State of Büchi Complementation

Ming-Hsien Tsai\textsuperscript{1}, Seth Fogarty\textsuperscript{2}, Moshe Y. Vardi\textsuperscript{2}, and Yih-Kuen Tsay\textsuperscript{1}

\textsuperscript{1} National Taiwan University \quad \textsuperscript{2} Rice University

Abstract. Büchi complementation has been studied for five decades since the formalism was introduced in 1960. Known complementation constructions can be classified into Ramsey-based, determinization-based, rank-based, and slice-based approaches. For the performance of these approaches, there have been several complexity analyses but very few experimental results. What especially lacks is a comparative experiment on all the four approaches to see how they perform in practice. In this paper, we review the state of Büchi complementation, propose several optimization heuristics, and perform comparative experimentation on the four approaches. The experimental results show that the determinization-based Safra-Piterman construction outperforms the other three and our heuristics substantially improve the Safra-Piterman construction and the slice-based construction.

1 Introduction

Büchi automata are nondeterministic finite automata on infinite words that recognize \(\omega\)-regular languages. It is known that Büchi automata are closed under Boolean operations, namely union, intersection, and complementation. Complementation was first studied by Büchi in 1960 for a decision procedure for second-order logic \cite{Buchi60}. Complementation of Büchi automata is significantly more complicated than that of nondeterministic finite automata on finite words. Given a nondeterministic finite automaton on finite words with \(n\) states, complementation yields an automaton with \(2^n\) states through the subset construction. Indeed, for nondeterministic Büchi automata, the subset construction is insufficient for complementation. In fact, Michel showed in 1988 that blow-up of Büchi complementation is at least \(n!\) (approximately \((n/e)^n\) or \((0.36n)^n\)), which is much higher than \(2^n\) \cite{Michel88}. This lower bound was later sharpened by Yan to \((0.76n)^n\) \cite{Yan99}, which was matched by an upper bound by Schewe \cite{Schewe04}.

There are several applications of Büchi complementation in formal verification, for example, verifying whether a system satisfies a property by checking if the intersection of the system automaton and the complement of the property automaton is empty \cite{Kupferman10}, testing the correctness of an LTL translation algorithm without a reference algorithm, etc. \cite{Kupferman04}. Although recently many works focus on

\* Work supported in part by the National Science Council, Taiwan (R.O.C.) under grant NSC97-2221-E-002-074-MY3, by NSF grants CCF-0613889, ANI-0216467, CCF-0728882, and OISE-0913807, by BSF grant 9800096, and by gift from Intel.
universality and containment testing without going explicitly through complementation [5, 6, 4], it is still unavoidable in some cases [16].

Known complementation constructions can be classified into four approaches: Ramsey-based approach [3, 22], determinization-based approach [20, 18, 2, 19], rank-based approach [24, 15, 13], and slice-based approach [10, 30]. The first three approaches were reviewed in [29]. Due to the high complexity of Büchi complementation, optimization heuristics are critical to good performance [9, 7, 21, 11, 14]. Unlike the rich theoretical development, empirical studies of Büchi complementation have been rather few [14, 9, 11, 26], as much recent emphasis has shifted to universality and containment. A comprehensive empirical study would allow us to evaluate the performance of these complementation approaches.

In this paper, we review the four complementation approaches and perform comparative experimentation on the best construction in each approach. Although the conventional wisdom is that the nondeterministic constructions are better than the deterministic construction, due to better worse-case bounds, the experimental results show that the deterministic construction is the best for complementation in general. At the same time, the Ramsey-based approach, which is competitive in universality and containment testing [11, 6], performs rather poorly in our complementation experiments. We also propose optimization heuristics for the determinization-based construction, the rank-based construction, and the slice-based construction. The experiment shows that the optimization heuristics substantially improve the three constructions. Overall, our work confirms the importance of experimentation and heuristics in studying Büchi complementation, as worst-case bounds are poor guides to actual performance.

This paper is organized as follows. Some preliminaries are given in Section 2. In Section 3 we review the four complementation approaches. We discuss the results of our comparative experimentation on the four approaches in Section 4. Section 5 describes our optimization heuristics and Section 6 shows the improvement made by our heuristics. We conclude in Section 7. More results of the experiments in Section 4 and Section 6 and further technical details regarding some of the heuristics can be found in [25].

2 Preliminaries

A nondeterministic ω-automaton $A$ is a tuple $(\Sigma, Q, q_0, \delta, F)$, where $\Sigma$ is the finite alphabet, $Q$ is the finite state set, $q_0 \in Q$ is the initial state, $\delta : Q \times \Sigma \rightarrow 2^Q$ is the transition function, and $F$ is the acceptance condition, to be described subsequently. $A$ is deterministic if $|\delta(q, a)| = 1$ for all $q \in Q$ and $a \in \Sigma$.

Given an ω-automaton $A = (\Sigma, Q, q_0, \delta, F)$ and an infinite word $w = a_0a_1 \cdots \in \Sigma^\omega$, a run $\rho$ of $A$ on $w$ is a sequence $q_0q_1 \cdots \in Q^\omega$ satisfying $\forall i : q_{i+1} \in \delta(q_i, a_i)$. A run is accepting if it satisfies the acceptance condition. A word is accepted if there is an accepting run on it. The language of an ω-automaton $A$, denoted by $L(A)$, is the set of words accepted by $A$. An ω-automaton $A$ is universal if $L(A) = \Sigma^\omega$. A state is live if it occurs in an accepting run on some word, and is dead otherwise. Dead states can be discovered using a nonemptiness algorithm, cf. [28], and can be pruned off without affecting the language of the automaton.