# The Büchi Complementation Saga

Moshe Y. Vardi*

Rice University, Department of Computer Science, Rice University, Houston, TX 77251-1892, U.S.A., Email: vardi@cs.rice.edu, URL: http://www.cs.rice.edu/~vardi

**Abstract.** The complementation problem for nondeterministic word automata has numerous applications in formal verification. In particular, the language-containment problem, to which many verification problems are reduced, involves complementation. For automata on finite words, which correspond to safety properties, complementation involves determinization. The $2^n$ blow-up that is caused by the subset construction is justified by a tight lower bound. For Büchi automata on infinite words, which are required for the modeling of liveness properties, optimal complementation constructions are quite complicated, as the subset construction is not sufficient. We review here progress on this problem, which dates back to its introduction in Büchi's seminal 1962 paper.

## 1 Introduction

The complementation problem for nondeterministic word automata has numerous applications in formal verification. In order to check that the language of an automaton $\mathcal{A}_1$ is contained in the language of a second automaton $\mathcal{A}_2$, one checks that the intersection of $\mathcal{A}_1$ with an automaton that complements $\mathcal{A}_2$ is empty. Many problems in verification and design are reduced to language containment. In model checking, the automaton $\mathcal{A}_1$ corresponds to the system, and the automaton $\mathcal{A}_2$ corresponds to the property we wish to verify [21, 37]. While it is easy to complement properties given in terms of formulas in temporal logic, complementation of properties given in terms of automata is not simple. Indeed, a word $w$ is rejected by a nondeterministic automaton $\mathcal{A}$ if *all* runs of $\mathcal{A}$ on $w$ rejects the word. Thus, the complementary automaton has to consider all possible runs, and complementation has the flavor of determinization.

For automata on finite words, determinization, and hence also complementation, is done via the subset construction [28]. Accordingly, if we start with a nondeterministic automaton with $n$ states, the complementary automaton may have $2^n$ states. The exponential blow-up that is caused by the subset construction is justified by a tight lower bound: it is proved in [31] that for every $n > 1$, there exists a language $L_n$ that is recognized by a nondeterministic automaton with $n$ states, yet a nondeterministic automaton for the complement of $L_n$ has at least $2^n$ states (see also [2]).

* Supported in part by NSF grants CCR-9988322, CCR-0124077, CCR-0311326, and ANI-0216467, by BSF grant 9800096, and by a grant from the Intel Corporation. This paper is based on joint work with Orna Kupferman.

For Büchi automata on infinite words, which are required for the modeling of liveness properties, optimal complementation constructions are quite complicated, as the subset construction is not sufficient (but see erroneous claim in [25]). Due to the lack of a simple complementation construction, the user is typically required to specify the property by a deterministic Büchi automaton [21] (it is easy to complement a deterministic Büchi automaton), or to supply the automaton for the negation of the property [14]. Similarly, specification formalisms like ETL [38], which have automata within the logic, involve complementation of automata, and the difficulty of complementing Büchi automata is an obstacle to practical use [1]. In fact, even when the properties are specified in LTL, complementation is useful: the translators from LTL into automata have reached a remarkable level of sophistication (c.f., [5, 33, 10, 11]). Even though complementation of the automata is not explicitly required, the translations are so involved that it is useful to checks their correctness, which involves complementation[1]. Complementation is interesting in practice also because it enables refinement and optimization techniques that are based on language containment rather than simulation [21][2]. Thus, an effective algorithm for the complementation of Büchi automata would be of significant practical value.

Efforts to develop complementation constructions for nondeterministic Büchi automata started early in the 60s, motivated by decision problems of second-order logics. Büchi introduced these automata in 1962 and described a complementation construction that involved a Ramsey-based combinatorial argument and a doubly-exponential blow-up in the state space [3]. Thus, complementing an automaton with $n$ states resulted in an automaton with $2^{2^{O(n)}}$ states. In [32], an improved implementation of Büchi's construction is described, with only $2^{O(n^2)}$ states (see also [27]). Finally, in [29], Safra described a determinization construction, which also enables an $O(n^{O(n)})$ complementation construction, matching a lower bound of $n!$ described by Michel [23] (cf. [22]). Thus, from a theoretical point of view, some considered the problem solved since 1988, since we seem to have matching asymptotic upper and lower bounds.

Nevertheless, a careful analysis of the exact blow-up in Safra's and Michel's bounds reveals an exponential gap in the constants hiding in the $O()$ notations: while the upper bound on the number of states in the complementary automaton constructed by Safra is $n^{2n}$, Michel's lower bound involves only an $n!$ blow up, which is roughly $(n/e)^n$. This is in contrast with the case of automata on finite words, where, as mentioned above, the upper and lower bounds coincide. In the rest of this paper we describe more recent efforts to narrow this gap.

