Towards a Grand Unification of Büchi Complementation Constructions

Moshe Y. Vardi[1], Seth Fogarty[2], Yong Li[3] and Yih-Kuen Tsay[4]

[1] Department of Computer Science, Rice University
[2] Department of Computer Science, Trinity University
[3] State Key Laboratory of Computer Science,
Institute of Software, Chinese Academy of Sciences
[4] National Taiwan University

Abstract. The complementation construction 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 is typically done by determinization using the subset construction. 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. Over the years, three different constructions have been developed for Büchi complementation, based on congruence relations (via Ramsey analysis), progress ranks, and profiles. In this work we unify the three constructions, by showing how profiles can also yield both optimal congruence relations and progress ranks.

1 Introduction

Complementation of nondeterministic word automta is a fundamental operation for both the automata-based model checking [40] and program termination checking [14]. In particular, in the automata-based model checking [40] framework, the verification problem of whether the behavior of a system A satisfies a specification B reduces to a language-containment problem between the corresponding automata modelling A and B, which is then reduced to the intersection of A and the complement of B. For verifying safety properties, automata on finite words [22] are sufficient, while for liveness properties, nondeterministic Büchi automata on *infinite* words (NBWs) [6], are usually utilized [40], as verification requires to represent the properties of nonterminating systems with infinite-length behaviors. This work focuses on the *complementation* of NBWs.

It is known that for automata on finite words, complementation involves determinization via the standard subset construction [18]. For NBWs, however, optimal complementation constructions are quite complicated, as the subset construction is not sufficient [34]. NBWs were originally proposed to prove the decidability of a restricted monadic second-order logic [6]. In this work [6], Büchi formulated the first proof of the existence of a complementary NBW implicitly using second-order formulae. Büchi's idea for complementation relies on a congruence relation and one can associate each equivalence class of the congruence relation with a state of the complementary automaton, similarly to the characterization provided by the Myhill-Nerode theorem for regular languages [18]. Due to the use of Ramsey's theorem in the proof, Büchi's complementation is widely referred to as *Ramsey-based* Büchi complementation. In this work, we will refer Büchi's complementation to as the *congruence-based* construction because it is defined with congruence relations. The congruence-based construction was later improved by Sistla *et al.* in 1987 with a blow-up of $2^{\mathcal{O}(n^2)}$ [34]. In 1988, Safra described a $2^{\mathcal{O}(n \log n)}$ complementation [32], widely known as the *determinization-based* complementation, matching the lower bound $2^{\Omega(n \log n)}$ established by Michel [27]. Work on complementation since then has focused on either providing simpler complementation algorithms with optimal complexity $2^{\mathcal{O}(n \log n)}$, such as *rank-based* complementation [22] and *slice-based* complementation [19], or further tightening the lower and upper bounds [33,41].

Over the past few decades, there are mainly three types of complementation constructions that have been actively studied, namely the congruence-based (alternatively Ramsey-based) [32,5,14,13,1,2,24], rank-based [22,16,15,20,33,12,8,17], and slice-based [19,12,36,3] constructions. Existing *direct* complementation constructions, in contrast to those going through determinization, all fall into one of the three types. The core idea of a complementation construction for NBWs is to identify the set of words rejected by the input NBW. To that end, rank-based constructions rely on progress ranks [21], while slice-based constructions were originally based on the construction of reduced split trees. In [12], the rank-based construction and slice-based construction are unified with a profile-based analysis, yielding a construction based on retrospective ranking.

Both rank-based and slice-based complementation constructions are based on tracking the (reduced) run DAG of an ω-word and determine whether the corresponding ω-word is accepted. Both constructions see an ω-word as a whole, while the congruence-based construction decomposes an ω-word into a finite prefix and a periodic finite word. This is why the congruence-based approach seems difficult to be related or unified with the rank-based and slice-based constructions so far. In this work, we further show that the profile-based analysis can also be applied to the congruence-based construction. This novel insight shows the connection between profiles and congruences, concluding that the three major constructions can all be unified based on profiles.

2 Preliminaries

We fix an *alphabet* Σ, and define a *word* as a finite or infinite sequence of letters from Σ. Let Σ^* and Σ^ω denote the set of all finite and infinite words (or ω-words), respectively. A *finitary language* is a subset of Σ^*; an *ω-language* is a subset of Σ^ω. Let L be a finitary language (resp., ω-language); the complementary language of L, written as $\overline{L}$, is $\Sigma^* \setminus L$ (resp., $\Sigma^\omega \setminus L$). Let ρ be a sequence; we denote by $\rho[i]$ the i-th element of ρ, and by $\rho[i..k]$ the subsequence of ρ starting at the i-th element and ending at the k-th element, inclusively. When $i > k$, then $\rho[i..k]$ is taken to be the empty sequence ϵ. We denote by $|\rho|$ the number of elements in ρ. We denote by $[m]$ the set $\{0, \cdots, m-1\}$ for $m > 0$. Given a finite word u and a word w, we denote by $u \cdot w$ (uw, for short) the concatenation of u and w. Given a finitary language L_1 and a finitary/ω-language L_2, the concatenation $L_1 \cdot L_2$ ($L_1 L_2$, for short) of L_1 and L_2 is the set $\{\, u \cdot w \mid u \in L_1, w \in L_2 \,\}$ and L_1^ω the infinite concatenation of L_1.

Automata. A (nondeterministic) automaton is a tuple $\mathcal{A} = (Q, I, \delta, F)$, where Q is a finite set of states, $I \subseteq Q$ is a set of initial states, $\delta \colon Q \times \Sigma \to 2^Q$ is a transition function, and $F \subseteq Q$ is a set of accepting states. We extend δ to sets $S \subseteq Q$ of states, by letting $\delta(S, a) = \bigcup_{q \in S} \delta(q, a)$. We also extend δ to finite words, by letting $\delta(S, \epsilon) = S$ and $\delta(S, a_1 a_2 \cdots a_k) = \delta(\delta(S, a_1), \cdots, a_k)$ for $k \geq 1$. When running on finite words, an automaton is called a *nondeterministic automaton on finite words* (NFW), while an automaton on ω-words is called a *nondeterministic Büchi automaton on infinite words* (NBW). An automaton $\mathcal{A}$ is said to be *deterministic* if $|I| = 1$ and, for each $q \in Q$ and $a \in \Sigma$, it holds that $|\delta(q, a)| \leq 1$; when considered on finite words, $\mathcal{A}$ is called a DFW, while in the context of infinite words, it is called DBW.

A *run* of an NFW/NBW $\mathcal{A}$ on a finite word u of length $n \geq 0$ is a sequence of states $\rho = q_0 q_1 \cdots q_n \in Q^+$, such that $q_0 \in I$ and $q_i \in \delta(q_{i-1}, u[i])$ for every $0 < i \leq n$. A finite word $u \in \Sigma^*$ is *accepted* by an NFW $\mathcal{A}$ if there is a run $q_0 \cdots q_n$ over u such that $q_n \in F$. Similarly, an *ω-run* of an NBW $\mathcal{A}$ on an ω-word w is an infinite sequence of states $\rho = q_0 q_1 \cdots$ such that $q_0 \in I$ and, for every $i > 0$, $q_i \in \delta(q_{i-1}, w[i])$. Let $\mathit{Inf}(\rho)$ be the set of states that occur infinitely often in the run ρ. An ω-word $w \in \Sigma^\omega$ is *accepted* by an NBW $\mathcal{A}$ if there is an ω-run ρ of $\mathcal{A}$ over w such that $\mathit{Inf}(\rho) \cap F \neq \emptyset$. The *finitary language* recognized by an NFW $\mathcal{A}$, denoted by $\mathcal{L}_*(\mathcal{A})$, is defined as the set of finite words accepted by it. Similarly, we denote by $\mathcal{L}(\mathcal{A})$ the *ω-language* recognized by an NBW $\mathcal{A}$, i.e., the set of ω-words accepted by $\mathcal{A}$. The complementation construction of $\mathcal{A}$ is to construct an NBW that accepts the complementary language of $\mathcal{A}$, i.e., $\overline{\mathcal{L}(\mathcal{A})}$. We note that the index of words starts with 1, in contrast to 0 for other sequences, such as runs.

Directed acyclic graphs of runs (or run DAGs) were proposed by Kupferman and Vardi in [23] for reasoning about all runs of an NBW on a given ω-word w. Let $\mathcal{A} = (Q, I, \delta, F)$ be an NBW and w be an ω-word. The run DAG $\mathcal{G}_w = \langle V, E \rangle$ of $\mathcal{A}$ over w is defined as follows:

- The vertices $V \subseteq Q \times \mathbb{N}$ is the set $\{\, \langle q, l \rangle \mid l \in \mathbb{N}, q \in \delta(I, w[1..l]) \,\}$.
- Edges: There is an edge from $\langle q, l \rangle$ to $\langle q', l' \rangle$ if $l' = l + 1$ and $q' \in \delta(q, w[l'])$.

A vertex $\langle q, l \rangle$ is said to be on level l; note there are at most $|Q|$ states on each level. A vertex $\langle q, l \rangle$ is called an *F-vertex* if $q \in F$. A finite (infinite) sequence of vertices $\hat{\rho} = \langle q_0, 0 \rangle \langle q_1, 1 \rangle \cdots$ is called a *branch* (resp. *ω-branch*) of $\mathcal{G}_w$ when for each $0 \leq l < k$ (resp. $0 \leq l$), there is an edge from $\langle q_l, l \rangle$ to $\langle q_{l+1}, l + 1 \rangle$.

A vertex $\langle q_j, j \rangle$ is *reachable* from $\langle q_l, l \rangle$ if there is a path from $\langle q_l, l \rangle$ to $\langle q_j, j \rangle$. We call a vertex $\langle q, l \rangle$ *finite* in $\mathcal{G}_w$ if it is not on an ω-branch; we call a vertex

$\langle q, l \rangle$ *F-free* if it is not finite and no F-vertices are reachable from $\langle q, l \rangle$ in $\mathcal{G}_w$. To a run $\rho = q_1 q_2 \cdots$ of $\mathcal{A}$ over w corresponds an ω-branch $\hat{\rho} = \langle q_0, 0 \rangle \langle q_1, 1 \rangle \cdots$. Therefore, w is accepted by $\mathcal{A}$ iff there is an ω-branch in $\mathcal{G}_w$ visiting F-vertices infinitely often; such an ω-branch is said to be *accepting*. $\mathcal{G}_w$ is accepting iff there is an accepting ω-branch in $\mathcal{G}_w$.

3 Complementation via Profiles

In this section we present the profile-based complementation algorithm introduced in [12], which is an alternative presentation of the slice-based complementation construction [19]. This construction uses a notion of *profiles* for the run DAGs, defined first over branches and then over vertices.

