteq_{\mathsf{fb}} g$. For supergraphs $\mathbf{g} = \langle x,g\rangle$ and $\mathbf{h} = \langle y,h\rangle$, let $\mathbf{g} \leqslant \mathbf{h}$ iff $x = y$ and $g \leqslant h$. A* minimization *of graphs is any function Min such that, for any graph h, $\mathit{Min}(h) \leqslant h$.*

Point 1 in the definition of $\leqslant$ allows some edges to be erased or their label decreased. Point 2 states that only subsumed arcs can be removed or have their label decreased. Note also that, clearly, $\mathit{Min}(\mathbf{h}) \leqslant \mathbf{h}$ holds for any supergraph $\mathbf{h}$. Finally, note that *Min* is not uniquely determined: First, there are many candidates satisfying $\mathit{Min}(h) \leqslant h$. Yet, an implementation will usually remove a maximal number of edges to keep the size of graphs to a minimum. Second, even if we required $\mathit{Min}(h)$ to be a $\leqslant$-smallest element (i.e., no further edge can be removed), the minimization process might encounter $\sqsubseteq_{\mathsf{fb}}$-equivalent edges, and in this case, we do not specify which ones get removed. Therefore, we prove correctness for any minimization satisfying $\mathit{Min}(h) \leqslant h$.

Intuitively, a minimized supergraph $\mathbf{g}$ can be seen as a small *representative* of all supergraphs $\mathbf{h} \in G^f$ with $\mathbf{g} \leqslant \mathbf{h}$, and of all the fragments of $\mathcal{L}(A)$ encoded by them.

[2] In [1], we used $\sqsubseteq_{\mathsf{f}}$ for minimization. The theory allowing the use of $\sqsubseteq_{\mathsf{fb}}$ is significantly more involved, but as as shown in Section 8, the use of $\sqsubseteq_{\mathsf{fb}}$ turns out to be much more advantageous.

Using representatives allows us to deal with a smaller number of smaller supergraphs. We now explain how (sufficiently many) representatives encoding fragments of $\mathcal{L}(\mathcal{A})$ can be *generated* and *tested* for inclusion in $\mathcal{L}(\mathcal{B})$.

Generating representatives of supergraphs. We need to create a representative of each supergraph in S^f by composing representatives only. Let $\mathbf{g} = \langle x, g\rangle$ and $\mathbf{h} = \langle y, h\rangle$ be two composable supergraphs, representing $\mathbf{g}' = \langle x, g'\rangle$ and $\mathbf{h}' = \langle y, h'\rangle$, respectively. If graph composition were $\leqslant$-monotone, i.e., $\mathbf{g};\mathbf{h} \leqslant \mathbf{g}';\mathbf{h}'$, then we would be done. However, graph composition is not monotone: The reason is that some composable edges $e \in g'$ and $f \in h'$ may be erased by minimization, and be represented by some $\hat{e} \in g$ and $\hat{f} \in h$ instead, with $e \sqsubseteq_{\mathsf{fb}} \hat{e}$ and $f \sqsubseteq_{\mathsf{fb}} \hat{f}$. But now, $\hat{e}$ and $\hat{f}$ are not necessarily composable anymore. Thus, $\mathbf{g};\mathbf{h} \not\leqslant \mathbf{g}';\mathbf{h}'$. We solve this problem in two steps: We allow composition to jump to $\preceq_{\mathsf{b}}$-larger states (Def. 2), and relax the notion of representative (Def. 3).

Definition 2. *Given graphs $g,h \in G$, their* jumping composition $g \fatsemi_{\mathsf{b}} h$ *contains an edge $\langle p,c,r\rangle \in g \fatsemi_{\mathsf{b}} h$ iff there are edges $\langle p,a,q\rangle \in g$, $\langle q',b,r\rangle \in h$ s.t. $q \preceq_{\mathsf{b}} q'$, and $c = \max_{q,q'}\{\max(a,b) \mid \langle p,a,q\rangle \in g, \langle q',b,r\rangle \in h, q \preceq_{\mathsf{b}} q'\}$. For two composable supergraphs $\mathbf{g} = \langle x, g\rangle$ and $\mathbf{h} = \langle y, h\rangle$, let $\mathbf{g} \fatsemi_{\mathsf{b}} \mathbf{h} = \langle x; y, g \fatsemi_{\mathsf{b}} h\rangle$.*

Jumping composition alone does not yet give the required monotonicity property. The problem is that $\mathbf{g} \fatsemi_{\mathsf{b}} \mathbf{h}$ is not necessarily a minimized version of $\mathbf{g}';\mathbf{h}'$, but it is only a minimized version of something $\simeq_{\mathsf{b}}$-equivalent to $\mathbf{g}';\mathbf{h}'$. This leads us to the following more liberal notion of representatives, which is based on $\leqslant$ modulo the equivalence $\simeq_{\mathsf{b}}$, and for which Lemma 5 proves the required monotonicity property.

Definition 3. *A graph $g \in G$ is a* representative *of a graph $h \in G^f$, denoted $g \trianglelefteq h$, iff there exists $\bar{h} \in G$ such that $g \leqslant \bar{h} \simeq_{\mathsf{b}} h$. For supergraphs $\mathbf{g} = \langle x, g\rangle, \mathbf{h} = \langle y, h\rangle \in S$, we say that $\mathbf{g}$ is a* representative *of $\mathbf{h}$, written $\mathbf{g} \trianglelefteq \mathbf{h}$, iff $x = y$ and $g \trianglelefteq h$. Let $S^R = \{\mathbf{g} \mid \exists \mathbf{h} \in S^f.\ \mathbf{g} \trianglelefteq \mathbf{h}\}$ be the set of representatives of supergraphs.*

Lemma 5. *For supergraphs $\mathbf{g},\mathbf{h} \in S^R$ and $\mathbf{g}',\mathbf{h}' \in S^f$, if $\mathbf{g} \trianglelefteq \mathbf{g}'$, $\mathbf{h} \trianglelefteq \mathbf{h}'$ and $\mathbf{g}',\mathbf{h}'$ are composable, then $\mathbf{g},\mathbf{h}$ are composable and $\mathbf{g} \fatsemi_{\mathsf{b}} \mathbf{h} \trianglelefteq \mathbf{g}';\mathbf{h}'$ and $\mathbf{g} \fatsemi_{\mathsf{b}} \mathbf{h} \in S^R$.*

Lemma 6. *Let $\mathbf{f} \in S$, $\mathbf{g} \in S^R$, and $\mathbf{h} \in S^f$. If $\mathbf{f} \leqslant \mathbf{g}$ and $\mathbf{g} \trianglelefteq \mathbf{h}$, then $\mathbf{f} \trianglelefteq \mathbf{h}$ (and thus $\mathbf{f} \in S^R$). In particular, the statement holds when $\mathbf{f} = \mathit{Min}(\mathbf{g})$.*

Lemmas 5, 6, and 3 imply that creating supergraphs by $\fatsemi_{\mathsf{b}}$-composing representatives, followed by further minimization, suffices to create a representative of each supergraph in S^f. This solves the problem of generating representatives of supergraphs.

Weak properness and Relaxed DGT. We now present a relaxed DGT proposed in [1], which we further improve below. The idea is to weaken the properness condition in order to allow more pairs of supergraphs to be eligible for LFT on their graph part. This may lead to a quicker detection of a counterexample. Weak properness is sound since it still produces fragments $Y_{\mathbf{gh}} \subseteq \mathcal{L}(\mathcal{A})$ as required by Condition α. Completeness is guaranteed since properness implies weak properness.

Definition 4. *(adapted from [1]) A pair of edges $(\langle p,a,q\rangle, \langle r,b,s\rangle)$ is* weakly proper *iff $p \in I_{\mathcal{A}}$, $r \preceq_{\mathsf{f}} q$, $r \preceq_{\mathsf{f}} s$, and $b = 1$,[3] and a pair of supergraphs $(\mathbf{g} = \langle x, g\rangle, \mathbf{h} = \langle y, h\rangle)$ is*

[3] We note that instead of testing $r \preceq_{\mathsf{f}} q$, testing inclusion of the languages of the states is sufficient. Furthermore, instead of testing $r \preceq_{\mathsf{f}} s$, one can test for *delayed simulation*, but not for language inclusion. See Lemma 34 in Appendix C.

weakly proper when (x,y) *is weakly proper. Supergraphs* $\mathbf{g},\mathbf{h}$ *pass the* relaxed double graph test*, denoted* $RDGT(\mathbf{g},\mathbf{h})$*, iff whenever* $(\mathbf{g},\mathbf{h})$ *is weakly proper, then* $LFT(g,h)$.

Lemma 7. *[1]* $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$ *iff for all* $\mathbf{g},\mathbf{h} \in S^f$*,* $RDGT(\mathbf{g},\mathbf{h})$.

Testing representatives of supergraphs. We need a method for testing inclusion in $\mathcal{L}(\mathcal{B})$ of the fragments of $\mathcal{L}(\mathcal{A})$ encoded by representatives of supergraphs that is equivalent to testing inclusion of fragments of $\mathcal{L}(\mathcal{A})$ encoded by the represented supergraphs. As with composition, minimization is not compatible with such testing since edges needed to find loops may be erased during the minimization process. Technically, this results in the LFT (and therefore RDGT) not being $\trianglelefteq$-monotone. Therefore, we generalize the LFT by allowing jumps to $\preceq_{\mathsf{b}}$-larger states, in a similar way as with $\fatsemi_{\mathsf{b}}$. Lemma 8 establishes the required monotonicity property.

Definition 5. *A pair of graphs* (g,h) *passes the* jumping lasso-finding test*, denoted* $LFT_{\mathsf{b}}(g,h)$*, iff there is an edge* $\langle p,a_0,q_0\rangle$ *in* g *and an infinite sequence of edges* $\langle q'_0,a_1,q_1\rangle$*,* $\langle q'_1,a_2,q_2\rangle,\ldots$ *in* h *s.t.* $p \in I$*,* $q_i \preceq_{\mathsf{b}} q'_i$ *for all* $i \geq 0$*, and* $a_j = 1$ *for infinitely many* j*'s. A pair of supergraphs* $(\mathbf{g},\mathbf{h})$ *passes the* jumping relaxed double graph test*, denoted* $RDGT_{\mathsf{b}}(\mathbf{g},\mathbf{h})$*, iff whenever* $(\mathbf{g},\mathbf{h})$ *is weakly proper, then* $LFT_{\mathsf{b}}(g,h)$.

Lemma 8. *For any* $\mathbf{g},\mathbf{h} \in S^R$ *and* $\mathbf{g}',\mathbf{h}' \in S^f$ *such that* $\mathbf{g} \trianglelefteq \mathbf{g}'$ *and* $\mathbf{h} \trianglelefteq \mathbf{h}'$*, it holds that* $RDGT_{\mathsf{b}}(\mathbf{g},\mathbf{h}) \iff RDGT(\mathbf{g}',\mathbf{h}')$.

Algorithm with minimization. By Lemma 8, $\mathrm{RDGT}_{\mathsf{b}}$ on representatives is equivalent to RDGT on the represented supergraphs. Together with Lemma 7, this means that it is enough to generate a representative of each supergraph from S^f, and test all pairs of the generated supergraphs with $\mathrm{RDGT}_{\mathsf{b}}$. Thus, we have obtained a modification of the basic algorithm which starts from minimized 1-letter supergraphs in $Min(S^1) = \{Min(\mathbf{g}) \mid \mathbf{g} \in S^1\}$, and constructs new supergraphs by $\fatsemi_{\mathsf{b}}$-composing already generated supergraphs with $Min(S^1)$ on the right. New supergraphs are further minimized with Min. Inclusion holds iff all pairs of generated supergraphs pass $\mathrm{RDGT}_{\mathsf{b}}$.

4.2 Optimization 2: Discarding Subsumed Supergraphs

The second optimization gives a rule for discarding supergraphs subsumed by some other supergraph. This is safe in the sense that if a subsumed supergraph can yield a counterexample to language inclusion, then also the subsuming one can yield a counterexample. We present an improved version of the subsumption from [1]. The new version uses both $\preceq_{\mathsf{f}}$ and $\preceq_{\mathsf{b}}$ on the $\mathcal{B}$ part of supergraphs instead of $\preceq_{\mathsf{f}}$ only. This allows us to discard significantly more supergraphs than in [1], as illustrated in Section 8.

Definition 6. *We say that a supergraph* $\mathbf{g} = \langle x,g\rangle$ subsumes *a supergraph* $\mathbf{g}' = \langle y,g'\rangle$*, written* $\mathbf{g} \sqsubseteq_{\mathsf{fb}} \mathbf{g}'$*, iff* $y \sqsubseteq_{\mathsf{f}} x$ *and* $g \sqsubseteq_{\mathsf{fb}} g'$.

Intuitively, if $y \sqsubseteq_{\mathsf{f}} x$, then x has more power for representing lassoes in $\mathcal{A}$ than y since, by the properties of forward simulation, it has a richer choice of possible forward continuations in $\mathcal{A}$. On the other hand, $g \sqsubseteq_{\mathsf{fb}} g'$ means that g' has more chance of representing lassoes in $\mathcal{B}$ than g: In fact, g' contains edges that have a richer choice of backward continuations (due to the $\preceq_{\mathsf{b}}$ on the left endpoints of the edges) as well as

a richer choice of forward continuations (due to the $\preceq_{\mathsf{f}}$ on the right endpoints). Thus, it is more likely for $\mathbf{g}$ than for $\mathbf{g}'$ to lead to a counterexample to language inclusion. This intuition is confirmed by the lemma below, stating the $\sqsubseteq_{\mathsf{fb}}$-monotonicity of $\mathrm{RDGT}_{\mathsf{b}}$.

Lemma 9. *For supergaphs* $\mathbf{g},\mathbf{h} \in S^R$ *and* $\mathbf{g}',\mathbf{h}' \in S$*, if* $\mathbf{g} \sqsubseteq_{\mathsf{fb}} \mathbf{g}'$ *and* $\mathbf{h} \sqsubseteq_{\mathsf{fb}} \mathbf{h}'$*, then* $RDGT_{\mathsf{b}}(\mathbf{g},\mathbf{h}) \Rightarrow RDGT_{\mathsf{b}}(\mathbf{g}',\mathbf{h}')$.

Therefore, no counterexample is lost by testing only $\sqsubseteq_{\mathsf{fb}}$-smaller supergraphs. To show that we can completely discard $\sqsubseteq_{\mathsf{fb}}$-larger supergraphs, we need to show that subsumption is compatible with composition, i.e., that descendants of larger supergraphs are (eventually) subsumed by descendants of smaller ones. Ideally, we would achieve this by showing the following more general fact: For two composable representatives $\mathbf{g}',\mathbf{h}' \in S^R$ that are subsumed by supergraphs $\mathbf{g}$ and $\mathbf{h}$, respectively, the composite supergraph $\mathbf{g} \fatsemi_{\mathsf{b}} \mathbf{h}$ subsumes $\mathbf{g}' \fatsemi_{\mathsf{b}} \mathbf{h}'$. The problem is that subsumption does not preserve composability: Even if $\mathbf{g}',\mathbf{h}'$ are composable, this needs not to hold for $\mathbf{g},\mathbf{h}$.

We overcome this difficulty by taking into account the specific way supergraphs are generated by the algorithm. Since we only generate new supergraphs by composing old ones on the right with 1-letter minimized supergraphs, we do not need to show that arbitrary composition is $\sqsubseteq_{\mathsf{fb}}$-monotone. Instead, we show that, for representatives $\mathbf{g},\mathbf{g}' \in S^R$ and a 1-letter minimized supergraph $\mathbf{h}' \in Min(S^1)$, if $\mathbf{g}$ subsumes $\mathbf{g}'$, then there will always be a supergraph $\mathbf{h}$ available which is composable with $\mathbf{g}$ such that $\mathbf{g} \fatsemi_{\mathsf{b}} \mathbf{h}$ subsumes $\mathbf{g}' \fatsemi_{\mathsf{b}} \mathbf{h}'$. Thus, we can safely discard $\mathbf{g}'$ from the rest of the computation.

Lemma 10. *For any* $\mathbf{g},\mathbf{g}' \in S^R$ *with* $\mathbf{g} \sqsubseteq_{\mathsf{fb}} \mathbf{g}'$ *and* $\mathbf{h}' \in Min(S^1)$ *such that* $\mathbf{g}'$ *and* $\mathbf{h}'$ *are composable, there exists* $\hat{\mathbf{h}} \in Min(S^1)$ *such that for all* $\mathbf{h} \in S^R$ *with* $\mathbf{h} \sqsubseteq_{\mathsf{fb}} \hat{\mathbf{h}}$*,* $\mathbf{g}$ *is composable with* $\mathbf{h}$ *and* $\mathbf{g} \fatsemi_{\mathsf{b}} \mathbf{h} \sqsubseteq_{\mathsf{fb}} \mathbf{g}' \fatsemi_{\mathsf{b}} \mathbf{h}'$.

Algorithm with minimization and subsumption. We have obtained a modification of the algorithm with minimization. It starts with a subset $Init \subseteq Min(S^1)$ of $\sqsubseteq_{\mathsf{fb}}$-smallest minimized one-letter supergraphs. New supergraphs are generated by $\fatsemi_{\mathsf{b}}$-composition on the right with supergraphs in *Init*, followed by minimization with *Min*. Generated supergraphs that are $\sqsubseteq_{\mathsf{fb}}$-larger than other generated supergraphs are discarded. The inclusion holds iff all pairs of generated supergraphs that are not discarded pass $\mathrm{RDGT}_{\mathsf{b}}$. (An illustration of a run of the algorithm can be found in Appendix D.)

5 Using Forward Simulation Between $\mathcal{A}$ and $\mathcal{B}$

Previously, we showed that some supergraphs can safely be discarded because some $\sqsubseteq_{\mathsf{fb}}$-smaller ones are retained, which preserves the chance to find a counterexample to language inclusion. Our subsumption relation $\sqsubseteq_{\mathsf{fb}}$ is based on forward/backward simulation on $\mathcal{A}$ and $\mathcal{B}$. In order to use forward simulation *between* $\mathcal{A}$ and $\mathcal{B}$, we describe a different reason to discard supergraphs. Generally, supergraphs can be discarded because they can neither find a counterexample to inclusion (i.e., always pass the RDGT) nor generate any supergraph that can find a counterexample. However, the RDGT is asymmetric w.r.t. the left and right supergraph. Thus, a supergraph that is useless (i.e., not counterexample-finding) in the left role is not necessarily useless in the right role (and vice-versa). The following condition C is sufficient for a supergraph to be *useless on the left*. Moreover, C is efficiently computable and compatible with subsumption. Therefore, its use preserves the soundness and completeness of our algorithm.

Definition 7. *For* $\mathbf{g}{=}\langle\langle p,a,q\rangle,g\rangle \in S$, $\mathsf{C}(\mathbf{g})$ *iff* $p \notin I_{\mathcal{A}} \vee (\exists\langle r,b,s\rangle \in g.\ r \in I_{\mathcal{B}} \wedge q \preceq_{\mathsf{f}}^{\mathcal{AB}} s)$.

The first part $p \notin I_{\mathcal{A}}$ of the condition is obvious because paths witnessing counterexamples to inclusion must start in an initial state. The second part $(\exists\langle r,b,s\rangle \in g.\ r \in I_{\mathcal{B}}, q \preceq_{\mathsf{f}}^{\mathcal{AB}} s)$ uses forward-simulation $\preceq_{\mathsf{f}}^{\mathcal{AB}}$ *between* $\mathcal{A}$ and $\mathcal{B}$ to witness that neither this supergraph nor any other supergraph generated from it will find a counterexample *when used on the left side of the RDGT*. It might still be needed for tests on the right side of the RDGT though. Instead of $\preceq_{\mathsf{f}}^{\mathcal{AB}}$, every relation implying language inclusion would suffice, but (as mentioned earlier) simulation preorder is efficiently computable while inclusion is PSPACE-complete. The following lemma shows the correctness of C.

Lemma 11. $\forall \mathbf{g},\mathbf{h} \in S^R.\ \mathsf{C}(\mathbf{g}) \Rightarrow \mathit{RDGT}_{\mathsf{b}}(\mathbf{g},\mathbf{h})$.

C is $\sqsubseteq_{\mathsf{fb}}$-upward-closed and closed w.r.t. right extensions. Hence, it is compatible with subsumption-based pruning of the search space and with the employed incremental construction of supergraphs (namely, satisfaction of the condition is inherited to supergraphs newly generated by right extension with one-letter supergraphs).[4]

Lemma 12. *Let* $\mathbf{g},\mathbf{h} \in S$ *s.t.* $\mathbf{g} \sqsubseteq_{\mathsf{fb}} \mathbf{h}$. *Then* $\mathsf{C}(\mathbf{g}) \Rightarrow \mathsf{C}(\mathbf{h})$.

Lemma 13. *Let* $\mathbf{g} \in S^R$, $\mathbf{h} \in \mathit{Min}(S^1)$ *be composable. Then* $\mathsf{C}(\mathbf{g}) \Rightarrow \mathsf{C}(\mathbf{g} \fatsemi_{\mathsf{b}} \mathbf{h})$.

In principle, one could store separate sets of supergraphs for use on the left/right in the RDGT, respectively. However, since all supergraphs need to be used on the right anyway, a simple flag is more efficient. We assign the label L to a supergraph to indicate that it is still useful on the left in the RDGT. If a supergraph satisfies condition C, then the L-label is removed. The algorithm counts the number of stored supergraphs that still carry the L-label. If this number drops to zero, then (1) it will remain zero (by Lemma 13), and (2) no RDGT will ever find a counterexample: In this case, the algorithm can terminate early and report inclusion. In the special case where forward-simulation holds even between the *initial* states of $\mathcal{A}$ and $\mathcal{B}$, condition C is true for *every* generated supergraph. Thus, all L-labels are removed and the algorithm terminates immediately, reporting inclusion. Of course, condition C can also help in other cases where simulation does not hold between initial states but "more deeply" inside the automata.

The following lemma shows that if some supergraph $\mathbf{g}$ can find a counterexample when used on the left in the RDGT, then at least one of its 1-letter right-extensions can also find a counterexample. Intuitively, the counterexample has the form of a prefix followed by an infinite loop, and the prefix can always be extended by one step. E.g., the infinite words $xy(abc)^\omega$ and $xya(bca)^\omega$ are equivalent. This justifies the optimization in line 15 of our algorithm (see Appendix G).