[1] For an LTL formula $\psi$, one typically checks that both the intersection of $\mathcal{A}_\psi$ with $\mathcal{A}_{\neg\psi}$ and the intersection of their complementary automata are empty.

[2] Since complementation of Büchi automata is complicated, current research is focused on ways in which fair simulation can approximate language containment [13], and ways in which the complementation construction can be circumvented by manually bridging the gap between fair simulation and language containment [15].

## 2 Background

Given an alphabet $\Sigma$, an *infinite word over* $\Sigma$ is an infinite sequence $w = \sigma_0 \cdot \sigma_1 \cdot \sigma_2 \cdots$ of letters in $\Sigma$. An *automaton on infinite words* is $\mathcal{A} = \langle \Sigma, Q, Q_{in}, \rho, \alpha \rangle$, where $\Sigma$ is the input alphabet, $Q$ is a finite set of states, $\rho : Q \times \Sigma \to 2^Q$ is a transition function, $Q_{in} \subseteq Q$ is a set of initial states, and $\alpha$ is an acceptance condition (a condition that defines a subset of $Q^\omega$). Intuitively, $\rho(q, \sigma)$ is the set of states that $\mathcal{A}$ can move into when it is in state $q$ and it reads the letter $\sigma$. Since the transition function of $\mathcal{A}$ may specify many possible transitions for each state and letter, $\mathcal{A}$ is not *deterministic*.

A *run* of $\mathcal{A}$ on $w$ is a function $r : \mathbb{N} \to Q$ where $r(0) \in Q_{in}$ (i.e., the run starts in an initial state) and for every $l \geq 0$, we have $r(l+1) \in \rho(r(l), \sigma_l)$ (i.e., the run obeys the transition function). In automata over finite words, acceptance is defined according to the last state visited by the run. When the words are infinite, there is no such thing as a "last state", and acceptance is defined according to the set $Inf(r)$ of states that $r$ visits *infinitely often*, i.e., $Inf(r) = \{q \in Q : \text{for i.m. } l \in \mathbb{N}, \text{we have } r(l) = q\}$. As $Q$ is finite, it is guaranteed that $Inf(r) \neq \emptyset$. The way we refer to $Inf(r)$ depends on the acceptance condition of $\mathcal{A}$. In *Büchi automata*, $\alpha \subseteq Q$, and $r$ is accepting iff $Inf(r) \cap \alpha \neq \emptyset$. Dually, in *co-Büchi automata*, $\alpha \subseteq Q$, and $r$ is accepting iff $Inf(r) \cap \alpha = \emptyset$.

Since $\mathcal{A}$ is not deterministic, it may have many runs on $w$. There are two, dual, ways in which we can refer to the many runs. When $\mathcal{A}$ is an *existential* automaton (or simply a *nondeterministic* automaton, as we shall call it in the sequel), it accepts an input word $w$ iff there exists an accepting run of $\mathcal{A}$ on $w$. When $\mathcal{A}$ is a *universal* automaton, it accepts an input word $w$ iff all the runs of $\mathcal{A}$ on $w$ are accepting. The language of $\mathcal{A}$, denoted $\mathcal{L}(\mathcal{A})$ consists of all words accepted by $\mathcal{A}$.

We use three-letter acronyms to describe types of automata. The first letter describes the transition structure and is one of "N" (nondeterministic), and "U" (universal). The second letter describes the acceptance condition; in this paper we only consider "B" (Büchi) and "C" (co-Büchi). The third letter describes the objects on which the automata run; in this paper we are only concerned with "W" (infinite words). Thus, for example, NBW designates a nondeterministic Büchi word automaton and UCW designates a universal co-Büchi word automaton.

A lower bound for complementing NBW was established by Michel [23] (cf. [22]). Consider the alphabet $\Sigma_n = \{1, \ldots, n\}$. Let $w = a_0, a_1, \ldots$ be a word over $\Sigma_n$. An *infinite path in* $w$ is a an infinite subsequence $a_{i_0}, a_{i_0+1}, a_{i_1}, a_{i_1+1}, \ldots$ such $a_{i_j+1} = a_{i_{j+1}}$ for $j \geq 0$; that is, an infinite path in $w$ is an infinite subword of matching pairs of leters. Let $L_n$ be the language of infinite words over $\Sigma_n$ with infinite paths.

**Theorem 1.** [23]

- *$L_n$ can be defined using an $n$-state NBW.*
- *$\Sigma_n^\omega - L_n$ cannot be defined using an NBW with fewer than $n!$ states.*

## 3 Complementation via Ranks

In [18][3], the following approach for NBW complementation is described: in order to complement an NBW, first dualize the transition function and the acceptance condition, and then translate the resulting UCW automaton back to an NBW. By [26], the dual automaton accepts the complementary language, and so does the nondeterministic automaton we end up with. Thus, rather than determinization, complementation is based on a translation of universal automata to nondeterministic ones, which turns out to be simpler. (See also [35].)

