RICE UNIVERSITY

Büchi Containment and Size-Change Termination

by

Seth Fogarty

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Master of Science

Approved, Thesis Committee:

Professor Moshe Y. Vardi, Chair
Karen Ostrum George Professor
Department of Computer Science

Professor Keith Cooper
Departments of Computer Science and
Electrical and Computer Engineering

Assistant Professor Luay K. Nakhleh
Department of Computer Science

Houston, Texas

May, 2008

Abstract

Büchi Complementation and Size-Change Termination

by

Seth Fogarty

We compare tools for complementing nondeterministic Büchi automata with a recent termination-analysis algorithm. Complementation of Büchi automata is a well-explored problem in program verification. Early solutions using a Ramsey-based combinatorial argument have been supplanted by rank-based constructions with exponentially better bounds. In 2001 Lee et al. presented the size-change termination (SCT) problem, along with both a reduction to Büchi automata and a Ramsey-based algorithm. This algorithm strongly resembles the initial complementation constructions for Büchi automata. This leads us to wonder if theoretical gains in efficiency are mirrored in empirical performance.

We prove the SCT algorithm is a specialized realization of the Ramsey-based complementation construction. Doing so allows us to generalize SCT solvers to handle Büchi automata. We experimentally demonstrate that, surprisingly, Ramsey-based approaches are superior over the domain of SCT problems, while rank-based approaches dominate automata universality tests. This reveals several interesting properties of the problem spaces and both approaches.

Acknowledgments

I cannot overstate the extent of my gratitude to my advisor Professor Moshe Y. Vardi. Beyond the significant academic guidance Dr. Vardi has provided, he has helped me to find the enthusiasm, focus, and joy that have enabled this work. I would like to thank Dr. Keith Cooper for his guidance, Dr. Luay K. Nakhleh for his advice, and both for being members on my thesis committee.

I am grateful to Chin Soon Lee and Amir Ben-Amram for sct/scp, the C implementation of SCT and it's polynomial-time approximation, to Neil Jones for pointers to some wonderful papers, and to all three for allowing me to use their SCT implementation, SCTP, which has been essential to my experiments. I want to thanks Damien Sereni for his OCaml implementation of the SCT principal, which provided many of the test cases for my experiments. I offer thanks to Laurent Doyen and Jean-François Raskin for letting me use and helping explain the source code for Mh, the rank-based Büchi universality solver with subsumption.

I owe a significant debt to Deian Tabakov for his continual willingness to provide perspective and advice, as well as for the Baccus symbolic Büchi universality solver and random automata generator. I offer thanks to all the members of the Formal Verification group who have made me feel welcome. Finally, I would like to thank my parents, Gail and Thomas, for their encouragement and firm support in my quest for knowledge.

This work was supported in part by NSF grant CCF-0728882.

Contents

1 Introduction **1**
- 1.1 The Story So Far . . . 1
- 1.2 Contributions . . . 2
- 1.3 Outline . . . 3

2 Preliminaries **4**
- 2.1 Ramsey-Based Universality . . . 4
- 2.2 Rank-Based Complementation . . . 7
- 2.3 Size-Change Termination . . . 8

3 Size-Change Termination and Ramsey-Based Containment **12**
- 3.1 Composition: Trading Space for Time . . . 12
- 3.2 Strongly Suffix Closed Languages . . . 14
- 3.3 Ramsey-Based Containment with Supergraphs . . . 15
- 3.4 Composition of Supergraphs . . . 17
- 3.5 Strong Suffix Closure and Containment . . . 19
- 3.6 From Ramsey-Based Containment to Size-Change Termination . . . 20

4 Towards an Empirical Comparison **24**
- 4.1 A Tighter Reduction from SCT to Büchi Containment . . . 24
- 4.2 Subsumption in the Single-Graph Search . . . 25
- 4.3 Search Space for the Two-Graph Search . . . 32
- 4.4 Grounded Supergraphs: Single-Graph Containment Test for Arbitrary Languages . . . 35

5 Experimental Results **45**
- 5.1 Tools: . . . 45
  - 5.1.1 Notation . . . 46
- 5.2 Size-Change Termination . . . 47
  - 5.2.1 Experimental Setup: . . . 47
  - 5.2.2 Experiment Results: . . . 48
- 5.3 Büchi automata . . . 51
  - 5.3.1 Experiment Setup . . . 51
  - 5.3.2 Experiment Results . . . 53

6 Conclusion **58**
- 6.1 Future Work . . . 59

List of Figures

2.1 A size-change terminating SCT problem. 9

4.1 Subsumption: two supergraphs $\widehat{g}$ and $\widehat{h}$, where $\widehat{g} \preceq \widehat{h}$. 26

5.1 Improvement in size obtained with revised reduction of SCT problems to Büchi automata. 49
5.2 Scaling of original and revised reductions from SCT problems to Büchi automata containment problems. 49
5.3 Scaling of Ramsey and rank-based approaches on SCT problems. . . . 51
5.4 Mh program: timeout percentage vs. acceptance/final Density (200 states). 53
5.5 SCTP program: timeout percentage vs. acceptance/final density (20 states). 54
5.6 sct/scp program: timeout percentage vs. acceptance/final density (60 and 100 states). 54
5.7 Scaling of Ramsey and rank-based approaches on Büchi automata universality problems. 55

Chapter 1
Introduction

1.1 The Story So Far

The automata-theoretic approach to formal program verification reduces questions about program adherence to a specification to questions about language containment. Representing liveness, fairness, or termination properties requires finite automata that operate on infinite words. One automaton, $\mathcal{A}$, encodes the behavior of the program, while another automaton, $\mathcal{B}$, encodes the formal specification. To ensure that the language of $\mathcal{A}$ is contained in the language of $\mathcal{B}$, verify that the intersection of $\mathcal{A}$ with the complement of $\mathcal{B}$ is empty. Unfortunately, for nondeterministic automata computing the complementary automata $\overline{\mathcal{B}}$ is PSPACE-hard [20]. A key problem in this framework is finding an effective algorithm to determine the containment of finite automata on infinite words.

Finite automata on infinite words are classified by their acceptance condition and transition structure. We here consider nondeterministic Büchi automata, which have a nondeterministic transition function, and in which a run is accepting when it visits at least one accepting state infinitely often. Michel provides a lower bound of $2^{O(n \log n)}$ to the complementation Büchi automata [13].

The first complementation constructions for nondeterministic Büchi automata employed a Ramsey-based combinatorial argument to partition infinite words into a finite set of regular languages. Proposed by Büchi in 1962, the original construction involved a doubly-exponential blowup in the state space [3]. In 1987 Sistla, Vardi, and Wopler demonstrated a $2^{O(n^2)}$ implementation [15], bringing the complementation problem into singly-exponential space but leaving a sizable gap between the solution and lower bound.

The gap was tightened in 1988, when Safra described a determinization construction for Büchi automata that provided a nearly matching $2^{O(n \log n)}$ construction. Work since then has focused on improving the practicality of $2^{O(n \log n)}$ constructions, either by providing simpler constructions, further tightening the bound, or improving the derived algorithms. In 2001 Kupferman and Vardi employed a rank-based analysis of Büchi automata to provide a simpler complementation procedure. Gurumurthy et al. combined this construction with minimization techniques [10]. Recently Doyen and Raskin tightly integrated the rank-based construction with a subsumption relation and a lasso-finding algorithm to provide a complementation solver that scales to automata several orders of magnitude larger than previous tools [6].

Separately in 2001, in the context of of program termination analysis, Lee, Jones,

and Ben-Amram presented the size-change termination principle [12]. This principle states that, for domains with well-founded values, if every infinite computation contains an infinitely decreasing value sequence, then no infinite computation is possible. Lee et al. describe a method of size-change termination analysis centered around a verification problem, and reduce this problem to the containment of two Büchi automata. Stating the lack of efficient Büchi containment solvers, they also propose a Ramsey-based combinatorial solution that captures all possible call sequences in a finite set of graphs. This algorithm was provided as a practical alternative to reducing the verification problem to Büchi containment, but bears a striking resemblance to the 1987 Ramsey-based complementation construction.

1.2 Contributions

We investigate the connection between these two Ramsey-based approaches. We first borrow from the Lee, Jones, and Ben-Amram (LJB) algorithm to provide a concrete universality-testing algorithm based on the 1987 Ramsey-based complementation construction. Observing differences between this algorithm and the SCT algorithm, we define a strong suffix closure property of languages and simplify the universality-testing algorithm for languages with this property. We then show how to extend this universality testing algorithm to Büchi automata containment problems. Finally we formally prove that the LJB size-change termination algorithm is a specialized application of the 1987 Ramsey-based complementation construction.

Having shown the intimate connection between SCT analysis and Büchi automata containment, we are presented with a rank-based containment algorithm implemented only in the formal-verifications community, and a Ramsey-based containment algorithm implemented only in the programming-languages community. To empirically compare their performance, we level the playing field by strengthening each algorithm. First, to fairly compare the two approaches on the domain of SCT problems, we describe an improved reduction from SCT to Büchi containment and formalize a subsumption relation presented as an optimization of the LJB algorithm in [2]. Second, we turn our eye towards Büchi automata containment problems. We examine some simple improvements to the Ramsey-based containment algorithm that simplify the search space in the general case, and leverage this improvement to extend the subsumption relation to arbitrary containment problems.

The practical performance of the two approaches can now be directly compared. First, we explore Lee et al.'s intuition that Ramsey-based algorithms are more practical than Büchi complementation tools on SCT problems. We experimentally measure the strength of existing SCT tools and best-of-breed Büchi complementation solvers on size-change termination problems extracted from the literature. The sparsity of such problems, and the difficulty in generating more, prevent us from developing a clear picture of this space. What is clear is that, despite the vast chasm between $2^{O(n^2)}$ and $2^{O(n \log n)}$, Ramsey-based tools are competitive with or even superior to

rank-based Büchi containment solvers.

To further explore this surprising development, we investigate the model of random automata proposed by Tabakov and Vardi [17] and used to evaluate automata-theoretic tools. We pit the same solvers against SCT tools generalized to handle arbitrary Büchi containment. We discover that, while Ramsey-based algorithms compare well with rank-based algorithms, the best Ramsey-based tools do not scale as well as the newest rank-based tools. Also notable is the observation that use of a subsumption relations, especially in combination with operations specialized for this relation, is essential to the practical implementation of either algorithm.

1.3 Outline

Chapter 2 discusses the relevant details of existing constructions and algorithms. In Chapter 3 we prove the connection between the 1987 Ramsey-based construction and the LJB algorithm for deciding SCT. Chapter 4 provides the foundations for empirically testing the performance of Ramsey-based and rank-based algorithms. In chapter 5 we compare the performance of each approach on two problem domains: SCT problems from the literature and a space of random automata. Finally, Chapter 6 examines the discrepancy in performance between the SCT problem space and the space of random automata.

Chapter 2
Preliminaries

In this section we review the relevant details of the Ramsey-based complementation construction [15], rank-based complementation [11], and size-change termination [12], introducing along the way the notation used throughout this thesis. An *nondeterministic Büchi automaton on infinite words* is a tuple $\mathcal{B} = \langle \Sigma, Q, Q^{in}, \rho, F \rangle$, where Σ is a finite nonempty alphabet, Q a finite nonempty set of states, $Q^{in} \subseteq Q$ a set of initial states, $F \subseteq Q$ a set of accepting states, and $\rho : Q \times \Sigma \to 2^Q$ a nondeterministic transition relation. We lift the ρ function to sets of states, and then to words of arbitrary length. For a set $R \subseteq Q$ of states and a character $a \in \Sigma$, let $\rho(R, a)$ be $\{s \mid r \in R, s \in \rho(r, a)\}$. For a set of states $R \subseteq Q$, let $\rho(R, \epsilon) = R$. For each $a \in \Sigma$ $u \in \Sigma^*$, define $\rho(R, au) = \rho(\rho(R, a), u)$.

A *run* of a Büchi automaton $\mathcal{B}$ on a word $w \in \Sigma^\omega$ is a infinite sequence of states $q_0 q_1 ... \in Q^\omega$ such that $q_0 \in Q^{in}$ and, for every $d \geq 0$, we have $q_{d+1} \in \rho(q_d, w_d)$. A run is *accepting* iff $q_d \in F$ for infinitely many $d \in \mathbb{N}$. A word $w \in \Sigma^\omega$ is accepted by $\mathcal{B}$ if there is an accepting run of $\mathcal{B}$ on w. The words accepted by $\mathcal{B}$ form the language of $\mathcal{B}$, denoted by $L(\mathcal{B})$. A *path* in $\mathcal{B}$ from q to r over $w = w_0 w_1 ... w_{n-1}$ is a finite sequence of states $p_0, ..., p_n$, such that $p_0 = q$, $p_n = r$, and for every $0 \leq d < n$, we have $p_{d+1} \in \rho(p_d, w_d)$. A path is *accepting* if $p_d \in F$ for some $0 \leq d \leq n$. A state r is *reachable* by a word w in an automaton $\mathcal{B}$ when $r \in \rho(Q^{in}, w)$. A state r is reachable if there is a word that reaches it. We know that the language of $\mathcal{B}$ is non-empty iff there are states $q \in Q^{in}$, $r \in F$ such that there is a path from q to r and an accepting path from r to itself. The initial path is called the prefix, and the combination of the prefix and cycle is called a *lasso*. Lasso finding is accomplished by either explicitly searching the state-space, or by representing sets of states symbolically [18].

A Büchi automaton $\mathcal{A}$ is contained in a Büchi automaton $\mathcal{B}$ iff $L(\mathcal{A}) \subseteq L(\mathcal{B})$, which can be checked by verifying that the intersection of $\mathcal{A}$ with the complement of $\mathcal{B}$ is empty: $L(\mathcal{A}) \cap L(\overline{\mathcal{B}}) = \emptyset$. The intersection of two automata can be constructed, having a number of states proportional to the product of the number states of the original automata [4]. The most computationally demanding step is constructing the complement of $\mathcal{B}$.

2.1 Ramsey-Based Universality

When Büchi introduced these automata in 1962, he described a complementation construction involving a Ramsey-based combinatorial argument and a doubly-

exponential blow-up in the state space [3]. In 1987 an improved implementation of Büchi's construction with only $2^{O(n^2)}$ states was presented [15]. We present this improved implementation.

To construct the complement of $\mathcal{B}$, where $Q = \{q_0, ..., q_{n-1}\}$, we construct a set $\widetilde{Q}_{\mathcal{B}}$ whose elements capture the essential behavior of $\mathcal{B}$. Each element corresponds to an answer to the following question. Given a finite nonempty word w, for every two states $q, r \in Q$: is there a path in $\mathcal{B}$ from q to r over w, and is some such path accepting?

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

Definition 2.1.1. *Let $\widetilde{g} \in \widetilde{Q}_{\mathcal{B}}$ and $w \in \Sigma^+$. Then $w \in L(\widetilde{g})$ iff, for all pairs of states $q, r \in Q$:*

(1) $\langle q, a, r\rangle \in \widetilde{g}$, $a \in \{0,1\}$, iff there is a path in $\mathcal{B}$ from q to r over w.

(2) $\langle q, 1, r\rangle \in \widetilde{g}$ iff there is an accepting path in $\mathcal{B}$ from q to r over w.

The following lemma describes how the graphs $\widetilde{g}$ capture the behavior of $\mathcal{B}$.

Lemma 2.1.2. [3, 15]

(1) $L(\widetilde{g})$, $\widetilde{g} \in \widetilde{Q}_{\mathcal{B}}$, form a partition of Σ^+

(2) If $u \in L(\widetilde{g})$, $v \in L(\widetilde{h})$, and $uv \in L(\widetilde{k})$, then $L(\widetilde{g}) \cdot L(\widetilde{h}) \subseteq L(\widetilde{k})$

Given this partition of Σ^+, we can devise a finite family of ω-languages that cover Σ^ω. For every $\widetilde{g}$, $\widetilde{h} \in \widetilde{Q}_{\mathcal{B}}$, let Y_{gh} be the ω-language $L(\widetilde{g}) \cdot L(\widetilde{h})^\omega$. Y_{gh} is *proper* if Y_{gh} is non-empty, $L(\widetilde{g}) \cdot L(\widetilde{h}) \subseteq L(\widetilde{g})$, and $L(\widetilde{h}) \cdot L(\widetilde{g}) \subseteq L(\widetilde{h})$. Say that the pair $(\widetilde{g}, \widetilde{h})$ is proper when Y_{gh} is proper. There is a finite, if exponential, number of such languages. A Ramsey argument can show that every infinite string belongs to a language of this form.

Lemma 2.1.3. [3, 15] $\Sigma^\omega = \bigcup\{Y_{gh} \mid Y_{gh} \text{ is proper}\}$

Proof: Consider an infinite word $w = a_0 a_1 ...$ By Lemma 2.1.2, every prefix of the word w is in some graph $\widetilde{g}_i$. Thus w defines a partition of $\mathbb{N}$ into k sets $D_1, ..., D_k$ such that $i \in D_l$ iff $a_0 ... a_{i-1} \in L(\widetilde{g}_l)$. Clearly there is some m such that D_m is infinite.

Similarly, by Lemma 2.1.2 we can use the the word w to define a partition of all unordered pairs of elements in D_m. This partition consists of k sets $C_1, ...C_k$, such that $\{i, j\} \in C_l$ iff $a_i ... a_{j-1} \in L(\widetilde{g}_l)$. Ramsey's Theorem tells us that, given

such a partition, there exists an infinite subset $\{i_1, i_2, ...\}$ of D_m and a C_n such that $\{i_j, i_k\} \in C_n$ for all pairs of distinct elements i_j, i_k.

This implies that the word w can be partitioned into

$$w_1 = a_0...a_{i_1-1}, \quad w_2 = a_{i_1}...a_{i_2-1}, \quad w_3 = a_{i_2}...a_{i_3-1}, \quad ...,$$

where $w_1 \in L(\widetilde{g}_m)$ and $w_i \in L(\widetilde{g}_n)$ for $i > 1$. As $a_0...a_{i_j-1} \in L(\widetilde{g}_m)$ for every i_j, we have that $w_1 w_2 \in L(\widetilde{g}_m)$. As $a_{i_j}...a_{i_k-1} \in L(\widetilde{g}_n)$ for every pair i_j, i_k, we have that $w_2 w_3 \in L(\widetilde{g}_n)$. By Lemma 2.1.2, it follows that $L(\widetilde{g}_m)L(\widetilde{g}_n) \subseteq L(\widetilde{g}_m)$ and $L(\widetilde{g}_n)L(\widetilde{g}_n) \subseteq L(\widetilde{g}_n)$ and $Y_{g_m g_n}$ is proper. □

In the proof above, we first defined a subset of indexes, D_m, and then used Ramsey's Theorem to partition unordered pairs of elements from D_m. The particulars of how the subset D_m is chosen do not affect the Ramsey argument. As we later explore other partitions of indexes, we generalize here the essential logic of Lemma 2.1.3 for later use.

Lemma 2.1.4. *Let $\mathcal{B}$ be a Büchi automaton, $\widetilde{Q}_\mathcal{B}$ the associated set of graphs, $w = a_0 a_1...$ an infinite word, and D an infinite set of indexes. There exists $D' \subseteq D$ and $\widetilde{g}_n \in \widetilde{Q}_\mathcal{B}$ so that $a_i...a_{j-1} \in L(\widetilde{g}_n)$ for every $i, j \in D'$, $i < j$.*

Lemma 2.1.5. [3, 15]

(1) For $\widetilde{g}, \widetilde{h} \in \widetilde{Q}_\mathcal{B}$, either $Y_{gh} \cap L(\mathcal{B}) = \emptyset$ or $Y_{gh} \subseteq L(\mathcal{B})$.

(2) $\overline{L(\mathcal{B})} = \bigcup\{Y_{gh} \mid Y_{gh}$ is proper and $Y_{gh} \cap L(\mathcal{B}) = \emptyset\}$.

To obtain the complementary Büchi automaton $\overline{\mathcal{B}}$, Sistla et al. construct a family of deterministic automata on finite words that accept, for each $\widetilde{g} \in \widetilde{Q}_\mathcal{B}$, $L(\widetilde{g})$. The state space of these automata is $\widetilde{Q}'_\mathcal{B} = \widetilde{Q}_\mathcal{B} \cup \{p_0\}$, the set of graphs with the addition of a start state p_0. $\mathcal{B}_g$ is then $\langle \Sigma, \widetilde{Q}'_\mathcal{B}, \widetilde{\rho}, p_0, \{\widetilde{g}\}\rangle$ where the deterministic transition function $\widetilde{\rho} : \widetilde{Q}'_\mathcal{B} \times \Sigma \to \widetilde{Q}_\mathcal{B}$ is:

$$\begin{aligned}
\widetilde{\rho}(p_0, a) &= \{\langle q, 0, r\rangle \mid r \in \rho(q, a),\ q \in Q,\ r \in Q \setminus F\} \\
&\cup \{\langle q, 1, r\rangle \mid r \in \rho(q, a),\ q \in Q,\ r \in F\} \\
\widetilde{\rho}(\widetilde{g}, a) &= \{\langle q, 0, r\rangle \mid \langle q, 0, s\rangle \in \widetilde{g},\ r \in \rho(s, a) \setminus F\} \\
&\cup \{\langle q, 1, r\rangle \mid \langle q, 0, s\rangle \in \widetilde{g},\ r \in \rho(s, a) \cap F\} \\
&\cup \{\langle q, 1, r\rangle \mid \langle q, 1, s\rangle \in \widetilde{g},\ r \in \rho(s, a)\}
\end{aligned}$$

Lemma 2.1.6. [15] $L(\mathcal{B}_g) = L(\widetilde{g})$

Using the automata $\mathcal{B}_g$, one can construct the complementary automaton $\overline{\mathcal{B}}$. For each proper Y_{gh}, construct from $\mathcal{B}_g$ and $\mathcal{B}_h$ a Büchi automaton $\mathcal{B}_{gh}$ that accepts

Y_{gh}. Now find all Y_{gh} disjoint from $L(\mathcal{B})$ and construct the union automaton of the corresponding $\mathcal{B}_{gh}$. We can then use a lasso-finding algorithm on $\overline{\mathcal{B}}$ to prove the emptiness of $\overline{\mathcal{B}}$, and thus the universality of $\mathcal{B}$.

We can avoid an explicit lasso search by employing the rich structure of the graphs in $\widetilde{Q}_{\mathcal{B}}$. For every two graphs $\widetilde{g}, \widetilde{h} \in \widetilde{Q}_{\mathcal{B}}$, determine if Y_{gh} is proper. If Y_{gh} is proper, test if it is contained in $L(\mathcal{B})$ by looking for a lasso with a prefix in $\widetilde{g}$ and a cycle in $\widetilde{h}$. $\mathcal{B}$ is universal if every proper Y_{gh} is so contained.

Lemma 2.1.7. *Given an Büchi automaton $\mathcal{B}$ and the set of graphs $\widetilde{Q}_{\mathcal{B}}$,*

(1) $\mathcal{B}$ is a universal Büchi automaton iff, for every pair of graphs $\widetilde{g}, \widetilde{h} \in \widetilde{Q}_{\mathcal{B}}$ such that Y_{gh} is proper, $Y_{gh} \subseteq L(\mathcal{B})$.

(2) Let $\widetilde{g}, \widetilde{h} \in \widetilde{Q}_{\mathcal{B}}$ be two graphs where Y_{gh} is proper. $Y_{gh} \subseteq L(\mathcal{B})$ iff there exists $q \in Q^{in},\ r \in Q,\ a \in \{0,1\}$ where $\langle q, a, r\rangle \in \widetilde{g}$ and $\langle r, 1, r\rangle \in \widetilde{h}$.

Lemma 2.1.7 yields a PSPACE algorithm to determine universality [15]. Simply check each $\widetilde{g}, \widetilde{h} \in \widetilde{Q}_{\mathcal{B}}$. If Y_{gh} is both proper and not contained in $L(\mathcal{B})$, then the pair $(\widetilde{g}, \widetilde{h})$ provide a counterexample to the universality of $\mathcal{B}$. If no such pair exists, the automaton must be universal.

2.2 Rank-Based Complementation

If a Büchi automaton $\mathcal{B}$ does not accept a word w, then every run of $\mathcal{B}$ on w must eventually cease visiting accepting states. The rank-based construction uses a notion of ranks to track the progress of each possible run towards fair termination. A *level ranking* for an automaton $\mathcal{B}$ with n states is a function $f : Q \to \{0...2n, \bot\}$, such that if $q \in F$ then $f(q)$ is even or $\bot$. Let a be a letter in Σ and f, f' be two level rankings f. Say that f covers f' under a when for all q and every $q' \in \rho(q, a)$, if $f(q) \neq \bot$ then $f'(q') \leq f(q)$; i.e. no path between f and f' on a increases in rank. Let F_r be the set of all level rankings.

If $\mathcal{B} = \langle \Sigma, Q, Q^{in}, \rho, F\rangle$ is a Büchi automaton, define $KV(\mathcal{B})$ to be the automaton $\langle \Sigma, F_r \times 2^Q, \langle f_{in}, \emptyset\rangle, \rho', F_r \times \{\emptyset\}\rangle$, where

- $f_{in}(q) = 2n$ for each $q \in Q^{in}$, $\bot$ otherwise.
- Define $\rho' : \langle F_r \times 2^Q\rangle \times \sigma \to 2^{\langle F_r \times 2^Q\rangle}$ to be
  - If $o \neq \emptyset$ then $\rho'(\langle rf, o\rangle, \sigma) =$ $\{\langle f', o' \setminus d\rangle \mid f \text{ covers } f' \text{ under } \sigma,\ o' = \rho(o, \sigma),\ d = \{q \mid f'(q) \text{ odd}\}\}$.
  - If $o = \emptyset$ then $\rho'(\langle f, o\rangle, \sigma) =$ $\{\langle f', f' \setminus d\rangle \mid f \text{ covers } f' \text{ under } a,\ d = \{q \mid f'(q) \text{ odd}\}\}$.

Lemma 2.2.1. [11] *For every Büchi automaton $\mathcal{B}$, $L(KV(\mathcal{B})) = \overline{L(\mathcal{B})}$.*

This automaton tracks the progress of $\mathcal{B}$ along a word $w = a_0 a_1 ...$ by attempting to find an infinite series of level rankings $f_0 f_1 ....$ We start with the most general possible level ranking, and ensure that every rank f_i covers f_{i+1} under a_i. Every run has a non-increasing rank, and so must eventually become trapped in some rank. To accept a word, we require require that each run visit an odd rank infinitely often. For a word rejected by $\mathcal{B}$, every run can eventually become trapped in an odd rank. Conversely, if there is an accepting run that visits an accepting node infinitely often, that run cannot visit an odd rank infinitely often and the complementary automaton rejects it. The subset construction is insufficient for complementing Büchi automata. Where the Ramsey-based construction can be seen as using n parallel subset constructions, the rank-based construction associates a monotonically decreasing rank with each element of the subset. An algorithm seeking to refute the containment of $\mathcal{A}$ in $\mathcal{B}$ can look for a lasso in the state-space of the intersection automaton $\mathcal{A} \cap KV(\mathcal{B})$.

The strongest algorithm performing this search takes advantage of the presence of a subsumption relation in the KV construction: one state $\langle f, s \rangle$ subsumes another $\langle f', s' \rangle$ if $f(x) \leq f'(x)$ for every $x \in Q$ and $s' \subseteq s$. When computing the backward-traversal lasso-finding fixed point, it is sufficient to represent a set of states with the maximal elements under this relation. Further, the predecessor operation over a single state and letter results in at most two incomparable elements. This algorithm has scaled to automata an order of magnitude larger than other approaches [6].

2.3 Size-Change Termination

In [12] Lee et al. proposed the size-change termination (SCT) principle: "If every infinite computation would give rise to an infinitely decreasing value sequence, then no infinite computation is possible." The original presentation concerned a first-order purely functional language, where every infinite computation arises from an infinite call sequence and values are always passed through a sequence of parameters. We here review the relevant details of the SCT principle and its algorithmic realization.

