# Ramsey-Based Analysis of Parity Automata

Oliver Friedmann[1] and Martin Lange[2]

[1] Dept. of Computer Science, Ludwig-Maximilians-University of Munich, Germany
[2] School of Electr. Eng. and Computer Science, University of Kassel, Germany

**Abstract.** Parity automata are a generalisation of Büchi automata that have some interesting advantages over the latter, e.g. determinisability, succinctness and the ability to express certain acceptance conditions like the intersection of a Büchi and a co-Büchi condition directly as a parity condition. Decision problems like universality and inclusion for such automata are PSPACE-complete and have originally been tackled via explicit complementation only. Ramsey-based methods are a later development that avoids explicit complementation but relies on an application of Ramsey's Theorem for its correctness. In this paper we develop new and explicit Ramsey-based algorithms for the universality and inclusion problem for nondeterministic parity automata. We compare them to Ramsey-based algorithms which are obtained from translating parity automata into Büchi automata first and then applying the known Ramsey-based analysis procedures to the resulting automata. We show that the speed-up in the asymptotic worst-case gained through the new and direct methods is exponential in the number of priorities in the parity automata. We also show that the new algorithms are much more efficient in practice.

## 1 Introduction

Nondeterministic Büchi automata (NBA) are the most well-known type of finite automata that work on infinite words. Much of their popularity is owed to two facts. (1) Their acceptance condition is conceptually very simple: a run is accepting iff it visits a certain subset of states infinitely often. (2) Despite this simplicity they form an expressively complete specification formalism with respect to Monadic Second-Order Logic [4], i.e. they accept exactly the regular languages of infinite words.

A lot of attention has been paid to the algorithmic treatment of fundamental decision and computation problems for regular languages represented by NBA. The complementation problem is combinatorially much more difficult than that for NFA. The fundamental difference is the fact that determinisation for NBA is provably impossible, and particularly a simple procedure like the powerset construction for NFA fails for finite automata on infinite words equipped with any reasonable acceptance condition, not just the Büchi condition. This has brought out numerous work on the complementation (and also determinisation) problem for NBA [4, 13, 8, 14, 9].

Clearly, other problems that generalise complementation in some way – e.g. universality, inclusion, equivalence – are also combinatorially challenging. For instance, in order to check whether $L(\mathcal{A}) \subseteq L(\mathcal{B})$ holds for two NBA $\mathcal{A}$ und $\mathcal{B}$, one would complement $\mathcal{B}$ to some $\overline{\mathcal{B}}$, build an automaton that accepts its language intersected with $L(\mathcal{A})$ – which is relatively simple using a small enhancement of the usual product construction for NFA – and check the result for emptiness. In the most complex steps of this procedure, one can choose among several complementation procedures. This choice was often made on the basis of worst-case analysis or modern aspects like symbolic implementability. For instance, Klarlund's procedure [8] runs in optimal time of $2^{\mathcal{O}(n \log n)}$ while Kupferman and Vardi's [9] can be made to work with BDDs at the expense of running in time $2^{\mathcal{O}(n^2)}$. Büchi's procedure [4] was not seen as practical which may also be caused by the fact that the literature falsely accused it of having doubly exponential running time whereas a careful analysis shows that it can be made to run in time $2^{\mathcal{O}(n^2)}$ as well.

Büchi's proof of correctness for his complementation procedure uses Ramsey's Theorem [12]. For a long time this has been regarded as a tool to handle the combinatorial difficulty in the correctness proof without much algorithmic value for the complementation problem, hence the focus on other procedures for practical applications. It was Ben-Amram, Jones and Lee [10] who first suggested to use this principle for a practical application in termination analysis called size-change termination which could have also been solved using Büchi complementation. They state that "for practical usage [...] the simple algorithm seems more promising than [...] the solution based on $\omega$-automata". The term "simple algorithm" refers to a procedure that builds a set of finite graphs through a composition operation and searches for an idempotent graph with certain properties in it. This is basically a direct usage of the computational content of Ramsey's Theorem for this particular decision problem. Henceforth, such simple algorithms will be said to be Ramsey-based.

Next, Dax et al. [5] introduced this Ramsey-based method to the domain of temporal logic: they gave an algorithm checking validity for a formula of the linear-time $\mu$-calculus ($\mu$-TL) [3], a temporal fixpoint logic extending the standard LTL [11]. These problems had – until then – solely been approached using automata-theoretic machinery, i.e. explicitly using the complementation problem for NBA [16]. Dax et al. showed that the Ramsey-based method can outperform those using automata explictly. Since $\mu$-TL is also expressively complete for regular languages, and there is a linear translation from NBA to $\mu$-TL – mapping universality to validity – this also defines a Ramsey-based method for the universality problem for NBA. After that, Fogarty and Vardi have made this connection explicit and also investigated its practical use for the NBA universality problem in general [6, 7]. The apparent use has then inspired work on further optimisations of this Ramsey-based approach to NBA universality and NBA inclusion [1].

Büchi automata are the simplest but not the only type of automaton on infinite words. Notably, the literature also considers the syntactically more general

Muller, Rabin, Streett and parity automata, all of them expressively complete w.r.t. $\omega$-regular languages. Here we consider nondeterministic parity automata (NPA) which are computationally most elegant among those models. There are several reasons for considering NPA as a generalisation of NBA.

1. *Succinctness.* Many properties can be expressed more succinctly with NPA than with NBA. Consider, for instance the language $L_0$ of all words over the alphabet $\{a, b, c\}$ that also contains infinitely many $b$'s when they contain infinitely many $a$'s. This can be accepted by an NPA with three states in which they signal the last letter that has been seen. A $b$ is then signaled with priority 2, an $a$ with priority 1 and a $c$ with priority 0. A similarly straightforward construction of an NBA for this language results in 5 states, and it does not look like a 3-state NBA for this language exists.
2. *Determinisability.* $L_0$ can be accepted by a deterministic parity automaton (DPA) but not by a deterministic Büchi automaton. In general, DPA are expressively complete whilst deterministic DBA are not.
3. *Expressiveness.* Certain acceptance conditions can be formulated as a parity condition but not as a Büchi condition, i.e. certain automata can be regarded as an NPA but not as an NBA.[3] For instance, the intersection of a Büchi condition with a co-Büchi condition stating that certain states should be seen infinitely often while others should only be seen finitely often, can easily be encoded by a parity condition with priorities $\{1, 2, 3\}$, compare this to $L_0$ above.