Consider a UCW $\mathcal{A} = \langle \Sigma, Q, Q_{in}, \delta, \alpha \rangle$. The runs of $\mathcal{A}$ on a word $w = \sigma_0 \cdot \sigma_1 \cdots$ can be arranged in an infinite DAG (directed acyclic graph) $\mathcal{G}_w = \langle V, E \rangle$, where

- $V \subseteq Q \times \mathbb{N}$ is such that $\langle q, l \rangle \in V$ iff some run of $\mathcal{A}$ on $w$ has $r(l) = q$. For example, the first level of $\mathcal{G}_w$ contains the nodes $Q_{in} \times \{0\}$.
- $E \subseteq \bigcup_{l \geq 0} (Q \times \{l\}) \times (Q \times \{l+1\})$ is such that $E(\langle q, l \rangle, \langle q', l+1 \rangle)$ iff $\langle q, l \rangle \in V$ and $q' \in \delta(q, \sigma_l)$.

Thus, $\mathcal{G}_w$ embodies exactly all the runs of $\mathcal{A}$ on $w$. We call $\mathcal{G}_w$ the *run* DAG of $\mathcal{A}$ on $w$, and we say that $\mathcal{G}_w$ is *accepting* if all its paths satisfy the acceptance condition $\alpha$. Note that $\mathcal{A}$ accepts $w$ iff $\mathcal{G}_w$ is accepting. We say that a node $\langle q', l' \rangle$ is a *successor* of a node $\langle q, l \rangle$ iff $E(\langle q, l \rangle, \langle q', l' \rangle)$. We say that $\langle q', l' \rangle$ is *reachable* from $\langle q, l \rangle$ iff there exists a sequence $\langle q_0, l_0 \rangle, \langle q_1, l_1 \rangle, \langle q_2, l_2 \rangle, \ldots$ of successive nodes such that $\langle q, l \rangle = \langle q_0, l_0 \rangle$, and there exists $i \geq 0$ such that $\langle q', l' \rangle = \langle q_i, l_i \rangle$. For a set $S \subseteq Q$, we say that a node $\langle q, l \rangle$ of $\mathcal{G}_w$ is an *$S$-node* if $q \in S$.

A short detour is now required. A *fair transition system* $M = (W, W_0, R, F)$ consists of a state set $W$ (not necessarily finite), an initial state set $W_0 \subseteq W$, a transition relation $R \subseteq W^2$, and a *fair* state set $F \subseteq W$. An *infinite trace* of $M$ is an infinite state sequence $w_0, w_1, \ldots$ such that $w_0 \in W_0$ and $(w_i, w_{i+1}) \in R$ for all $i \geq 0$. This trace is *fair* if $w_i \in F$ for infinitely many $i$'s. We say that $M$ *fairly terminates* if it has no fair infinite trace. Fair termination is a fundamental property of transition systems, as verification of linear temporal properties for transition systems can be reduced to fair-termination checking [36].

Emerson and Clarke characterized fair termination in terms of a nested fixpoint computation [6]. Let $X, Y \subseteq W$. Define $until(X, Y)$ as the set of states in $X$ that can properly reach $Y$ while staying in $X$. That is, $until(X, Y)$ consists of states $x$ such that there is a sequence $x_0, \ldots, x_k$, $k > 0$, where $x_k \in Y$ and $x_i \in X$ for $0 \leq i < k$. Clearly, $until(X, Y)$ can be defined in terms of a least fixpoint. Consider now the following greatest fixpoint "algorithm", which we refer to by EC:

$Q \leftarrow W$
while change do
    $Q \leftarrow Q \cap until(Q, Q \cap F)$

[3] Preliminary version appeared in [17].

endwhile
return $(W_0 \cap Q = \emptyset)$

Emerson and Clarke showed that EC returns TRUE precisely when $M$ fairly terminates. The intuition is that we can safely delete states that cannot be on a fair infinite trace because they cannot properly reach $F$ even once. Note that the inner fixpoint, required to compute $until(Q, Q \cap F$ always converges in $\omega$ stages, since it concerns only finite traces, while the outer fixpoint may require transfinite stages to converge, when $W$ is infinite. For finite transition systems, EC is a real algorithm for fair-termination detection [7], which is used widely in symbolic model checking [4].