Lemma 14. *Let* $\mathbf{g},\mathbf{h} \in S^R$. *If* $\neg\mathit{RDGT}_{\mathsf{b}}(\mathbf{g},\mathbf{h})$, *then there exists a* $\sqsubseteq_{\mathsf{fb}}$*-minimal supergraph* $\mathbf{f}$ *in* $\mathit{Min}(S^1)$ *and* $\mathbf{e} \in S^R$ *s.t.* $\neg\mathit{RDGT}_{\mathsf{b}}(\mathbf{g} \fatsemi_{\mathsf{b}} \mathbf{f},\mathbf{e})$.[5]

[4] We note that there is no good analogous condition for *useless on the right* because of the need to be closed under right extensions.

[5] A slightly modified version, Lemma 37 in Appendix E, holds for the version of the RDGT mentioned in the footnote on Definition 4.

6 Metagraphs and a New RDGT

Since many supergraphs share the same graph for $\mathcal{B}$, they can be more efficiently represented by a combined structure that we call a *metagraph*. Moreover, metagraphs allow to define a new RDGT where several $\mathcal{A}$-edges jointly witness a counterexample to inclusion, so that counterexamples can be found earlier than with individual supergraphs.

A metagraph is a structure (X,g) where $X \subseteq E_{\mathcal{A}}$ is a set of $\mathcal{A}$-edges and $g \in G_{\mathcal{B}}$. The metagraph (X,g) represents the set of all supergraphs $\langle x,g\rangle$ with $x \in X$. The L-labels of supergraphs then become labels of the elements of X since the graph g is the same.

We lift basic concepts from supergraphs to metagraphs. For every character $\sigma \in \Sigma$, there is exactly one single-letter metagraph $(E^{\sigma}_{\mathcal{A}}, E^{\sigma}_{\mathcal{B}})$. Let $M^1 = \{(E^{\sigma}_{\mathcal{A}}, E^{\sigma}_{\mathcal{B}}) \mid \sigma \in \Sigma\}$. Thus, the set of single-letter metagraphs M^1 represents all single-letter supergraphs in S^1. The function *RightExtend* defines the composition of two metagraphs such that $\mathit{RightExtend}((X,g),(Y,h)) = (X;Y, g \fatsemi_{\mathsf{b}} h)$, which is the metagraph containing the supergraphs that are $\fatsemi_{\mathsf{b}}$-right extensions of supergraphs contained in (X,g) by supergraphs contained in (Y,h). The L-labels of the elements $z \in X;Y$ are assigned after testing condition C. The function $\mathit{Min}_{\mathsf{f}}$ is defined on sets $X \subseteq E_{\mathcal{A}}$ s.t. $\mathit{Min}_{\mathsf{f}}(X)$ contains the $\sqsubseteq_{\mathsf{f}}$-minimal edges of X. If some edges are $\sqsubseteq_{\mathsf{f}}$-equivalent, then $\mathit{Min}_{\mathsf{f}}(X)$ contains just any of them. Let $\mathit{Min}_M(X,g) = (\mathit{Min}_{\mathsf{f}}(X), \mathit{Min}(g))$. Thus, $\mathit{Min}_M(X,g)$ contains exactly one representative of every $\simeq_{\mathsf{fb}}$ equivalence class of the $\sqsubseteq_{\mathsf{fb}}$-minimal supergraphs in (X,g).

It is not meaningful to define subsumption for metagraphs. Instead, we need to remove certain supergraphs (i.e., $\mathcal{A}$-edges) from some metagraph if another metagraph contains a $\sqsubseteq_{\mathsf{fb}}$-smaller supergraph. If no $\mathcal{A}$-edge remains, i.e., $X = \emptyset$ in (X,g), then this metagraph can be discarded. This is the purpose of introducing the function *Clean*: It takes two metagraphs (X,g) and (Y,h), and it returns a metagraph (Z,g) that describes all supergraphs from (X,g) for which there is no $\sqsubseteq_{\mathsf{fb}}$-smaller supergraph in (Y,h). Formally, if $h \sqsubseteq_{\mathsf{fb}} g$, then $x \in Z$ iff $x \in X$ and $\nexists y \in Y$ s.t. $x \sqsubseteq_{\mathsf{f}} y$. Otherwise, if $h \not\sqsubseteq_{\mathsf{fb}} g$, then $Z = X$. Now we define a generalized RDGT on metagraphs.

Definition 8. *A pair of sets of $\mathcal{A}$-edges $X,Y \subseteq E_{\mathcal{A}}$ passes the* forward-downward jumping lasso-finding test*, denoted $\mathit{LFT}_{\mathsf{f}}(X,Y)$, iff there is an arc $\langle p, a_0, q_0\rangle$ in X (with the L-label) and an infinite sequence of arcs $\langle q'_0, a_1, q_1\rangle, \langle q'_1, a_2, q_2\rangle, \ldots$ in Y s.t. $p \in I_{\mathcal{A}}$, $q'_i \preceq_{\mathsf{f}} q_i$ for all $i \geq 0$, and $a_j = 1$ for infinitely many j's.*

Definition 9. $\mathit{RDGT}^M_{\mathsf{b}}((X,g),(Y,h))$ *iff, whenever $\mathit{LFT}_{\mathsf{f}}(X,Y)$, then $\mathit{LFT}_{\mathsf{b}}(g,h)$.*

The following lemma shows the soundness of the new RDGT.

Lemma 15. *Let $(X,g),(Y,h)$ be metagraphs where all contained supergraphs are in S^R. If $\neg\mathit{RDGT}^M_{\mathsf{b}}((X,g),(Y,h))$, then $\mathcal{L}(\mathcal{A}) \not\subseteq \mathcal{L}(\mathcal{B})$.*

If there are $x \in X, y \in Y$ s.t. $\neg\mathit{RDGT}_{\mathsf{b}}(\langle x,g\rangle, \langle y,h\rangle)$, then $\neg\mathit{RDGT}^M_{\mathsf{b}}((X,g),(Y,h))$, by Definitions 4, 8, and 9. Thus the completeness of the new RDGT follows already from Lemmas 7 and 8. Checking $\mathit{RDGT}^M_{\mathsf{b}}((X,g),(Y,h))$ can be done very efficiently for large numbers of metagraphs, by using an abstraction technique that extracts test-relevant information from the metagraphs and stores it separately (see Appendix G).

7 The Main Algorithm

Algorithm 1 describes our inclusion testing algorithm. The function *Clean* is extended to sets of metagraphs in the standard way and implemented in procedures $Clean_1$ and $Clean_3$ in which the result overwrites the first argument (the two procedures differ in the role of the first argument, and $Clean_3$ in addition discards empty metagraphs). Lines 1-6 compute the metagraphs which contain the subsumption-minimal 1-letter supergraphs. Lines 7-10 initialize the set *Next* with these metagraphs and assign the correct labels by testing condition C. $L(x)$ denotes that the $\mathcal{A}$-arc x is labeled with L. Lines 11-21 describe the main loop. It runs until *Next* is empty or there are no more L-labels left. In the main loop, metagraphs are tested (lines 13-14) and then moved from *Next* to *Processed* without the L-label (line 15). Moreover, new metagraphs are created and some parts of them discarded by the *Clean* operation (lines 16-21). Extra bookkeeping is needed to handle the case where L-labels are regained by supergraphs in *Processed* in line 19 (see $Clean_2$ in Appendix F).

Algorithm 1: *Inclusion Checking with Metagraphs*

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 $M^1_{\mathcal{A},\mathcal{B}}$.
Output: TRUE if $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$. Otherwise, FALSE.

1 $Next := \{Min_M((X,g)) \mid (X,g) \in M^1_{\mathcal{A},\mathcal{B}}\}$; $Init := \emptyset$;
2 **while** $Next \neq \emptyset$ **do**
3 Pick and remove a metagraph (X,g) from *Next*;
4 $Clean_1((X,g), Init)$;
5 **if** $X \neq \emptyset$ **then**
6 $Clean_3(Init, (X,g))$; Add (X,g) to *Init*;
7 $Processed := \emptyset$; $Next := Init$;
8 **foreach** $(X,g) \in Next$ **do**
9 **foreach** $x \in X$ **do**
10 **if** $\neg \mathsf{C}(\langle x, g \rangle)$ **then** label x with L
11 **while** $Next \neq \emptyset \wedge \exists (X,g) \in Next \cup Processed.\ \exists x \in X.L(x)$ **do**
12 Pick a metagraph (X,g) from *Next* and remove (X,g) from *Next*;
13 **if** $\neg RDGT^M_\mathsf{b}((X,g),(X,g))$ **then return** *FALSE*;
14 **if** $\exists (Y,h) \in Processed : \neg RDGT^M_\mathsf{b}((Y,h),(X,g)) \vee \neg RDGT^M_\mathsf{b}((X,g),(Y,h))$ **then return** *FALSE*;
15 Create (X',g) from (X,g) by removing the L-labels from X and add (X',g) to *Processed*;
16 **foreach** $(Y,h) \in Init$ **do**
17 $(Z,f) := Min_M(RightExtend((X,g),(Y,h)))$;
18 **if** $Z \neq \emptyset$ **then** $Clean_1((Z,f), Next)$;
19 **if** $Z \neq \emptyset$ **then** $Clean_2((Z,f), Processed)$;
20 **if** $Z \neq \emptyset$ **then**
21 $Clean_3(Next, (Z,f))$; $Clean_3(Processed, (Z,f))$; Add (Z,f) to *Next*;
22 **return** *TRUE*;

Theorem 1. *Algorithm 1 terminates. It returns* TRUE *iff* $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$.

Table 1. Language inclusion checking on mutual exclusion protocols. Forward simulation holds between initial states. The option `-c` is extremely effective in such cases.

Protocol	$\mathcal{A}$		$\mathcal{B}$		Algorithm	New Algorithm			
	Trans.	States	Trans.	States	of [1]	default	`-b`	`-b -qr`	`-b -qr -c`
Peterson	33	20	34	20	0.46s	0.39s	0.54	0.61s	0.03s
Phils	49	23	482	161	12h36m	11h3m	7h21m	7h23m	0.1s
Mcs	3222	1408	21503	7963	>24h	2m43s	2m32s	2m49s	1m24s
Bakery	2703	1510	2702	1509	>24h	>24h	>24h	>24h	12s
Fischer	1395	634	3850	1532	4h50m	2m38s	2m50s	27s	3.6s
FischerV2	147	56	147	56	13m15s	5m14s	1m26s	1m1s	0.1s

Table 2. Language inclusion checking on mutual exclusion protocols. Language inclusion holds, but forward simulation does not hold between initial states (we call this category "inclusion"). The new alg. is much better in FischerV3, due to metagraphs. Option `-b` is effective in FischerV4. BakeryV2 is a case where `-c` is useful even if simulation does not hold between initial states.

Protocol	$\mathcal{A}$		$\mathcal{B}$		Algorithm	New Algorithm			
	Trans.	States	Trans.	States	of [1]	default	`-b`	`-b -qr`	`-b -qr -c`
FischerV3	1400	637	1401	638	3h6m	45s	10s	11s	7s
FischerV4	147	56	1506	526	>24h	>24h	1h31m	2h12m	2h12m
BakeryV2	2090	1149	2091	1150	>24h	>24h	>24h	>24h	18s

8 Experimental Results

We have implemented the proposed inclusion-checking algorithm in Java (the implementation is available at `http://www.languageinclusion.org/CONCUR2011`) and tested it on automata derived from (1) mutual exclusion protocols [14] and (2) the Tabakov-Vardi model [17]. We have compared the performance of the new algorithm with the one in [1] (which only uses supergraphs, not metagraphs, and subsumption and minimization based on forward simulation on $\mathcal{A}$ and on $\mathcal{B}$), and found it better on average, and, in particular, on difficult instances where the inclusion holds. Below, we present a condensed version of the results. Full details can be found in Appendix H.

In the first experiment, we inject artificial errors into models of several mutual exclusion protocols from [14][6], translate the modified versions into BA, and compare the sequences of program states (w.r.t. occupation of the critical section) of the two versions. For each protocol, we test language inclusion $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$ of two variants $\mathcal{A}$ and $\mathcal{B}$. We use a timeout of 24 hours and a memory limit of 4GB. We record the running time and indicate a timeout by ">24h". We compare the algorithm from [1] against its various improvements proposed above. The basic new setting (denoted as "default" in the results) uses forward simulation as in [1] together with metagraphs from Section 6 (and some further small optimizations described in Appendix G). Then, we gradually add the use of backward simulation proposed in Section 4 (denoted by `-b` in the results) and forward simulation between $\mathcal{A}$ and $\mathcal{B}$ from Section 5 (denoted by `-c`, finally yielding the algorithm of Section 7). We also consider repeated quotienting w.r.t. forward/backward-simulation-equivalence before starting the actual inclusion checking (denoted by `-qr`), while the default does quotienting w.r.t. forward simulation only. In order to better show the capability of the new techniques, the results are categorized into

[6] The models in [14] are based on guarded commands. We derive variants from them by randomly weakening or strengthening the guard of some commands.

Table 3. Language inclusion checking on mutual exclusion protocols. Language inclusion does not hold. Note that the new algorithm uses a different search strategy (BFS) than the alg. in [1].

Protocol	$\mathcal{A}$		$\mathcal{B}$		Algorithm	New Algorithm			
	Trans.	States	Trans.	States	of [1]	default	`-b`	`-b -qr`	`-b -qr -c`
BakeryV3	2090	1149	2697	1506	12m19s	5s	6s	16s	15s
FischerV5	3850	1532	1420	643	7h28m	1m6s	1m47s	39s	36s
PhilsV2	482	161	212	80	1.1s	0.7s	0.8s	1s	1s
PhilsV3	464	161	212	80	1s	0.7s	0.8s	1.2s	1.1s
PhilsV4	482	161	464	161	10.7s	3.8s	4.5s	4.8s	4.8s

Table 4. Results of the Tabakov-Vardi experiments on two selected configurations. In each case, we generated 100 random automata and set the timeout to one hour. The new algorithm found more cases with simulation between initial states because the option `-qr` (do fw/bw quotienting repeatedly) may change the forward simulation in each iteration. In the "Hard" case, most of the timeout instances probably belong to the category "inclusion" (Inc).

	Hard: td=2, ad=0.1, size=30		Easy, but nontrivial: td=3, ad=0.6, size=50	
	The Algorithm of [1]	New Algorithm	The Algorithm of [1]	New Algorithm
Sim	1%, 32m42s	2%, 0.025s	13%, 2m5s	21%, 0.14s
Inc	16%, 43m	20%, 30m42s	68%, 26m14s	64%, 6m12s
nInc	49%, 0.17s	49%, 0.2s	15%, 0.3s	15%, 0.3s
TO	34%	29%	4%	0%

three classes, according to whether (1) simulation holds, (2) inclusion holds (but not simulation), and (3) inclusion does not hold. See, resp., Tables 1, 2, and 3. On average, the newly proposed approach using all the mentioned options produces the best result.

In the second experiment, we use the Tabakov-Vardi random model[7] with fixed alphabet size 2. There are two parameters, *transition density* (td; average number of transitions per state and alphabet symbol) and *acceptance density* (ad; percentage of accepting states). The results of a complete test for many parameter combinations and automata of size 15 can be found in Table 19 in Appendix H. Its results can be summarized as follows. In those cases where simulation holds between initial states, the time needed is negligible. Also the time needed to find counterexamples is very small. Only the "inclusion" cases are interesting. Based on the results in Table 19, we picked two configurations (Hard: td=2, ad=0.1, size=30) and (Easy, but nontrivial: td=3, ad=0.6, size=50) for an experiment with larger automata. Both configurations have a substantial percentage of the interesting "inclusion" cases. The results can be found in Table 4.

9 Conclusions

We have presented an efficient method for checking language inclusion for Büchi automata. It augments the basic Ramsey-based algorithm with several new techniques such as the use of weak subsumption relations based on combinations of forward and

[7] Note that automata generated by the Tabakov-Vardi model are very different from a control-flow graph of a program. They are almost unstructured, and thus on average the density of simulation is much lower. Hence, we believe it is not a fair evaluation benchmark for algorithms aimed at program verification. However, since it was used in the evaluation of most previous works on language inclusion testing, we also include it as one of the evaluation benchmarks.

backward simulation, the use of simulation relations between automata in order to limit the search space, and methods for eliminating redundant tests in the search procedure. We have performed a wide set of experiments to evaluate our approach, showing its practical usefulness. An interesting direction for future work is to characterize the roles of the different optimizations in different application domains. Although their overall effect is to achieve a much better performance compared to existing methods, the contribution of each optimization will obviously vary from one application to another. Such a characterization would allow a portfolio approach in which one can predict which optimization would be the dominant factor on a given problem. In the future, we also plan to implement both the latest rank-based and Ramsey-based approaches in a uniform way and thoroughly investigate how they behave on different classes of automata.

References

1. P. A. Abdulla, Y.-F. Chen, L. Clemente, L. Holík, C. D. Hong, R. Mayr, and T. Vojnar. Simulation Subsumption in Ramsey-Based Büchi Automata Universality and Inclusion Testing, *In Proc. of CAV'10*, LNCS, volume 6174, Springer, 2010.
2. P. A. Abdulla, Y.-F. Chen, L. Holík, R. Mayr, and T. Vojnar. When Simulation Meets Antichains: On Checking Language Inclusion of Nondeterministic Finite (Tree) Automata. In *Proc. of TACAS'10*, LNCS 6015. Springer, 2010.
3. D. Dill, A. Hu and H. Wong-Toi. Checking for language inclusion using simulation preorders. In *Proc. of CAV'92*, LNCS 575. Springer, 1992.
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, 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. In *Proc. of TACAS'10*, LNCS 6015. Springer, 2010.
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. 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.
12. O. Kupferman, M.Y. Vardi. Verification of fair transition systems. In *Proc. of CAV'96*, 1996.
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, M.Y. Vardi. Model Checking Büchi Specifications. In *Proc. of LATA'07*, 2007.
18. M. Y. Vardi and P. Wolper, An automata-theoretic approach to automatic program verification. In *Proc. of LICS'86*, IEEE Comp. Soc. Press, 1986.
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.

A Proofs for Section 3

In this appendix, we provide some more basic facts about the principles of Ramsey-based inclusion checking.

Lemma 16. *For proper edges x and y, $\mathcal{L}(x)\cdot\mathcal{L}(y)\subseteq\mathcal{L}(x)$ and $\mathcal{L}(y)\cdot\mathcal{L}(y)\subseteq\mathcal{L}(y)$.*

Proof. Immediate from the definition. □

For a countably infinite set A, let $\mathcal{H}(A)$ be the set of unordered pairs of elements from A, i.e., $\mathcal{H}(A)=\{\{x,y\}\mid x,y\in A\wedge x\neq y\}$. The following is a suitable version of the infinite Ramsey theorem. It says that for any finite coloring (partitioning) of $\mathcal{H}(A)$, there exists a complete and infinite monochromatic subset of $\mathcal{H}(A)$.

Lemma 17 (Infinite Ramsey Theorem). *Let A be a countably infinite set, and let $B=\mathcal{H}(A)$. For any partitioning of B into finitely many classes $B_1,\ldots,B_m$, there exists an infinite subset A' of A s.t. $\mathcal{H}(A')\subseteq B_k$, for some k.*

Lemma 18. *Let $v_0v_1\cdots\in\Sigma^\omega$, where each v_i is in Σ^+. Then, there exist graphs g,h s.t. $v_0v_1\cdots v_{i_0-1}\in\mathcal{L}(g)$, and $v_{i_k}v_{i_k+1}\cdots v_{i_{k+1}-1}\in\mathcal{L}(h)$ for any $k\geq 0$.*

Proof. Let $w=v_0v_1\cdots$ be an ω-word, where v_i is in Σ^+ for any $i\geq 0$. We have to show that it is possible to represent w as $z_0z_1\cdots$, with $z_0=v_0v_1\cdots v_{i_0-1}$ and $z_{k+1}=v_{i_k}v_{i_k+1}\cdots v_{i_{k+1}-1}$ for any $k\geq 0$, s.t. $z_0\in\mathcal{L}(g)$ and $z_{k+1}\in\mathcal{L}(h)$ for any $k\geq 0$, for some graphs g,h.

Consider prefixes $w_i=v_0\ldots v_i$ of w. Let $A=\{w_i\mid i\geq 0\}$. For any $w_i,w_j\in A$ with $i\leq j$, let $w_j\ominus w_i=v_{i+1}\ldots v_j$ (and define $w_j\ominus w_i$ as $w_i\ominus w_j$ if $i>j$), and let $B=\mathcal{H}(A)$ be the set of unordered pairs of strings from A. Each $w_i\ominus w_j$ belongs to the language of exactly one graph (since the languages of graphs partition Σ^+) and there are only finitely many graphs. Therefore, we can define the partitioning of $B=\bigcup_{h\in G}B_h$ 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 and an infinite subset A' of A 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 $\mathcal{L}(h)$. That is, it is possible to split the word w as follows:

$$w=v_0\ldots v_{i_0-1}\quad v_{i_0}\ldots v_{i_1-1}\quad v_{i_1}\ldots v_{i_2-1}\quad v_{i_2}\ldots$$

where $z_0:=v_0\ldots v_{i_0-1}$ is in $\mathcal{L}(g)$ for some graph g (which exists since graphs partition Σ^+), and $z_{k+1}:=v_{i_k}\ldots v_{i_{k+1}-1}$ is in $\mathcal{L}(h)$ for $k\geq 0$. □

Lemma 2. For proper edges (x,y) and $w\in Y_{xy}$, there exist graphs g,h s.t. $w\in Y_{\langle x,g\rangle\langle y,h\rangle}$.

