Empirical Performance Investigation of a Büchi Complementation Construction

Daniel Weibel

July 27, 2015

Abstract

This will be the abstract.

Acknowledgements

Contents

1 Introduction **2**
- 1.1 Context . . . 3
  - 1.1.1 Büchi Automata and Büchi Complementation . . . 3
  - 1.1.2 Applications of Büchi Complementation . . . 4
  - 1.1.3 Significance of Büchi Complementation . . . 6
- 1.2 Motivation . . . 6
  - 1.2.1 Theoretical Investigation of Worst-Case Performance . . . 7
  - 1.2.2 Need for Empirical Investigation of Actual Performance . . . 7
- 1.3 Aim and Scope . . . 9
- 1.4 Overview . . . 9

A Plugin Installation and Usage **10**

B Median Complement Sizes of the GOAL Test Set **11**

C Execution Times **14**

Chapter 1

Introduction

At the beginning of the 1960's, a Swiss logician named Julius Richard Büchi was looking for a way to decide the satisfiability of formulas of the monadic second order logic with one successor (S1S). In his quest, Büchi observed that an S1S formula can be represented by a certain type of finite state automaton that runs on infinite words, such that this automaton accepts a word if and only if the corresponding interpretation satisfies the formula. The proof of this equivalence, which is today known as *Büchi's Theorem*, led Büchi to his desired decision procedure for the satisfiability of S1S formulas: to test whether a formula φ is satisfiable, translate it to an equivalent automaton A, and test whether A is empty (that is, accepts no words at all). If A is empty, then φ is unsatisfiable, if A is non-empty, then φ is satsifiable. [4]

The type of automata that Büchi used for solving his logical problem was later called *Büchi automata*. This type of automata lies at the core of this thesis, and we will examine it in detail in later parts. The impact of Büchi's result was not limited to a satisfiability decision procedure for S1S. Rather, Büchi set the foundation for the automata-theoretic approach to logic, which aims at reducing problems in logic to problems in automata theory, and which is a major paradigm today [52].

- automata-theoretic approach to logic - Büchi's Theorem - Proof includes proving that Büchi automata are closed under complementation (most difficult part).

1.1 Context

1.1.1 Büchi Automata and Büchi Complementation

Büchi automata are finite state automata that process words of infinite length, so called ω-words. If Σ is the alphabet of a Büchi automaton, then the set of all the possible ω-words that can be generated from this alphabet is denoted by Σ^ω. A word $\alpha \in \Sigma^\omega$ is accepted by a Büchi automaton if it results in at least one run that contain infinitely many occurrences of at least one accepting state. A run of a Büchi automaton on a word is an infinite sequence of states. Deterministic Büchi automata have at most one run for each word in Σ^ω, whereas non-deterministic Büchi automata may have multiple runs for a word.

The complement of a Büchi automaton A is another Büchi automaton[1] denoted by $\overline{A}$. Both, A and $\overline{A}$, share the same alphabet Σ. Regarding a word $\alpha \in \Sigma^\omega$, the relation between an automaton and its complement is as follows:

$$\alpha \text{ accepted by } A \iff \alpha \text{ not accepted by } \overline{A}$$

That is, all the words of Σ^ω that are *accepted* by an automaton are *rejected* by its complement, and all the words of Σ^ω that are *rejected* by an automaton are *accepted* by its complement. In other words, there is no single word of Σ^ω that is accepted or rejected by *both* of an automaton and its complement.

A Büchi complementation construction is an algorithm that, given a Büchi automaton, creates the complement of this Büchi automaton. The difficulty of this operation depends on whether the input automaton is determinstic or non-deterministic. The complementation of deterministic Büchi automata is "easy" and can be done in polynomial time and linear space [17]. The complementation of non-deterministic Büchi automata, however, is very complex. The understanding and reduction of its complexity is a domain of active research and lies at the centre of this thesis.

Consequently, when in the following we talk about "Büchi complementation", we specifically mean the complementation of *non-determinstic* Büchi automata. The main problem with the complexity of Büchi complementation is the so-called state growth or state complexity (sometimes also called state blow-up or state explosion). This is the number of states of the output automaton in relation to the number of states of the input automaton. In simple words, Büchi complementation constructions produce complements that may be very, very large.

This inhibits the practial applicaion of Büchi complementation, because in this case the limited computing and time resources may not be high enough to accommodate for this high complexity. In the following subsections we highlight an important application that Büchi complementation has in practice, and thereby motivate the research on Büchi complementation and of this thesis.

[1] Büchi automata are closed under complementation. This has been proved by Büchi [4], who, to this end, described the first Büchi complementation construction in history.

1.1.2 Applications of Büchi Complementation

Language Containment of ω-Regular Languages

Büchi complementation is used for testing language containment of ω-regular languages. The ω-regular languages are the class of formal languages that is equivalent to non-deterministic Büchi automata[2]. At this point, we briefly describe the language containment in general, before in turn describing an application of the language containment problem in the next subsection.

Given two ω-regular languages L_1 and L_2 over alphabet Σ^ω the language containment problem consists in testing whether $L_1 \subseteq L_2$. This is true if all the words of L_1 are also in L_2, and false if L_1 contains at least one word that is not in L_2. The way this problem is algorithmically solved is by testing $L_1 \cap \overline{L_2} = \varnothing$. Here, $\overline{L_2}$ denotes the complement language of L_2, which means $\overline{L_2}$ contains all the words of Σ^ω that are *not* in L_2. The steps for testing $L_1 \cap \overline{L_2} = \varnothing$ are the following:

- Create the complement language $\overline{L_2}$ of L_2
- Create the intersection language $L_{1,\overline{2}}$ of L_1 and $\overline{L_2}$
- Test whether $L_{1,\overline{2}}$ is empty (that is, contains no words at all)

Thus, the language containment problem is reduced to three operations on languages, *complementation*, *intersection*, and *emptiness testing*. The common way to work with formal languages is not to handle the languages themselves, but more compact structures that represent them, such as automata. In the case of ω-regular languages, these are non-deterministic Büchi automata.