3.1 Profiles

Fix a run DAG $\mathcal{G}_w = \langle V, E \rangle$ of an NBW $\mathcal{A} = (Q, I, \delta, F)$ over a word $w \in \Sigma^\omega$. Let $\mathbf{f}_F : V \to \{0, 1\}$ be such that $\mathbf{f}_F(\langle q, i \rangle) = 1$ if $q \in F$ and $\mathbf{f}_F(\langle q, i \rangle) = 0$ otherwise. Thus, $\mathbf{f}_F$ labels F-vertices by 1 and all other vertices by 0. We first define the *profile* of a branch in $\mathcal{G}_w$ as the sequence of labels of vertices in the branch. For a finite branch $b = v_0 v_1 \ldots v_n$ in $\mathcal{G}_w$, define the profile of b, written as h_b, to be $\mathbf{f}_F(v_0)\mathbf{f}_F(v_1) \cdots \mathbf{f}_F(v_n)$. For an ω-branch $b = v_0 v_1 \ldots$, its profile is $h_b = \mathbf{f}_F(v_0)\mathbf{f}_F(v_1) \cdots$. We next define the profile of a vertex in $\mathcal{G}_w$ to be the lexicographically *maximal* profile of all branches that end in that vertex. Formally, let $\leq$ be the lexicographic ordering on $\{0, 1\}^* \cup \{0, 1\}^\omega$ such that $h_b < h_{b'}$ if there is a prefix $\alpha 0$ of h_b and $\alpha 1$ is a prefix of $h_{b'}$. We then say that h_b is lexicographically *smaller* than $h_{b'}$. Finally, the profile of a vertex v, written as h_v, is the lexicographically maximal element of $\{ h_b \mid b \text{ is a branch to } v \}$.

The lexicographic order of profiles induces a preorder over the vertices/states on the same level. The sequence of preorders $\preceq_i$ over the vertices/states on level $i \geq 0$ of $\mathcal{G}_w$ are defined as follows.

Definition 1 (Preorder $\preceq_i$). *For every two vertices u and v on level $i \geq 0$ in $\mathcal{G}_w$, we have that $u \prec_i v$ if $h_u < h_v$, $u \preceq_i v$ if $h_u \leq h_v$ and $u \approx_i v$ if $h_u = h_v$.*

By abuse of terminology, we can conflate vertices on level i of $\mathcal{G}_w$ with their underlying states and say $q \preceq_i r$ when $\langle q, i \rangle \preceq_i \langle r, i \rangle$.

One can verify that every two vertices/states on level i are comparable under $\preceq_i$ by definition. We also use the preorder $\preceq_i$ over states in Sects. 3.2 and 5.2. As $\preceq_i$ is transitive, $\approx_i$ is an equivalence relation.

For a vertex v on level i, its equivalence class under $\approx_i$ is denoted as $[v]_{\approx_i}$, or simply $[v]$ when it is clear from the context. Since the last element of a vertex's profile is 1 iff the vertex is an F-vertex, all vertices in an equivalence class of $\approx_i$ must agree on membership in F. We call an equivalence class an F-class when all its members are F-vertices, and a non-F-class when none of its members is an F-vertex.

We now use profiles in order to remove from $\mathcal{G}_w$ edges that are not on lexicographically maximal branches. Let $\mathcal{G}'_w = \langle V, E' \rangle$ be the subgraph of $\mathcal{G}_w$ induced by removing all edges (u, v) where there is an edge (u', v) such that $u \prec_{|u|} u'$.

Intuitively, we only keep the ω-branch with the maximal profile among the ω-branches that join together in the pruned run DAG $\mathcal{G}'_w$. Observe that the removal of such edges does not change the profiles of vertices, and further that vertices derive their profiles from their parents in $\mathcal{G}'_w$, as formalized here:

Lemma 1 ([12]). *For every two vertices u and v in $\mathcal{G}'_w$ (and hence $\mathcal{G}_w$), if $(u, v) \in E'$, then $h_v \in \{h_u 0, h_u 1\}$.*

While it is possible for two vertices with different profiles to share a child in $\mathcal{G}_w$, Lemma 1 precludes this possibility in $\mathcal{G}'_w$. If two vertices join in $\mathcal{G}'_w$, they must have the same profile and be in the same equivalence class. We can thus conflate vertices and equivalence classes, and for every edge $(u, v) \in E'$, consider the equivalence class $[v]$ of v to be the child of the equivalence class $[u]$ of u. Lemma 1 then entails that the class $[u]$ can have at most two children: the class of F-vertices with profile $h_u 1$, and the class of non-F-vertices with profile $h_u 0$. We call the first class the F-child, and the second class the non-F-child of $[u]$.

By using lexicographic ordering we can derive the preorder for level $i + 1$ of $\mathcal{G}_w$ solely from the preorder for the previous level i. To determine the ordering relation between two vertices, we need only know the relation between the parents of those vertices, and whether the vertices are F-vertices. Formally:

Lemma 2 ([12]). *For vertices u, v on level i, and vertices u', v' where $(u, u') \in E'$ and $(v, v') \in E'$ hold in $\mathcal{G}'_w$, we have:*

- *If $u \prec_i v$, then $u' \prec_{i+1} v'$.*
- *If $u \approx_i v$ and either both u' and v' are F-vertices, or neither are F-vertices, then $u' \approx_{i+1} v'$.*
- *If $u \approx_i v$ and v' is an F-vertex while u' is not, then $u' \prec_{i+1} v'$.*

We guarantee the correctness of the pruning on $\mathcal{G}_w$ with Lemma 3: an accepting ω-branch can only be pruned when it joins with another accepting ω-branch. Therefore we still capture an accepting ω-branch in $\mathcal{G}'_w$ for an accepting run DAG $\mathcal{G}_w$, as justified by Lemma 3.

Lemma 3 ([12]). *$\mathcal{G}'_w$ has an accepting branch iff $\mathcal{G}_w$ has an accepting branch.*

As a further step of pruning, we remove from $\mathcal{G}'_w$ all finite vertices. Let $\mathcal{G}''_w = \mathcal{G}'_w \setminus \{v \mid v \text{ is finite in } \mathcal{G}'_w\}$. Note there may be vertices that are not finite in $\mathcal{G}_w$ but are finite in $\mathcal{G}'_w$. An important observation is that $\mathcal{G}_w$ may have infinitely many F-vertices, and still not contain an accepting branch: there may be infinitely many branches each with a finite number of F-vertices. Lemma 4 demonstrates that the transition from $\mathcal{G}_w$ via $\mathcal{G}'_w$ to $\mathcal{G}''_w$ removes this possibility, and the presence of infinitely many F-vertices in $\mathcal{G}''_w$ does imply the existence of an accepting branch.

Lemma 4 ([12]). *$\mathcal{G}_w$ has an accepting ω-branch iff $\mathcal{G}''_w$ has infinitely many F-vertices.*

The property of $\mathcal{G}''_w$ in Lemma 4 is vital for the profile-based complementation construction, as shown in Sect. 3.2.

3.2 Complementing with Profiles

We now complement the Büchi automaton $\mathcal{A} = (Q, I, \delta, F)$ by constructing an NBW $\mathcal{B}_p$ that employs Lemma 4 to determine if an ω-word w is in $L(\mathcal{A})$. The NBW $\mathcal{B}_p$ constructs $\mathcal{G}'_w$ level by level, while guessing which vertices are finite in $\mathcal{G}'_w$ and therefore not present in $\mathcal{G}''_w$. To build $\mathcal{G}'_w$, $\mathcal{B}_p$ encodes each level as a set S of states that occurs on the level. Every such set S is labeled with a guess of which vertices are finite and which are infinite. States that are guessed to be infinite, which are thus kept in $\mathcal{G}''_w$, are labeled $\top$. States that are guessed to be finite, which are thus omitted from $\mathcal{G}''_w$, are labeled $\bot$. In order to track the edges of $\mathcal{G}'_w$, $\mathcal{B}_p$ needs to know the lexicographic order of vertices. Thus $\mathcal{B}_p$ also maintains the preorder $\preceq_i$ as in Definition 1 over states on the corresponding level of $\mathcal{G}'_w$. To verify whether states labeled $\bot$ are indeed finite, $\mathcal{B}_p$ utilizes the cut-point construction of Miyano and Hayashi [28], keeping an "obligation set" of states currently being verified as finite. Finally, to ensure that w is rejected by $\mathcal{A}$, $\mathcal{B}_p$ enforces that there are finitely many F-vertices in $\mathcal{G}''_w$, using a bit b to guess the level from which no more F-vertices appear in $\mathcal{G}''_w$. From this point on, it enforces that all F-vertices are labeled $\bot$.

Before we define $\mathcal{B}_p$, we formalize *preordered subsets* in Definition 1 and operations over them. For a set Q of states, we define $\mathbf{Q} = \{ \langle S, \preceq \rangle \mid S \subseteq Q \}$ to be the set of preordered subsets of Q where $\preceq$ is a preorder over S. Note that we write $\preceq_i$ as $\preceq$ here since the level number can be omitted. Let $\langle S, \preceq \rangle$ be an element in $\mathbf{Q}$. When considering the successors of a state, we only consider edges that remain in $\mathcal{G}'_w$. Formally, for every state $q \in S$ and $\sigma \in \Sigma$, define $\rho_{\langle S, \preceq \rangle}(q, \sigma) = \{ r \in \delta(q, \sigma) \mid \text{for every } q' \in S, \text{if } r \in \delta(q', \sigma) \text{ then } q' \preceq q \}$. Intuitively, $\rho_{\langle S, \preceq \rangle}(q, \sigma)$ defines the set of states at the next level in $\mathcal{G}'_w$ that can be reached by q.

Definition 2. *We define the* σ-successor of $\langle S, \preceq \rangle$ *as the tuple* $\langle \delta(S, \sigma), \preceq' \rangle$*, denoted as* $\langle \delta(S, \sigma), \preceq' \rangle = \mathcal{T}(\langle S, \preceq \rangle, \sigma)$*, where for every* $q, r \in S$*,* $q' \in \rho_{\langle S, \preceq \rangle}(q, \sigma)$*, and* $r' \in \rho_{\langle S, \preceq \rangle}(r, \sigma)$*:*

- *If $q \prec r$, then $q' \prec' r'$.*
- *If $q \approx r$ and either both $r' \in F$ and $q' \in F$, or both $r' \notin F$ and $q' \notin F$, then $q' \approx' r'$.*
- *If $q \approx r$ and one of q' and r', say r', is in F while the other, q', is not, then $q' \prec' r'$.*

One can see that the preorder $\preceq'$ over $\delta(S, \sigma)$ is exactly the one defined in Lemma 2 excluding the level number $i + 1$.