In this paper we develop Ramsey-based algorithms for NPA. These extend corresponding methods for NBA. The benefit of this extension is empirically shown: it is known that NPA can be translated to NBA at a moderate blow-up. We compare the new methods for NPA with the old methods for NBA obtained under this translation showing that the new methods are not only asymptotically faster but also behave better in practice. Furthermore, there is a reduction from the inclusion problem to universality that can be made to work for various types of automata including NBA and NPA. We show that this does not alleviate the use of a direct method for NPA inclusion: performing the reduction to universality and then applying the universality method for NPA is again asymptotically and practically worse. In essence, the Ramsey-based methods developed in this paper are justified by their superiority over reductions to existing methods both in theory and in practice.

## 2 Preliminaries

As usual, for a finite alphabet $\Sigma$ we write $\Sigma^*$ / $\Sigma^+$ / $\Sigma^\omega$ to denote the set of all finite / finite non-empty / infinite words over $\Sigma$. An infinite word $w \in \Sigma^\omega$ is *regular* if there are $u \in \Sigma^*$ and $v \in \Sigma^+$ s.t. $w = uv^\omega$. If $w$ is a finite word then $|w|$ denotes its length.

[3] Note that their *language* is still NBA-recognisable, but this may require a different underlying automaton.

A *nondeterministic parity automaton* (NPA) is a $\mathcal{A} = (Q, \Sigma, q_0, \delta, \Omega)$ where $Q$ is a finite set of states, $\Sigma$ is a finite alphabet, $q_0 \in Q$ is a designated starting state, $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation, and $\Omega : Q \to \mathbb{N}$ is the priority function.

A *run* of $\mathcal{A}$ on a word $w = a_0 a_1 a_2 \ldots \in \Sigma^\omega$ is a sequence $\rho = q_0, q_1, \ldots$ s.t. $q_0$ is the designated starting state and for all $i \in \mathbb{N}$ we have: $(q_i, a_i, q_{i+1}) \in \delta$. It is *accepting* if $\max\{\Omega(q) \mid \forall i \in \mathbb{N}.\exists j \geq 1.q_j = q\}$ is even. The *language* of $\mathcal{A}$ is $L(\mathcal{A}) = \{w \mid \text{there is an accepting run of } \mathcal{A} \text{ on } w\}$.

We introduce two important complexity measures for an NPA. The *size* of $\mathcal{A}$ is $|\mathcal{A}| = |Q|$. The *index* of $\mathcal{A}$ is $idx(\mathcal{A}) = |\{\Omega(q) \mid q \in Q\}|$, i.e. the number of distinct priorities used in $\mathcal{A}$. Clearly we always have $idx(\mathcal{A}) \leq |\mathcal{A}|$.

NPA are a natural generalisation of the well-known nondeterministic Büchi automata which are traditionally defined using the concept of acceptance state rather than a priority function. An accepting run is one that visists the acceptance set infinitely often. The definition used in the following is easily seen to be equivalent to that. A *nondeterministic Büchi automaton* (NBA) is a special kind of an NPA of index 2, s.t. $\Omega(q) \in \{1, 2\}$ for all $q \in Q$.

An *$\omega$-regular language* or just *regular language* for short is a language that can be accepted by an NBA. It is known that NPA, despite being a generalisation, do not accept more than the regular languages.

**Proposition 1.** *For every NPA $\mathcal{A}$ of size $n$ and index $k$ there is an NBA $\mathcal{B}$ of size $\leq n \cdot c$ s.t. $L(\mathcal{A}) = L(\mathcal{B})$ where $c = \frac{k}{2} + 1$ if $k$ is even and $c = \lceil \frac{k}{2} \rceil$ if $k$ is odd.*

We quickly sketch the idea behind this construction because it is used in the comparison of the direct Ramsey-based methods for NPA with those for NBA. It is based on the fact that a run $\rho = q_0, q_1, \ldots$ of an NPA on a word is accepting iff there is an $i \in \mathbb{N}$ and an even priority $p$ s.t. for all $j \geq i$ we have $\Omega(q_j) \leq p$, and $\Omega(q_j) = p$ for infinitely many $j$. Thus, the required NBA can be constructed as follows. It contains a copy of $\mathcal{A}$ with the starting state and no final states. It also has, for every even priority $p$, a copy of $\mathcal{A}$ in which only states with priorities at most $p$ are preserved, and those with priority $p$ are final. The transitions in each copy are as they are in $\mathcal{A}$. Also, there are transitions from every state in the original copy to its successors in the additional copies if they exist. This way, the NBA can mimick an accepting run of the NPA by staying in the original non-final copy until no greater priorities than the one causing acceptance are seen, and then it changes into the respective copy verifying that this priority is being seen infinitely often, and no greater one is being seen anymore.

We are particularly interested in the following decision problems for NPA.

- $\textsc{Univ}^{\text{P}}$: Given an NPA $\mathcal{A}$, decide whether or not $L(\mathcal{A}) = \Sigma^\omega$.
- $\textsc{Incl}^{\text{P}}$: Given NPA $\mathcal{A}$ and $\mathcal{B}$, decide whether or not $L(\mathcal{A}) \subseteq L(\mathcal{B})$.

The complexity of these problems for NBA is well known; they are PSPACE-complete [14]. Together with Prop. 1 and the fact that every NBA is an NPA we immediately obtain the following.

**Proposition 2.** $\textsc{Univ}^{\text{P}}$ *and* $\textsc{Incl}^{\text{P}}$ *are PSPACE-complete.*

## 3 The Ramsey-Based Method for Parity Automata

In this section we describe how to decide universality and inclusion for NPA directly using a Ramsey-based method. We compare the results with the obvious method of translating NPA into NBA first and then using the Ramsey-based methods for NBA. We focus on universality first; inclusion proves to be just a small extension of this. The completeness proofs rely on the following theorem. For any linear order $(A, <)$ let $A^2_< := \{(a, b) \mid a, b \in A, a < b\}$.

**Theorem 1 (Ramsey, 1928).** *Let $F$ be a finite set and $c : \mathbb{N}^2_< \to F$. Then there is an $M \subseteq \mathbb{N}$ and an $f \in F$ such that $|M| = \infty$ and $c(i, j) = f$ for all $i, j \in M$ with $i < j$.*

### 3.1 Universality for NPA

For the remainder of this section fix an NPA $\mathcal{A} = (Q, \Sigma, q_0, \delta, \Omega)$ as well as $n := |\mathcal{A}|$ and $k := idx(\mathcal{A})$. Let $P = \{\Omega(q) \mid q \in Q\}$ be the set of all $\mathcal{A}$-priorities.