For solving $L_1 \subseteq L_2$, one thus works with two Büchi automata A_1 and A_2 that represent L_1 and L_2, respectively. The problem then becomes $L(A_1) \subseteq L(A_2)$, and equivalently, $L(A_1) \cap \overline{L(A_2)} = \varnothing$. This is automata-theoretically solved as $\mathsf{empty}(A_1 \cap \overline{A_2})$, which includes the three following steps:

- Construct the complement automaton $\overline{A_2}$ of A_2
- Construct the intersection automaton $A_{1,\overline{2}}$ of A_1 and A_2
- Test whether $A_{1,\overline{2}}$ is empty (that is, accepts no words at all)

If the final emptiness test on automaton $A_{1,\overline{2}}$ is true, then $L_1 \subseteq L_2$ is true, and if the emptiness test is false, then $L_1 \subseteq L_2$ is false. In this way, the language containment problem of ω-regular languages is reduced to three operations of *complementation*, *intersection*, and *emptiness testing* of non-deterministic Büchi automata. Thus, Büchi complementation is an intergral part of language contaiment of ω-regular languages.

Automata-Theoretic Model Checking via Language Containment

In the last subsection, we have seen that Büchi complementation is used for testing language containment of ω-regular languages. In this subsection, we will see what in turn language containment of ω-regular languages is used for. To this end, we describe one important application of it, namely the language containment approach to automata-theoretic model checking. In the following, we first describe basic working of this technique in general, and then point out the significance that Büchi complementation bears for it.

Basics

The language containment approach to automata-theoretic model checking is an approach to automata-theoretic model checking, which is an approach to model checking, which in turn is an approch to formal verification [?]. Figure 1.1 shows the branch of the family of formal verification techniques that contains the language containment approach to automata-theoretic model checking.

Formal verification is the use of mathematical techniques for proving the correctness of a system (software of hardware) with respect to a specified property [?]. A typical example is to verify that a program is

[2] Note that deterministic Büchi automata have a lower expressivity than non-deterministic Büchi automata, and are equivalent to only a subset of the ω-regular languages.

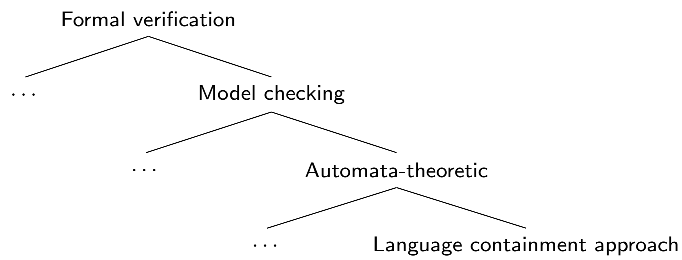

Figure 1.1: Branch of the family of formal verification techniques that contains the language containment approach to automata-theoretic model checking.

deadlock-free (in which case the property would be "deadlock-freeness"). In general, formal verification techniques consist of the following three parts [**?**]:

1. A framework for modelling the system to verify
2. A framework for specifying the property to be verified
3. A verification method for testing whether the system satisfies the property

For the language containment approach to automata-theoretic model checking, the frameworks for points 1 and 2 are Büchi automata representing ω-regular languages. The verification method (point 3) is testing language containment of the system automaton's language in the property automaton's language. In some more detail, this works as follows. [49][**?**]

The system s to verify is modelled as a Büchi automaton, say S. This Büchi automaton represents an ω-regular language $L(S)$, and each word of $L(S)$ represents in turn a possible *computation trace* of the system. A computation trace is an infinite[3] sequence of "situations" that the system is in at any point in time. Such a situation consists of a finite amount of information of, for example, the values of variables, registers, or buffers. The observation that such a trace can be represented as a word of an ω-regular langauges comes from the fact that it can be represented intuitively as a linear Kripke structure (which in turn is an interpretation for a temporal logic formula that can also be used to represent computations), which in turn can be represented by a word of a language whose alphabet is ranges over the powerset of the atomic propositions of the Kripke strucutre. A work that explains these intimate relations between computation, temporal logic, formal languages, and automata in more detail is [49]. In simple words, the language $L(S)$, represented by the system automaton S, represents everything that the system *can* do.

Similarly, a property p to be verified is represented as a Büchi automaton, say P, which represents the ω-regular language $L(P)$, whose words represent computation traces. These computation traces are all the computations of a system like s that satisfy the property p. If for example p is "deadlock-freeness", then the words of $L(P)$ represent all the possible computation traces that do *not* contain a deadlock. In this way, the language $L(P)$ represents everything that the system is *allowed* to do, with respect to a certain property.

The verification step is finally done by testing $L(S) \subseteq L(P)$. If this is true, then everything that the system *can* do is *allowed* to do, and the system satisfies the property p. If the language containment test is false, then the system *can* do a computation that is *not allowed* to do, and the system does not satisify the property p.

Summarising, the language containment approach to automata-theoretic model checking requires language containment of ω-regular languages, which, as we have seen, requires Büchi complementation. In the following, we will highlight the particular importance of Büchi complementation for this type of formal verification.

[3]The infinity of computation traces suggests that this type of formal verification (and model checking in general) is used for systems that are not expected to terminate and may run indefinitely. This type of systems is called *reactive* systems. They contrast with systems that are expected to terminate and produce a result. For this latter type of systems other formal verification techniques than model checking are used. See for example [**?**] and [**?**] for works that cover the formal verification of both types of systems.

1.1.3 Significance of Büchi Complementation

The complexity of Büchi complementation makes the just described model checking approach nearly inapplicable in practice [**?**]. According to [**?**], there are so far no tools that include this approach. This is unfortunate, because the other Büchi operations for language containment, intersection and emptiness testing, have highly efficient solutions [**?**] (cf. [49]), and thus Büchi complementation is the only bottleneck. Existing practical applications are thus forced to circumvent the need for Büchi complementation. This is possible, has however certain disadvantages as we will see in the following.

One way to circumvent the complementation of non-deterministic Büchi automata is to specify the property as a deterministic Büchi automaton [**?**][**?**]. As we have mentioned, the complementation of deterministic Büchi automata has an efficient solution. The disadvantage of this approach is, however, that the property automaton may become exponentially larger, and that it is generally more complicated and less intuitive to represent a language as a deterministic automaton [**?**].

