Performance Investigation of a Subset-Tuple Büchi Complementation Construction

Daniel Weibel

October 23, 2014

Contents

1 Introduction

A Büchi complementation construction takes as input a Büchi automaton A and produces as output another Büchi automaton B which accepts the complement language of the input automaton A. Complement language denotes the "contrary" language, that is, B must *accept* (over a given alphabet) every word that A *does not* accept, and must in turn *not accept* every word that A *accepts*.

Büchi automata are finite automata (that is, having a finite number of states) which operate on infinite words (that is, words that "never end"). Operating on infinite words, they belong thus to the category ω-automata. An important application of Büchi automata is in model checking which is a formal system verification technique. There, they are used to represent both, the description of the system to be checked for the presence of a correctness property, and (the negation of) this correctness property itself.

In one approach to model checking, the correctness property is directly specified as a Büchi automaton One approach to model checking requires that the Büchi automaton representing the correctness property is complemented. It is here that the problem of Büchi complementation has one of its practical applications.

The complementation of non-deterministic Büchi automata is hard. It has been proven to have an exponential lower bound in the number of generated states [cite]. That is, the number of states of the output automaton is, in the worst case, an exponential function of the number of states of the input automaton. However, since the introduction of Büchi automata in the 1960's, significant process in reducing the complexity (in other words, the degree of exponentiality) of the Büchi complementation problem has been made. Some numbers [list complexities of the different constructions].

2 The Büchi Complementation Problem

2.1 Preliminaries

2.1.1 Büchi Automata

Definition

Informally speaking, a Büchi automaton is a finite state automaton running on input words of infinite length. That is, once started reading a word, a Büchi automaton never stops. A word is accepted by a Büchi automaton, if there exists a run (sequence of states) for it that includes infinite repetitions of at least one accepting state. In other words, if during reading a word a Büchi automaton goes through one or more accepting states infinitely often, the word is accepted, and otherwise it is rejected.

More formally, a Büchi automaton A consists of a 5-tuple $A = (\Sigma, Q, q_0, \delta, F)$, the components of which are the following.

- Σ: a finite alphabet
- Q: a finite set of states
- q_0: an initial state, $q_0 \in Q$
- δ: a transition function $q : Q \times \Sigma \to 2^Q$
- F: a set of accepting states, $F \in 2^Q$

This is so far the same definition as for normal finite state automata. What distinguishes a Büchi automaton from a normal finite state automata is its acceptance condition. To define the Büchi acceptance condition, we first have to define the following notions.

- Σ^ω isthe set of all possible words of infinite length over an alphabet Σ
- A *run* of Büchi automaton A on a word $x \in \Sigma^\omega$ is a sequence of states $q_0 q_1 q_3 \ldots$ such that q_0 is A's initial state and $\forall i \geq 0 : q_{i+1} \in \delta(q_i, x_i)$
- $\inf(\rho) \in 2^Q$ is the subset of states of Q that occur infinitely often in a run ρ
- An *accepting run* is a run ρ for which $\inf(\rho) \cap F \neq \varnothing$

A word $x \in \Sigma^\omega$ is then accepted by a Büchi automaton A if and only if there exists an accepting run of A on x.

Deterministic and Non-Deterministic Büchi Automata

As for normal finite state automata, there are deterministic and non-deterministic versions of Büchi automata. The difference lies in the transition function. For the non-deterministic case it is $\delta : \Sigma \times Q \to 2^Q$, but for the deterministic case it is strictly $\delta : \Sigma \times Q \to Q$. That means, whereas in a non-deterministic Büchi automaton a state q may have zero, one, or more successors on a given symbol a (denoted by $\delta(q, a) \geq 0$), in a deterministic Büchi automaton this number is always exactly one ($\delta(q, a) = 1$).

The definition of a Büchi automaton given above is thus in fact the definition of a non-deterministic Büchi automaton. This coincides with a convention that we adopt in this thesis. If not explicitly stated, when we say "Büchi automaton" we actually mean a non-deterministi Büchi automaton. This is first, because deterministic Büchi automata are a special case of non-deterministic Büchi automata, and second, the problem of complementation that this thesis is about is only significant for the non-deterministic case.

A Büchi automaton is complete if every state has at least one outgoing transition for every symbol of the alphabet. Formally, this means that $|\delta(q, a)| \geq 1, \forall q \in Q, \forall a \in \Sigma$. Note that deterministic Büchi automata are always complete, and thus only non-deterministic Büchi automata can be incomplete

Equivalences

An important property of Büchi automata is that deterministic Büchi automata are less expressive than non-deterministic Büchi automata. That means that there exist non-deterministic Büchi automata for which no deterministic Büchi automata accepting the same language exists.

Non-deterministic Büchi automata in turn are equivalent to the ω-regular languages. That means that every language that is recognised by any Büchi automaton is an ω-regular language, and for every ω-regular language there exists a non-deterministic Büchi automaton recognising it.

Furthermore, non-deterministic Büchi automata are equivalent to other classes of ω-automata such as Muller, Rabin, Streett, and Parity automata. Within these classes, determinstic and non-deterministic automata are equivalent. This means that every Büchi automaton can be translated to an equivalent deterministic or non-deterministic Muller, Rabin, Streett, or Parity automaton, and any of these latter automata can be translated to a non-deterministic Büchi automaton. Figure 2.1 summarises these expressive relations of Büchi automata.