**Words as Partial Functions from State Pairs to Priorities.** We will use two total orders on the extension of $\mathbb{N}$ by one element $\dagger$. The first one is denoted $\leq$ and is the ordinary total order of type $\omega + 1$. Thus, we have $0 < 1 < \ldots < \dagger$. The *reward ordering* $\preceq$ is defined by $\dagger \ldots 3 \prec 1 \prec 0 \prec 2 \prec 4 \prec \ldots$ This reward ordering reflects the intuition of how valuable a priority of an NPA's state is for acceptance: even priorities are generally better than odd ones, and the bigger an even one the better, while small odd priorities are better than bigger ones because it is easier to subsume them in a run with an even priority elsewhere. Note that $\dagger$ is maximal for $\leq$ but minimal for $\preceq$.

**Definition 1.** A *box*[4] is a partial function of type $Q \times Q \dashrightarrow P$. We will sometimes write $f(q, q') = \dagger$ to denote that the value of the box $f$ on the argument pair $(q, q')$ is undefined.

Let $f, g$ be two boxes. Its *composition* $f \circ g$ is the box defined by

$$(f \circ g)(q, q') \;:=\; \max_{\preceq}\{\max_{\leq}\{f(q, q''), g(q'', q')\} \mid q'' \in Q\}$$

Note that the maxima are taken with respect to the two different total orders.

We will associate with every finite word $w \in \Sigma^*$ a box $[w] : Q \times Q \dashrightarrow P$ by induction on the length of $w$. The base cases for words of length 0 and 1 are the following.

$$[\epsilon](q, q') \;=\; \begin{cases} \Omega(q) & , \text{ if } q = q' \\ \dagger & , \text{ otherwise} \end{cases}$$

$$[a](q, q') \;=\; \begin{cases} \max_{\leq}\{\Omega(q), \Omega(q')\} & , \text{ if } q' \in \delta(q, a) \\ \dagger & , \text{ otherwise} \end{cases}$$

[4] See their graphical representation in Fig. 1 for an idea about the choice of this name.

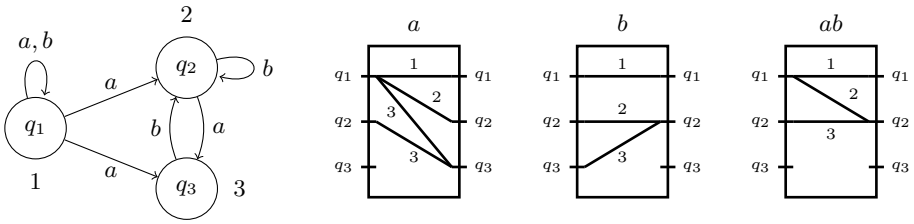


**Fig. 1.** An NPA with three of its boxes: $[a]$, $[b]$, and $[ab]$.

We then use box composition in order to lift this to arbitrary words: $[av] := [a] \circ [v]$.

Associativity of the composition operation is not hard to establish.

**Lemma 1.** *For all boxes $f, g, h$ we have $f \circ (g \circ h) = (f \circ g) \circ h$.*

With $\mathfrak{B}_{\mathcal{A}}$ we denote the set of all boxes defined by any word w.r.t. the NPA $\mathcal{A}$: $\mathfrak{B}_{\mathcal{A}} = \{[w] \mid w \in \Sigma^*\}$. It will be used as a search space in order to decide universality of $\mathcal{A}$.

*Example 1.* Boxes and the concept of composition can be visualised greatly. One regards the states in $Q$ as in- and out-ports of a connector. There can be connections between in-ports and out-ports, and these connections are labeled with a priority.

Fig. 1 shows an NPA with three states having the priorities $1, 2, 3$. It also shows the boxes $[a]$ and $[b]$ w.r.t. to this NPA as well as their composition $[ab]$ which, intuitively, is obtained by merging the out-ports of $[a]$ with the in-ports of $[b]$. The priority of a merged connection is the maximum w.r.t. $\leq$ of the two connections that are being merged. Note that this may result in more than one connection, for instance between $q_1$ and $q_2$ which can go via $q_2$ or $q_3$. In the composition, only the one with the maximum w.r.t. $\preceq$ survives, i.e. the one with priority 2 rather than 3.

**Definition 2.** A box $[w]$ is *idempotent* if $[w] \circ [w] = [w]$, where this equality is to be understood as equality of partial functions.

Let $[w]$ be a box and $q \in Q$. We write $q[w]$ for the set of all $q'$ that are connected to $q$ in this box, i.e. $q[w] = \{q' \mid [w](q, q') \neq \dagger\}$. Furthermore for some $Q' \subseteq Q$ let $Q'[w] := \bigcup_{q' \in Q'} q'[w]$.

A box $[w]$ is called *bad* w.r.t. some $Q' \subseteq Q$ if for all $q \in Q'[w]$ we have: $[w](q, q)$ is either undefined or odd. A *good* box is a box that is not bad. In other words, in a bad box w.r.t. some $Q'$ one considers first all connections from any input $q'$ to any output $q$, and then all the connections from such a $q$ to itself. We will consider bad boxes only in the context of idempotent boxes. Hence, this considers all connections in an infinite iteration of $[w]$ that are reachable from some $q' \in Q'$.

*Example 2.* It is easy to verify that $[ab]$ in Fig. 1 is idempotent. Note that any run in the NPA of Fig. 1 under the word $abab$ is also possible under $ab$ already, which is essentially what idempotency of $[ab]$ means.

Furthermore, $[ab]$ is bad for any subset of $\{q_1, q_2, q_3\}$, since following connections from any of them in $[ab]$ can only lead to a subset of $\{q_1, q_2\}$, and both connections from such a state to itself are labeled with an odd priority. Note that the word $(ab)^\omega$ cannot be accepted from any state in the NPA in Fig. 1 which is essentially what badness and idempotency of $[ab]$ means.

In order to prove correctness of the universality check to be presented, we need to relate boxes to runs. We write $q \xrightarrow{w}_p q'$ if state $q'$ is reachable from $q$ in the transition graph of $\mathcal{A}$ on a path whose labels compose to $w$ s.t. the highest (w.r.t. $\leq$) priority seen on this path is $p$. A proof of the following lemma is given in the appendix.