Proof. Let $w\in Y_{xy}$. Therefore, it is possible to write w as $v_0v_1v_2\cdots$, where $v_0\in\mathcal{L}(x)$ and $v_{i+1}\in\mathcal{L}(y)$ for any $i\geq 0$. By Lemma 18, there exist graphs g,h s.t. $v_0v_1\cdots v_{i_0-1}\in\mathcal{L}(g)$, and $v_{i_k}v_{i_k+1}\cdots v_{i_k-1}\in\mathcal{L}(h)$ for any $k\geq 0$. By Lemma 16, $v_0v_1\cdots v_{i_0-1}\in\mathcal{L}(x)$ and $v_{i_k}v_{i_k+1}\cdots v_{i_k-1}\in\mathcal{L}(y)$ for any $k\geq 0$. Therefore, $w\in(\mathcal{L}(x)\cap\mathcal{L}(g))\cdot(\mathcal{L}(y)\cap\mathcal{L}(h))^\omega$. Let $\mathbf{g}=\langle x,g\rangle$ and $\mathbf{h}=\langle y,h\rangle$. By the definition of $Y_{\mathbf{gh}}$, $w\in Y_{\mathbf{gh}}$. □

Lemma 19 (Lemma 14 in [1]). *For $g,h\in G^f$, $LFT(g,h)$ iff $Y_{gh}\cap\mathcal{L}(\mathcal{B})\neq\emptyset$.*

Lemma 19 gives the basis for proving correctness of using DGT for testing language inclusion, which is stated by Lemma 4.

B Properties of graphs

The following auxiliary lemma has been proved in [7]. It is used in the proof of Lemma 5.

Lemma 20. *(Lemma 3.1.1 in [7])* $\forall g,h \in G : \mathcal{L}(g) \cdot \mathcal{L}(h) \subseteq \mathcal{L}(g;h)$.

Lemma 21. *For any* $f,g \in G$, $f \leqslant g \implies f \simeq_{\mathsf{fb}} g$ *and* $f \trianglelefteq g \implies f \simeq_{\mathsf{fb}} g$ *(or, equivalently,* $\leqslant \subseteq \simeq_{\mathsf{fb}}$ *and* $\trianglelefteq \subseteq \simeq_{\mathsf{fb}}$*).*

Proof. The first implication follows directly from the definition of $\leqslant$, since $\sqsubseteq \subseteq \sqsubseteq_{\mathsf{fb}}$. To show the second implication, let $\bar{f} \in G$ with $f \leqslant \bar{f} \simeq_{\mathsf{b}} g$ be a witness for $f \trianglelefteq g$. From the previous point, $f \simeq_{\mathsf{fb}} \bar{f}$. By the transitivity of $\simeq_{\mathsf{fb}}$ and $\simeq_{\mathsf{b}} \subseteq \simeq_{\mathsf{fb}}$, we have $f \simeq_{\mathsf{fb}} g$. □

Lemma 22. *Given* $g \in G^f$, $\langle p,a,q\rangle \in g$, *and* $r \in Q$, *it holds that*

1. *if* $p \preceq_{\mathsf{f}} r$ *then there is* $\langle r,a',q'\rangle \in g$ *such that* $q \preceq_{\mathsf{f}} q'$ *and* $a \leq a'$,
2. *if* $q \preceq_{\mathsf{b}} r$ *then there is* $\langle p',a',r\rangle \in g$ *such that* $p \preceq_{\mathsf{b}} p'$ *and* $a \leq a'$.

Proof. Point (1) of the lemma was shown in [1]. Point (2) can be proved analogously. □

The following lemma states that jumping composition is $\sqsubseteq$-monotone.

Lemma 23. *For graphs* $f,f',g,g' \in G$, *if* $f \sqsubseteq f'$ *and* $g \sqsubseteq g'$, *then* $f \fatsemi_{\mathsf{b}} g \sqsubseteq f' \fatsemi_{\mathsf{b}} g'$.

Proof. Let $\langle p,x,r\rangle \in f \fatsemi_{\mathsf{b}} g$. By the definition of composition, there exist arcs $\langle p,a,q\rangle \in f$ and $\langle q',b,r\rangle \in g$ with $x = \max(a,b)$ and $q \preceq_{\mathsf{b}} q'$. Since $f \sqsubseteq f'$, there exists an arc $\langle p,a',q\rangle \in f'$ with $a \leq a'$, and similarly, as $g \sqsubseteq g'$, there exists an arc $\langle q',b',r\rangle \in g'$ with $b \leq b'$. Take $y = \max(a',b')$. Clearly, $x \leq y$. By the def. of composition, there exists an arc $\langle p,y',r\rangle \in f' \fatsemi_{\mathsf{b}} g'$ with $x \leq y \leq y'$. □

The following lemma states that jumping composition is $\sqsubseteq_{\mathsf{b}}$-monotone.

Lemma 24. *For graphs* $f,f',g,g' \in G$, *if* $f \sqsubseteq_{\mathsf{b}} f'$ *and* $g \sqsubseteq_{\mathsf{b}} g'$, *then* $f \fatsemi_{\mathsf{b}} g \sqsubseteq_{\mathsf{b}} f' \fatsemi_{\mathsf{b}} g'$. *Moreover, if* $f' \in G^f$ *has non-empty language, then* $f \fatsemi_{\mathsf{b}} g \sqsubseteq_{\mathsf{b}} f';g'$.

Proof. Let $\langle p,x,r\rangle \in f \fatsemi_{\mathsf{b}} g$. By the definition of composition, there exist arcs $\langle p,a,q\rangle \in f$ and $\langle q',b,r\rangle \in g$ with $x = \max(a,b)$ and $q \preceq_{\mathsf{b}} q'$. Since $f \sqsubseteq_{\mathsf{b}} f'$, there exists an arc $\langle p',a',q\rangle \in f'$ with $a \leq a'$ and $p \preceq_{\mathsf{b}} p'$. Similarly, since $g \sqsubseteq_{\mathsf{b}} g'$, there exists an arc $\langle q'',b',r\rangle \in g'$ with $b \leq b'$ and $q \preceq_{\mathsf{b}} q' \preceq_{\mathsf{b}} q''$. Consequently, by the definition of composition, there exists an arc $\langle p',y,r\rangle \in f' \fatsemi_{\mathsf{b}} g'$ with $x \leq \max(a',b') \leq y$.

For the second part, further assume $f' \in G^f$ and recall $\langle p',a',q\rangle \in f'$, $\langle q'',b',r\rangle \in g'$ and $q \preceq_{\mathsf{b}} q''$. Then, by Lemma 22 (2), there exists an arc $\langle p'',a'',q''\rangle \in f'$ with $p' \preceq_{\mathsf{b}} p''$ and $a' \leq a''$. Thus, by the definition of composition, it follows that there exists an arc $\langle p'',y',r\rangle \in f';g'$ (i.e., no jumps), with $x \leq \max(a',b') \leq \max(a'',b') \leq y'$. □

Finally, the following lemma states a limited form of $\sqsubseteq_{\mathsf{fb}}$-monotonicity of composition.

Lemma 25. *For $f, f', g \in G$ and $g' \in G^f$, if $f' \sqsubseteq_{\mathsf{fb}} f$ and $g' \sqsubseteq_{\mathsf{fb}} g$, then $f';g' \sqsubseteq_{\mathsf{fb}} f \fatsemi_{\mathsf{b}} g$.*

Proof. Let $\langle p,x,r\rangle \in f';g'$. By the definition of composition, there exist arcs $\langle p,a,q\rangle \in f'$ and $\langle q,b,r\rangle \in g'$, with $x = \max(a,b)$. Since $f' \sqsubseteq_{\mathsf{fb}} f$, there exists an arc $\langle p',a',q'\rangle \in f$ s.t. $p \preceq_{\mathsf{b}} p'$, $a \leq a'$ and $q \preceq_{\mathsf{f}} q'$. Since $g' \in G^f$ has non-empty language, by Lemma 22 (1) there exists an arc $\langle q',b',r'\rangle \in g'$ s.t. $b \leq b'$ and $r \preceq_{\mathsf{f}} r'$. Since $g' \sqsubseteq_{\mathsf{fb}} g$, there exists an arc $\langle q'',b'',r''\rangle \in g$ s.t. $q \preceq_{\mathsf{b}} q' \preceq_{\mathsf{b}} q''$, $b \leq b' \leq b''$ and $r \preceq_{\mathsf{f}} r' \preceq_{\mathsf{f}} r''$. By the definition of composition, there exists an arc $\langle p',y,r''\rangle \in f \fatsemi_{\mathsf{b}} g$, with $x \leq \max(a',b'') \leq y$. □

We are now ready to prove a kind of monotonicity of composition w.r.t. $\trianglelefteq$.

Lemma 26. *For graphs $f, g \in G^R$ and $f', g' \in G^f$, if $f \trianglelefteq f'$ and $g \trianglelefteq g'$, then $f \fatsemi_{\mathsf{b}} g \trianglelefteq f';g'$.*

Proof. Assume $f \trianglelefteq f'$, $g \trianglelefteq g'$, with $f', g' \in G^f$. By the definition of $\trianglelefteq$, there are witnesses $\bar{f}$ and $\bar{g}$ such that $f \leqslant \bar{f} \simeq_{\mathsf{b}} f'$ and $g \leqslant \bar{g} \simeq_{\mathsf{b}} g'$. We prove the lemma by showing that $\bar{f} \fatsemi_{\mathsf{b}} \bar{g}$ is a witness for $f \fatsemi_{\mathsf{b}} g \trianglelefteq f';g'$, i.e., that $f \fatsemi_{\mathsf{b}} g \leqslant \bar{f} \fatsemi_{\mathsf{b}} \bar{g} \simeq_{\mathsf{b}} f';g'$.

The equivalence $\bar{f} \fatsemi_{\mathsf{b}} \bar{g} \simeq_{\mathsf{b}} f';g'$ is immediate by a double application of Lemma 24: $\bar{f} \fatsemi_{\mathsf{b}} \bar{g} \sqsubseteq_{\mathsf{b}} f';g' \sqsubseteq f' \fatsemi_{\mathsf{b}} g' \sqsubseteq_{\mathsf{b}} \bar{f} \fatsemi_{\mathsf{b}} \bar{g}$.

To show $f \fatsemi_{\mathsf{b}} g \leqslant \bar{f} \fatsemi_{\mathsf{b}} \bar{g}$, we apply the definition of $\leqslant$, and we verify a) $f \fatsemi_{\mathsf{b}} g \sqsubseteq \bar{f} \fatsemi_{\mathsf{b}} \bar{g}$ and b) $\bar{f} \fatsemi_{\mathsf{b}} \bar{g} \sqsubseteq_{\mathsf{fb}} f \fatsemi_{\mathsf{b}} g$. We first prove Point a): From $g \leqslant \bar{g}$ and $f \leqslant \bar{f}$ we have, by the definition of $\leqslant$, $g \sqsubseteq \bar{g}$ and $f \sqsubseteq \bar{f}$. Then, $f \fatsemi_{\mathsf{b}} g \sqsubseteq \bar{f} \fatsemi_{\mathsf{b}} \bar{g}$ follows by the $\sqsubseteq$-monotonicity of composition (by Lemma 23).

We now prove Point b). In the first part, we have already proved the equivalence $\bar{f} \fatsemi_{\mathsf{b}} \bar{g} \simeq_{\mathsf{b}} f';g'$. Since $\sqsubseteq_{\mathsf{b}} \subseteq \sqsubseteq_{\mathsf{fb}}$, it suffices to show $f';g' \sqsubseteq_{\mathsf{fb}} f \fatsemi_{\mathsf{b}} g$. The latter claim follows from Lemma 25, since $f' \sqsubseteq_{\mathsf{fb}} f$ and $g' \sqsubseteq_{\mathsf{fb}} g$ by Lemma 21. □

Lemma 27. *For any graphs $f \in G$, $g \in G^R$, and $h \in G^f$ such that $f \leqslant g$ and $g \trianglelefteq h$, it holds that $f \trianglelefteq h$.*

Proof. By the definition of $\trianglelefteq$, there is $\bar{g} \in G$ with $g \leqslant \bar{g} \simeq_{\mathsf{b}} h$. Since $\leqslant$ is transitive, we have $f \leqslant \bar{g} \simeq_{\mathsf{b}} h$, that is, $f \trianglelefteq h$. □

The following lemma states that the jumping lasso finding test is $\sqsubseteq_{\mathsf{b}}$-monotone.

Lemma 28. *For graphs $f, g, f', g' \in G$ with $f \sqsubseteq_{\mathsf{b}} f'$ and $g \sqsubseteq_{\mathsf{b}} g'$, $LFT_{\mathsf{b}}(f,g) \implies LFT_{\mathsf{b}}(f',g')$.*

Proof. Let $\langle p,a_0,q_0\rangle \in f, \langle q'_0,a_1,q_1\rangle \in g, \langle q'_1,a_2,q_2\rangle \in g, \ldots$ be the sequence of arcs witnessing $LFT_{\mathsf{b}}(f,g)$, where, in particular, $q_i \preceq_{\mathsf{b}} q'_i$ for any $i \geq 0$ and $p \in I_{\mathcal{B}}$. Since $f \sqsubseteq_{\mathsf{b}} f'$, there exists an arc $\langle p',a'_0,q_0\rangle \in f'$ with $a_0 \leq a'_0$, $p \preceq_{\mathsf{b}} p'$ and $p' \in I_{\mathcal{B}}$ by the def. of $\preceq_{\mathsf{b}}$. Since $g \sqsubseteq_{\mathsf{b}} g'$, for any $i \geq 0$, there exists an arc $\langle q''_i,a'_{i+1},q_{i+1}\rangle$ with $q_i \preceq_{\mathsf{b}} q'_i \preceq_{\mathsf{b}} q''_i$ and $a_{i+1} \leq a'_{i+1}$. Therefore, the sequence $\langle p',a'_0,q_0\rangle \in f', \langle q''_0,a'_1,q_1\rangle \in g', \langle q''_1,a'_2,q_2\rangle \in g', \ldots$ is a witness for $LFT_{\mathsf{b}}(f',g')$. □

The following lemma states that the jumping lasso finding test is redundant on graphs with non-empty language.

Lemma 29. *For graphs $f', g' \in G^f$ with non-empty language, $LFT_{\mathsf{b}}(f',g') \implies LFT(f',g')$.*

Proof. Let $\langle p, a_0, q_0\rangle \in f', \langle q'_0, a_1, q_1\rangle \in g', \langle q'_1, a_2, q_2\rangle \in g', \ldots$ be the sequence of arcs witnessing $LFT_{\mathsf{b}}(f', g')$, i.e., $p \in I$, $q_i \preceq_{\mathsf{b}} q'_i$ and $a_j = 1$ for infinitely many j's. We proceed in two steps: (1) we show that there are longer and longer finite paths with arbitrarily many occurrences of 1-arcs, and (2) we show the existence of a single infinite path infinitely many 1-arcs.

For Step 1, we proceed by induction, using the properties of backward simulation. We prove the following claim: For every $n \geq 0$ and every s_n with $q_n \preceq_{\mathsf{b}} s_n$, there exists a sequence of arcs

$$\langle r, b_0, s_0\rangle \in f', \langle s_0, b_1, s_1\rangle \in g', \langle s_1, b_2, s_2\rangle \in g', \ldots, \langle s_{n-1}, b_n, s_n\rangle \in g'$$

where $p \preceq_{\mathsf{b}} r$ (thus $r \in I$), $a_i \leq b_i$, and $q_i \preceq_{\mathsf{b}} s_i$ for all i.

Let $n = 0$. Since $\langle p, a_0, q_0\rangle \in f'$ and $q_0 \preceq_{\mathsf{b}} s_0$ by assumption, by Lemma 22 (2), there is $\langle r, b_0, s_0\rangle \in f'$ with $p \preceq_{\mathsf{b}} r$ and $a_0 \leq b_0$. For $n > 0$, we proceed similarly. Since $\langle q_{n-1}, a_n, q_n\rangle \in g'$ and $q_n \preceq_{\mathsf{b}} s_n$ by assumption, by Lemma 22 (2), there is $\langle s_{n-1}, b_n, s_n\rangle \in g'$ with $q_{n-1} \preceq_{\mathsf{b}} s_{n-1}$ and $a_n \leq b_n$. By induction hypothesis, there exists a sequence of arcs

$$\langle r, b_0, s_0\rangle \in f', \langle s_0, b_1, s_1\rangle \in g', \langle s_1, b_2, s_2\rangle \in g', \ldots, \langle s_{n-2}, b_{n-1}, s_{n-1}\rangle \in g'$$

where $p \preceq_{\mathsf{b}} r$ (thus $r \in I$), $a_i \leq b_i$, and $q_i \preceq_{\mathsf{b}} s_i$. By extending this sequence with the arc $\langle s_{n-1}, b_n, s_n\rangle \in g'$ found above, we have shown the claim.

For Step 2, it is enough to notice that there are only finitely many different arcs in g'. Therefore, there exists n sufficiently large s.t. an arc $\langle s_i, b_{i+1}, s_{i+1}\rangle \in g'$ in the sequence above repeats twice (and it can be chosen with $b_{i+1} = 1$). Thus, the required infinite path may be obtained by repeating infinitely often the appropriate sequence of arcs. This shows $LFT(f', g')$. □

The following lemma states a limited form of $\sqsubseteq_{\mathsf{fb}}$-monotonicity of the jumping lasso finding test.

Lemma 30. *For graphs $f', g' \in G^f$ and $\hat{f}, \hat{g} \in G$ s.t. $f' \sqsubseteq_{\mathsf{fb}} \hat{f}$ and $g' \sqsubseteq_{\mathsf{fb}} \hat{g}$, $LFT(f', g') \Longrightarrow LFT_{\mathsf{b}}(\hat{f}, \hat{g})$.*

Proof. Let $\langle p, a_0, q_0\rangle \in f', \langle q_0, a_1, q_1\rangle \in g', \langle q_1, a_2, q_2\rangle \in g', \ldots$ be a sequence of arcs witnessing $LFT(f', g')$, i.e., $p \in I$ and $a_j = 1$ for infinitely many j's. We show that there exists a sequence $\langle r, b_0, s_0\rangle \in \hat{f}, \langle s'_0, b_1, s_1\rangle \in \hat{g}, \langle s'_1, b_2, s_2\rangle \in \hat{g}, \ldots$ witnessing $LFT_{\mathsf{b}}(\hat{f}, \hat{g})$, s.t. $p \preceq_{\mathsf{b}} r$ (implying $r \in I$), $a_i \leq b_i$, and $s_i \preceq_{\mathsf{b}} s'_i$, with the additional property $q_i \preceq_{\mathsf{f}} s_i$ needed within the induction argument. Since $f' \sqsubseteq_{\mathsf{fb}} \hat{f}$, there exists an arc $\langle r, b_0, s_0\rangle \in \hat{f}$ s.t. $p \preceq_{\mathsf{b}} r$, $a_0 \leq b_0$, and $q_0 \preceq_{\mathsf{f}} s_0$ starting the sequence. We show that for any $n \geq 0$, if the prefix $\langle r, b_0, s_0\rangle \in \hat{f}, \langle s'_0, b_1, s_1\rangle \in \hat{g}, \ldots, \langle s'_{n-1}, b_n, s_n\rangle \in \hat{g}$ of the sequence exists, then it can be extended by one arc.

By Lemma 22 (1) and the assumptions $q_n \preceq_{\mathsf{f}} s_n$ and $\langle q_n, a_{n+1}, q_{n+1}\rangle \in g'$, there is $\langle s_n, \bar{a}_{n+1}, \bar{q}_{n+1}\rangle \in g'$ with $a_{n+1} \leq \bar{a}_{n+1}$ and $q_{n+1} \preceq_{\mathsf{f}} \bar{q}_{n+1}$. Then, since $g' \sqsubseteq_{\mathsf{fb}} \hat{g}$, there is $\langle s'_n, b_{n+1}, s_{n+1}\rangle \in \hat{g}$ such that $s_n \preceq_{\mathsf{b}} s'_n$, $\bar{a}_{n+1} \leq b_{n+1}$, and $\bar{q}_{n+1} \preceq_{\mathsf{f}} s_{n+1}$. We obtain $a_{n+1} \leq b_{n+1}$ and $q_{n+1} \preceq_{\mathsf{f}} s_{n+1}$ by transitivity, which concludes the proof. □

Lemma 31. *For any $f, g \in G^R$ and $f', g' \in G^f$ such that $f \trianglelefteq f'$ and $g \trianglelefteq g'$, $LFT_{\mathsf{b}}(f, g) \Longleftrightarrow LFT(f', g')$.*

Proof. The "only if" direction follows from Lemmas 28 and 29, by recalling that $\trianglelefteq \subseteq \sqsubseteq_{\mathsf{b}}$. The "if" direction follows from Lemma 30, since $\trianglelefteq \subseteq \simeq_{\mathsf{fb}}$ (by Lemma 21). □

Lemma 32. *Given $f,g \in G^R$ and $\hat{f},\hat{g} \in G$ where $f \sqsubseteq_{\mathsf{fb}} \hat{f}$ and $g \sqsubseteq_{\mathsf{fb}} \hat{g}$, it holds that $LFT_{\mathsf{b}}(f,g) \implies LFT_{\mathsf{b}}(\hat{f},\hat{g})$.*

Proof. By the definition of G^R, there are $f',g' \in G^f$ with $f \trianglelefteq f'$ and $g \trianglelefteq g'$. From Lemma 31, we obtain $LFT(f',g')$. Since $\trianglelefteq \subseteq \simeq_{\mathsf{fb}}$ and $f \sqsubseteq_{\mathsf{fb}} \hat{f}, g \sqsubseteq_{\mathsf{fb}} \hat{g}$, by transitivity, we get $f' \sqsubseteq_{\mathsf{fb}} \hat{f}$ and $g' \sqsubseteq_{\mathsf{fb}} \hat{g}$. Thus, $LFT_{\mathsf{b}}(\hat{f},\hat{g})$ by Lemma 30. □

Lemma 33. *Given graphs $f,g \in G^R$ and $\hat{f},\hat{g} \in G$ such that $f \sqsubseteq_{\mathsf{fb}} \hat{f}$ and $g \sqsubseteq_{\mathsf{fb}} \hat{g}$, it holds that $f \fatsemi_{\mathsf{b}} g \sqsubseteq_{\mathsf{fb}} \hat{f} \fatsemi_{\mathsf{b}} \hat{g}$.*

Proof. Since $f,g \in G^R$, there exist graphs $f',g' \in G^f$ s.t. $f \trianglelefteq f'$ and $g \trianglelefteq g'$. By Lemma 21, $f \trianglelefteq f' \sqsubseteq_{\mathsf{fb}} \hat{f}$ and $g \trianglelefteq g' \sqsubseteq_{\mathsf{fb}} \hat{g}$. Then, $f \fatsemi_{\mathsf{b}} g \trianglelefteq f';g' \sqsubseteq_{\mathsf{fb}} \hat{f} \fatsemi_{\mathsf{b}} \hat{g}$, where the former relation follows from Lemma 26 and the latter from Lemma 25. Therefore, by Lemma 21, $f \fatsemi_{\mathsf{b}} g \sqsubseteq_{\mathsf{fb}} \hat{f} \fatsemi_{\mathsf{b}} \hat{g}$. □

C Proofs for Section 4

The lemma below is used to show correctness of weak properness even when *delayed simulation* is used. For the notion of delayed simulation used in the lemma below, please refer to [6]. As usual, for a state q, define its language $\mathcal{L}(q) = \{w \in \Sigma^\omega \mid$ there exists an accepting run on w starting from $q\}$.