2.1.2 Other ω-Automata

2.1.3 Complementation of Büchi Automata

Büchi automata are closed under complementation. This result has been proved by Büchi himself when he introduced Büchi automata in [1]. Basically, this means that for

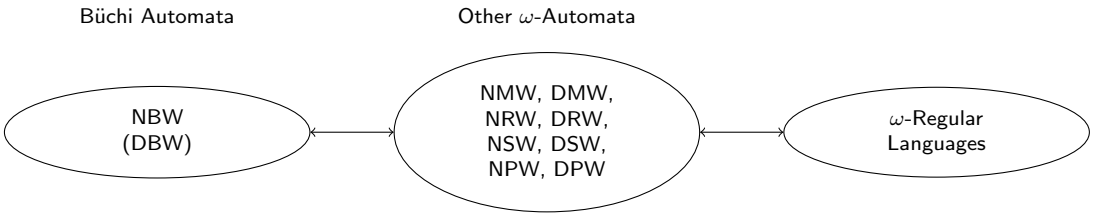

Figure 2.1: Non-deterministic Büchi automata (NBW) are expressively equivalent to Muller, Rabin, Streett, and parity automata (both deterministic and non-deterministic), and to the ω-regular languages. Deterministic Büchi automata (DBW) are less expressive than NBW.

every Büchi automata A, there exists another Büchi automaton B that recognises the complement language of A, that is, $L(B) = \overline{L(A)}$.

It is interesting to see that this closure does not hold for the specific case of DBW. That means that while for every DBW a complement Büchi automaton does indeed exist, following from the above closure property for Büchi automata in general, this automaton is not necessarily a DBW. The complement of a DBW may be, and often is, as we will see, a NBW. This result is proved in [7] (p. 15).

The problem of Büchi complementation consists now in finding a procedure (usually called a construction) that takes as input any Büchi automaton A and outputs another Büchi automaton B with $L(B) = \overline{L(A)}$, as shown below.

For complementation of automata in general, construction usually differ depending on whether the input automaton A is deterministic or non-deterministic. Complementation of deterministic automata is often simpler and may sometimes even provide a solution for the complementation of the non-deterministic ones.

To illustrate this, we can briefly look at the complementation of the ordinary finite state automata on finite words (FA). FA are also closed under complementation [2] (p. 133). A DFA can be complemented by simply switching its accepting and non-accepting states [2] (p. 133). Now, since NFA and DFA are equivalent [2] (p. 60), a NFA can be complemented by converting it to an equivalent DFA first, and then complement this DFA. Thus, the complementation construction for DFA provides a solution for the complementation of NFA.

Returning to Büchi automata, the case is more complicated due to the inequivalence of NBW and DBW. The complementation of DBW is indeed "easy", as was the complementation of DFA. There is a construction, introduced in 1987 by Kurshan [3], that can complement a DBW to a NBW in polynomial time. The size of the complement NBW is furthermore at most the double of the size of the input DBW.

If now for every NBW there would exist an equivalent DBW, an obvious solution to the general Büchi complementation problem would be to transform the input automaton

to a DBW (if it is not already a DBW) and then apply Kurshan's construction to the DBW. However, as we have seen, this is not the case. There are NBW that cannot be turned into equivalent DBW.

Hence, for NBW, other ways of complementing them have to be found. In the next section we will review the most important of these "other ways" that have been proposed in the last 50 years since the introduction of Büchi automata. The Fribourg construction, that we present in Chapter 3, is another alternative way of achievin this same aim.

2.1.4 State Complexity of Büchi Complementation

Constructions for complementing NBW turned out to be very complex. Especially the blow-up in number of states from the input automaton to the output automaton is significant. For example, the original complementation construction proposed by Büchi [1] involved a doubly exponential blow-up. That is, if the input automaton has n states, then for some constant c the output automaton has, in the worst case, c^{c^n} states [6]. If we set c to 2, then an input automaton with six states would result in a complement automaton with about 18 quintillion (18×10^{18}) states.

Generally, state blow-up functions, like the c^{c^n} above, mean the absolute worst cases. It is the maximum number of states a construction *can* produce. For by far most input automata of size n a construction will produce much fewer states. Nevertheless, worst case state blow-ups are an important (the most important?) performance measure for Büchi complementation constructions. A main goal in the development of new constructions is to bring this number down.

A question that arises is, how much this number can be brought down? Researchers have investigated this question by trying to establish so called lower bounds. A lower bound is a function for which it is proven that no state blow-up of any construction can be less than it. The first lower bound for Büchi complementation has been established by Michel in 1988 at $n!$ [4]. This means that the state blow-up of any Büchi complementation construction can never be less than $n!$.

There are other notations that are often used for state blow-ups. One has the form $(xn)^n$, where x is a constant. Michel's bound of $n!$ would be about $(0.36n)^n$ in this case [8]. We will often use this notation, as it is convenient for comparisons. Another form has 2 as the base and a big-O term in the exponent. In this case, Michel's $n!$ would be $2^{O(n \log n)}$ [8].

Michel's lower bound remained valid for almost two decades until in 2006 Yan showed a new lower bound of $(0.76n)^n$ [8]. This does not mean that Michel was wrong with his lower bound, but just too reserved. The best possible blow-up of a construction can now be only $(0.76n)^n$ and not $(0.36n)^n$ as believed before. In 2009, Schewe proposed a construction with a blow-up of exactly $(0.76n)^n$ (modulo a polynomial factor) [5]. He provided thus an upper bound that matches Yan's lower bound. The lower bound of $(0.76n)^n$ can thus not rise any further and seems to be definitve.

2.2 Review of Proposed Büchi Complementation Algorithms

2.2.1 Ramsey-Based Approaches

2.2.2 Determinisation-Based Approaches

2.2.3 Rank-Based Approaches

2.2.4 Slice-Based Approaches

2.3 Empirical Performance Investigations

3 The Fribourg Construction

4 Performance Investigation of the Fribourg Construction

5 Results

6 Discussion

7 Conclusions

Bibliography

[1] 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.

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

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

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

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

[6] 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.

[7] 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.

[8] 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.