We now define $\mathcal{B}_p$. The states of $\mathcal{B}_p$ are tuples $\langle S, \preceq, \lambda, O, b \rangle$ where: $\langle S, \preceq \rangle \in \mathbf{Q}$ is preordered subset of Q; $\lambda : S \to \{\top, \bot\}$ is a labeling indicating which states are guessed to be finite ($\bot$) or infinite ($\top$), $O \subseteq S$ is the obligation set, and $b \in \{0,1\}$ is a bit indicating whether we have seen the last F-vertex in $\mathcal{G}''_w$. To transition between states of $\mathcal{B}_p$, say that $\boldsymbol{t'} = \langle S', \preceq', \lambda', O', b' \rangle$ *follows* $\boldsymbol{t} = \langle S, \preceq, \lambda, O, b \rangle$ *under* σ when:

1. $\langle S', \preceq' \rangle$ is the σ-successor of $\langle S, \preceq \rangle$, i.e., $\langle S', \preceq' \rangle = \mathcal{T}(\langle S, \preceq \rangle, \sigma)$.
2. λ' is such that for every $q \in S$:
   (a) If $\lambda(q) = \top$, then there exists $r \in \rho_{\langle S, \preceq \rangle}(q, \sigma)$ such that $\lambda'(r) = \top$,
   (b) If $\lambda(q) = \bot$, then for every $r \in \rho_{\langle S, \preceq \rangle}(q, \sigma)$, it holds that $\lambda'(r) = \bot$.
3. $O' = \begin{cases} \bigcup_{q \in O} \rho_{\langle S, \preceq \rangle}(q, \sigma) & O \neq \emptyset, \\ \{q \mid q \in S' \text{ and } \lambda'(q) = \bot\} & O = \emptyset. \end{cases}$
4. $b' \geq b$.

We can see that once b has been set to 1, then b will be 1 forever from this $\mathcal{B}_p$-state; that is, we guess that all its runs of $\mathcal{B}_p$ from this state reach a suffix where all F-vertices are labeled finite. Thus we also need to exploit a labeling λ that reflects this guess. To this end, given a $\mathcal{B}_p$-state $\langle S, \preceq, \lambda, O, b \rangle$, we say that λ is *F-free* if for every $q \in S \cap F$ we have $\lambda(q) = \bot$. We can now define the complementation construction, and state its theorem of correctness and complexity.

Definition 3 ([12]). *For an NBW $\mathcal{A} = \langle Q, I, \delta, F \rangle$, let $\mathcal{B}_p$ be the constructed NBW $\langle Q_p, I_p, \delta_p, F_p \rangle$ where:*

- $Q_p = \{\, \langle S, \preceq, \lambda, O, b \rangle \mid$ *if* $b = 1$ *then* λ *is* F*-free*, $S \subseteq Q \,\}$,
- $I_p = \{\, \langle I, \preceq, \lambda, \emptyset, 0 \rangle \mid$ *for all* $q, r \in I$, $q \preceq r$ *iff* $q \notin F$ *or* $r \in F \,\}$,
- $\delta_p(\boldsymbol{t}, \sigma) = \{\, \boldsymbol{t'} \mid \boldsymbol{t'}$ *follows* $\boldsymbol{t}$ *under* $\sigma \,\}$, *and*
- $F_p = \{\, \langle S, \preceq, \lambda, \emptyset, 1 \rangle \mid S \subseteq Q \,\}$.

Theorem 1 ([12]). *Let $\mathcal{A}$ be an NBW with n states. Then we have that $\mathcal{L}(\mathcal{B}_p) = \overline{\mathcal{L}(\mathcal{A})}$ and $\mathcal{B}_p$ has at most $(2n)^n$ states.*

The correctness proof of Theorem 1 connects runs of $\mathcal{B}_p$ with $\mathcal{G}'_w$. For a more precise bound on the size, we note that if $n = |Q|$, the number of preordered subsets is roughly $(0.53n)^n$ [12]. As there are 2^n labellings, and a further 2^n obligation sets, the state space of $\mathcal{B}_p$ is at most $(2n)^n$. The slice-based automaton obtained in [19] coincides with $\mathcal{B}_p$, modulo the details of labeling states and the cut-point construction. The correctness proof in [19], however, is given by means of reduced split trees, whereas here we operate directly on the run DAG.

4 Complementation via Ranks

In this section, we present the rank-based complementation construction introduced in [23], and show how profiles can be used as a tool for this construction.

4.1 Rank-Based Complementation

Unlike the profile-based algorithm, the original rank-based complementation proposed by Kupferman and Vardi [23] constructs the complementary NBW by tracking the nonaccepting run DAGs, without pruning.

As shown in [38], checking whether a run DAG $\mathcal{G}_w$ is nonaccepting can be reduced to verifying fair termination; the core idea is that $\mathcal{G}_w$ can be viewed as a fair transition system [39] that fairly terminates if none of its runs satisfies the fair condition. Let $\mathcal{G}_w = \langle V, E \rangle$. The corresponding fair transition system is $M_w = (V, I \times \{0\}, E, F \times \mathbb{N})$ where $F \times \mathbb{N}$ is the fair condition. Clearly, $\mathcal{G}_w$ is nonaccepting iff M_w fairly terminates, i.e., all ω-branches in $\mathcal{G}_w$ visit only finitely many F-vertices.

To identify nonaccepting run DAGs, we use the model checking algorithm called One-Way-Catch-Them-Young (henceforth OWCTY) [10], an improvement of the Emerson-Lei algorithm (henceforth EL) [9]. A run DAG $\mathcal{G}_w$ can be seen as a fair transition system $M = (W, W_0, R, F) = M_w$, the input of OWCTY. Let $X, Y \subseteq W$ be two sets of states. We denote by $\mathit{next}(X)$ the states who have successors in X, i.e., for each state $x \in \mathit{next}(X)$, there is a state $y \in X$ such that $(x, y) \in R$. We represent $\mathit{until}(X, Y)$ as the set of states in X that can properly reach Y while still staying in X. That is, for each state $x \in \mathit{until}(X, Y)$, there is a sequence $x_0, \cdots, x_k, k > 0$, where $x_k \in Y$, $x_i \in X$ and $(x_i, x_{i+1}) \in R$ for $0 \leq i < k$. We give the OWCTY version presented in [39] as follows:

```
$Q \leftarrow W$
repeat
    repeat
        $Q \leftarrow Q \cap \mathit{next}(Q)$
    until $Q$ not changed
    $Q \leftarrow Q \cap \mathit{until}(Q, Q \cap F)$
until $Q$ not changed
return $(W_0 \cap Q = \emptyset)$
```

Intuitively, the inner loop deletes the states that have only finitely many successors; such states surely cannot lie on a fair infinite trace (i.e., trace with infinitely many F-states). The outer loop removes all states that cannot reach accepting states in F. It is shown in [23] that the outer loop of OWCTY always converges in at most n iterations when applied to fair transition systems of the form M_w for an NBW $\mathcal{A}$ with n states. Intuitively, each level of $\mathcal{G}_w$ has at most n vertices, and each iteration of the outer loop either halts or removes an infinite trace with finitely many accepting states. This allows us to assign finite ranks to the vertices of $\mathcal{G}_w$ as follows:

– Assign a vertex v rank $2i$ if it is deleted in the i-th iteration of the outer loop by the statement $Q \leftarrow Q \cap \mathit{next}(Q)$.
– Assign a vertex v rank $2i + 1$ if it is deleted in the i-th iteration of the outer loop by the statement $Q \leftarrow Q \cap \mathit{until}(Q, Q \cap F)$.

Intuitively, ranks measure the "progress" made by a vertex towards acceptance [21]. While, in general, transfinite ranks for a fair-transition system are required, we can use here exactly the ranks $0, \cdots, 2n-2$ in the above procedure for a run DAG of an n-state NBW $\mathcal{A}$, according to [16,23]. Thus, given a run DAG $\mathcal{G}_w$, we can obtain a function $f : V \to \{0, \ldots, 2n-2\}$ for $\mathcal{G}_w$ in terms of ranks assigned by OWCTY. This is called in [39] the *C-ranking* of $\mathcal{G}_w$.

We can see that a vertex must be removed by OWCTY no sooner than its parents in $\mathcal{G}_w$ and the F-vertices are always removed in the inner loop. Therefore the ranks in the C-ranking f along a branch does not increase and F-vertices get only even ranks. Therefore, it is easy to see that all ω-branches in $\mathcal{G}_w$ eventually get trapped in odd ranks if $\mathcal{G}_w$ is nonaccepting. We say that the C-ranking f is *odd* if all the ω-branches of $\mathcal{G}_w$ eventually get trapped in odd ranks.

Lemma 5 ([23]). *$\mathcal{A}$ rejects w iff there is an odd C-ranking for $\mathcal{G}_w$.*

The odd C-ranking provides a unique way of rank assignments for identifying nonaccepting DAGs since the described OWCTY operates deterministically on a given run DAG. When constructing a complementary NBW $\mathcal{B}_r$ of $\mathcal{A}$, it is impossible to foresee the precise rank of C-ranking for a vertex in $\mathcal{G}_w$ and we have to guess the ranks level by level while reading the ω-word w. Recall that the maximum rank for $\mathcal{G}_w$ is $2n-2$. So along an input word w, we can encode the ranking of a level in $\mathcal{G}_w$ by utilizing a *level-ranking* function $f : Q \to [2n-2] \cup \{\bot\}$ for the states S at a level of $\mathcal{G}_w$ such that if $q \in S \cap F$, then $f(q)$ is even, and $f(q) = \bot$ if $q \in Q \setminus S$. We denote by $\mathcal{R}$ the set of all possible level-ranking functions. In order to define a valid ranking for the next level, we define the following coverage relation for level-ranking functions.

Definition 4. *Let δ be the transition function of $\mathcal{A}$, σ a letter in Σ and f, f' two level-ranking functions. Let $\alpha(f) = \{ q \in Q \mid f(q) \neq \bot \}$. We say f* covers f' under *letter σ, denoted by $f' \leq^{\delta}_{\sigma} f$, if $q' \in \delta(\alpha(f), \sigma)$, we have $0 \leq f'(q') \leq f(q)$ for every $q \in \alpha(f)$ such that $q' \in \delta(q, \sigma)$, otherwise $f'(q') = \bot$ if $q' \notin \delta(\alpha(f), \sigma)$.*

The coverage relation indicates that the level-rankings f and f' of two consecutive levels of $\mathcal{G}_w$ do not increase in ranks on a branch, as aforementioned.

In order to verify whether the guess about the ranking of $\mathcal{G}_w$ is correct, rank-based complementation also uses the *cut-point construction* in [28]. This construction employs a set of states $O \subseteq Q$ to check that the vertices assigned with even ranks are finite. That is, the ranking of a nonaccepting run DAG $\mathcal{G}_w$ eventually gets trapped in odd ranks. The formal definition of $\mathcal{B}_r$ is given below.