Another way is to use a different model checking approach altogether, which leads us back to the essence of model checking. In basic model checking, the property to be verified is represented as a formula φ of a temporal logic (typically LTL). The system to verify is represnted as a Kripke structure K, which serves as an interpretation of the formula φ. The verification step consists in checking whether K is a *model* of φ. An interpretation K is a model of a formula φ, if every state of the interpretation *satisifes* the formula, written as $K \models \varphi$. This test of modelhood is the reason that this verification approach is called *model checking* [49].

The modelhood test can be done automata-theoretically without the need for Büchi complementation [**?**] (in Figure 1.1, this would be a sibling to the language containment approach). The Kripke structure K is translated to a Büchi automaton A_K. The formula φ is negated and translated to the Büchi automaton $A_{\neg\varphi}$. Finally, one tests $\mathsf{empty}(A_K \cap A_{\neg\varphi})$. This correponds to teh language containment test $L(K) \subseteq L(\varphi)$, which is equivalent to the modelhood test $K \models \varphi$. The trick is that the complementation of the property, that is required for the language containment test, is pushed off from the complementation of a Büchi automaton to the negation of a temporal logic formula, which is trivial. This approach is used, for example, by the SPIN model checker [**?**]. The disadvantage is that the typically used temporal logic LTL is less expressive than Büchi automata, and hence the breadth of properties that can be expressed is limited. It has been stated that the expressivity of LTL is unsufficient for industrial verification applications [**?**].

For more information on model checking, as well as other formal verification techniques, we refer to the following works: [**?**][**?**][**?**].

As can be seen from these elaborations, having efficient procedures for Büchi complementation would be of great practical value. Even though handling the "worst-cases" will forever be unefficient,

1.2 Motivation

In the previous section we have seen that Büchi complementation is complex, and that it would be of practical value to better understand this complexity. In this section, we highlight the need for looking at this complexity in a way that has not received much attention in the past, namely empirically rather than theoretically.

In the following, we first present the traditional way of analysing the worst-case performance of complementation constructions, and then describe the empirical way for investigating their actual peformance. This includes a review of the work that has been done so far. Note that we are using the terms complexity and performance interchangeably, and they both mean basically state growth.

1.2.1 Theoretical Investigation of Worst-Case Performance

The traditional performance measure for Büchi complementation constructions is their *worst-case state growth*[4]. This is the maximum number of states the construction *can* generate, in relation to the number of states of the input automaton.

For example, the initial complementation construction by Büchi (1962) [4] has a worst-case state growth of $2^{2^{O(n)}}$ does not mean that it produces a larger complement than Schewe's construction, for this concrete example. It might well be smaller. In fact, worst-case state complexities only allow to adequately deduce something about the specific worst-cases, and not about all the other automata. From a practical point of view, these worst cases are however not interesting, as their application is impracticable anyway (at least starting from a certain input automaton size). , where n is the number of states of the input automaton. At this point, two short comments. First, the state growth is often not given as an exact function, but uses the big-O notation. Second, for notating state growths, we will consistenly use the variable n, whose meaning is the number of states of the input automaton. This means, for example, that for an input automaton with 8 states, the maximum number of states that the output automaton of Büchi's construction can have is $1.16 \times 10^7 7$ (if assuming the concrete function 2^{2^n}).

Different constructions exhibit different worst-case state growths, and one of the main objectives of construction creators is to reduce this number. For example, the much more recent construction by Schewe (2009) [35] has a worst-case state growth as low as $(0.76n)^n + n^2$. Given an input automaton with 8 states, the maximum number of states of the output automaton is approximately 119.5 million.

A related objective of research is the quest for the theoretical worst-case state growth of Büchi complementation *per se*. A first result of $n!$ has been proposed in 1988 by Michel [20]. He proved that there exists a family of automata whose complement *cannot* have less than $n!$ states (these automata are known as Michel automata, and we will use them as part of the test data for our experiments). This proves a *lower bound* for the fundamental worst-case complexity of Büchi complementation, as it is not known whether the Michel automata are the real worst cases, or if there are even worse cases. Indeed, in 2007, Yan [55] proved a new higher lower bound of $(0.76n)^n$ (Michel's $n!$ corresponds to approximately $(0.36n)^n$). The worst-case state growth of a concrete construction naturally serves as an *upper bound* to a known lower bound. Given Schewe's number $(0.76n)^n n^2$, the lower bound of $(0.76n)^n$ by Yan is regarded as "sharp", as the gap between the lower and upper bound is very narrow, and consequently, the lower bound canot rise much anymore.

Many construction developers aim at bringing the worst-case state growth of their construction close to the currently known lower bound. It goes so far that a construction matching this lower bound is regarded as "optimal".

1.2.2 Need for Empirical Investigation of Actual Performance

Worst-case state growths are interesting from a theoretical point of view, but they are poor guides to the actual performance of a construction [42]. For example, if we have a concrete automaton of, say, 15 states, and we complement it with Schewe's construction, the fact that the worst-case state complexity is $(0.76n)^n n^2$ does not reveal anything about how the construction will perform on this concrete automaton. In any case, we are not expecting the complement to have 1.6 quintillion (1.6×10^{18}) states (which would be the worst case), because this would most likely be practically infeasible.

Furthermore, if a construction has a higher worst-case state growth than another, it does not mean that it performs worse on a concrete case. In fact, worst-case state complexities only allow to adequately deduce the performance on the worst-case automata, but not on all the other automata. However, from a practical point of view, these worst cases are not interesting, as their application in practice is anyway infeasible [**?**] (at least starting from a certain input automaton size).

From a practical perspective we are interested how constructions perform on automata as they could occur in a concrete application of Büchi complementation, such as automata-theoretic model checking. This may include questions like the following. What is a reasonable complement size to expect for the

[4]As mentioned previously, also known as state complexity, state blow-up, or state explosion.

given automaton with n states? Are there generally easier and harder automata? What are the factors that make an automaton especially easy or hard? How does the performance of different constructions on the same automata vary? Are there constructions which are better suited for a certain type of automata than other constructions?

Questions like this can be attempted to answer by empirical peformance investigations. As its two most important elements this includes an *implementation* of the investigated constructions and *test data*. With test data, we mean a set of concrete automata on which the implementations of the constructions are run. The analysis is then done on the generated complement automata.