Proving that a program is size-change terminating is done in two phases. The first extracts from a program a set of size-change graphs containing guarantees about the size of values at each function call site. The second phase, and the phase we focus on, analyzes these graphs to determine if every infinite call sequence has a value that descends infinitely along a well-ordered set. For a discussion of the abstraction of language semantics, refer to the original paper [12]. We consider here a set H of functions and a set C of call sites, written $c : f_1 \to f_2$ for a call to function f_2 occurring in the body of f_1. Denote the parameters of a function f as $P(f)$.

Definition 2.3.1. *A* size-change graph *(SCG) from f_1 to f_2, written $G : f_1 \to f_2$, is a bipartite $\{0, 1\}$-arc-labeled graph from the parameters of f_1 to the parameters of f_2, where $G \subseteq P(f_1) \times \{0, 1\} \times P(f_2)$ does not contain both $x \xrightarrow{1} y$ and $x \xrightarrow{0} y$.*

Size-change graphs capture information about a function call. An arc $x \xrightarrow{1} y$

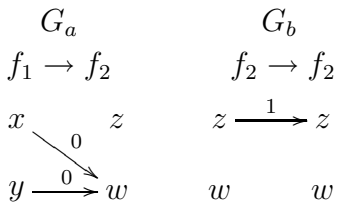


Figure 2.1: A size-change terminating SCT problem.

indicates that the value of x in the function f_1 is strictly greater than the value passed as y to function f_2. An arc $x \xrightarrow{0} y$ indicates that x's value is greater than or equal to the value given to y. We assume that all call sites in C are reachable from the entry points of the program[1]. The first phase of proving size-change termination abstracts from the program a size-change graph for each call site, providing an initial set of size-change graphs $\mathcal{G}$.

Definition 2.3.2. *A size-change termination (SCT) problem is a tuple $L = \langle H, P, C, \mathcal{G} \rangle$ where H is a set of functions, P a mapping from each function to its parameters, C a set of call sites between these functions, and $\mathcal{G}$ a set of size-change graphs for C.*

Definition 2.3.3.

(1) A call sequence *in L is a infinite sequence $cs = c_0, c_1, \ldots \in C^\omega$, such that there exists a sequence of functions $f_0, f_1, ..,$ where $c_0 : f_0 \to f_1$, $c_1 : f_1 \to f_2$....*

(2) A thread *in a call sequence $c_0, c_0, \ldots$ is a connected path of arcs beginning at a call c_i, through the corresponding sequence of size-change graphs $G_i, G_{i+1} \ldots$ such that $x \xrightarrow{a} y \in G_i, y \xrightarrow{b} z \in G_{i+1}, \ldots,$*

(3) A SCT problem is size-change terminating *if every call sequence contains a thread with infinitely many 1-labeled arcs.*

Note that a thread need not begin at the start of a call sequence. Consider the example in Figure 2.1. This example has two functions, f_1 and f_2, two calls, $a : f_2 \to f_2$ and $b : f_2 \to f_2$, and the corresponding size-change graphs G_a and G_b. This problem is size-change terminating. There are two possibly infinite call sequences, b^ω and ab^ω. In the first call sequence the 1-labeled arc from z to itself forms an infinite thread from the beginning. In the call sequence starting with a, no source parameter in G_a has a path to the accepting cycle in G_b. None the less, the thread that begins with z in the second graph and continues along the 1-labeled

[1]The reference implementation provided by Lee et al. [12] also make this assumption, and does not explicitly check that a prefix exists for the call sequence. Thus in the presence of unreachable functions, these tools may not be able to determine size-change termination.

arc ensures that non-termination can only occur if a well-founded value decreases infinitely often. This holds in all cases: threads can begin at any function call, in any parameter. We call this the *late-start property* of SCT problems, and revisit it in Sections 3.2 and 3.5.

If we associate a letter with each call site, every call sequence can be represented as a word in C^ω, and a size-change termination problem reduced to the containment of two ω-languages. The first language, $Flow(L)$, contains all call sequences, while the second language $Desc(L)$ contains all call sequences containing a thread with infinitely many 1-labeled arcs. If $Flow(L) \subseteq Desc(L)$, then every infinite call sequence has a value that descends infinitely.

Definition 2.3.4.

(1) $Flow(L) = \{cs \in C^\omega \mid cs \text{ is a call sequence}\}$.

(2) $Desc(L) = \{cs \in Flow(L) \mid \text{some thread in } cs \text{ has infinitely many 1-labeled arcs}\}$

Lemma 2.3.5. [12] *A size-change problem L is size-change terminating if and only if $Flow(L) \subseteq Desc(L)$.*

Lee et al. [12] describe two Büchi automata, $\mathcal{A}_{Flow(L)}$ and $\mathcal{A}_{Desc(L)}$, that accept these languages. $\mathcal{A}_{Flow(L)}$ is simply the call graph of p, and has one state for each function. $\mathcal{A}_{Desc(L)}$ waits in a copy of the call graph and nondeterministically chooses the beginning point of a descending thread. From there it ensures that a 1-labeled arc is taken an infinite number of times, using two states for each parameter of each function.

Definition 2.3.6. [2]

$$
\begin{array}{lll}
\mathcal{A}_{Flow(L)} & = & \langle C, H, H, \rho_F, H\rangle \\
 & \text{where} & \rho_F(f_1, c) = \{f_2 \mid c : f_1 \to f_2\} \\
\mathcal{A}_{Desc(L)} & = & \langle C, Q_1 \cup H, H, \rho_D, F\rangle \\
 & \text{where} & Q_1 = \bigcup_{f\in H} P(f) \times \{1, 0\} \\
 & & \rho_D(f_1, c) = \{f_2 \mid c : f_1 \to f_2\} \cup \{\langle x, i\rangle \mid c : f_1 \to f_2,\ x \in P(f_2),\ i \in \{0, 1\}\} \\
 & & \rho_D(\langle x, r\rangle, c) = \{\langle x', r'\rangle \mid x \xrightarrow{r'} x' \in \mathcal{G}_c\} \\
 & & F = \bigcup_{f\in H} P(f) \times \{1\}
\end{array}
$$

Lemma 2.3.7. $L(\mathcal{A}_{Flow(L)}) = Flow(L)$, *and* $L(\mathcal{A}_{Desc(L)}) = Desc(L)$

[2] *The original LJB construction [12] restricted edges from functions to parameters to the 0-labeled parameters. This was changed to simplify Section 3.6. As outgoing transitions from 0 and 1 labeled parameters are the same, the modification does not change the accepted language.*

Using the complementation constructions of either Section 2.1 or 2.2 and a lasso-finding algorithm, we can determine the containment of $\mathcal{A}_{Flow(L)}$ in $\mathcal{A}_{Desc(L)}$. Lee et al. propose an alternative graph-theoretic algorithm, employing size-change graphs to encode descent information about call sequences. A notion of composition is provided, where the composition of a sequence of graphs describes the relationship of values over that call sequence. The closure of $\mathcal{G}$ under the composition operation is then searched for a counterexample describing an infinite call sequence with no path of infinite descent.

Definition 2.3.8. *Let $G : f_1 \to f_2$ and $G' : f_2 \to f_3$ be two size-change graphs. Their composition $G; G'$ is defined as $G'' : f_1 \to f_3$ where:*

$$\begin{aligned} G'' &= \{x \xrightarrow{1} z \mid x \xrightarrow{a} y \in G,\ y \xrightarrow{b} z \in G',\ y \in P(f_2),\ a = 1 \text{ or } b = 1\} \\ &\cup \{x \xrightarrow{0} z \mid x \xrightarrow{0} y \in G,\ y \xrightarrow{0} z \in G',\ y \in P(f_2),\ \text{and} \\ &\qquad \forall y', r, r' \ .\ x \xrightarrow{r} y' \in G \wedge y' \xrightarrow{r'} z \in G' \text{ implies } r = r' = 0\} \end{aligned}$$

Lemma 2.3.9. *Graph composition is associative.*

Lemma 2.3.10. *A call sequence $a_0 ... a_{n-n}$ has a thread from x to y over its entire length, containing at least one 1-labeled arc, if and only if $x \xrightarrow{1} y \in G_0; ... G_{n-1}$.*

Call a size-change graph graph $G : f \to f$ *idempotent* if $G = G; G$, and a *counterexample graph* if it is idempotent and does *not* contain an arc of the form $x \xrightarrow{1} x$.

Theorem 2.3.11. [12] *A size-change termination (SCT) problem $L = \langle H, P, C, \mathcal{G} \rangle$ is* not *size-change terminating iff the closure of $\mathcal{G}$ under composition contains a counterexample graph.*

Theorem 2.3.11 yields an algorithm that determines the size-change termination of an SCT problem $L = \langle H, P, C, \mathcal{G} \rangle$ by ensuring the absence of a counterexample in the closure of $\mathcal{G}$ under composition. First, use an iterative algorithm to build the closure set S: initialize S as $\mathcal{G}$, and for every $G : f_1 \to f_2$ and $G' : f_2 \to f_3$ in S, include $G; G'$ in S. Second, check every $G : f_1 \to f_1 \in S$ to ensure that if $G = G; G$, then G is not a counterexample graph.

Chapter 3
Size-Change Termination and Ramsey-Based Containment

The Ramsey-based test of Section 2.1 and the LJB algorithm of Section 2.3 bear a more than passing similarity. In this chapter we bridge the gap between the Ramsey-based universality test and the LJB algorithm. In the context of size-change termination, Lee et al. provide a strategy that trades space for time. We now apply this strategy to the universality test of Lemma 2.1.7 and present a constructive algorithm that builds on Lemma 2.1.7. We then examine a language property that allow us to further simplify universality-checking algorithms. Finally, we lift the universality-testing algorithm to containment. This allows us to prove that the LJB algorithm of Section 2.3 is a specialized realization of the Ramsey-based universality test.

3.1 Composition: Trading Space for Time

The Ramsey-based PSPACE algorithm we obtain from Lemma 2.1.7 checks the universality of a Büchi automaton, $\mathcal{B} = \langle \Sigma, Q, Q^{in}, \rho, F \rangle$. To ensure $\mathcal{B}$ is universal, search every two graphs $\widetilde{g}, \widetilde{h} \in \widetilde{Q}_{\mathcal{B}}$ for cases where Y_{gh} is proper. Observe, however, that if either $L(\widetilde{g})$ or $L(\widetilde{h})$ is empty then Y_{gh} is trivially not proper. Thus, this algorithm may needlessly consider many graphs with empty languages. Further, it is not immediately obvious how to determine if the language of a graph is empty. We extend the notion of composition from Section 2.3 to graphs in $\widetilde{Q}_{\mathcal{B}}$, allowing us to use exponential space to compute exactly the graphs with nonempty languages. This simplifies the test for properness. Naively checking the emptiness or containment of languages is a PSPACE-hard problem. Using composition we provide polynomial tests for all aspects of properness.

As defined in Section 2.1, $\widetilde{Q}_{\mathcal{B}} = 2^{Q \times \{0,1\} \times Q}$. Elements $\widetilde{g} \in \widetilde{Q}_{\mathcal{B}}$ are $\{0,1\}$-arc-labeled graphs over the states of $\mathcal{B}$ where each arc $\langle q, a, r \rangle \in \widetilde{g}$ represents a path in $\mathcal{B}$ from q to r, with $a = 1$ iff some such path is accepting. Each graph $\widetilde{g}$ has an associated language $L(\widetilde{g})$ of words with exactly these paths through $\mathcal{B}$.

Given two graphs $\widetilde{g}, \widetilde{h} \in \widetilde{Q}_{\mathcal{B}}$, define their composition $\widetilde{g}; \widetilde{h}$ to be the graph:

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

Lemma 3.1.1. *For every two graphs $\widetilde{g}$ and $\widetilde{h}$, $L(\widetilde{g}) \cdot L(\widetilde{h}) \subseteq L(\widetilde{g};\widetilde{h})$.*

Proof: Consider two words $w_1 \in L(\widetilde{g})$, $w_2 \in L(\widetilde{h})$. By Definition 2.1.1, to prove $w_1 w_2 \in L(\widetilde{g};\widetilde{h})$ we must show that for every $q, r \in Q$: (1) $\langle q, a, r\rangle \in \widetilde{g};\widetilde{h}$ iff there is a path from q to r over $w_1 w_2$, and (2) that $a = 1$ iff there is an accepting path.

If an arc $\langle q, a, r\rangle \in \widetilde{g};\widetilde{h}$ exists, then there is an $s \in Q$ such that $\langle q, b, s\rangle \in \widetilde{g}$ and $\langle s, c, r\rangle \in \widetilde{h}$. By Definition 2.1.1, this implies the existence of a path $x_1 s$ from q to s over w_1, and a path $s x_2$ from s to r over w_2. Thus $x_1 s x_2$ is a path from q to r over $w_1 w_2$.

If a is 1, then either b or c must be 1. By Definition 2.1.1, b (resp., c) is 1 iff there is an accepting path $x_1' s$ (resp., $s x_2'$) over w_1 (resp.,w_2) from q to s (resp., s to r). In this case $x_1' s x_2$ (resp., $x_1 s x_2'$) is an accepting path in $\mathcal{B}$ from q to r over $w_1 w_2$.

Symmetrically, if there is a path x from q to r over $w_1 w_2$, then after reading w_1 we are in some state s and have split x into $x_1 s x_2$, so that $x_1 s$ is a path from q to s and $s x_2$ a path from s to r. Thus by Definition 2.1.1 $\langle q, b, s\rangle \in \widetilde{g}$, $\langle s, c, r\rangle \in \widetilde{h}$, and $\langle q, a, s\rangle \in \widetilde{g};\widetilde{h}$.

Further, if there is an accepting path from q to r over $w_1 w_2$, then after reading w_1 we are in some state s and have split the path into $x_1 s x_2$, so that $x_1 s$ is a path from q to s, and $s x_2$ a path from s to r. Either $x_1 s$ or $s x_2$ must be accepting, and thus by Definition 2.1.1 $\langle q, b, s\rangle \in \widetilde{g}$, $\langle s, c, r\rangle \in \widetilde{h}$, and either b or c must be 1. Therefore a must be 1. □

For a Büchi automaton $\mathcal{B}$, define the set of graphs corresponding to single letters to be $\widetilde{Q}^1_{\mathcal{B}} = \{\widetilde{g} \mid \widetilde{g} \in \widetilde{Q}_{\mathcal{B}},\ a \in \Sigma,\ a \in L(\widetilde{g})\}$. Let $\widetilde{Q}^f_{\mathcal{B}}$ be the closure of $\widetilde{Q}^1_{\mathcal{B}}$ under composition.

Lemma 3.1.2. *For a Büchi automaton $\mathcal{B}$, every $\widetilde{g} \in \widetilde{Q}_{\mathcal{B}}$ such that $L(\widetilde{g}) \neq \emptyset$ is in $\widetilde{Q}^f_{\mathcal{B}}$.*

Proof: Take an $\widetilde{g}$ where $L(\widetilde{g}) \neq \emptyset$. This implies there is at least one word $w = a_0 ... a_{n-1} \in L(\widetilde{g})$. Let $\widetilde{g}_{a_i}$ be the graph in $\widetilde{Q}^1_{\mathcal{B}}$ corresponding to a_i, and let $\widetilde{g}_w$ be $\widetilde{g}_{a_0};\widetilde{g}_{a_0};...;\widetilde{g}_{a_{n-1}}$.

By Lemma 3.1.1, $w \in \widetilde{g}_w$. By Lemma 2.1.2, every word w is in the language of exactly one graph. Therefore $\widetilde{g} = \widetilde{g}_w$, and $\widetilde{g}$ is in $\widetilde{Q}^f_{\mathcal{B}}$. □

We can now generate exactly the set of graphs with non-empty languages. From the transition function ρ, construct the set of graphs $\widetilde{Q}^1_{\mathcal{B}}$ corresponding to single characters. Using composition, iteratively compute graphs for larger words until we reach the fixed point $\widetilde{Q}^f_{\mathcal{B}}$.

The remaining test for properness required for Lemma 2.1.7 involve the containment of languages. Given a graph $\widetilde{g}$, we can express the condition $L(\widetilde{g}) \cdot L(\widetilde{g}) \subseteq L(\widetilde{g})$ in terms of composition.[1] We strengthen Lemma 3.1.1 and prove composition is

[1] The essence of this test was described in [16].

equivalent to language concatenation for arbitrary graphs $\widetilde{g}, \widetilde{h}, \widetilde{k} \in \widetilde{Q}^f_{\mathcal{B}}$. This allows concatenation and language containment to be tested using composition and equality.

Lemma 3.1.3. *Let $\widetilde{g}, \widetilde{h}, \widetilde{k}$ be graphs in $\widetilde{Q}^f_{\mathcal{B}}$. $L(\widetilde{g}) \cdot L(\widetilde{h}) \subseteq L(\widetilde{k})$ iff $\widetilde{g}; \widetilde{h} = \widetilde{k}$*

Proof: Assume $\widetilde{g}; \widetilde{h} = \widetilde{k}$. Lemma 3.1.1 implies $L(\widetilde{g}) \cdot L(\widetilde{h}) \subseteq L(\widetilde{g}; \widetilde{h}) = L(\widetilde{k})$.

In the other direction, if $L(\widetilde{g}) \cdot L(\widetilde{h}) \subseteq L(\widetilde{k})$, we show that $\widetilde{g}; \widetilde{h} = \widetilde{k}$. As $\widetilde{g}, \widetilde{h} \in \widetilde{Q}^f_{\mathcal{B}}$ and are non-empty, there is a word $w \in L(\widetilde{g}) \cdot L(\widetilde{h})$. By assumption, then, $w \in L(\widetilde{k})$. By Lemma 3.1.1, $w \in L(\widetilde{g}; \widetilde{h})$. Finally, by Lemma 2.1.2, w is the language of exactly one graph, so $\widetilde{g}; \widetilde{h} = \widetilde{k}$. □

3.2 Strongly Suffix Closed Languages

Theorem 2.3.11 suggests that, for some languages, a cycle implies the existence of a lasso. For Büchi automata with these languages, it is sufficient, when disproving universality, to search for a graph $\widetilde{h} \in \widetilde{Q}_{\mathcal{B}}$, where $\widetilde{h}; \widetilde{h} = \widetilde{h}$, with no arc $\langle r, 1, r \rangle$. Call a graph $\widetilde{h}$ *idempotent* when $\widetilde{h}; \widetilde{h} = \widetilde{h}$. This single-graph search reduces the complexity of our algorithm significantly. What enables this in size-change termination is the late-start property: threads can begin at any point. We define this in terms of automata universality and define the class of automata amenable to optimization.

In size-change termination, an accepting cycle can start at any point. Thus the arc $\langle r, 1, r \rangle \in \widetilde{h}$ does not need an explicit matching prefix $\langle q, a, r \rangle$ in some $\widetilde{g}$. In the context of universality, we can apply this method when it is safe to add or remove arbitrary prefixes of a word. To describe these languages we extend the standard notion of *suffix closure*. A language L is suffix closed when, for every $w \in L$, every suffix of w is in L.

Definition 3.2.1. *A language L is* strongly suffix closed *if it is suffix closed and for every $w \in L$, $w_1 \in \Sigma^+$, we have that $w_1 w \in L$.*

Lemma 3.2.2. *Let $\mathcal{B}$ be an Büchi automaton where every state in Q is reachable and $L(\mathcal{B})$ is strongly suffix closed. $\mathcal{B}$ is* not *universal iff $\widetilde{Q}^f_{\mathcal{B}}$ contains an idempotent graph $\widetilde{h}$ with no arc of the form $\langle r, 1, r \rangle$.*

Proof: Assume that there is an idempotent graph $\widetilde{h} \in \widetilde{Q}^f_{\mathcal{B}}$ with no arc of the form $\langle r, 1, r \rangle$. We show that Y_{hh} is a proper language not contained in $L(\mathcal{B})$, and therefore that $\mathcal{B}$ is not universal. As $\widetilde{h} \in \widetilde{Q}^f_{\mathcal{B}}$, we know $L(\widetilde{h})$ is not empty, implying Y_{hh} is non-empty. Along with the premise that $\widetilde{h}$ is idempotent, i.e. $\widetilde{h}; \widetilde{h} = \widetilde{h}$, and Lemma 3.1.3, this shows that Y_{hh} is proper. Finally, if there is no $\langle r, 1, r \rangle \in \widetilde{h}$, then there certainly cannot be q, r so that $\langle q, a, r \rangle \in \widetilde{h}$ and $\langle r, 1, r \rangle \in \widetilde{h}$. By Lemma 2.1.7, Y_{hh} is a counterexample to $\mathcal{B}$'s universality.

Conversely, assume that every idempotent graph $\widetilde{h} \in \widetilde{Q}^f_{\mathcal{B}}$ has a arc $\langle r, 1, r \rangle$. We prove that every word $w \in \Sigma^\omega$ is in $L(\mathcal{B})$. Take a word w. By Lemma 2.1.3, w is in

some proper language Y_{gh}. Thus $w = w_1 w_2$, where $w_1 \in L(\widetilde{g})$ and $w_2 \in L(\widetilde{h})^\omega$. Since Y_{gh} is proper, $\widetilde{h};\widetilde{h} = \widetilde{h}$. By assumption, $\widetilde{h}$ contains an arc $\langle r, 1, r\rangle$, and thus there is an accepting run of $\mathcal{B}$ on w_2 beginning at r. As all states are reachable, there is $q \in Q^{in}$ and $u \in \Sigma^+$ so that there is a path from q to r over u. By Lemma 2.1.7, this implies uw_2 is accepted by $\mathcal{B}$. As $L(\mathcal{B})$ is strongly suffix closed, we can conclude $w_2 \in L(\mathcal{B})$, which implies $w_1 w_2 \in L(\mathcal{B})$. □

3.3 Ramsey-Based Containment with Supergraphs

To test the containment of a Büchi automaton $\mathcal{A}$ in a Büchi automaton $\mathcal{B}$, we could construct the complement of $\mathcal{B}$ using either the Ramsey-based or rank-based construction, compute the intersection automaton of $\mathcal{A}$ and $\overline{\mathcal{B}}$, and search this intersection automaton for a lasso. With universality, however, we avoided directly constructing $\overline{\mathcal{B}}$ by exploiting the structure of states in the Ramsey-based construction (see Lemma 2.1.7 and Section 3.1). We demonstrate a similar test for containment.

Consider two automata, $\mathcal{A} = \langle \Sigma, Q_\mathcal{A}, Q^{in}_\mathcal{A}, \rho_\mathcal{A}, F_\mathcal{A}\rangle$ and $\mathcal{B} = \langle \Sigma, Q_\mathcal{B}, Q^{in}_\mathcal{B}, \rho_\mathcal{B}, F_\mathcal{B}\rangle$. When testing the universality of $\mathcal{B}$, any word not in $L(\mathcal{B})$ is a sufficient counterexample. To test $L(\mathcal{A}) \subseteq L(\mathcal{B})$ we must restrict our search to the subset of Σ^ω accepted by $\mathcal{A}$. In Section 2.1, we defined a set $\widetilde{Q}_\mathcal{B}$ whose elements describe all paths in $\mathcal{B}$ over a word. We now define a set $\widehat{Q}_{\mathcal{A},\mathcal{B}}$, each element of which simultaneously captures all paths in $\mathcal{B}$ and a single path in $\mathcal{A}$. Whereas $\widetilde{Q}_\mathcal{B}$ provides a family of languages that covers Σ^ω (see Lemma 2.1.3), $\widehat{Q}_{\mathcal{A},\mathcal{B}}$ defines a family of languages covering $L(\mathcal{A})$.

We first define $\bar{Q}_\mathcal{A} = Q_\mathcal{A} \times Q_\mathcal{A}$ to capture the connectivity in $Q_\mathcal{A}$. An element $\langle q, r\rangle \in \bar{Q}_\mathcal{A}$ is a single arc asserting the existence of a path in $\mathcal{A}$ from q to r. Call q the source, and r the sink. With each arc $\bar{g}$ we associate a language $L(\bar{g})$, the set of words that connect the source and sink. Let $\langle q, r\rangle \in \bar{Q}_\mathcal{A}$ and $w \in \Sigma^+$.

Definition 3.3.1. *$w \in L(\langle q, r\rangle)$ iff there is a path in $\mathcal{A}$ from q to r over w.*

Define $\widehat{Q}_{\mathcal{A},\mathcal{B}}$ as $\bar{Q}_\mathcal{A} \times \widetilde{Q}_\mathcal{B}$. The elements of $\widehat{Q}_{\mathcal{A},\mathcal{B}}$, called *supergraphs*, are pairs consisting of an arc from $\bar{Q}_\mathcal{A}$ and a graph from $\widetilde{Q}_\mathcal{B}$. The language $L(\langle \bar{g}, \widetilde{g}\rangle)$ is then $L(\bar{g}) \cap L(\widetilde{g})$, or the set of words for which $\mathcal{A}$ is correctly described by $\bar{g}$ and $\mathcal{B}$ is correctly described by $\widetilde{g}$. Note that the languages $L(\widehat{g})$, $\widehat{g} \in \widehat{Q}_{\mathcal{A},\mathcal{B}}$ do not form a partition of finite substrings of $L(\mathcal{A})$: a single word may be in the language of multiple supergraphs. For convenience, we implicitly take $\widehat{g} = \langle \bar{g}, \widetilde{g}\rangle$, and say $\langle q, a, r\rangle \in \widehat{g}$ when $\langle q, a, r\rangle \in \widetilde{g}$.

Lemma 3.3.2. *If $u \in L(\widehat{g})$, $v \in L(\widehat{h})$, $uv \in L(\widehat{k})$, and $L(\bar{g}) \cdot L(\bar{h}) \subseteq L(\bar{k})$, then $L(\widehat{g}) \cdot L(\widehat{h}) \subseteq L(\widehat{k})$*

Proof: Assume we have such an u and v. Consider a word $w \in (L(\bar{g}) \cap L(\widetilde{g})) \cdot (L(\bar{h}) \cap L(\widetilde{h}))$. We show $w \in L(\widehat{k})$. First, w must be in $L(\bar{g}) \cdot L(\bar{h})$ and $L(\widetilde{g}) \cdot L(\widetilde{h})$. Second, we know that $u \in L(\widetilde{g})$, $v \in L(\widetilde{h})$, and $uv \in L(\widetilde{k})$. Thus by Lemma 2.1.2,

$L(\widetilde{g}) \cdot L(\widetilde{h}) \subseteq L(\widetilde{k})$ and $w \in L(\widetilde{k})$. Along with the premise $L(\bar{g}) \cdot L(\bar{h}) \subseteq L(\bar{k})$, we can now conclude $w \in L(\bar{k}) \cap L(\widetilde{k})$, and therefore $(L(\bar{g}) \cap L(\widetilde{g})) \cdot (L(\bar{h}) \cap L(\widetilde{h})) \subseteq (L(\bar{k}) \cap L(\widetilde{k}))$. □

The languages $L(\widehat{g})$, $\widehat{g} \in \widehat{Q}_{\mathcal{A},\mathcal{B}}$, cover all finite substrings of $L(\mathcal{A})$. With them we define a finite family of ω-languages that cover $L(\mathcal{A})$. Given $\widehat{g}$, $\widehat{h} \in \widehat{Q}_{\mathcal{A},\mathcal{B}}$, let Z_{gh} be the ω-language $L(\widehat{g}) \cdot L(\widehat{h})^{\omega}$. Let Z_{gh} be called *proper* if: (1) Z_{gh} is non-empty; (2) $\bar{g} = \langle q, r\rangle$ and $\bar{h} = \langle r, r\rangle$ where $q \in Q^{in}_{\mathcal{A}}$ and $r \in F_{\mathcal{A}}$; (3) $L(\widehat{g}) \cdot L(\widehat{h}) \subseteq L(\widehat{g})$ and $L(\widehat{h}) \cdot L(\widehat{h}) \subseteq L(\widehat{h})$. Call the pair $(\widehat{g}, \widehat{h})$ proper when Z_{gh} is proper. We note that Z_{gh} is non-empty if $L(\widehat{g})$ and $L(\widehat{h})$ are non-empty, and that, by the second condition, every proper Z_{gh} is contained in $L(\mathcal{A})$.