A run DAG can be viewed as a fair transition system. Consider a UCW $\mathcal{A} = \langle \Sigma, Q, Q_{in}, \delta, \alpha \rangle$, with a run DAG $\mathcal{G}_w = \langle V, E \rangle$. The corresponding fair transition system is $M_w = (V, Q_{in} \times \{0\}, E, \alpha \times \mathbb{N})$. Clearly, $\mathcal{G}_w$ is accepting iff $M_w$ fairly terminates. EC can therefore be applied to $M_w$. Using this characterization of acceptance, we can assign *ranks* to the nodes of $V$, as follows: a node is assigned rank $i$ if it is deleted at the $i$-th iteration of the loop in EC. Since all nodes of $\mathcal{G}_w$ are reachable from $Q_{in} \times \{0\}$, all nodes will be assigned a rank if $\mathcal{G}_w$ is accepting. Intuitively, ranks measure the "progress" made by a node towards acceptance [16]. We can view these ranks as evidence that $\mathcal{G}_w$ is accepting. As we noted, however, transfinite ranks are required in general, while we desire finite ranks for the complementation construction.

To that end we refer to a heuristic improvement of EC, developed in [8], and referred to by OWCTY. Let $X \subseteq W$ be a set of states in a transition system $M = (W, W_0, R, F)$. By $next(X)$ we refer to states who has successors in $X$, that is, all states $x \in W$ such that there is a state $y \in W$ where $(x, y) \in R$ and $y \in X$. OWCTY is obtained from EC by adding an inner loop[4]:

$Q \leftarrow W$
while change do
    while change do
        $Q \leftarrow Q \cap next(Q)$
    endwhile
    $Q \leftarrow Q \cap until(Q, Q \cap F)$
endwhile
return $(W_0 \cap Q = \emptyset)$

Note that the additional inner loop deletes states that have no successor. Such states surely cannot lie on a fair infinite trace, which ensure that OWCTY is a correct characterization of fair termination. Surprisingly, while EC requires, in general, transfinitely many stages to converge, it is shown in [18] that when OWCTY is applied to fair transition systems of the form $M_w$ for a UCW $\mathcal{A}$ with $n$ states, the external loop always converges in at most $n$ iterations. The crucial fact here is that each level of $\mathcal{G}_w$ has at most $n$ nodes. This enables us to assign finite ranks to the nodes of $\mathcal{G}_w$ as follows (we count iterations from 0):

[4] The additional loop here precedes the inner statement of EC, while in [8] it succeeds it. This is not an essential change.

– Assign a node $v$ rank $2i$ if it is deleted in the $i$-th iteration by the statement $Q \leftarrow Q \cap next(Q)$.
– Assign a node $v$ rank $2i+$ if it is deleted in the $i$-th iteration by the statement $Q \leftarrow Q \cap until(Q, Q \cap F)$.

It is shown in [12, 18] that precisely the ranks $0, \ldots, 2n-2$ are needed (see also [16]).

We can now characterize accepting run DAGs in terms of ranks. Consider an $n$-state UCW $\mathcal{A} = \langle \Sigma, Q, Q_{in}, \delta, \alpha \rangle$, with a run DAG $\mathcal{G}_w = \langle V, E \rangle$. A *C-ranking* for $\mathcal{G}_w$ is a mapping $f : V \to \{0, \ldots, 2n-2\}$ such that

1. For all nodes $\langle q, l \rangle \in V$, if $f(\langle q, l \rangle)$ is odd, then $q \notin \alpha$.
2. For all edges $\langle \langle q, l \rangle, \langle q', l+1 \rangle \rangle \in E$, we have $f(\langle q', l+1 \rangle) \leq f(\langle q, l \rangle)$.

Thus, a C-ranking associates with each node in $\mathcal{G}_w$ a rank so that the ranks along paths do not increase, and $\alpha$-nodes get only even ranks. We say that a node $\langle q, l \rangle$ is an *odd node* if $f(\langle q, l \rangle)$ is odd. Note that each path in $\mathcal{G}_w$ eventually gets trapped in some rank. We say that the C-ranking $f$ is an *odd C-ranking* if all the paths of $\mathcal{G}_w$ eventually get trapped in odd ranks. Formally, $f$ is odd iff for all paths $\langle q_0, 0 \rangle, \langle q_1, 1 \rangle, \langle q_2, 2 \rangle, \ldots$ in $\mathcal{G}_w$, there is $l \geq 0$ such that $f(\langle q_l, l \rangle)$ is odd, and for all $l' \geq l$, we have $f(\langle q_{l'}, l' \rangle) = f(\langle q_l, l \rangle)$. Note that, equivalently, $f$ is odd if every path of $\mathcal{G}_w$ has infinitely many odd nodes.