There have been relatively few empirical attempts in the history of Büchi complementation [42], compared to the number of theoretical works. In the following, we give (non-exhaustive) list of empirical works in the past that illustrate the approach, and also show the line of research in which the work of this thesis is situated.

1995 Tasiran et al. [**?**] create an efficient implementation of Safra' construction[33] (determinisation-based) and used it for for automata-theoretic model checking tasks with the HSIS verification tool [**?**]. They state that they could easily complement property automata with some hundreds of states, however, they do not provide a statistical evaluation of the results.

2003 Gurumurthy et al. [**?**] implement Friedgut, Kupferman, and Vardi's construction [16] (rank-based) along with various optimisations that they propose as a part of the tool Wring [**?**]. They complement 1000 small automata, generated by translation from LTL formulas, and evaluate exeution time, and number of states and transitions of the complement for the different versions of the construction.

2006 Althoff et al. [2] implement Safra's [33] and Muller and Schupp's [24] determinisation constructions[5] in a tool called OmegaDet, applied them on the Michel automata with 2 to 6 states, and compared the number of states of the determinised output automata.

2008 Tsay et al. [45] carry out a first comparative experiment with the publicly available[6] GOAL tool [44][45][46][43]. They include the constructions by Safra [33] (determinisation-based), Piterman [28] (determinisation-based), Thomas [41] (WAPA[7]), and Kupferman and Vardi[16] (rank-based or WAA[8]). These constructions are pre-implemented in GOAL. As the test data, they use 300 Büchi automata, translated from LTL formulas, with an average size of 5.4 states. They evaluate and compare execution times, as well as number of states and transitions of the complements.

2009 Kamarkar and Chakraborty [**?**] propose an improvement of Schewe's construction [35] (rank-based) and implement it, as well as Schewe's original construction, on top of the model checker NuSMV [**?**][**?**]. They run the constructions on 12 test automata and compare the sizes of the complements. Furthermore, they run the same tests with the constructions by Kupferman, and Vardi [16] (rank-based or WAA) and Piterman [28] (determinisation-based) within GOAL, and compare the results to the ones of their implementation of Schewe's construction.

2010 Tsai et al. [42] (paper entitled "State of Büchi Compelentation") carry out another experiment with GOAL. They compare the constructions by Piterman [28] (determinisation-based), Schewe [35] (rank-based), and Vardi and Wilke [52] (slice-based), with various optimsiations that they propose in the same paper. As the test data, they use 11,000 randomly generated automata with 15 states and an alphabet size of 2. The test set is organised into 110 automata classes that consist of the combinations of 11 transition densities and 10 accceptance densities. This test set is repeatedly used in subsequent work (including in this thesis), and we will refer to it as the GOAL test set (because it has been generated with the GOAL tool). Tsai et al. provide sophisticated evaluation of the states of the complements for all the tested constructions and construction versions.

2010 Breuers [**?**] proposes an improvement for the construction by Sistla, Vardi, and Wolper [37] (Ramsey-based), and creates an implementation of it. He generates his own test data (inspired by the work of Tsai et al. [42]) consisting of *easy*, *medium*, and *hard* automata, based on different

[5]These determinisation constructions transform a non-deterministic Büchi automaton to a deterministic Rabin automataon (see Section **??**), however, the are used as the base for determinisation-based complementation constructions.
[6]http://goal.im.ntu.edu.tw/wiki/doku.php
[7]Via **W**eak **A**lternating **P**arity **A**utomaton
[8]Via **W**eak **A**lternating **A**utomaton

transition density and acceptance density values. He evaluates the complement sizes produced by the construction for autoamta of sizes 5, 10, and 15 of all these difficulty categories.

2012 Breuers et al. [3] wrap the implementation of their improvement of Sistla, Vardi, and Wolper's construction [37] in the publicly available tool Alekto[9], and and run it on the GOAL test set. They compare the generated complement sizes, as well as the number of aborted complementation tasks (due to exceeding resource requirements) to the corresponding result for different constructions on the same test set by Tsai et al. [42].

2013 Göttel [8] creates a C implementation of the Fribourg construction [1], including the R2C optimisation (see Chapter **??**), and executes it on the GOAL test set, as well as on the Michel automata with 3 to 6 states. He analyses the resulting complement sizes and execution times separately for each of the 110 classes that the GOAL test set consists of. The Fribourg construction[10] is a slice-based complementation construction that is being developed at the university of Fribourg, and which lies at the heart of this thesis. The entire Chapter **??** of this thesis is dedicated to explaining the Fribourg construction.

1.3 Aim and Scope

The aim of this thesis is an in-depth empirical performance investigation of the Fribourg construction. As mentioned, the Fribourg construction is a Büchi complementation construction that is being developed at the University of Fribourg [1]. By empirically investigating the behaviour of this specific construction, we want to follow up the track of empirical research that we have outlined in the last section.

This thesis is certainly not sufficient to describe the performance of the Fribourg construction in its entiretey, or in a way that is adequate to be relied on in industiral applications. Neither this thesis can answer general questions about the observed behaviour of Büchi complementation. Rather, we see this piece of work as a mosaic stone that we add to the very complex and multi-faceted picture of the complexity of Büchi complementation.

Aim: empirical performance investigation of a specific Büchi complementaiton construction, comparison with other constructions

Scope: two test sets, relatively small automata, no real world or "typical" examples,

1.4 Overview

[9] http://www.automata.rwth-aachen.de/research/Alekto/

[10] The authors of the constructions use the name *subset-tuple construction*, however, in this thesis, we will use the more consise name *Fribourg construction*.

Appendix A

Plugin Installation and Usage

Since between the 2014–08–08 and 2014–11–17 releases of GOAL certain parts of the plugin interfaces have changed, and we adapted our plugin accordingly, the currently maintained version of the plugin works only with GOAL versions 2014-11-17 or newer. It is thus essential for any GOAL user to update to this version in order to use our plugin.