**Lemma 2.** *Let $w \in \Sigma^*$, $q, q' \in Q$, and $p \in P$. If $[w](q, q') = p$ then $q \xrightarrow{w}_p q'$.*

The converse direction is not true. Suppose that $q \xrightarrow{w}_p q'$ holds. Then we need not necessarily have $[w](q, q') = p$. simply because there may be different paths from $q$ to $q'$ in $\mathcal{A}$ under $w$, and $p$ may only be the maximal priority on one of them. However, $[w]$ only stores one maximal priority over all such paths, namely the greatest one w.r.t. $\preceq$. Thus, we have a statement that is weaker than the converse of Lemma 2 but still sufficient to prove correctness of the search procedures in the next section. Its proof is also given in the appendix.

**Lemma 3.** *Let $w \in \Sigma^*$, $q, q' \in Q$, and $p \in P$. If $q \xrightarrow{w}_p q'$ then there is a $p'$ s.t. $p \preceq p'$ and $[w](q, q') = p'$.*

**Non-Universality Via Relation Testing.** The following characterises (non)-universality of an NPA $\mathcal{A}$ in terms of the elements of $\mathfrak{B}_\mathcal{A}$.

**Theorem 2.** *$L(\mathcal{A}) \neq \Sigma^\omega$ iff there are boxes $[u]$ and $[v]$ s.t. $[v]$ is idempotent and bad w.r.t. $q_0[u]$.*

*Proof.* "$\Leftarrow$" Suppose that $[v]$ is idempotent and bad w.r.t. $q_0[u]$. We claim that $uv^\omega \notin L(\mathcal{A})$. For the sake of contradiction assume that $uv^\omega \in L(\mathcal{A})$. Let $q_0, q_1, \ldots$ be an accepting run of $\mathcal{A}$ on $uv^\omega$. Let $l = |u|$ and $k = |v|$. Clearly, we have $q_0 \xrightarrow{u}_p q_l$ for some $p \in P$. According to Lemma 3 we also have $[u](q_0, q_l) = p'$ for some $p' \in P$ with $p \preceq p'$.

Since there are only finitely many states, there must be some state $q$ that appears infinitely often in the sequence $q_l, q_{l+k}, q_{l+2k}, \ldots$. Let $i_0, i_1, \ldots$ denote the infinite ascending sequence of indices s.t. $q_{l+i_j k} = q$ for all $j$. Let $p_j = \max_{\leq}\{\Omega(q_{l+i_j k}), \ldots, \Omega(q_{l+i_{j+1} k})\}$. By assumption the run is accepting, hence, infinitely many $p_j$ must be even.

Let now $j$ be arbitrary s.t. $p_j$ is even. It follows that $q \xrightarrow{v^{i_{j+1}-i_j}}_{p_j} q$, and, by Lemma 3, that $[v^{i_{j+1}-i_j}](q, q) = p$ for some $p \succeq p_j$. Since $p_j$ is even, $p$ must be

even, too. Hence, $[v^{i_{j+1}-i_j}]$ is good for $q$. But since $[v]$ is idempotent, it follows that $[v^{i_{j+1}-i_j}] = [v]$, and this contradicts the assumption that $[v]$ is bad w.r.t. $q_0[u]$.

"⇒" Suppose that $L(\mathcal{A}) \neq \Sigma^\omega$, i.e. there is a word $w = a_0 a_1 \ldots \notin L(\mathcal{A})$. Consider the following colouring $c : \mathbb{N}^2 \to \mathfrak{B}_\mathcal{A}$ where $\mathbb{N}^2 := \{(i,j) \mid i \in \mathbb{N}, j \in \mathbb{N}, i < j\}$, defined by $c(i,j) = [a_i \ldots a_{j-1}]$. Since $|\mathfrak{B}_\mathcal{A}| < \infty$, Thm. 1 yields an infinite sequence $i_0, i_1, \ldots$ of indices and an $f \in \mathfrak{B}_\mathcal{A}$ s.t. $c(i_j, i_h) = f$ for all $j, h$ with $j < h$.

First define $u := a_0 \ldots a_{i_0-1}$. According to Lemma 2, for every $q \in q_0[u]$ we have $q_0 \xrightarrow{u}_p q$ for some $p$. Next, note that $f$ is idempotent because

$$\begin{aligned} f \circ f &= c(i_0, i_1) \circ c(i_1, i_2) = [a_{i_0} \ldots a_{i_1-1}] \circ [a_{i_1} \ldots a_{i_2-1}] = [a_{i_0} \ldots a_{i_2-1}] \\ &= c(i_0, i_2) = f \end{aligned}$$

according to Lemma 1. Then define $v_j := a_{i_j} \ldots a_{i_{j+1}-1}$ for every $j \in \mathbb{N}$. Note that $w = uv_0v_1v_2\ldots$, and that $f = [v_j]$ for every $j \in \mathbb{N}$.

It remains to be seen that $f$ is bad w.r.t. $q_0[u]$. Suppose that this was not the case, i.e. it was good. Then there would be a $q \in q'[v]$ for some $q' \in q_0[u]$ s.t. $p = [v](q,q)$ is even. According to Lemma 2 we would have $q_0 \xrightarrow{u}_{p'} q'$ for some $p'$, $q' \xrightarrow{v}_q p''$ for some $p''$ and $q \xrightarrow{v}_p q$. This can be iterated to form an infinite run $q_0, \ldots, q', \ldots, q, \ldots, q, \ldots$ on $uv_0v_1\ldots$ s.t. $p$ is the greatest priority (w.r.t. $\geq$) that is seen infinitely often on this run, because it is the greatest (w.r.t. $\geq$) that occurs on the parts from $q$ to itself. Since $p$ is even, this run would be accepting, contradicting the assumption that $w \notin L(\mathcal{A})$. □

Thm. 2 can then be used to decide (non-)universality as follows, see Algorithm 1. We keep generating boxes $[u], [v]$ to see whether some $[v]$ is idempotent and bad w.r.t. to $q_0[u]$. If this is the case then $uv^\omega$ cannot be accepted by the NPA. Finiteness of the space of all boxes guarantees termination. Algorithm UP uses a set $V$ in order to store such boxes $[v]$ whereas boxes $[u]$ need not be stored explicitly. It suffices to store all states which can be reached from the initial state under some $u \in \Sigma^*$. A set $R$ is maintained in order to track all states that are reachable from the initial state with corresponding witnessing words, since in a non-universality check they all need to be tested for non-extendability with a loop.