Definition 5 ([23]). *Let $\mathcal{A} = (Q, I, \delta, F)$ be an NBW. We define an NBW $\mathcal{B}_r = (Q_r, I_r, \delta_r, F_r)$ as follows.*

- *$Q_r \subseteq \mathcal{R} \times 2^Q$,*
- *$I_r = (f, \emptyset)$ where $f(q) = 2n-2$ if $q \in I$ and $f(q) = \bot$ otherwise.*
- *δ_r is defined as follows:*
  1. *if $O \neq \emptyset$, then $\delta_r((f, O), \sigma) = \{ (f', \delta(O, \sigma) \setminus odd(f')) \mid f' \leq^{\delta}_{\sigma} f \}$*

   2. *if $O = \emptyset$, then $\delta_r((f, O), \sigma) = \{ (f', even(f')) \mid f' \leq^{\delta}_{\sigma} f \}$*
 – $F_r = \{ (f, O) \in Q_r \mid O = \emptyset \}$.

where $odd(f) = \{ q \in Q \mid f(q)$ is odd $\}$ and $even(f) = \{ q \in Q \mid f(q)$ is even $\}$.

Here in (f, O), f is the guessed level-ranking for the current level and O contains vertices/states along the branches that have not visited a vertex with an odd rank since the last time O has been empty. Let w be an ω-word. Intuitively, every state (f, O) in $\mathcal{B}_r$ corresponds to a level of the DAG $\mathcal{G}_w$ over w. If w is accepted by $\mathcal{B}_r$, i.e., O becomes empty for infinitely many times, then we conclude that all the ω-branches of $\mathcal{G}_w$ eventually get trapped in odd ranks. It follows that no branches are accepting in $\mathcal{G}_w$, i.e., $w \notin \mathcal{L}(\mathcal{A})$. The other direction is also easy to prove: since we have guessed all possible rankings of $\mathcal{G}_w$, if $\mathcal{G}_w$ is nonaccepting, then at least one guess will be an odd C-ranking. Clearly, $\mathcal{B}_r$ will accept w since odd C-ranking will induce infinitely many $\mathcal{B}_r$-states with empty O. Thus we conclude that $\mathcal{L}(\mathcal{B}_r) = \Sigma^\omega \setminus \mathcal{L}(\mathcal{A})$. Since $f \in \mathcal{R}$ is a function from Q to $[2n-2] \cup \{\perp\}$, the number of possible f functions is at most $(2n)^n$. Therefore, the number of states in $\mathcal{A}$ is at most $2^n \times (2n)^n$.

Theorem 2 ([23]). *Let $\mathcal{A}$ be an NBW with n states and $\mathcal{B}_r$ the NBW defined in Definition 5. Then $\mathcal{L}(\mathcal{B}_r) = \overline{\mathcal{L}(\mathcal{A})}$ and $\mathcal{B}_r$ has $\mathcal{O}((4n)^n)$ states.*

With the optimizations proposed in [15,33], the number of states in $\mathcal{B}_r$ can be improved to $\mathcal{O}((0.76n)^n)$, matching the lower bound given in [41].

4.2 Connection between Ranks and Profiles

In this subsection, we introduce a connection between the rank-based construction and the profile-based construction by defining a *retrospective ranking* of $\mathcal{G}_w$ based on profiles. We say a level $k \geq 1$ is a *stable level* in $\mathcal{G}'_w$ if all F-vertices starting from level k are finite. According to the proof of Lemma 7 in [12], we have that there is a stable level $k \geq 0$ in $\mathcal{G}'_w$ iff $\mathcal{G}'_w$ is nonaccepting. The following lemma is a reformulation of Corollary 10 in [12], adapted to our notations.

Lemma 6 (Adapted from [12]). *$\mathcal{A}$ does not accept w iff there exists a stable level $k \geq 0$ in $\mathcal{G}'_w$.*

Recall that we use the labeling function λ in Sect. 3.2 to indicate whether a vertex in $\mathcal{G}'_w$ is finite ($\perp$) or infinite ($\top$) for constructing $\mathcal{B}_p$; the function λ has to guess the labeling of every vertex in $\mathcal{G}'_w$ when constructing $\mathcal{B}_p$. With the existence of the stable level k for a nonaccepting $\mathcal{G}'_w$, we do not have to guess for each vertex but just guess the stable level of $\mathcal{G}'_w$. That is, we first guess a stable level k of $\mathcal{G}'_w$ and before the stable level k, every vertex is labelled with $\top$; after the stable level, all F-vertices and their descendants are labeled with $\perp$. We denote this labeling by λ^k since it is dependent on the stable level k. Let S_i be the set of vertices on level i of $\mathcal{G}_w$. Formally, for $i \geq 0$, the labeling function $\lambda^k : S_i \to \{\top, \perp\}$ is defined as follows:

– If $i \leq k$, then for every $u \in S_i$ we define $\lambda^k(u) = \top$.
– If $i > k$, then for every $u \in S_i$:
  - If u is an F-vertex, then $\lambda^k(u) = \bot$.
  - Otherwise, $\lambda^k(u) = \lambda^k(v)$, for a vertex v where $E'(v, u)$.

We note that λ^k is well defined when $i > k$ and u is not an F-vertex, since by Lemma 1, all its parents in $\mathcal{G}'$ belong to the same equivalence class; so $\lambda^k(u)$ does not depend on the choice of the vertex v where $E'(v, u)$; see [12] for detailed reasoning. The correctness of this labeling is justified by Lemma 6.

As a byproduct, the constructed NBW has nondeterminism only in the transition to stable level k, and is deterministic in the limit. Note that the labeling λ^k is defined on the edges of $\mathcal{G}'_w$ rather than on edges of $\mathcal{G}_w$ (see $E'(v, u)$ in the definition above). We say a labeling λ^k is *legal* if it correctly labels the finite/infinite vertices in the run DAG. Based on the definition of λ^k, Corollary 1 is a direct consequence of Lemma 6.

Corollary 1. *$\mathcal{G}_w$ is rejecting iff for some k, the labeling λ^k is legal.*

Now we are ready to derive an odd ranking for $\mathcal{G}_w$ from the function λ^k, thus relating the profile-based analysis behind λ^k with the rank-based analysis of [23]. We have defined in Sect. 4.1 the C-ranking for $\mathcal{G}_w$ such that the C-ranking is odd iff $\mathcal{G}_w$ is nonaccepting. In order to define such a ranking function for identifying nonaccepting run DAGs, we introduce below the so-called *retrospective ranking* for $\mathcal{G}_w$ [12]. Unlike C-ranking, which predicts the progress towards the acceptance in future, retrospective ranking is defined based on the past profiles we have seen. For the retrospective ranking to be odd for a nonaccepting run DAG $\mathcal{G}_w$, we only need to care about the ranks of vertices after stable level k in which all ω-branches get trapped in odd ranks.

Consider again $\mathcal{G}'_w$ and the labeling λ^k. We know that after the stable level k, if λ^k is legal, an ω-branch with only label $\top$ is nonaccepting. So, the retrospective ranking for $\mathcal{G}_w$ after level k, gives odd ranks to $\top$-labeled vertices and even ranks to $\bot$-labeled classes. Here the ranks increase in inverse lexicographic order, i.e., we define the rank of the maximal $\top$-labeled class as 1. A good property of this ranking is that we do not need to distinguish between two adjacent $\bot$-labeled classes. Formally, we have the following k-retrospective ranking for $\mathcal{G}_w$.

Definition 6. *Consider a run DAG $\mathcal{G}_w$, $k \in \mathbb{N}$, and a labeling $\lambda^k : \mathcal{G}_w \to \{\top, \bot\}$. Let $m = 2|Q \setminus F|$. For a vertex u on level i of $\mathcal{G}_w$, let $\alpha(u)$ be the number of $\top$-labeled classes larger than u; $\alpha(u) = |\{\, [v] \mid \lambda^k(v) = \top \text{ and } u \prec_i v \,\}|$. The* k-retrospective ranking *of $\mathcal{G}'_w$ is the function $\mathbf{r}^k : V \to \{0..m\}$ defined for every vertex u on level i as follows.*

$$\mathbf{r}^k(u) = \begin{cases} m & \text{if } i \leq k, \\ 2\alpha(u) & \text{if } i > k \text{ and } \lambda^k(u) = \bot, \\ 2\alpha(u) + 1 & \text{if } i > k \text{ and } \lambda^k(u) = \top. \end{cases}$$

According to Lemma 1, every equivalence class $[u]$ has at most two child equivalence classes, one F-class and one non-F-class. So, on the same level, each

$\top$-labeled class is given the odd rank greater by two than the rank of the next lexicographically larger $\top$-labeled class. The number of $\top$-labeled classes on each level is at most $\frac{m}{2} = |Q \setminus F|$. Moreover, since the number of larger $\top$-labeled classes does not increase, the ranks of child equivalence classes on the next level are no larger than the rank of their parent. Formally, we have the following.

Lemma 7 ([12]). *For a level $j \geq k$, we have*

- *If $u \prec_j u'$ then $\mathbf{r}^k(u) \geq \mathbf{r}^k(u')$, where u and u' are two vertices on level j.*
- *If $(u, v) \in E'$, then $\mathbf{r}^k(u) \geq \mathbf{r}^k(v)$, where u and v are two vertices on level j and $j + 1$, respectively.*

Thus, the following lemma holds if λ^k is legal and $\mathcal{G}_w$ is nonaccepting, according to Lemma 15 of [12].

Lemma 8 ([12]). *$\mathcal{A}$ does not accept w iff the retrospective ranking $\mathbf{r}^k$ bounded by m for $\mathcal{G}_w$ is an odd ranking.*

4.3 Complementing with Retrospective Rankings

We have introduced in Section 4.2 about how to determine whether w is accepted by $\mathcal{A}$ by defining the retrospective rankings based on the profiles of branches after the stable level in $\mathcal{G}_w$ (cf. Lemma 7). Now we are ready to define the complementary NBW $\mathcal{A}_L$ based on the retrospective ranking.

First, we need to guess the stable level in $\mathcal{G}_w$ before we can define the retrospective ranking for checking whether the run DAG $\mathcal{G}_w$ is accepting. In order to correctly identify the stable level k for a given run DAG $\mathcal{G}_w$, we partition the construction of $\mathcal{A}_L$ into two phases, namely *initial phase* and *ranking phase*. In the initial phase, the NBW $\mathcal{A}_L$ deterministically tracks all preordered subsets and we assume all levels in this phase are before the stable level k. Once $\mathcal{A}_L$ nondeterministically moves to the ranking phase, we assume that we have reached the stable level k and need to deterministically track the ranks in order to check whether $\mathcal{G}_w$ is accepting. Therefore, the only nondeterminism in $\mathcal{A}_L$ lies in the guess of k, i.e., the transition between the initial phase and the ranking phase.

Recall that we denote by $\mathbf{Q}$ the set of preordered subsets of Q in Sect. 3.2. Let $\mathcal{R}^m$ be the set of level rankings bounded by m. There are three types of transitions in $\mathcal{A}_L$: transitions in the initial phase, transitions from the initial phase to the ranking phase, and transitions within the ranking phase. The first type of transitions is the σ-successor relation between preordered subsets for $\mathcal{B}_p$, as described in Sect. 3.2. We only give the other two types of transitions below.