**Lemma 1.** [18] *The following are equivalent.*

1. *All paths of $\mathcal{G}_w$ have only finitely many $\alpha$-nodes.*
2. *There is an odd C-ranking for $\mathcal{G}_w$.*

The fact that the nodes of a run DAG can be assigned finite ranks means that we can characterize acceptance using a variation of the subset construction, where each element of the subset also carries a rank. It is easy to check that the two conditions of C-ranking hold, since these involve only local conditions. Here is a first attempt to construct an NBW $\mathcal{A}'$ that is equivalent to the UCW $\mathcal{A}$. When $\mathcal{A}'$ reads a word $w$, it guesses a C-ranking for the run DAG $\mathcal{G}_w$ of $\mathcal{A}$ on $w$. At a given point of a run of $\mathcal{A}'$, it keeps in its memory a whole level of $\mathcal{G}_w$ and a guess for the ranks of the nodes at this level.

Before we define $\mathcal{A}'$, we need some notation. A *level ranking* for $\mathcal{A}$ is a function $g : Q \to \{0, \ldots, 2n-2\}$, such that if $g(q)$ is odd, then $q \notin \alpha$. Let $\mathcal{R}$ be the set of all level rankings. For a subset $S$ of $Q$ and a letter $\sigma$, let $\delta(S, \sigma) = \bigcup_{s \in S} \delta(s, \sigma)$. Note that if level $l$ in $\mathcal{G}_w$, for $l \geq 0$, contains the states in $S$, and the $(l+1)$-th letter in $w$ is $\sigma$, then level $l+1$ of $\mathcal{G}_w$ contains the states in $\delta(S, \sigma)$. For two level rankings $g$ and $g'$ in $\mathcal{R}$, a set $S \subseteq Q$, and a letter $\sigma$, we say that $g'$ *covers* $\langle g, S, \sigma \rangle$ if for all $q \in S$ and $q' \in \delta(q, \sigma)$, we have $g'(q') \leq g(q)$. Thus, if the nodes of level $l$ contain exactly all the states in $S$, $g$ describes the ranks of these nodes, and the $(l+1)$-th letter in $w$ is $\sigma$, then $g'$ is a possible level ranking for level $l+1$. Finally, for $g \in \mathcal{R}$, let $odd(g) = \{q : g(q) \text{ is odd}\}$. Thus, a state of $Q$ is in $odd(g)$ if has an odd rank.

We can now try to define $\mathcal{A}'$ as follows. For the state set we take $Q' = 2^S \times \mathcal{R}$ and $Q'_{in} = Q_{in} \times \mathcal{R}$. Thus, a state of $\mathcal{A}'$ is simply a ranked subset of $Q$. Now we can define the transition function by $\delta'(\langle S, g\rangle, \sigma) = \{\langle \delta(S, \sigma), g'\rangle : g' \text{ covers } \langle g, S, \sigma\rangle\}$. This definition guarantees that $\mathcal{A}'$ is guessing a C-ranking of a run DAG $\mathcal{G}_w$. Unfortunately, this is not sufficient. To ensure that $\mathcal{G}_w$ is accepting we need to find an *odd* C-ranking. It is not clear how $\mathcal{A}'$ can check for oddness, which seems to be a global condition. To overcome this difficulty we use a technique due to [24], which uses a second subset construction to ensure that no path of $\mathcal{G}_w$ get stuck in an odd rank.

Let $\mathcal{A}' = \langle \Sigma, Q', Q'_{in}, \delta', \alpha'\rangle$, where

- $Q' = 2^Q \times 2^Q \times \mathcal{R}$, where a state $\langle S, O, g\rangle \in Q'$ indicates that the current level of the run DAG contains the states in $S$, the set $O \subseteq S$ contains states along paths that have not visited an odd node since the last time $O$ has been empty, and $g$ is the guessed level ranking for the current level.
- $Q'_{in} = \{Q_{in}\} \times \{\emptyset\} \times \mathcal{R}$.
- $\delta'$ is defined, for all $\langle S, O, g\rangle \in Q'$ and $\sigma \in \Sigma$, as follows.
  - If $O \neq \emptyset$, then

    $$\delta'(\langle S, O, g\rangle, \sigma) = \{\langle \delta(S, \sigma), \delta(O, \sigma) \setminus \mathit{odd}(g'), g'\rangle : g' \text{ covers } \langle g, S, \sigma\rangle\}.$$

  - If $O = \emptyset$, then

    $$\delta'(\langle S, O, g\rangle, \sigma) = \{\langle \delta(S, \sigma), \delta(S, \sigma) \setminus \mathit{odd}(g'), g'\rangle : g' \text{ covers } \langle g, S, \sigma\rangle\}.$$

- $\alpha' = 2^Q \times \{\emptyset\} \times \mathcal{R}$.