Note that using sets of boxes is not necessarily the most clever way of implementing this algorithm. One can use priority lists etc. in order to avoid testing the same pair of boxes multiple times in line 7. Also, the step in line 14 is meant to remove pairs $(u, Q')$ from $R$ only for as long as there is still another $(v, Q'') \in R$.

At last, we analyse the asymptotic complexity of testing NPA universality this way.

**Theorem 3.** *For an NPA $\mathcal{A}$ with $|\mathcal{A}| = n$ and $idx(\mathcal{A}) = k$, algorithm* UP *tests universality in time $\mathcal{O}(2^{2((n^2 \log k)+n)})$.*

**Algorithm 1** UP for $\textsc{Univ}^{\mathrm{P}}$

$R \leftarrow \{(\epsilon, \{q_0\})\}$
$V \leftarrow \{[a] \mid a \in \Sigma\}$
$R' \leftarrow \emptyset$
$V' \leftarrow \emptyset$
**while** $R \neq R'$ **or** $V \neq V'$ **do**
  **for** $[v] \in V$ that are idempotent **do**
    **if** $\exists (u, Q') \in R$ s.t. $[v]$ bad w.r.t. $Q'$ **then**
      **return** "$L(\mathcal{A})$ is not universal: $uv^\omega \notin L(\mathcal{A})$"
    **end if**
  **end for**
  $R' \leftarrow R$
  $V' \leftarrow V$
  $R \leftarrow R \cup \{(uv, Q'[v]) \mid (u, Q') \in R, [v] \in V\}$
  reduce $R$ s.t. it contains at most one $(u, Q')$ for every $Q' \subseteq Q$
  $V \leftarrow V \cup V \circ V$
**end while**
**return** "$L(\mathcal{A})$ is universal"

*Proof.* First observe that the set of boxes $V$ can only increase (or stay the same) in an iteration of the algorithm, hence there can be at most $(k+1)^{n^2}$ many different sets $V$ in a run. Second, observe that the set of reachable state sets $R$ can only increase modulo inclusion in an iteration of the algorithm, hence, there cannot be more than $2^n$ many different sets $R$ in a run. It follows that the number of iterations is bounded by $(k+1)^{n^2} + 2^n = \mathcal{O}(2^{n^2 \log k})$.

The number of iterations of the inner loop is bounded by $|V| \cdot |R|$, which is $(k+1)^{n^2} \cdot 2^n = \mathcal{O}(2^{(n^2 \log k)+n})$ in the worst case. The other operations in the outer loop can be easily bounded by the same term. □

Remember that an NBA is just an NPA with priorities $1, 2$. Hence, algorithm UP can be restricted to NBA as input as well. In order to refer to it later, we call this restriction UB. It cannot be distinguished from UP in terms of pseudo code. The difference is that the search space is only of size $2^{n^2 \log 3}$. We also remark that UB coincides with the previously known Ramsey-based universality test for NBA [7].

### 3.2 Inclusion for NPA

There is a conceptually simple but evidently not well-known reduction from the inclusion problem to the universality problem for finite automata which can also be made to work for NPA. A proof sketch is given in the appendix.

**Proposition 3.** *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two NPA over some alphabet $\Sigma$, s.t. $\mathcal{A}_i$ has size $n_i$ states, $e_i$ transitions and index $k_i$ for $i \in \{1, 2\}$. There is an NPA $\mathcal{A}$ with $4 + n_1 + k_1 + n_2$ states and index $\max\{2, k_1, k_2\}$ over some alphabet $\Delta$ with $|\Delta| = e_1$ s.t. $L(\mathcal{A}) = \Sigma^\omega$ iff $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$.*

This construction, when applied to two NBA, does not necessarily yield an NBA. With a minor modification though it can also be used to reduce $\textsc{Incl}^{\mathrm{B}}$ to $\textsc{Univ}^{\mathrm{B}}$. The resulting automaton would have $3n_2 + n_1 + 1$ states.

This, together with Thm. 3 clearly yields a Ramsey-based algorithm for the inclusion problem for NPA. However, there is also a direct method which is asymptotically better. For the remainder of this section fix two NPA $\mathcal{A} = (Q^{\mathcal{A}}, \Sigma, q_0^{\mathcal{A}}, \delta^{\mathcal{A}}, \Omega^{\mathcal{A}})$ and $\mathcal{B} = (Q^{\mathcal{B}}, \Sigma, q_0^{\mathcal{B}}, \delta^{\mathcal{B}}, \Omega^{\mathcal{B}})$. We are interested to know whether or not $L(\mathcal{A}) \subseteq L(\mathcal{B})$ holds. Let $P^{\mathcal{A}} := \{\Omega^{\mathcal{A}}(q) \mid q \in Q^{\mathcal{A}}\}$ be the set of all priorities occurring in $\mathcal{A}$ and $P^{\mathcal{B}}$ be defined likewise.

Remember that in the previous section we associated to every word $w \in \Sigma^*$ a unique box $[w]$. This is not possible anymore; a word can be associated with several objects of type

$$Q^{\mathcal{A}} \times P^{\mathcal{A}} \times Q^{\mathcal{A}} \times \left(Q^{\mathcal{B}} \times Q^{\mathcal{B}} \to P^{\mathcal{B}}\right) .$$

Thus, such an object is obtained by extending a box for $\mathcal{B}$—as defined in the previous section—with two states and a priority of $\mathcal{A}$. We call these objects *typed boxes* because the two states of $Q^{\mathcal{A}}$ act as input and output types for the composition on them. A typed box of the form $(q, p, q', [w])$ is written ${}^{q}[w]_{p}^{q'}$.

A typed box for the empty word $\epsilon$ is ${}^{q}[\epsilon]_{\Omega(q)}^{q}$ for any $q \in Q^{\mathcal{A}}$, a typed box for words $a$ of length 1 is ${}^{q}[a]_{\max\{\Omega(q),\Omega(q')\}}^{q'}$ for any $(q, a, q') \in \delta^{\mathcal{A}}$ and the composition of two typed boxes extends the composition of boxes in the following way:

$${}^{q}[u]_{p}^{q'} \circ {}^{q'}[v]_{p'}^{q''} \;:=\; {}^{q}[uv]_{m}^{q''}$$

where $m := \max_{\leq}\{p, p'\}$. Note that this composition is only defined if the output type of the left component equals the input type of the right component. We write $\mathfrak{B}_{\mathcal{A},\mathcal{B}}$ for the space of all typed boxes for the pair $\mathcal{A}, \mathcal{B}$ of NPA. A proof of the following lemma is given in the appendix.