Recall that the rank of a vertex u only depends on the number of $\top$-labeled classes larger than it, denoted $\alpha(u)$. The transitions moving between phases are transitions from a preordered set $\langle S, \preceq \rangle$ to the level ranking of its σ-successor, as defined below. For each $q \in S$, let $\beta(q) = |\{[p] \mid p \in S \setminus F, q \prec p\}|$ be the number of non-F-classes larger than q (On level $k + 1$, a vertex is labeled with $\top$ iff it is an non-F-vertex and $\beta(q)$ is easy to compute). We define a function

$\texttt{torank} : \mathbf{Q} \to \mathcal{R}^m$ that obtains from a preordered set $\langle S, \preceq \rangle$ a level-ranking function $f \in \mathcal{R}^m$ such that for each $q \in Q$:

$$f(q) = \begin{cases} \bot & \text{if } q \notin S, \\ 2\beta(q) & \text{if } q \in S \cap F, \\ 2\beta(q) + 1 & \text{if } q \in S \setminus F. \end{cases}$$

Now we define the transitions between the level rankings of $\mathcal{R}^m$. For a level ranking $f \in \mathcal{R}^m$, $\sigma \in \Sigma$, and $q' \in Q$, let $\texttt{pred}(q', \sigma, f) = \{\, q \mid f(q) \neq \bot,\ q' \in \rho(q, \sigma) \,\}$ be the predecessors of q' with non-$\bot$ rank. One can see that the predecessor in $\texttt{pred}(q', \sigma, f)$ with the lowest rank has the maximal profile in $\mathcal{G}$ among $\texttt{pred}(q', \sigma, f)$. For $h \in \mathbb{N}$, let $\lfloor h \rfloor_{even}$ be h if h is even and be $h - 1$ if h is odd. Now we define the *σ-successor of f* to be f' where for each $q' \in Q$:

$$f'(q') = \begin{cases} \bot & \text{if } \texttt{pred}(q', \sigma, f) = \emptyset, \\ \lfloor \texttt{min}(\{f(q) \mid q \in \texttt{pred}(q', \sigma, f)\}) \rfloor_{even} & \text{if } \texttt{pred}(q', \sigma, f) \neq \emptyset \text{ and } q' \in F, \\ \texttt{min}(\{\, f(q) \mid q \in \texttt{pred}(q', \sigma, f) \,\}) & \text{if } \texttt{pred}(q', \sigma, f) \neq \emptyset \text{ and } q' \notin F. \end{cases}$$

Here the rank of a vertex v is computed based on the rank of its predecessor u with maximal profile in $\mathcal{G}'_w$. Note, however, that λ^k may label a finite class $\top$; in such a case, a $\top$-labeled class larger than u has no children, thus $f(u) > f'(v)$. Hence we know that λ^k now is illegal, so an optimization is that we can just ignore the computation of following successors; this optimization is omitted in the construction for clarity. We formalize the construction of $\mathcal{A}_L$ as below.

Definition 7. *For an NBW $\mathcal{A} = \langle Q, I, \delta, F \rangle$, let $\mathcal{A}_L$ be the NBW $\langle \mathbf{Q} \cup (\mathcal{R}^m \times 2^Q), I_L, \delta_L, \mathcal{R}^m \times \{\emptyset\} \rangle$, where*

- *$I_L = \{\langle I, \preceq_0 \rangle\}$, where $\preceq_0$ is such that for all $q, r \in I$, $q \preceq_0 r$ iff $q \notin F$ or $r \in F$.*
- *$\delta_L(\mathcal{S}, \sigma) = \{\mathcal{S}'\} \cup \{\langle \texttt{torank}(\mathcal{S}'), \emptyset \rangle\}$, where $\mathcal{S}'$ is the σ-successor of $\mathcal{S}$.*
- *$\delta_L(\langle f, O \rangle, \sigma) = \{\langle f', O' \rangle\}$ where f' is the σ-successor of f*
  $$\text{and } O' = \begin{cases} \rho(O, \sigma) \setminus odd(f') & \text{if } O \neq \emptyset, \\ even(f') & \text{if } O = \emptyset. \end{cases}$$

Theorem 3 ([12]). *For an NBW $\mathcal{A}$ with n states, we have that $L(\mathcal{A}_L) = \overline{L(\mathcal{A})}$ and $\mathcal{A}_L$ has $\mathcal{O}((8n)^n)$ states.*

The maximum rank of a state is $m = 2|Q \setminus F| \leq 2n$, so the number of states in the ranking phase is at most $(2n + 1)^n \times 2^n \leq (4n + 2)^n$. The number of states in the initial phase is at most $\mathcal{O}((0.53n)^n)$ [12]; so, in total $\mathcal{A}_L$ has at most $(4n + 2)^n + \mathcal{O}((0.53n)^n) \in \mathcal{O}((8n)^n)$ states. We remark that the level rankings can be further tightened, as done in [12], which, however, is not the goal of this section.

5 Complementation via Congruence Relations

In this section, we first give a general framework of complementation constructions from a language-theoretic perspective. We then recall the classical congruence relations defined in [34] and give optimal congruence relations based on profiles [24]. Büchi [6] proposed a construction based on congruence relations for complementing NBWs, involving a Ramsey-based analysis, which is why it is widely known as the Ramsey-based complementation (RBC). His argument was refined in [34], where the construction of congruence relation was optimized. Later, Thomas showed that the Ramsey argument was not necessary [35].

The language-theoretic perspective to [6,34] offers the observation that every ω-regular language (including the universe of all ω-words) is a finite union of languages of the form UV^{ω}, where U and V (the latter does not contain the empty word) are regular languages. One way to construct the complementary language of an NBW $\mathcal{A}$, is to find a finite collection of languages $U_i V_i^{\omega}$ such that (i) the union of the finite collection covers Σ^{ω}, the universe of all ω-words, while (ii) the union of a subset of the finite collection equals $\mathcal{L}(\mathcal{A})$ and the union of the rest of the collection equals the complement of $\mathcal{L}(\mathcal{A})$. We refer to these properties below as Properties (i) and (ii).

When dealing with words from UV^{ω}, we can consider only the *ultimately periodic* (UP)-words of the form uv^{ω}, where $u \in U$ and $v \in V$ and totally ignore that there can be words that are not ultimately periodic [6,34,7]. We denote by $\mathrm{UP}(L)$ the set of UP-words of L, i.e., $\{\, uv^{\omega} \in L \mid u \in \Sigma^{*}, v \in \Sigma^{+} \,\}$. With the following theorem, $\mathrm{UP}(L)$ can be seen as the "fingerprint" of L.

Theorem 4 ([7]). *(1) Every non-empty ω-regular language L contains at least one UP-word. (2) Let L, L' be two ω-regular languages. Then, $L = L'$ if and only if $UP(L) = UP(L')$.*

Therefore, we are concerned only with UP-words in the remainder of this paper. In the following, we present a general framework for obtaining the complementary language of a given ω-language L.

To find a finite union of languages for constructing the complement of L, the approach of [6,34] is to design first a finite partition of Σ^{+} satisfying Property (i). Assume that $\mathcal{P} = \{U_0, \cdots, U_{k-1}\}$, for $k \geq 1$, is a finite *partition* of Σ^{+}, so $\Sigma^{+} = \bigcup_{0 \leq i < k} U_i$ and $U_i \cap U_j = \emptyset$ for $0 \leq i < j < k$. We call each element of $\mathcal{P}$ a *block* of $\mathcal{P}$. By Theorem 4, Lemma 9 holds since we can prove $\mathrm{UP}(\Sigma^{\omega}) = \mathrm{UP}(\bigcup_{0 \leq i,j < k} U_i U_j^{\omega})$ from the fact that $\mathcal{P}$ is a partition of Σ^{+}.

Lemma 9 (Coverage of Σ^{ω}). $\Sigma^{\omega} = \bigcup_{0 \leq i,j < k} U_i U_j^{\omega}$.

In order to obtain the complement language of L, the partition $\mathcal{P}$ has to satisfy Property (ii) as well, so we introduce the saturation property of $\mathcal{P}$. We say $\mathcal{P}$ *saturates* L if each language $U_i U_j^{\omega}, 0 \leq i, j < k$ is either inside L or outside L. To fulfill Property (ii), we can just let $\mathcal{P}$ saturate L. The following lemma holds immediately.

Lemma 10 (Saturation of L). *Let $\mathcal{P}$* saturate *L. Then for every $U_i U_j^{\omega}, 0 \leq i, j < k$, $U_i U_j^{\omega} \cap L \neq \emptyset$ implies $U_i U_j^{\omega} \subseteq L$.*

With $\mathcal{P}$ saturating L, we finally obtain the complement of L by the following theorem, which is a direct consequence of Lemmas 9 and 10.

Theorem 5 (Complementary language of L). *Let $\mathcal{P}$ saturate L. Then $\overline{L} = \bigcup\{\, U_i U_j^\omega \mid U_i, U_j \in \mathcal{P}, U_i U_j^\omega \cap L = \emptyset \,\}$.*

To obtain the partition $\mathcal{P}$ of Σ^+, we use a congruence relation $\backsim$. A *right congruence* (RC) relation $\backsim$ over Σ^* is an equivalence relation such that $x \backsim y$ implies $xv \backsim yv$ for all $v \in \Sigma^*$. A *congruence relation* $\backsim$ over Σ^* is an equivalence relation such that $x \backsim y$ implies $uxv \backsim uyv$ for every $x, y, u, v \in \Sigma^*$. We denote by $|\backsim|$ the *index* of the equivalence relation $\backsim$, i.e., the number of equivalence classes of $\backsim$. A *finite congruence relation* is a congruence relation with a finite index. We use $\Sigma^*/_\backsim$ to denote the set of equivalence classes of $\backsim$. Given $x \in \Sigma^*$, we denote by $[x]_\backsim$ the equivalence class of $\backsim$ that contains x.

A congruence relation $\backsim$ yields a finite partition $\Sigma^*/_\backsim$ of Σ^*; so we can also obtain a finite partition $\mathcal{P}$ of Σ^+. It follows that we can construct the complementary language of a given ω-language L, based on a congruence relation $\backsim$ if the partition induced by $\backsim$ satisfies Properties (i) and (ii), i.e., satisfying Lemmas 9 and 10. Below we introduce such a congruence relation.

5.1 Classical Congruence Relations

Sistla, Vardi, and Wolper [34] define the congruence relation $\backsim$ by distinguishing finite words in Σ^+ with the transition graph of $\mathcal{A}$. More precisely, given a word $u \in \Sigma^+$, for every two states $q, r \in Q$, we care about the following two questions:

- Is there a path in $\mathcal{A}$ from q to r over u? (we denote by $q \xrightarrow{u} r$ if so.)
- Is there a path in $\mathcal{A}$ from q to r over u that visits some accepting state? (we denote by $q \xRightarrow{u} r$ if so.)