An easy analysis show that $\mathcal{A}'$ has at most $(6n)^n$) states. This should be contrasted with the bound of $n^{2n}$ that results from determinization [29].

**Theorem 2.** [18] *Let $\mathcal{A}$ be a UCW with $n$ states. Then $\mathcal{A}'$ has at most $(6n)^n$ states and $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$.*

A report on an implementation of this construction, which includes also many optimizations, can be found in [12].

## 4 Tight Rankings

While the upper bound bound of $(6n)^n$ described above is exponentially better than the bound of $n^{2n}$ obtained via determinization, is is still exponentially far from the lower bound of $n!$. Recent results have improved both the upper and lower bounds.

For the upper bound, it was shown in [9] that the rank-based construction can be tightened. Consider a UCW $\mathcal{A}$ and a word $w \in \Sigma^\omega$ accepted by $\mathcal{A}$. For the run dag $\mathcal{G}_w$ of $\mathcal{A}$ on $w$, let $\mathit{max\_rank}(\mathcal{G}_w)$ be the maximal rank that a node in $\mathcal{G}_w$ gets. For a rank $j \in \{0, \ldots, 2n-2\}$, let $[j]^{odd}$ be all odd ranks less than or equal to $j$.

**Lemma 2.** [9] *There is a limit level $l \geq 0$ such that for each level $l' > l$, and for all ranks $j \in [max\_rank(G_w)]^{odd}$, there is a node $\langle q, l' \rangle$ such that $rank(q, l') = j$.*

Recall that a level ranking for $\mathcal{A}$ is a function $g : Q \to \{0, \ldots, 2n-2\}$, such that if $g(q)$ is odd, then $q \notin \alpha$. Let $max\_odd(g)$ be the maximal odd number in the range of $g$.

**Definition 1.** *We say that a level ranking $g$ is* tight *if*

1. *the maximal rank in the range of $g$ is odd, and*
2. *for all $j \in [max\_odd(g)]^{odd}$, there is a state $q \in Q$ with $g(q) = j$.*

**Lemma 3.** [9] *There is a level $l \geq 0$ such that for each level $l' > l$, the level ranking that corresponds to $l'$ is tight.*

It follows that we can improve the earlier complementation construction and restrict the set $\mathcal{R}$ of possible level rankings to the set of tight level rankings. Since, however, the tightness of the level ranking is guaranteed only beyond the limit level $l$ of $\mathcal{G}_w$, we also need to guess this level, and proceed with the usual subset construction until we reach it. Formally, we suggest the following modified construction.

Let $\mathcal{A} = \langle \Sigma, Q, Q_{in}, \delta, \alpha \rangle$ be a UCW, and let $\mathcal{R}_{tight}$ be the set of tight level rankings for $\mathcal{A}$. Let $\mathcal{A}' = \langle \Sigma, Q', Q'_{in}, \delta', \alpha' \rangle$, where

- $Q' = 2^Q \cup (2^Q \times 2^Q \times \mathcal{R}_{tight})$, where a state $S \in Q'$ indicates that the current level of the run DAG contains the states in $S$, and a state $\langle S, O, g \rangle \in Q'$ is similar to the states in the earlier construction; in particular, $O \subseteq S$.
- $Q'_{in} = \{Q_{in}\}$. Thus, the run starts in a "subset mode", corresponding to a guess that the limit level has not been reached yet.
- For all states in $Q'$ of the form $S \in 2^Q$ and $\sigma \in \Sigma$, we have that

$$\delta'(S, \sigma) = \{\delta(S, \sigma)\} \cup \{\langle \delta(S, \sigma), \emptyset, g \rangle : \text{ and } g \in \mathcal{R}_{tight}\}.$$

Thus, at each point in the subset mode, $\mathcal{A}'$ may guess that the current level is the limit level, and move to a "subset+ranks" mode, where it proceeds as the NBW constructed earlier. Thus, for states of the form $\langle S, O, g \rangle$, the transition function is as described earlier, except that level rankings are restricted to tight ones.

**Theorem 3.** [9] *Let $\mathcal{A}$ be a UCW. Then $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$.*

It remains to analyze carefully the complexity of this construction. Let $tight(n)$ be the number of tight level rankings for automata with $n$ states. Is is easy to see that $\mathcal{A}'$ needs at most $3^n \cdot tight(n)$ states. A careful analysis, based on an asymptotic approximation of Stirling Numbers of The Second Kind [34], yields that $tight(n)$ is bounded by $(0.76n)^n$. We also have a factor of $3^n$ that results from the two subset constructions; recall that a state has the form $\langle S, O, g \rangle$, in which $S$ and $O$ are subsets of the state space of the original automaton, with

$O \subseteq S$, and $g$ is a tight level ranking. This analysis ignores possible relations between the pair $\langle S, O\rangle$ and the tight level ranking $g$ associated with it.