**Lemma 4.** *Let $w \in \Sigma^*$, $q, q' \in Q^{\mathcal{A}}$, $p \in P^{\mathcal{A}}$, $s, s' \in Q^{\mathcal{B}}$, and $p' \in P^{\mathcal{B}}$. If ${}^{q}[w]_{p}^{q'}(s, s') = p'$ then $q \xrightarrow{w}_{p} q'$ and $s \xrightarrow{w}_{p'} s'$.*

An *idempotent* typed box is, as usual, a ${}^{q}[w]_{p}^{q'}$ s.t. ${}^{q}[w]_{p}^{q'} \circ {}^{q}[w]_{p}^{q'} = {}^{q}[w]_{p}^{q'}$. Note that this necessarily requires $q = q'$. A *bad* typed box w.r.t. some $Q' \subseteq Q^{\mathcal{B}}$ is a ${}^{q}[w]_{p}^{q'}$ s.t. $p$ is even and the underlying untyped box $[w]$ is bad w.r.t. $Q'$ in the sense of the previous section, i.e. there is a $q \in Q^{\mathcal{B}}$ s.t. $[w](q, q)$ is undefined or odd.

**Theorem 4.** *We have $L(\mathcal{A}) \not\subseteq L(\mathcal{B})$ iff there are $u \in \Sigma^*$, $v \in \Sigma^+$ and typed boxes ${}^{q_0^{\mathcal{A}}}[u]_{p}^{q}$ and ${}^{q}[v]_{p'}^{q}$ s.t. ${}^{q}[v]_{p'}^{q}$ is idempotent and bad w.r.t. $q_0^{\mathcal{B}}[u]$.*

*Proof.* "$\Leftarrow$" Suppose there are ${}^{q_0^{\mathcal{A}}}[u]_{p}^{q}$ and ${}^{q}[v]_{p'}^{q}$ s.t. ${}^{q}[v]_{p'}^{q}$ is idempotent and bad w.r.t. $q_0^{\mathcal{B}}[u]$. Using Lemma 4 we get a run $q_0^{\mathcal{A}}, \ldots, q, \ldots, q, \ldots$ of $\mathcal{A}$ on $uv^{\omega}$, s.t. the maximal priority occurring infinitely often in this run is $p'$. This $p'$ must

be even for otherwise ${}^{q}[v]^{q}_{p'}$ would not be bad. Hence, we have $uv^\omega \in L(\mathcal{A})$. It remains to be seen that $uv^\omega \notin L(\mathcal{B})$.

Suppose $q_0, q_1, \ldots$ was an accepting run of $\mathcal{B}$ on $uv^\omega$. Note that if ${}^{q}[v]^{q}_{p'}$ is idempotent and bad w.r.t. some set $Q'$, then so is its underlying untyped box $[v]$. Thus, $uv^\omega \notin L(\mathcal{B})$ can be proved as in the "$\Leftarrow$"-part of the proof of Thm. 2.

"$\Rightarrow$" Suppose there is a $w = a_0 a_1 \ldots \in L(\mathcal{A}) \cap \overline{L(\mathcal{B})}$. Take an accepting run $q_0, q_1, \ldots$ of $\mathcal{A}$ on $w$. Here we consider the colouring $c : \mathbb{N}^2 \to \mathfrak{B}_{\mathcal{A},\mathcal{B}}$ defined by $c(i,j) = {}^{q_i}[a_i \ldots a_j]^{q_{j+1}}_{p}$ where $p$ is the maximal (w.r.t. $\leq$) priority occuring in the sequence $\Omega^{\mathcal{A}}(q_i), \ldots, \Omega^{\mathcal{A}}(j+1)$. Since $\mathfrak{B}_{\mathcal{A},\mathcal{B}}$ is finite, Thm. 1 yields words $u \in \Sigma^*$, $v \in \Sigma^+$ s.t. ${}^{q_{|u|}}[v]^{q_{|u|}}_{p'}$ is idempotent. In a way analogous to the "$\Rightarrow$"-part of the proof of Thm. 2 one can show that ${}^{q_{|u|}}[v]^{q_{|u|}}_{p'}$ is bad w.r.t. $q_0[u]$. □

Inclusion can then be tested again by searching for an idempotent bad box w.r.t. to some set of reachable states. Here we maintain a set $R$ of triples $(u, q, Q')$ s.t. $q$ is reachable in $\mathcal{A}$ from $q_0^{\mathcal{A}}$ under $u$, and all $q' \in Q'$ are reachable from $q_0^{\mathcal{B}}$ under $u$ in $\mathcal{B}$.

**Algorithm 2** IP for $\textsc{Incl}^{\mathrm{P}}$

$R \leftarrow \{(\epsilon, q_0^{\mathcal{A}}, \{q_0^{\mathcal{B}}\})\}$
$V \leftarrow \{{}^{q}[a]^{q'}_{p} \mid (q, a, q') \in \delta^{\mathcal{A}}, p = \Omega(q)\}$
$R' \leftarrow \emptyset$
$V' \leftarrow \emptyset$
**while** $R \neq R'$ **or** $V \neq V'$ **do**
  **for** ${}^{q}[v]^{q}_{p} \in V$ that are idempotent **do**
    **if** $\exists (u, q, Q') \in R$ s.t. ${}^{q}[v]^{q}_{p}$ bad w.r.t. $Q'$ **then**
      **return** "$L(\mathcal{A}) \not\subseteq L(\mathcal{B})$: $uv^\omega \in L(\mathcal{A}) \cap \overline{L(\mathcal{B})}$"
    **end if**
  **end for**
  $R' \leftarrow R$
  $V' \leftarrow V$
  $R \leftarrow R \cup \{(uv, q', Q'[v]) \mid {}^{q}[v]^{q'}_{p} \in V, (u, q, Q') \in R\}$
  reduce $R$ s.t. it contains at most one $(u, q, Q')$ for every pair $q, Q'$
  $V \leftarrow V \cup V \circ V$
**end while**
**return** "$L(\mathcal{A}) \subseteq L(\mathcal{B})$"

**Theorem 5.** *Algorithm 2 tests inclusion in time* $\mathcal{O}(2^{2((n^2 \log k)+\log m+\log k')})$ *for NPA* $\mathcal{A}$ *and* $\mathcal{B}$ *with* $m$ *resp.* $n$ *states and index* $k'$ *resp.* $k$.