Lemma 3.3.3.

(1) $L(\mathcal{A}) = \bigcup\{Z_{gh} \mid Z_{gh}$ *is proper*$\}$

(2) For all proper Z_{gh}*, either* $Z_{gh} \cap L(\mathcal{B}) = \emptyset$ *or* $Z_{gh} \subseteq L(\mathcal{B})$

(3) $L(\mathcal{A}) \cap \overline{L(\mathcal{B})} = \bigcup\{Z_{gh} \mid Z_{gh}$ *is proper and* $Z_{gh} \cap L(\mathcal{B}) = \emptyset\}$.

Proof:
(1): We extend the Ramsey argument of Lemma 2.1.3 to supergraphs.

Consider an infinite word $w = a_0a_1...$ with an accepting run $p = p_0p_1...$ in $\mathcal{A}$. As p is accepting, we know that $p_0 \in Q^{in}_{\mathcal{A}}$ and $p_i \in F_{\mathcal{A}}$ for infinitely many i. Since $F_{\mathcal{A}}$ is finite, at least one accepting state q must appear infinitely often. Let $C \subseteq \mathbb{N}$ be the set of indexes i such that $p_i = q$.

We pause to observe that, by Definition 3.3.1, for every $i \in C$ the word $a_0...a_{i-1}$ is in $L(\langle p_0, q\rangle)$, and for every $i, j \in C$, $i < j$, the word $a_i...a_{j-1} \in L(\langle q, q\rangle)$. Every language Z_{hk} where $\bar{h} = \langle p_0, q\rangle$ and $\bar{h} = \langle q, q\rangle$ thus satisfies the second requirement of properness.

In addition to restricting our attention the subset of vertices where $p_i = q$, we further partition C into k sets $D_1, ..., D_k$ based on the prefix of w until that point. By Lemma 2.1.2, every finite word is in the language of some $\widetilde{g}$. Say that $i \in D_l$ iff $a_0...a_{i-1} \in L(\widetilde{g}_l)$. As k is finite, for some m D_m must be infinite. Let $\widetilde{h} = \widetilde{g}_m$.

By Lemma 2.1.4, there is a subset $D' \subseteq D_m$ and a graph $\widetilde{k}$ so that, for every $i_j, i_k \in D'$, $a_{i_j}...a_{i_k-1} \in L(\widetilde{k})$. D' thus partitions the word w into

$$w_1 = a_0...a_{i_1-1}, \quad w_2 = a_{i_1}...a_{i_2-1}, \quad w_3 = a_{i_2}...a_{i_3-1}, \quad ...,$$

such that $w_1 \in L(\widetilde{h})$ and $w_i \in L(\widetilde{k})$ for $i > 1$. Let $\widehat{h} = \langle\langle p_0, q\rangle, \widetilde{h}\rangle$ and let $\widehat{k} = \langle\langle q, q\rangle, \widetilde{k}\rangle$. By the above partition of w, we know that $w \in Z_{hk}$. We now show that Z_{hk} is proper.

First, as $a_0...a_{i-1} \in L(\widetilde{h})$ for every $i \in D'$, we have that $w_1w_2 \in L(\widehat{h})$. As mentioned above, w_1 and w_1w_2 are in both $L(\langle p_0, q\rangle)$ and so w_1, $w_1w_2 \in L(\widehat{h})$. By Definition 3.3.1, $L(\langle p_0, q\rangle) \cdot L(\langle q, q\rangle) \subseteq L(\langle p_0, q\rangle)$. Thus by Lemma 3.3.2, can conclude

that $L(\widehat{h}) \cdot L(\widehat{k}) \subseteq L(\widehat{h})$. Secondly, $a_i ... a_{j-1} \in L(\widetilde{k})$ for every pair $i, j \in D'$, we have that $w_2 w_3 \in L(\widetilde{k})$. As w_2, $w_2 w_3$ are both in $\langle q, q \rangle$, it holds that w_2, $w_2 w_3 \in L(\widehat{k})$. By the definition of a path, $L(\langle q, q \rangle) \cdot L(\langle q, q \rangle) \subseteq L(\langle q, q \rangle)$. By Lemma 3.3.2 we can now conclude $L(\widehat{k}) \cdot L(\widehat{k}) \subseteq L(\widehat{k})$. Therefore Z_{hk} is proper and contains w.

(2): Consider two supergraphs $\widehat{g}, \widehat{h}$. Recall that Y_{gh} is the language $L(\widetilde{g}) \cdot L(\widetilde{h})^\omega$. Note that $L(\widehat{g}) \subseteq L(\widetilde{g})$ and $L(\widehat{h}) \subseteq L(\widetilde{h})$, and therefore $Z_{gh} \subseteq Y_{gh}$. By Lemma 2.1.5 either $Y_{gh} \cap L(\mathcal{B}) = \emptyset$ or $Y_{gh} \subseteq L(\mathcal{B})$. Therefore $Z_{gh} \cap L(\mathcal{B}) = \emptyset$ or $Z_{gh} \subseteq L(\mathcal{B})$.

(3): Immediate from (1) and (2). □

Lemma 3.3.4. *Let $\mathcal{A}$ and $\mathcal{B}$ be two Büchi automata, and $\widehat{Q}_{\mathcal{A},\mathcal{B}}$ be the corresponding set of supergraphs.*

(1) $L(\mathcal{A}) \subseteq L(\mathcal{B})$ iff, for every pair of supergraphs $\widehat{g}, \widehat{h}$ such that Z_{gh} is proper, $Z_{gh} \subseteq L(\mathcal{B})$.

(2) Let $\widehat{g}, \widehat{h}$ be two supergraphs such that Z_{gh} is proper. $Z_{gh} \subseteq L(\mathcal{B})$ iff there exists $q \in Q^{in}_{\mathcal{B}},\ r \in Q_{\mathcal{B}},\ a \in \{0, 1\}$ such that $\langle q, a, r \rangle \in \widehat{g}$ and $\langle r, 1, r \rangle \in \widehat{h}$.

Proof:

(1) Immediate from Lemma 3.3.3.

(2) Recall that Y_{gh} is the ω-language $L(\widetilde{g}) \cdot L(\widetilde{h})^\omega$ and that $Z_{gh} \subseteq Y_{gh}$. By Lemma 2.1.3 either $Y_{gh} \subseteq L(\mathcal{B})$ or $Y_{gh} \cap L(\mathcal{B}) = \emptyset$. Further, by Lemma 2.1.7 $Y_{gh} \subseteq L(\mathcal{B})$ iff a q, r and a exist such that $\langle q, a, r \rangle \in \widetilde{g}$ and $\langle r, 1, r \rangle \in \widetilde{h}$. Therefore $Z_{gh} \subseteq L(\mathcal{B})$ iff such a q, r and a exist.

□

Lemma and 3.3.4 provides an algorithm for testing the containment of two automata, $\mathcal{A}$ and $\mathcal{B}$. Search all pairs of supergraphs, $\widehat{g}, \widehat{h} \in \widehat{Q}_{\mathcal{A},\mathcal{B}}$ for a pair is both proper and for which there does not exist a $q \in Q^{in}_{\mathcal{B}}$, $r \in Q_{\mathcal{B}}, a \in \{0, 1\}$ such that $\langle q, a, r \rangle \in \widehat{g}$ and $\langle r, 1, r \rangle \in \widehat{h}$. Such a pair is a counterexample to containment. If no such pair exists, then $L(\mathcal{A}) \subseteq L(\mathcal{B})$. In order to realize this algorithm, we need an easy way to check if a pair of supergraphs is proper. Composition is employed towards this end.

3.4 Composition of Supergraphs

We are again faced with the prospect of searching over supergraphs with empty languages. Just as in universality, we can leverage the notion of composition from

Lee et al. [12] to compute exactly the set of supergraphs with non-empty languages, in essence trading space for time, as well as to test the properness of a pair of graphs in polynomial time. Define the composition of two supergraphs $\widehat{g} = \langle \langle q, r\rangle,\ \widetilde{g}\rangle, \widehat{h} = \langle \langle r, s\rangle,\ \widetilde{h}\rangle$, written $\widehat{g};\widehat{h}$, as the supergraph $\langle \langle q, s\rangle, \widetilde{g};\widetilde{h}\rangle$. Call a supergraph $\widehat{g} = \langle \langle q, q\rangle,\ \widetilde{g}\rangle$ *idempotent* when $\widehat{g};\widehat{g} = \widehat{g}$.

Lemma 3.4.1. *Let $\widehat{g}, \widehat{h} \in \widehat{Q}_{\mathcal{A},\mathcal{B}}$ be two supergraphs. If $\bar{g} = \langle q, r\rangle$, $\bar{h} = \langle r, s\rangle$, then $L(\widehat{g}) \cdot L(\widehat{h}) \subseteq L(\widehat{g};\widehat{h})$.*

Proof: Take two words $u \in L(\widehat{g}),\ v \in L(\widehat{h})$. Note that $\widehat{g};\widehat{h} = \langle \langle q, s\rangle, \widetilde{g};\widetilde{h}\rangle)$. By construction, $u \in L(\langle q, r\rangle)$ and $v \in L(\langle r, s\rangle)$. Definition 3.3.1 therefore implies the existence of a path from q to r over u and a path from r to s over v. Thus $uv \in L(\langle q, s\rangle)$. Similarly, $u \in L(\widetilde{g}),\ v \in L(\widetilde{h})$, and Lemma 3.1.1 implies that $uv \in L(\widetilde{g};\widetilde{h})$. Thus uv is in $L(\langle \langle q, r\rangle, \widetilde{g};\widetilde{h}\rangle)$. □

For a containment problem $L(\mathcal{A}) \subseteq L(\mathcal{B})$, define the subset of $\widehat{Q}_{\mathcal{A},\mathcal{B}}$ corresponding to single letters to be $\widehat{Q}^1_{\mathcal{A},\mathcal{B}} = \{\widehat{g} \mid \widehat{g} \in \widehat{Q}_{\mathcal{A},\mathcal{B}},\ a \in \Sigma,\ a \in L(\widehat{g})\}$. For completeness, we present a constructive definition of $\widehat{Q}^1_{\mathcal{A},\mathcal{B}}$.

Definition 3.4.2.

$$\begin{aligned}\widehat{Q}^1_{\mathcal{A},\mathcal{B}} = \{\langle \langle q, r\rangle, \widetilde{g}\rangle \quad | \quad & q \in Q_{\mathcal{A}}, r \in \rho_{\mathcal{A}}(q, a),\ a \in \Sigma, \\ & \widetilde{g} = \{\langle q', 0, r'\rangle \mid q' \in Q_{\mathcal{B}},\ r' \in (\rho_{\mathcal{B}}(q', a) \setminus F_{\mathcal{B}})\} \cup \\ & \qquad \{\langle q', 1, r'\rangle \mid q' \in Q_{\mathcal{A}},\ r' \in (\rho_{\mathcal{B}}(q', a) \cap F_{\mathcal{B}})\}\}\end{aligned}$$

Let $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$ be the closure of $\widehat{Q}^1_{\mathcal{A},\mathcal{B}}$ under composition.

Lemma 3.4.3. *For two Büchi automata $\mathcal{A}, \mathcal{B}$, every $\widehat{h} \in \widehat{Q}_{\mathcal{A},\mathcal{B}}$ such that $L(\widehat{h}) \neq \emptyset$ is in $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$.*

Proof: Let $\widehat{h} = \langle \langle q, r\rangle,\ \widetilde{h}\rangle$ where $L(\widehat{h}) \neq \emptyset$. This implies there is at least one word $w = a_0...a_{n-1} \in L(\widehat{h})$, which is to say $w \in L(\langle q, r\rangle) \cap L(\widetilde{h})$. By Definition 3.3.1, there is a path $p = p_0...p_n$ in $\mathcal{A}$ over w such that $p_0 = q$ and $p_n = r$.

In the context of universality, we defined $\widetilde{g}_{a_i}$ to be the graph in $\widetilde{Q}^1_{\mathcal{B}}$ containing a_i. Let $\widehat{g}_{a_i}$ be $\langle \langle p_i, p_{i-1}\rangle, \widetilde{g}_{a_i}\rangle$, and let $\widehat{g}_w$ be $\widehat{g}_{a_0};\widehat{g}_{a_1};...;\widehat{g}_{a_{n-1}}$. Note that each $\widehat{g}_{a_i} \in \widehat{Q}^1_{\mathcal{A},\mathcal{B}}$. By Lemma 3.4.1 $w \in \widetilde{g}_w$. By Lemma 2.1.2 w is in only one graph and $\widetilde{g}_w = \widetilde{h}$. By construction, $\bar{g}_w = \langle q, r\rangle$. Therefore $\langle \bar{g}_w, \widetilde{g}_w\rangle = \langle \langle q, r\rangle, \widetilde{h}\rangle = \widehat{h}$ is in the closure of $\widehat{Q}^1_{\mathcal{A},\mathcal{B}}$ under composition. □

We also employ composition to test the containment of languages.

Lemma 3.4.4. *Let $\widehat{g}, \widehat{h}, \widehat{k}$ be supergraphs in $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$ such that $\bar{g} = \langle q, r\rangle$, $\bar{h} = \langle r, s\rangle$, and $\bar{k} = \langle q, s\rangle$. Then $\widehat{g};\widehat{h} = \widehat{k}$ iff $L(\widehat{g}) \cdot L(\widehat{h}) \subseteq L(\widehat{k})$*

Proof: Take $\widehat{g};\widehat{h} = \widehat{k}$ as a premise. By Lemma 3.4.1, $L(\widehat{g}) \cdot L(\widehat{h}) \subseteq L(\widehat{k})$.

In the other direction, if $L(\widehat{g}) \cdot L(\widehat{h}) \subseteq L(\widehat{k})$, we show that $\widehat{g};\widehat{h} = \widehat{k}$. Recall that when $\bar{g} = \langle q, r\rangle$ and $\bar{h} = \langle r, s\rangle$, the definition of $\widehat{g};\widehat{h}$ is $\langle\langle q, s\rangle, \widetilde{g};\widetilde{h}\rangle$. As $\widehat{g}, \widehat{h} \in \widehat{Q}^f_{\mathcal{A},\mathcal{B}}$ and is therefore non-empty, there is a word $w \in L(\widehat{g}) \cdot L(\widehat{h})$. This expands to $w \in (L(\bar{g}) \cap L(\widetilde{g})) \cdot (L(\bar{h}) \cap L(\widetilde{h}))$, which implies $w \in L(\widetilde{g}) \cdot L(\widetilde{h})$. By Lemma 3.1.1, w is then in $L(\widetilde{g};\widetilde{h})$. Since, by Lemma 2.1.2, w is the language of exactly one graph, we have that $\widetilde{g};\widetilde{h} = \widetilde{k}$, which proves $\widehat{g};\widehat{h} = \widehat{k}$. □

Lemma 3.4.5. *Given two supergraphs $\widehat{g}, \widehat{h} \in \widehat{Q}^f_{\mathcal{A},\mathcal{B}}$, the pair $(\widehat{g}, \widehat{h})$ is proper iff $\widehat{g};\widehat{h} = \widehat{g}$, $\widehat{h};\widehat{h} = \widehat{h}$, $\bar{g} = \langle q, r\rangle$, $\bar{h} = \langle r, r\rangle$, where $q \in Q^{in}_{\mathcal{B}}$ and $r \in F_{\mathcal{B}}$.*

Proof: Immediate from Lemmas 3.3.4, 3.4.3, and 3.4.4. □

Lemma 3.3.4 and Lemma 3.4.5 provide an algorithm for testing $L(\mathcal{A}) \subseteq L(\mathcal{B})$. First, compute the set $\widehat{Q}^1_{\mathcal{A},\mathcal{B}} \subseteq \widehat{Q}_{\mathcal{A},\mathcal{B}}$. Then construct $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$ by iteratively computing the closure of $\widehat{Q}^1_{\mathcal{A},\mathcal{B}}$ under composition. Finally, search all pairs of supergraphs, $\widehat{g}, \widehat{h} \in \widehat{Q}^f_{\mathcal{A},\mathcal{B}}$, for a pair that fulfills the above conditions and does not contain an $s \in Q^{in}_{\mathcal{B}}$ and $t \in Q_{\mathcal{B}}$ such that $\langle s, a, t\rangle \in \widehat{g}$ and $\langle t, 1, t\rangle \in \widehat{h}$. If such $\widehat{g}, \widehat{h}$ exist, then Z_{gh} is not contained in $L(\mathcal{B})$ and is a counterexample for $L(\mathcal{A})$'s containment in $L(\mathcal{B})$. The absence of such a pair is sufficient to prove containment. We call this search the *double-graph search*, to contrast algorithms in the next section that look for only a single supergraph.

Corollary 3.4.6. *Let $\mathcal{A}$ and $\mathcal{B}$ be two Büchi automata. $L(\mathcal{A}) \subseteq L(\mathcal{B})$ iff: for every pair $(\widehat{g}, \widehat{h})$ of supergraphs in $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$ such that $\widehat{g};\widehat{h} = \widehat{g}$, $\widehat{h};\widehat{h} = \widehat{h}$, $\bar{g} = \langle q, r\rangle$, $\bar{h} = \langle r, r\rangle$, where $q \in Q^{in}_{\mathcal{B}}$ and $r \in F_{\mathcal{B}}$, there exists $s \in Q^{in}_{\mathcal{B}}$, $t \in Q_{\mathcal{B}}$ such that $\langle s, a, t\rangle \in \widehat{g}$ and $\langle t, 1, t\rangle \in \widehat{h}$.*

3.5 Strong Suffix Closure and Containment

In Section 3.1 we defined the strong suffix closure property of languages and used it to simplify the Ramsey-based universality test of certain Büchi automata. We here extend the this notion to handle containment questions $L_1 \subseteq L_2$. As in Section 3.3, we restrict our focus to words in L_1. Instead of requiring L_2 to be closed under arbitrary prefixes, L_2 need only be closed under prefixes that keep the word in L_1.

Definition 3.5.1. *A language L_2 is* strongly suffix closed with respect to L_1 *when L_2 is suffix closed and, for every $w \in L_1 \cap L_2$, $w_1 \in \Sigma^+$, if $w_1 w \in L_1$ then $w_1 w \in L_2$.*

Lemma 3.5.2. *Let $\mathcal{A}$ and $\mathcal{B}$ be two Büchi automata where $Q^{in}_{\mathcal{A}} = Q_{\mathcal{A}}$,[2] every state in $Q_{\mathcal{B}}$ is reachable, and $L(\mathcal{B})$ is strongly suffix closed with respect to $L(\mathcal{A})$. Then*

[2] *With a small amount of work, the restriction that $Q^{in}_{\mathcal{A}} = Q_{\mathcal{A}}$ can be relaxed to the requirement that $L(\mathcal{A})$ be suffix closed.*

$L(\mathcal{A}) \not\subseteq L(\mathcal{B})$ iff $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$ contains a supergraph $\widehat{h} = \langle\langle s, s\rangle, \widetilde{h}\rangle$ where $s \in F_{\mathcal{A}}$, $\widehat{h};\widehat{h} = \widehat{h}$ and there is no arc $\langle r, 1, r\rangle \in \widehat{h}$.

Proof: In one direction, assume $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$ contains a supergraph $\widehat{h} = \langle\langle s, s\rangle, \widetilde{h}\rangle$ where $s \in F_{\mathcal{A}}$, $\widehat{h};\widehat{h} = \widehat{h}$ and there is no arc $\langle r, 1, r\rangle \in \widehat{h}$. We show that Z_{hh} is a proper language not contained in $L(\mathcal{B})$. As $\widehat{h} \in \widehat{Q}^f_{\mathcal{A},\mathcal{B}}$, we know $L(\widehat{h})$ is not empty, implying Z_{hh} is non-empty. As $Q_{\mathcal{A}} = Q^{in}_{\mathcal{A}}$, $s \in Q^{in}_{\mathcal{A}}$. By Lemma 3.4.4, the premise $\widehat{h};\widehat{h} = \widehat{h}$ implies $L(\widehat{h}) \cdot L(\widehat{h}) \subseteq L(\widehat{h})$, and Z_{hh} is proper. Finally, as there is no $\langle r, 1, r\rangle \in \widehat{h}$, by Lemma 3.3.4, Z_{hh} is a counterexample to $L(\mathcal{A}) \subseteq L(\mathcal{B})$.

In the opposite direction, assume that $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$ does not contain a supergraph $\widehat{h} = \langle\langle s, s\rangle, \widetilde{h}\rangle$ where $s \in F_{\mathcal{A}}$, $\widehat{h};\widehat{h} = \widehat{h}$, and there is no arc $\langle r, 1, r\rangle \in \widehat{h}$. We prove that every word $w \in L(\mathcal{A})$ is in $L(\mathcal{B})$. Take a word $w \in L(\mathcal{A})$. By Lemma 3.3.3, w is in some proper language Z_{gh} and can be broken into $w_1 w_2$ where $w_1 \in L(\widehat{g}),\ w_2 \in L(\widehat{h})^\omega$.

Because Z_{gh} is proper, Lemma 3.4.4 implies $\widehat{h} = \langle\langle s, s\rangle, \widetilde{h}\rangle$ where $s \in F_{\mathcal{A}}$ and $\widehat{h};\widehat{h} = \widehat{h}$. By assumption, $\widehat{h}$ then must contain an arc $\langle r, 1, r\rangle$. Since all states in $Q_{\mathcal{B}}$ are reachable, there is $q \in Q^{in}_{\mathcal{B}}$ and $u \in \Sigma^+$ with a path in $\mathcal{B}$ from q to r over u. By Lemma 2.1.7, this implies uw_2 is accepted by $\mathcal{B}$. For $L(\mathcal{B})$ to strongly suffix closed with respect to $L(\mathcal{A})$, it must be suffix closed. Therefore $w_2 \in L(\mathcal{B})$. Note that as $Q^{in}_{\mathcal{A}} = Q_{\mathcal{A}}$, $L(\mathcal{A})$ is suffix closed, and thus the premise $w_1 w_2 \in L(\mathcal{A})$ implies $w_2 \in L(\mathcal{A})$. Since $L(\mathcal{B})$ is strongly suffix closed with respect to $L(\mathcal{A})$, $w_1 w_2 \in L(\mathcal{B})$. □

Lemma 3.5.2 provides a direct test for the containment of $\mathcal{A}$ in $\mathcal{B}$ when $L(\mathcal{B})$ is strongly suffix closed with respect to $L(\mathcal{A})$. Search all supergraphs in $\widehat{Q}_{\mathcal{A},\mathcal{B}}$ for an supergraph $\widehat{h}$ where $\widehat{h};\widehat{h} = \widehat{h}$ that does not contain an arc of the form $\langle r, 1, r\rangle$. The presence of this counterexample refutes containment, and the absence of such a supergraph proves containment. We call this search the *single-graph search*.

3.6 From Ramsey-Based Containment to Size-Change Termination

We can now delve into the connection between the graph-theoretic LJB algorithm for size-change termination and the Ramsey-based containment test of $Flow(L) \subseteq Desc(L)$. Size-change graphs in the graph-theoretic solution of an SCT problem are direct analogues of supergraphs in the Ramsey-based containment test of $Flow(L) \subseteq Desc(L)$. We define this analogy and show that the LJB algorithm presented in Section 2.3 is a specialized realization of the Ramsey-based containment test of Lemma 3.5.2.

Noting that the LJB algorithm examines single size-change graphs $G = G;G$, we show that for an SCT problem $L = \langle H, P, C, \mathcal{G}\rangle$ the conditions of Lemma 3.5.2 are met. First, every state in $\mathcal{A}_{Flow(L)}$ is an initial state. Second, every state in $\mathcal{A}_{Desc(L)}$

corresponding to a function in $\mathcal{A}_{Desc(L)}$ is initial, and every state corresponding to a labeled parameter is reachable, and so every state in $\mathcal{A}_{Desc(L)}$ is reachable.[3] Finally, we demonstrate that $Desc(L)$ is strongly suffix closed with respect to $Flow(L)$. Recall that $Flow(L)$ is the language of all call sequences, and $Desc(L)$ the language of all call sequences with a thread with infinitely many 1-labels. The suffix of a call sequence is still a call sequence, so given a call sequence w with a such a thread, every suffix of w will be a call sequence containing contain a suffix of that thread, and so will remain in $Desc(L)$. Further, for every w_1 so that $w_1 w \in Flow(L)$, we know $w_1 w$ is a call sequence with the same thread, and is thus in $Desc(L)$. Therefore $Desc(L)$ is strongly suffix closed with respect to $Flow(L)$.

So $\mathcal{A}_{Flow(L)}$ and $\mathcal{A}_{Desc(L)}$ satisfy the requirements of Lemma 3.5.2, and we can use the single-graph search. Also, observe that every state in $\mathcal{A}_{Flow(L)}$ is accepting. Thus we can conclude that $\mathcal{A}_{Flow(L)} \not\subseteq \mathcal{A}_{Desc(L)}$ iff there is a supergraph $\widehat{h} \in \widehat{Q}^f_{\mathcal{A}_{Flow(L)},\mathcal{A}_{Desc(L)}}$ so that $\widehat{h};\widehat{h} = \widehat{h}$ and $\widehat{h}$ contains no arc of the form $\langle r, 1, r\rangle$.

Consider supergraphs in $\widehat{Q}_{\mathcal{A}_{Flow(L)},\mathcal{A}_{Desc(L)}}$. In the context of size-change termination, a word corresponds to a call sequence. The state space of $\mathcal{A}_{Flow(L)}$ is the set of functions H, and the state space of $\mathcal{A}_{Desc(L)}$ is the union of H and Q_1, the set of all $\{0,1\}$-labeled parameters. A supergraph in $\widehat{Q}_{\mathcal{A}_{Flow(L)},\mathcal{A}_{Desc(L)}}$ thus comprises an arc $\langle q, r\rangle$ in H and a $\{0,1\}$-labeled graph $\widetilde{g}$ over $H \cup Q_1$. The arc asserts the existence of a call path from q to r, and the graph $\widetilde{g}$ captures the relevant information about corresponding paths in $\mathcal{A}_{Desc(L)}$.

These supergraphs are almost the same as size-change graphs, $G : q \to r$. Aside from notational differences, both contain an arc asserting the existence of a call path between two functions and a $\{0,1\}$-labeled graph. There are vertices in both graphs that correspond to parameters of functions, and arcs between two such vertices describe a thread between the corresponding parameters. The analogy falls short, however, on three points:

(1) In size-change graphs, vertices are unlabeled. In supergraphs, vertices are labeled either 0 or 1.

(2) In size-change graphs, vertices are always parameters of functions. In supergraphs, vertices can be either parameters of functions or function names.

(3) In size-change graphs, all vertices correspond to parameters of a two specific functions. In supergraphs, vertices exist that correspond to all parameters of all functions.

We show in turn that each differences is an opportunity to specialize the Ramsey-based containment algorithm. In the end we obtain a set $\widehat{J}$ of simplified supergraphs, so that every $\widehat{h} \in \widehat{Q}_{\mathcal{A}_{Flow(L)},\mathcal{A}_{Desc(L)}}$ that could be a counterexample has a corresponding

[3] In the original reduction, 1-labeled parameters may not have been reachable.

supergraph in $\widehat{J}$. In specific $\widehat{J}^f$, the subset of $\widehat{J}$ corresponding to $\widehat{Q}^f_{\mathcal{A}_{Flow(L)},\mathcal{A}_{Desc(L)}}$, in one-to-one correspondence with S, the closure of the set of size-change graphs under composition, and a counterexample exists in S iff $\widehat{J}^f$ contains an idempotent supergraph $\widehat{h}$ with no arc $\langle r, 1, r\rangle$.

(1) The labels on parameters are the result of encoding a Büchi edge acceptance condition in a Büchi state acceptance condition automaton, and can be dropped from supergraphs with no loss of information. Consider an arc $\langle\langle f, a\rangle, b, \langle g, c\rangle\rangle$ in a supergraph from $\widehat{Q}_{\mathcal{A}_{Flow(L)},\mathcal{A}_{Desc(L)}}$. If b is 1, we know the corresponding thread contains a descending arc. The value of c tells us if the final arc in the thread is descending, but we do not care which arc in the thread is descending. When constructing the initial set of supergraphs, we can look at the SCGs to discover which threads of length one contain descent and mark the corresponding arcs with 1. Thus it is safe to simplify supergraphs in $\widehat{J}$ by removing labels on parameters.