Lemma 34. *If $q_1 \overset{w}{\rightsquigarrow}_F q_2$ for some $w \in \Sigma^+$, and $q_2 \succeq^{de}_{\mathcal{A}} q_1$ then $w^\omega \in \mathcal{L}(q_1) \cap \mathcal{L}(q_2)$.*[8]

Proof. We assume without loss of generality that $q_1 \in F$. Indeed, if $q_1 \notin F$, then, by $q_1 \overset{w}{\rightsquigarrow}_F q_2$, there exists $q_1' \in F$ s.t. $q_1 \overset{u}{\rightsquigarrow} q_1' \overset{v}{\rightsquigarrow} q_2$, with $w = uv$ and $u \neq \varepsilon$. Then, since $q_2 \succeq^{de}_{\mathcal{A}} q_1$, there exists q_2' s.t. $q_2 \overset{u}{\rightsquigarrow} q_2' \succeq^{de}_{\mathcal{A}} q_1'$. Thus, if we let $w' = vu$, we have $q_1' \overset{w'}{\rightsquigarrow} q_2'$ and $q_2' \succeq^{de}_{\mathcal{A}} q_1' \in F$. Now, by invoking the lemma we have $w'^\omega \in L(q_1') \cap L(q_2')$. But $w^\omega = (uv)(uv)(uv)\cdots = u(vu)(vu)\cdots = uw'^\omega$, therefore, $w \in L(q_1) \cap L(q_2)$.

Now, let $q_1 \overset{w}{\rightsquigarrow} q_2$, with $q_1 \in F$ and $q_2 \succeq^{de}_{\mathcal{A}} q_1$. We explain the intuition behind the proof by using the metaphor of simulation games. In the simulation game between q_1 and q_2 there are two players, the attacker (moving from q_1) and the defender (moving from q_2). Intuitively, the attacker and the defender alternate in choosing successors, and they build two infinite paths: The attacker chooses successors starting from q_1 (resulting in the infinite path π^A), while the defender replies by choosing successors starting from q_2 (resulting in the infinite path π^D). In delayed simulation the winning condition requires that whenever a state in π is accepting (say at round k), then it is the case that there exists a round $k' \geq k$ s.t. the state in π' at round k' is accepting as well. Since $q_2 \succeq^{de}_{\mathcal{A}} q_1$, then (by definition) the defender has a winning strategy in the simulation

[8] It is possible to generalize the lemma by replacing delayed simulation with k-pebble delayed simulation [5].

game between q_1 and q_2, i.e., a strategy which is winning against *any* attacker's strategy. Throughout the rest of the proof, we therefore assume that the defender is using such a winning strategy.

The simulation game is actually played as follows. The attacker first plays $q_1 \overset{w}{\rightsquigarrow} q_2$, and the defender responds by $q_2 \overset{w}{\rightsquigarrow} q_3$, for some $q_3 \succeq^{de}_{\mathcal{A}} q_2$, Then, the attacker plays $q_2 \overset{w}{\rightsquigarrow} q_3$, imitating the defender's previous moves, and the defender responds by $q_3 \overset{w}{\rightsquigarrow} q_4$, for some $q_4 \succeq^{de}_{\mathcal{A}} q_3$, and so on. On "doomsday", the attacker builds the infinite sequence $\pi^A = q_1 \overset{w}{\rightsquigarrow} q_2 \overset{w}{\rightsquigarrow} q_3 \overset{w}{\rightsquigarrow} q_4 \cdots$, and the defender builds the infinite sequence $\pi^D = q_2 \overset{w}{\rightsquigarrow} q_3 \overset{w}{\rightsquigarrow} q_4 \overset{w}{\rightsquigarrow} q_5 \cdots$. If $w = a_1 a_1 \cdots a_h$, then we can rewrite π^A as $\pi^A = q_1 \overset{a_1}{\rightsquigarrow} q_1^1 \overset{a_2}{\rightsquigarrow} q_1^2 \overset{a_3}{\rightsquigarrow} \cdots \overset{a_h}{\rightsquigarrow} q^2 \overset{a_1}{\rightsquigarrow} q_2^1 \cdots$, for some intermediate states $q_1^1, q_1^2, \ldots$, and by renaming states sequentially as $s_1, s_2, s_3, \ldots$, and the input symbols as $w^\omega = b_1 b_2 b_3 \cdots$, we obtain:

$$\pi^A = s_1 \overset{b_1}{\rightsquigarrow} s_2 \overset{b_2}{\rightsquigarrow} s_3 \cdots \tag{1}$$

$$\pi^D = s_h \overset{b_{h+1}}{\rightsquigarrow} s_{h+2} \overset{b_{h+2}}{\rightsquigarrow} s_{h+3} \cdots \tag{2}$$

Since s_1 $(= q_1)$ is accepting (in π^A) at round 1, and the defender is playing according to a winning strategy, it is the case that there exists $k_1 \geq 0$ s.t. s_{h+k_1} is accepting (in π^D) at round k_1. But now s_{h+k_1} is also accepting (in π^A) at a later round $h + k_1 > k_1$. Therefore, there exists $k_2 \geq 0$ s.t. $s_{h+k_1+k_2}$ is accepting (in π^D) at round $k_1 + k_2$, and so on. It is easy to see that this mechanism guarantees that infinitely many s_i are accepting. Therefore, the sequence $s_1 s_2 s_3 \cdots$ is an accepting run over w^ω from q_1 $(= s_1)$, and the sequence $s_h s_{h+1} s_{h+2} \cdots$ is an accepting run over w^ω from q_2 $(= s_h)$. Therefore, $w^\omega \in L(q_1) \cap L(q_2)$. □

A variant of Lemma 7 was shown in [1]. In our setting, however, the edge-part of a supergraph carries a label, while this was not the case in [1]. We prove that weak properness is sound even in our more general setting. The proof of Lemma 7 relies on Lemma 34, which allows us to prove that weak properness is sound even when based on delayed simulation.

Lemma 7. $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$ iff for all $\mathbf{g}, \mathbf{h} \in S^f$, $RDGT(\mathbf{g}, \mathbf{h})$.

Proof. We show instead that there is a pair of supergraphs $\mathbf{g}, \mathbf{h}$ in S^f that fails RDGT iff $\mathcal{L}(\mathcal{A}) \nsubseteq \mathcal{L}(\mathcal{B})$. In the following, let $\mathbf{g} = \langle \langle p_\mathbf{g}, l_\mathbf{g}, q_\mathbf{g} \rangle, g \rangle$ and $\mathbf{h} = \langle \langle p_\mathbf{h}, l_\mathbf{h}, q_\mathbf{h} \rangle, h \rangle$.

First, assume that $\langle \mathbf{g}, \mathbf{h} \rangle$ fails the relaxed double-graph test for some $\mathbf{g}, \mathbf{h} \in S^f$. This means that $\langle \mathbf{g}, \mathbf{h} \rangle$ is weakly proper (in the sense of Definition 4), and that $\langle g, h \rangle$ fails the lasso finding test. Let $Y_\mathbf{gh}$ be the ω-regular language $\mathcal{L}(\mathbf{g}) \cdot \mathcal{L}(\mathbf{h})^\omega$. $Y_\mathbf{gh}$ is non-empty because $\mathbf{g}, \mathbf{h} \in S^f$.

By the definition of being weakly proper, we have $l_\mathbf{h} = 1$. Since $\mathcal{L}(\mathbf{h}) \neq \emptyset$, there is some $w \in \mathcal{L}(\mathbf{h})$ s.t. $p_\mathbf{h} \overset{w}{\rightsquigarrow} q_\mathbf{h}$, and at least one accepting state from $F_\mathcal{A}$ is visited on the path. Since we also have $q_\mathbf{h} \succeq^\mathcal{A}_\mathsf{f} p_\mathbf{h}$, Lemma 34 yields that $w^\omega \in \mathcal{L}(p_\mathbf{h})$. From $q_\mathbf{g} \succeq^\mathcal{A}_\mathsf{f} p_\mathbf{h}$ we obtain that $w^\omega \in \mathcal{L}(q_\mathbf{g})$. Since $\mathcal{L}(\mathbf{g}) \neq \emptyset$, there is some $w' \in \mathcal{L}(\mathbf{g})$ s.t. $p_\mathbf{g} \overset{w'}{\rightsquigarrow} q_\mathbf{g}$. Therefore $w'w^\omega \in \mathcal{L}(p_\mathbf{g})$. Since $p_\mathbf{g} \in I_\mathcal{A}$ we have $w'w^\omega \in \mathcal{L}(\mathcal{A})$. Furthermore, $w'w^\omega \in Y_\mathbf{gh}$ and thus $Y_\mathbf{gh} \cap \mathcal{L}(\mathcal{A}) \neq \emptyset$.

Since $\neg LFT(g,h)$, by Lemma 19, we obtain $Y_{gh} \cap \mathcal{L}(\mathcal{B}) = \emptyset$ and thus, by the definition of the languages of supergraphs, $Y_{\mathbf{gh}} \cap \mathcal{L}(\mathcal{B}) = \emptyset$. Since $Y_{\mathbf{gh}} \neq \emptyset$, we finally have $\mathcal{L}(\mathcal{A}) \not\subseteq \mathcal{L}(\mathcal{B})$.

For the other direction assume $\mathcal{L}(\mathcal{A}) \not\subseteq \mathcal{L}(\mathcal{B})$. Then, by Lemma 4, there is a pair of supergraphs $\mathbf{g}, \mathbf{h}$ in S^f s.t. $\neg DGT(\mathbf{g},\mathbf{h})$. This implies that $(\mathbf{g},\mathbf{h})$ is proper and $\neg LFT(g,h)$. Since properness implies weak properness (by Definition 4), $(\mathbf{g},\mathbf{h})$ is weakly proper. Therefore, we obtain $\neg RDGT(\mathbf{g},\mathbf{h})$. □

Lemma 8. For any $\mathbf{g},\mathbf{h} \in S^R$ and $\mathbf{g}',\mathbf{h}' \in S^f$ such that $\mathbf{g} \trianglelefteq \mathbf{g}'$ and $\mathbf{h} \trianglelefteq \mathbf{h}'$, it holds that $RDGT_{\mathsf{b}}(\mathbf{g},\mathbf{h}) \iff RDGT(\mathbf{g}',\mathbf{h}')$.

Proof. Let $\mathbf{g} = \langle x,g\rangle, \mathbf{h} = \langle y,h\rangle, \mathbf{g}' = \langle x',g'\rangle, \mathbf{h}' = \langle y',h'\rangle$ with $\mathbf{g} \trianglelefteq \mathbf{g}'$ and $\mathbf{h} \trianglelefteq \mathbf{h}'$. By the definition of $\trianglelefteq$, we have $x = x'$, $y = y'$, $g \trianglelefteq g'$, $h \trianglelefteq h'$. Therefore, $(\mathbf{g},\mathbf{h})$ is weakly proper iff $(\mathbf{g}',\mathbf{h}')$ is weakly proper. Lemma 31 says that $LFT_{\mathsf{b}}(g,h)$ iff $LFT(g',h')$. The statement of the lemma thus follows immediately. □

In the subsequent proofs, we use the following immediate consequence of Lemma 7 and Lemma 8.

Lemma 35. *$\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$ iff $RDGT_{\mathsf{b}}(\mathbf{g},\mathbf{h})$ for all $\mathbf{g},\mathbf{h} \in S^f$.*

Lemma 5. For supergraphs $\mathbf{g},\mathbf{h} \in S^R$ and $\mathbf{g}',\mathbf{h}' \in S^f$, if $\mathbf{g} \trianglelefteq \mathbf{g}'$, $\mathbf{h} \trianglelefteq \mathbf{h}'$ and $\mathbf{g}',\mathbf{h}'$ are composable, then $\mathbf{g},\mathbf{h}$ are composable and $\mathbf{g} \fatsemi_{\mathsf{b}} \mathbf{h} \trianglelefteq \mathbf{g}';\mathbf{h}'$ and $\mathbf{g} \fatsemi_{\mathsf{b}} \mathbf{h} \in S^R$.

Proof. Let $\mathbf{g} = \langle x,g\rangle$ with $x = \langle p,a,q\rangle$, and $\mathbf{h} = \langle y,h\rangle$ with $y = \langle q',b,r\rangle$. By the definition of $\trianglelefteq$, we have $\mathbf{g}' = \langle x,g'\rangle$ and $\mathbf{h}' = \langle y,h'\rangle$, where $g',h' \in G^f$ and $g \trianglelefteq g'$, $h \trianglelefteq h'$. Since $\mathbf{g}',\mathbf{h}'$ are composable, we have $q = q'$. Clearly, $\mathbf{g}$ and $\mathbf{h}$ are composable as well. We now show $\mathbf{g} \fatsemi_{\mathsf{b}} \mathbf{h} \trianglelefteq \mathbf{g}';\mathbf{h}'$. By the definitions of $\fatsemi_{\mathsf{b}}$ and ; on supergraphs, we have $\mathbf{g} \fatsemi_{\mathsf{b}} \mathbf{h} = \langle x;y, g \fatsemi_{\mathsf{b}} h\rangle$ and $\mathbf{g}';\mathbf{h}' = \langle x;y,g';h'\rangle$. Therefore, by the definition of $\trianglelefteq$, it suffices to show $g \fatsemi_{\mathsf{b}} h \trianglelefteq g';h'$. But this follows from the assumptions and by Lemma 26.

For completeness, we show that $\mathbf{g}';\mathbf{h}' \in S^f$, so that $\mathbf{g} \fatsemi_{\mathsf{b}} \mathbf{h} \in S^R$. By the definition of S^f and as $\mathbf{g}',\mathbf{h}' \in S^f$, there are words $w_1 \in \mathcal{L}(x) \cap \mathcal{L}(g')$ and $w_2 \in \mathcal{L}(y) \cap \mathcal{L}(h')$. The word $w_1 \cdot w_2$ must also be in $\mathcal{L}(x;y)$, and, by Lemma 20, the word $w_1 \cdot w_2$ is in $\mathcal{L}(g';h')$ as well. This means that $\mathcal{L}(g';h') \cap \mathcal{L}(x;y) \neq \emptyset$, and thus $\mathbf{g}';\mathbf{h}' \in S^f$. □

Lemma 6. Let $\mathbf{f} \in S$, $\mathbf{g} \in S^R$, and $\mathbf{h} \in S^f$. If $\mathbf{f} \leqslant \mathbf{g}$ and $\mathbf{g} \trianglelefteq \mathbf{h}$, then $\mathbf{f} \trianglelefteq \mathbf{h}$ (and thus $\mathbf{f} \in S^R$). In particular, the statement holds when $\mathbf{f} = Min(\mathbf{g})$.

Proof. The statement follows directly from the definition of $\trianglelefteq$ and transitivity of $\leqslant$. □

Lemma 36. *For any $\mathbf{f},\mathbf{g} \in S$, $\mathbf{f} \leqslant \mathbf{g} \implies \mathbf{f} \simeq_{\mathsf{fb}} \mathbf{g}$ and $\mathbf{f} \trianglelefteq \mathbf{g} \implies \mathbf{f} \simeq_{\mathsf{fb}} \mathbf{g}$ (or, equivalently, $\leqslant \subseteq \simeq_{\mathsf{fb}}$ and $\trianglelefteq \subseteq \simeq_{\mathsf{fb}}$ when interpreted on supergraphs).*

Proof. Immediate from Lemma 21 and the definition of $\leqslant$. □

Lemma 9. For supergaphs $\mathbf{g},\mathbf{h} \in S^R$ and $\mathbf{g}',\mathbf{h}' \in S$, if $\mathbf{g} \sqsubseteq_{\mathsf{fb}} \mathbf{g}'$ and $\mathbf{h} \sqsubseteq_{\mathsf{fb}} \mathbf{h}'$, then $RDGT_{\mathsf{b}}(\mathbf{g},\mathbf{h}) \Rightarrow RDGT_{\mathsf{b}}(\mathbf{g}',\mathbf{h}')$.

Proof. We equivalently show that if $(\mathbf{g}',\mathbf{h}')$ fails the relaxed double-graph test, then $(\mathbf{g},\mathbf{h})$ fails the relaxed double-graph test as well. In the following, let $\mathbf{g}' = \langle\langle p,a',q'\rangle,g'\rangle$ and $\mathbf{h} = \langle\langle r,b',s'\rangle,h'\rangle$. Assume that $\langle\mathbf{g}',\mathbf{h}'\rangle$ fails the relaxed double-graph test. Then, $(\mathbf{g}',\mathbf{h}')$ is weakly proper, i.e., $q' \preceq_{\mathsf{f}} r$, $s' \preceq_{\mathsf{f}} r$, $p \in I_{\mathcal{A}}$ and $b' = 1$, and $\neg LFT_{\mathsf{b}}(g',h')$.

From $\mathbf{g} \sqsubseteq_{\mathsf{fb}} \mathbf{g}'$ and $\mathbf{h} \sqsubseteq_{\mathsf{fb}} \mathbf{h}'$, we have that $\mathbf{g} = \langle\langle p,a,q\rangle,g\rangle$ and $\mathbf{f} = \langle\langle r,b,s\rangle,h\rangle$ where $a \geq a'$, $q \preceq_{\mathsf{f}} q'$, and $b \geq b'$, $s \preceq_{\mathsf{f}} s'$.

Since $b' = 1$, we obtain $b = 1$. Furthermore, $p \preceq_{\mathsf{f}} p' \preceq_{\mathsf{f}} r$ and $s \preceq_{\mathsf{f}} s' \preceq_{\mathsf{f}} r$. This means exactly that the pair $(\mathbf{g},\mathbf{h})$ is weakly proper. It remains to show that $\neg LFT_{\mathsf{b}}(g,h)$.

Since $\mathbf{g},\mathbf{h} \in S^R$ we have $g,h \in G^R$, and since $\mathbf{g} \sqsubseteq_{\mathsf{fb}} \mathbf{g}'$ and $\mathbf{h} \sqsubseteq_{\mathsf{fb}} \mathbf{h}'$, we have have $g \sqsubseteq_{\mathsf{fb}} g'$ and $h \sqsubseteq_{\mathsf{fb}} h'$. Thus, from $\neg LFT_{\mathsf{b}}(g',h')$ and Lemma 32, we obtain $\neg LFT_{\mathsf{b}}(g,h)$. Hence, $(\mathbf{g},\mathbf{h})$ fails the relaxed double-graph test, which concludes the proof. □

Lemma 10. For any $\mathbf{g},\mathbf{g}' \in S^R$ with $\mathbf{g} \sqsubseteq_{\mathsf{fb}} \mathbf{g}'$ and $\mathbf{h}' \in Min(S^1)$ such that $\mathbf{g}'$ and $\mathbf{h}'$ are composable, there exists $\hat{\mathbf{h}} \in Min(S^1)$ such that for all $\mathbf{h} \in S^R$ with $\mathbf{h} \sqsubseteq_{\mathsf{fb}} \hat{\mathbf{h}}$, $\mathbf{g}$ is composable with $\mathbf{h}$ and $\mathbf{g} \fatsemi_{\mathsf{b}} \mathbf{h} \sqsubseteq_{\mathsf{fb}} \mathbf{g}' \fatsemi_{\mathsf{b}} \mathbf{h}'$.

Proof. Let $\mathbf{g} = \langle x,g\rangle, \mathbf{g}' = \langle x',g'\rangle \in S^R$, and $\mathbf{h}' = \langle y',h'\rangle \in Min(S^1)$. Let $\mathbf{g}',\mathbf{h}'$ composable and $\mathbf{g} \sqsubseteq_{\mathsf{fb}} \mathbf{g}'$. Therefore, $g \sqsubseteq_{\mathsf{fb}} g'$, and the arcs x, x' and y' take the following form: $x = \langle p,a,q\rangle$, $x' = \langle p,a',q'\rangle$ and $y' = \langle q',b',r'\rangle$, with $x' \sqsubseteq_{\mathsf{f}} x$, i.e., $a' \leq a$ and $q' \preceq_{\mathsf{f}} q$. We prove that 1) there exists $\hat{\mathbf{h}} \in S^1$ s.t. $\mathbf{g},\hat{\mathbf{h}}$ are composable and $\mathbf{g} \fatsemi_{\mathsf{b}} \hat{\mathbf{h}} \sqsubseteq_{\mathsf{fb}} \mathbf{g}' \fatsemi_{\mathsf{b}} \mathbf{h}'$, and 2) the same is true for any representative $\mathbf{h}$ subsuming $\hat{\mathbf{h}}$, that is, for every $\mathbf{h} \in S^R$ s.t. $\mathbf{h} \sqsubseteq_{\mathsf{fb}} \hat{\mathbf{h}}$, $\mathbf{g},\mathbf{h}$ are composable and $\mathbf{g} \fatsemi_{\mathsf{b}} \mathbf{h} \sqsubseteq_{\mathsf{fb}} \mathbf{g}' \fatsemi_{\mathsf{b}} \mathbf{h}'$.