Appendix B

Median Complement Sizes of the GOAL Test Set

Bla bla bla

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
1.0	269	308	254	236	238	297	266	156	207	68
1.2	960	1,407	1,479	2,150	1,152	1,090	942	1,206	718	104
1.4	3,426	2,915	2,752	3,393	2,693	3,265	2,263	2,425	1,844	154
1.6	3,799	3,698	4,901	3,926	3,960	3,655	2,580	1,905	2,124	155
1.8	3,375	3,169	3,420	3,967	3,943	3,132	2,246	1,144	971	114
2.0	1,906	2,261	2,383	2,884	2,354	2,096	1,169	932	568	98
2.2	1,467	1,633	1,795	1,942	1,611	1,640	569	499	330	78
2.4	924	1,232	1,319	1,317	1,056	886	514	314	182	59
2.6	625	763	880	945	828	684	316	175	132	44
2.8	483	584	836	690	575	395	240	151	103	41
3.0	319	450	557	523	367	313	155	116	84	32

(a) Fribourg

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
1.0	269	308	254	236	238	297	266	156	207	68
1.2	960	1,407	1,479	2,150	1,152	1,090	942	1,206	718	104
1.4	3,426	2,915	2,752	3,393	2,693	3,265	2,263	2,425	1,844	154
1.6	3,799	3,698	4,901	3,926	3,960	3,655	2,580	1,905	2,124	155
1.8	3,375	3,169	3,420	3,967	3,943	3,093	2,246	1,144	971	114
2.0	1,906	2,184	2,383	2,818	2,354	1,989	1,127	885	568	97
2.2	1,410	1,561	1,639	1,884	1,609	1,588	496	464	284	78
2.4	884	1,200	1,234	1,184	939	806	373	256	165	55
2.6	575	731	815	860	751	575	246	162	114	43
2.8	431	530	672	466	371	274	174	120	85	36
3.0	232	325	344	360	269	169	91	85	53	27

(b) Fribourg+R2C

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
1.0	390	438	434	324	328	459	337	204	227	40
1.2	1,576	2,394	2,505	2,996	1,613	1,551	1,166	1,542	1,002	58
1.4	5,007	4,336	4,652	4,877	3,458	3,956	3,169	3,380	1,868	86
1.6	5,067	5,032	6,444	4,868	4,575	3,864	3,211	1,731	1,892	85
1.8	4,016	3,701	3,647	4,523	3,548	3,009	1,808	451	336	62
2.0	1,663	2,276	2,676	3,035	1,925	1,932	464	307	150	54
2.2	989	1,514	1,621	1,826	1,121	846	155	127	93	45
2.4	560	821	919	771	529	267	133	87	55	32
2.6	388	519	524	441	259	219	84	50	41	26
2.8	311	317	396	242	165	95	64	44	33	22
3.0	173	224	211	169	102	72	41	34	27	18

(c) Fribourg+R2C+C

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
1.0	225	223	195	181	187	199	189	124	161	68
1.2	731	971	946	1,071	629	562	488	568	388	104
1.4	2,228	1,701	1,543	1,732	1,241	1,287	945	944	727	154
1.6	2,489	2,263	2,331	2,133	1,777	1,443	964	757	889	155
1.8	2,381	2,027	2,009	2,075	1,618	1,243	1,005	592	515	114
2.0	1,390	1,569	1,416	1,573	1,093	1,008	594	464	330	98
2.2	1,118	1,197	1,150	1,151	879	809	317	330	241	78
2.4	712	885	836	809	580	535	316	231	145	59
2.6	498	569	601	627	497	412	217	137	113	44
2.8	391	455	578	456	374	263	173	119	90	41
3.0	258	350	392	354	253	208	119	97	74	32

(d) Fribourg+M1

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
1.0	215	213	189	174	175	192	186	121	156	68
1.2	712	914	913	1,075	619	563	526	620	416	104
1.4	2,075	1,620	1,503	1,650	1,254	1,339	1,003	1,006	848	154
1.6	2,344	2,062	2,340	2,016	1,755	1,520	1,053	858	986	155
1.8	2,205	1,873	1,920	2,040	1,689	1,315	1,080	664	598	114
2.0	1,290	1,485	1,405	1,522	1,134	1,044	652	531	392	98
2.2	1,023	1,119	1,092	1,127	868	875	376	359	262	78
2.4	674	849	790	807	617	544	355	251	156	59
2.6	478	549	594	597	510	431	231	147	116	44
2.8	370	439	559	455	382	283	182	124	93	41
3.0	249	341	388	348	260	225	123	101	77	32

(e) Fribourg+M1+M2

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
1.0	225	223	195	181	187	199	189	124	161	68
1.2	731	971	946	1,071	629	562	488	568	388	104
1.4	2,228	1,701	1,543	1,732	1,241	1,287	945	944	727	154
1.6	2,489	2,263	2,331	2,133	1,777	1,443	964	757	889	155
1.8	2,381	2,027	2,009	2,075	1,618	1,215	1,005	592	515	114
2.0	1,390	1,513	1,416	1,542	1,093	1,003	594	441	330	97
2.2	1,019	1,156	1,064	1,104	859	785	304	303	221	78
2.4	672	867	789	772	544	478	269	191	139	55
2.6	466	542	572	568	452	348	183	129	99	43
2.8	368	407	480	337	260	197	129	96	75	36
3.0	201	261	266	272	199	136	83	74	50	27

(f) Fribourg+M1+R2C

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
1.0	329	303	279	240	229	288	230	157	160	40
1.2	988	1,392	1,356	1,352	751	741	608	704	516	58
1.4	2,939	2,581	2,066	2,190	1,351	1,622	1,132	1,261	932	86
1.6	3,150	2,900	2,842	2,218	1,885	1,563	1,177	821	896	85
1.8	2,782	2,485	2,047	2,180	1,625	1,269	855	395	309	62
2.0	1,338	1,638	1,544	1,566	979	957	349	261	147	54
2.2	838	1,125	993	1,027	667	521	153	125	93	45
2.4	494	700	624	524	296	214	126	87	55	32
2.6	327	434	383	334	212	163	82	50	41	26
2.8	283	273	305	202	144	95	60	44	33	22
3.0	164	200	173	142	92	72	41	34	27	18

(g) Fribourg+M1+R2C+C

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
1.0	126	118	97	60	51	52	62	36	48	30
1.2	432	517	345	262	160	126	92	120	109	40
1.4	1,044	331	133	89	45	22	19	31	27	20
1.6	358	24	11	5	4	6	5	3	3	4
1.8	19	5	1	1	1	1	1	1	1	1
2.0	1	1	1	1	1	1	1	1	1	1
2.2	1	1	1	1	1	1	1	1	1	1
2.4	1	1	1	1	1	1	1	1	1	1
2.6	1	1	1	1	1	1	1	1	1	1
2.8	1	1	1	1	1	1	1	1	1	1
3.0	1	1	1	1	1	1	1	1	1	1

(h) Fribourg+R

Figure B.1: Median complement sizes of the 10,939 effective samples of the internal tests on the GOAL test set. The rows (1.0 to 3.0) are the transition densities, and the columns (0.1 to 1.0) are the acceptance densities.