(2) No functions in H are accepting for $\mathcal{A}_{Desc(L)}$, and once we transition out of H into Q_1 we can never return to H. Therefore vertices corresponding to function names can never be part of a descending arc $\langle r, 1, r\rangle$. Since we only search $\widehat{J}$ for a cycle $\langle r, 1, r\rangle$, we can simplify supergraphs in $\widehat{J}$ by removing all vertices corresponding to functions.

(3) While all parameters are states in $\mathcal{A}_{Desc(L)}$, each supergraph describes threads in a call sequence between two functions. There are no threads in this call sequence between parameters of other functions, and so no supergraph with a non-empty language has arcs between the parameters of other functions. We can thus simplify supergraphs in $\widehat{J}$ by removing all vertices corresponding to other parameters.

We now define the set $\widehat{J}$ for an SCT problem $L = \langle H, P, C, \mathcal{G}\rangle$

Definition 3.6.1. $\widehat{J}_L = \{\langle\langle f_1, f_2\rangle, \widetilde{\jmath}\rangle \mid f_1, f_2 \in H,\ \widetilde{\jmath} \subseteq 2^{P(f_1)\times\{0,1\}\times P(f_2)}\}$

Say that $\langle r, \widetilde{g}\rangle \in \widehat{Q}_{\mathcal{A}_{Flow(L)},\mathcal{A}_{Desc(L)}}$ *simplifies to* $\langle r, \widetilde{\jmath}\rangle \in \widehat{J}$ when $\langle q, b, r\rangle \in \widetilde{\jmath}$ iff there exists $a, c \in \{0, 1\}$ such that $\langle\langle q, a\rangle, b, \langle r, c\rangle\rangle \in \widetilde{g}$. Let $\widehat{J}^1$ be $\{\widehat{\jmath} \mid \widehat{\jmath} \in \widehat{J},\ \widehat{g} \in \widehat{Q}^1_{\mathcal{A}_{Flow(L)},\mathcal{A}_{Desc(L)}},\ \widehat{g}$ simplifies to $\widehat{\jmath}\}$, and $\widehat{J}^f$ be the closure of $\widehat{J}^1$ under composition.

We can map SCGs directly to elements of $\widehat{J}$. Say $G : f_1 \to f_2 \equiv \langle\langle f_1, f_2\rangle, \widetilde{g}\rangle$ when $q \xrightarrow{a} r \in G$ iff $\langle q, a, r\rangle \in \widetilde{g}$. Note that the composition operations for supergraphs of this form is identical to the composition of SCGs. If $G_1 \equiv \widehat{g}$ and $G_2 \equiv \widehat{h}$, then $G_1; G_2 \equiv \widehat{g}; \widehat{h}$. Therefore every element of $\widehat{Q}^f_{\mathcal{A}_{Flow(L)},\mathcal{A}_{Desc(L)}}$ simplifies to some element of $\widehat{J}^f$.

We can thus specialize the Ramsey-based containment algorithm for $\mathcal{A}_{Flow(L)} \subseteq \mathcal{A}_{Desc(L)}$ in two ways. First, by Lemma 3.5.2 we know that $Flow(L) \subseteq Desc(L)$ if and only if $\widehat{Q}_{\mathcal{A}_{Flow(L)},\mathcal{A}_{Desc(L)}}$ contains an idempotent graph $\widehat{g} = \widehat{g}; \widehat{g}$ with no arc of the form $\langle r, 1, r\rangle$. Secondly, we can simplify supergraphs in $\widehat{Q}_{\mathcal{A}_{Flow(L)},\mathcal{A}_{Desc(L)}}$ by

removing the labels on parameters and keeping only the vertices associated with appropriate parameters, thus obtaining a set $\widehat{J}$ of supergraphs that are in one-to-one correspondence with SCGs. As every state in $Flow(L)$ is accepting, every idempotent supergraph can serve as a counterexample. Since the supergraphs whose languages contain single characters are in one-to-one correspondence with $\mathcal{G}$, every idempotent graph in $\widehat{J}^f$ contains an arc of the form$\langle r, 1, r\rangle$ exactly when $\mathcal{G}$ is cycle terminating.

Lemma 3.6.2. *Let $L = \langle H, P, C, \mathcal{G}\rangle$ be an SCT problem.*

(1) $\widehat{J}^1_L$ is in one-to-one correspondence with $\mathcal{G}$

(2) L is not *size-change terminating iff $\widehat{J}^f_L$ contains a supergraph $\widehat{\jmath}$ where $\widehat{\jmath};\widehat{\jmath} = \widehat{\jmath}$ that contains no arc of the form $\langle r, 1, r\rangle$.*

Proof:
(1): Every size-change graph $G : f_1 \to f_2 \in \mathcal{G}$ is the SCG for a call site c from f_1 to f_2. This is a call sequence of length one. Thus there is a $\widehat{g} \in \widehat{Q}^1_{\mathcal{A}_{Flow(L)},\mathcal{A}_{Desc(L)}}$ so that $c \in L(\widehat{g})$ and $\bar{g} = \langle f_1, f_2\rangle$. $\widehat{g}$ simplifies to some $\widehat{\jmath} \in \widehat{J}$. By Definition 2.3.6, the arc $\langle q, a, r\rangle \in \widehat{\jmath}$ when $q \xrightarrow{a} r \in G$. Thus $\widehat{\jmath} \equiv G : f_1 \to f_2$.

(2): By (1), $\mathcal{G}$ is in one-to-one correspondence with $\widehat{J}^1$. Therefore S is in one-to-one correspondence with the $\widehat{J}^f$. This this follows both from Theorem 2.3.11 and from Lemma 3.5.2. □

Chapter 4
Towards an Empirical Comparison

Having shown that the LJB algorithm for size-change termination is a specialized realization of the Ramsey-based containment test, we want to compare existing SCT analysis tools to existing rank-based complementation tools. We propose two problem domains over which we can empirically measure the scalability of Ramsey-based and rank-based tools. The first domain consists of size-change termination problems. To facilitate a fair comparison, we present an improved reduction from size-change termination to Büchi automata containment and formalize a subsumption relation presented by Lee and Ben-Amram in [2]. The second domain we want to examine the behavior of Ramsey-based and rank-based tools on consists of Büchi containment problems. To do so, we modify existing SCT tools to handle arbitrary Büchi automata containment problems. In moving from SCT problems to containment problems, Ramsey-based algorithms must abandon the convenience of strongly suffix-closed languages and the single-graph search, instead testing every proper pair of supergraphs. This induces an exponential blowup in the search space. We examine the constraints of properness to reduce this blowup. We then show how to leverage the subsumption relation presented above for arbitrary Büchi automata containment problem.

4.1 A Tighter Reduction from SCT to Büchi Containment

In constructing the analogy between SCGs in the LJB algorithm and supergraphs in the Ramsey-based containment algorithm, we noticed that supergraphs contain vertices for every parameter, while size-change graphs contain only vertices corresponding to parameters of relevant functions. These vertices are states in $\mathcal{A}_{Desc(L)}$. While we can specialize the Ramsey-based test to avoid them, Büchi containment solvers might suffer. These states duplicate information. As we already know which functions each supergraph corresponds to, there is no need for each vertex to be unique to a specific function.

The extra states emerge because $Desc(L)$ only accepts strings that are contained in $Flow(L)$, and in doing so demands that parameters only be reached by appropriate call paths. But the behavior of $\mathcal{A}_{Desc(L)}$ on strings not in $Flow(L)$ is irrelevant to the question of $Flow(L) \subseteq Desc(L)$, and we can replace the names of parameters in $\mathcal{A}_{Desc(L)}$ with their location in the argument list. Further, we can rely on $Flow(L)$ to verify the sequence of function calls before our accepting thread and make do with a

single waiting state.

Definition 4.1.1. *Given an SCT problem $L = \langle H, P, C, \mathcal{G}\rangle$ and a projection Ar of all parameters onto their positions $1..n$ in the argument list, define:*

$$\begin{array}{rcl} \mathcal{A}'_{Desc(L)} & = & \langle C, S \cup \{q_0\}, \{q_0\}, \rho_D, F\rangle \\ & \text{where} & S = \{1..n\} \times \{1, 0\} \\ & & \rho_D(q_0, c) = \{q_0\} \cup \{\langle Ar(x), 0\rangle \mid c : f_1 \to f_2,\ x \in P(f_2)\} \\ & & \rho_D(\langle n, a\rangle, c) = \{\langle Ar(y), a'\rangle \mid x \xrightarrow{a'} y \in \mathcal{G}_c,\ n = Ar(x)\} \\ & & F = \{1..n\} \times \{1\} \end{array}$$

Lemma 4.1.2. *$L(\mathcal{A}_{Flow(L)}) \subseteq L(\mathcal{A}_{Desc(L)})$ iff $L(\mathcal{A}_{Flow(L)}) \subseteq L(\mathcal{A}'_{Desc(L)})$*

Proof: The languages of $\mathcal{A}_{Desc(L)}$ and $\mathcal{A}'_{Desc(L)}$ are not the same. What we demonstrate is that for each word in $Flow(L)$, we can convert an accepting run in one of $\mathcal{A}_{Desc(L)}$ or $\mathcal{A}'_{Desc(L)}$ into an accepting run in the other. Recall that the states of $\mathcal{A}_{Flow(L)}$ are functions $f \in H$. States of $\mathcal{A}_{Desc(L)}$ are either elements of H or elements of $\bigcup_{f\in H} P(f) \times \{1, 0\}$, the set of labeled parameters.

Consider a word w with an accepting run $r = r_0 r_1 ...$ of $\mathcal{A}'_{Desc(L)}$ and an accepting run $s = s_0 s_1 ...$ of $\mathcal{A}_{Flow(L)}$. We define an accepting run $t = t_0 t_1 ..$ of $\mathcal{A}_{Desc(L)}$ on w. Each t_i depends on the corresponding r_i and s_i. If $r_i = q_0$ and $s_i = f$, then $t_i = f$. If $r_i = \langle k, a\rangle$, $1 \le k \le n$, and $s_i = f$, then $t_i = \langle x, a\rangle$ where x is the kth parameter in f's argument list.

For a call $c : f_1 \to f_2$, take two labeled parameters, q a labeled parameter of f_1 and r a labeled parameter of f_2. Consider the corresponding pairs $\langle f_1, Ar(q)\rangle, \langle f_2, Ar(r)\rangle$. If $\langle Ar(q), c, Ar(r)\rangle$ is a transition in $\mathcal{A}'_{Desc(L)}$ then $\langle q, c, r\rangle$ is a transition in $\mathcal{A}_{Desc(L)}$. Therefore t is a run of $\mathcal{A}_{Desc(L)}$ on w. Further, note that $\langle x, a\rangle \in F_{\mathcal{A}_{Desc(L)}}$ and $\langle Ar(x), a\rangle \in F_{\mathcal{A}_{Desc(L)}}$ iff $a = 1$. Therefore t is an accepting run.

Conversely, consider an accepting run $r = r_0 r_1 ...$ of $\mathcal{A}_{Desc(L)}$ over a word w. Let $s = s_0 s_1 ...$ be the sequence of states in $\mathcal{A}'_{Desc(L)}$ where if r_i is a function then $s_i = q_0$, and $s_i = Ar(r_i)$ otherwise. By the definition of $\mathcal{A}'_{Desc(L)}$, q_0 always transitions to q_0 and a transition between r_i and r_{i+1} implies a transition between $Ar(r_i)$ and $Ar(r_{i+1})$. As above, if $\langle x, a\rangle$ is an accepting state in $\mathcal{A}_{Desc(L)}$, $a = 1$ and then $\langle Ar(x), a\rangle$ is an accepting state $\mathcal{A}'_{Desc(L)}$. Thus, s is an accepting run of $\mathcal{A}'_{Desc(L)}$ over w. □

4.2 Subsumption in the Single-Graph Search

In [2], Lee and Ben-Amram optimize the LJB algorithm for size-change termination by removing certain size-change graphs when computing the closure under composition. This suggests that the algorithm afforded by Lemma 3.2.2 can benefit

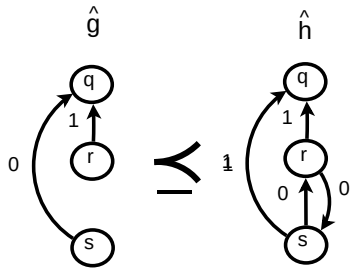


Figure 4.1: Subsumption: two supergraphs $\widehat{g}$ and $\widehat{h}$, where $\widehat{g} \preceq \widehat{h}$.

from a subsumption relation. We here formalize this subsumption relation on supergraphs, showing how to discard supergraphs that are conservatively approximated by other supergraphs. In order to do so safely, we must then replace the search for a single arc in idempotent graphs with a search for a strongly connected component in all graphs.

When computing the closure of a set of supergraphs under compositions, we can ignore elements when they are conservatively approximated by other elements. Intuitively, a supergraph $\widehat{g}$ conservatively approximates another supergraph $\widehat{h}$ when it is strictly harder to find a 1-labeled sequence of arcs through $\widehat{g}$ than through $\widehat{h}$. If we can replace $\widehat{h}$ by $\widehat{g}$ in every composition without removing arcs from the result, we do not have to consider $\widehat{h}$. When the right arc can be found in $\widehat{g}$, then it also occurs in $\widehat{h}$. When $\widehat{g}$ does not have a satisfying arc, then we already have a counterexample supergraph.

Formally, given two graphs $\widehat{g}, \widehat{h} \in \widehat{Q}_{\mathcal{A},\mathcal{B}}$ where $\bar{g} = \bar{h}$, say that $\widehat{g}$ *conservatively approximates* $\widehat{h}$, written $\widehat{g} \preceq \widehat{h}$, when for every arc $\langle q, a, r\rangle \in \widehat{g}$ there is an arc $\langle q, a', r\rangle \in \widehat{h}$, where if $a = 1$ then $a' = 1$. An example is provided in Figure 4.2. Note that conservative approximation is a transitive relation. Say that $\widehat{g} \prec \widehat{h}$ when $\widehat{g} \preceq \widehat{h}$ and $\widehat{g} \neq \widehat{h}$. Using the notion of conservative approximation, we present an algorithm that computes a subset of $\widehat{Q}^{f}_{\mathcal{A},\mathcal{B}}$.

Algorithm 4.2.1.

Given a set of supergraphs $\widehat{Q}^{1}_{\mathcal{A},\mathcal{B}}$, let $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$ be a set obtained by[1]*:*

(1) Initialize $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$ as $\widehat{Q}^{1}_{\mathcal{A},\mathcal{B}}$

(2) For every $\widehat{g}, \widehat{h} \in \widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$, if there is no $\widehat{k} \in \widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$ such that $\widehat{k} \preceq \widehat{g}; \widehat{h}$:

(a) Include $\widehat{g}; \widehat{h}$ in $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$

[1] *It can be proven that this set is unique, but this is not necessary for this algorithm*

(b) For every $\widehat{k} \in \widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}} \setminus \widehat{Q}^{1}_{\mathcal{A},\mathcal{B}}$, if $\widehat{g};\widehat{h} \prec \widehat{k}$ then remove $\widehat{k}$ from $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$

We can now limit our search to supergraphs in $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$. However, we cannot replace $\widehat{Q}^{f}_{\mathcal{A},\mathcal{B}}$ by $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$ and continue to test only idempotent elements. Since we are now removing elements, it is conceivable that we might never compute the idempotent element of $\widehat{Q}^{f}_{\mathcal{A},\mathcal{B}}$ that provides the counterexample to containment. To proceed, we simplify Lemma 3.5.2 by removing the requirement for supergraphs to be idempotent. Instead of examining only idempotent supergraphs $\langle\langle s, s\rangle, \widetilde{g}\rangle$ where $s \in F_{\mathcal{A}}$ and $\widetilde{g};\widetilde{g} = \widetilde{g}$, we examine every supergraph with a reflexive arc $\langle\langle s, s\rangle, \widetilde{g}\rangle$ where $s \in F_{\mathcal{A}}$. When examining such a supergraph that may not be idempotent, we must test for a path from q to q instead of testing for a single arc $\langle q, 1, q\rangle$. We test for this path by computing the strongly connected components of $\widehat{g}$, and testing if the strongly connected component containing q has a 1-labeled arc.

A *strongly connected component* (SCC) of a supergraph $\widehat{g}$ is a maximal set of vertices, S, so that for every $q, r \in S$ there is a path from q to r, and a path from r to q. Computing the strongly connected components of a graph can be done in linear time with a depth-first search [5]. Say that an SCC S is 1-labeled when there are $q, r \in S$ with an arc $\langle q, 1, r\rangle \in \widehat{g}$. Once we partition the vertices into strongly connected components, we can simply search for a 1-labeled SCC. As we later show, $\widehat{Q}^{f}_{\mathcal{A},\mathcal{B}}$ contains an idempotent supergraph $\langle\langle s, s\rangle, \widetilde{g}\rangle$ with an arc $\langle q, 1, q\rangle$ precisely when $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$ contains a supergraph $\langle\langle s, s\rangle, \widetilde{h}\rangle$ with a 1-labeled SCC containing q. This provides a containment testing algorithm that avoids computing the entirety of the set $\widehat{Q}^{f}_{\mathcal{A},\mathcal{B}}$.

Algorithm 4.2.2.

(1) Construct the set $\widehat{Q}^{1}_{\mathcal{A},\mathcal{B}}$ of all single-character supergraphs

(2) Compute $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$ using Algorithm 4.2.1

(3) When computing each $\langle\langle q, q\rangle, \widetilde{g}\rangle \in \widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$, if $q \in F_{\mathcal{A}}$ then:

- *Compute the strongly connected components of $\widetilde{g}$*
- *Ensure there exists a 1-labeled strongly connected component of $\widetilde{g}$*

To prove Algorithm 4.2.2 correct, we proceed in two steps. First, we demonstrate that testing the strongly connected components of every supergraph with a reflexive arc is equivalent to testing for 1-labeled arcs in idempotent supergraphs. Second, we demonstrate that $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$ contains a supergraph with a 1-labeled SCC if and only if $\widehat{Q}^{f}_{\mathcal{A},\mathcal{B}}$ contains a supergraph with a 1-labeled SCC.

Let $\widehat{g} = \langle\langle s, s\rangle, \widetilde{g}\rangle$ be a supergraph with a reflexive arc, where $s \in F_{\mathcal{A}}$. In testing for an arc $\langle q, 1, q\rangle \in \widehat{g}$, we are asking if $L(\widehat{g})^{\omega}$ has an accepting run in $\mathcal{B}$ starting in each state. By ensuring that $\widehat{g}$ is idempotent, i.e. $\widehat{g};\widehat{g} = \widehat{g}$, we know that every path in $\mathcal{B}$ over a word in $L(\widehat{g})^{+}$ is described in $\widehat{g}$. Thus, if there is no arc $\langle q, 1, q\rangle \in \widehat{g}$, there cannot be an accepting path in $\mathcal{B}$ from q to q on a word in $L(\widehat{g})^{+}$. If no q

has such an arc, we know there cannot be an accepting run beginning in any state on a word in $L(\widehat{g})^\omega$. In order to remove the requirement $\widehat{g};\widehat{g} = \widehat{g}$, we can no longer assume that a path in $\mathcal{B}$ from q to q on a word in $L(\widehat{g})^+$ is matched by an arc $\langle q, a, q\rangle \in \widehat{g}$. Instead we must search for a path from q to q, i.e. a sequence of arcs in $\widehat{g}$, $\langle q, a_0, p_1\rangle, \langle p_1, a_1, p_2\rangle, ...\langle p_{n-1}, a_{n-1}, q\rangle$ where $a_i = 1$ for some i. Say a sequence of arcs is 1-labeled when it contains at least one 1-labeled arc. Say that a supergraph has a *1-labeled q-cyclic path* when there is a 1-labeled sequence of arcs from q to q.

To discuss sequences of arcs in supergraphs, we consider exponentiation of supergraphs. Let $\widehat{g}^n$ be the composition of n copies of $\widehat{g}$, $\widehat{g};...\widehat{g}$. Let $\widehat{g}^*$ be the first $\widehat{g}^n$ where $\widehat{g}^n;\widehat{g}^n = \widehat{g}^n$. We first show that $\widehat{g}^*$ is well-defined for any supergraph with a reflexive arc.

Lemma 4.2.3. *Let $\widehat{g} \in \widehat{Q}_{\mathcal{A},\mathcal{B}}$ be a supergraph with a reflexive arc. There exists an $n \in \mathbb{N}$ such that $\widehat{g}^n;\widehat{g}^n = \widehat{g}^n$*

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

Let a be the index of the first occurrence of $\widehat{h}$, so that $\widehat{g}^a = \widehat{h}$. Let b be the first index larger than $2a$ so that $\widehat{g}^b = \widehat{h}$. Let c be $b - 2a$. We prove that $a + c$ is such an n, i.e. $\widehat{g}^{a+c};\widehat{g}^{a+c} = \widehat{g}^{a+c}$.

(1) As $\widehat{g}^a = \widehat{g}^b$, for every $i > 0$ it holds that $\widehat{g}^{a+i} = \widehat{g}^{b+i}$

(2) By the definition of exponentiation, $\widehat{g}^{a+c};\widehat{g}^{a+c} = \widehat{g}^{2a+2c}$

(3) By definition, $2a + c = b$

(4) By (2) and (3), $\widehat{g}^{a+c};\widehat{g}^{a+c} = \widehat{g}^{b+c}$

(5) By (1), $\widehat{g}^{a+c} = \widehat{g}^{b+c}$

(6) By (4) and (5), $\widehat{g}^{a+c};\widehat{g}^{a+c} = \widehat{g}^{a+c}$

□

We now show that it is safe to perform the single-graph search by searching for a supergraph with a reflexive arc but no 1-labeled q-cyclic path. We first relate a 1-labeled q-cyclic path in $\widehat{g}$ to the arc $\langle q, 1, q\rangle$ in $\widehat{g}^*$, and then show that checking every supergraph with a reflexive arc for a 1-labeled q-cyclic path is equivalent to check idempotent supergraphs for arcs of the form $\langle q, 1, q\rangle$.

Lemma 4.2.4.

For every $\widehat{g} \in \widehat{Q}_{\mathcal{A},\mathcal{B}}$ of the form $\langle\langle s, s\rangle, \widetilde{g}\rangle$:

(1) If $\widehat{g};\widehat{g} = \widehat{g}$ and $\widehat{g}$ has a 1-labeled q-cyclic path, then $\widehat{g}$ has an arc $\langle q, 1, q\rangle$.

(2) If $\widehat{g}^*$ has an arc $\langle q, 1, q\rangle$, then $\widehat{g}$ has a 1-labeled q-cyclic path.

Proof:
(1) Assume $\widehat{g}$ has a 1-labeled q-cyclic path. This path is a sequence of arcs in $\widehat{g}$ of length n: $\langle q, a_0, p_1\rangle, ...\langle p_n, a_n, q\rangle$, where $a_i = 1$ for some i. By the definition of composition, $\langle q, 1, q\rangle \in \widehat{g}^n$. As $\widehat{g};\widehat{g} = \widehat{g}$, we have that $\widehat{g}^n = \widehat{g}$ and $\langle q, 1, q\rangle \in \widehat{g}$.

(2) First, note that, by Lemma 4.2.3, $\widehat{g}^*$ exists. Assume $\widehat{g}^*$ has an arc of the form $\langle q, 1, q\rangle$. We show that $\widehat{g}$ has a 1-labeled q-cyclic path. The premise implies that, for some n, $\widehat{g}^n$ has an arc $\langle q, 1, q\rangle$. By the definition of composition, this implies there is a sequence of n arcs in $\widehat{g}$, $\langle q, a_0, r_1\rangle, \langle r_1, a_1, r_2\rangle ...\langle r_{n-1}, a_{n-1}, q\rangle$ where $a_i = 1$ for some i. This is a 1-labeled q-cyclic path in $\widehat{g}$. □

Lemma 4.2.5. *Let $\widehat{Q}^1_{\mathcal{A},\mathcal{B}}$ be a set of supergraphs, and $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$ the closure of $\widehat{Q}^1_{\mathcal{A},\mathcal{B}}$ under composition. $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$ contains an idempotent supergraph $\widehat{g} = \langle \langle s, s\rangle, \widetilde{g}\rangle$ without the arc $\langle q, 1, q\rangle$ iff $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$ contains a supergraph $\widehat{h} = \langle \langle s, s\rangle, \widetilde{h}\rangle$ with no 1-labeled q-cyclic path.*

Proof: In one direction, assume $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$ contains an supergraph $\widehat{g} = \langle \langle s, s\rangle, \widetilde{g}\rangle$, where $s \in F_{\mathcal{A}}$ and $\widehat{g} = \widehat{g};\widehat{g}$, that does not contain the arc $\langle q, 1, q\rangle$. By the first clause of Lemma 4.2.4, $\widehat{g}$ does not contain a 1-labeled q-cyclic path. So take $\widehat{h}$ to be $\widehat{g}$.

In the other direction, assume $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$ contains a graph $\widehat{h} = \langle \langle s, s\rangle, \widetilde{h}\rangle$, where $s \in F_{\mathcal{A}}$, with no 1-labeled q-cyclic path. By Lemma 4.2.3, there is an power, $\widehat{h}^*$, of $\widehat{h}$ such that $\widehat{h}^*;\widehat{h}^* = \widehat{h}^*$. Since $\bar{h} = \langle s, s\rangle$, by the definition of composition $\bar{h}^* = \langle s, s\rangle$. By the second clause of Lemma 4.2.4, $\widehat{h}^*$ does not contain the arc $\langle q, 1, q\rangle$. Finally, since $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$ is closed under composition, $\widehat{h}^* \in \widehat{Q}^f_{\mathcal{A},\mathcal{B}}$. So take $\widehat{g}$ to be $\widehat{h}^*$. □

To search for a 1-labeled q-cyclic path, we partition the vertices into strongly connected components and search for an 1-labeled SCC. Recall that we say say an SCC S is 1-labeled when there are $q, r \in S$ with an arc $\langle q, 1, r\rangle \in \widehat{g}$. Once we partition the vertices into strongly connected components, we can simply search for a 1-labeled SCC. Every 1-labeled q-cyclic path must go through a single strongly connected component, and every 1-labeled strongly connected component S has a 1-labeled q-cyclic path for each state $q \in S$.

Lemma 4.2.6. *For every $\widetilde{g} \in \widetilde{Q}_{\mathcal{A},\mathcal{B}}$, $\widetilde{g}$ has a 1-labeled q-cyclic path iff $\widetilde{g}$ has a 1-labeled SCC containing q.*

Proof: Assume $\widehat{g}$ has a 1-labeled q-cyclic path. There is an arc $\langle r, 1, s\rangle$ somewhere in this 1-labeled sequence. The sequence until this point is a path from q to r, and the remainder of the sequence a path from s to q. Thus r and s belongs to the same SCC as q, and this SCC is 1-labeled.

In the other direction, assume $\widehat{g}$ has a 1-labeled SCC S. S contains two vertices q and r with an arc $\langle q, 1, r\rangle$. As q and r are in the same SCC, there is a sequence from

r to q, and thus a 1-labeled sequence from q to itself. Therefore $\widehat{g}$ has a 1-labeled q-cyclic path. □

When given a potential counterexample supergraph $\widehat{g} = \langle\langle s, s\rangle, \widetilde{g}\rangle$ where $s \in F_{\mathcal{A}}$, we can examine $\widehat{g}$ in two ways. First, we can ensure that if $\widehat{g};\widehat{g} = \widehat{g}$, then $\widehat{g}$ has an arc of the form $\langle q, 1, q\rangle$. Alternatively, Lemma 4.2.6 allows us to simply test $\widehat{g}$ for a 1-labeled strongly connected component. As the strongly connected components of a graph can be computed in linear time, doing so is no more expensive than testing idempotence.