*Proof.* As in Thm. 3, it is not hard to see that the number of outer iterations is bounded by the maximal number of typed boxes and reachability sets. Since the number of typed boxes is an upper bound on the number of reachability sets, we observe that the number of iterations can be bounded by $\mathcal{O}(2^{(n^2 \log k)+\log m+\log k'})$. The runtime of the inner loop as well as the other operations can also be bounded by $\mathcal{O}(2^{(n^2 \log k)+\log m+\log k'})$. □

We call IB the restriction of IP to NBA only. Its search space is decreased to $m \cdot 2^{(n^2 \log 3)+\log 3}$.

### 3.3 Comparing Direct and Indirect Methods

Regarding the universality problem for NPA we consider two different approaches: (1) using algorithm UP directly, and (2) translating an NPA into an NBA and then using UB, the restriction of UP to NBA. The asymptotic worst-case runtimes for these two approaches compare as follows. As usual, $n$ denotes the number of states of the input NPA, $k$ denotes its index.

| | reductions | algorithm | runtime |
|---|---|---|---|
| (1) | —— | UP | $2^{\mathcal{O}(n^2 \log k)}$ |
| (2) | $\textsc{Univ}^{\mathrm{P}} \mapsto \textsc{Univ}^{\mathrm{B}}$ | UB | $2^{\mathcal{O}(n^2 k^2)}$ |

This shows that the direct method devised here is asymptotically much better than the one that can be obtained from previously known reductions and methods for NBA: (1) is polynomial in the number of occurring priorities, (2) is exponential in the square of this number. Thus, one should expect the direct method to perform much better on NPA with more than just a very small number of priorities.

Now consider the inclusion problem between an NPA with $m$ states and index $k'$ and an NPA with $n$ states and index $k$. There are even more Ramsey-based approaches available:

1. the direct method using algorithm IP;
2. translating both NPA into NBA (Prop. 1), then using IB;
3. the reduction from inclusion to universality on the NPA side (Prop. 3), then using algorithm UP;
4. reducing the inclusion problem for NPA to the universality problem (Prop. 3), then translating the resulting single NPA into an NBA (Prop. 1), and testing it for universality with algorithm UB;
5. first translating the NPA into NBA (Prop. 1), then performing the reduction from inclusion to universality on the NBA side (Prop. 3), and finally using algorithm UB as well.

The asymptotic worst-case runtimes are as follows.

| | reductions | alg. | runtime |
|---|---|---|---|
| (1) | —— | IP | $2^{\mathcal{O}(n^2 \log k+\log(mk'))}$ |
| (2) | $\textsc{Incl}^{\mathrm{P}} \mapsto \textsc{Incl}^{\mathrm{B}}$ | IB | $2^{\mathcal{O}((nk)^2+\log(mk'))}$ |
| (3) | $\textsc{Incl}^{\mathrm{P}} \mapsto \textsc{Univ}^{\mathrm{P}}$ | UP | $2^{\mathcal{O}((n+k+m)^2+\log k')}$ |
| (4) | $\textsc{Incl}^{\mathrm{P}} \mapsto \textsc{Univ}^{\mathrm{P}} \mapsto \textsc{Univ}^{\mathrm{B}}$ | UB | $2^{\mathcal{O}((n+k+m)^2 \cdot (\max\{k,k'\})^2)}$ |
| (5) | $\textsc{Incl}^{\mathrm{P}} \mapsto \textsc{Incl}^{\mathrm{B}} \mapsto \textsc{Univ}^{\mathrm{B}}$ | UB | $2^{\mathcal{O}((nk+mk')^2)}$ |

One can vaguely say that the more reductions one uses, the worse the asymptotic runtime gets. Again, only the direct method devised here is polynomial in the number of involved priorities whereas using any of the four other methods involving a reduction of some kind results in a runtime that is exponential in at least the number of different priorities in one of the involved automata.

## 4 Experimental Evaluation

The previous section argues that the direct Ramsey-based methods for parity automata devised in this paper are asymptotically, i.e. in theory, better than the methods one can obtain through reductions to Ramsey-based methods for Büchi automata. In this section, we show that this is also the case in practice. Due to space restrictions we restrict ourselves to the universality problem. Preliminary tests with the inclusion problem also show that the direct methods outperform those obtained by reductions.

### 4.1 A Random Model of Parity Automata

We extend the Tabakov-Vardi random model for NBA [15] to one for NPA. It is parameterized by two natural numbers $n > 0$ and $p > 0$ that result in the following automata scheme for an NPA $(Q, \Sigma, 1, \delta, \Omega)$ where $Q = \{1, \ldots, n\}$, $\Sigma = \{a, b\}$, and $\delta$ and $\Omega$ are chosen arbitrarly at random by the following distribution:

- $q' \in \delta(q, s)$ with probability $\frac{2}{n}$ for every $1 \leq q, q' \leq n$ and every $s = a, b$, and
- $\Omega(q) = \begin{cases} 2p'+1 & \text{with probability } \frac{1}{2p} \text{ and } 0 \leq p' < p \\ 2p'+2 & \text{with probability } \frac{1}{2p} \text{ and } 0 \leq p' < p \end{cases}$

In other words, an NPA of this model has $n$ states, an alphabet of size two, an expected transition density of two outgoing edges per state and symbol and a priority assignment that maps every state to a priority based on a uniform distribution of $1, \ldots, 2p$. Experimentally it can be seen that this results in an NPA accepting the universal language with probability of approx. 50%.

### 4.2 Comparison in Practice

All tests have been carried out on a 64-bit machine with four quad-core Opteron™ CPUs. The implementation does not support parallel computations, hence, each test is run on one core only. The following tables feature the average time to decide universality over 1000 automata of a certain parameterization of the random model. They also show the average rounded number of boxes that have been created during these tests.