Consider a state $\langle S, O, g\rangle$ of the NBW $\mathcal{A}'$ constructed. Since we are interested only in the ranks of states in $S$, we can assume without loss of generality that $g$ assigns the rank 0 to all states not in $S$. In addition, as $O$ maintains the set of states that have not been visited an odd vertex, $g$ maps all the states in $O$ to an even rank. A careful combinatorial analysis now yields the following.

**Theorem 4.** [9] *Let $\mathcal{A}$ be a UCW with $n$ states. Then there is an NBW $\mathcal{A}'$ with at most $(0.97n)^n$ states such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.*

In particular, the upper bound is lower than $n^n$, which would have been a "clean" bound. Recent progress has also been made on the lower-bound front. It is shown in [39] that the complementary automaton needs to maintain all tight level rankings, resulting in a lower bound of $(0.76n)^n$, which is exponentially stronger than the previous bound of $n! \approx (n/e)^n$. An exponential bound remains between the upper bound of $(0.97n)^n$ and the lower bound of $(0.76n)^n$). Closing this gap is a tantalizing open question.

## 5 Concluding Remarks

Our focus in this paper was on the theoretical aspect of Büchi complementation. It is important to note that this is also an important practical problem. No verification tool so far supports the unrestricted use of Büchi automata as a specification formalism, due to the perceived difficulty of complementation. In spite of some recent progress in implementing Büchi complementation [12], more work needs to be done to make this practically viable.

It should also be noted that complementation is important for automata on infinite words with stronger acceptance conditions, such as generalized Büchi automata [20] and Streett automata [19]. In particular, Streett automata express strong fairness in a natural way. A Streett acceptance condition consists of a set of pairs $(L, R)$ of sets of states. The requirement is that if a run visits $L$ infinitely often, it also visits $R$ infinitely often. The best known upper bound for complementing a Streett automaton with $n$ states and $k$ pairs is $(kn)^{O(kn)}$ [16, 19, 30]. The only known lower bound is of $(kn)^{O(n)}$ [39].

## References


1. R. Armoni, L. Fix, A. Flaisher, R. Gerth, B. Ginsburg, T. Kanza, A. Landver, S. Mador-Haim, E. Singerman, A. Tiemeyer, M.Y. Vardi, and Y. Zbar. The ForSpec temporal logic: A new temporal property-specification logic. In *Proc. 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 2280 of *Lecture Notes in Computer Science*, pages 296–211, Grenoble, France, April 2002. Springer-Verlag.
2. J.C. Birget. Partial orders on words, minimal elements of regular languages, and state complexity. *Theoretical Computer Science*, 119:267–291, 1993.

3. J.R. Büchi. On a decision method in restricted second order arithmetic. In *Proc. International Congress on Logic, Method, and Philosophy of Science. 1960*, pages 1–12, Stanford, 1962. Stanford University Press.
4. J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic model checking: $10^{20}$ states and beyond. *Information and Computation*, 98(2):142–170, June 1992.
5. N. Daniele, F. Guinchiglia, and M.Y. Vardi. Improved automata generation for linear temporal logic. In *Computer Aided Verification, Proc. 11th International Conference*, volume 1633 of *Lecture Notes in Computer Science*, pages 249–260. Springer-Verlag, 1999.
6. E.A. Emerson and E.M. Clarke. Characterizing correctness properties of parallel programs using fixpoints. In *Proc. 7th InternationalColloq. on Automata, Languages and Programming*, pages 169–181, 1980.
7. E.A. Emerson and C.-L. Lei. Temporal model checking under generalized fairness constraints. In *Proc. 18th Hawaii International Conference on System Sciences*, North Holywood, 1985. Western Periodicals Company.
8. K. Fisler, R. Fraer, G. Kamhi, M.Y. Vardi, and Z. Yang. Is there a best symbolic cycle-detection algorithm? In *7th International Conference on Tools and algorithms for the construction and analysis of systems*, number 2031 in Lecture Notes in Computer Science, pages 420–434. Springer-Verlag, 2001.
9. E. Friedgut, O. Kupferman, and M.Y. Vardi. Büchi complementation made tighter. *Int'l J. of Foundations of Computer Science*, 17(4):851–867, 2006.
10. P. Gastin and D. Oddoux. Fast LTL to büchi automata translation. In *Computer Aided Verification, Proc. 13th International Conference*, volume 2102 of *Lecture Notes in Computer Science*, pages 53–65. Springer-Verlag, 2001.
11. S. Gurumurthy, R. Bloem, and F. Somenzi. Fair simulation minimization. In *Computer Aided Verification, Proc. 14th International Conference*, volume 2404 of *Lecture Notes in Computer Science*, pages 610–623. Springer-Verlag, 2002.
12. S. Gurumurthy, O. Kupferman, F. Somenzi, and M.Y. Vardi. On complementing nondeterministic Büchi automata. In *12th Advanced Research Working Conference on Correct Hardware Design and Verification Methods*, volume 2860 of *Lecture Notes in Computer Science*, pages 96–110. Springer-Verlag, 2003.
13. T.A. Henzinger, O. Kupferman, and S. Rajamani. Fair simulation. *Information and Computation*, 173(1):64–81, 2002.
14. G.J. Holzmann. The model checker SPIN. *IEEE Trans. on Software Engineering*, 23(5):279–295, May 1997. Special issue on Formal Methods in Software Practice.
15. Y. Kesten, N. Piterman, and A. Pnueli. Bridging the gap between fair simulation and trace containment. In *Computer Aided Verification, Proc. 15th International Conference*, volume 2725 of *Lecture Notes in Computer Science*, pages 381–393. Springer-Verlag, 2003.
16. N. Klarlund. Progress measures for complementation of $\omega$-automata with applications to temporal logic. In *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, pages 358–367, San Juan, October 1991.
17. O. Kupferman and M.Y. Vardi. Weak alternating automata are not that weak. In *Proc. 5th Israeli Symp. on Theory of Computing and Systems*, pages 147–158. IEEE Computer Society Press, 1997.
18. O. Kupferman and M.Y. Vardi. Weak alternating automata are not that weak. *ACM Trans. on Computational Logic*, 2(2):408–429, July 2001.
19. O. Kupferman and M.Y. Vardi. Complementation constructions for nondeterministic automata on infinite words. In *Proc. 11th International Conf. on Tools and*