Lemma 4.2.7. *Let $\mathcal{A}$ and $\mathcal{B}$ be two Büchi automata where $Q^{in}_{\mathcal{A}} = Q_{\mathcal{A}}$, every state in $Q_{\mathcal{B}}$ is reachable, and $L(\mathcal{B})$ is strongly suffix closed with respect to $L(\mathcal{A})$. Then $L(\mathcal{A}) \not\subseteq L(\mathcal{B})$ iff $\widehat{Q}^{f}_{\mathcal{A},\mathcal{B}}$ contains a supergraph $\widehat{h} = \langle\langle s, s\rangle, \widetilde{h}\rangle$, where $s \in F_{\mathcal{A}}$, with no 1-labeled strongly connected component.*

Proof: Immediate from Lemmas 3.5.2, 4.2.5, and 4.2.6 □

Now that we have shown it is sufficient to search $\widehat{Q}^{f}_{\mathcal{A},\mathcal{B}}$ for supergraphs with 1-labeled SCC's, we show that it is safe to restrict our search to graphs in $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$. Recall that we say $\widehat{g} \preceq \widehat{h}$ when, for every arc $\langle q, a, r\rangle \in \widehat{g}$, there is an arc $\langle q, a', r\rangle \in \widehat{h}$, where if $a = 1$ then $a' = 1$.

Lemma 4.2.8. *Let $\widehat{g}, \widehat{h}$ be two supergraphs where $\widehat{h} \preceq \widehat{g}$. If $\widehat{h}$ has a 1-labeled SCC, then $\widehat{g}$ has a l-labeled SCC.*

Proof: Let S be a 1-labeled SCC in $\widehat{h}$. Note that for every two states q, r, if there is a path in $\widehat{h}$ from q to r, there must also be a path in $\widehat{g}$ from q to r. Thus all vertices in S are in the same SCC T of $\widehat{h}$. Further, there must be $q, r \in S$ with an arc $\langle q, 1, r\rangle \in \widehat{h}$. This arc must also appear in $\widehat{g}$. Therefore T is a 1-labeled SCC of $\widehat{g}$. □

Lemma 4.2.9. *Let $\widehat{g}, \widehat{h}, \widehat{k}$ be three supergraphs so that $\widehat{g};\widehat{k}$ and $\widehat{h};\widehat{k}$ are well defined. If $\widehat{h} \preceq \widehat{g}$, then $\widehat{h};\widehat{k} \preceq \widehat{g};\widehat{k}$.*

Proof: Take an arc $\langle q, a, s\rangle \in \widehat{h};\widehat{k}$. First assume a is 0. This arc exists because there is state r, an arc $\langle q, 0, r\rangle \in \widehat{h}$, and an arc $\langle r, 0, s\rangle \in \widehat{k}$. As $\widehat{h} \preceq \widehat{g}$, there is an arc $\langle q, a', r\rangle \in \widehat{g}$. Therefore the arc $\langle q, a', s\rangle \in \widehat{g};\widehat{k}$. Since a is not 1, a' is unconstrained.

If a is 1, there is state r, an arc $\langle q, a_1, r\rangle \in \widehat{h}$ and an arc $\langle r, a_2, s\rangle \in \widehat{k}$ where $a_1 = 1$ or $a_2 = 1$. As $\widehat{h} \preceq \widehat{g}$, there is an arc $\langle q, a'_1, r\rangle \in \widehat{g}$. If $a_1 = 1$, then $a'_1 = 1$ and $\langle q, 1, s\rangle \in \widehat{g};\widehat{k}$. If $a_2 = 1$, then regardless of a_1 the arc $\langle q, 1, s\rangle \in \widehat{g};\widehat{k}$. We can now conclude that $\widehat{h};\widehat{k} \preceq \widehat{g};\widehat{k}$. □

Say that a set, $\widehat{Q}_{\mathcal{A},\mathcal{B}}$, of supergraphs is $\preceq$-*closed under composition* when, for every two supergraphs, $\widehat{g}, \widehat{h} \in \widehat{Q}_{\mathcal{A},\mathcal{B}}$, there is a supergraph $\widehat{k} \in \widehat{Q}_{\mathcal{A},\mathcal{B}}$ so that $\widehat{k} \preceq \widehat{g}; \widehat{h}$. Given a set $\widehat{Q}^1_{\mathcal{A},\mathcal{B}}$ of supergraphs we show that it is sufficient to test a $\preceq$-closure subset of $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$, its closure under composition.

Lemma 4.2.10. *Given a set of supergraphs $\widehat{Q}^1_{\mathcal{A},\mathcal{B}}$, and a set $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$ obtained by the procedure above, it holds that $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$ is $\preceq$-closed, $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}} \subseteq \widehat{Q}^f_{\mathcal{A},\mathcal{B}}$, and $\widehat{Q}^1_{\mathcal{A},\mathcal{B}} \subseteq \widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$.*

Proof: First, note that the composition of two elements is only omitted from $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$, on line (2), when it is conservatively approximated by an element already in $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$. Thus $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$ is indeed $\preceq$-closed. Second, note that that $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$ is indeed a subset of $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$, as every element in $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$ is the composition of a finite number of elements of $\widehat{Q}^1_{\mathcal{A},\mathcal{B}}$. Third, since $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$ begins as $\widehat{Q}^1_{\mathcal{A},\mathcal{B}}$, and no element of $\widehat{Q}^1_{\mathcal{A},\mathcal{B}}$ is removed on line (2)(b), every element of $\widehat{Q}^1_{\mathcal{A},\mathcal{B}}$ is in $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$. □

Lemma 4.2.11. *Let $\widehat{Q}^1_{\mathcal{A},\mathcal{B}}$ be a set of supergraphs, $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$ the closure of $\widehat{Q}^1_{\mathcal{A},\mathcal{B}}$ under composition, and $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$ a set obtained by Algorithm 4.2.1. Then $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$ contains a supergraph $\widehat{g} = \langle \langle s, s \rangle, \widetilde{g} \rangle$ with no 1-labeled SCC iff $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$ contains a supergraph $\widehat{h} = \langle \langle s, s \rangle, \widetilde{h} \rangle$, with no 1-labeled SCC.*

Proof: In one direction, assume $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$ contains a supergraph $\widehat{h} = \langle \langle s, s \rangle, \widetilde{h} \rangle$ with no 1-labeled SCC. By Lemma 4.2.10, $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}} \subseteq \widehat{Q}^f_{\mathcal{A},\mathcal{B}}$. Therefore take $\widehat{g}$ to be $\widehat{h}$, and $\widehat{g} \in \widehat{Q}^f_{\mathcal{A},\mathcal{B}}$.

In the other direction, assume $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$ contains a supergraph $\widehat{g} = \langle \langle s, s \rangle, \widetilde{g} \rangle$ with no 1-labeled SCC. Then $\widehat{g}$ is the finite composition $\widehat{g}_0; \widehat{g}_1; ...; \widehat{g}_{n-1}$ of n elements of $\widehat{Q}^1_{\mathcal{A},\mathcal{B}}$. Let $\widehat{g}_{0...i}$ be the sequence of compositions $\widehat{g}_0; ...; \widehat{g}_i$. We prove by induction that, for each i, there is a supergraph $\widehat{h}_i \in \widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$ so that $\widehat{h}_i \preceq \widehat{g}_{0...i}$. Since $\widehat{g} = \widehat{g}_{1...(n-1)}$, this implies $\widehat{h}_{n-1} \preceq \widehat{g}$. As $\widehat{g}$ does not have a 1-labeled SCC, by Lemma 4.2.8, $\widehat{h}_{n-1}$ cannot have a 1-labeled SCC.

As a base case, $\widehat{g}_0$ is in $\widehat{Q}^1_{\mathcal{A},\mathcal{B}}$ and thus by Lemma 4.2.10 is in $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$.

Inductively, assume there is a supergraph $\widehat{h}_i \in \widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$ so that $\widehat{h}_i \preceq \widehat{g}_{0...i}$. We must show that there exists a $\widehat{h}_{i+1} \in \widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$ so that $\widehat{h}_{i+1} \preceq \widehat{g}_{0...i}; \widehat{g}_{i+1}$. First, note that by Lemma 4.2.9 $\widehat{h}_i; \widehat{g}_{i+1} \preceq \widehat{g}_{0...i}; \widehat{g}_{i+1}$. Second, as $\widehat{g}_{i+1} \in \widehat{Q}^1_{\mathcal{A},\mathcal{B}}$, Lemma 4.2.10 implies that $\widehat{g}_{i+1} \in \widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$. As $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$ is $\preceq$-closed under composition, either $\widehat{h}_i; \widehat{g}_{i+1}$ is in $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$, or it is approximated by some $\widehat{k} \in \widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$. In the former case, take $\widehat{h}_{i+1}$ to be $\widehat{h}_i; \widehat{g}_{i+1}$. In the later case, recall that conservative approximation is transitive and take $\widehat{h}_{i+1}$ to be $\widehat{k}$.

We have shown there is there is a $\widehat{h}_{n-1} \in \widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$ so that $\widehat{h}_{n-1} \preceq \widehat{g}$. By the definition of conservative approximation, $\bar{h}_{n-1}$ must be $\langle s, s \rangle$. As $\widehat{g}$ does not have a 1-labeled SCC, neither does $\widehat{h}_{n-1}$. So let $\widehat{h}$ be $\widehat{h}_{n-1}$, and $\widehat{h} \in \widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$. □

Algorithm 4.2.2 is now proven correct.

Lemma 4.2.12. *Let $\mathcal{A}$ and $\mathcal{B}$ be two Büchi automata where $Q^{in}_{\mathcal{A}} = Q_{\mathcal{A}}$, every state in $Q_{\mathcal{B}}$ is reachable, and $L(\mathcal{B})$ is strongly suffix closed with respect to $L(\mathcal{A})$. $L(\mathcal{A})$ is* not *contained in $L(\mathcal{B})$ iff $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$ contains a supergraph $\langle\langle s, s\rangle, \widetilde{g}\rangle$, where $s \in F_{\mathcal{A}}$, with no 1-labeled strongly connected component.*

Proof: Immediate from Lemmas 4.2.7 and 4.2.11. □

4.3 Search Space for the Two-Graph Search

When solving problems whose languages are strongly suffix closed, such as size-change termination problems, a Ramsey-based algorithm can employ the single-graph search and examine only single supergraphs. In addition to allowing the use of a subsumption operation, this significantly reduces the size of the search space. Consider a containment problem, $L(\mathcal{A}) \subseteq L(\mathcal{B})$, where $\mathcal{B}$ has n states and $\mathcal{A}$ has m states. There are potentially $m^2 3^{n^2}$ supergraphs in $\widehat{Q}^{f}_{\mathcal{A},\mathcal{B}}$. If $L(\mathcal{B})$ is strongly suffix closed with respect to $L(\mathcal{A})$, a Ramsey-based algorithm needs to check every supergraph $\widehat{g} \in \widehat{Q}^{f}_{\mathcal{A},\mathcal{B}}$. If $L(\mathcal{B})$ does not have this property, the algorithm needs to use the two-graph search and check every pair $(\widehat{g}, \widehat{h})$ of supergraphs. This blows up the search space from $m^2 3^{n^2}$ to $m^4 9^{n^2}$.

If we could avoid storing all the information about each graph, we could reduce the search space. The second clause of Lemma 3.3.4 provides us with a test for containment that searches for two arcs in every proper pair of supergraphs. Recall that, by Lemma 3.4.5, a pair $(\widehat{g}, \widehat{h})$ of supergraphs from $\widehat{Q}^{f}_{\mathcal{A},\mathcal{B}}$ is proper when:

(1) $\bar{g} = \langle p, q\rangle$

(2) $\bar{h} = \langle q, q\rangle$

(3) $p \in Q^{in}_{\mathcal{A}}$

(4) $q \in F_{\mathcal{A}}$

(5) $\widehat{h}; \widehat{h} = \widehat{h}$

(6) $\widehat{g}; \widehat{h} = \widehat{g}$

Say $(\widehat{g}, \widehat{h})$ *passes the two-state test* when there are two states, $r \in Q^{in}_{\mathcal{B}}, s \in Q_{\mathcal{B}}$, so that $\langle r, a, s\rangle \in \widehat{g}$ and $\langle s, 1, s\rangle \in \widehat{h}$. By Lemma 3.3.4, for a proper pair $(\widehat{g}, \widehat{h})$ the two-state test determines if $Z_{gh} \subseteq L(\mathcal{B})$. That is, $L(\mathcal{A})$ is not contained in $L(\mathcal{B})$ if and only if there is a proper pair of supergraphs that fails the two-state test.

We can phrase the two-state test as a set intersection problem. Let $(\widehat{g}, \widehat{h})$ be a pair of supergraphs. Define the set $R(\widehat{g}) = \{s \mid r \in Q^{in}_{\mathcal{B}},\ \langle r, a, s\rangle \in \widehat{g}\}$, and the

set $S(\widehat{h}) = \{s \mid \langle s, 1, s\rangle \in \widetilde{h}\}$. $(\widehat{g}, \widehat{h})$ passes the two-state test iff $R(\widehat{g}) \cap S(\widehat{h}) \neq \emptyset$. There are only 2^n subsets of $Q_{\mathcal{B}}$, a far smaller search space than the space of all supergraphs. Unfortunately, we cannot store only $R(\widehat{g})$ and $S(\widehat{h})$: we need to ensure that a counterexample pair is proper. Testing properness requires testing identity under composition.

Similar to Section 4.2, we weaken the notion of properness. By doing so we can store some graphs as sets of states and simplify our search space. Here we consider pairs of supergraphs that do not necessarily satisfy constraint 6, that $\widehat{g}; \widehat{h} = \widehat{g}$. Call a pair of supergraphs $(\widehat{g}, \widehat{h})$ *weakly proper* if it satisfies constraints 1-5. Given a weakly proper pair $(\widehat{g}, \widehat{h})$ of supergraphs, we can no longer limit our search to two states r and s with arcs $\langle r, a, s\rangle \in \widehat{g}$, $\langle s, 1, s\rangle \in \widehat{h}$. We need to search for a sequence of states, $t_0, ...t_n$ where $t_0 \in Q^{in}_{\mathcal{B}}$, with an arc $\langle t_0, a_0, t_1\rangle \in \widehat{g}$, for $0 < i < n$ arcs $\langle t_i, a_i, t_{i+1}\rangle \in \widehat{h}$, and the arc $\langle t_n, 1, t_n\rangle \in \widehat{h}$. Note that when $\widehat{h}; \widehat{h} = \widehat{h}$, t_1 must have an arc to every t_i, $i > 1$. In specific, there is an arc between t_1 and t_n in $\widehat{h}$. Thus it suffices to find only three states to fill the roles of t_0, t_1 and t_n.

Definition 4.3.1. *Let $(\widehat{g}, \widehat{h})$ be a pair of supergraphs. Say $(\widehat{g}, \widehat{h})$* passes the three-state test *iff there are three states $r, s, t \in Q_{\mathcal{B}}$ so that:*

(1) $r \in Q^{in}_{\mathcal{B}}$

(2) $\langle r, a, s\rangle \in \widehat{g}$

(3) $\langle s, a', t\rangle \in \widehat{h}$

(4) $\langle t, 1, t\rangle \in \widehat{h}$

Like the two-state test, the three-state test can be phrased as a set intersection problem. Given a supergraph $\widehat{h}$, let $S'(\widehat{h}) = \{s \mid \langle s, a, t\rangle \in \widehat{h},\ \langle t, 1, t\rangle \in \widehat{h}\}$. A pair $(\widehat{g}, \widehat{h})$ passes the three-state test iff $R(\widehat{g}) \cap S'(\widehat{h}) \neq \emptyset$. There are now no constraints on $\widetilde{g}$, and, so long as $p \in Q^{in}_{\mathcal{A}}$, we can represent the supergraph $\langle\langle p, q\rangle, \widetilde{g}\rangle$ as the pair of q and $R(\widehat{h})$.

Given a state $q \in Q_{\mathcal{A}}$ and set of states $R \subseteq Q_{\mathcal{B}}$, call the pair $[q, R]$ a *reachable subset* of $\mathcal{A}$ and $\mathcal{B}$ when there is a word $w \in \Sigma^*$ so that $q \in \rho_{\mathcal{A}}(Q^{in}_{\mathcal{A}}, w)$ and $R = \rho_{\mathcal{B}}(Q^{in}_{\mathcal{B}}, w)$. Let $P_{\mathcal{A},\mathcal{B}}$ be the set of reachable subsets. For each element $[q, R] \in P_{\mathcal{A},\mathcal{B}}$, there is a word w so that q is a state in $\mathcal{A}$ that can also be reached by w and R is the set of states in $\mathcal{B}$ which can be reached on w. Note the size of $P_{\mathcal{A},\mathcal{B}}$ is at most $m2^n$. We leverage the notion of reachable subsets to provide a containment-testing algorithm with a reduced search space.

Algorithm 4.3.2.

(1) Compute the set $P_{\mathcal{A},\mathcal{B}}$ of all reachable subsets:[2]

(a) Initialize $P_{\mathcal{A},\mathcal{B}}$ as $Q^{in}_{\mathcal{A}} \times \{Q^{in}_{\mathcal{B}}\}$

(b) For every $[q,R] \in P_{\mathcal{A},\mathcal{B}}$ and $\sigma \in \Sigma$, add every element of $\{[q', R'] \mid q' \in \rho_{\mathcal{A}}(q,\sigma),\ R' = \rho_{\mathcal{B}}(R,\sigma)\}$ to $P_{\mathcal{A},\mathcal{B}}$

(2) Compute $\widehat{Q}^1_{\mathcal{A},\mathcal{B}}$, the set of initial supergraphs

(3) Compute $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$ by taking closure of $\widehat{Q}^1_{\mathcal{A},\mathcal{B}}$ under composition

(4) When computing each $\langle\langle q, q\rangle, \widetilde{g}\rangle \in \widehat{Q}^f_{\mathcal{A},\mathcal{B}}$, if $q \in F_{\mathcal{A}}$ and $\widetilde{g};\widetilde{g} = \widetilde{g}$ then:

(a) Compute the set $S'(\widetilde{g}) = \{s \mid \langle s,a,t\rangle \in \widetilde{g},\ \langle t,1,t\rangle \in \widetilde{g}\}$

(b) Ensure that for every reachable subset of the form $[q,R] \in P_{\mathcal{A},\mathcal{B}}$, it holds that $R \cap S'(\widetilde{g}) \neq \emptyset$

In order to prove Algorithm 4.3.2 correct, we first connect weakly proper supergraphs and the three-state test to proper supergraphs and the two-state test.

Lemma 4.3.3. *Let $(\widehat{g},\widehat{h})$ be a weakly proper pair of supergraphs. If $(\widehat{g},\widehat{h})$ passes the three-state test, then $(\widehat{g};\widehat{h},\widehat{h})$ passes the two-state test.*

Proof: Assume we have a weakly proper pair $(\widehat{g},\widehat{h})$ that passes the three-state test. Note that, by constraints 1-2 and the definition of composition, $\widehat{g};\widehat{h}$ is well defined and equal to $\langle \bar{g}, \widetilde{g}; \widetilde{h}\rangle$. As $(\widehat{g},\widehat{h})$ passes the three-state test, there are three states $r \in Q^{in}_{\mathcal{B}}$, $s,t \in Q_{\mathcal{B}}$ so that $\langle r,a,s\rangle \in \widehat{g}$, $\langle s,a',t\rangle \in \widehat{h}$, and $\langle t,1,t\rangle \in \widehat{h}$. By the definition of composition $\langle r,1,t\rangle \in \widehat{g};\widehat{h}$. As $\langle t,1,t\rangle \in \widehat{h}$, $(\widehat{g};\widehat{h},\widehat{h})$ passes the two-state test. □

We now prove Algorithm 4.3.2 correct, demonstrating that $P_{\mathcal{A},\mathcal{B}}$ contains all reachable subsets corresponding to graphs in $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$.

Lemma 4.3.4. *Let $\mathcal{A},\mathcal{B}$ be two Büchi automata and $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$ the corresponding set of non-empty supergraphs. $L(\mathcal{A}) \not\subseteq L(\mathcal{B})$ iff $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$ contains a pair of supergraphs $(\widehat{g},\widehat{h})$ that is weakly proper and fails the three-state test.*

[2] *Note that we can also define a conservative approximation relation among reachable subsets. One reachable subset $[q,R]$ conservatively approximates another, $[q,R']$, when $R \subseteq R'$. For a given set S', if $R' \cap S' \neq \emptyset$, then certainly $R \cap S' \neq \emptyset$. Further, when computing the reachable subsets $P_{\mathcal{A},\mathcal{B}}$ we can safely remove reachable subsets that are conservatively approximated.*

Proof: In one direction, if $L(\mathcal{A}) \not\subseteq L(\mathcal{B})$, then by Lemma 3.3.4 there is a proper pair $(\widehat{g}, \widehat{h})$ of supergraphs in $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$ that fails the two-state test. Since the pair $(\widehat{g}, \widehat{h})$ is proper, $\widehat{g}$ is equal to $\widehat{g}; \widehat{h}$. Thus $(\widehat{g}; \widehat{h}, \widehat{h})$ fails the two-state test and by Lemma 4.3.3, $(\widehat{g}, \widehat{h})$ fails the three-state test.

In the other direction, assume $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$ contains a pair of supergraphs $(\widehat{g}, \widehat{h})$ that is weakly proper and fails the three-state test. Let $\widehat{g}' = \widehat{g}; \widehat{h}$. We show that the pair $(\widehat{g}', \widehat{h})$ is proper and fails the two-state test.

Since $\widehat{h} = \widehat{h}; \widehat{h}$, it holds that $\widehat{g}'; \widehat{h} = \widehat{g}'$. Since $(\widehat{g}, \widehat{h})$ are weakly proper, it holds that $\bar{g} = \bar{g}' = \langle p, q \rangle$, $p \in Q^{in}_{\mathcal{A}}$, $q \in F_{\mathcal{A}}$, and $\bar{h} = \langle q, q \rangle$. Thus, the pair $(\widehat{g}', \widehat{h})$ are proper. Further, as $(\widehat{g}, \widehat{h})$ fails the three-state test, there are no three states $r, s, t \in Q_{\mathcal{B}}$ so that $\langle r, a, s \rangle \in \widehat{g}$, $\langle s, a', t \rangle \in \widehat{h}$, and $\langle t, 1, t \rangle \in \widehat{h}$. By the definition of composition there are no two states $r \in Q^{in}_{\mathcal{B}}, t \in Q_{\mathcal{B}}$ so that $\langle r, a, t \rangle \in \widehat{g}'$, and $\langle t, 1, t \rangle \in \widehat{h}$. Therefore $(\widehat{g}', \widehat{h})$ fails the two-state test. By Lemma 3.3.4, we infer $L(\mathcal{A}) \not\subseteq L(\mathcal{B})$. □

Lemma 4.3.5. *Let $\mathcal{A}$ and $\mathcal{B}$ be two Büchi automata, $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$ the corresponding set of non-empty supergraphs, and $P_{\mathcal{A},\mathcal{B}}$ the reachable subsets of $\mathcal{A}$ and $\mathcal{B}$. $L(\mathcal{A}) \not\subseteq L(\mathcal{B})$ iff there is a reachable subset $[q, R] \in P_{\mathcal{A},\mathcal{B}}$ and a supergraph $\widehat{h} \in \widehat{Q}^f_{\mathcal{A},\mathcal{B}}$ such that $q \in F_{\mathcal{A}}$, $\bar{h} = \langle q, q \rangle$, $\widehat{h}; \widehat{h} = \widehat{h}$ and $R \cap S'(\widehat{h}) = \emptyset$.*

Proof: Assume there is such a reachable subset $[q, R]$ and supergraph $\widehat{h} \in \widehat{Q}^f_{\mathcal{A},\mathcal{B}}$, where $q \in F_{\mathcal{A}}$. There is an $w \in \Sigma^*$ and $p \in Q^{in}_{\mathcal{A}}$ so that $q \in \rho_{\mathcal{A}}(p, w)$ and $R = \rho_{\mathcal{B}}(Q^{in}_{\mathcal{B}}, w)$. By Lemma 2.1.1, w is in the language some graph $\widetilde{g}$. Therefore $w \in L(\widehat{g})$, where $\widehat{g} = \langle \langle p, q \rangle, \widetilde{g} \rangle$, and $R(\widehat{g}) = R$. As $p \in Q^{in}_{\mathcal{A}}$, $q \in F_{\mathcal{A}}$, and $\widehat{h}; \widehat{h} = \widehat{h}$, $(\widehat{g}, \widehat{h})$ are a weakly proper pair. Since $R(\widehat{g})$ and $S'(\widehat{h})$ are disjoint, $(\widehat{g}, \widehat{h})$ must fail the three-state test. By Lemma 4.3.4, $L(\mathcal{A}) \not\subseteq L(\mathcal{B})$.

Conversely, assume $L(\mathcal{A}) \not\subseteq L(\mathcal{B})$. Then, by Lemma 4.3.4, there is a pair of supergraphs $(\widehat{g}, \widehat{h}) \in \widehat{Q}^f_{\mathcal{A},\mathcal{B}}$ that is weakly proper and fail the three-state test. Let $\bar{g} = \langle p, q \rangle$. Since the two supergraphs are weakly proper, $p \in Q^{in}_{\mathcal{A}}$ and $q \in F_{\mathcal{A}}$. Further, as $L(\widehat{g})$ is non-empty, there is a word w that reaches both q in $\mathcal{A}$ and $R(\widehat{g})$ in $\mathcal{B}$. Thus $[q, R(\widehat{g})]$ is a reachable subset in $P_{\mathcal{A},\mathcal{B}}$. Since $\widehat{g}, \widehat{h}$ fail the three-state test, $R(\widehat{g}) \cap S'(\widehat{h}) = \emptyset$. □

4.4 Grounded Supergraphs: Single-Graph Containment Test for Arbitrary Languages

It would be convenient to apply the containment test and subsumption relation of Algorithm 4.2.2 to Büchi automata containment problems without strongly suffix-closed languages. To enable this, we construct a set of graphs that requires only the single-graph search to determine containment. Once we demonstrate that the single-graph search determines containment of arbitrary languages, we can use the

subsumption relation of Section 4.2 directly. The idea is to replace the arcs in supergraphs, which are pairs of states in $Q_{\mathcal{A}}$, with pairs of reachable subsets of $\mathcal{A}$ and $\mathcal{B}$. Whereas Lemma 4.3.5 computes separately the reachable subsets and supergraphs of an automaton, here we construct supergraphs that describe only paths that share the common prefix word of some reachable subset. Doing so comes at the expense of increasing the size of the closure set, but allows us to avoid storing the reachable subsets $P_{\mathcal{A},\mathcal{B}}$. While this does not improve the theoretical complexity of Lemma 4.3.5, the resulting algorithm is amenable to simpler implementation and the subsumption relation of Section 4.2

Define an extension of supergraphs called grounded supergraphs. In a grounded supergraph, we pair a graph $\widetilde{g}$ in $\widetilde{Q}_{\mathcal{B}}$ with a pair $\langle p[R], q[S]\rangle$ of reachable subsets in $P_{\mathcal{A},\mathcal{B}}$. Further, we restrict $\widetilde{g}$ to arcs between R and S. Formally, define the set of grounded supergraphs $\ddot{Q}_{\mathcal{A},\mathcal{B}}$ to be $\{\langle\langle [p,R],[q,S]\rangle, \widetilde{\widetilde{g}}\rangle \mid p,q \in Q_{\mathcal{A}},\ R,S \subseteq Q_{\mathcal{B}},\ \langle r,a,s\rangle \in \widetilde{g}$ only if $r \in R$ and $s \in S\}$. Given a grounded supergraph $\langle\langle [p,R],[q,S]\rangle, \widetilde{\widetilde{g}}\rangle$, call R the source nodes and S the sink nodes.