Thomas [35] suggests to present the answers to these two questions as the following equivalence relation $\backsim$.

Definition 8 ([34,35]). *For all $u_1, u_2 \in \Sigma^+$, $u_1 \backsim u_2$ if and only if for all $q, r \in Q$, (1) $q \xrightarrow{u_1} r$ iff $q \xrightarrow{u_2} r$; and (2) $q \xRightarrow{u_1} r$ iff $q \xRightarrow{u_2} r$.*

Let u_1 and u_2 be two finite words such that $u_1 \backsim u_2$. We have that $xu_1y \backsim xu_2y$ holds for all $x, y \in \Sigma^*$ because independently from the states reached by $\mathcal{A}$ after reading x, by Definition 8 $\mathcal{A}$ reaches the same state by reading u_1 and u_2, hence it can reach the same state after continuing with y. Therefore, we have following result.

Lemma 11. *The equivalence relation $\backsim$ is a (right-)congruence relation.*

Since $\backsim$ is defined by reachability between states, we can map each of the n^2 pairs of states (q, r) to either both $q \xRightarrow{u} r$ and $q \xrightarrow{u} r$, or just $q \xrightarrow{u} r$ or none of them. Thus, we have $|\backsim| = |\Sigma^+/_\backsim| \leq 3^{n^2}$. That is, the number of equivalence classes induced by $\backsim$, say $k \geq 1$, is at most 3^{n^2}, as stated in the following lemma.

Lemma 12 ([34,35]). *Let $\backsim$ be as given in Definition 8. Then $|\backsim| \le 3^{n^2}$.*

Let $\mathcal{P}_\backsim = \{U_0, U_1, \cdots, U_{k-1}\}$ where U_i is $(i+1)$-th equivalence class induced by $\backsim$ in Σ^+ and k is the index of $\backsim$. By Lemma 9, for all partitions of Σ^+, including $\mathcal{P}_\backsim$, we can cover the universe of Σ^ω. Thus, we have:

Lemma 13 (Coverage of Σ^ω). $\Sigma^\omega = \bigcup_{U,V \in \mathcal{P}_\backsim} UV^\omega$.

An important property we want to have is that the finite partition $\mathcal{P}_\backsim$ saturates $\mathcal{L}(\mathcal{A})$, which indeed holds for $\backsim$.

Lemma 14 (Saturation of $\mathcal{L}(\mathcal{A})$ [34,35]). *For all $U, V \in \mathcal{P}_\backsim$, we have that $(UV^\omega) \cap \mathcal{L}(\mathcal{A}) \neq \emptyset$ implies $UV^\omega \subseteq \mathcal{L}(\mathcal{A})$.*

The intuition is that an accepting run ρ of the form $q_0 \xrightarrow{u} q_1 \xrightarrow{v} q_2 \cdots q_i \xrightarrow{v} q_{i+1} \cdots$ for $uv^\omega \in (UV^\omega) \cap \mathcal{L}(\mathcal{A})$ with $u \in U$ and $v \in V$ induces an accepting run over $u'v'^\omega$ with $u' \in U$ and $v' \in V'$ of the form $q_0 \xrightarrow{u'} q_1 \xrightarrow{v'} q_2 \cdots q_i \xrightarrow{v'} q_{i+1} \cdots$, according to Definition 8. This means that for each $U, V \in \mathcal{P}$, either $UV^\omega \subseteq \mathcal{L}(\mathcal{A})$ or $UV^\omega \subseteq \Sigma^\omega \setminus \mathcal{L}(\mathcal{A})$. Thus, we obtain the complementary language of $\mathcal{A}$:

Theorem 6 (Complementary language of $\mathcal{L}(\mathcal{A})$ [34,35]). $\Sigma^\omega \setminus \mathcal{L}(\mathcal{A}) = \bigcup\{\, UV^\omega \mid U, V \in \mathcal{P}_\backsim, UV^\omega \cap \mathcal{L}(\mathcal{A}) = \emptyset \,\}$.

Construction of complementary NBWs. For a given RC $\backsim$ of a regular language L, it is well-known that Myhill-Nerode theorem [30,31] defines a unique minimal DFW D of L, in which each state of D corresponds to an equivalence class defined by $\backsim$ over Σ^*. Therefore, we can construct a DFW $\mathcal{D}[\backsim]$ from $\backsim$ in a standard way [18]. We already have the congruence relation $\backsim$, which yields the finite collection of $U_i V_i^\omega$ that equals to $\overline{\mathcal{L}(\mathcal{A})}$ (cf. Theorem 6) where $U_i, V_i \in \mathcal{P}$. To obtain the complementary NBW of $\mathcal{A}$, we construct, for each language $U_i V_i^\omega$, its NBW $\mathcal{B}_{U_i,V_i}$ with two copies of the DFW $\mathcal{D}[\backsim]$ where the first copy accepts U_i while the second copy is modified from $\mathcal{D}[\backsim]$ to accept the words V_i^ω by adding a unique accepting state to connect the initial state and the original accepting states in $\mathcal{D}[\backsim]$. Each $\mathcal{D}[\backsim]$ has at most 3^{n^2} states, so $\mathcal{B}_{U_i,V_i}$ has at most $2 \times 3^{n^2} + 1$ states. Then the complementary NBW of $\mathcal{A}$, say $\overline{\mathcal{A}}$, can be obtained by computing the union of all such NBWs $\mathcal{B}_{U_i,V_i}$, where $U_i V_i^\omega \cap \mathcal{L}(\mathcal{A}) = \emptyset$. Moreover, the number of possible such NBWs $\mathcal{B}_{U_i,V_i}$ is $3^{n^2} \times 3^{n^2}$ according to Lemma 12. In total, the number of states in $\overline{\mathcal{A}}$ is in $2^{\mathcal{O}(n^2)}$. Therefore, we finally have the following theorem for the complementation algorithm based on congruence relation $\backsim$.

Theorem 7 ([34]). *For an NBW $\mathcal{A}$ with n states, we have that $\mathcal{L}(\overline{\mathcal{A}}) = \overline{\mathcal{L}(\mathcal{A})}$ and $\overline{\mathcal{A}}$ has at most $2^{\mathcal{O}(n^2)}$ states.*

In the remainder of the section, we show that we can obtain tighter congruence relations to obtain the complementary language $\Sigma^\omega \setminus \mathcal{L}(\mathcal{A})$ based on profiles.

5.2 Profile-Based Congruence Relations

In Theorem 5, we show that it is possible to obtain the complementary language of a given ω-regular language L from a finite partition $\mathcal{P}$ of Σ^+ that saturates and covers L. The congruence relation $\backsim$ defined in Section 5.1 implements such a partition $\mathcal{P}$ for a given ω-regular language $\mathcal{L}(\mathcal{A})$ specified with an NBW $\mathcal{A}$. The requirement of a full congruence relation is, however, too strong and contains redundant information, which may lead to unnecessary blow-up; see, e.g., [24]. In fact, the authors in [24] have shown that full congruence relations are not needed for obtaining partitions that saturate $\mathcal{L}(\mathcal{A})$, and RCs suffice. Since RC relations are coarser than full congruence relations, the complementation algorithms based on RCs can give us tighter constructions of the complementary NBWs.

In fact, the use of RCs in [24] dates back to earlier works [26,4]. In [26,4], the authors showed that one can define separate partitions for the finite prefixes U and the periodic words V for a given ω-regular language L. More precisely, one first defines a finite partition $\mathcal{P} = \{U_0, \cdots, U_{k-1}\}$ of Σ^+ for finite prefixes with an RC $\approx$. Then, for each block $U_i \in \mathcal{P}$, we define a finite partition $\mathcal{P}_i = \{V_{i,0}, \cdots, V_{i,k_i-1}\}$ of Σ^+ for the periodic words with an RC $\approx_{U_i}$. The families of partitions defined in [26,4] satisfy certain properties of saturation and coverage, which we call *family coverage* and *family saturation*, as formalized below.

(1) Family Coverage: $\Sigma^\omega = \bigcup_{U_i \in \mathcal{P}} (\bigcup_{\{ V_{i,j} \in \mathcal{P}_i \mid U_i V_{i,j} = U_i \}} U_i V_{i,j}^\omega)$.
(2) Family Saturation: For every pair $U_i \in \mathcal{P}$ and $V_{i,j} \in \mathcal{P}_i$, if $U_i V_{i,j} = U_i$, it holds that either $U_i V_{i,j}^\omega \subseteq L$ or $U_i V_{i,j}^\omega \subseteq \Sigma^\omega \setminus L$.

Similarly to Theorem 6, we have the following result.

Theorem 8 (Complementary language of L [26,4]). *Let $(\mathcal{P}, \{\mathcal{P}_i\}_{0 \leq i < k})$ be a family of partitions satisfying family coverage and family saturation. Then* $\overline{L} = \bigcup_{U_i \in \mathcal{P}} \{ U_i V_{i,j}^\omega \mid U_i \in \mathcal{P}, V_{i,j} \in \mathcal{P}_i, U_i V_{i,j} = U_i, U_i V_{i,j}^\omega \cap L = \emptyset \}$.

In this section, we introduce the RCs that define such families of partitions for an ω-regular language $\mathcal{L}(\mathcal{A})$ specified with an NBW $\mathcal{A} = (Q, I, \delta, F)$, using the framework of profiles described in Subsection 3.1. We first define the RC $\approx$ that induces the finite partition $\mathcal{P}$ for finite prefixes. To that end, we first describe a profile-based preorder $\preceq_u$ on the set $\delta(I, u)$ of states for a given finite prefix $u \in \Sigma^*$ [12,11].

Recall that when defining the congruence relation $\backsim$, we reasoned about the reachability relation between every pair of states of $\mathcal{A}$ over a finite word u (cf. Definition 8). Here, we focus on the set $\delta(I, u)$ of states reached from the initial states over a finite prefix $u \in \Sigma^*$, and we make use of the pruned run DAG as defined in Section 3. We show that one can define RCs to distinguish two finite prefixes u_1 and u_2 by the set of ordered states in their respective pruned run DAG $\mathcal{G}'_{u_1 w'}$ and $\mathcal{G}'_{u_2 w'}$, for an arbitrary $w' \in \Sigma^\omega$. We introduced in Subsection 3.1 the preorder $\preceq_i$ for the states on level i in a run DAG $\mathcal{G}'_w$ for $w \in \Sigma^\omega$ (cf. Definition 1). Since the preorder $\preceq_i$ is only dependent on the prefix $w[1 \cdots i]$, we can just describe $\preceq_i$ with respect to the finite prefix $u = w[1 \cdots i]$, denoted $\preceq_u$ as below.