We first show Point 1). Since $\mathbf{h}'$ is in $Min(S^1)$, there exists $\mathbf{h}'' \in S^1$ s.t. $\mathbf{h}' \trianglelefteq \mathbf{h}''$. By the definition of $\trianglelefteq$, we have $\mathbf{h}'' = \langle y',\hat{h}\rangle$ where $h' \trianglelefteq \hat{h}$. Since $q' \preceq_{\mathsf{f}} q$, by a reasoning analogous to Lemma 22 (1), there is an arc $\hat{y} = \langle q,\hat{b},\hat{r}\rangle \in E_{\mathcal{A}}$, with $b' \leq \hat{b}$ and $r' \preceq_{\mathsf{f}} \hat{r}$, s.t. the supergraph $\hat{\mathbf{h}} = \langle \hat{y},\hat{h}\rangle$ has non-empty language. Clearly, $\mathbf{g},\hat{\mathbf{h}}$ are composable, and their composition is $\mathbf{g} \fatsemi_{\mathsf{b}} \hat{\mathbf{h}} = \langle x;\hat{y}, g \fatsemi_{\mathsf{b}} \hat{h}\rangle$. Since $\mathbf{g}' \fatsemi_{\mathsf{b}} \mathbf{h}' = \langle x';y', g' \fatsemi_{\mathsf{b}} h'\rangle$ and clearly $x';y' \sqsubseteq_{\mathsf{f}} x;\hat{y}$, it remains to prove $g \fatsemi_{\mathsf{b}} \hat{h} \sqsubseteq_{\mathsf{fb}} g' \fatsemi_{\mathsf{b}} h'$. From $\mathbf{g} \sqsubseteq_{\mathsf{fb}} \mathbf{g}'$, we have $g \sqsubseteq_{\mathsf{fb}} g'$. Since $h' \trianglelefteq \hat{h}$, we have $\hat{h} \sqsubseteq_{\mathsf{fb}} h'$ by Lemma 21. Since $\trianglelefteq$ is reflexive, we have $\hat{h} \in G^R$, and so $g \fatsemi_{\mathsf{b}} \hat{h} \sqsubseteq_{\mathsf{fb}} g' \fatsemi_{\mathsf{b}} h'$ by Lemma 33. Therefore, $\mathbf{g} \fatsemi_{\mathsf{b}} \hat{\mathbf{h}} \sqsubseteq_{\mathsf{fb}} \mathbf{g}' \fatsemi_{\mathsf{b}} \mathbf{h}'$ holds by the definition of $\sqsubseteq_{\mathsf{fb}}$.

For Point 2), let $\mathbf{h} \in S^R$ be any representative s.t. $\mathbf{h} \sqsubseteq_{\mathsf{fb}} \hat{\mathbf{h}}$. Then, $\mathbf{h} = \langle y,h\rangle$ with $\hat{y} \sqsubseteq_{\mathsf{f}} y$ and $h \sqsubseteq_{\mathsf{fb}} \hat{h}$, where $h \in G^R$. Thus, y has the form $y = \langle q,b,r\rangle$, with $\hat{b} \leq b$ and $r' \preceq_{\mathsf{f}} \hat{r} \preceq_{\mathsf{f}} r$. Clearly, $\mathbf{g},\mathbf{h}$ are still composable, and $\mathbf{g} \fatsemi_{\mathsf{b}} \mathbf{h} = \langle x;y, g \fatsemi_{\mathsf{b}} h\rangle$. Notice that $x;\hat{y} \sqsubseteq_{\mathsf{f}} x;y$. Since $g \fatsemi_{\mathsf{b}} h \sqsubseteq_{\mathsf{fb}} g \fatsemi_{\mathsf{b}} \hat{h}$ by Lemma 33, we have $\mathbf{g} \fatsemi_{\mathsf{b}} \mathbf{h} \sqsubseteq_{\mathsf{fb}} \mathbf{g} \fatsemi_{\mathsf{b}} \hat{\mathbf{h}} \sqsubseteq_{\mathsf{fb}} \mathbf{g}' \fatsemi_{\mathsf{b}} \mathbf{h}'$ by the definition of $\sqsubseteq_{\mathsf{fb}}$. By transitivity, we finally obtain $\mathbf{g} \fatsemi_{\mathsf{b}} \mathbf{h} \sqsubseteq_{\mathsf{fb}} \mathbf{g}' \fatsemi_{\mathsf{b}} \mathbf{h}'$. □

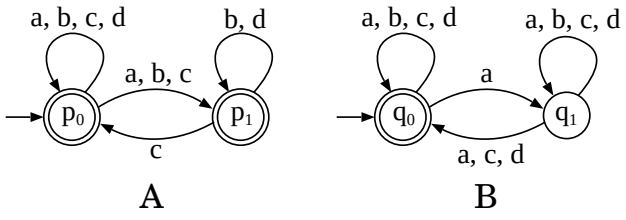

Fig. 1. A running example

D A Running Example

Below, we illustrate on an example the notions of minimization and subsumption discussed in the paper. We consider the BA from Figure 1. The following forward simulation relations hold in the given automata (we do not list the relations corresponding to the identity): $p_0 \preceq_{\mathsf{f}} q_0$, $p_1 \preceq_{\mathsf{f}} p_0$, $p_1 \preceq_{\mathsf{f}} q_0$, $q_0 \preceq_{\mathsf{f}} p_0$, $q_1 \preceq_{\mathsf{f}} p_0$, and $q_1 \preceq_{\mathsf{f}} q_0$. The backward simulation relations are then the following (again ignoring the identity): $p_0 \preceq_{\mathsf{b}} q_0$, $p_1 \preceq_{\mathsf{b}} p_0$, $p_1 \preceq_{\mathsf{b}} q_0$, $q_0 \preceq_{\mathsf{b}} p_0$, $q_1 \preceq_{\mathsf{b}} p_0$, $q_1 \preceq_{\mathsf{b}} p_1$, and $q_1 \preceq_{\mathsf{b}} q_0$.

We first consider using only *forward simulation* for minimization (denoted Min_f) and subsumption as proposed in [1]. The following one-letter supergraphs are generated:

- Using letter a: The corresponding non-minimized one-letter graph over B is $g^a = \{(q_1,0,q_1),(q_1,1,q_0),(q_0,1,q_1),(q_0,1,q_0)\}$. The first edge is $\sqsubseteq_{\mathsf{f}}$-subsumed by the second, and the third by the fourth. Hence, $Min_f(g^a) = \{(q_1,1,q_0),(q_0,1,q_0)\}$. Based on $g^a_m = Min_f(g^a)$, two one-letter supergraphs are obtained: $\mathbf{g}^a_1 = ((p_0,1,p_1), g^a_m)$ and $\mathbf{g}^a_2 = ((p_0,1,p_0), g^a_m)$. However, since $\mathbf{g}^a_2 \sqsubseteq_{\mathsf{f}} \mathbf{g}^a_1$, we may discard $\mathbf{g}^a_1$.
- Using letter b: The corresponding non-minimized one-letter graph over B is $g^b = \{(q_1,0,q_1),\ (q_0,1,q_0)\}$. Minimization will not help here, and we can notice that $g^b \sqsubseteq_{\mathsf{f}} g^a_m$. Based on g^b, three one-letter supergraphs are obtained: $\mathbf{g}^b_1 = ((p_1,1,p_1), g^b)$, $\mathbf{g}^b_2 = ((p_0,1,p_1), g^b)$, and $\mathbf{g}^b_3 = ((p_0,1,p_0), g^b)$. Since $\mathbf{g}^b_3 \sqsubseteq_{\mathsf{f}} \mathbf{g}^b_2$, we may discard $\mathbf{g}^b_2$. Moreover, since $\mathbf{g}^b_3 \sqsubseteq_{\mathsf{f}} \mathbf{g}^a_2$, we may discard $\mathbf{g}^a_2$ too.
- Using letter c: The corresponding non-minimized one-letter graph over B is $g^c = \{(q_1,0,q_1),(q_1,1,q_0),(q_0,1,q_0)\}$. Here, the first edge is $\sqsubseteq_{\mathsf{f}}$-subsumed by the second, and hence, $Min_f(g^c) = g^c_m = g^a_m$. Based on g^c_m, three one-letter supergraphs are obtained: $\mathbf{g}^c_1 = ((p_1,1,p_0), g^c_m)$, $\mathbf{g}^c_2 = ((p_0,1,p_1), g^c)$, and $\mathbf{g}^c_3 = ((p_0,1,p_0), g^c)$. However, since $\mathbf{g}^b_3 \sqsubseteq_{\mathsf{f}} \mathbf{g}^c_3 \sqsubseteq_{\mathsf{f}} \mathbf{g}^c_2$, we retain $\mathbf{g}^c_1$ only.
- Using letter d: The corresponding non-minimized one-letter graph over B is $g^d = \{(q_1,0,q_1),(q_1,1,q_0),(q_0,1,q_0)\}$. Here, the first edge is $\sqsubseteq_{\mathsf{f}}$-subsumed by the second, and hence, $Min_f(g^d) = g^d_m = g^a_m$. Based on g^d_m, two one-letter supergraphs are obtained: $\mathbf{g}^d_1 = ((p_1,1,p_1), g^d_m)$ and $\mathbf{g}^d_2 = ((p_0,1,p_0), g^d_m)$. However, since $\mathbf{g}^b_1 \sqsubseteq_{\mathsf{f}} \mathbf{g}^d_1$ and $\mathbf{g}^b_3 \sqsubseteq_{\mathsf{f}} \mathbf{g}^d_2$, we may discard both $\mathbf{g}^d_1$ and $\mathbf{g}^d_2$.

Hence, the main loop of the inclusion checking is started with the set of supergraphs $\{\mathbf{g}_1^b, \mathbf{g}_3^b, \mathbf{g}_1^c\}$. These supergraphs will be used for generating new supergraphs by right extension. In the main loop, assume we start by processing $\mathbf{g}_1^b$. It is clearly the case that $RDGT(\mathbf{g}_1^b, \mathbf{g}_1^b)$ passes since p_1 is not an initial state. It is then possible to extend $\mathbf{g}_1^b$ by $\mathbf{g}_1^b$ and $\mathbf{g}_1^c$. However, since $\mathbf{g}_1^b;\mathbf{g}_1^b = \mathbf{g}_1^b$ and $\mathbf{g}_1^b;\mathbf{g}_1^c = \mathbf{g}_1^c$, no new supergraph is generated. In a similar manner, the main loop will process the two other supergraphs. All *RDGT* tests (testing the supergraphs being processed against themselves as well as against the previously processed supergraphs) will pass, and no new supergraph will be generated. Hence, the algorithm will terminate with the result that the conclusion holds.

Next, we illustrate the effect of using both forward and backward simulation for minimization and subsumption as proposed in this paper. The following one-letter supergraphs are generated this time:

- Using letter a: The corresponding non-minimized one-letter graph over B is $g^a = \{(q_1,0,q_1),(q_1,1,q_0),(q_0,1,q_1),(q_0,1,q_0)\}$ as before. However, now, all the first three edges are $\sqsubseteq_{\mathsf{fb}}$-subsumed by the last one, and hence, $Min(g^a) = \{(q_0,1,q_0)\}$. Based on $h^a = Min(g^a)$, two one-letter supergraphs are obtained: $\mathbf{h}_1^a = ((p_0,1,p_1), h^a)$ and $\mathbf{h}_2^a = ((p_0,1,p_0),h^a)$. However, since $\mathbf{h}_2^a \sqsubseteq_{\mathsf{fb}} \mathbf{h}_1^a$, we may discard $\mathbf{h}_1^a$.
- Using letter b: The corresponding non-minimized one-letter graph over B is $g^b = \{(q_1,0,q_1),\ (q_0,1,q_0)\}$. Since the first edge is $\sqsubseteq_{\mathsf{fb}}$-subsumed by the second one, $h^b = Min(g^b) = \{(q_0,1,q_0)\} = h^a$. Based on h^b, three one-letter supergraphs are obtained: $\mathbf{h}_1^b = ((p_1,1,p_1),h^b)$, $\mathbf{h}_2^b = ((p_0,1,p_1),h^b)$, and $\mathbf{h}_3^b = ((p_0,1,p_0),h^b)$. Since $\mathbf{h}_3^b \sqsubseteq_{\mathsf{fb}} \mathbf{h}_2^b$, we may discard $\mathbf{h}_2^b$. Moreover, we have that $\mathbf{h}_3^b = \mathbf{h}_2^a$.
- Using letter c: The corresponding non-minimized one-letter graph over B is $g^c = \{(q_1,0,q_1),(q_1,1,q_0),(q_0,1,q_0)\}$. Here, the first edge is $\sqsubseteq_{\mathsf{fb}}$-subsumed by the second, the second by the third, and hence, $Min(g^c) = h^c = h^a$. Based on h^c, three one-letter supergraphs are obtained: $\mathbf{h}_1^c = ((p_1,1,p_0),h^c)$, $\mathbf{h}_2^c = ((p_0,1,p_1),h^c)$, and $\mathbf{h}_3^c = ((p_0,1,p_0),h^c)$. However, since $\mathbf{h}_3^c \sqsubseteq_{\mathsf{fb}} \mathbf{h}_2^c$, $\mathbf{h}_2^c$ can be discarded. Moreover, $\mathbf{h}_3^c = \mathbf{h}_2^a$. Further, $\mathbf{h}_1^c \sqsubseteq_{\mathsf{fb}} \mathbf{h}_1^b$, and so $\mathbf{h}_1^b$ can be discarded too.
- Using letter d: The corresponding non-minimized one-letter graph over B is $g^d = \{(q_1,0,q_1),(q_1,1,q_0),(q_0,1,q_0)\}$. Here, the first edge is $\sqsubseteq_{\mathsf{fb}}$-subsumed by the second, the second by the third, and hence, $Min(g^d) = h^d = h^a$. Based on h^d, two one-letter supergraphs are obtained: $\mathbf{h}_1^d = ((p_1,1,p_1),\ h^d)$ and $\mathbf{h}_2^d = ((p_0,1,p_0),h^d)$. However, since $\mathbf{h}_1^c \sqsubseteq_{\mathsf{fb}} \mathbf{h}_1^d$, $\mathbf{h}_1^d$ can be discarded. Moreover, $\mathbf{h}_2^d = \mathbf{h}_2^a$.

Hence, the main loop of the inclusion checking is started with the set of supergraphs $\{\mathbf{h}_2^a, \mathbf{h}_1^c\}$. The main loop will process the two supergraphs, all tests on them will pass, and no new supergraph will be generated. Hence, the algorithm terminates with the conclusion that the inclusion holds. Note that, even in this simple example, one less supergraph is generated compared to the approach above, and, moreover, there is only a single minimized graph component used that is smaller than the graph components used before.

E Proofs for Sections 5 and 6

Lemma 11. $\forall \mathbf{g},\mathbf{h} \in S^R.\ \mathsf{C}(\mathbf{g}) \Rightarrow \mathit{RDGT}_{\mathsf{b}}(\mathbf{g},\mathbf{h})$.

Proof. To show the correctness of C, we assume the contrary and derive a contradiction. Let $\mathbf{g} = \langle\langle p_{\mathbf{g}}, l_{\mathbf{g}}, q_{\mathbf{g}}\rangle, g\rangle$ and $\mathbf{h} = \langle\langle p_{\mathbf{h}}, l_{\mathbf{h}}, q_{\mathbf{h}}\rangle, h\rangle$, and assume $\mathsf{C}(\mathbf{g})$ and $\neg \mathit{RDGT}_{\mathsf{b}}(\mathbf{g},\mathbf{h})$. From $\neg \mathit{RDGT}_{\mathsf{b}}(\mathbf{g},\mathbf{h})$, we obtain $p_{\mathbf{g}} \in I_{\mathcal{A}}$, and thus, by $\mathsf{C}(\mathbf{g})$, there exists some edge $\langle x,a,y\rangle \in g$ s.t. $x \in I_{\mathcal{B}}$, $a \geq 0$ and $q_{\mathbf{g}} \preceq_{\mathsf{f}}^{\mathcal{AB}} y$. Furthermore, $\neg \mathit{RDGT}_{\mathsf{b}}(\mathbf{g},\mathbf{h})$ implies $p_{\mathbf{h}} \preceq_{\mathsf{f}}^{\mathcal{A}} q_{\mathbf{g}}$, and thus $p_{\mathbf{h}} \preceq_{\mathsf{f}}^{\mathcal{AB}} y$. In particular, $\mathcal{L}(p_{\mathbf{h}}) \subseteq \mathcal{L}(y)$. (This inclusion is all we need here, and thus $\preceq_{\mathsf{f}}^{\mathcal{AB}}, \preceq_{\mathsf{f}}^{\mathcal{A}}$ could be replaced by any relation implying ω-language inclusion.) Since $\mathbf{g},\mathbf{h} \in S^R$, there exist supergraphs $\mathbf{g}',\mathbf{h}' \in S^f$ s.t. $\mathbf{g} \trianglelefteq \mathbf{g}'$ and $\mathbf{h} \trianglelefteq \mathbf{h}'$. We have $\mathbf{g}' = \langle\langle p_{\mathbf{g}}, l_{\mathbf{g}}, q_{\mathbf{g}}\rangle, g'\rangle \in S^f$ s.t. $g \trianglelefteq g'$ and $g' \in G^f$, and $\mathbf{h}' = \langle\langle p_{\mathbf{h}}, l_{\mathbf{h}}, q_{\mathbf{h}}\rangle, h'\rangle \in S^f$ s.t. $h \trianglelefteq h'$ and $h' \in G^f$. Now $\neg \mathit{RDGT}_{\mathsf{b}}(\mathbf{g},\mathbf{h})$ implies $l_{\mathbf{h}} = 1$ and $q_{\mathbf{h}} \succeq_{\mathsf{f}}^{\mathcal{A}} p_{\mathbf{h}}$ (even delayed simulation would suffice here). Thus, since $\mathcal{L}(\mathbf{h}') \neq \emptyset$, there exists some $w \in \Sigma^+ \cap \mathcal{L}(\mathbf{h}')$ s.t. $p_{\mathbf{h}} \overset{w}{\rightsquigarrow} q_{\mathbf{h}}$, and at least one accepting state from $F_{\mathcal{A}}$ is visited on the path. By Lemma 34, we have $w^{\omega} \in \mathcal{L}(p_{\mathbf{h}})$. Since $\mathcal{L}(p_{\mathbf{h}}) \subseteq \mathcal{L}(y)$, we obtain $w^{\omega} \in \mathcal{L}(y)$. Thus, there exists an infinite sequence of states y_i $(i = 0,1,2,\dots)$ s.t. $y_0 = y$ and $y_i \overset{w}{\rightsquigarrow} y_{i+1}$ for every i, and an accepting state from $F_{\mathcal{B}}$ is visited in infinitely many (but not necessarily all) of these sequences $y_i \overset{w}{\rightsquigarrow} y_{i+1}$. Since $h' \in G^f$, there exist arcs $\langle y_i, l_i, y_{i+1}\rangle \in h'$ with $l_i \geq 0$ for every i, and $l_i = 1$ for infinitely many i. Furthermore, $\langle x,a,y\rangle \in g$ with $a \geq 0$, $x \in I_{\mathcal{B}}$ and $g \trianglelefteq g'$, i.e., there exists a witness $\bar{g}$ s.t. $g \leqslant \bar{g} \simeq_{\mathsf{b}} g'$. By Definition 1, there exists an arc $\langle x,a',y\rangle \in \bar{g}$ with $a' \geq a \geq 0$. By the definition of $\simeq_{\mathsf{b}}$, there is an arc $\langle x',a'',y\rangle \in g'$ with $a'' \geq a' \geq 0$ and $x' \succeq_{\mathsf{b}}^{\mathcal{B}} x \in I_{\mathcal{B}}$, and thus $x' \in I_{\mathcal{B}}$. So, we have proved $\mathit{LFT}(g',h')$. Finally, Lemma 31 implies $\mathit{LFT}_{\mathsf{b}}(g,h)$, which implies $\mathit{RDGT}_{\mathsf{b}}(\mathbf{g},\mathbf{h})$, contradicting our assumption. □

Lemma 12. Let $\mathbf{g},\mathbf{h} \in S$ s.t. $\mathbf{g} \sqsubseteq_{\mathsf{fb}} \mathbf{h}$. Then $\mathsf{C}(\mathbf{g}) \Rightarrow \mathsf{C}(\mathbf{h})$.

Proof. Let $\mathbf{g} = \langle\langle p_{\mathbf{g}}, l_{\mathbf{g}}, q_{\mathbf{g}}\rangle, g\rangle$ and $\mathbf{h} = \langle\langle p_{\mathbf{h}}, l_{\mathbf{h}}, q_{\mathbf{h}}\rangle, h\rangle$. Since $\mathbf{g} \sqsubseteq_{\mathsf{fb}} \mathbf{h}$, we have $p_{\mathbf{g}} = p_{\mathbf{h}}$.

If $\mathsf{C}(\mathbf{g})$, then there are two cases. If $p_{\mathbf{g}} \notin I_{\mathcal{A}}$, then $p_{\mathbf{h}} \notin I_{\mathcal{A}}$, and thus $\mathsf{C}(\mathbf{h})$. Otherwise, $p_{\mathbf{g}} \in I_{\mathcal{A}}$ and there is some arc $\langle x,a,y\rangle \in g$ s.t. $x \in I_{\mathcal{B}}$ and $q_{\mathbf{g}} \preceq_{\mathsf{f}}^{\mathcal{AB}} y$. Since $\mathbf{g} \sqsubseteq_{\mathsf{fb}} \mathbf{h}$, we have $g \sqsubseteq_{\mathsf{fb}} h$, and thus there exists some $\langle x',a',y'\rangle \in h$ s.t. $x \preceq_{\mathsf{b}}^{\mathcal{B}} x' \in I_{\mathcal{B}}$, $a' \geq a \geq 0$ and $q_{\mathbf{h}} \preceq_{\mathsf{f}}^{\mathcal{A}} q_{\mathbf{g}} \preceq_{\mathsf{f}}^{\mathcal{AB}} y \preceq_{\mathsf{f}}^{\mathcal{B}} y'$, which implies $\mathsf{C}(\mathbf{h})$. □

Lemma 13. Let $\mathbf{g} \in S^R$ and $\mathbf{h} \in \mathit{Min}(S^1)$ be composable. Then $\mathsf{C}(\mathbf{g}) \Rightarrow \mathsf{C}(\mathbf{g} \fatsemi_{\mathsf{b}} \mathbf{h})$.