The first benchmark measures the effect of the number of states on the runtime. Thus, it fixes $p = 2$, i.e. the only priorities occurring are $1, \ldots, 4$. The

| Benchmark 1 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | universal | | | | non-universal | | | |
| | UP | | UB | | UP | | UB | |
| states | time | boxes | time | boxes | time | boxes | time | boxes |
| 5 | 0.00s | 21 | 0.01s | 23 | 0.00s | 5 | 0.00s | 6 |
| 10 | 0.08s | 190 | 0.95s | 583 | 0.01s | 53 | 0.04s | 64 |
| 15 | 1.23s | 817 | 70.39s | 6,388 | 0.07s | 145 | 0.57s | 272 |
| 20 | 3.90s | 1,497 | 1,555.04s | 40,776 | 0.46s | 401 | 2.98s | 811 |
| 25 | 19.70s | 3,877 | 1,867.92s | 43,728 | 0.90s | 648 | 2.46s | 846 |
| 30 | 72.62s | 6,486 | — | — | 2.70s | 1,106 | 49.18s | 4,780 |
| 35 | 154.67s | 8,868 | — | — | 5.11s | 1,489 | 59.44s | 5,901 |
| 40 | 221.01s | 11,318 | — | — | 10.93s | 2,112 | 70.26s | 6,601 |

| Benchmark 2 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | universal | | | | non-universal | | | |
| | UP | | UB | | UP | | UB | |
| priorities | time | boxes | time | boxes | time | boxes | time | boxes |
| 2 | 0.97s | 745 | 1.14s | 677 | 0.05s | 114 | 0.08s | 111 |
| 4 | 2.74s | 1,370 | 29.57s | 5,294 | 0.15s | 200 | 0.92s | 238 |
| 6 | 2.89s | 1,479 | 797.65s | 13,049 | 0.17s | 255 | 1.79s | 332 |
| 8 | 5.28s | 2,297 | 1,158.08s | 28,261 | 0.22s | 327 | 2.02s | 511 |
| 10 | 3.56s | 2,226 | — | — | 0.34s | 400 | 6.39s | 939 |
| 12 | 4.03s | 2,120 | — | — | 0.33s | 477 | 8.37s | 1,498 |
| 14 | 4.13s | 1,766 | — | — | 0.23s | 374 | 10.69s | 1,450 |
| 16 | 4.36s | 2,755 | — | — | 0.31s | 450 | 31.11s | 1,402 |

**Fig. 2.** Average runtimes and number of created boxes in the benchmarks.

results are presented in the first table in Fig. 2. The average runtimes distinguish the two cases of NPA accepting the universal language and a non-universal language because non-universality is much easier to establish than universality. Note that the latter requires the creation of all boxes while the former only needs to find a counterexample. This benchmark shows very clearly that the direct Ramsey-based method UP for NPA is much faster in practice than the method UB on NBA that have been obtained by translating NPA into NBA.

The second benchmark measures the effect that the number of different priorites has on the runtime. It fixes $n = 16$, i.e. every automaton has exactly 16 states. See the second table in Fig. 2 for the results. Again, the direct method of algorithm UP outperforms the indirect method of algorithm UB by far.

## 5 Conclusion and Further Work

We have presented direct Ramsey-based methods that solve the universality and inclusion problem for nondeterministic parity automata. These direct methods turn out to be more efficient than indirect methods obtained from translating parity into Büchi automata and then performing the corresponding Ramsey-

based analysis on these. Also, the reduction from inclusion to universality is equally non-viable in this context.

The work presented here can be continued into several directions. It remains to be seen whether optimisations for Ramsey-based methods as they can be done for NBA [1, 2] can be lifted to yield similar speed-ups in the Ramsey-based methods for NPA.

## References


1. P. A. Abdulla, Y.-F. Chen, L. Clemente, L. Holík, C.-D. Hong, R. Mayr, and T. Vojnar. Simulation subsumption in ramsey-based Büchi automata universality and inclusion testing. In *Proc. 22nd Int. Conf. on Computer Aided Verification, CAV'10*, volume 6174 of *LNCS*, pages 132–147. Springer, 2010.
2. P. A. Abdulla, Y.-F. Chen, L. Clemente, L. Holík, C.-D. Hong, R. Mayr, and T. Vojnar. Advanced ramsey-based Büchi automata inclusion testing. In *Proc. 22nd Int. Conf. on Concurrency Theory, CONCUR'11*, LNCS. Springer, 2011.
3. B. Banieqbal and H. Barringer. Temporal logic with fixed points. In *Proc. Coll. on Temporal Logic in Spec.*, volume 398 of *LNCS*, pages 62–73. Springer, 1989.
4. J. R. Büchi. On a decision method in restricted second order arithmetic. In *Proc. Congress on Logic, Method, and Philosophy of Science*, pages 1–12, Stanford, CA, USA, 1962. Stanford University Press.
5. C. Dax, M. Hofmann, and M. Lange. A proof system for the linear time $\mu$-calculus. In *Proc. 26th Conf. on Foundations of Software Technology and Theoretical Computer Science, FSTTCS'06*, volume 4337 of *LNCS*, pages 274–285. Springer, 2006.
6. S. Fogarty and M. Y. Vardi. Büchi complementation and size-change termination. In *Proc. 15th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'09*, volume 5505 of *LNCS*, pages 16–30. Springer, 2009.
7. S. Fogarty and M. Y. Vardi. Efficient Büchi universality checking. In *Proc. 16th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'10*, volume 6015 of *LNCS*, pages 205–220. Springer, 2010.
8. N. Klarlund. Progress measures for complementation of $\omega$-automata with applications to temporal logic. In *Proc. 32nd Annual Symp. on Foundations of Computer Science, FOCS'91*, pages 358–367. IEEE, 1991.
9. O. Kupferman and M. Y. Vardi. Weak alternating automata are not that weak. *ACM Transactions on Computational Logic*, 2(3):408–429, 2001.
10. C. S. Lee, N. D. Jones, and A. M. Ben-Amram. The size-change principle for program termination. *ACM SIGPLAN Notices*, 36(3):81–92, 2001.
11. A. Pnueli. The temporal logic of programs. In *Proc. 18th Symp. on Foundations of Computer Science, FOCS'77*, pages 46–57, Providence, RI, USA, 1977. IEEE.
12. F. P. Ramsey. On a problem of formal logic. *Proc. London Mathematical Society, Series 2*, 30(4):338–384, 1928.
13. S. Safra. On the complexity of $\omega$-automata. In *Proc. 29th Symp. on Foundations of Computer Science, FOCS'88*, pages 319–327. IEEE, 1988.
14. A. P. Sistla, M. Y. Vardi, and P. Wolper. The complementation problem for Büchi automata with applications to temporal logic. *TCS*, 49(2–3):217–237, 1987.
15. D. Tabakov and M. Y. Vardi. Experimental evaluation of classical automata constructions. In *In LPAR 2005, LNCS 3835*, pages 396–411. Springer, 2005.
16. M. Y. Vardi. A temporal fixpoint calculus. In ACM, editor, *Proc. Conf. on Principles of Programming Languages, POPL'88*, pages 250–259, NY, USA, 1988. ACM.