We offer an intuition of the meaning of grounded supergraphs. A graph in $\widetilde{Q}_{\mathcal{B}}$ encodes information about all paths in $\mathcal{B}$ over a set of words. Supergraphs pair an arc in $\bar{Q}_{\mathcal{A}}$ with a graph over $Q_{\mathcal{B}}$. A supergraph asserts the existence of a path in $\mathcal{A}$ over a set of words, and describes all corresponding paths in $\mathcal{B}$ over the same set of words. In contrast, grounded supergraphs contain a pair of reachable subsets of $\mathcal{A}$ and $\mathcal{B}$. Like a supergraph, a grounded supergraph asserts the existence of a path in $\mathcal{A}$ over a set of words, and describes some paths in $\mathcal{B}$ over the same set. Unlike supergraphs, a grounded supergraph also asserts that the path in $\mathcal{A}$ is reachable by some word, and only describes paths in $\mathcal{B}$ that share this prefix word.

Using grounded supergraphs we can employ the single-graph search to determine the containment of the languages of two arbitrary automata, $\mathcal{A}$ and $\mathcal{B}$. This algorithm proceeds in three steps. We start with a base set of grounded supergraphs $\ddot{Q}^0_{\mathcal{A},\mathcal{B}}$ covering transition out of start states. We then define an extension operation and proceed outwards to find $\ddot{Q}^1_{\mathcal{A},\mathcal{B}}$ by taking the iterative closure of $\ddot{Q}_{\mathcal{A},\mathcal{B}}$ under extension. The set $\ddot{Q}^1_{\mathcal{A},\mathcal{B}}$ is analogous to $\widehat{Q}^1_{\mathcal{A},\mathcal{B}}$ and describes all paths of length 1, i.e. all transitions in $\mathcal{A}$ and $\mathcal{B}$. Finally, we define composition and create $\ddot{Q}^f_{\mathcal{A},\mathcal{B}}$, the closure of $\ddot{Q}^1_{\mathcal{A},\mathcal{B}}$ under composition.

Recall that for each $a \in \Sigma$, $\widetilde{g}_a$ is the graph in $\widetilde{Q}_{\mathcal{B}}$ whose language contains a. It is useful to restrict a graph in $\widetilde{Q}_{\mathcal{B}}$ to arcs that lay between two reachable subsets. Given an graph $\widetilde{g} \in \widetilde{Q}_{\mathcal{B}}$ and two reachable subsets $[p,R],[q,S] \in P_{\mathcal{A},\mathcal{B}}$ let $\Pi_{([p,R],[q,S])}\widetilde{g}$ be the grounded supergraph $\langle\langle [p,R],[q,S]\rangle, \widetilde{g}'\rangle$ where $\widetilde{g}' = \{\langle r,a,s\rangle \mid \langle r,a,s\rangle \in \widetilde{g},\ r \in R, s \in S\}$.

Definition 4.4.1. *Given two Büchi automata $\mathcal{A}$ and $\mathcal{B}$, define $\ddot{Q}^0_{\mathcal{A},\mathcal{B}}$ to be $\{\Pi_{([p,Q^{in}_{\mathcal{B}}],[q,S])}\widetilde{g}_a \mid a \in \Sigma, p \in Q^{in}_{\mathcal{A}}, q \in \rho_{\mathcal{A}}(p,a), S = \rho_{\mathcal{B}}(Q^{in}_{\mathcal{B}},a)\}$.*

Now that we have $\ddot{Q}^0_{\mathcal{A},\mathcal{B}}$, we can construct $\ddot{Q}^1_{\mathcal{A},\mathcal{B}}$. Let $\ddot{g} = \langle\langle [q_1,R],[q_2,S]\rangle, \widetilde{g}\rangle$ be a grounded supergraph. Define the *extension of* $\ddot{g}$ to be the set of grounded supergraphs

$\{\Pi_{([q_2,S],[q_3,T])}\widetilde{g}_a \mid a \in \Sigma,\ q_3 \in \rho_{\mathcal{A}}(q_2,a),\ T = \rho_{\mathcal{B}}(S,a)\}$. This set is empty if q_2 has no outgoing transition. Let $\ddot{Q}^1_{\mathcal{A},\mathcal{B}}$ be fixed point obtained by taking the union of $\ddot{Q}^0_{\mathcal{A},\mathcal{B}}$ and the extension of all its elements. (Is it better to say: Let $\ddot{Q}^1_{\mathcal{A},\mathcal{B}}$ be the iterative closure of $\ddot{Q}^0_{\mathcal{A},\mathcal{B}}$ under extension?)

We have now obtained a set $\ddot{Q}^1_{\mathcal{A},\mathcal{B}}$ describing all transitions in $\mathcal{A}$ and the corresponding transitions in $\mathcal{B}$. To complete the construction of $\ddot{Q}^f_{\mathcal{A},\mathcal{B}}$, we define composition. Composition of grounded supergraphs is nearly identical to composition of supergraphs, save the type of object contained in the arcs. Let $\ddot{g} = \langle\langle[q_1,R],[q_2,S]\rangle,\widetilde{g}\rangle$ and $\ddot{h} = \langle\langle[q_2,S],[q_3,T]\rangle,\widetilde{h}\rangle$. Their composition $\ddot{g};\ddot{h}$ is the grounded supergraph $\langle\langle[q_1,R],[q_3,T]\rangle,\widetilde{g};\widetilde{h}\rangle$. Notice that composition of two supergraphs $\ddot{g},\ddot{h}$, requires that the second element of $\ddot{g}$'s arc be identical to the first element of $\ddot{h}$'s arc. This implies that $\ddot{g};\ddot{g}$ is only well-defined if $\ddot{g}$'s arc is reflexive. We now define a single-pass algorithm that uses subsumption to determine the language containment of two arbitrary automata.

Algorithm 4.4.2.

(1) Construct the set $\ddot{Q}^0_{\mathcal{A},\mathcal{B}}$ using Definition 4.4.1

(2) Construct the set $\ddot{Q}^1_{\mathcal{A},\mathcal{B}}$ by taking the closure of $\ddot{Q}^0_{\mathcal{A},\mathcal{B}}$ under extension

(3) Compute $\ddot{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$ using the procedure of Algorithm 4.2.1

(4) When computing each $\langle\langle q[R],q[R]\rangle,\widetilde{g}\rangle \in \widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$, if $q \in F_{\mathcal{A}}$ then:

- *Compute the strongly connected components of $\widetilde{g}$*
- *Ensure there exists a 1-labeled strongly connected component of $\widetilde{g}$*

To prove Algorithm 4.4.2 correct, and to capture this intuition about grounded supergraphs, we associate two languages of finite words with each grounded supergraph. The prefix language, $L^p(\langle\langle[p,R],[q,S]\rangle,\widetilde{g}\rangle) \subseteq \Sigma^*$, is the set of words w that reach every state in R in $\mathcal{B}$ and reach p in $\mathcal{A}$. The suffix language, $L^s(\langle\langle[p,R],[q,S]\rangle,\widetilde{g}\rangle) \subseteq \Sigma^+$ is the set of words y with a path in $\mathcal{A}$ from p to q, and whose paths from R to S are described by $\widetilde{g}$. Formally,

Definition 4.4.3. *Let $\langle\langle[p,R],[q,S]\rangle,\widetilde{g}\rangle$ be a grounded supergraph in $\ddot{Q}_{\mathcal{A},\mathcal{B}}$.*

(1) Let $w \in \Sigma^*$. Then $w \in L^p(\langle\langle[p,R],[q,S]\rangle,\widetilde{g}\rangle)$ when:

(a) $p \in \rho_{\mathcal{A}}(Q^{in}_{\mathcal{A}},w)$
(b) $R = \rho_{\mathcal{B}}(Q^{in}_{\mathcal{B}},w)$

(2) Let $y \in \Sigma^+$. Then $y \in L^s(\langle\langle[p,R],[q,S]\rangle,\widetilde{g}\rangle)$ when:

(a) $q \in \rho_{\mathcal{A}}(p,y)$

(b) $S = \rho_{\mathcal{B}}(R, y)$

(c) For every $r \in R, s \in S,\ a \in \{0, 1\}$

i. $\langle r, a, s\rangle \in \widetilde{g}$ *iff there is a path in* $\mathcal{B}$ *from* r *to* s *over* y

ii. $\langle r, 1, s\rangle \in \widetilde{g}$ *iff there is an accepting path in* $\mathcal{B}$ *from* r *to* s *over* y

Lemma 4.4.4. *Let* $\langle\langle [p, R], [q, S]\rangle, \widetilde{g}\rangle$ *and* $\langle\langle [p, R], [q, S]\rangle, \widetilde{h}\rangle$ *be two grounded supergraphs. If* $L^s(\langle\langle [p, R], [q, S]\rangle, \widetilde{g}\rangle) \cap L^s(\langle\langle [p, R], [q, S]\rangle, \widetilde{h}\rangle) \neq \emptyset$ *then* $\widetilde{g} = \widetilde{h}$.

Proof: As the arc $\langle [p, R], [q, S]\rangle$ is the same in both grounded supergraphs, only arcs between R and S can occur in $\widetilde{g}$ or $\widetilde{h}$. Definition 4.4.3 completely specifies which arcs between R to S must occur in $\widetilde{g}$ and $\widetilde{h}$. □

First we relate the languages of graphs to the language of supergraphs. Recall that given a graph $\widetilde{g} \in \widetilde{Q}_{\mathcal{B}}$ and two reachable subsets $[p, R], [q, s] \in P_{\mathcal{A},\mathcal{B}}$, we define $\Pi_{([p,R],[q,S])}\widetilde{g}$ to be the grounded supergraph $\langle\langle [p, R], [q, S]\rangle, \widetilde{g}'\rangle$ where $\widetilde{g}' = \{\langle r, a, s\rangle \mid \langle r, a, s\rangle \in \widetilde{g},\ r \in R, s \in S\}$.

Lemma 4.4.5. *Let* $\widetilde{g} \in \widetilde{Q}_{\mathcal{B}}$, $[p, R]$, $[q, s] \in P_{\mathcal{A},\mathcal{B}}$, *and* $w \in L(\widetilde{g})$. *If* $q \in \rho_{\mathcal{A}}(p, w)$ *and* $S = \rho_{\mathcal{B}}(R, w)$, *then* $w \in L^s(\Pi_{([p,R],[q,S])}\widetilde{g})$.

Proof: Note that by Definition 2.1.1, if the arc $\langle r, a, s\rangle$ is in $\widehat{g}$ and $r \in R$, it must be the case that $s \in S$.

For every two states $r, s \in Q_{\mathcal{B}}$, by construction the arc $\langle r, a, s\rangle \in \Pi_{([p,R],[q,S])}\widetilde{g}$ iff $r \in R$ and $\langle r, a, s\rangle \in \widetilde{g}$. By Definition 2.1.1, $\langle r, a, s\rangle \in \widetilde{g}$ iff there is a path in $\mathcal{B}$ from r to s on w. Similarly, $\langle r, 1, s\rangle \in \Pi_{([p,R],[q,S])}\widetilde{g}$ iff $\langle r, 1, s\rangle \in \widetilde{g}$, which is the case iff there is an accepting path from r to s on w. By Definition 4.4.3, this implies $w \in L^s(\Pi_{([p,R],[q,S])}\widetilde{g})$. □

The following lemma shows how the suffix languages of grounded supergraphs are related by concatenation.

Lemma 4.4.6. *Let* $\ddot{g} = \langle\langle [q_1, R], [q_2, S]\rangle, \widetilde{g}\rangle$, $\ddot{h} = \langle\langle [q_2, S], [q_3, T]\rangle, \widetilde{h}\rangle$, *and* $\ddot{k} = \langle\langle [q_1, R], [q_3, T]\rangle, \widetilde{k}\rangle$ *be three grounded supergraphs in* $\ddot{Q}_{\mathcal{A},\mathcal{B}}$, *and* $x, y \in \Sigma^+$. *If* $x \in L^s(\ddot{g})$, $y \in L^s(\ddot{h})$, *and* $xy \in L^s(\ddot{k})$, *then* $L^s(\ddot{g}) \cdot L^s(\ddot{h}) \subseteq L^s(\ddot{k})$

Proof: By assumption there exists an $x, y \in \Sigma^+$, $x \in L^s(\ddot{g})$, $y \in L^s(\ddot{h})$ and $xy \in L^s(\ddot{k})$.

We first observe that arcs in $\ddot{k}$ have matching pairs of arcs in $\ddot{g}$ and $\ddot{h}$. To show this, note that for every two states $r \in R, t \in T$, the arc $\langle r, a, t\rangle \in \ddot{k}$ iff there is a path in $\mathcal{B}$ from r to t on xy. By the definition of a path, this is the case iff there is an $s \in S$ so that there is a path from r to s on x, and a path from s to t on y. These paths exist iff there are arcs $\langle r, a', s\rangle \in \ddot{g}$ and $\langle s, a'', t\rangle \in \ddot{h}$.

Similarly, we know a is 1 iff there is an accepting path from r to t on xy. There is an accepting path iff there is a state $s \in c$ so that there is a path from r to s, a path from s to t, and at least one of the paths is accepting. These paths exist exactly when there are arcs $\langle r, a', s\rangle \in \ddot{g}$ and $\langle s, a'', t\rangle \in \ddot{h}$, and at least one of a' and a'' is 1. We can therefore conclude that an arc $\langle r, a, t\rangle \in \ddot{k}$ iff there is an $s \in S$, and arcs $\langle r, a', s\rangle \in \ddot{g}$ and $\langle s, a'', t\rangle \in \ddot{h}$. Further, a is 1 iff there is an s so that a' or a'' is 1.

We use this observation to show that $L^s(\ddot{g}) \cdot L^s(\ddot{h}) \subseteq L^s(\ddot{k})$. Let $x' \in L^s(\ddot{g})$, $y' \in L^s(\ddot{h})$. We show $x'y' \in L^s(\ddot{k})$. By Definition 4.4.3, $S = \rho_{\mathcal{B}}(R, x')$ and $T = \rho_{\mathcal{B}}(S, y')$, and so $T = \rho_{\mathcal{B}}(R, x'y')$. For every two states $r \in R, t \in T$, there is a path in $\mathcal{B}$ from r to t on $x'y'$ iff there is a state $s \in S$, a path from r to s on x', and a path from s to t on y'. This holds iff there are arcs $\langle r, a', s\rangle \in \ddot{g}$ and $\langle s, a'', t\rangle \in \ddot{h}$. By the above observation, this is the case exactly when there is an arc $\langle r, a, t\rangle \in \ddot{k}$. Similarly, an accepting path implies and is implied by the arc $\langle r, 1, t\rangle \in \ddot{k}$. Therefore $x'y' \in L(\ddot{k})$. □

In analogy to to Section 4.3, say that a grounded supergraph $\ddot{g} = \langle\langle [p, R], [p, R]\rangle, \widetilde{g}\rangle$ is *weakly proper* when $p \in F_{\mathcal{A}}$, $L^p(\ddot{g}) \neq \emptyset$, $L^s(\ddot{g}) \neq \emptyset$, and $L^s(\ddot{g}) \cdot L^s(\ddot{g}) \subseteq L^s(\ddot{g})$. For each $\ddot{g} \in \ddot{Q}_{\mathcal{A},\mathcal{B}}$, let the ω-language $L^\omega(\ddot{g})$ be $L^p(\ddot{g}) \cdot L^s(\ddot{g})^\omega$. The ω-languages of $\ddot{Q}_{\mathcal{A},\mathcal{B}}$ cover $L(\mathcal{A})$.

Lemma 4.4.7.

(1) $L(\mathcal{A}) = \bigcup\{L^\omega(\ddot{g}) \mid \ddot{g} \in \ddot{Q}_{\mathcal{A},\mathcal{B}},\ \ddot{g}$ *is weakly proper*$\}$

(2) For all weakly proper grounded supergraphs $\ddot{g}$*, either* $L^\omega(\ddot{g}) \cap L(\mathcal{B}) = \emptyset$*, or* $L^\omega(\ddot{g}) \subseteq L(\mathcal{B})$

(3) $L(\mathcal{A}) \cap \overline{L(\mathcal{B})} = \bigcup\{L^\omega(\ddot{g}) \mid \ddot{g}$ *is weakly proper and* $L^\omega(\ddot{g}) \cap L(\mathcal{B}) = \emptyset\}$.

Proof:
(1) First, for a weakly proper grounded supergraph the state p in $\mathcal{A}$ is reachable and an accepting state. Thus there is an accepting path of $\mathcal{A}$ on every word in $L^\omega(\ddot{g})$.

We extend the Ramsey-based argument of Lemma 3.3.3 to grounded supergraphs. Recall that there are $k = |Q_{\mathcal{A}}|^2 3^{|Q_{\mathcal{B}}|^2}$ supergraphs in $\widehat{Q}_{\mathcal{A},\mathcal{B}}$. Let $\widehat{g}_1 ... \widehat{g}_k$ be an ordering of these.

Take an infinite word $w = a_0 a_1 ...$ with an accepting run in $\mathcal{A}$ $p = p_0 p_1 ....$ The argument for Lemma 3.3.3 observes that some accepting state q must occur infinitely often, and defines a subset of indexes i where $p_i = q$.

Consider the sequence $R = R_0 R_1 ...$ of subsets of $Q_{\mathcal{B}}$, where where $R_0 = Q_{\mathcal{B}}^{in}$ and $R_{i+1} = \rho_{\mathcal{B}}(R_i, a_i)$. We similarly observe that some subset of $Q_{\mathcal{B}}$ must occur infinitely often. Since $F_{\mathcal{A}}$ and $2^{Q_{\mathcal{B}}}$ are finite, there is an accepting state $q \in F_{\mathcal{A}}$ and a set $R_a \subseteq Q_{\mathcal{B}}$ so that $p_i = q$ and $R_i = R_a$ for infinitely many i. Let C be the set of such indexes i.

By Lemma 2.1.4, there is a subset $D \subseteq C$ and a graph $\widetilde{h}$ so that $a_i...a_{j-1} \in L(\widetilde{h})$ for every $i, j \in D$, $i < j$. D partitions the word w into $w_1 w_2...$ so that $w_i \in L(\widetilde{h})$ for $i > 1$.

Let $\ddot{h}$ be $\Pi_{([q,R_a],[q,R_a])}\widetilde{h}$, the restriction of $\widetilde{h}$ to the states in R_a. We now show that $w \in L^\omega(\ddot{h})$ and that $\ddot{h}$ is weakly proper. First, as $\rho_{\mathcal{B}}(Q^{in}, w_1) = R_a$ and $p_i = q$, by Lemma 4.4.5 the word $w_1 \in L^p(\ddot{h})$. Secondly note that as for every $i > 1$, $w_i \in L(\widetilde{h})$, $R_a = \rho_{\mathcal{B}}(R_a, w_i)$ and $p_i = q$, by Definition 4.4.3 and Lemma 4.4.5 it holds that for every $i > 1$ $w_i \in L^s(\ddot{h})$. Since $w_1 \in L^p(\ddot{h})$ and for every $i > 1$, $w_i \in L^s(\ddot{h})$, it holds that $w \in L^\omega(\ddot{h})$.

Finally, as $a_i...a_j \in L(\widetilde{h})$ for $i, j \in D'$, we have w_2, w_3, $w_2 w_3 \in L(\widetilde{h})$. By Lemma 4.4.6 this implies $L^s(\ddot{h}) \cdot L^s(\ddot{h}) \subseteq L^s(\ddot{h})$, and $\ddot{h}$ is weakly proper.

(2) Let $\ddot{g}$ be a grounded supergraph $\langle\langle [q,R],[q,R]\rangle, \widetilde{g}\rangle$. We prove that if one word in $L^\omega(\ddot{g})$ is in $L(\mathcal{B})$, then every word in $L^\omega(\ddot{g})$ is in $L(\mathcal{B})$. A word $w \in L^\omega(\ddot{g})$ can be decomposed into $w_1 w_2 w_3...$ where $w_1 \in L^p(\ddot{g})$ and, for every $i > 1$, $w_i \in L^s(\ddot{g})$. If $w \in L(\mathcal{B})$, then there is an accepting run p. Let r_i be the state of $\mathcal{B}$ in this accepting run after reading $w_1...w_i$. Since $w_1 \in L^p(\langle\langle [q,R],[q,R]\rangle, \widetilde{g}\rangle)$, we know $r_1 \in R$. Since every w_i, $i > 1$, is in $L^s(\langle\langle [q,R],[q,R]\rangle, \widetilde{g}\rangle)$, we know every $r_i \in R$. If r is an accepting run, then there is an accepting state in the path between infinitely many r_i and r_{i+1}.

Any other word $w' \in L^\omega(\ddot{g})$ can be similarly decomposed into $w'_1 w'_2...$ where $w'_1 \in L^p(\ddot{g})$ and $w'_i \in L^s(\ddot{g})$ for every $i > 1$. By Definition 4.4.3, there is a path from an initial state to p_1 over w'_1 in $\mathcal{B}$. Similarly, there is a path between p_i and p_{i+1} over every w_{i+1}, and there is a path with an accepting state over w'_i when there is a path with an accepting state over w_i. Infinitely many such paths over substrings of w contain an accepting state, and w' is accepted by $\mathcal{B}$.

(3) Immediate from (1) and (2). □

Lemma 4.4.8. *Given Büchi automata $\mathcal{A}$ and $\mathcal{B}$ and the set of grounded supergraphs $\ddot{Q}_{\mathcal{A},\mathcal{B}}$,*

(1) $L(\mathcal{A}) \subseteq L(\mathcal{B})$ iff, for every weakly proper grounded supergraph $\ddot{g}$, $L^\omega(\ddot{g}) \subseteq L(\mathcal{B})$.

(2) For every weakly proper $\ddot{g} \in \ddot{Q}_{\mathcal{A},\mathcal{B}}$, $L^\omega(\ddot{g}) \subseteq L(\mathcal{B})$ iff there exists an arc $\langle r, 1, r\rangle \in \ddot{g}$.

Proof: (1) Immediate from Lemma 4.4.7.

(2) Take let $\ddot{g} = \langle\langle [q,R],[q,R]\rangle, \widetilde{g}\rangle$ and $w \in L^\omega(\ddot{g})$. By the definition of L^ω, w can be broken up into $w_0 w_1 w_2...$, where $w_0 \in L^p(\ddot{g})$ and $w_i \in L^s(\ddot{g})$ for $i > 0$. In one direction, assume an arc $\langle r, 1, r\rangle \in \ddot{g}$. By the definition of grounded supergraphs, $r \in R$. As $w_0 \in L^p(\ddot{g})$, Definition 4.4.3 implies there is a path in $\mathcal{B}$ from an initial state to r on w_0. Further, as $w_i \in L^s(\ddot{g})$ for $i > 0$, for $i > 0$ there

is an accepting path in $\mathcal{B}$ from r to r on w_i. Concatenating these paths creates a run of $\mathcal{B}$ on w that contains infinitely many accepting states, and therefor $w \in L(\mathcal{B})$.

In the other direction, assume $w \in L(\mathcal{B})$. This implies there is an accepting run $q = q_0 q_1 ...$ of $\mathcal{B}$ on w. Let r_i be the state this run is in after read $w_0 ... w_i$. By Definition 4.4.3 each $r_i \in R$. Clearly some $r \in R$ occurs infinitely often. As there are infinitely many accepting states, this implies there is an accepting state on a path between $r_i = r$ and $r_j = r$. This path is a path in $\mathcal{B}$ on $w_{i+1} ... w_j$. As $\ddot{g}$ is weakly proper, $L^s(\ddot{g}) \cdot L^s(\ddot{g}) \subseteq L^s(\ddot{g})$. Therefor $w_{i+1} ... w_j \in L^s(\ddot{g})$. By Definition 4.4.3, the arc $\langle r, 1, r \rangle \in \ddot{g}$.

□

We now prove that $\ddot{Q}^f_{\mathcal{A},\mathcal{B}}$ is the set of all grounded supergraphs with non-empty prefix and suffix languages. Recall we began with a base set of grounded supergraphs $\ddot{Q}^0_{\mathcal{A},\mathcal{B}}$ covering transition out of start states. We first prove $\ddot{Q}^0_{\mathcal{A},\mathcal{B}}$ describes grounded supergraphs where the prefix language contains the empty string and the suffix language a single letter.

Lemma 4.4.9. *For every $\ddot{g} \in \ddot{Q}_{\mathcal{A},\mathcal{B}}$, if there exists $a \in \Sigma$ so that $\epsilon \in L^p(\ddot{g})$ and $a \in L^s(\ddot{g})$, then $\ddot{g} \in \ddot{Q}^0_{\mathcal{A},\mathcal{B}}$.*

Proof: Let $\ddot{g} = \langle \langle [p, R], [q, S] \rangle, \widetilde{g} \rangle$. By Definition 4.4.3, since $\epsilon \in L^p(\ddot{g})$, it holds that $p \in Q^{in}_{\mathcal{A}}$ and $R = Q^{in}_{\mathcal{B}}$. Further, since $a \in L^s(\ddot{g})$ it holds that $q \in \rho_{\mathcal{A}}(p, a)$ and $S = \rho_{\mathcal{B}}(R, a)$. Therefor by Definition 4.4.1 the grounded supergraph $\Pi_{([p,R],[q,S])}\widetilde{g}_a)$ is in $\ddot{Q}^0_{\mathcal{A},\mathcal{B}}$. By Lemma 4.4.5 $a \in L^s(\Pi_{([p,Q^{in}_{\mathcal{B}}],[q,S])}\widetilde{g}_a)$. Thus, by Lemma 4.4.4 $\Pi_{([p,Q^{in}_{\mathcal{B}}],[q,S])}\widetilde{g}_a) = \ddot{g}$. □

Above, the extension operation is used to define the set $\ddot{Q}^1_{\mathcal{A},\mathcal{B}}$, which we show to be the set of all grounded supergraphs with a suffix language of a single letter.

Lemma 4.4.10. *For every $\ddot{g} \in \ddot{Q}_{\mathcal{A},\mathcal{B}}$, if there exists an $w \in \Sigma^*$ and $a \in \Sigma$ so that $w \in L^p(\ddot{g})$ and $a \in L^s(\ddot{g})$, then $\ddot{g} \in \ddot{Q}^1_{\mathcal{A},\mathcal{B}}$.*

Proof: We prove this lemma by induction on the length of w. As a base case, by Lemma 4.4.9 all graphs $\ddot{g}$ where $w = \epsilon$ are in $\ddot{Q}^0_{\mathcal{A},\mathcal{B}}$.

For the inductive step, let $w = w_0 ... w_{n-1}$ be a word in Σ^n, a a letter in Σ, and $\ddot{g} \in \ddot{Q}_{\mathcal{A},\mathcal{B}}$ a supergraph where $w \in L^p(\ddot{g})$ and $a \in L^s(\ddot{g})$. Let $\ddot{g} = \langle \langle [q_2, S], [q_3, T] \rangle, \widetilde{g} \rangle$. Since w is in the prefix language of $\ddot{g}$, it holds that there is a path from some initial state in $Q^{in}_{\mathcal{A}}$ to q_2 over w. Given a this path to q_2, after reading $w_0 ... w_{n-2}$, we must be in some state q_1. Define the set R to be $\rho_{\mathcal{B}}(Q^{in}_{\mathcal{B}}, w_0 ... w_{n_2})$. We now consider the grounded supergraph $\ddot{h} = \Pi_{([q_1,R],[q_2,S])}\widetilde{g}_{w_{n-1}}$ By the above observations, $w_0 ... w_{n-2} \in L^p(\ddot{h})$. By definition of q_1, $q_2 \in \rho_{\mathcal{A}}(q_1, w_{n-1})$. Similarly, $S = \rho_{\mathcal{B}}(R, w_{n-1})$. Therefor by Lemma 4.4.5, $w_{n-1} \in L^s(\ddot{h})$. By the inductive hypothesis, $\ddot{h} \in \ddot{Q}^1_{\mathcal{A},\mathcal{B}}$.