Proof. Let $\mathbf{g} = \langle\langle p_{\mathbf{g}}, l_{\mathbf{g}}, q_{\mathbf{g}}\rangle, g\rangle \in S^R$ and $\mathbf{h} = \langle\langle p_{\mathbf{h}}, l_{\mathbf{h}}, q_{\mathbf{h}}\rangle, h\rangle \in \mathit{Min}(S^1)$ be two composable supergraphs. In particular this implies that $q_{\mathbf{g}} = p_{\mathbf{h}}$. Let $\mathbf{f} := \mathbf{g} \fatsemi_{\mathsf{b}} \mathbf{h}$, with $\mathbf{f} = \langle\langle p_{\mathbf{f}}, l_{\mathbf{f}}, q_{\mathbf{f}}\rangle, f\rangle$. Then, by Definition 2, we have $p_{\mathbf{f}} = p_{\mathbf{g}}$, $l_{\mathbf{f}} = \max\{l_{\mathbf{g}}, l_{\mathbf{h}}\}$, $q_{\mathbf{f}} = q_{\mathbf{h}}$ and $f = g \fatsemi_{\mathsf{b}} h$. If $\mathsf{C}(\mathbf{g})$, then there are two cases.

- If $p_{\mathbf{g}} \notin I_{\mathcal{A}}$, then $p_{\mathbf{f}} \notin I_{\mathcal{A}}$, and thus $\mathsf{C}(\mathbf{f})$ because $p_{\mathbf{f}} = p_{\mathbf{g}}$.
- Otherwise, we have $p_{\mathbf{g}} \in I_{\mathcal{A}}$ and there is some $\langle x,a,y\rangle \in g$ s.t. $x \in I_{\mathcal{B}}$ and $q_{\mathbf{g}} \preceq_{\mathsf{f}}^{\mathcal{AB}} y$. Since $\mathbf{h} \in \mathit{Min}(S^1)$ is single-letter supergraph, there exists some letter $e \in \Sigma$ s.t. $q_{\mathbf{h}} \in \delta_{\mathcal{A}}(p_{\mathbf{h}}, e)$. Since $p_{\mathbf{h}} = q_{\mathbf{g}} \preceq_{\mathsf{f}}^{\mathcal{AB}} y$, there exists some state z s.t. $z \in \delta_{\mathcal{B}}(y,e)$ and $q_{\mathbf{h}} \preceq_{\mathsf{f}}^{\mathcal{AB}} z$. Since $\mathbf{h} \in \mathit{Min}(S^1)$, there exists some supergraph $\mathbf{h}' = \langle\langle p_{\mathbf{h}}, l_{\mathbf{h}}, q_{\mathbf{h}}\rangle, h'\rangle \in S^1$ with $h \leqslant h' \in G^f$. It follows that $\langle y,l,z\rangle \in h'$ for some $l \geq 0$. By Definition 1, there

is some $\langle y', l', z' \rangle \in h$ s.t. $y' \succeq_{\mathsf{b}}^{\mathcal{B}} y$, $l' \geq l$ and $z' \succeq_{\mathsf{f}}^{\mathcal{B}} z$. Therefore, $\langle x, \max\{a, l'\}, z' \rangle \in g \fatsemi_{\mathsf{b}} h = f$, where $x \in I_{\mathcal{B}}$ and $q_{\mathbf{f}} = q_{\mathbf{h}} \preceq_{\mathsf{f}}^{\mathcal{AB}} z \preceq_{\mathsf{f}}^{\mathcal{B}} z'$. This implies $\mathsf{C}(\mathbf{f})$, as needed. □

Lemma 14. Let $\mathbf{g}, \mathbf{h} \in S^R$. If $\neg RDGT_{\mathsf{b}}(\mathbf{g}, \mathbf{h})$, then there exists a $\sqsubseteq_{\mathsf{fb}}$-minimal supergraph $\mathbf{f}$ in $Min(S^1_{\mathcal{A},\mathcal{B}})$ and $\mathbf{e} \in S^R$ s.t. $\neg RDGT_{\mathsf{b}}(\mathbf{g} \fatsemi_{\mathsf{b}} \mathbf{f}, \mathbf{e})$.

Proof. Let $\mathbf{g} = \langle \langle p_{\mathbf{g}}, l_{\mathbf{g}}, q_{\mathbf{g}} \rangle, g \rangle$ and $\mathbf{h} = \langle \langle p_{\mathbf{h}}, l_{\mathbf{h}}, q_{\mathbf{h}} \rangle, h \rangle$ in S^R. Then, there exist supergraphs $\mathbf{g}', \mathbf{h}' \in S^f$ s.t. $\mathbf{g} \trianglelefteq \mathbf{g}'$ and $\mathbf{h} \trianglelefteq \mathbf{h}'$. So, we get $\mathbf{g}' = \langle \langle p_{\mathbf{g}}, l_{\mathbf{g}}, q_{\mathbf{g}} \rangle, g' \rangle$ and $\mathbf{h}' = \langle \langle p_{\mathbf{h}}, l_{\mathbf{h}}, q_{\mathbf{h}} \rangle, h' \rangle$, with $g \trianglelefteq g'$ and $h \trianglelefteq h'$.

From $\neg RDGT_{\mathsf{b}}(\mathbf{g}, \mathbf{h})$, we obtain $p_{\mathbf{g}} \in I_{\mathcal{A}}$, $q_{\mathbf{g}} \succeq_{\mathsf{f}}^{\mathcal{A}} p_{\mathbf{h}}$, $q_{\mathbf{h}} \succeq_{\mathsf{f}}^{\mathcal{A}} p_{\mathbf{h}}$, $l_{\mathbf{h}} = 1$ and $\neg LFT_{\mathsf{b}}(g, h)$. By Lemma 31, we have $\neg LFT(g', h')$.

By Lemma 3, $\exists \mathbf{g}_1, \ldots, \mathbf{g}_n \in S^1_{\mathcal{A},\mathcal{B}} :\ \mathbf{g}' = \mathbf{g}_1; \ldots; \mathbf{g}_n$ and $\exists \mathbf{h}_1, \ldots, \mathbf{h}_m \in S^1_{\mathcal{A},\mathcal{B}} :\ \mathbf{h}' = \mathbf{h}_1; \ldots; \mathbf{h}_m$. Let $a_i \in \Sigma$ be the letter associated with the single-letter supergraph $\mathbf{g}_i$, let b_j be the letter associated with the single-letter supergraph $\mathbf{h}_j$, and let $w_1 = a_1 \ldots a_n$ and $w_2 = b_1 \ldots b_m$. Notice that $p_{\mathbf{g}} \overset{w_1}{\rightsquigarrow} q_{\mathbf{g}}$ and $p_{\mathbf{h}} \overset{w_2}{\rightsquigarrow} q_{\mathbf{h}}$. By Lemma 34, we get $w_2^{\omega} \in \mathcal{L}(p_{\mathbf{h}})$. Since $q_{\mathbf{g}} \succeq_{\mathsf{f}}^{\mathcal{A}} p_{\mathbf{h}}$, we have $\mathcal{L}(p_{\mathbf{h}}) \subseteq \mathcal{L}(q_{\mathbf{g}})$, and thus we obtain $w_2^{\omega} \in \mathcal{L}(q_{\mathbf{g}})$ and $w_1 w_2^{\omega} \in \mathcal{L}(p_{\mathbf{g}}) \subseteq \mathcal{L}(\mathcal{A})$. However, $w_1 w_2^{\omega} \notin \mathcal{L}(\mathcal{B})$ because $\neg LFT(g', h')$.

Let $\mathbf{h}_j = \langle \langle p^j_{\mathbf{h}}, l^j_{\mathbf{h}}, q^j_{\mathbf{h}} \rangle, h_{b_j} \rangle$ where h_{b_j} is the single-letter graph for letter b_j. In particular, $q^j_{\mathbf{h}} = p^{j+1}_{\mathbf{h}}$, since the supergraphs are composable. Since $q^m_{\mathbf{h}} = q_{\mathbf{h}} \succeq_{\mathsf{f}}^{\mathcal{A}} p_{\mathbf{h}} = p^1_{\mathbf{h}}$ and $p^1_{\mathbf{h}} \overset{b_1}{\rightsquigarrow} q^1_{\mathbf{h}} = p^2_{\mathbf{h}}$, there exists a state q_e s.t. $q_{\mathbf{h}} = q^m_{\mathbf{h}} \overset{b_1}{\rightsquigarrow} q_e \succeq_{\mathsf{f}}^{\mathcal{A}} p^2_{\mathbf{h}}$. Thus, there is a single-letter supergraph $\mathbf{h}'_1 = \langle \langle q_{\mathbf{h}}, l, q_{\mathbf{e}} \rangle, h_{b_1} \rangle \in S^1_{\mathcal{A},\mathcal{B}}$ s.t. $l \geq l^1_{\mathbf{h}}$. Let $\mathbf{e} = \mathbf{h}_2; \ldots; \mathbf{h}_m; \mathbf{h}'_1 = \langle \langle p_{\mathbf{e}}, l_{\mathbf{e}}, q_{\mathbf{e}} \rangle, e \rangle$ with $e = h_{b_2}; \ldots h_{b_m}; h_{b_1} \in G^f$, $p_{\mathbf{e}} = p^2_{\mathbf{h}}$, $l_{\mathbf{e}} \geq l_{\mathbf{h}} = 1$ and $q_{\mathbf{e}} \succeq_{\mathsf{f}}^{\mathcal{A}} p_{\mathbf{e}}$.

Similarly, since $q_{\mathbf{g}} \succeq_{\mathsf{f}}^{\mathcal{A}} p_{\mathbf{h}} = p^1_{\mathbf{h}} \overset{b_1}{\rightsquigarrow} q^1_{\mathbf{h}} = p^2_{\mathbf{h}} = p_{\mathbf{e}}$, there exists a state $q'_{\mathbf{f}}$ s.t. $q_{\mathbf{g}} \overset{b_1}{\rightsquigarrow} q'_{\mathbf{f}} \succeq_{\mathsf{f}}^{\mathcal{A}} p_{\mathbf{e}}$. Thus, there is a single-letter supergraph $\mathbf{f}' = \langle \langle p'_{\mathbf{f}}, l'_{\mathbf{f}}, q'_{\mathbf{f}} \rangle, f' \rangle \in S^1_{\mathcal{A},\mathcal{B}}$, where $f' = h_{b_1}$, $p'_{\mathbf{f}} = q_{\mathbf{g}}$ and $l'_{\mathbf{f}} \geq 0$.

We get $\mathbf{g}'; \mathbf{f}' = \langle \langle p_{\mathbf{g}}, l', q'_{\mathbf{f}} \rangle, g'; f' \rangle$, and the pair $\langle \mathbf{g}'; \mathbf{f}', \mathbf{e} \rangle$ satisfies the weak properness condition because $p_{\mathbf{g}} \in I_{\mathcal{A}}$, $q'_{\mathbf{f}} \succeq_{\mathsf{f}}^{\mathcal{A}} p_{\mathbf{e}}$, $l_{\mathbf{e}} = 1$ and $q_{\mathbf{e}} \succeq_{\mathsf{f}}^{\mathcal{A}} p_{\mathbf{e}}$. Furthermore, we know that $w_1 w_2^{\omega} \notin \mathcal{L}(\mathcal{B})$. Since $w_1 b_1 (b_2 \ldots b_n b_1)^{\omega} = w_1 w_2^{\omega}$, we obtain $w_1 b_1 (b_2 \ldots b_n b_1)^{\omega} \notin \mathcal{L}(\mathcal{B})$, and thus $\neg LFT(g'; f', e)$. By Lemma 31, we get $\neg LFT_{\mathsf{b}}(g'; f', e)$. Therefore, $\neg RDGT_{\mathsf{b}}(\mathbf{g}'; \mathbf{f}', \mathbf{e})$. Since $g', f' \in G^f$ and $\trianglelefteq$ is reflexive and implies $\simeq_{\mathsf{fb}}$, Lemma 26 yields $g'; f' \simeq_{\mathsf{fb}} g' \fatsemi_{\mathsf{b}} f'$. Thus, we get $\mathbf{g}'; \mathbf{f}' \simeq_{\mathsf{fb}} \mathbf{g}' \fatsemi_{\mathsf{b}} \mathbf{f}'$.

Now, choose as $\mathbf{f} = \langle \langle p_{\mathbf{f}}, l_{\mathbf{f}}, q_{\mathbf{f}} \rangle, f \rangle$ a supergraph in $Min(S^1_{\mathcal{A},\mathcal{B}})$ which is minimal w.r.t. $\sqsubseteq_{\mathsf{fb}}$ and $\mathbf{f} \sqsubseteq_{\mathsf{fb}} \mathbf{f}'$. (This exists because $\mathbf{f}' \in S^1_{\mathcal{A},\mathcal{B}}$ and Min preserves $\simeq_{\mathsf{fb}}$.) By the definition of $\sqsubseteq_{\mathsf{fb}}$, we have $p_{\mathbf{f}} = p'_{\mathbf{f}} = q_{\mathbf{g}}$, and thus $\mathbf{g}$ and $\mathbf{f}$ are composable. Since also $\mathbf{g} \sqsubseteq_{\mathsf{fb}} \mathbf{g}'$, we get $\mathbf{g} \fatsemi_{\mathsf{b}} \mathbf{f} \sqsubseteq_{\mathsf{fb}} \mathbf{g}' \fatsemi_{\mathsf{b}} \mathbf{f}' \simeq_{\mathsf{fb}} \mathbf{g}'; \mathbf{f}'$ by Lemma 10.

Finally, by Lemma 9, we obtain $\neg RDGT_{\mathsf{b}}(\mathbf{g} \fatsemi_{\mathsf{b}} \mathbf{f}, \mathbf{e})$, which concludes the proof. □

The following lemma is a slightly modified version of Lemma 14. Like Lemma 14, it justifies the optimization of removing L-labels from supergraphs in the set *Processed*. However, it works even under the weaker assumptions on the RDGT, as mentioned in the footnote on Def. 4.

Lemma 37. *Let* $\mathbf{g}, \mathbf{h} \in S^R$. *If* $\neg RDGT_{\mathsf{b}}(\mathbf{g}, \mathbf{h})$, *then there exist supergraphs* $\mathbf{f}, \mathbf{e} \in S^f$ *s.t.* $\neg RDGT_{\mathsf{b}}(\mathbf{g} \fatsemi_{\mathsf{b}} \mathbf{f}, \mathbf{e})$.

Proof. Let $\mathbf{g} = \langle\langle p_\mathbf{g}, l_\mathbf{g}, q_\mathbf{g}\rangle, g\rangle$ and $\mathbf{h} = \langle\langle p_\mathbf{h}, l_\mathbf{h}, q_\mathbf{h}\rangle, h\rangle$ in S^R. Then, there exist supergraphs $\mathbf{g}', \mathbf{h}' \in S^f$ s.t. $\mathbf{g} \trianglelefteq \mathbf{g}'$ and $\mathbf{h} \trianglelefteq \mathbf{h}'$. So, we get $\mathbf{g}' = \langle\langle p_\mathbf{g}, l_\mathbf{g}, q_\mathbf{g}\rangle, g'\rangle$ and $\mathbf{h}' = \langle\langle p_\mathbf{h}, l_\mathbf{h}, q_\mathbf{h}\rangle, h'\rangle$, with $g \trianglelefteq g'$ and $h \trianglelefteq h'$.

Using only the weak version of the RDGT (i.e., with inclusion and delayed simulation instead of direct simulation, as explained in the footnote on Def. 4) it follows from $\neg RDGT_\mathsf{b}(\mathbf{g}, \mathbf{h})$ that $p_\mathbf{g} \in I_\mathcal{A}$, $\mathcal{L}(q_\mathbf{g}) \supseteq \mathcal{L}(p_\mathbf{h})$, $q_\mathbf{h} \succeq^{de}_\mathcal{A} p_\mathbf{h}$, $l_\mathbf{h} = 1$ and $\neg LFT_\mathsf{b}(g,h)$. By Lemma 31, we have $\neg LFT(g', h')$.

Since $\mathbf{g}', \mathbf{h}' \in S^f$, there exist words $w_1, w_2 \in \Sigma^+$ s.t. $p_\mathbf{g} \overset{w_1}{\rightsquigarrow} q_\mathbf{g}$ and $p_\mathbf{h} \overset{w_2}{\rightsquigarrow} q_\mathbf{h}$. By Lemma 34, we get $w_2^\omega \in \mathcal{L}(p_\mathbf{h})$. Since $\mathcal{L}(p_\mathbf{h}) \subseteq \mathcal{L}(q_\mathbf{g})$, we obtain $w_2^\omega \in \mathcal{L}(q_\mathbf{g})$ and $w_1 w_2^\omega \in \mathcal{L}(p_\mathbf{g}) \subseteq \mathcal{L}(\mathcal{A})$. However, $w_1 w_2^\omega \notin \mathcal{L}(\mathcal{B})$ because $\neg LFT(g', h')$.

Since $w_2^\omega \in \mathcal{L}(q_\mathbf{g})$, there exists an infinite sequence of states $x_1, x_2, \ldots$ in $\mathcal{A}$ s.t. $q_\mathbf{g} \overset{w_2}{\rightsquigarrow} x_1$, and $x_i \overset{w_2}{\rightsquigarrow} x_{i+1}$ for every $i \geq 0$, and for infinitely many such i an accepting state from $F_\mathcal{A}$ is visited between x_i and x_{i+1}. Since the number of states of $\mathcal{A}$ is finite, there exists a $j \geq 1$ and a $k \geq 1$ s.t. $x_j \overset{w_2^k}{\rightsquigarrow} x_j$, and an accepting state from $F_\mathcal{A}$ is visited on the way. Let $x = x_j$. Then, there exists a supergraph $\mathbf{e} = \langle\langle x, l_\mathbf{e}, x\rangle, e\rangle \in S^f$ with $l_\mathbf{e} = 1$ and $e = h; \ldots (k \text{ times}) \ldots; h$. Moreover, there exists a supergraph $\mathbf{f} = \langle\langle q_\mathbf{g}, l_\mathbf{f}, x\rangle, f\rangle \in S^f$ with $l_\mathbf{f} \geq 0$ and $f = h; \ldots (j \text{ times}) \ldots; h$.

We get $\mathbf{g}'; \mathbf{f} = \langle\langle p_\mathbf{g}, l', x\rangle, g'; f\rangle$ with $l' \geq 0$, and the pair $\langle \mathbf{g}'; \mathbf{f}, \mathbf{e}\rangle$ satisfies the modified weak properness condition because $p_\mathbf{g} \in I_\mathcal{A}$, $\mathcal{L}(x) \subseteq \mathcal{L}(x)$, $l_\mathbf{e} = 1$ and $x \succeq^{de}_\mathcal{A} x$.

Furthermore, we know that $w_1 w_2^\omega \notin \mathcal{L}(\mathcal{B})$. Since $w_1 w_2^j (w_2^k)^\omega = w_1 w_2^\omega$, we obtain $w_1 w_2^j (w_2^k)^\omega \notin \mathcal{L}(\mathcal{B})$, and thus $\neg LFT(g'; f, e)$. By Lemma 31, we have $\neg LFT_\mathsf{b}(g'; f, e)$. Therefore, $\neg RDGT_\mathsf{b}(\mathbf{g}'; \mathbf{f}, \mathbf{e})$. Since $g', f' \in G^f$ and $\trianglelefteq$ is reflexive and implies $\simeq_\mathsf{fb}$, Lemma 26 yields $g'; f' \simeq_\mathsf{fb} g' \fatsemi_\mathsf{b} f'$. Thus, we get $\mathbf{g}'; \mathbf{f}' \simeq_\mathsf{fb} \mathbf{g}' \fatsemi_\mathsf{b} \mathbf{f}'$.

Since $\mathbf{g} \sqsubseteq_\mathsf{fb} \mathbf{g}'$, then, by Lemma 10, we get $\mathbf{g} \fatsemi_\mathsf{b} \mathbf{f} \sqsubseteq_\mathsf{fb} \mathbf{g}' \fatsemi_\mathsf{b} \mathbf{f}' \simeq_\mathsf{fb} \mathbf{g}'; \mathbf{f}'$. Finally, by Lemma 9, we obtain $\neg RDGT_\mathsf{b}(\mathbf{g} \fatsemi_\mathsf{b} \mathbf{f}, \mathbf{e})$. □

Lemma 15 Let $(X,g), (Y,h)$ be metagraphs where all contained supergraphs are in S^R. If $\neg RDGT^M_\mathsf{b}((X,g),(Y,h))$ then $\mathcal{L}(\mathcal{A}) \not\subseteq \mathcal{L}(\mathcal{B})$.

Proof. Let $(X,g), (Y,h)$ be two metagraphs where all contained supergraphs are in S^R s.t. $\neg RDGT^M_\mathsf{b}((X,g),(Y,h))$.

We show that $\mathcal{L}(\mathcal{A}) \not\subseteq \mathcal{L}(\mathcal{B})$ by assuming the contrary and deriving a contradiction. So now we assume that $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$.

Since $g, h \in G^R$ there exist graphs $g', h' \in G^f$ s.t. $g \trianglelefteq g'$ and $h \trianglelefteq h'$.

Since $\neg RDGT^M_\mathsf{b}((X,g),(Y,h))$ we have $\neg LFT_\mathsf{b}(g,h)$ by Definition 9. By Lemma 31 we obtain $\neg LFT(g', h')$.

Let $Z_{g'h'}$ be the ω-regular language $\mathcal{L}(g') \cdot \mathcal{L}(h')^\omega$. The language $Z_{g'h'}$ is non-empty because $g', h' \in G^f$.

Since $\neg RDGT^M_\mathsf{b}((X,g),(Y,h))$ we have $LFT_\mathsf{f}(X,Y)$ by Definition 9.

By Definition 8, there is an arc $\langle p, a_0, q_0\rangle$ in X and an infinite sequence of arcs $\langle q'_0, a_1, q_1\rangle, \langle q'_1, a_2, q_2\rangle, \ldots$ in Y s.t. $p \in I_\mathcal{A}$, $q'_i \preceq_\mathsf{f} q_i$ for all $i \in \mathbb{N}$, and $a_j = 1$ for infinitely many $j \in \mathbb{N}$.

Since the number of states in automaton $\mathcal{A}$ is finite, we obtain that there exists a finite sequence of arcs $\langle q'_0, a_1, q_1\rangle, \langle q'_1, a_2, q_2\rangle, \ldots \langle q'_k, a_k, q_k\rangle$ in Y s.t. $q'_0 \preceq_\mathsf{f} q_k$, $q'_i \preceq_\mathsf{f} q_i$ for all $0 \leq i \leq k$, and $a_j = 1$ for at least one $j \in \{1, \ldots, k\}$.