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
1.0	130	117	109	77	69	61	56	40	40	29
1.2	387	456	352	281	155	136	101	105	75	45
1.4	822	683	394	376	230	204	151	120	105	63
1.6	890	594	458	321	237	178	134	114	113	61
1.8	624	507	324	275	196	136	110	92	89	41
2.0	362	286	211	176	117	103	79	64	59	34
2.2	248	222	124	116	82	73	56	52	50	28
2.4	147	145	114	87	56	48	43	39	35	19
2.6	115	117	67	61	47	42	32	29	29	15
2.8	95	71	52	45	38	29	27	25	23	13
3.0	59	60	47	35	32	27	22	21	20	10

(a) Piterman+EQ+RO

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
1.0	171	174	166	124	118	117	100	67	84	35
1.2	622	833	803	877	529	398	320	372	215	53
1.4	2,086	1,618	1,367	1,676	1,065	967	664	682	494	78
1.6	2,465	2,073	2,182	1,959	1,518	1,259	767	545	623	78
1.8	2,310	1,963	1,950	1,988	1,485	1,095	746	418	346	57
2.0	1,318	1,482	1,393	1,461	981	871	434	338	228	50
2.2	1,068	1,145	1,085	1,067	772	747	263	235	158	40
2.4	689	838	809	751	524	466	240	159	93	30
2.6	469	531	555	565	437	360	169	94	71	23
2.8	369	421	536	405	329	224	130	81	58	21
3.0	244	327	360	322	219	176	85	64	49	16

(b) Slice+P+RO+MADJ+EG

Figure B.2: Median complement sizes of the 10,998 effective samples of the external tests without the Rank construction. The rows (1.0 to 3.0) are the transition densities, and the columns (0.1 to 1.0) are the acceptance densities.

Appendix C

Execution Times

Construction	Mean	Min.	P25	Median	P75	Max.	Total	≈ hours
Fribourg	8.5	2.5	3.3	4.9	7.3	586.0	93,351.2	259
Fribourg+R2C	6.6	2.2	2.9	4.2	6.4	219.7	72,545.7	202
Fribourg+R2C+C	8.5	2.2	2.6	3.5	6.4	582.9	93,396.2	259
Fribourg+M1	4.9	2.5	3.2	4.1	5.9	55.1	54,061.3	150
Fribourg+M1+M2	4.6	2.2	2.9	3.8	5.1	38.4	49,848.0	138
Fribourg+M1+R2C	4.4	2.2	2.8	3.6	5.3	42.5	48,572.0	135
Fribourg+M1+R2C+C	5.6	2.5	3.2	4.0	6.5	147.4	60,918.9	169
Fribourg+R	7.5	2.2	3.0	3.9	6.3	470.5	82,387.3	229

Table C.1: Execution times in CPU time seconds for the 10,939 effectvie samples of the GOAL test set.

Construction	Mean	Min.	P25	Median	P75	Max.	Total	≈ hours
Piterman+EQ+RO	3.0	2.2	2.6	2.8	3.0	42.9	21,410.6	59
Slice+P+RO+MADJ+EG	3.7	2.2	2.7	3.2	4.1	36.7	26,398.9	73
Rank+TR+RO	16.0	2.3	2.8	3.7	9.3	443.3	115,563.9	321
Fribourg+M1+R2C	4.0	2.2	2.7	3.1	4.4	410.4	28,970.8	80

Table C.2: Execution times in CPU time seconds for the 7,204 effectvie samples of the GOAL test set.

Construction	Mean	Min.	P25	Median	P75	Max.	Total	≈ hours
Piterman+EQ+RO	3.6	2.2	2.7	2.9	3.4	365.7	39,663.4	110
Slice+P+RO+MADJ+EG	4.3	2.2	2.9	3.7	5.0	42.4	47,418.2	132
Fribourg+M1+R2C	4.7	2.2	2.8	3.6	5.3	410.4	52,149.0	145

Table C.3: Execution times in CPU time seconds for the 10,998 effectvie samples of the GOAL test set without the Rank construction.

Construction	Michel 1	Michel 2	Michel 3	Michel 4	Fitted curve	Std. error
Fribourg	2.3	4.0	88.8	100,976.0	$(1.14n)^n$	0.64%
Fribourg+R2C	2.3	3.4	27.4	27,938.3	$(0.92n)^n$	0.64%
Fribourg+M1	2.2	3.6	17.9	6,508.4	$(0.72n)^n$	0.63%
Fribourg+M1+M2	2.3	3.5	13.8	2,707.4	$(0.62n)^n$	0.62%
Fribourg+M1+M2+R2C	2.5	3.5	10.8	2,332.6	$(0.61n)^n$	0.62%
Fribourg+R	2.4	3.7	86.0	101,809.6	$(1.14n)^n$	0.64%

Table C.4: Execution times in CPU time seconds for the four Michel automata.

Construction	Michel 1	Michel 2	Michel 3	Michel 4	Fitted curve	Std. error
Piterman+EQ+RO	2.5	3.8	42.6	75,917.4	$(1.08n)^n$	0.64%
Slice+P+RO+MADJ+EG	2.3	3.6	11.4	159.5	$(0.39n)^n$	0.38%
Rank+TR+RO	2.2	3.0	6.4	30.0	$(0.29n)^n$	0.18%
Fribourg+M1+M2+R2C	2.5	3.5	10.8	2,332.6	$(0.61n)^n$	0.62%

Table C.5: Execution times in CPU time seconds for the four Michel automata.

Bibliography

[1] J. Allred, U. Ultes-Nitsche. Complementing Büchi Automata with a Subset-Tuple Construction. Tech. rep.. University of Fribourg, Switzerland. 2014.