We now show that that $\ddot{g}$ is the extension of $\ddot{h}$ by a, and thus in $\ddot{Q}^1_{\mathcal{A},\mathcal{B}}$. Note that as $q_3 \in \rho_{\mathcal{A}}(q_2, a)$ and $T = \rho_{\mathcal{B}}(S, a)$, $\Pi_{([q_2,S],[q_3,T])}\widetilde{g}_a$ is in the extension of $\widehat{g}$ by a. By Lemma 4.4.5, $a \in L(\Pi_{([q_2,S],[q_3,T])}\widetilde{g}_a)$. Since a is also in $L^s(\ddot{g})$, by Lemma 4.4.4 $\Pi_{([q_2,S],[q_3,T])}\widetilde{g}_a = \ddot{g}$ and $\ddot{g}$ is in the extension of $\ddot{h}$ by a. □

Next, we demonstrate that composition is strongly related to language concatenation.

Lemma 4.4.11. *Let* $\ddot{g} = \langle\langle[q_1, R], [q_2, S]\rangle, \widetilde{g}\rangle, \ddot{h} = \langle\langle[q_2, S], [q_3, T]\rangle, \widetilde{h}\rangle$, *and* $\ddot{k} = \langle\langle[q_2, S], [q_3, T]\rangle, \widetilde{k}\rangle$ *be three grounded supergraphs in* $\ddot{Q}_{\mathcal{A},\mathcal{B}}$ *with non-empty prefix and suffix languages.*

(1) $L^p(\ddot{g}) = L^p(\ddot{g}; \ddot{h})$

(2) $L^s(\ddot{g}) \cdot L^s(\ddot{h}) \subseteq L^s(\ddot{g}; \ddot{h})$

(3) $L^s(\ddot{g}) \cdot L^s(\ddot{h}) \subseteq L^s(\ddot{k})$ *iff* $\ddot{g}; \ddot{h} = \ddot{k}$

Proof:
(1) Immediate from Definition 4.4.3.

(2) Given $x \in L^s(\ddot{g})$ and $y \in L^s(\ddot{h})$. we show $xy \in L^s(\langle\langle[q_1, R], [q_2, S]\rangle, \widetilde{g}\rangle; \langle\langle[q_2, S], [q_3, T]\rangle, \widetilde{h}\rangle)$, By definition, $\langle\langle[q_1, R], [q_2, S]\rangle, \widetilde{g}\rangle; \langle\langle[q_2, S], [q_3, T]\rangle, \widetilde{h}\rangle = \langle\langle[q_1, R], [q_3, T]\rangle, \widetilde{g}; \widetilde{h}\rangle$. So to prove $xy \in L^s(\langle\langle[q_1, R], [q_2, S]\rangle, \widetilde{g}\rangle; \langle\langle[q_2, S], [q_3, T]\rangle, \widetilde{h}\rangle)$, we first prove that for every $r, t \in Q_{\mathcal{B}}$, an arc $\langle r, a, t\rangle \in \widetilde{g}; \widetilde{h}$ iff $r \in R$ and there is a path in $\mathcal{B}$ from r to t over xy. Secondly, we show that that $a = 1$ iff there is an accepting path. Finally, we show that $T = \rho_{\mathcal{B}}(R, xy)$ and note that $q_3 \in \rho_{\mathcal{A}}(q_1, xy)$. By Definition 4.4.3, this shows that $xy \in L^s(\langle\langle[q_1, R], [q_2, S]\rangle, \widetilde{g}\rangle; \langle\langle[q_2, S], [q_3, T]\rangle, \widetilde{h}\rangle)$.

First, let r, t be two states in Q. If $r \notin R$, then no arc from r can be in $\widetilde{g}$, and $\langle q, a, t\rangle \notin \widetilde{g}; \widetilde{h}$. So assume $r \in R$. By the definition of a path, there is a path from r to t over xy iff there exists $s \in Q$, a path from r to s over x, and a path from s to t over y. In this case, since $S = \rho_{\mathcal{B}}(R, x)$, $s \in S$. Since $x \in L^s(\ddot{g})$, Definition 4.4.3 implies there is a path from r to s over x iff $\langle r, a', s\rangle \in \widetilde{g}$. Similarly there is a path from s to t over y iff $\langle s, a'', t\rangle \in \widetilde{h}$. By the definition of composition, $\langle r, a, t\rangle \in \widetilde{g}; \widetilde{h}$ exactly when there exists an $s \in Q_{\mathcal{B}}$ so that $\langle r, a', s\rangle \in \widetilde{g}$ and $\langle s, a'', t\rangle \in \widetilde{h}$. Therefore there is a path from r to t over xy iff $\langle r, a, t\rangle \in \widetilde{g}; \widetilde{h}$.

Secondly, $a = 1$ iff there is an $s \in Q$ so that $\langle r, 1, s\rangle \in \widetilde{g}$ or $\langle s, 1, t\rangle \in \widetilde{h}$. This is the case iff there is an accepting path from r to s over x or an accepting path from s to t over y. There is an accepting path from r to t over xy precisely when such an s exists. Therefore $a = 1$ if there is an accepting path.

Finally, we know that T is the correct sink. Since $x \in L^s(\ddot{g})$, we know that $S = \rho(R, x)$. Thus every state reachable from R on x is in S. Similarly, every state reachable from S on y is in T. Therefore $T = \rho(R, xy)$.

(3) Assume $L^s(\ddot{g}) \cdot L^s(\ddot{h}) \subseteq L^s(\ddot{k})$. By hypothesis there is a word $x \in L^s(\ddot{g})$ and a

word $y \in L^s(\ddot{h})$. By clause (2) above, we know that $xy \in L^s(\widetilde{\langle\langle[q_1, R], [q_3, T]\rangle, g; h\rangle})$. By assumption, $xy \in L^s(\ddot{k})$. By Lemma 4.4.4, $\widetilde{\langle\langle[q_1, R], [q_3, T]\rangle, g; h\rangle} = \ddot{k}$.

In the other direction, assume $\langle\langle[q_1, R], [q_2, S]\rangle, \widetilde{g}\rangle; \langle\langle[q_2, S], [q_3, T]\rangle, \widetilde{h}\rangle = \langle\langle[q_1, R], [q_3, T]\rangle, \widetilde{k}\rangle$. Then by clause (2) above, every word $x \in L^s(\ddot{g}) \cdot L^s(\ddot{h})$ is also in $L^s(\ddot{k})$.

□

Recall that $\ddot{Q}^f_{\mathcal{A},\mathcal{B}}$ is defined to be the closure of $\ddot{Q}^1_{\mathcal{A},\mathcal{B}}$ under composition. We now prove that $\ddot{Q}^f_{\mathcal{A},\mathcal{B}}$ contains every grounded supergraph with non-empty prefix and suffix languages, and thus every weakly proper grounded supergraph.

Lemma 4.4.12. *For every $\ddot{g}$ with non-empty prefix and suffix languages, $\ddot{g} \in \ddot{Q}^f_{\mathcal{A},\mathcal{B}}$.*

Proof: Let $\ddot{g}$ be a grounded supergraph with a word $x \in L^p(\ddot{g})$ and $y \in L^s(\ddot{g})$. Let $y = y_1...y_n$, and $\ddot{g} = \langle\langle[p, R], [q, S]\rangle, \widetilde{h}\rangle$. By Definition 4.4.3 there is a path in $\mathcal{A}$, $p_1p_2p_3...p_n$ from p to q on y. Similarly, let $R_1R_2R_3...R_n$ be the sequence of sets of states of $\mathcal{B}$, $R_i \subseteq Q_{\mathcal{B}}$, so that $R_1 = R$ and $R_{i+1} = \rho_{\mathcal{B}}(R_i, y_{i-1})$.

By Definition 4.4.3, for every $1 \leq i \leq n$ there is a grounded supergraph $\ddot{g}_i = \langle\langle[p_i, R_i], [p_{i+1}, R_{i+1}]\rangle, \widetilde{g}_i\rangle$ so that $xy_1...y_{i-1} \in L^p(\ddot{g}_i)$ and the character $y_i \in L^s(\ddot{g}_i)$.

By Lemma 4.4.10, we know these graphs are in H^1. Let $\ddot{g}_y$ be $\ddot{g}_{y_1}; ...; \ddot{g}_{y_n}$. By definition, the source nodes of $\ddot{g}_i$ are the vertices reachable from a start state on $xy_1...y_{i-1}$. The sink nodes of $\ddot{g}_i$ are all states reachable from the source nodes on y_i. These are the source nodes of $\ddot{g}_{i+1}$. Therefore the composition $\ddot{g}_i; \ddot{g}_{i+1}$ is well defined.

By the first clause of Lemma 4.4.11, $x \in L^p(\ddot{g}_y)$. By the second clause, $y \in L^s(\ddot{g}_y)$. Since $\ddot{Q}^f_{\mathcal{A},\mathcal{B}}$ is closed under composition, $\ddot{g}_y \in \ddot{Q}^f_{\mathcal{A},\mathcal{B}}$. Since $y \in L^s(\ddot{g}_y)$ and $y \in L^s(\ddot{g})$, by Lemma 4.4.4 $\ddot{g}_y = \ddot{g}$ and $\ddot{g} \in \ddot{Q}^f_{\mathcal{A},\mathcal{B}}$. □

Lemma 4.4.12 implies that every grounded supergraph with a non-empty prefix and suffix language is in $\ddot{Q}^f_{\mathcal{A},\mathcal{B}}$. Thus every weakly proper grounded supergraph is in $\ddot{Q}^f_{\mathcal{A},\mathcal{B}}$. By Lemma 4.4.8, we can verify $L(\mathcal{A}) \subseteq L(\mathcal{B})$ by searching $\ddot{Q}^f_{\mathcal{A},\mathcal{B}}$ for a single $\langle\langle[q, R], [q, R]\rangle, \widetilde{g}\rangle$, $\widetilde{g}; \widetilde{g} = \widetilde{g}$ that does not contain an arc $\langle r, 1, r\rangle$.

Lemma 4.4.13. *Let $\mathcal{A}$ and $\mathcal{B}$ be two Büchi automaton and $\ddot{Q}^f_{\mathcal{A},\mathcal{B}}$ the corresponding set of grounded supergraphs. $L(\mathcal{A})$ is* not *contained in $L(\mathcal{B})$ iff there is $\langle\langle[q, R], [q, R]\rangle, \widetilde{g}\rangle \in \widetilde{Q}^f_{\mathcal{A},\mathcal{B}}$ so that $q \in F_{\mathcal{A}}$, $\widetilde{g}; \widetilde{g} = \widetilde{g}$, and there is no arc of the form $\langle r, 1, r\rangle \in \widetilde{g}$.*

Proof: By the third clause of Lemma 4.4.11, $\widetilde{g}; \widetilde{g} = \widetilde{g}$ implies $L^s(\langle\langle[q, R], [q, R]\rangle, \widetilde{g}\rangle) \cdot L^s(\langle\langle[q, R], [q, R]\rangle, \widetilde{g}\rangle) \subseteq L^s(\langle\langle[q, R], [q, R]\rangle, \widetilde{g}\rangle)$. The rest is immediate from Definition 4.4.3 and Lemmas 4.4.8 and 4.4.12. □

After $\ddot{Q}^1_{\mathcal{A},\mathcal{B}}$ has been computed, the algorithm suggested by Lemma 4.4.13 is analogous to the single-graph search proposed by Lemma 3.5.2. Lemma 3.5.2 takes a set $\widehat{Q}^1_{\mathcal{A},\mathcal{B}}$ and determines containment as follows. First, compute $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$, the closure of $\widehat{Q}^1_{\mathcal{A},\mathcal{B}}$

under composition. Second, search $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$ for an idempotent supergraph $\widehat{g} = \langle \langle s, s \rangle, \widetilde{g} \rangle$ where $s \in F_{\mathcal{A}}$ that contains no arc of the form $\langle q, 1, q \rangle$.

In Section 4.2 we defined a subsumption relation on supergraphs, so that $\langle \bar{g}, \widetilde{g} \rangle \preceq \langle \bar{h}, \widetilde{h} \rangle$ implies $\bar{g} = \bar{h}$. We then showed how to determine containment by computing a subset $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$ of $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$ using Algorithm 4.2.1, and then searching $\widehat{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$ for a supergraph with a reflexive arc, $\widehat{g} = \langle \langle s, s \rangle, \widetilde{g} \rangle$, where $s \in F_{\mathcal{A}}$ that contains no 1-labeled strongly connected component.

Analogously, Lemma 4.4.2 takes a set $\ddot{Q}^1_{\mathcal{A},\mathcal{B}}$ and determines containment as follows. First, compute $\ddot{Q}^f_{\mathcal{A},\mathcal{B}}$, the closure of $\ddot{Q}^1_{\mathcal{A},\mathcal{B}}$ under composition. Second, search $\ddot{Q}^f_{\mathcal{A},\mathcal{B}}$ for an idempotent grounded supergraph $\ddot{g} = \langle \langle [q, R], [q, R] \rangle, \widetilde{g} \rangle$ where $[q, R] \in F_{\mathcal{A}} \times 2^{Q_{\mathcal{B}}}$ that contains no arc of the form $\langle q, 1, q \rangle$. We can thus generalize the subsumption relation on supergraphs to grounded supergraphs and determine containment by computing a subset $\ddot{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$ of $\ddot{Q}^f_{\mathcal{A},\mathcal{B}}$ using Algorithm 4.2.1, and then searching $\ddot{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$ for a grounded supergraph with a reflexive arc, $\widehat{g} = \langle \langle q[R], q[R] \rangle, \widetilde{g} \rangle$, where $[q, R] \in F_{\mathcal{A}} \times 2^{Q_{\mathcal{B}}}$ that contains no 1-labeled strongly connected component.

Lemma 4.4.14. *Let $\mathcal{A}$ and $\mathcal{B}$ be two Büchi automata and $\ddot{Q}^1_{\mathcal{A},\mathcal{B}}$ the corresponding set of initial grounded supergraphs. Then $L(\mathcal{A}) \not\subseteq L(\mathcal{B})$ iff $\widehat{Q}^f_{\mathcal{A},\mathcal{B}}$ contains a supergraph $\widehat{h} = \langle \langle s, s \rangle, \widetilde{h} \rangle$, where $s \in F_{\mathcal{A}}$, with no 1-labeled strongly connected component.*

Proof: This follows from Lemma 4.4.13, an analogue of Lemma 4.2.5, and Lemma 4.2.6. Lemma 4.2.5 does not rely on the contents of the arcs of supergraphs, only on their reflexivity, and so can be applied to a set of grounded supergraphs under the same logic used to apply Lemma 4.2.5 to a set of supergraphs □

Lemma 4.4.15. *Let $\mathcal{A}, \mathcal{B}$ be two Büchi automata and $\ddot{Q}^1_{\mathcal{A},\mathcal{B}}$ the corresponding set of initial grounded supergraphs. $L(\mathcal{A}) \not\subseteq L(\mathcal{B})$ iff the $\ddot{Q}^{\preceq}_{\mathcal{A},\mathcal{B}}$ does not contain a grounded supergraph $\ddot{g} = \langle \langle [q, R], [q, R] \rangle, \widetilde{g} \rangle$ where $q \in F_{\mathcal{A}}$ and $\ddot{g}$ has no 1-labeled SCC.*

Proof: This proof of this lemma proceeds from Lemma 4.4.14 analogously to the manner in which the proof of Lemma 4.2.12 proceeds from Lemma 4.2.7. The proof of Lemma 4.2.12 builds on Lemmas 4.2.9, 4.2.8, and 4.2.10. These three lemmas are concerned with the arcs of supergraphs only in terms of composition, equality and reflexivity. The arcs of grounded supergraphs are used in the same manner, and are composed in the same way as the arcs of supergraphs. □

Chapter 5
Experimental Results

All the Ramsey-based algorithms presented in Chapter 4 and Section 2.3 have worst-case running times exponentially slower than the rank-based algorithms. All existing SCT solvers use these Ramsey-based algorithms, begging comparison with rank-based tools. We first compare rank-based and Ramsey-based approaches on the domain of size-change termination problems. Surprisingly, despite the exponential gap in running time the Ramsey-based tools perform better than rank-based tools on SCT problems. This leads us to examine the performance of Ramsey-based algorithms on Büchi universality problems. Our results indicate that, while the use of optimizations such as subsumption is vital to either approach, rank-based solvers do scale better on random Büchi universality problems.

5.1 Tools:

The formal-verification community has implemented rank-based tools in to measure the scalability of various approaches. The programming-languages community has implemented several Ramsey-based size-change termination tools. To compare the two approaches, we used the best-of-breed rank-based tool, and have lifted two SCT tools to handle general Büchi containment problems.

Mh: Mh is a tool, developed by Doyen and Raskin [6], that tests Büchi automata universality using the rank-based complementation construction of Section 2.2 and a subsumption relation on ranks. Mh combines a lasso-finding fixed-point computation, which requires only the union and predecessor operations, with a subsumption relation and specialized implementations of those operations. In doing so it can solve some problems with an exponential number of states by manipulating a polynomial number of representative states. Mh can test the universality of a single Büchi automaton with a language of two characters.

We expanded the Mh tool to handle Büchi containment problems with arbitrary languages, using the fixed-point and subsumption operation defined by Doyen and Raskin [6]. This required implementing a intersection operation specialized with respect to the subsumption relation, making it the first

implementation of the containment-checking algorithm presented in their paper.

SCTP: A direct implementation of the Ramsey-based size-change termination algo-

rithm, written by Carl C. Frederiksen in Haskell [8], SCTP solves size-change termination problems using the single-graph search of Theorem 2.3.11.

We have generalized SCTP to handle all Büchi containment problems. To handle problems $L(\mathcal{A}) \subseteq L(\mathcal{B})$ where $\mathcal{A}$ contains non-accepting states, SCTP was modified to track accepting states and check the arcs of potential counterexamples. This allows SCTP to test the containment $L(\mathcal{A}) \subseteq L(\mathcal{B})$ as long as $L(\mathcal{B})$ is strongly suffix closed with respect to $L(\mathcal{A})$. In order to handle the containment of arbitrary automata, SCTP can use either the double-graph search over supergraphs, or the single-graph search over grounded supergraphs. SCTP implements a straightforward double-graph search, as described in Corollary 3.4.6, without the reachable subset optimization of Lemma 4.3.5. When using grounded supergraphs, SCTP implements the single-graph search without the use of subsumption or strongly connected components, as in Lemma 4.4.13.

SCTP was also used as the front-end for the sct/scp containment solver described below. Given a containment problem $L(\mathcal{A}) \subseteq L(\mathcal{B})$, SCTP can create the initial set of grounded supergraphs, $\ddot{Q}^1_{\mathcal{A},\mathcal{B}}$, specified in Section 4.4. Finally, we have extended SCTP to reduce SCT problems to Büchi containment problems, using either Definition 2.3.6 or Definition 4.1.1.

sct/scp: A C implementation of the Ramsey-based size-change termination algorithm by Ben-Amram and Lee, sct/scp uses the subsumption relation proven in Section 4.2. When generating a size-change graph, sct/scp avoids considering it further if it is conservatively approximated by a size-change graph that has already been seen. This implements most of the optimizations afforded by the subsumption operation in Section 4.2, but does not remove existing graphs that are approximated by a newly generated graph. The sct/scp program also divides the initial set of supergraphs into strongly connected components, based on the arc of each supergraph, and considers each SCC separately.

We extended sct/scp to implement the single-graph search of Algorithm 4.2.2, which solves containment problems with strongly suffix-closed languages. In order to solve arbitrary Büchi containment problems using Algorithm 4.4.2, sct/scp uses SCTP as a front-end to generate the initial set of grounded supergraphs, $\ddot{Q}^1_{\mathcal{A},\mathcal{B}}$, from the two automata $\mathcal{A}$ and $\mathcal{B}$. Optionally, sct/scp can avoid discarding subsumed grounded supergraphs, allowing a direct observation of the effectiveness of subsumption.

5.1.1 Notation

In the following graphs, programs and settings are identified by the following labels.

(1) **SCTP:** SCTP solver using the double-graph search to find and test proper pairs of supergraphs, as in Corollary 3.4.6.

(2) **SCTP(GSG):** SCTP solver using the single-graph search to find and test weakly proper grounded supergraphs, as in Lemma 4.4.13.

(3) **sct/scp:** sct/scp solver using the single graph search with subsumption to test the strongly connected components of grounded supergraphs, as in Algorithm 4.4.2.

(4) **sct/scp(N):** sct/scp solver using the single-graph search *without subsumption*, to test the strongly connected components of grounded supergraphs, as proven in Lemma 4.4.14.

(5) **Mh:** Mh solver using the rank-based algorithm of [6], based on Lemma 2.2.1, that leverages subsumption.

5.2 Size-Change Termination

All experiments on SCT problems were performed on a Dell Optiplex GX620 with a single 3.2Ghz Intel Pentium 4 CPU and 1 GB of RAM.

Because of the exponential gap in running time between Ramsey-based and rank-based approaches, one might expect rank-based containment solvers to outperform Ramsey-based SCT solvers on size-change termination problems. Since using rank-based containment solvers requires reducing size-change termination problems to Büchi containment problems, we first compare the original LJB reduction to the more compact reduction of Definition 4.1.1. Discovering that the new reduction produces easier Büchi containment problems, we then compare existing, Ramsey-based, SCT tools tools to rank-based Büchi containment solvers on the domain of SCT problems, with surprising results.

5.2.1 Experimental Setup:

Problem Space: Existing experiments on the practicality of SCT solvers focus on examples extracted from the literature [2][1]. We combine examples from Arne Glenstrup's Master's thesis [9], the benchmarks performed by Ben-Amram and Chin Soon Lee [2], and a variety of other sources [1, 8, 12, 14, 19]. Each tool was given a little under an hour, 3500 seconds, to solve each problem. The time spent reducing SCT problems to Büchi automata never took longer than 0.1 seconds and was dominated by I/O. Thus this time was not counted[2]

[1]We note that this thesis considers the number of initial size-change graphs to indicate the size of a problem. However, the Ramsey-based SCT algorithm is not exponential in the number of size-change graphs, rather it is exponential with regard to the largest number of parameters in any graph.

[2]Experimenting reveals that reading in pre-generated automata obtains a statistically insignificant slowdown when compared with reading the SCT problem and performing the reduction.

Procedure: The problem space was examined in two ways. First, for each SCT problem we compared the difficulty of the Büchi containment problem produced by each reduction from the SCT problem. Each SCT problem was converted into a Büchi containment problem using both the original reduction of Definition 2.3.6 and the revised reduction of Definition 4.1.1. The difference in size was measured, and the resulting Büchi automata containment problems were passed as input the Mh and SCTP tools. Second, we compared the performance of the rank-based Mh solver on the derived Büchi containment problem to the performance of the existing SCT tools on the original SCT problem. If a SCT problem was solved in all incarnations and by all tools in 0.01 seconds or less, the problem was discarded as uninteresting.

5.2.2 Experiment Results:

The sample space of SCT problems is very small and hand constructed. This makes it difficult to draw firm conclusions. Of the 242 SCT problems derived from the literature, only 62 provided interesting results as defined above. We measure performance as a percentage of the interesting problems completed in a given time, and scaling by increasing the allotted time.

Reduction Comparison: On SCT problems of a single function, the revised reduction of Definition 4.1.1 produces identical automata to the original reduction of Definition 2.3.6. On other problems the revised reduction produces consistently smaller automata. Measuring the size of the output file in kilobytes, Figure 5.1 shows the percentage decrease in size the revised reduction obtained with respect to the original reduction. Larger files benefited from a more significant decrease in size, most likely due to an increase the number of functions. While size may not be correlated with difficulty, this does confirm that the revised reduction produces smaller automata containment problems.

To compare the difficulty of the problems produced by the two reductions, the SCTP and Mh tools were run on the automata containment problem each reduction produced from each size-change termination problem. Figure 5.2 charts the percentage of interesting problems that could be completed in a given time, using each tool and each reduction. The smaller state space and, to a lesser extent, transition function significantly improved the scalability of the rank-based Büchi containment tool, Mh, on the given SCT problems. The Ramsey-based containment tool noticed less of a difference: although it was able to solve certain problems faster, all problems that could be solved using the revised reduction could be solved using the original reduction in only slightly more time.

One factor in the difference may be that the rank-based tool uses an encoding of states that is dense. States in the rank-based complementation construction are rankings: a subset of the states in the original automaton, each associated with a logarithmically sized rank. Mh stores these ranking by using an array to hold the

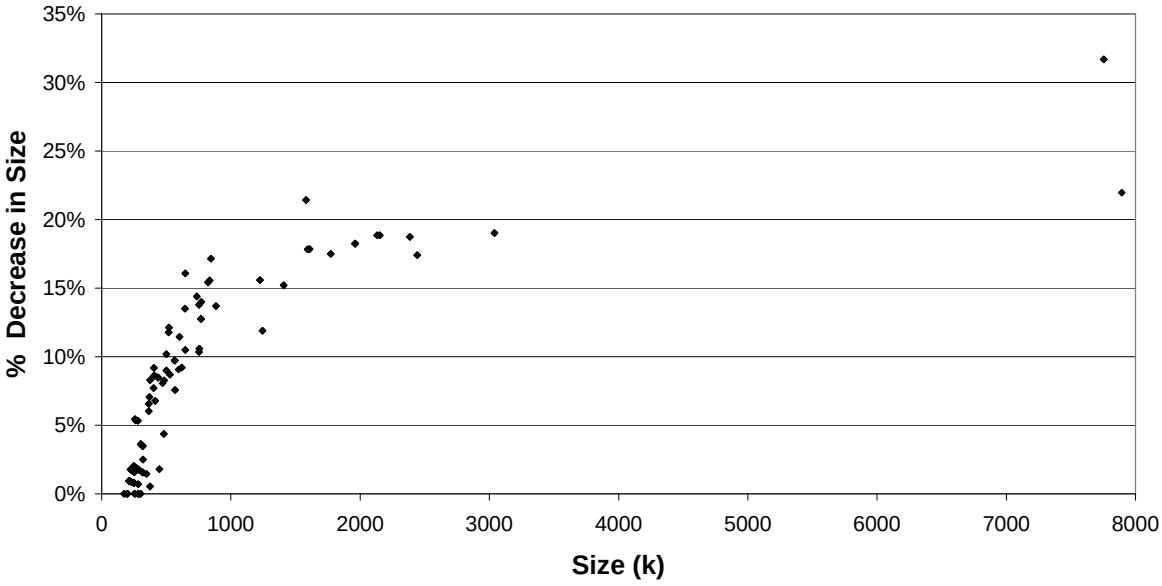


Figure 5.1: Improvement in size obtained with revised reduction of SCT problems to Büchi automata.

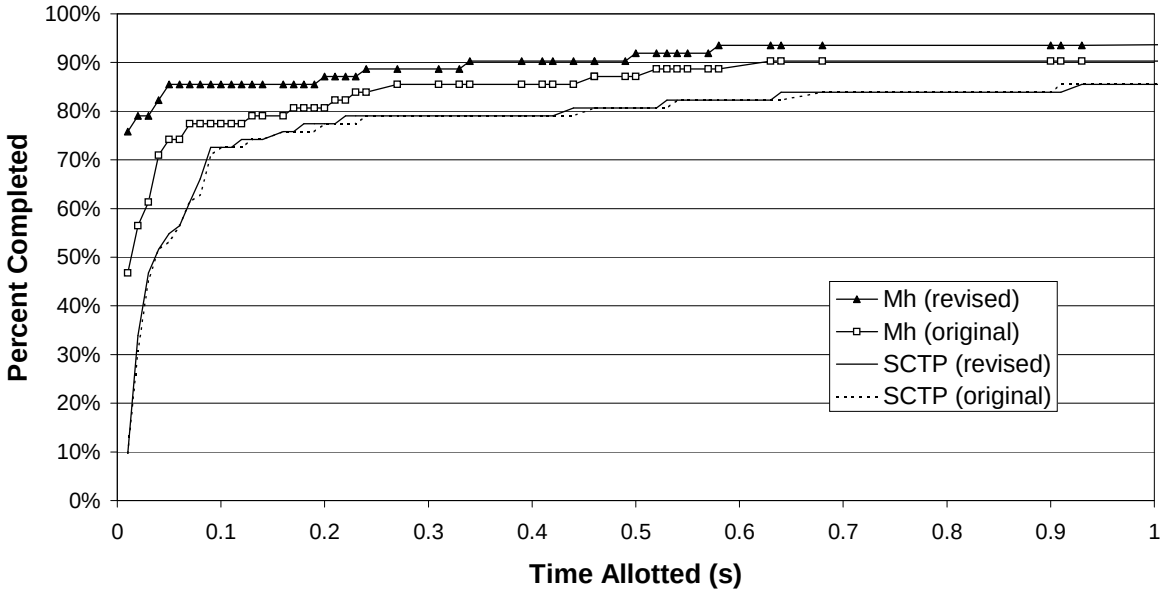


Figure 5.2: Scaling of original and revised reductions from SCT problems to Büchi automata containment problems.

rank of each state. If the state does not appear in the subset, it is given a invalid rank. Therefore increasing the number of states increases the size of this array and the amount of space required to store each ranking, even if the new states do not appear in the ranking. In contrast the Ramsey-based tool uses an encoding of supergraphs that is sparse, storing only the arcs present in each supergraph. Thus increasing the number of transitions does increase the amount of space required to store a supergraph, but increasing only the number of states does not. Since switching to the new reduction decreases the number of states more drastically than the number of transitions, we believe the difference in performance to be more related to the encoding style than the algorithm used.