Definition 9 (Preorder $\preceq_{w[1\cdots i]}$). *Let $w \in \Sigma^\omega$, $u = w[1\cdots i]$, and $q, r \in \delta(I, u)$. We have that $q \preceq_u r$ iff $q \preceq_i r$ in $\mathcal{G}'_w$. Furthermore, we define an equivalence relation $q \simeq_u r$ if $q \simeq_i r$ in $\mathcal{G}'_w$.*

Let $w \in \Sigma^\omega$, $u = w[1\cdots i]$, and $P = \delta(I, u)$. Let $[q]_{\preceq_u} = \{ r \in P \mid q \simeq_u r \}$ be the equivalence class of $q \in P$ under $\preceq_u$; We denote by $P/_{\preceq_u}$ the family of such equivalence classes. Note that every two states q, r in P, rather than in the whole set Q, are comparable under the preorder $\preceq_u$ since every two states in P are comparable under $\preceq_i$ (cf. Definition 1). We denote by $\langle P, \preceq_u \rangle$ the ordered family of equivalence classes of P under $\preceq_u$. Recall that the preorder $\preceq_i$ of $\mathcal{G}'_w$ is equivalent to $\preceq_u$ (cf. Definition 9). Therefore, the a-successor of $\langle P, \preceq_u \rangle$ can be computed with $\mathcal{T}(\langle P, \preceq_u \rangle, a)$ as given in Definition 2. It is immediate that:

Corollary 2. *Let $u \in \Sigma^*, a \in \Sigma$ and $P = \delta(I, u)$. Then $\langle \delta(I, ua), \preceq_{ua} \rangle = \mathcal{T}(\langle P, \preceq_u \rangle, a)$.*

We are now ready to define the RC $\approx$ given in [24] with our notations. Instead of considering every pair of states (q, r) of $\mathcal{A}$ to define the congruence relation $\backsim$ (cf. Definition 8), we use the finite prefix u by tracing the reachable states $\delta(I, u)$ with the preorder $\preceq_u$ to get the RC $\approx$.

Definition 10 (RC $\approx$). *For $u_1, u_2 \in \Sigma^*$, we say $u_1 \approx u_2$ iff $\langle \delta(I, u_1), \preceq_{u_1} \rangle = \langle \delta(I, u_2), \preceq_{u_2} \rangle$.*

It is easy to see that $\approx$ is indeed an RC.

Lemma 15. *The equivalence relation $\approx$ is an RC.*

Since each equivalence class $[u]_\approx$, $u \in \Sigma^+$, can be uniquely encoded as the set $\langle \delta(I, u), \preceq_u \rangle$, i.e., an *ordered* partition of Q, by [37,12] we have that the number of possible ordered partitions over Q is approximately $(0.53n)^n \leq n^n$. Thus we have the following upper bound for $\approx$.

Lemma 16 ([24]). *Let $\approx$ be the RC in Definition 10. Then $|\approx| \leq n^n$.*

We now define the RC $\approx_u$ for processing the periodic words based on the reachability between equivalence classes in $\mathcal{G}'_{uw}$, where $w \in \Sigma^\omega$. More precisely, we define reachability between equivalence classes under the preorder $\preceq_u$ (at level $|u|$) and $\preceq_{uv}$ (at level $|uv|$) in the pruned run DAG $\mathcal{G}'_{uvw}$. We say that $[q]_{\preceq_u}$ v-reaches $[r]_{\preceq_{uv}}$, denoted by $[q]_{\preceq_u} \xrightarrow{v} [r]_{\preceq_{uv}}$, if there are two vertices $\tau \in [q]_{\preceq_u} \times \{|u|\}$ and $\tau' \in [r]_{\preceq_{uv}} \times \{|uv|\})$ in $\mathcal{G}'_{uvw}$ such that $\tau \xrightarrow{v} \tau'$. We write $[q]_{\preceq_u} \xRightarrow{v} [r]_{\preceq_{uv}}$ if such path from τ to τ' also visits an F-vertex. One can see that the reachability relation between the equivalence classes does not depend on levels after $|uv|$ in $\mathcal{G}'_{uvw}$, i.e., w is not used there. So w can be an arbitrary ω-word in Σ^ω.

Definition 11 (RC $\approx_u$). *Given $u, v_1, v_2 \in \Sigma^*$, we say $v_1 \approx_u v_2$ if (1) $uv_1 \approx uv_2$, and (2) for all states $q \in P, r \in P'$, where $P = \delta(I, u)$ and $P' = \delta(I, uv_1) = \delta(I, uv_2)$, we have*

(i) $[q]_{\preceq_u} \xrightarrow{v_1} [r]_{\preceq_{uv_1}}$ holds in $\mathcal{G}'_{uv_1w}$ iff $[q]_{\preceq_u} \xrightarrow{v_2} [r]_{\preceq_{uv_2}}$ holds in $\mathcal{G}'_{uv_2w}$, and

(ii) $[q]_{\preceq_u} \overset{v_1}{\Longrightarrow} [r]_{\preceq_{uv_1}}$ *holds in* $\mathcal{G}'_{uv_1w}$ *iff* $[q]_{\preceq_u} \overset{v_2}{\Longrightarrow} [r]_{\preceq_{uv_2}}$ *holds in* $\mathcal{G}'_{uv_2w}$.

Note that under the assumption $uv_1 \approx uv_2$, we have that the ordered partitions $\langle \delta(I, uv_1), \preceq_{u_1} \rangle$ and $\langle \delta(I, uv_2), \preceq_{u_2} \rangle$ are equal by definition of $\approx$. It follows that $\delta(I, uv_1) = \delta(I, uv_2)$ also holds.

Definition 11 is designed to formalize the following idea for recognizing the ω-words accepted and rejected by $\mathcal{A}$. We want to use the RC $\approx$ for the finite prefixes and the RC $\approx_u$ for the periodic finite words of u to establish the family saturation property introduced before. That is, under the assumption that $u \approx uv_1$ and $u \approx uv_2$, we want to guarantee that if $v_1 \approx_u v_2$, then $uv_1^\omega \in \mathcal{L}(\mathcal{A})$ if and only if $uv_2^\omega \in \mathcal{L}(\mathcal{A})$. To achieve this, the first condition we impose – Item (1) of Definition 11 – is to ensure visiting infinitely often the same ordered partition under $\preceq_u$ over the ω-words uv_1^ω and uv_2^ω; so we require $uv_1 \approx uv_2$. The second condition is to guarantee that the profiles of branches in the pruned run DAG $\mathcal{G}'_{uv_1^k w}$ and $\mathcal{G}'_{uv_2^k w}$, $k \geq 1$, share visits to F-vertices; so, when extending to infinite words, their profiles either both have infinitely many 1s or neither of them does. This ensures that $uv_1^\omega \in \mathcal{L}(\mathcal{A})$ if and only if $uv_2^\omega \in \mathcal{L}(\mathcal{A})$. To guarantee that, we first require that the reachability relation between every pair of equivalence classes or blocks under $\preceq_u$ over finite words v_1 and v_2 either holds for both or neither of them (cf. condition (2)-(i)); then, we demand that they also share the visits to accepting states (cf. condition (2)-(ii)).

As stated before Definition 11, the index of $\approx_u$ is indeed in $2^{\mathcal{O}(n \log n)}$.

Lemma 17 ([24]). *Given* $u \in \Sigma^*$*, let* $\approx_u$ *be the RC from Definition 11. Then* $|{\approx_u}| \leq n^n \times (n+1)^n \times 2^n \in 2^{\mathcal{O}(n \log n)}$.

The upper bound for $|{\approx_u}|$ can be deduced from the encoding we use for $[v]_{\approx_u}$. $[v]_{\approx_u}$ is mapped to the pair $\langle \langle \delta(I, uv), \preceq_{uv} \rangle, f \rangle$ where the function f keeps track of the satisfaction of the pair of states $q, r \in Q$ of the conditions in Definition 11, i.e., whether $[q]_{\preceq_u} \overset{v}{\rightarrow} [r]_{\preceq_{uv}}$ and $[q]_{\preceq_u} \overset{v}{\Rightarrow} [r]_{\preceq_{uv}}$ in Conditions (2)-(i) and (2)-(ii) for the such states. Each equivalence class $[r]_{\preceq_{uv}}$ can only be reached by exactly one equivalence class under $\preceq_u$. There are at most n equivalence classes defined by both $\preceq_u$ and $\preceq_{uv}$. Then the codomain of f has size $2n+1 < 2(n+1)$, so the possible different functions f are $(2(n+1))^n = 2^n \times (n+1)^n$, while by [12] the possible sets $\langle \delta(I, uv), \preceq_{uv} \rangle$ are n^n, hence $|{\approx_u}| \leq n^n \times (n+1)^n \times 2^n \in 2^{\mathcal{O}(n \log n)}$.

Let $\mathcal{P} = \{U_0, \cdots, U_{k-1}\}$ be a partition of Σ^+ induced by $\approx$. For each block $U_i \in \mathcal{P}$, let $\mathcal{P}_i = \{V_{i,0}, \cdots, V_{i,k_i-1}\}$ be the partition of Σ^+ induced by $\approx_u$, where $[u]_\approx = U_i$. We first show that to cover Σ^ω, we do not need to consider all U_i and $V_{i,j}$ pairs, i.e., the Property (1) as aforementioned. To that end, we first prove that the concatenation of an equivalence class U_i of $\mathcal{P}$ and an equivalence class $V_{i,j}$ of $\mathcal{P}_i$ is also an equivalence class of $\mathcal{P}$.

Lemma 18. *For all* $U_i \in \mathcal{P}$ *and* $V_{i,j} \in \mathcal{P}_i$*, we have* $U_i V_{i,j} \in \mathcal{P}$ *where* $0 \leq i < k$ *and* $0 \leq j < k_i$.

We now show that we can focus on a subset of pairs of equivalence classes $U_i \in \mathcal{P}$ and $V_{i,j} \in \mathcal{P}_i$ to cover the universe Σ^ω, i.e., Property (1).

Lemma 19 (Coverage of Σ^ω [24]). $\Sigma^\omega = \bigcup_{0 \leq i < k, 0 \leq j < k_i} \{ U_i V_{i,j}^\omega \mid U_i V_{i,j} = U_i \}$.

The proof idea of Lemma 19 is similar to that of Lemma 9. That is, one can just prove that the UP-words of two sets are equivalent. We refer interested readers to [24] for the proof details.

We now need to show that that the partition $(\mathcal{P}, \{\mathcal{P}_i\})$ saturates the ω-regular language $\mathcal{L}(\mathcal{A})$, i.e., Property (2). The proof of Lemma 20 is similar to that of Lemma 14, except that we consider the reachability relation over equivalence classes in reduced run DAGs $\mathcal{G}'_w$ rather than over states.