By the condition that all supergraphs contained in $(X,g),(Y,h)$ are in S^R, we obtain $\langle\langle p,a_0,q_0\rangle,g\rangle \in S^R$ and $\langle\langle q'_{i-1},a_i,q_i\rangle,h\rangle \in S^R$ for $1 \leq i \leq k$. Thus $\langle\langle p,a_0,q_0\rangle,g'\rangle \in S^f$ and $\langle\langle q'_{i-1},a_i,q_i\rangle,h'\rangle \in S^f$ for $1 \leq i \leq k$. Therefore, there exists a word $w \in \mathcal{L}(g')$ with $p \xrightarrow{w} q_0$, and words $w_1,\ldots,w_k \in \mathcal{L}(h')$ with $q'_{i-1} \xrightarrow{w_i} q_i$ for $1 \leq i \leq k$ s.t. at least one accepting state is visited in the computation $q'_{j-1} \xrightarrow{w_j} q_j$.

Thus there exists some state q'_k with $q'_0 \preceq_{\mathsf{f}} q_k \preceq_{\mathsf{f}} q'_k$ s.t. $q'_0 \overset{w_1\ldots w_k}{\rightsquigarrow}_F q'_k$. By Lemma 34 we obtain $(w_1w_2\ldots w_k)^\omega \in \mathcal{L}(q'_0) \subseteq \mathcal{L}(q_0)$ (and this is even true when using delayed simulation $\preceq^{de}$ instead of direct simulation $\preceq_{\mathsf{f}}$). So we obtain $w(w_1w_2\ldots w_k)^\omega \in \mathcal{L}(p) \subseteq \mathcal{L}(\mathcal{A})$, because $p \in I_{\mathcal{A}}$. Furthermore, $w(w_1w_2\ldots w_k)^\omega \in Z_{g'h'}$ and thus $Z_{g'h'} \cap \mathcal{L}(\mathcal{A}) \neq \emptyset$.

Since we have $\neg LFT(g',h')$, by Lemma 19, we obtain $Z_{g'h'} \cap \mathcal{L}(\mathcal{B}) = \emptyset$. However, since $Z_{g'h'} \cap \mathcal{L}(\mathcal{A}) \neq \emptyset$, this is a contradiction to our assumption $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$. □

F Auxiliary Procedures and Proof of Correctness of the Algorithm

Here we describe the auxiliary procedures $Clean_1$, $Clean_2, Clean_3$ which are used in Algorithm 1. They perform the operation of function *Clean* (described in Section 6), plus some extra bookkeeping and optimizations described below.

Algorithm 2: *Procedure* $Clean_1$

Input: A minimized metagraph (X,g) and a set of minimized metagraphs M.
Output: Those parts of X for which there exist $\sqsubseteq_{\mathsf{fb}}$-smaller supergraphs in M are discarded.

```
foreach (Y,h) ∈ M and while X ≠ ∅ do
    if h ⊑^{∀∃} g then
        foreach ⟨p,a,q⟩ ∈ X do
            if ∃⟨p,a',q'⟩ ∈ Y. a' ≥ a ∧ q' ≽^A_f q then Remove ⟨p,a,q⟩ from X
```

The procedure $Clean_3$ is similar to $Clean_1$, except that the roles of the arguments is reversed.

The procedure $Clean_2$ is similar to $Clean_1$, but does some extra bookkeeping to handle the case where supergraphs in *Processed* regain the L-label.

Algorithm 3: *Procedure Clean*$_3$

Input: A set of minimized metagraphs M and a minimized metagraph (X,g).
Output: Those parts of the metagraphs in M for which there exist $\sqsubseteq_{\mathsf{fb}}$-smaller supergraphs in (X,g) are discarded. Empty metagraphs are discarded.

foreach $(Y,h) \in M$ **do**
  if $g \sqsubseteq^{\forall\exists} h$ **then**
    foreach $\langle p,a,q\rangle \in Y$ **do**
      if $\exists\langle p,a',q'\rangle \in X.\, a' \geq a \wedge q' \succeq_{\mathsf{f}}^{\mathcal{A}} q$ **then** Remove $\langle p,a,q\rangle$ from Y
    if $Y = \emptyset$ **then** Remove (Y,h) from M

Algorithm 4: *Procedure Clean*$_2$

Input: A minimized metagraph (X,g) and the set *Processed*.
Output: Those parts of X for which there exist $\sqsubseteq_{\mathsf{fb}}$-smaller supergraphs in *Processed* are discarded. Certain A-arcs in *Processed* may regain the L-label if they cause L-labeled arcs in X to be discarded. This then requires some extra RDGT.

foreach $(Y,h) \in$ *Processed and while* $X \neq \emptyset$ **do**
  if $h \sqsubseteq^{\forall\exists} g$ **then**
    regained $:= \emptyset$;
    foreach $\langle p,a,q\rangle \in X$ **do**
      if $\exists\langle p,a',q'\rangle \in Y.\, a' \geq a \wedge q' \succeq_{\mathsf{f}}^{\mathcal{A}} q$ **then**
        Remove $\langle p,a,q\rangle$ from X;
        if $L(\langle p,a,q\rangle) \wedge \neg L(\langle p,a',q'\rangle)$ **then**
          Label $\langle p,a',q'\rangle$ with L;
          regained $:=$ *regained* $\cup \{\langle p,a',q'\rangle\}$
    if *regained* $\neq \emptyset$ **then**
      if $\exists(Z,f) \in$ *Processed* : $\neg RDGT_{\mathsf{b}}^{M}((\mathit{regained},h),(Z,f))$ **then return** *FALSE*;

Theorem 1. Algorithm 1 terminates. It returns TRUE iff $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$.

Proof. **Termination.** First we show that all the auxiliary procedures used in Algorithm 1 always terminate. The functions C and $RDGT_{\mathsf{b}}$ are simple tests on finite metagraphs which always terminate. The functions Min_M and *RightExtend* are simple operations on finite metagraphs which always terminate. The operations $Clean_1$, $Clean_2$ and $Clean_3$ always terminate because their loops are just simple iterations through finite sets.

Now we consider the termination of Algorithm 1. First, the set M^1 of single letter metagraphs is finite and thus *Next* is finite on line 1. The loop of lines 2–6 successively removes elements from *Next* until it is empty and thus terminates. Parts of *Next* are added to the set *Init* which is therefore also finite, and thus *Next* is finite on line 7. The loop of lines 8–10 just iterates through finite sets and thus terminates. Now we consider the main loop of lines 11–21. All operations inside this main loop are just iterations through finite sets and thus terminate. The main loop itself removes elements from *Next* (line 12) and adds them to *Processed* (line 15; with the *L*-labels removed, but this does not matter here), but it also possibly adds new elements to *Next* (line 21). By contraposition, assume that the main loop does not terminate. Then, since the number of supergraphs is finite, some supergraph must be added to *Next* twice (as part of some metagraph). However, this is impossible, because newly created metagraphs are subjected to the *Clean* operation (lines 18-19) w.r.t. *Next* and *Processed*, and thus every already existing supergraph would be removed. Contradiction. Therefore, the main loop terminates. By combining the termination properties proved above, we obtain that Algorithm 1 terminates.

Correctness; First implication. We prove that if Algorithm 1 returns FALSE then $\mathcal{L}(\mathcal{A}) \not\subseteq \mathcal{L}(\mathcal{B})$. If the algorithm returns FALSE (either on lines 13–14, or on line 11 in procedure $Clean_2$), then there exist metagraphs $(X,g),(Y,h)$ s.t. $\neg RDGT_{\mathsf{b}}^{M}((X,g),(Y,h))$. Since we obtained these metagraphs from M^1 by applying the operations *RightExtend*, Min_M and $Clean_1$, $Clean_2$, $Clean_3$, it follows from Lemma 5 and Lemma 6 that all supergraphs contained in (X,g) and (Y,h) are in S^R. Thus, by Lemma 15, we obtain $\mathcal{L}(\mathcal{A}) \not\subseteq \mathcal{L}(\mathcal{B})$.

Correctness; Reverse implication. We prove that if $\mathcal{L}(\mathcal{A}) \not\subseteq \mathcal{L}(\mathcal{B})$ then Algorithm 1 returns FALSE. If $\mathcal{L}(\mathcal{A}) \not\subseteq \mathcal{L}(\mathcal{B})$ then, by Lemma 35, there exist supergraphs $\mathbf{g}',\mathbf{h}' \in S^f$ s.t. $\neg RDGT_{\mathsf{b}}(\mathbf{g}',\mathbf{h}')$.

By Lemma 3, all supergraphs from S^f (and thus in particular $\mathbf{g}',\mathbf{h}'$) can be generated by composing (w.r.t. ;) single letter supergraphs from S^1.

However, Algorithm 1 does not necessarily generate $\mathbf{g}',\mathbf{h}'$ because it uses *Min*-minimization, $\fatsemi_{\mathsf{b}}$-composition and $\sqsubseteq_{\mathsf{fb}}$-subsumption. If the algorithm would not discard any supergraphs because of $\sqsubseteq_{\mathsf{fb}}$-subsumption, then, by Lemma 5 and Lemma 6, it would generate a representative supergraph in S^R for every supergraph in S^f, i.e., it would generate supergraphs $\mathbf{g},\mathbf{h} \in S^R$ s.t. $\mathbf{g} \trianglelefteq \mathbf{g}'$ and $\mathbf{h} \trianglelefteq \mathbf{h}'$.

Algorithm 1 does not necessarily generate $\mathbf{g},\mathbf{h}$ because it discards supergraphs which are $\sqsubseteq_{\mathsf{fb}}$-larger than others (in the *Clean*-operations). However, in these cases the algorithm always retains the $\sqsubseteq_{\mathsf{fb}}$-smaller supergraphs. Since we generate new supergraphs by right-extension with single-letter supergraphs, by Lemma 10, the algorithm will generate supergraphs $\mathbf{e},\mathbf{f} \in S^R$ s.t. $\mathbf{e} \sqsubseteq_{\mathsf{fb}} \mathbf{g}$ and $\mathbf{f} \sqsubseteq_{\mathsf{fb}} \mathbf{h}$.

Since $\neg RDGT_{\mathsf{b}}(\mathbf{g}',\mathbf{h}')$ and $\mathbf{e} \sqsubseteq_{\mathsf{fb}} \mathbf{g} \sqsubseteq_{\mathsf{fb}} \mathbf{g}'$ and $\mathbf{f} \sqsubseteq_{\mathsf{fb}} \mathbf{h} \sqsubseteq_{\mathsf{fb}} \mathbf{h}'$, we obtain $\neg RDGT_{\mathsf{b}}(\mathbf{e},\mathbf{f})$ by Lemma 9.

If the optimization on line 15 of removing L-labels from supergraphs in *Processed* were not used then we would have the following situation. (The correctness of line 15 will be discussed below.) Since $\neg RDGT_{\mathsf{b}}(\mathbf{e},\mathbf{f})$ we have $\neg \mathsf{C}(\mathbf{e})$ by Lemma 11 and thus $L(\mathbf{e})$ because the L-label is assigned after testing condition C in line 10 and in function *RightExtend* in line 17. The supergraphs $\mathbf{e},\mathbf{f}$ will be created as parts of some metagraphs $(X,e),(Y,f)$. Since $\neg RDGT_{\mathsf{b}}(\mathbf{e},\mathbf{f})$ and $L(\mathbf{e})$, it follows from Definitions 4, 8 and 9, that $\neg RDGT^{M}_{\mathsf{b}}((X,e),(Y,f))$. Thus Algorithm 1 will return FALSE on lines 13-14.

Finally, we prove the correctness of the optimization of line 15. The problematic case is the following. Since we remove the L-label from supergraphs in *Processed*, it could happen that $\mathbf{e}$ is in *Processed* and $\neg L(\mathbf{e})$ in spite of the fact that $\neg \mathsf{C}(\mathbf{e})$. Thus we might not get $\neg RDGT^{M}_{\mathsf{b}}((X,e),(Y,f))$ as required, if $\mathbf{e}$ in (X,e). However, by Lemma 14, other supergraphs $\mathbf{e}',\mathbf{f}' \in S^{R}$ will be generated where $\mathbf{e}'$ is a single-letter right-extension of $\mathbf{e}$ and $\neg RDGT_{\mathsf{b}}(\mathbf{e}',\mathbf{f}')$. If $\mathbf{e}'$ (as part of some metagraph) is still in *Next* when it is RDGT tested with $\mathbf{f}'$ (as part of another metagraph) then the counterexample will be found here (by Definitions 4, 8 and 9 as above). Otherwise, we apply Lemma 14 again end expect the counterexample from another right-extension of $\mathbf{e}'$, and so on. This chain cannot go on indefinitely because we use subsumption. There are three ways how it can end.

1. Some multi-step extension of $\mathbf{e}$ eventually finds the counterexample and the algorithm returns FALSE.
2. The (multi-step)extension of $\mathbf{e}$ will be discarded because there is a $\sqsubseteq_{\mathsf{fb}}$-smaller supergraph $\hat{\mathbf{e}}$ in *Next*. This case can happen only finitely often, because there are no infinite decreasing $\sqsubseteq_{\mathsf{fb}}$-chains. In this case $\hat{\mathbf{e}}$ or one of its (multi-step) right-extensions can find the counterexample (by monotonicity of RDGT w.r.t. $\sqsubseteq_{\mathsf{fb}}$) and the algorithm correctly returns FALSE.
3. The extension of $\mathbf{e}$ will be discarded because there is a $\sqsubseteq_{\mathsf{fb}}$-smaller $\hat{\mathbf{e}}$ in *Processed*. In this case $\hat{\mathbf{e}}$ will regain the L-label by line 8 of procedure $Clean_2$. Furthermore, it will recover the skipped RDGT against elements of *Processed* in line 11 of procedure $Clean_2$. If $\mathbf{f}$ is in *Processed* by this time then the counterexample will be found here. Otherwise, if $\mathbf{f}$ is still in *Next* or if $\mathbf{f}$ will be generated later and then be in *Next*, then the counterexample will be found in the regular RDGT in line 14 of Algorithm 1. In every case the algorithm correctly returns FALSE.

By combining the two correctness implications shown above with the fact that Algorithm 1 terminates and returns TRUE iff it does not return FALSE, the result follows. □

G Further Optimizations and Implementation Details

Here we describe some optimizations used in our implementation and the command line options used to activate them.

Removing dead states. The option `-rd` removes dead states (and their transitions) from the input automata. Dead states are states that cannot be reached from any initial state, or states from which no accepting loop is reachable. Note that even reachable accepting states can be dead, because they might not be able to reach any accepting loop. Removing dead states preserves the language of Büchi automata, as required.

Using backward simulation. The option `-b` activates the use of backward-simulation for subsumption, as described in the paper. (Forward-simulation is always used.) This leads to a larger subsumption relation and thus to fewer generated supergraphs/metagraphs. On average, this makes the algorithm faster. However, it cannot be guaranteed that every single instance is solved more quickly, for several reasons. First, computing backward simulation has a small overhead. Second, a different subsumption relation can influence the search order, and thus a counterexample might be found only later. Finally, in some rare cases, implementation details of the used data-structures cause paradoxical results. E.g., the data-structure for storing sets of arcs controlled by the `-l` option (see below) works better if the backward simulation relation is small.

Quotienting. The option `-q` reduces the size of the input-automata by quotienting them w.r.t. forward-simulation, i.e., by collapsing forward-simulation equivalent states into a single state. This operation preserves the language (and even forward-simulation itself). Quotienting w.r.t. forward-simulation is almost always beneficial.

If additionally the option `-qr` is used, then the automata are repeatedly quotiented w.r.t. forward-simulation and backward-simulation until either a fixpoint is reached or the number of such operations exceeds a given threshold (currently 10). Quotienting w.r.t. forward-simulation/backward-simulation preserves the language. However, quotienting w.r.t. forward-simulation does not preserve backward-simulation, and vice-versa; thus the need for repeated quotienting. In most practical examples, the fixpoint is reached after 3-4 quotienting iterations. However, in the worst case the number of quotienting iterations could be very high, and thus we impose this limit of 10. The smaller automata obtained by repeated quotienting can make the algorithm faster, but this is not certain in every case. Since quotienting w.r.t. forward-simulation does not preserve backward-simulation (and vice-versa), this option yields a different subsumption on the obtained automaton, and this might unpredictably work either better or worse.

Using Forward-simulation between $\mathcal{A}$ and $\mathcal{B}$. The option `-c` activates the use of forward-simulation between the automata $\mathcal{A}$ and $\mathcal{B}$, as described in Section 5.

In the special case where forward-simulation holds even between the *initial* states of $\mathcal{A}$ and $\mathcal{B}$, condition C is true for *every* generated supergraph. Thus all L-labels are removed and the algorithm terminates immediately, reporting inclusion.

However, condition C can also help in other cases where simulation does not hold between the initial states but "more deeply" inside the automata. In such cases the number of L-labeled supergraphs drops to zero long before the set *Next* gets empty, and the algorithm reports language inclusion.

In order to maximize the chance of this early termination, we make another optimization: Our Algorithm (Section 7) maintains two sets of supergraphs/metagraphs called *Next* and *Processed*, where *Next* contains supergraphs that will generate new supergraphs by right extension, while the supergraphs in *Processed* have already done this.

By Lemma 14, if some supergraph $\mathbf{f}$ can find a counterexample (to inclusion) when used on the left in the RDGT, then at least one of its children (i.e., 1-letter right-extensions) can also find such a counterexample. Thus a supergraph $\mathbf{f}$ in *Processed* does not need to have the L-label (even if it does not satisfy C) because there is still some L-labeled child (i.e., right-extension) of $\mathbf{f}$ in *Next* which can find the counterexample instead (provided that any counterexample exists). I.e., the role of a supergraph $\mathbf{f}$ on the left side of the RDGT can be assumed by one of its own children. Therefore, the algorithm (in line 15) removes the L-label from all supergraphs that are moved from *Next* to *Processed*.

The only problematic case is when saturation occurs. Some supergraph $\mathbf{f}$ in *Processed* (now without the L-label) might be $\sqsubseteq_{\mathsf{fb}}$-smaller than another supergraph $\mathbf{f}' \in$ *Next* which has the L-label, causing $\mathbf{f}'$ to be discarded. This discarded supergraph $\mathbf{f}'$ might be the child (or descendent) of $\mathbf{f}$ which is needed to find the counterexample. (Another scenario is that $\mathbf{f}'$ was $\sqsubseteq_{\mathsf{fb}}$-smaller than this descendent of $\mathbf{f}$ and thus took its place.) So by discarding $\mathbf{f}'$ we might lose our chance to find the counterexample. We fix this problem in the following way. In this described case, the old supergraph $\mathbf{f}$ in *Processed* must regain the L-label, and some skipped RDGT-tests must be recovered. This is implemented in the procedure $Clean_2$ (Appendix F).

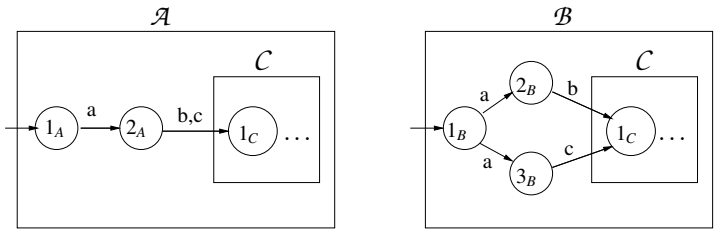

Fig. 2. Using condition C can help even if simulation does not hold between the initial states.

Example: Consider the example in Figure 2. $\mathcal{C}$ is a large automaton with initial state 1_C and the automata $\mathcal{A}$ and $\mathcal{B}$ are very similar.

We have $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$, but 1_B cannot simulate 1_A. The single-letter supergraph for letter 'a' has the form $\langle\langle 1_A, 0, 2_A\rangle, \{\langle 1_B, 0, 2_B\rangle, \langle 1_B, 0, 3_B\rangle\}\rangle$. Since it does not satisfy condition C, it retains the L-label. However, it cannot witness any counterexample, and gets moved from *Next* to *Processed*. Crucially, it loses the L-label when it moves to

Processed (line 15 in the algorithm) because its children in *Next* would be sufficient to find the counterexample (if any existed). The two children supergraphs in *Next*, corresponding to strings '*ab*' and '*ac*' respectively, are in this case identical and have the form $\langle\langle 1_A, 0, 1_C\rangle, \{\langle 1_B, 0, 1_C\rangle\}\rangle$. This supergraph in *Next* does satisfy the condition C and thus loses the L-label. Then there are no more supergraphs with the L-label and the algorithm reports inclusion. Note that this only works because the first supergraph lost the L-label when it moved to *Processed* because otherwise the number of L-labeled supergraphs would always have stayed ≥ 1.

Extended Forward-Simulation. The option `-fplus` activates the use of extended forward simulation, both inside automata $\mathcal{A}$ and $\mathcal{B}$ and between $\mathcal{A}$ and $\mathcal{B}$. The extended forward simulation relation $\preceq_{\mathsf{f}}^{+}$ is defined as follows. We have $p \preceq_{\mathsf{f}}^{+} q$ iff $p \preceq_{\mathsf{f}} q$ or $p \in F$ and $q \notin F$ and for every transition $p \xrightarrow{\sigma} p'$, $\sigma \in \Sigma$, there exists a transition $q \xrightarrow{\sigma} q'$ s.t. $q' \in F$ and $p' \preceq_{\mathsf{f}} q'$.

The extended forward simulation $\preceq_{\mathsf{f}}^{+}$ is larger than the standard direct forward-simulation $\preceq_{\mathsf{f}}$, and can replace $\preceq_{\mathsf{f}}$ for all different applications in our algorithm, namely quotienting, subsumption and condition C. Furthermore, $\preceq_{\mathsf{f}}^{+}$ is very easy to compute, once $\preceq_{\mathsf{f}}$ is given.

Intuitively, the relation $\preceq_{\mathsf{f}}^{+}$ allows to delay the acceptance condition by one step only. This is compatible with our subsumption relations on arcs and graphs because for an arc $\langle p, a, q\rangle$ to have label $a = 1$ it is sufficient (though not necessary) that $p \in F$ or $q \in F$. So delaying the acceptance condition by one step never changes the label. (This would not be true any more if one allowed a delay of 2 or more, because the computation witnessing the arc $\langle p, a, q\rangle$ might only be one step long.)