As noted, for a problem with a single function there is no difference between the automata produced by Definitions 2.3.6 and 4.1.1. Most of the interesting problems have one or two functions, and were constructed to be difficult for the number of parameters. For these cases there is a relatively small difference between the number of states in $\mathcal{A}_{Desc(L)}$ and $\mathcal{A}'_{Desc(L)}$. Results from this experiment might not generalize to the behavior of real-life problems containing a larger number of functions, but with simpler size-change graphs.

Ramsey-based vs. Rank-based Solutions: Figure 5.3 compares the performance of the rank-based Mh solver against the performance of the existing SCT tools, displaying the percentage of problems each tool could complete in a given time. Note that SCTP was used here to solve the original size-change termination problem, not the derived automata containment problem. Time taken to reduce SCT problems to automata containment problems, using Definition 4.1.1, was not counted.

Table 5.1 displays the maximum percentage handled by each tool, and the time at which all solvable problems were completed. The test suite was very sparse: despite the hour-long timeout given, only one problem that was solved required more than one minute.

Tool	% Completed	Time (s)
sct/scp	100%	92.36
SCTP	94.1%	7.42
Mh	94.1%	8.83

Table 5.1: Percentage of SCT problems completed by tool.

The small problem space makes it difficult to draw firm conclusions, but it is clear that Ramsey-based tools are comparable to rank-based tools on size-change termination problems. In fact, the only tool able to solve all problems in an hour was Ramsey-based. This is surprising given the significant difference in worst-case complexity. By way of explanation, we note that all problems in this experiment have size-change graphs whose vertices have an in-degree of at most 1. In the presence of this property, a size-change graph can have no more than n arcs, one entering each

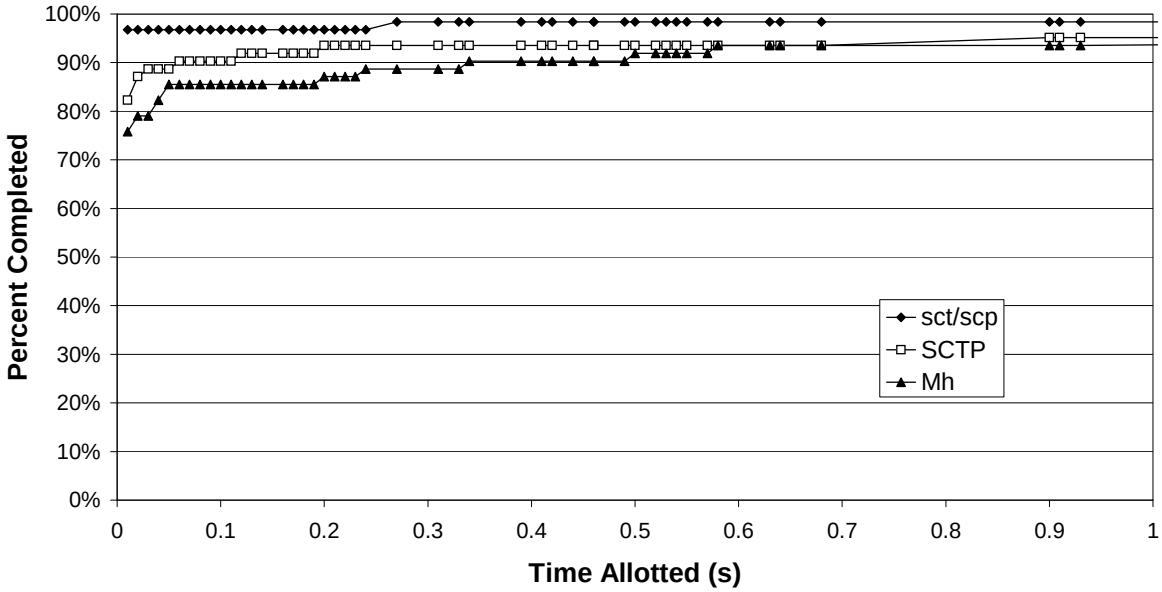


Figure 5.3: Scaling of Ramsey and rank-based approaches on SCT problems.

vertex. This reduces the number of possible size-change graphs to $2^{O(n \log n)}$, reducing the worst-case complexity of the Ramsey-based solutions to roughly the same as the worst-case complexity of rank-based solutions. In the automata corresponding to a SCT problem, this property emerges as reverse determinism: no state has two incoming arcs labeled with the same character.

5.3 Büchi automata

All experiments on Büchi automata were performed on the Ada Cray XD1 Research Cluster[3], a cluster 158 nodes, each with two dual-core 2.2 GHz AMD Opteron CPUs, 1 MB L2 cache and 8 GB of RAM.

It is apparent that Ramsey-based tools are competitive with or even superior to rank-based tools on SCT problems. Unfortunately the sample space is very small and artificially constructed. This leads us to wonder about the utility of Ramsey-based tools on Büchi universality problems. Is the massive gap in the theoretical running times of the two algorithms reflected in actual performance?

5.3.1 Experiment Setup

Problem Space: To evaluate the performance of various tools on Büchi universality

[3]http://rcsg.rice.edu/ada/

problems, we employed the random model proposed by Tabakov and Vardi and later used by Doyen and Raskin [17, 6]. This model fixes the input alphabet as $\Sigma = \{0, 1\}$, and considers the containment of Σ^ω in, and thus the universality of, the language of a random automata. Each automaton $\mathcal{A} = \langle \Sigma, Q, Q^{in}, \rho, F \rangle$ is constructed with a given *size* n, *transition density* r, and *acceptance density* f. Q is simply the set $\{0..n\}$, and $Q^{in} = \{0\}$. For each letter $\sigma \in \Sigma$, $n * r$ pairs of states $(s, s') \in q^2$ are chosen uniformly at random and the transition $\langle s, \sigma, s' \rangle$ is include in ρ. We imposed one exception for the initial state: when building the random graphs we ensure that the initial node has an outgoing transition for each letter of the alphabet, which helps us avoid trivial cases of non-universality. The set F of accepting states comprises $n * f$ states, likewise chosen uniformly at random. Data points are the mean of a hundred random automata with the given n, r, and f. The value of n ranges from 10 to 200, r from 1 to 3, and f from near 0 to near 1.

Procedure: We conducted two experiments over this space of automata. First, we held size constant and varied the acceptance and transition density. Doing so allowed us to discover which configurations are difficult for which tools. Second, we took the most difficult parameters and compare Ramsey and rank-based Büchi universality solvers, with and without subsumption, on problems with these parameters. All Ramsey-based solvers handle only containment problems, and thus universality problems were converted to containment problems by adding a universal automaton. Each problem was given 3500 seconds, nearly one hour, to complete.

Mh was used as the rank-based solver with subsumption. We used both SCTP and sct/scp as subsumption-free Ramsey-based solvers, and sct/scp as the Ramsey-based solver with subsumption. As sct/scp only implements the single-graph search over grounded supergraphs, Büchi universality problems must first be reduced to initial sets of grounded supergraphs by SCTP. This is a potentially exponential reduction.

There are a couple important procedural notes. First, when subsumption is not used in sct/scp the program still suffers the overhead associated with comparing states. While this does not increase the theoretical complexity, sct/scp still payed the cost of subsumption even when not reaping the benefits. Second, as mentioned above the sct/scp program requires the SCTP front-end to convert automata into an initial set of grounded supergraphs, which are closed under extension. The sct/scp tool then performs the single-graph search on this initial set of grounded supergraphs. The front-end conversion exponentially increases the number of graphs by a factor of up to $2^{O(n)}$. However, the number of vertices per graph does not increase. As the single-graph search is exponential in the number of vertices, the blowup is not compounded and the $2^{O(n^2)}$ single-graph search theoretically dominates. Because SCTP is not implemented efficiently, however, in practice the front-end can consume a significant amount of time. To be as charitable as possible to Ramsey-based approaches the time consumed by the front-end is not counted towards the timeout period.

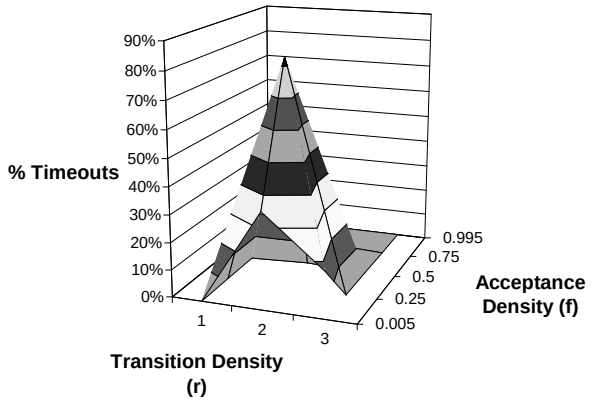


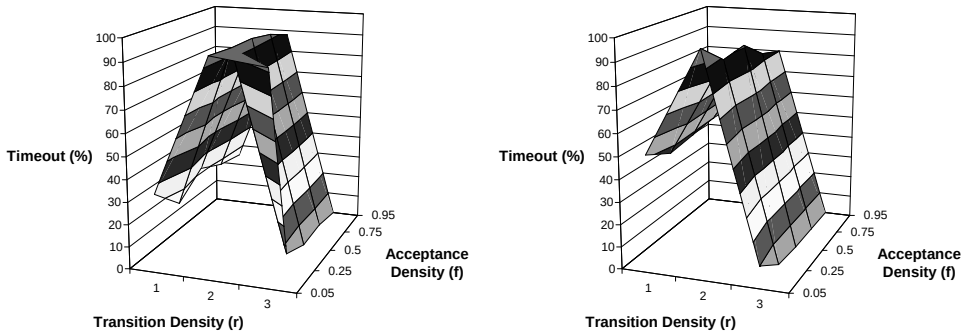
Figure 5.4: Mh program: timeout percentage vs. acceptance/final Density (200 states).

5.3.2 Experiment Results

Structural Parameters To measure the scalability of each algorithm on random automata, we first established which structural parameters induce the most difficult problems. Previous research has shown that for rank-based solvers a transition density (r) of 2 is vastly more difficult than problems with $r = 1$ or $r = 3$ [17]. Acceptance density (f) has had less effect on performance, but problems become more difficult for existing tools as f nears 0.3 [17]. The goal of these measurements is not to compare performance between tools, but to measure the difficulty of various transition and acceptance densities within a tool.

As shown in Figure 5.4, and consistent with the results of [17], Mh has the most difficulty with random problem of transition density 2. Unlike the results in [17], Mh encountered difficulty only with acceptance densities of 0.005 and 0.25. Mh's subsumption operation groups all accepting states under one chain, and so appears to handle problems with many accepting states very well.

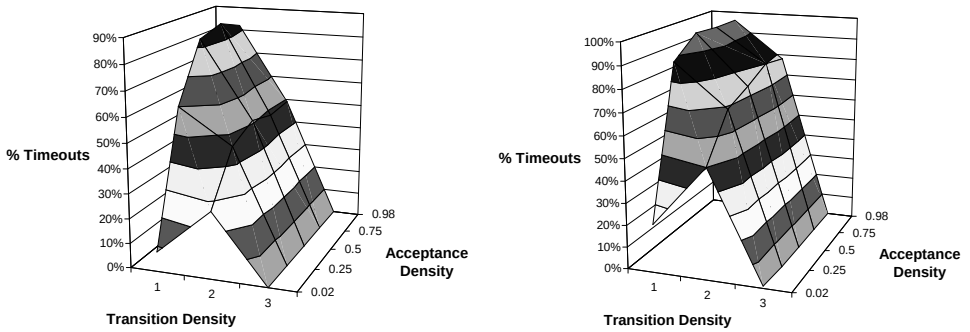
Figure 5.5 describes the behavior of the SCTP solver. Figure 5.5(a) shows the difficulty encountered using the double-graph search algorithm: create all supergraphs by composition, and then search pairs of supergraphs for a counterexample. With some slight increase in timeouts towards problems with few accepting states, the SCTP solver demonstrates the same behavior described in [17]: problems with a transition density (r) of $r = 2$ are by far the most difficult. Figure 5.5(b) displays the characteristics SCTP when using the single-graph search based on the grounded supergraphs of Section 4.4. Although it performs slightly better on the hardest problems where $r = 2$, we draw attention to the behavior of the single-graph search on problems with a transition density (r) of 1. While the double-graph search timed out in roughly 30% of these cases, the single-graph search using grounded supergraphs timed out more than 50% of the time. This behavior can be explained by observing that the number



(a) Double-Graph Search (Corollary 3.4.6)

(b) Single-Graph Search (Lemma 4.4.13)

Figure 5.5: SCTP program: timeout percentage vs. acceptance/final density (20 states).



(a) 60 States

(b) 100 States

Figure 5.6: sct/scp program: timeout percentage vs. acceptance/final density (60 and 100 states).

of grounded supergraphs is dependent on the number of reachable subsets of states. With low transitions densities, reachable subsets can grow more slowly and remain more numerous. On smaller problems, this behavior dominates and the single-graph search over grounded supergraphs is slower on problems with $r = 1$. This behavior suggests that unlike other approaches, the single-graph search does find problems with a transition density of 1 to be difficult. It is an open question if the algorithm of Lemma 4.3.5, which searches pairs of reachable subsets and supergraphs, suffers from the same problem.

The picture for sct/scp in Figure 5.6 is less clear. As sct/scp only implements the algorithm using grounded supergraphs, we observe difficulty with low transition-density (r) problems. This is exacerbated by the fact that sct/scp relies on SCTP as a front-end. As SCTP is not written with efficiency in mind, scaling in the front-end

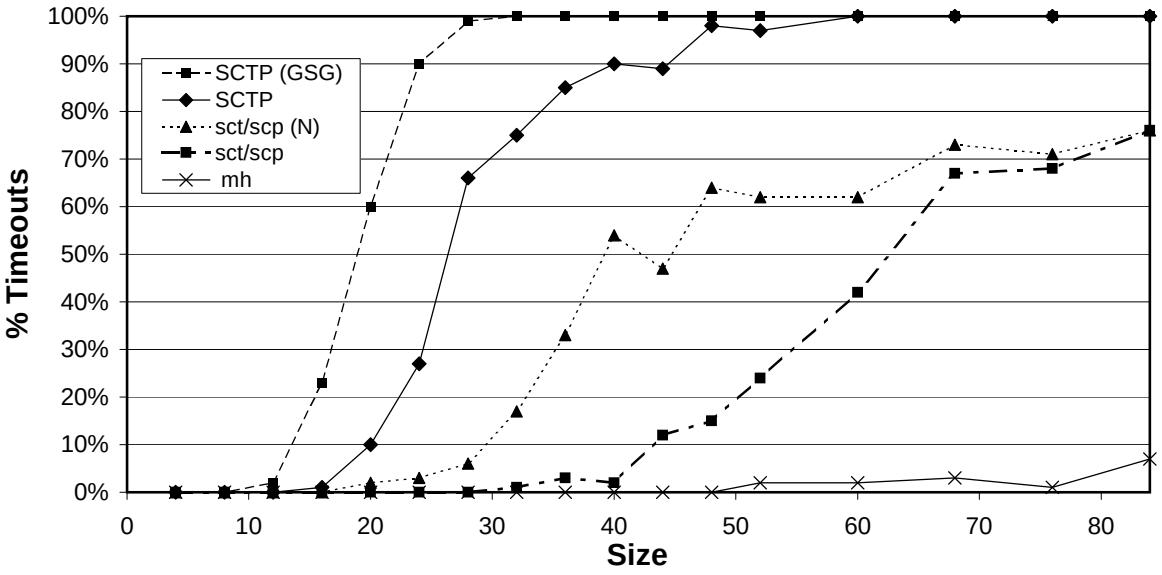


Figure 5.7: Scaling of Ramsey and rank-based approaches on Büchi automata universality problems.

becomes a problem with larger automata. On problems with a low transition density, $r = 1$, the front-end task of generating the initial set of grounded supergraphs can take as much time as performing the single-graph search on this set. Cases where the front-end consumed more than one hour were considered unsolvable, otherwise the time taken by the front-end is discounted. Even disregarding the time taken by the front-end, it can be seen that problems with an $r = 1$ and acceptance density (f) of 0.25 are the most difficult, while problems with a r of 2 remain problematic. By comparing the graphs for two different sizes, we can observe that the gap in difficulty between $r = 1$ and $r = 2$ is inversely proportional to the size of the automata, but even at the limit of what sct/scp can handle problems with $r = 1$ remain the most difficult. It is possible that, as with SCTP, the double-graph search is better equipped to handle problems with low transition density. Given the current tools, however, Ramsey-based approaches cannot compete with rank-based approaches on problems with $r = 1$.

Scalability The comparison we are most interested in is the scalability of Ramsey-based algorithms versus rank-based algorithms, and the effect of subsumption on both. As reported in [6], subsumption is critical to the performance of rank-based algorithms. We want to investigate the utility of subsumption in Ramsey-based approaches. The tools were compared on problems where $r = 2$ and $f = 0.25$, structural parameters that every tool has difficulty with. The results of this experiment are reported in Figure 5.7.

With regard to SCTP, while the algorithm using grounded supergraphs and the single-graph search performs worse, it appears to have roughly the same slope as the algorithm using supergraphs and the double-graph search. This conclusion supports the intuition that, as the two algorithms have the same worst-case complexity, they will scale similarly.

Observing the data on sct/scp, however, it appears that subsumption does increase the scalability of Ramsey-based algorithms. Even though sct/scp only implements half of the subsumption relation on grounded supergraphs, it scales significantly better when this subsumption operation is used. It is very interesting to note that, for large automata, the benefits of using subsumption appear to vanish. With automata of 84 states, sct/scp can solve no more problems with subsumption than without subsumption.

To explain this behavior, we note that when given automata with more than 52 states, sct/scp without subsumption only terminated on non-universal automata. For automata with 52 or more states, roughly 30% were proven to be non-universal, 67% universal, and 2% could not be solved using any tool. The sct/scp solver was able to complete testing every known non-universal automata whether or nor subsumption was used. These automata were declared non-universal as soon a counterexample grounded supergraph was discovered. Many of these counterexamples were already present in the initial set of grounded supergraphs provided by SCTP, which comprise of all grounded supergraphs that have a single character in their suffix language. With automata of size 84, 22 of the 24 known non-universal automata had a counterexample grounded supergraph in this initial set. In these cases subsumption was never a factor: composition was never required to find the counterexample. Even the remaining two counterexamples required only the composition of two to four grounded supergraphs. In fact, subsumption was most useful in sct/scp lies when proving automata universal: with subsumption, sct/scp discovered 72 universal automata of with 52 or more states. Without subsumption, sct/scp could only discover 4.

In comparing Ramsey and rank-based approaches, first recall that sct/scp uses the SCTP program as a front-end to produce an initial set of grounded supergraphs. Doing so can take up to eight minutes on large problems, and this time was not counted towards the timeout. Even when not counting this time the rank-based Mh solver scales significantly better than Ramsey-based approaches. In the absence of reverse determinism, it appears that the exponential gap in theoretical worst-case running time between rank-based and Ramsey-based approaches to universality checking is mirrored in empirical performance on the domain of random problems.

It is interesting to note that the rank-based solution outperformed the Ramsey-based solution only on universal automata. The rank-based tool, Mh, terminated on all 337 known universal automata with 52 or more states, while the Ramsey-based sct/scp tool solved only 72. However, Mh was only able to solve 148 of the 152 automata with 52 or more states that sct/scp determined were non-universal. To explain this behavior, we note that Mh searches for a lasso through the complementary

automaton, and so can terminate before computing the entire fixed point in cases where the automaton can be proven universal. To prove an automaton non-universal, Mh must compute the entire fixed point. By contrast, the Ramsey-based solution searches for a counterexample grounded supergraph, and so can terminate as soon as such a grounded supergraph is found and the automaton is proven non-universal. To prove an automaton universal, however, sct/scp must compute the entire fixed point and check every grounded supergraph.

Chapter 6
Conclusion

In this thesis we demonstrate that the Ramsey-based size-change termination algorithm proposed by Lee, Ben-Amram, and Jones is a specialized realization of the 1987 Ramsey-based complementation construction by SVW. With this link established, we compare rank-based and Ramsey-based tools on the domain of SCT problems. Despite the presence of new constructions with exponentially better worst-case bounds, we discover to our surprise that size-change termination problems are best solved using Ramsey-based approaches. This prompts us to, for the first time, empirically investigate the viability of Ramsey-based tools on Büchi universality problems. Despite a valiant effort, in the end Ramsey-based tools cannot compete with best-of-breed rank-based tools on the domain of Büchi universality problems. We hypothesize that the presence of reverse determinism in all existing size-change termination problems is responsible for difference between the two problem domains.

Our experiments yield two other interesting observations. First, subsumption appears to be critical to the performance of Büchi complementation tools using both rank and Ramsey based algorithms. It has already been established that rank-based tools benefit strongly from the use of subsumption [6]. Our results demonstrate that Ramsey-based tools also benefit strongly from subsumption, as sct/scp scales better when conservatively approximated graphs are ignored. Second, we note that rank-based tools seem better suited to proving automata universality and containment problems, and Ramsey-based tools seem slightly better suited to disproving automata universality and containment problems. In application, disproving a automata containment problem corresponds to discovering a bug in the program. Thus Ramsey-based tools may find more bugs than their rank-based counterparts.

It is also worth observing that importance of implementation details on scalability. We find that the different in scalability between SCTP and sct/scp is as great, if not greater, than the difference imposed by subsumption. While some of this can be attributed to sct/scp's use of strongly connected components in both dividing the problem and testing graphs, we believe that the method of implementation is also significant. While Haskell is a compiled language, the SCTP program is intended as a reference implementation. Many operations that might be better handled by hashtables, trees, or arrays are instead managed by lists. The sct/scp tool is written in C, and uses low-level and concrete code.

6.1 Future Work

Experimental: There are several experimental paths suggested by these results. Improving the front-end for sct/scp, either by more efficient implementation or by use of a subsumption relation, could significantly improve the final performance. Further, every tool used in this thesis is non-symbolic. Previous research [17] has indicated that symbolic approaches may scale better. A symbolic implementation of Doyen and Raskin's algorithm might prove interesting.

Theoretical: On the theoretical side, we are very interested in extending the subsumption relation present in sct/scp. The requirement that automata containment problems be phrased as a single-graph search over grounded supergraphs in order to leverage subsumption imposes additional complexity. Extending the subsumption relation to the double-graph search of Lemma 4.3.5 would simplify the implementation greatly. Further, the subsumption relation is not fully integrated into the sct/scp solver. Mh demonstrates the benefits of a strong theoretical basis for subsumption, and a similar treatment to Ramsey-based solvers might yield improvements.

On another track, the effects of reverse determinism on the complementation of automata bear study. Reverse determinism is not an obscure property, it is known that automata derived from LTL formula are reverse deterministic [7]. As noted above, the Ramsey-based approach improves exponentially when operating on reverse deterministic automata. It can also be shown that the rank-based approach can be improved exponentially, to $2^{O(n)}$, when it is guaranteed to receive reverse deterministic automata. Further, Ben-Amram and Lee have defined SCP, a polynomial-time approximation algorithm for size-change termination. For a wide subset of SCT problems, including the set used in this thesis, SCP is exact. SCP is exact in cases where graphs have restricted in degrees. In terms of automata, this property is similar, although perhaps not identical, to reverse determinism. The presence of an exact polynomial algorithm for the SCT case suggests a subset of Büchi containment problems may be solvable in polynomial time. The first step in this direction would be to determine what properties a containment problem must have to be solved exactly in this fashion.

Utility: Considering the scalability and the newly implemented ability to solve Büchi containment problems, the Mh tool could be considered for use in actual applications. In the case of the satisfiability problem for first-order logic, tools can solve problems resulting from applications that are larger, by several orders of magnitude, than solvable random problems. This might also occur in the space of Büchi containment problems. The possibility of using a Büchi containment solver for actual verification is intriguing.

Bibliography

[1] Daedalus. Available on: `http://www.di.ens.fr/~cousot/projects/DAEDALUS/index.shtml`.

[2] Amir M. Ben-Amram and Chin Soon Lee. Program termination analysis in polynomial time. *ACM Trans. Program. Lang. Syst*, 29(1), 2007.

[3] J.R. Büchi. On a decision method in restricted second order arithmetic. In *Proc. Int. Congress on Logic, Method, and Philosophy of Science. 1960*, pages 1–12. Stanford University Press, 1962.

[4] Y. Choueka. Theories of automata on ω-tapes: A simplified approach. *Journal of Computer and Systems Science*, 8:117–141, 1974.

[5] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 1990.

[6] Laurent Doyen and Jean-François Raskin. Improved algorithms for the automata-based approach to model-checking. In *TACAS*, pages 451–465, 2007.

[7] A.E. Emerson and A.P. Sistla. Deciding full branching time logics. *Information and Control*, 61(3):175–201, 1984.

[8] Carl Christian Frederiksen. A simple implementation of the size-change termination principle. 2001.

[9] Arne J. Glenstrup. Terminator ii: Stopping partial evaluation of fully recursive programs. Master's thesis, DIKU, University of Copenhagen, June 1999.

[10] S. Gurumurthy, O. Kupferman, F. Somenzi, and M.Y. Vardi. On complementing nondeterministic Büchi automata. In *Proc. 12th Conf. on Correct Hardware Design and Verification Methods*, volume 2860 of *Lecture Notes in Computer Science*, pages 96–110. Springer, 2003.

[11] O. Kupferman and M.Y. Vardi. Synthesizing distributed systems. In *Proc. 16th IEEE Symp. on Logic in Computer Science*, pages 389–398, 2001.

[12] Chin Soon Lee, Neil D. Jones, and Amir M. Ben-Amram. The size-change principle for program termination. In *POPL*, pages 81–92, 2001.

[13] M. Michel. Complementation is more difficult with automata on infinite words. CNET, Paris, 1988.

[14] Damien Sereni and Neil D. Jones. Termination analysis of higher-order functional programs. In *APLAS 2005*, Lecture Notes in Computer Science, 2005.

[15] A.P. Sistla, M.Y. Vardi, and P. Wolper. The complementation problem for Büchi automata with applications to temporal logic. In *Proc. 12th Int. Colloq. on Automata, Languages, and Programming*, volume 194, pages 465–474. Springer, 1985.

[16] A.P. Sistla, M.Y. Vardi, and P. Wolper. The complementation problem for Büchi automata with applications to temporal logic. *Theoretical Computer Science*, 49:217–237, 1987.

[17] D. Tabakov and M.Y. Vardi. Model checking buechi specifications. In *unknown*, unknown, page unknown. unknown, 2007.

[18] M.Y. Vardi. Automata-theoretic model checking revisited. In *Proc. 8th Int. Conf. on Verification, Model Checking, and Abstract Interpretation*, volume 4349 of *Lecture Notes in Computer Science*, pages 137–150. Springer, 2007.

[19] David Wahlstedt. Detecting termination using size-change in parameter values. Master's thesis, Göteborgs Universitet, 2000.

[20] P. Wolper. Specification and synthesis of communicating processes using an extended temporal logic. In *Proc. 9th ACM Symp. on Principles of Programming Languages*, pages 20–33, 1982.