[2] C. Althoff, W. Thomas, N. Wallmeier. Observations on Determinization of Büchi Automata. In J. Farré, I. Litovsky, S. Schmitz, eds., *Implementation and Application of Automata*. vol. 3845 of *Lecture Notes in Computer Science*. pp. 262–272. Springer Berlin Heidelberg. 2006.

[3] S. Breuers, C. Löding, J. Olschewski. Improved Ramsey-Based Büchi Complementation. In L. Birkedal, ed., *Foundations of Software Science and Computational Structures*. vol. 7213 of *Lecture Notes in Computer Science*. pp. 150–164. Springer Berlin Heidelberg. 2012.

[4] J. R. Büchi. On a Decision Method in Restricted Second Order Arithmetic. In *Proc. International Congress on Logic, Method, and Philosophy of Science, 1960*. Stanford University Press. 1962.

[5] S. J. Fogarty, O. Kupferman, T. Wilke, et al. Unifying Büchi Complementation Constructions. *Logical Methods in Computer Science*. 9(1). 2013.

[6] E. Friedgut, O. Kupferman, M. Vardi. Büchi Complementation Made Tighter. In F. Wang, ed., *Automated Technology for Verification and Analysis*. vol. 3299 of *Lecture Notes in Computer Science*. pp. 64–78. Springer Berlin Heidelberg. 2004.

[7] E. Friedgut, O. Kupferman, M. Y. Vardi. Büchi Complementation Made Tighter. *International Journal of Foundations of Computer Science*. 17(04):pp. 851–867. 2006.

[8] C. Göttel. Implementation of an Algorithm for Büchi Complementation. BSc Thesis, University of Fribourg, Switzerland. November 2013.

[9] R. L. Graham, B. L. Rothschild, J. H. Spencer. *Ramsey theory*. vol. 20. John Wiley & Sons. 1990.

[10] J. E. Hopcroft, R. Motwani, J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley. 2nd edition ed.. 2001.

[11] D. Kähler, T. Wilke. Complementation, Disambiguation, and Determinization of Büchi Automata Unified. In L. Aceto, I. Damgård, L. Goldberg, et al, eds., *Automata, Languages and Programming*. vol. 5125 of *Lecture Notes in Computer Science*. pp. 724–735. Springer Berlin Heidelberg. 2008.

[12] N. Klarlund. Progress measures for complementation of omega-automata with applications to temporal logic. In *Foundations of Computer Science, 1991. Proceedings., 32nd Annual Symposium on*. pp. 358–367. Oct 1991.

[13] J. Klein. Linear Time Logic and Deterministic Omega-Automata. *Master's thesis, Universität Bonn*. 2005.

[14] J. Klein, C. Baier. Experiments with Deterministic ω-Automata for Formulas of Linear Temporal Logic. In J. Farré, I. Litovsky, S. Schmitz, eds., *Implementation and Application of Automata*. vol. 3845 of *Lecture Notes in Computer Science*. pp. 199–212. Springer Berlin Heidelberg. 2006.

[15] O. Kupferman, M. Y. Vardi. Weak Alternating Automata Are Not that Weak. In *Proceedings of the 5th Israeli Symposium on Theory of Computing and Systems*. pp. 147–158. IEEE Computer Society Press. 1997.

[16] O. Kupferman, M. Y. Vardi. Weak Alternating Automata Are Not that Weak. *ACM Trans. Comput. Logic.* 2(3):pp. 408–429. Jul. 2001.

[17] R. Kurshan. Complementing Deterministic Büchi Automata in Polynomial Time. *Journal of Computer and System Sciences.* 35(1):pp. 59 – 71. 1987.

[18] C. Löding. Optimal Bounds for Transformations of ω-Automata. In C. Rangan, V. Raman, R. Ramanujam, eds., *Foundations of Software Technology and Theoretical Computer Science.* vol. 1738 of *Lecture Notes in Computer Science.* pp. 97–109. Springer Berlin Heidelberg. 1999.

[19] R. McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control.* 9(5):pp. 521 – 530. 1966.

[20] M. Michel. Complementation is more difficult with automata on infinite words. *CNET, Paris.* 15. 1988.

[21] A. Mostowski. Regular expressions for infinite trees and a standard form of automata. In A. Skowron, ed., *Computation Theory.* vol. 208 of *Lecture Notes in Computer Science.* pp. 157–168. Springer Berlin Heidelberg. 1985.

[22] D. E. Muller. Infinite Sequences and Finite Machines. In *Switching Circuit Theory and Logical Design, Proceedings of the Fourth Annual Symposium on.* pp. 3–16. Oct 1963.

[23] D. E. Muller, A. Saoudi, P. E. Schupp. Alternating automata, the weak monadic theory of the tree, and its complexity. In L. Kott, ed., *Automata, Languages and Programming.* vol. 226 of *Lecture Notes in Computer Science.* pp. 275–283. Springer Berlin Heidelberg. 1986.

[24] D. E. Muller, P. E. Schupp. Simulating Alternating Tree Automata by Nondeterministic Automata: New Results and New Proofs of the Theorems of Rabin, McNaughton and Safra. *Theoretical Computer Science.* 141(1–2):pp. 69 – 107. 1995.

[25] F. Nießner, U. Nitsche, P. Ochsenschläger. Deterministic Omega-Regular Liveness Properties. In S. Bozapalidis, ed., *Preproceedings of the 3rd International Conference on Developments in Language Theory, DLT'97.* pp. 237–247. Citeseer. 1997.

[26] J.-P. Pecuchet. On the complementation of Büchi automata. *Theoretical Computer Science.* 47(0):pp. 95 – 98. 1986.

[27] N. Piterman. From Nondeterministic Buchi and Streett Automata to Deterministic Parity Automata. In *Logic in Computer Science, 2006 21st Annual IEEE Symposium on.* pp. 255–264. 2006.

[28] N. Piterman. From Nondeterministic Buchi and Streett Automata to Deterministic Parity Automata. *Logical Methods in Computer Science.* 3(5):pp. 1–21. 2007.

[29] M. Rabin, D. Scott. Finite Automata and Their Decision Problems. *IBM Journal of Research and Development.* 3(2):pp. 114–125. April 1959.

[30] M. O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society.* 141:pp. 1–35. July 1969.