On the other hand, $\preceq_{\mathsf{f}}^{+}$ is still smaller than delayed simulation [6]. However, we do not use delayed simulation for several reasons:

- While delayed simulation can be used for quotienting and condition C, it cannot be used for subsumption on graphs. Intuitively, the 0/1-labels of arcs in graphs record the fact that some accepting state has been visited in a given finite computation between two states. While the relations $\preceq_{\mathsf{f}}$ and $\preceq_{\mathsf{f}}^{+}$ preserve this property, delayed simulation does not because it just expresses the obligation to visit some accepting state in the indefinite future (without any fixed bound). There are simple counterexamples that show that replacing direct simulation with delayed simulation in the subsumption would yield incorrect results.
- Unlike $\preceq_{\mathsf{f}}$ and $\preceq_{\mathsf{f}}^{+}$, delayed simulation is not very efficiently computable in practice.

The overhead for computing $\preceq_{\mathsf{f}}^{+}$ is negligible. On most examples, the effect of `-fplus` is small, but it can help in some cases (where many states, but not all, are accepting).

The new RDGT. The RDGT on metagraphs, as described in Def. 9, is optimized by using an abstraction of the metagraphs. In order to test $RDGT_{\mathsf{b}}^{M}((X,g),(Y,h))$, it is not necessary to know the metagraphs (X,g) and (Y,h) exactly. Instead, it suffices to know an abstraction of the test-relevant information in them, which depends on whether the metagraph is used on the left side or the right side of the RDGT. About X it is sufficient

to know which states in $\mathcal{A}$ can be reached from an initial state by some single arc in X. Let X_L be this set of states and $X_L \downarrow_f$ its downward-closure w.r.t. forward-simulation. (If some arc in X does not have the L-label, then its target state does not need to be considered because it will certainly not be part of any counterexample.) About Y it is sufficient to know from which states in $\mathcal{A}$ infinite accepting sequences start, where these sequences consist of Y-arcs and forward-simulation downward-jumps, as in the definition of LFT_{f} in Def. 8. Let Y_R be this set of states and $Y_R \uparrow_f$ its upward-closure w.r.t. forward-simulation. Then, we have $LFT_{\mathsf{f}}(X,Y) \;\Leftrightarrow\; X_L \downarrow_f \cap Y_R \uparrow_f \neq \emptyset$.

Similarly, about g it is sufficient to know which states in $\mathcal{B}$ can be reached from an initial state by some single arc in g. Let g_L be this set of states and $g_L \uparrow_b$ its upward-closure w.r.t. backward-simulation. About h it is sufficient to know from which states in $\mathcal{B}$ infinite accepting sequences start, where these sequences consist of h-arcs and backward-simulation upward-jumps, as in the definition of $\mathrm{LFT}_{\mathsf{b}}$ in Def. 5. Let h_R be this set of states and $h_R \downarrow_b$ its downward-closure w.r.t. backward-simulation. Then we have $LFT_{\mathsf{b}}(g,h) \;\Leftrightarrow\; g_L \uparrow_b \cap h_R \downarrow_b \neq \emptyset$. For each metagraph these respective sets of states (for left and right roles) are computed *only once* when the metagraph is created, and then stored separately (as bitvectors). Since the RDGT now only needs to check the non-emptiness of the intersection of sets of states, it can be done very efficiently by operations on bitvectors.

Moreover, a separate subsumption relation is applied to this test-relevant information. Consider two metagraphs (X^1, g^1) and (X^2, g^2) and their test-relevant information for the left role $(X^1_L \downarrow_f, g^1_L \uparrow_b)$ and $(X^2_L \downarrow_f, g^2_L \uparrow_b)$. If $X^1_L \downarrow_f \supseteq X^2_L \downarrow_f$ and $g^1_L \uparrow_b \subseteq g^2_L \uparrow_b$ then the information $(X^2_L \downarrow_f, g^2_L \uparrow_b)$ can be discarded. (Similarly for the right roles.) Indeed in most examples the number of different combinations of test-relevant information (for left and right) is much lower than the number of generated metagraphs.

Note that the test-relevant information $(X_L \downarrow_f, g_L \uparrow_b)$ and $(X_R \uparrow_f, g_R \downarrow_b)$ for a metagraph (X,g) cannot completely replace this metagraph itself. This is because it does not encode enough information to generate new metagraphs by right-extension. Intuitively, the property that a state is part of a loop of the form $(ab)^\omega$ gives no information about loops of the form $(abc)^\omega$, or vice-versa.

The option `-v` (verbose) displays more information about the current status of the algorithm. It shows the number of metagraphs in *Next* and *Processed*, as well as the number of different test-relevant information (for left and right) derived from metagraphs in *Processed*.

Layered Arc Sets. The option `-l` (layered arc sets) activates a different internal data-structure for storing sets of arcs. It happens very often that the algorithm searches a set of arcs for an arc with a particular left end point. This data-structure makes it very easy to access the subset of arcs with a given left end point, which increases the speed of the algorithm significantly for larger automata.

BFS vs. DFS vs. SFS The default search strategy is breadth-first search (BFS). Here the set *Next* behaves as a queue where metagraphs are added at the end and removed from the front. The option `-DFS` switches the strategy to depth-first search, where *Next* behaves as a stack and metagraphs are added and removed at the front. In most cases where language inclusion holds, BFS performs slightly better than DFS. In those cases

where there is a counterexample, BFS and DFS are incomparable, i.e., one could find the counterexample much earlier than the other, or vice-versa. It would be possible to try various other search heuristics here. We consider another heuristic `-SFS` (smallest-first search), which picks the metagraph from *Next* that has the $\subseteq$-smallest graph. It often performs better than BFS, but not uniformly.

Summary.

- In general, the best performance is achieved with the options `-q -rd -fplus -l -b -qr -c`.
- If this fails, then one might try running it without the `-qr` option, or change the search strategy with the `-DFS` or `-SFS` option.
- By using the verbose option `-v`, one can track the progress of the algorithm. If the size of the set *Next* (first column) is large, then the algorithm is unlikely to report the result 'Inclusion' soon. However, it could still report 'Non-inclusion' at any moment.

Table 5. Peterson. Simulation holds between initial states. One may find that with the option `-c` (using the simulation between automata), the algorithm may detect simulation between the initial states while doing repeated quotienting w.r.t. forward/backward simulation (option `-qr`) and terminate before the repeated quotienting is done. This also happens in many other tables.

Version	$\mathcal{A}$		$\mathcal{B}$		Minimized $\mathcal{A}$		Minimized $\mathcal{B}$		# of	Time
	Trans.	States	Trans.	States	Trans.	States	Trans.	States	Metagraphs	
default	33	20	34	20	22	14	13	9	126	0.39s
`-b`					22	14	13	9	126	0.54s
`-b -qr`					22	14	10	7	126	0.61s
`-b -qr -c`					29	18	34	20	0	0.3s
`-b -qr -c -DFS`					29	18	34	20	0	0.2s
Algorithm of [1]					22	14	13	9	-	0.46s

Table 6. Phils. Simulation holds between initial states.

Version	$\mathcal{A}$		$\mathcal{B}$		Minimized $\mathcal{A}$		Minimized $\mathcal{B}$		# of	Time
	Trans.	States	Trans.	States	Trans.	States	Trans.	States	Metagraphs	
default	49	23	482	161	48	22	324	108	49964	11h3m
`-b`					48	22	324	108	43301	7h21m
`-b -qr`					48	22	324	108	43301	7h23m
`-b -qr -c`					48	22	482	161	0	0.1s
`-b -qr -c -DFS`					29	18	482	161	0	0.12s
Algorithm of [1]					48	22	324	108	-	12h36m

H Further Details of the Experiments

For the experiments on models from [14], we provide further data in this section, e.g, the size of the automata after minimization (dead state removal and quotienting) and the number of metagraphs added to the set *Next*. In Table 5-18, we present the results of several different versions of the algorithm and also the algorithm of [1]. The default options are `-q -rd -fplus -l`, and for a row that begins with `-b -qr`, we mean that the options `-q -rd -fplus -l -b -qr` are enabled.

The results of the pretest on the Tabakov-Vardi random can be found in Table 19. Here one can observe that for cases where simulation holds between initial states, the time needed is negligible. Also the time needed to find counterexamples is very small. Only the "inclusion" cases are interesting. Based on the above observation, we picked two important configurations (highlighted in the table) (Hard: td=2, ad=0.1, size=30) and (Easy, but nontrivial: td=3, ad=0.6, size=50) for larger experiments. In both the above two configurations, the percentage of the "inclusion" cases are close to 50% and the time needed is not negligible.

Table 7. Mcs. Simulation holds between initial states.

Version	$\mathcal{A}$		$\mathcal{B}$		Minimized $\mathcal{A}$		Minimized $\mathcal{B}$		# of	Time
	Trans.	States	Trans.	States	Trans.	States	Trans.	States	Metagraphs	
default	3222	1408	21503	7963	19	12	213	94	2172	2m43s
`-b`					19	12	213	94	2052	2m32s
`-b -qr`					17	11	211	93	2021	2m49s
`-b -qr -c`					48	22	482	161	0	1m24s
`-b -qr -c -DFS`					29	18	482	161	0	1m25s
Algorithm of [1]					48	22	324	108	-	>24h

Table 8. Bakery. Simulation holds between initial states.

Version	$\mathcal{A}$		$\mathcal{B}$		Minimized $\mathcal{A}$		Minimized $\mathcal{B}$		# of	Time
	Trans.	States	Trans.	States	Trans.	States	Trans.	States	Metagraphs	
default	2703	1510	2702	1509	1364	766	1365	767	-	>24h
`-b`					1364	766	1365	767	-	>24h
`-b -qr`					826	497	827	498	-	>24h
`-b -qr -c`					2633	1468	2632	1467	0	12s
`-b -qr -c -DFS`					2633	1468	2632	1467	0	12s
Algorithm of [1]					1364	766	1365	767	-	>24h

Table 9. Fischer. Simulation holds between initial states. Here the option `-b` (using backward simulation) helps to reduces the number of metagraphs added to *Next* by almost 50%. However, the run time is still longer than the default version. We did some further experiments and find that this is caused by using the option `-l` (layered arc sets). The option `-l` works better when backward simulation is smaller. We tried to disable the `-l` option and again compare the two versions. The version without `-b` needs 12m37s while the one with `-b` uses only 4m1s.

Version	$\mathcal{A}$		$\mathcal{B}$		Minimized $\mathcal{A}$		Minimized $\mathcal{B}$		# of	Time
	Trans.	States	Trans.	States	Trans.	States	Trans.	States	Metagraphs	
default	1395	634	3850	1532	60	28	1176	426	1075	2m38s
`-b`					60	28	1176	426	596	2m50s
`-b -qr`					40	16	774	219	323	27s
`-b -qr -c`					1395	634	3850	1532	0	3.6s
`-b -qr -c -DFS`					1395	634	3850	1532	0	3.7s
Algorithm of [1]					60	28	1176	426	-	4h50m

Table 10. FischerV2. Simulation holds between initial states. Here one finds that `-b` helps a lot, even though the number of metagraphs is not much lower. This is because in this example backward simulation helps a lot in reducing the size (number of arcs) of each metagraph. We record the sum of the numbers of arcs of all the generated metagraphs after minimization. The version without `-b` generates 771783 arcs while the one with `-b` generates only 196143 arcs.

Version	$\mathcal{A}$		$\mathcal{B}$		Minimized $\mathcal{A}$		Minimized $\mathcal{B}$		# of	Time
	Trans.	States	Trans.	States	Trans.	States	Trans.	States	Metagraphs	
default	147	56	147	56	147	56	147	56	1534	5m14s
`-b`					147	56	147	56	1361	1m26s
`-b -qr`					147	56	147	56	1361	1m1s
`-b -qr -c`					147	56	147	56	0	0.1s
`-b -qr -c -DFS`					147	56	147	56	0	0.1s
Algorithm of [1]					147	56	147	56	-	13m15s

Table 11. FischerV3. Language inclusion holds, but simulation does not hold between initial states.

Version	$\mathcal{A}$		$\mathcal{B}$		Minimized $\mathcal{A}$		Minimized $\mathcal{B}$		# of	Time
	Trans.	States	Trans.	States	Trans.	States	Trans.	States	Metagraphs	
default	1400	637	1401	638	127	59	126	58	961	45s
`-b`					127	59	126	58	431	10s
`-b -qr`					97	40	95	37	653	11s
`-b -qr -c`					97	40	95	37	0	7s
`-b -qr -c -DFS`					97	40	95	37	0	9s
Algorithm of [1]					127	59	126	58	-	3h6m

Table 12. FischerV4. Language inclusion holds, but simulation does not hold between initial states.

Version	$\mathcal{A}$		$\mathcal{B}$		Minimized $\mathcal{A}$		Minimized $\mathcal{B}$		# of	Time
	Trans.	States	Trans.	States	Trans.	States	Trans.	States	Metagraphs	
default	147	56	1506	526	147	56	1381	451	-	>24h
`-b`					147	56	1381	451	4631	1h31m
`-b -qr`					147	56	1375	449	5091	2h12m
`-b -qr -c`					147	56	1375	449	5091	2h12m
`-b -qr -c -DFS`					147	56	1375	449	6665	1h37m
Algorithm of [1]					147	56	1381	451	-	>24h

Table 13. BakeryV2. Language inclusion holds, but simulation does *not hold* between the initial states. Note that option `-c` helps a lot here.

Version	$\mathcal{A}$		$\mathcal{B}$		Minimized $\mathcal{A}$		Minimized $\mathcal{B}$		# of	Time
	Trans.	States	Trans.	States	Trans.	States	Trans.	States	Metagraphs	
default	2090	1149	2091	1150	1241	674	1240	673	-	>24h
`-b`					1241	674	1240	673	-	>24h
`-b -qr`					1241	674	1240	673	-	>24h
`-b -qr -c`					1241	674	1240	673	4	18s
`-b -qr -c -DFS`					1241	674	1240	673	-	>24h
Algorithm of [1]					1241	674	1240	673	-	>24h

Table 14. BakeryV3. Language inclusion does not hold.

Version	$\mathcal{A}$		$\mathcal{B}$		Minimized $\mathcal{A}$		Minimized $\mathcal{B}$		# of	Time
	Trans.	States	Trans.	States	Trans.	States	Trans.	States	Metagraphs	
default	2090	1149	2697	1506	1241	674	1517	853	0	5s
`-b`					1241	674	1517	853	0	6s
`-b -qr`					1241	674	1052	620	0	16s
`-b -qr -c`					1241	674	1052	620	0	15s
`-b -qr -c -DFS`					1241	674	1052	620	0	16s
Algorithm of [1]					1241	674	1517	853	-	12m19s

Table 15. FischerV5. Language inclusion does not hold.

Version	$\mathcal{A}$		$\mathcal{B}$		Minimized $\mathcal{A}$		Minimized $\mathcal{B}$		# of	Time
	Trans.	States	Trans.	States	Trans.	States	Trans.	States	Metagraphs	
default	3850	1532	1420	643	1176	426	64	29	988	1m6s
`-b`					1176	426	64	29	852	1m47s
`-b -qr`					774	219	42	16	702	39s
`-b -qr -c`					774	219	42	16	702	36s
`-b -qr -c -DFS`					774	219	42	16	1686	49s
Algorithm of [1]					1176	426	64	29	-	7h28m

Table 16. PhilsV2. Language inclusion does not hold.

Version	$\mathcal{A}$		$\mathcal{B}$		Minimized $\mathcal{A}$		Minimized $\mathcal{B}$		# of	Time
	Trans.	States	Trans.	States	Trans.	States	Trans.	States	Metagraphs	
default	482	161	212	80	324	108	48	22	30	0.7s
`-b`					324	108	48	22	30	0.8s
`-b -qr`					324	108	48	22	30	1s
`-b -qr -c`					324	108	48	22	30	1s
`-b -qr -c -DFS`					324	108	48	22	1120	11.5s
Algorithm of [1]					324	108	48	22	-	1.1s

Table 17. PhilsV3. Language inclusion does not hold.

Version	$\mathcal{A}$		$\mathcal{B}$		Minimized $\mathcal{A}$		Minimized $\mathcal{B}$		# of	Time
	Trans.	States	Trans.	States	Trans.	States	Trans.	States	Metagraphs	
default	464	161	212	80	392	134	48	22	30	0.7s
`-b`					392	134	48	22	30	0.8s
`-b -qr`					392	134	48	22	30	1.2s
`-b -qr -c`					392	134	48	22	30	1.1s
`-b -qr -c -DFS`					392	134	48	22	1875	25.7s
Algorithm of [1]					392	134	48	22	-	1s

Table 18. PhilsV4. Language inclusion does not hold.

Version	$\mathcal{A}$		$\mathcal{B}$		Minimized $\mathcal{A}$		Minimized $\mathcal{B}$		# of	Time
	Trans.	States	Trans.	States	Trans.	States	Trans.	States	Metagraphs	
default	482	161	464	161	324	108	392	134	144	3.8s
`-b`					324	108	392	134	144	4.5s
`-b -qr`					324	108	392	134	144	4.8s
`-b -qr -c`					324	108	392	134	144	4.8s
`-b -qr -c -DFS`					324	108	392	134	5460	6m12s
Algorithm of [1]					324	108	392	134	-	10.7s

Table 19. Results of the Tabakov-Vardi experiments on automata of size 15. We let $td = 1.5, 2, 2.5, 3$ and $ad = 0.1, 0.2, \ldots, 1.0$. For each combination of *td* and *ad*, we generate 100 pairs of BA of size 15 (i.e., 4000 automata in total), and test language inclusion with a timeout of 10 minutes on an Intel Xeon 2.66GHz processor with 4GB memory. For the new algorithm, the options `-q -rd -l -fplus -b -qr -c` are enabled.

td	type		Algorithm of [1]										New Algorithm									
			ad										ad									
			0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1**
1.5	Sim	Time	87.3	34.58	15.75	117.38	159.14	43.61	28.48	46.53	26.57	7.15	0.01	0.02	0.01	0.02	0.02	0.01	0.02	0.03	0.02	0.02
		%	3%	1%	3%	4%	8%	7%	5%	6%	11%	7%	3%	1%	3%	4%	8%	7%	5%	6%	11%	7%
	Inc	Time	183.42	149.46	102.55	107.52	109.75	131.76	92.07	76.93	112.91	94.45	177.19	124.95	125.6	124.36	121.37	172.28	106.98	84.54	129.02	108.97
		%	10%	13%	16%	16%	17%	17%	16%	11%	10%	14%	10%	13%	16%	16%	17%	17%	16%	11%	10%	14%
	nInc	Time	0.06	0.08	0.06	0.06	0.12	0.11	0.08	0.09	0.13	0.09	0.13	0.15	0.13	0.13	0.19	0.17	0.15	0.16	0.2	0.17
		%	85%	84%	81%	78%	72%	76%	76%	79%	74%	77%	85%	84%	81%	78%	72%	76%	76%	79%	74%	77%
	TO	%	2%	2%	0%	2%	3%	0%	3%	4%	5%	2%	2%	2%	0%	2%	3%	0%	3%	4%	5%	2%
2	Sim	Time	0	35.53	6.27	4.93	0.59	3.06	1.42	0.82	0.59	0.27	0	0.02	0.02	0.03	0.03	0.02	0.03	0.03	0.04	0.04
		%	0%	1%	4%	9%	2%	7%	15%	19%	45%	53%	0%	1%	4%	9%	2%	7%	15%	19%	45%	53%
	Inc	Time	**45.04**	28.9	20.47	19.29	18.33	18.13	13.9	15.91	10.43	7.16	**46.15**	28.39	18.13	17.16	15.75	17.75	11.62	15.07	9.7	6.22
		%	**47%**	48%	48%	51%	55%	51%	54%	45%	21%	12%	**47%**	48%	48%	51%	55%	51%	54%	45%	21%	12%
	nInc	Time	0.08	0.08	0.07	0.08	0.06	0.08	0.08	0.06	0.06	0.07	0.14	0.16	0.14	0.15	0.13	0.16	0.14	0.12	0.1	0.11
		%	53%	51%	48%	40%	43%	42%	31%	36%	34%	35%	53%	51%	48%	40%	43%	42%	31%	36%	34%	35%
	TO	%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
2.5	Sim	Time	0.67	0.43	1.21	0.43	0.31	0.5	0.25	0.16	0.17	0.1	0.04	0.03	0.04	0.03	0.04	0.04	0.03	0.04	0.04	0.04
		%	1%	1%	5%	8%	13%	19%	44%	61%	69%	74%	1%	1%	5%	8%	13%	19%	44%	61%	69%	74%
	Inc	Time	14.23	9.62	6.59	6.11	5.13	4.05	2.34	1.33	0.61	1.16	10.91	6.86	4.72	4.22	3.83	2.88	1.68	0.91	0.41	0.87
		%	79%	83%	78%	72%	70%	64%	44%	27%	13%	3%	79%	83%	78%	72%	70%	64%	44%	27%	13%	3%
	nInc	Time	0.11	0.06	0.07	0.07	0.09	0.07	0.06	0.08	0.07	0.08	0.15	0.11	0.12	0.12	0.14	0.14	0.1	0.11	0.08	0.09
		%	20%	16%	17%	20%	17%	17%	12%	12%	18%	23%	20%	16%	17%	20%	17%	17%	12%	12%	18%	23%
	TO	%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
3	Sim	Time	0.67	0.62	0.38	0.44	0.27	0.19	0.13	0.12	0.1	0.1	0.03	0.04	0.04	0.03	0.03	0.03	0.04	0.04	0.04	0.04
		%	2%	2%	6%	18%	33%	34%	66%	72%	91%	95%	2%	2%	6%	18%	33%	34%	66%	72%	91%	95%
	Inc	Time	7.47	5.23	3.58	3.1	2.17	**1.5**	0.65	0.5	0.1	0	4.99	3.38	2.27	2.03	1.55	**1.02**	0.41	0.33	0.06	0
		%	89%	91%	89%	76%	60%	**56%**	25%	18%	2%	0%	89%	91%	89%	76%	60%	**56%**	25%	18%	2%	0%
	nInc	Time	0.07	0.07	0.06	0.07	0.06	0.07	0.08	0.07	0.08	0.08	0.11	0.15	0.1	0.17	0.13	0.1	0.1	0.09	0.08	0.07
		%	9%	7%	5%	6%	7%	10%	9%	10%	7%	5%	9%	7%	5%	6%	7%	10%	9%	10%	7%	5%
	TO	%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%