*Algorithms for The Construction and Analysis of Systems*, volume 3440 of *Lecture Notes in Computer Science*, pages 206–221. Springer-Verlag, 2005.
20. O. Kupferman and M.Y. Vardi. From complementation to certification. *Theoretical Computer Science*, 305:591–606, 2005.
21. R.P. Kurshan. *Computer Aided Verification of Coordinating Processes*. Princeton Univ. Press, 1994.
22. C. Löding. Optimal bounds for the transformation of omega-automata. In *Proc. 19th Conference on the Foundations of Software Technology and Theoretical Computer Science*, volume 1738 of *Lecture Notes in Computer Science*, pages 97–109, December 1999.
23. M. Michel. Complementation is more difficult with automata on infinite words. CNET, Paris, 1988.
24. S. Miyano and T. Hayashi. Alternating finite automata on $\omega$-words. *Theoretical Computer Science*, 32:321–330, 1984.
25. D.E. Muller. Infinite sequences and finite machines. In *Proc. 4th IEEE Symp. on Switching Circuit Theory and Logical design*, pages 3–16, 1963.
26. D.E. Muller and P.E. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54:267–276, 1987.
27. J.P. Pécuchet. On the complementation of büchi automata. *Theor. Comput. Sci.*, 47(3):95–98, 1986.
28. M.O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:115–125, 1959.
29. S. Safra. On the complexity of $\omega$-automata. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 319–327, White Plains, October 1988.
30. S. Safra. Exponential determinization for $\omega$-automata with strong-fairness acceptance condition. In *Proc. 24th ACM Symp. on Theory of Computing*, Victoria, May 1992.
31. W. Sakoda and M. Sipser. Non-determinism and the size of two-way automata. In *Proc. 10th ACM Symp. on Theory of Computing*, pages 275–286, 1978.
32. 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.
33. F. Somenzi and R. Bloem. Efficient Büchi automata from LTL formulae. In *Computer Aided Verification, Proc. 12th International Conference*, volume 1855 of *Lecture Notes in Computer Science*, pages 248–263. Springer-Verlag, 2000.
34. N.M. Temme. Asimptotic estimates of Stirling numbers. *Stud. Appl. Math.*, 89:233–243, 1993.
35. W. Thomas. Complementation of Büchi automata revised. In J. Karhumäki, H. A. Maurer, G. Paun, and G. Rozenberg, editors, *Jewels are Forever*, pages 109–120. Springer, 1999.
36. M.Y. Vardi. Verification of concurrent programs - the automata-theoretic framework. *Annals of Pure and Applied Logic*, 51:79–98, 1991.
37. M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, November 1994.
38. P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1–2):72–99, 1983.
39. Q. Yan. Lower bounds for complementation of $\omega$-automata via the full automata technique. In *Proc. 33rd Intl. Colloq. on Automata, Languages and Pr ogramming*, volume 4052 of *Lecture Notes in Computer Science*, pages 589–600. Springer-Verlag, 2006.