Lemma 20 (Saturation of $\mathcal{L}(\mathcal{A})$ [24]). *For $U_i \in \mathcal{P}, V_{i,j} \in \mathcal{P}_i$, if $U_i V_{i,j} = U_i$, $U_i V_{i,j}^\omega \cap \mathcal{L}(\mathcal{A}) \neq \emptyset$ implies $U_i V_{i,j}^\omega \subseteq \mathcal{L}(\mathcal{A})$.*

Finally, it follows that we can construct the complementary language of $\mathcal{L}(\mathcal{A})$.

Theorem 9 (Complementary language of $\mathcal{L}(\mathcal{A})$ [24]).
$\Sigma^\omega \setminus \mathcal{L}(\mathcal{A}) = \bigcup_{0 \leq i < k, 0 \leq j < k_i} \{ U_i V_{i,j}^\omega \mid U_i V_{i,j} = U_i, U_i V_{i,j}^\omega \cap \mathcal{L}(\mathcal{A}) = \emptyset \}$.

Breuers *et al.* [5] used a subset construction to define an RC for processing the finite prefix u of a UP-word uv^ω in $\Sigma^\omega \setminus \mathcal{L}(\mathcal{A})$; however, they still used the classical congruence relation $\backsim$ in Definition 8 for recognizing the periodic word v of uv^ω. The congruence relation for processing v in [5] has also been optimized with a preorder, leading to the same upper bound $2^{\mathcal{O}(n \log n)}$ as our work. As pointed out in [24], the complementation construction in [5] uses more than one congruences for recognizing v for a given u; instead, we need only one RC here since the equivalence class $[u]_\approx$ of $\approx$ only relates with one RC $\approx_u$.

In Theorem 7, we are able to build a complementary NBW for $\mathcal{A}$ with the congruence relation $\backsim$, which in fact only requires $\backsim$ being an RC. Similarly, we can now construct a complementary NBW $\mathcal{A}^c$ that accepts $\Sigma^\omega \setminus \mathcal{L}(\mathcal{A})$, based on $\approx$ and $\approx_u, u \in \Sigma^*$. Since the index of $\approx_u$ is in $2^{\mathcal{O}(n \log n)}$, the number of states in $\mathcal{A}^c$ is also in $2^{\mathcal{O}(n \log n)}$.

Theorem 10 ([24]). *For an NBW $\mathcal{A}$ with n states, we have that $\mathcal{L}(\mathcal{A}^c) = \overline{\mathcal{L}(\mathcal{A})}$ and $\mathcal{A}^c$ has at most $2^{\mathcal{O}(n \log n)}$ states.*

We remark that the RCs $\backsim$ and $\approx_u, u \in \Sigma^*$ allow us to construct a family of DFWs [4] that accept either $\mathcal{L}(\mathcal{A})$ or $\Sigma^\omega \setminus \mathcal{L}(\mathcal{A})$ [24]; we refer to [24] for the detailed construction.

6 Concluding Remarks

Over the past few decades, several different approaches have been proposed for complementing NBWs: congruence-based (alternatively Ramsey-based), rank-based and slice-based (alternatively profile-based) constructions. In this work we show that the profile-based analysis is the one tool underlying all of them.

Profiles have been used in [11] for the determinization-based complementation construction. As future work, we will look into the problem of whether profile-based analysis can also be used to explain determinization-based complementation constructions, such as Safra's [32] and Muller-Schupp's [29], possibly inspired by the unified approaches presented in [11,25].

Acknowledgements. We thank the anonymous reviewers for their valuable suggestions to this paper. This work is supported in part by the National Natural Science Foundation of China (Grant Nos. 62102407 and 61836005), NSF grants IIS-1527668, CCF-1704883, IIS-1830549, CNS-2016656, DoD MURI grant N00014-20-1-2787, and an award from the Maryland Procurement Office.

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 T. Touili, B. Cook, and P. B. Jackson, editors, *CAV*, 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 J.-P. Katoen and B. König, editors, *CONCUR*, volume 6901 of *LNCS*, pages 187–202. Springer, 2011.
3. J. D. Allred and U. Ultes-Nitsche. A simple and optimal complementation algorithm for Büchi automata. In A. Dawar and E. Grädel, editors, *LICS*, pages 46–55. ACM, 2018.
4. D. Angluin and D. Fisman. Learning regular omega languages. *Theoretical Computer Science*, 650:57–72, 2016.
5. S. Breuers, C. Löding, and J. Olschewski. Improved Ramsey-based Büchi complementation. In *FOSSACS*, pages 150–164, 2012.
6. J. R. Büchi. On a decision method in restricted second order arithmetic. In *Proc. Int. Congress on Logic, Method, and Philosophy of Science. 1960*, pages 1–12. Stanford University Press, 1962.
7. H. Calbrix, M. Nivat, and A. Podelski. Ultimately periodic words of rational ω-languages. In *MFPS*, pages 554–566. Springer, 1993.
8. Y.-F. Chen, V. Havlena, and O. Lengál. Simulations in rank-based Büchi automata complementation. In A. W. Lin, editor, *APLAS*, volume 11893 of *Lecture Notes in Computer Science*, pages 447–467. Springer, 2019.
9. E. A. Emerson and C.-L. Lei. Temporal reasoning under generalized fairness constraints. In B. Monien and G. Vidal-Naquet, editors, *STACS*, volume 210 of *Lecture Notes in Computer Science*, pages 21–36. Springer, 1986.
10. K. Fisler, R. Fraer, G. Kamhi, M. Y. Vardi, and Z. Yang. Is there a best symbolic cycle-detection algorithm? In T. Margaria and W. Yi, editors, *TACAS*, volume 2031 of *Lecture Notes in Computer Science*, pages 420–434. Springer, 2001.
11. S. Fogarty, O. Kupferman, M. Y. Vardi, and T. Wilke. Profile trees for Büchi word automata, with application to determinization. *Inf. Comput.*, 245:136–151, 2015.
12. S. Fogarty, O. Kupferman, T. Wilke, and M. Y. Vardi. Unifying Büchi complementation constructions. *Log. Methods Comput. Sci.*, 9(1), 2013.
13. S. Fogarty and M. Y. Vardi. Efficient Büchi universality checking. In J. Esparza and R. Majumdar, editors, *TACAS*, volume 6015 of *Lecture Notes in Computer Science*, pages 205–220. Springer, 2010.
14. S. Fogarty and M. Y. Vardi. Büchi complementation and size-change termination. *Logical Methods in Computer Science*, 8(1), 2012.
15. E. Friedgut, O. Kupferman, and M. Y. Vardi. Büchi complementation made tighter. *Int. J. Found. Comput. Sci.*, 17(4):851–868, 2006.

16. S. Gurumurthy, O. Kupferman, F. Somenzi, and M. Y. Vardi. On complementing nondeterministic Büchi automata. In D. Geist and E. Tronci, editors, *CHARME*, volume 2860 of *Lecture Notes in Computer Science*, pages 96–110. Springer, 2003.
17. V. Havlena and O. Lengál. Reducing (to) the ranks: Efficient rank-based Büchi automata complementation (technical report). *CoRR*, abs/2010.07834, 2020.
18. J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation, Second Edition.* Addison-Wesley, 2000.
19. D. Kähler and T. Wilke. Complementation, disambiguation, and determinization of Büchi automata unified. In *ICALP*, pages 724–735. Springer, 2008.
20. H. Karmarkar and S. Chakraborty. On minimal odd rankings for Büchi complementation. In Z. Liu and A. P. Ravn, editors, *ATVA*, volume 5799 of *Lecture Notes in Computer Science*, pages 228–243. Springer, 2009.
21. N. Klarlund. Progress measures for complementation of omega-automata with applications to temporal logic. In *FOCS*, pages 358–367. IEEE Computer Society, 1991.
22. O. Kupferman and M. Y. Vardi. Model checking of safety properties. *Formal Methods in System Design*, 19(3):291–314, 2001.
23. O. Kupferman and M. Y. Vardi. Weak alternating automata are not that weak. *ACM Transactions on Computational Logic*, 2(3):408–429, 2001.
24. Y. Li, Y. Tsay, A. Turrini, M. Y. Vardi, and L. Zhang. Congruence relations for büchi automata. In M. Huisman, C. S. Pasareanu, and N. Zhan, editors, *FM*, volume 13047 of *LNCS*, pages 465–482. Springer, 2021.
25. C. Löding and A. Pirogov. Determinization of Büchi automata: Unifying the approaches of Safra and Muller-Schupp. In C. Baier, I. Chatzigiannakis, P. Flocchini, and S. Leonardi, editors, *ICALP*, volume 132 of *LIPIcs*, pages 120:1–120:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.
26. O. Maler and L. Staiger. On syntactic congruences for omega-languages. *Theor. Comput. Sci.*, 183(1):93–112, 1997.
27. M. Michel. Complementation is more difficult with automata on infinite words. Technical report, CNET, Paris (Manuscript), 1988.
28. S. Miyano and T. Hayashi. Alternating finite automata on omega-words. *Theor. Comput. Sci.*, 32:321–330, 1984.
29. D. E. Muller and P. E. Schupp. Simulating Alternating Tree Automata by Nondeterministic Automata: New Results and New Proofs of the Theorems of Rabin, McNaughton and Safra. *Theor. Comput. Sci.*, 141(1&2):69–107, 1995.
30. J. Myhill. Finite automata and the representation of events. In *Technical Report WADD TR-57-624*, page 112–137, 1957.
31. A. Nerode. Linear automaton transformations. In *American Mathematical Society*, page 541–544, 1958.
32. S. Safra. On the complexity of ω-automata. In *FOCS*, pages 319–327. IEEE, 1988.
33. S. Schewe. Büchi complementation made tight. In *STACS*, volume 3 of *LIPIcs*, pages 661–672. Schloss Dagstuhl, Germany, 2009.
34. A. P. Sistla, M. Y. Vardi, and P. Wolper. The complementation problem for Büchi automata with applications to temporal logic. *Theoretical Computer Science*, 49(2-3):217–237, 1987.
35. W. Thomas. Automata on infinite objects. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 133–191. Elsevier and MIT Press, 1990.
36. M.-H. Tsai, S. Fogarty, M. Y. Vardi, and Y.-K. Tsay. State of Büchi complementation. *Log. Methods Comput. Sci.*, 10(4), 2014.

37. M. Y. Vardi. Expected properties of set partitions. *Technical report, The Weizmann Institute of Science*, 1980.
38. M. Y. Vardi. Verification of concurrent programs: The automata-theoretic framework. *Ann. Pure Appl. Log.*, 51(1-2):79–98, 1991.
39. M. Y. Vardi. The Büchi complementation saga. In W. Thomas and P. Weil, editors, *STACS*, volume 4393 of *Lecture Notes in Computer Science*, pages 12–22. Springer, 2007.
40. M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *LICS*, pages 332–344. IEEE, 1986.
41. Q. Yan. Lower bounds for complementation of ω-automata via the full automata technique. *Logical Methods in Computer Science*, 4(1:5), 2008.