[31] F. P. Ramsey. On a Problem of Formal Logic. *Proceedings of the London Mathematical Society.* s2-30(1):pp. 264–286. 1930.

[32] M. Roggenbach. Determinization of Büchi-Automata. In E. Grädel, W. Thomas, T. Wilke, eds., *Automata Logics, and Infinite Games.* vol. 2500 of *Lecture Notes in Computer Science.* pp. 43–60. Springer Berlin Heidelberg. 2002.

[33] S. Safra. On the Complexity of Omega-Automata. *Journal of Computer and System Science.* 1988.

[34] S. Safra. On the Complexity of Omega-Automata. In *Foundations of Computer Science, 1988., 29th Annual Symposium on.* pp. 319–327. Oct 1988.

[35] S. Schewe. Büchi Complementation Made Tight. In *26th International Symposium on Theoretical Aspects of Computer Science-STACS 2009.* pp. 661–672. 2009.

[36] A. Sistla, M. Vardi, P. Wolper. The complementation problem for Büchi automata with applications to temporal logic. In W. Brauer, ed., *Automata, Languages and Programming*. vol. 194 of *Lecture Notes in Computer Science*. pp. 465–474. Springer Berlin Heidelberg. 1985.

[37] A. P. Sistla, M. Y. Vardi, P. Wolper. The Complementation Problem for Büchi Automata with Applications to Temporal Logic. *Theoretical Computer Science*. 49(2–3):pp. 217 – 237. 1987.

[38] R. S. Streett. Propositional dynamic logic of looping and converse is elementarily decidable. *Information and Control*. 54(1–2):pp. 121 – 141. 1982.

[39] W. Thomas. Automata on Infinite Objects. In J. van Leeuwen, ed., *Handbook of Theoretical Computer Science (Vol. B)*. chap. Automata on Infinite Objects, pp. 133–191. MIT Press, Cambridge, MA, USA. 1990.

[40] W. Thomas. Languages, Automata, and Logic. In G. Rozenberg, A. Salomaa, eds., *Handbook of Formal Languages*. pp. 389–455. Springer Berlin Heidelberg. 1997.

[41] W. Thomas. Complementation of Büchi Automata Revisited. In J. Karhumäki, H. Maurer, G. Păun, et al, eds., *Jewels are Forever*. pp. 109–120. Springer Berlin Heidelberg. 1999.

[42] M.-H. Tsai, S. Fogarty, M. Vardi, et al. State of Büchi Complementation. In M. Domaratzki, K. Salomaa, eds., *Implementation and Application of Automata*. vol. 6482 of *Lecture Notes in Computer Science*. pp. 261–271. Springer Berlin Heidelberg. 2011.

[43] M.-H. Tsai, Y.-K. Tsay, Y.-S. Hwang. GOAL for Games, Omega-Automata, and Logics. In N. Sharygina, H. Veith, eds., *Computer Aided Verification*. vol. 8044 of *Lecture Notes in Computer Science*. pp. 883–889. Springer Berlin Heidelberg. 2013.

[44] Y.-K. Tsay, Y.-F. Chen, M.-H. Tsai, et al. Goal: A Graphical Tool for Manipulating Büchi Automata and Temporal Formulae. In O. Grumberg, M. Huth, eds., *Tools and Algorithms for the Construction and Analysis of Systems*. vol. 4424 of *Lecture Notes in Computer Science*. pp. 466–471. Springer Berlin Heidelberg. 2007.

[45] Y.-K. Tsay, Y.-F. Chen, M.-H. Tsai, et al. Goal Extended: Towards a Research Tool for Omega Automata and Temporal Logic. In C. Ramakrishnan, J. Rehof, eds., *Tools and Algorithms for the Construction and Analysis of Systems*. vol. 4963 of *Lecture Notes in Computer Science*. pp. 346–350. Springer Berlin Heidelberg. 2008.

[46] Y.-K. Tsay, Y.-F. Chen, M.-H. Tsai, et al. Tool support for learning Büchi automata and linear temporal logic. *Formal Aspects of Computing*. 21(3):pp. 259–275. 2009.

[47] Y.-K. Tsay, M.-H. Tsai, J.-S. Chang, et al. Büchi Store: An Open Repository of Büchi Automata. In P. Abdulla, K. Leino, eds., *Tools and Algorithms for the Construction and Analysis of Systems*. vol. 6605 of *Lecture Notes in Computer Science*. pp. 262–266. Springer Berlin Heidelberg. 2011.

[48] U. Ultes-Nitsche. A Power-Set Construction for Reducing Büchi Automata to Non-Determinism Degree Two. *Information Processing Letters*. 101(3):pp. 107 – 111. 2007.

[49] M. Vardi. An automata-theoretic approach to linear temporal logic. In F. Moller, G. Birtwistle, eds., *Logics for Concurrency*. vol. 1043 of *Lecture Notes in Computer Science*. pp. 238–266. Springer Berlin Heidelberg. 1996.

[50] M. Vardi. The Büchi Complementation Saga. In W. Thomas, P. Weil, eds., *STACS 2007*. vol. 4393 of *Lecture Notes in Computer Science*. pp. 12–22. Springer Berlin Heidelberg. 2007.

[51] M. Y. Vardi. Buchi Complementation: A Forty-Year Saga. In *5th symposium on Atomic Level Characterizations (ALC'05)*. 2005.

[52] M. Y. Vardi, T. Wilke. Automata: From Logics to Algorithms. In J. Flum, E. Grädel, T. Wilke, eds., *Logic and Automata: History and Perspectives*. vol. 2 of *Texts in Logic and Games*. pp. 629–736. Amsterdam University Press. 2007.

[53] T. Wilke. ω-Automata. In J.-E. Pin, ed., *Handbook of Automata Theory*. European Mathematical Society. To appear, 2015.

[54] Q. Yan. Lower Bounds for Complementation of ω-Automata Via the Full Automata Technique. In M. Bugliesi, B. Preneel, V. Sassone, et al, eds., *Automata, Languages and Programming*. vol. 4052 of *Lecture Notes in Computer Science*. pp. 589–600. Springer Berlin Heidelberg. 2006.

[55] Q. Yan. Lower Bounds for Complementation of omega-Automata Via the Full Automata Technique. *CoRR*. abs/0802.1226. 2008.